University of Bamberg
Prof. Otto K. Ferstl
Prof. Elmar J. Sinz

Sven Eckert, Stephan Mantel,
Thomas Reeg, Martin Schissler,
Otto K. Ferstl, Elmar J. Sinz

Inter-company integration of application systems – a survey of development methodologies

Feldkirchenstraße 21, D-96045 Bamberg,
Tel. +49 951 / 863 - 25 12, Fax +49 951 / 9 37 04 12
mantel@forwin.de, http://www.forwin.de

FORWIN-Bericht-Nr.: FWN-2004-002

Abstract

When designing business-to-business processes, integrating application systems across company borders is a crucial success factor. This specific kind of integration requires an architecture for coupling the involved application systems. Nowadays, numerous platforms are available to serve as foundation for the development of coupling architectures. The high complexity of typical couplings requires approaches that go beyond the scope of architecture and consider modelling and procedural aspects as well. They represent comprehensive development methodologies. This article presents a framework, which allows to summarise the basic aspects of such approaches. It describes four approaches with respect to that framework.

Keywords

application system, integration, coupling system, development methodologies, modelling, process model

Zusammenfassung

Die überbetriebliche Integration von Anwendungssystemen ist ein wichtiger Erfolgsfaktor zur Gestaltung überbetrieblicher Geschäftsprozesse. Eine solche Integration erfordert eine entsprechende Kopplungsarchitektur der beteiligten Anwendungssysteme. Als Basis für die Entwicklung dieser Kopplungen sind mittlerweile zahlreiche Plattformen verfügbar. Darüber hinaus werden aufgrund der hohen Komplexität typischer Kopplungen umfassende Ansätze benötigt, die auch die Aspekte Modellierung und Vorgehen im Rahmen einer Entwicklungsmethodik berücksichtigen. Der vorliegende Artikel stellt ein Raster zur Erfassung der wesentlichen Aspekte solcher Ansätze vor. Anschließend werden vier ausgewählte Ansätze unter Verwendung dieses Rasters beschrieben und gegenübergestellt.

Schlüsselworte

Anwendungssystem, Integration, Kopplungssystem, Entwicklungsmethodik, Modellierung, Vorgehensmodell

Inhalt

1 Introduction

Business-to-business processes incorporate tasks from several companies, pursuing joint goals. These processes are common practice in supply chain management or virtual companies. Successful design of business-to-business processes requires that the integration of the application systems involved is founded on a coupling architecture. There is a wide range of products available on the market for realising this kind of couplings, such as Microsoft® BizTalk™ or IBM® WebSphere® MQ Integrator®. The high complexity of typical couplings requires comprehensive approaches, which also take into account modelling techniques as well as procedural aspects of the development of application system coupling. This article presents a survey of four development methodologies for inter-company coupling of application systems. To that end, a methodology developed in the OASYS project at the University of Bamberg is compared to other state-of-the-art approaches. The next chapter of the article describes the design of business-to-business processes and how coupled application systems support these processes. Chapter 3 gives an overview of the four approaches under consideration. In order to describe and categorise these approaches, a unified framework is introduced. Finally, chapter 4 summarises the final results.

2 Supporting business-to-business processes through coupled application systems

Business processes within a single company are oriented towards the company's goals. At the company's boundaries, they interact with processes of neighbouring companies. Business concepts like supply chain management and virtual companies aim at combining interacting business processes to form a single business-to-business process following joint goals. To tap the full potential of business-to-business processes, they should be automated by means of application systems, which, as a consequence, need to be coupled among the companies involved. This enables e.g. cutting order cycling times, reducing warehouse stock, and more flexible order processing [Chri98, pp. 31 et seqq.; Pico^{+}01, pp. 9 et seqq.].

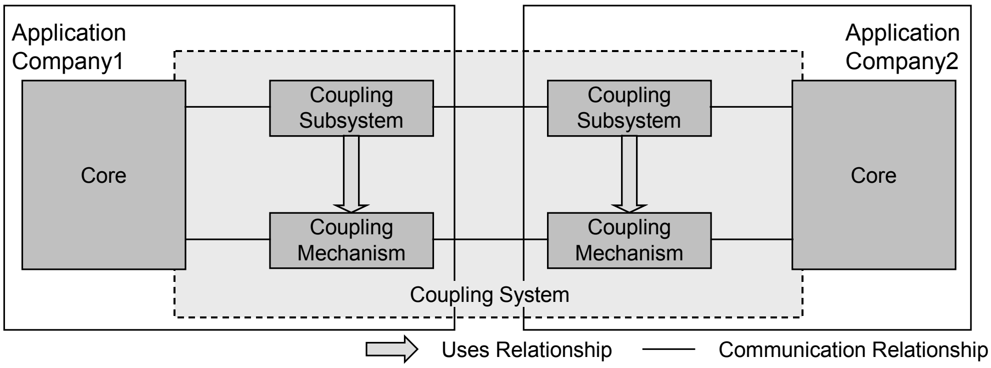

Figure 1: Coupling system (CS)

The inter-company integration of application systems is accomplished using a coupling system containing all elements relevant for the coupling of application systems. For all application systems, the system core can be distinguished from the coupling subsystem. The coupling subsystem contains all elements of the application system that are needed for the coupling with other application systems. It employs coupling mechanisms providing basic services and communication protocols. The core of the application system comprises all elements not involved in coupling including the interfaces to the coupling subsystem. The coupling system of coupled application systems contains the coupling subsystems and the interfaces of the cores. Its architecture is specified by means of a specific coupling architecture.

3 Development methodologies

The following section presents four approaches for the coupling of application systems. The key element of each of these approaches is their particular development methodology. Four approaches were chosen to give a representative overview of current development methodologies for inter-company coupling of application systems. Two approaches result from cooperation of academia and industry (MOVE, Juric et al.), one approach originates from a standardisation group (ebXML) and one is the outcome of a research project (OASYS).

In order to enable the approaches to be presented in a structured manner, the following framework is introduced to serve as a guideline. It considers the design problem, development methodology, availability of reusable model elements and tool support, when using the methodology (Figure 2).

The *design problem* comprises the design subject and the design objective of the methodology. The design subject is distinguished by whether the methodology covers only inter-company coupling on a technical level or, additionally, deals with business-to-business processes. In the same way, the *design objective* either describes the design of coupling systems (CS) or additionally the design of the business-to-business processes.

A *development methodology* is characterised in terms of supported model levels, the process model and consideration of special requirements for the coupling system.

A methodology consists of three *model levels*: (1) the *level of the application domain*, containing the functional requirements of the universe of discourse, (2) the *level of software design*, specifying the software components for the coupling architecture and their relationships and (3) the *implementation level*, realising the specifications within a coupling system.

Concerning the *process model*, the framework distinguishes whether an approach supports an iterative or incremental process. An *iterative* approach uses a step-by-step procedure based on a decomposition of the entire design task. Multiple cycles of the steps are possible, but not mandatory. In contrast, an *incremental* approach divides the object of the design task into individual sub-objects. These sub-objects are either treated independently of each other, or processed in a concerted way.

Particular *requirements* , which need to be considered during coupling system design, such as scalability or adaptability, constitute the methodology's third element. These requirements can be specified as an unstructured list or as a systematic catalogue of requirements.

It will be further analysed, if the considered approaches support a *reuse of model elements* at individual model levels. The framework distinguishes, if such elements exist or if the approach provides additional support for element selection.

			MOVE	ebXML	Juric	OASYS
Design problem	Design subject	B2B processes	+	+	-	+
		Inter-company CS	+	+	o	+
	Design objective	Design B2B processes	+	+	-	+
		Design CS	+	+	o	+
Methodology	Model levels	Application domain	+	+	+	+
		Software design	-	+	+	+
		Implementation	+	+	+	+
	Process model	Iterative	+	o	+	+
		Incremental	o	-	+	+
	Requirements		-	o	o	+
Availability of reusable elements		Application domain	+	+	-	o
		Software design	-	-	o	+
		Implementation	+	-	o	+
Tool support		Application domain	+	+	+	+
		Software design	-	o	+	+
		Implementation	+	o	+	+
		Process model	-	-	-	o

\+ Applies fully o Applies partly - Doesn‘t apply

Figure 2: Classification of the four approaches into the description framework

Finally the framework considers *tool support* of individual model levels and for the process model.

To illustrate the four approaches, figures in the respective chapters show the assignment of process model steps to individual model levels. A step of the process model is associated with a specific level if it generates models on that level. The edges represent sequence relationships between steps.

3.1 MOVE

The MOVE project (Modellierung einer verteilten Architektur für die Entwicklung unternehmensübergreifender Informationssysteme und ihre Validierung im Handelsbereich) deals with modelling of distributed architectures of inter-company information systems. It is a joint project of BFK GmbH, a consultancy and software vendor, ICL Retail Systems GmbH, a manufacturer of trading information systems, the European Trading Institute EHI and the Department of Business Information Systems 1 at the University of Paderborn, Germany [Fisc+98].

3.1.1 Design problem

The project's design subjects are business-to-business processes in the context of supply chain management and the support that inter-organisational systems (IOS) provide for these processes. An IOS can be considered as an application system that is able to operate autonomously with regard to intra-company application systems [Fisc+99, pp. 17 et seq.] and acts as a coupling system. The project pursues two design objectives. On the one hand, it aims at analysis and design of business-to-business processes with special emphasis on inter-company information flows. On the other hand, it addresses the development of IOS based on an architecture, which was investigated in the MOVE project and which is capable to exchange business documents [Fisc+98; Fisc+99, p. 24]. An adaptation of intra-company application systems is not intended [Fisc+99, p. 17].

3.1.2 Methodology

The MOVE approach does not distinguish different model levels explicitly [Fisc+99, pp. 10 et seq.]. However, the application domain level and the implementation level can be identified.

At the *level of the application domain*, a value chain is modelled on the basis of an object-oriented meta model. The meta model consists of the four design elements client, interaction, object and channel. *Clients*, e.g. enterprises or public authorities, exchange services, payments and information in the form of *interactions*. Interactions are specified by the *object*, as the subject of the interaction, and the *channel*, which is necessary for its implementation. Each design element can be assigned with attributes, methods and roles [Fisc+99, pp. 11 et seqq., p. 159]. Every model element must be derived from a repository that has been accepted by all participating companies [Fisc+99, p. 162].

The MOVE approach distinguishes three views at the level of the application domain. (1) The *high-level business process model* specifies all clients of a particular value chain and their interactions [Fisc+99, pp. 169 et seq.]. (2) *Interaction models* describe in detail the interactions of two interrelated clients. For every interaction, the transferred object and the used channel are indicated [Fisc+99, p. 171]. (3) *Structural models of information objects* describe the attributes and methods of an object transferred during an information interaction [Fisc+99, pp. 171 et seqq.].

At the *implementation level*, an IOS is implemented on the basis of a Java™ framework. Alternatively, implementation may be based on component technologies like DCOM or CORBA®, but this alternative is not worked out in detail [Fisc+99, p. 16]. Therefore, it is not addressed in the following.

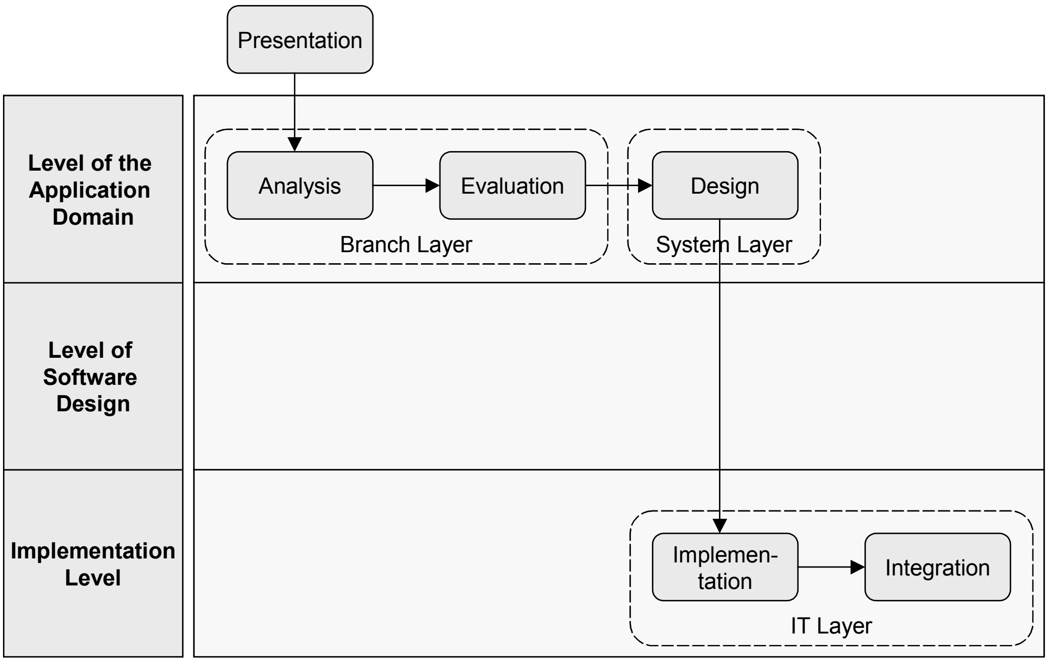

Figure 3: Model levels and phases of the MOVE approach

From the developer's point of view there is a direct transition from the level of the application domain to the implementation level. The level of software design is predetermined by the design of the MOVE Java™ framework. The developer does not create a software design of an IOS. He rather generates software artefacts in the form of Java™ classes directly from models of the application domain and integrates them into the framework [Fisc+99, p. 184].

The MOVE process model for the development of an IOS is organised into the following steps, which are called phases (Figure 3):

- During the *"Presentation"* phase, interested companies become familiar with the use of the MOVE approach [Fisc+99, pp. 228 et seq.].
- The "Analysis" phase covers the analysis of the business-to-business process and the identification of potential profits of a MOVE implementation. To that end, the effects on costs and business volume of such an implementation are qualified using a cost utility analysis [Fisc+99, pp. 93 et seqq., p. 229].
- By simulating different business process alternatives, the *"Evaluation"* phase tries to determine an appropriate design for the business-to-business process [Fisc+99, pp. 121 et seqq., p. 229].
- During the *"Design"* phase, the developer creates a model of the selected business-to-business process alternative. This phase's outcome is a detailed description of information objects and their relationships [Fisc+99, p. 153, p. 229].
- *"Implementation"* comprehends the generation of Java™ classes from design elements described in the design phase and their integration into the Java™ framework [Fisc+99, p. 183, p. 230].
- During the *"Integration"* phase, the implemented classes are embedded into the IT infrastructure of the company [Fisc+99, p. 230].

The MOVE approach is structured by three layers: Branch layer, System layer and IT layer. These layers can be considered as high-level phases for the development of an IOS. The Branch layer comprises the phases Analysis and Evaluation, the System layer the Design phase and the IT layer the phases Implementation and Integration.

The MOVE process model permits to return to previous phases and can therefore be characterised as cyclic and iterative [Fisc+99, p. 19]. Additionally, it allows an incremental process, where subprojects are identified and dealt with independently, called "Inkblot approach". Therefore, it may be necessary to adapt these increments during integration. It is not specified how this should be achieved [Fisc+99, p. 232].

The MOVE approach does not contain the acquisition of special requirements for the design of an IOS, because the Java™ framework predetermines the design and the implementation of an IOS. Thus, in a concrete integration project, it is not possible to consider specific requirements.

3.1.3 Availability of reusable model elements

MOVE provides a repository at the level of the application domain. This repository contains reference classes for the design elements client, object, interaction and channel in the form of class diagrams. Furthermore it comprises a catalogue of roles for clients and objects as well as information elements that are used to compose the attributes of information objects [Fisc+99, p. 162]. The Java™ framework offers reuse at implementation level.

3.1.4 Tool support

A web-based presentation tool is available for the presentation of the project [Fisc+99, pp. 228 et seq.]. The MOVE-NWA tool supports the cost utility analysis, MOVE-SIM the simulation of business process alternatives [Fisc+99, pp. 19 et seq.]. An object-oriented design tool is used to create the high-level business process model, the interaction models and the structural models of information objects [Fisc+99, pp. 176 et seq.]. An administration module supports maintenance of the reusable model elements at the level of the application domain [Fisc+99, p. 177]. A software generator creating classes for the Java™ framework facilitates the transition between the application domain level and the implementation level [Fisc+99, pp. 188 et seqq.].

3.2 ebXML

The term *Electronic Business Extensible Markup Language* (ebXML) denotes an international initiative, which is led by the „United Nations Centre for Trade Facilitation and Electronic Business” (UN/CEFACT) and the “Organization for the Advancement of Structured Information Standards” (OASIS). The main objective is the realisation of an open, XML-based framework to create a standardised and consistent basis for the secure exchange of information in the context of inter-company integration. ebXML is organised modularly in specifications. To create an ebXML solution, a developer has to implement these specifications. The ebXML initiative claims to establish an open, international standard to realise a „Single Global Electronic Market“ [ebXM01a, p. 7].

3.2.1 Design problem

The design subjects of ebXML are business-to-business processes and coupling systems. Design objectives are analysis and design of these business processes as well as the development of a coupling system according to a coupling architecture [Busi01, pp. 5 et seq.]. An extended SOAP protocol is used as coupling mechanism. Customising existing application systems is not in the scope of the ebXML framework.

3.2.2 Methodology

The ebXML framework does not distinguish model levels explicitly. However, it is possible to identify a classification of model levels according to the description framework by using the Framework Specification Schema (Figure 4).

Precise scenarios, based on the ebXML Business Process Specification Schema (ebBPSS) as a meta model, are described at the *application domain level*. Thereby business processes are specified in terms of business-to-business interactions. These so-called ebXML Business Process Specifications (ebBPS) give an in-detail description of roles and activities assigned to participating trading partners. Interaction takes place in form of transactions, which are represented by the exchange of Business Documents. ebXML recommends the use of the UN/CEFACT Modeling Methodology (UMM), which uses the Unified Modeling Language (UML™) for modelling business processes. The ebBPSS forms a subset of the UMM Meta Model, which permits an automated transformation of UML™-modelled business processes to XML-based ebBPSS. UMM supports the use of design patterns, corresponding Production Rules for XML representations are part of the ebBPSS. The definition of Business Documents is mainly based on reusable model elements, the ebXML Core Components (ebCC). Analogically to ebBPSS, the ebCC Meta Model can be interpreted as a subset of the UMM Meta Model. Production Rules are provided in this case as well [Busi01, pp. 4 et seqq.].

The Collaboration Protocol Profile and Agreement Specification defines a set of rules for the application domain level and the *software design level*. This rule set is used for generating the Collaboration Protocol Profiles (CPP) and the Collaboration Profile Agreements (CPA). A CPP represents a company-specific profile, created according to an ebBPS. During the creation process, companies are assigned to relevant roles and activities. In addition, functional requirements, specified during the modelling of the business process, are enhanced with technical parameters, e.g. encryption requirements. Hence, a CPP represents a company's concrete functional and technical requirements with regard to an ebBPS in XML format. The CPA is a machine interpretable run-time specification of an ebBPS, which describes all the functional and technical requirements that are obligatory for collaboration between firms. It is derived from matching the CPP of two companies [Busi01, pp. 9 et seq.].

The ebXML Message Service (ebMS), described by the Message Service Specification, is located at the *implementation level*. It includes technical aspects of ebXML-Messaging, describes the structure of exchanged messages and defines services like routing or packaging. The implementation of the ebMS realises a coupling subsystem, which includes a Message Service Interface (MSI) for the interaction with existing applications and a Message Service Handler (MSH) to encapsulate the main functionality of the ebMS [OASI02, pp. 8 et seqq.].

An explicit process model for implementing the portrayed specifications is not defined within the ebXML framework. However, single steps can be identified based upon dependencies between the specifications [ebXM01b, p. 8]:

- "*Build ebBPS*" models domain-specific business processes by means of the ebBPSS. Main objective is an ebXML-compliant representation of the ebBPSS in the form of a machine interpretable XML-document.

- The "*Build CPP*" phase creates company-specific, XML-formatted profiles, based upon an underlying ebBPS.
- "*Build CPA*" covers the generation of a CPA from the CPP of two companies. The CPA specifies parameters, which are required for an automated configuration of coupling subsystems.
- "*Implement ebMS*" refers to the configuration of the MSH and the MSI by the CPA-supplied parameters. Through this, an ebBPS specific realisation of coupling subsystems is accomplished.

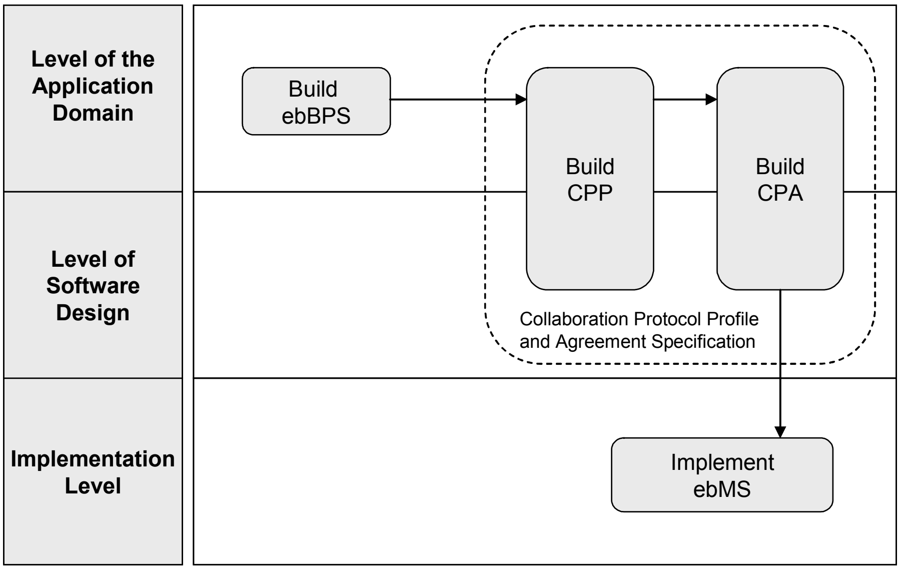

Figure 4: Model levels and steps of the ebXML framework

The definition of requirements does not lead to a separate phase within the ebXML framework. However, functional requirements to specific ebBPS can be considered during modelling time. Company specific, technical requirements enter the development process during the CPP creation. A resulting CPA finally contains the requirements for the realisation of inter-company integration.

The realisation of an ebXML solution is an iterative process. Returns to previous phases are not described explicitly within the specifications. It must be pointed out though that the sequence of steps described above is not part of an ebXML-specification. Differing process models are also feasible as a consequence of the framework's modular structure and the self-reliance of the specifications.

3.2.3 Availability of reusable model elements

Reusable model elements are available at the application domain level in the form of ebCC. These elements comprehend data and data types, which can be utilised to describe Business

Documents [ebXM01b, pp. 23f). Furthermore, predefined ebBPS can be reused both in a domain-specific and a domain-independent way.

3.2.4 Tool support

Concrete tool implementations are not provided in the ebXML framework. However, the ebXML Registry Service (ebRS) can be considered as a tool to support the reuse of model elements, even though specified as part of the framework. Beside ebCC, ebRS manages and deploys ebBPS, CPP or CPA. Further starting-points for the development of tools lie particularly in the fields of automated generation of CPP from ebBPS or CPA from CPP. Additional use of third-party tools is basically feasible within the scope of the UML™ modelling at the application domain level. Furthermore, concrete implementations of individual specifications can be found as components of integration products, which provide a basis to create an ebXML solution.

3.3 Juric et al.

In [Juri+01], Juric et al. describe how J2EE™ can be used to couple application systems. The main focus of the publication is on the implementation level.

3.3.1 Design problem

The publication mainly deals with the integration of the application systems of a single company. The main objective is the development of a *composite information system* [Juri+01, pp. 33 et seq., p. 75], which is a new application system that uses existing systems and integrates them. The existing application systems are wrapped up by so-called virtual components [Juri+01, pp. 77 et seqq.]. An adaptation of existing application systems is not intended. The approach does not aim at the design of business processes.

The main topic of the approach is intra-company integration of application systems, but inter-company integration is also supported. Major parts of the coupling architecture needed for inter-company integration are covered by the composite information system. Additionally, the authors describe how the composite information system can be adapted to realise the intended integration [Juri+01, pp. 102 et seqq., pp. 851 et seqq.].

Juric et al. distinguish four layers of integration. (1) *Data level integration* supports the shared use of data by several application systems directly accessing the databases [Juri+01, pp. 80 et seqq.]. (2) *Application interface level integration* deals with the common use of functions and data by several application systems. This is achieved by application programming interfaces (API) of the application systems [Juri+01, pp. 85 et seqq.]. (3) The same goal is pursued by *business method level integration* [Juri+01, pp. 91 et seqq.]. In contrast to application interface level integration, where the virtual components provide merely a technical abstraction of the application systems, a domain-related abstraction is achieved in this case. The virtual components are defined according to the requirements of the composite information system. (4) *Presentation level integration* offers the users an integrated interface to the functions and the data of several application systems [Juri+01, pp. 97 et seqq.].

3.3.2 Methodology

The approach does not distinguish model levels explicitly. However, the employed models can be assigned to the model levels introduced in Figure 2. The approach mainly uses UML™ diagrams to represent the models [OMG01].

At the *application domain level*, UML™ use case diagrams play a central role [Juri+01, pp. 167 et seqq.]. Domain-related class diagrams and sequence diagrams are used in addition for the business method level integration [Juri+01, pp. 402 et seqq.].

At the *level of software design*, two kinds of models are created: models capturing the relevant properties of the existing application systems and models, which are used to describe how the stated requirements can be realised by a coupling system. Component diagrams, deployment diagrams and sequence diagrams offered by UML™ are mainly used for those tasks [Juri+01, pp. 415 et seqq.]. Components serve as representation for existing application systems and describe the virtual components that have to be developed. In the context of data level integration several methods for data modelling are being proposed depending on the database model [Juri+01, pp. 215 et seqq.], e.g. Entity-Relationship modelling.

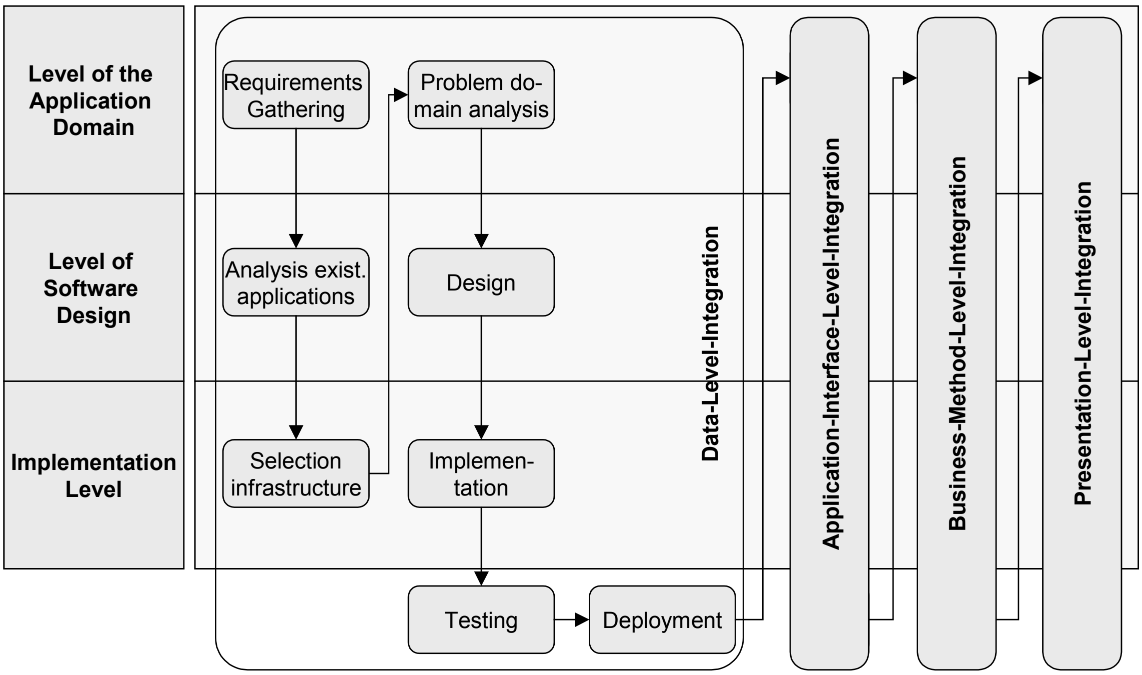

Figure 5: Model levels, phases and activities of the approach of Juric et al.

Due to the fact that the approach concentrates on J2EE™, the *implementation level* uses mainly the Java™ programming language elements and mechanisms offered by J2EE™, e.g. Java Database Connectivity (JDBC™), Java Message Service (JMS), Java Connector Architecture (JCA).

As shown in Figure 5, the process for the development of a composite information system consists of four *integration phases* corresponding to the four layers of integration. The sequence of those phases is chosen in a way that allows building on top of the results of the previous phases.

Within the integration phases several *activities* have to be conducted [Juri+01, pp. 155 et seqq.]:

- *„Requirements gathering"* aims at determination of the requirements for the composite information system.
- The activity *„Analysis of existing applications"* records the relevant properties of the existing application systems.
- The activity *„Selection of integration infrastructure"* deals with the selection of coupling mechanisms.
- The activity *„Problem domain analysis"* creates several domain-related models of the coupling system.
- The corresponding technical models of the coupling system are developed in the *„Design"* activity.
- The process concludes with the activities *„Implementation"*, *„Test"* and *„Deployment"*, in which the coupling system is implemented, tested and installed.

The approach complies with the Rational Unified Process® [Kruc00]. It supports a cyclic, iterative and partially coordinated process [Juri+01, p. 154]. The increments are defined according to the use cases identified during the "Requirements gathering" activity.

Juric et al. consider special requirements for the coupling system in several sections. However, a systematic treatment cannot be found.

3.3.3 Availability of reusable model elements

Major sections of [Juri+01] describe how a coupling architecture can be implemented using J2EE™ code. These examples can be used as basic templates when implementing a coupling architecture.

Juric et al. describe several patterns that can be used in software design. A central role plays the architectural pattern, which describes the structure of a composite information system

[Juri+01, pp. 74 et seqq., p. 96, p. 100]. The composite information system is based on a multi-tier architecture comprising a user interface tier, a business logic tier and a data persistence tier [Juri+01, p. 76]. Existing application systems are included in the data persistence tier. The business logic tier is composed of virtual components that communicate via an integration broker. The user interface tier consists of user interfaces developed during presentation level integration.

A web service tier enhances the architecture outlined above when an inter-company coupling system should be implemented. This tier contains web services that use the virtual components in the business logic tier [Juri+01, pp. 102 et seqq.].

Selection of model elements is partially supported by the description of consequences that result when one of the offered elements is included into a model.

3.3.4 Tool support

Because the approach uses mainly UML™ diagrams at the application domain and software design level, it is possible to choose from a large variety of UML™ tools. Same is true for implementation, since many tools are available for J2EE™ covering the implementation, the code generation und the management of reusable model elements.

3.4 OASYS

The research project „Open Application Systems Architectures in inter-company value chains“ (OASYS), conducted at the University of Bamberg, Germany, is part of the „Bavarian Information Systems Research Network“ (FORWIN). The objective of the project is the development and testing of coupling systems for the integration of heterogeneous application systems belonging to different companies. The main focus is particularly on methodologies and tools to support the integration process.

3.4.1 Design problem

Design subjects of the OASYS approach are business-to-business processes in inter-company value chains and the associated coupling systems. Objectives are the design of business-to-business processes and the development of coupling systems for inter-company integration. In this approach, modification of existing application systems’ cores is necessary only with respect to the interfaces between the cores and the coupling systems.

OASYS differentiates three kinds of coupling architectures, varying in their specific objectives [Mant+02; Schi+02b]. (1) *Event-oriented coupling architectures* transfer events and related data between loosely coupled application systems through message passing [FeSi01,

p. 225]. (2) *Data-oriented coupling architectures* enable the manipulation of shared data by tightly coupled functions operating on these data [FeSi01, p. 225]. (3) *Function-oriented coupling architectures* enable different application systems to share functions and, where required, related data.

3.4.2 Methodology

The OASYS approach is structured into the following four model levels: the business-to-business process level, the application system level, the coupling architecture level and the implementation level [Mant+02, p. 4]. Figure 6 assigns these levels to the levels introduced in the description framework.

The *business-to-business process level* describes processes using the approach of the Semantic Object Model (SOM) [FeSi01, pp. 179 et seqq.]. A business process is modelled as a distributed system consisting of autonomous and loosely coupled business objects. These business objects coordinate themselves using transactions [FeSi01, pp. 181 et seq.]. The SOM approach provides two different views on a business process. The first view is called the interaction schema and specifies the static structure of a business process. It contains objects as well as transactions. The second view is called the task-event schema and presents the dynamic behaviour of a business process. It consists of tasks, events, and transactions in their roles as events. The approach supports domain-related as well as domain-independent patterns [FeSi01, pp. 189 et seqq.; Fers+98]. Categorised task integration patterns (TIP) define partitions of the business process, which have to be integrated separately.

The *application system level* describes the application systems from an outside view. An application system serves as an actor for the automated tasks of a business process.

The *coupling architecture level* describes the coupling of application systems in the form of coupling architectures. It describes the static structure consisting of coupling subsystems and communication relationships between these subsystems. In addition it is intended to represent the dynamic behaviour in the form of a specification of the protocols between the coupling subsystems. Furthermore this level includes a description of relevant properties of the application systems, e.g. interfaces of the cores of the application systems or available coupling mechanisms. Patterns describe alternative coupling architectures on the basis of a standardised schema [Schi+02a, p. 7].

At the *implementation level*, the coupling system is realised based on the developed coupling architecture. Integration tools like Microsoft® BizTalk™ or IBM® Web Sphere® MQ Integrator® can be employed here. Alternatively a self-developed JAVA™ framework will be available.

The process model of the approach comprises eight steps (Figure 6):

- The step *"Definition of functional requirements"* defines functional requirements for the design of the business-to-business process.
- Analysis and design of the business-to-business process using the SOM approach is purpose of the second step called *"Design of the business-to-business process"*. At that stage, the functional requirements from the previous step have to be taken into account.
- The *"Mapping of the application systems"* specifies the degree of automation of tasks and transactions and assigns application systems to the tasks they support.
- *"Identification of TIP"* means defining partitions of the business process that are relevant for the integration in the form of TIP.
- *"Specification of technical requirements on the application system integration with respect to the TIP"* aims at the acquisition of technical requirements for each TIP.
- During the step *"Description of the application systems"* the developer identifies the necessary information about the technical properties of the application systems.
- The *"Design of the coupling architecture"* comprehends the development of a concrete coupling architecture based on the specified requirements and the properties of the application systems.
- The *"Implementation"* of the coupling system is realised by customising existing integration tools respectively the JAVA™ framework. This step is automated as far as possible.

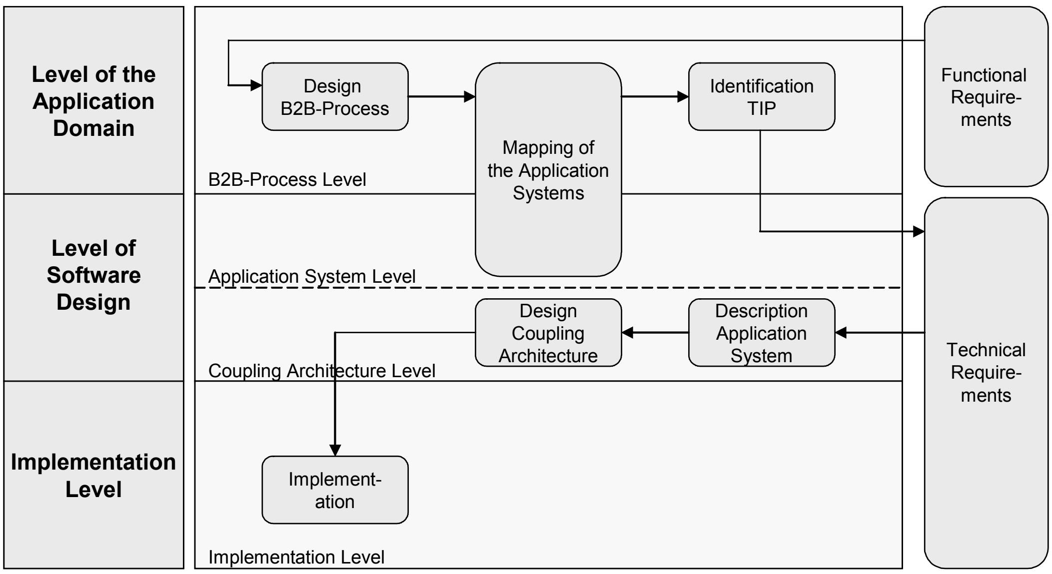

Figure 6: Model levels and steps of the OASYS approach

The OASYS approach uses a cyclic and iterative process. The steps are executed sequentially. Returns to previous steps are allowed. For each TIP an incremental process with coordinated development of the partial coupling systems is intended.

The OASYS approach explicitly supports acquisition and examination of technical requirements for the coupling of application systems. These are acquired by using a structured catalogue of requirements, divided into the four categories correctness, real-time behaviour, integration and flexibility [Fers92, p. 11]. The requirements associated with these categories are partly based on ISO standard 9126-1, e.g. the features availability and security [ISO01].

Further plans include establishing a concept describing the relationships between requirements and different alternatives of coupling architectures. Thus it will be possible to derive design alternatives from specified requirements.

3.4.3 Availability of reusable model elements

At the business-to-business level, the OASYS approach provides reference models e.g. for Vendor Managed Inventory. Additionally, domain-related as well as domain-independent patterns, developed for the SOM approach, can be applied. Examples for such patterns are leasing patterns, the negotiation principle and the feedback control principle [Fers+98; FeSi01, pp. 189 et seqq.]. At the coupling architecture level the approach provides alternative coupling architectures. The selection of alternatives will be supported by the concept mentioned above. At the implementation level, reusability is enabled through the JAVA™ framework respectively elements for customising available integration tools. Current research activities concentrate on reusability enhancements.

3.4.4 Tool support

One objective of the OASYS project is the development of an integrated tool, covering all model levels of the development methodology as well as the process model. The tool's current functionality supports the application domain level and the level of software design. The tool allows the developer to specify requirements and to select alternatives of coupling architectures. Therefore it provides a catalogue of questions, concerning the design of the coupling architecture. The tool immediately visualises the impact of given answers. Tool support will be enhanced in future by including the concept to derive coupling architectures from specified requirements mentioned above and by covering different integration tools.

4 Summary

The article describes four approaches to design inter-company coupling systems using a uniform framework, without evaluating the approaches. MOVE, ebXML and OASYS support the design of business-to-business processes, in addition to the design of coupling systems. All four approaches offer a methodology, reusable model elements and tools. They differ in the emphasis placed on specific model levels of the description framework and in the degree of specialisation at these model levels.

Using the MOVE approach, the developer focuses primarily on the level of the application domain and only secondarily deals with the level of software design and the implementation level. Hence, he needs no particular technical knowledge. This process is due to the restriction of the coupling system to the exchange of business documents. MOVE supports implementation through automated generation of Java™ classes. These classes are integrated into the given framework.

ebXML is also characterised by the specialisation in the exchange of business documents. In contrast to MOVE, ebXML supports a more comprehensive adaptation of the coupling system to the concrete technical requirements, such as scalability. To specify these adaptations, a developer operates not only at the level of the application domain, but also at the level of software design. At the implementation level, ebXML aims for an automated configuration of the coupling subsystems according to the Message Service Specification.

The approach by Juric et al. focuses on intra-company integration, but makes also a substantial contribution to inter-company integration. The approach emphasises all three model levels similarly. There is no specialisation on one of the model levels. Therefore, the covered scope of application exceeds that of MOVE and ebXML. However, the developer needs comprehensive technical knowledge, especially implementation knowledge, to apply this approach.

Like Juric et al., the OASYS approach tries to achieve a broad coverage at all three model levels. The developer concentrates on the level of the application domain and the level of software design, though. Implementation of the coupling architectures should be accomplished through a widely automated adaptation of integration tools respectively a self-designed Java™ framework.

References

[Busi01] Business Process Project Team: ebXML Business Process Specification Schema Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf, 2001, (Accessed: 2003, January 10).

[Chri98] Christopher, M.: Logistics and Supply Chain Management - Strategies for Reducing Cost and Improving Service. 2nd edn, Financial Times Prentice Hall: London, 1998.

[ebXM01a] ebXML Requirements Team: ebXML Requirements Specification Version 1.06. http://www.ebxml.org/specs/ebREQ.pdf, 2001, (Accessed: 2003, January 10).

[ebXM01b] ebXML Technical Architecture Project Team: ebXML Technical Architecture Specification Version 1.0.4. http://www.ebxml.org/specs/ebTA.pdf, 2001, (Accessed: 2003, January 10).

[Fers92] Ferstl, O. K.: Integrationskonzepte Betrieblicher Anwendungssysteme. Research Report, University of Koblenz-Landau, No.1/1992.

[Fers+98] Ferstl O. K.; Sinz E. J.; Hammel Ch.; Schlitt M.; Wolf S.; Popp K.; Kehlenbeck R.; Pfister A.; Kniep H.; Nielsen N.; Seitz A.: WEGA - Wiederverwendbare und erweiterbare Geschäftsprozeß- und Anwendungssystemarchitekturen. Closing Report, Walldorf, 1998.

[FeSi01] Ferstl, O. K.; Sinz, E. J.: Grundlagen der Wirtschaftsinformatik. 4th edn, Oldenbourg: Munich, 2001.

[Fisc+98] Fischer, J.; Hammer, G.; Kern, U.; Rulle, A.; Städler, M.; Steffen, T.: Verbundprojekt MOVE - Modellierung einer verteilten Architektur für die Entwicklung unternehmensübergreifender Informationssysteme und ihre Validierung im Handelsbereich. In: Projektträger Informationstechnik des BMBF beim DLR e.V.: Status Seminar of the BMBF Softwaretechnologie, 23-24 March, Bonn, 1998.

[Fisc+99] Fischer, J.; Atzberger, M.; Städler, M.; Kambergs, P.; Hluchy, R.; Hoos, J.; Pauls, M.; Walter, F.; Steffen, T.; Dresing, H.; Rulle, A.; Brentano, F.: MOVE - Objektorientierte Modelle und Werkzeuge für unternehmensübergreifende Informationssysteme im Rahmen des Electronic Commerce. Closing Report, Paderborn, 1999.

[ISO01] ISO: Software engineering – Product quality – Part 1: Quality model ISO/EIC 9126-1:2001. ISO, 2001.

[Juri+01] Juric, M. B.; Basha, S. J.; Leander, R.; Nagappan, R.: Professional J2EE EAI. Wrox: Birmingham, 2001.

[Kruc00] Kruchten, P.: The Rational Unified Process - An Introduction. 2nd edn, Addison Wesley: Boston, 2000.

[Mant+02] Mantel, S; Eckert, S.; Schissler, M.; Ferstl, O. K.; Sinz, E. J.: Entwicklungsmethodik für überbetriebliche Kopplungsarchitekturen von Anwendungssystemen. In: Bartmann, D. (Ed.): Kopplung von Anwendungssystemen - Forwin-Tagung 2002, Shaker: Aachen, 2002, pp. 183-202.

[OASI02] OASIS ebXML Messaging Services Technical Committee: Message Service Specification Version 2.0. http://www.ebxml.org/specs/ebMS2.pdf, 2002, (Accessed: 2003, January 10).

[OMG01] OMG: OMG Unified Modeling Language Specification Version 1.4. http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf, 2001, (Accessed: 2003, January 28).

[Pico+01] Picot, A.; Reichwald, R.; Wigand, R. T.: Die grenzenlose Unternehmung - Information, Organisation und Management. 4th edn, Gabler: Wiesbaden, 2001.

[Schi+02a] Schissler, M.; Mantel, S.; Ferstl, O. K.; Sinz, E. J.: Architekturen zur überbetrieblichen Kopplung von Anwendungssystemen. Internal Research Report, Bamberg, 2002.

[Schi+02b] Schissler, M.; Mantel, S.; Ferstl, O. K.; Sinz, E. J.: Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3. In: Wirtschaftsinformatik, vol. 44, no. 5, pp. 459-468.

[Sinz02] Sinz, E. J.: Architektur von Informationssystemen. In: Rechenberg, P.; Pomberger, G. (Eds.): Informatik-Handbuch. 3rd edn, Hanser: Munich, 2002, pp. 1055-1068.

Folgende FORWIN-Berichte sind bisher erschienen:

FWN-2000-001
Mertens, P.
FORWIN – Idee und Mission
E-Business * Supply Chain Management * Betriebliche Software-Bausteine

FWN-2000-002
Sinz, E. J.
Die Projekte im Bayerischen Forschungsverbund Wirtschaftsinformatik (FORWIN)

FWN-2000-003
Kaufmann, Th.
Marktplatz für Bausteine heterogener betrieblicher Anwendungssysteme

FWN-2000-004
Schaub, A., Zeier, A.
Eignung von Supply-Chain-Management-Software für unterschiedliche Betriebstypen und Branchen – untersucht am Beispiel des Produktions-Prozessmodells zum System SAP APO

FWN-2000-005
Friedrich, M.
Konzeption eines Componentware-basierten Supply-Chain-Management-Systems für kleine und mittlere Unternehmen

FWN-2000-006
Schmitzer, B.
Klassifikationsaspekte betriebswirtschaftlich orientierter Frameworks

FWN-2000-007
Zeier, A., Hauptmann, S.
Ein Beitrag zu einer Kern-Schalen-Architektur für Supply-Chain-Management (SCM)-Software, Teil I: Anforderungen an den Kern einer SCM-Software und deren Abdeckung in SAP APO 2.0/3.0

FWN-2000-008
Maier, M.
Bestandsaufnahme zu Jobbörsen im WWW

FWN-2000-009

Mantel, S., Knobloch, B.; Rüffer, T., Schissler, M., Schmitz, K., Ferstl, O. K., Sinz, E. J.

Analyse der Integrationspotenziale von Kommunikationsplattformen für verteilte Anwendungssysteme

FWN-2000-010

Franke, Th., Barbian, D.

Platform for Privacy Preferences Project (P3P) - Grundsätze, Struktur und Einsatzmöglichkeiten im Umfeld des "Franken-Mall"-Projekts

FWN-2000-011

Thome, R., Hennig, A., Ollmert, C.

Kategorisierung von eC-Geschäftsprozessen zur Identifikation geeigneter eC-Komponenten für die organisierte Integration

FWN-2001-001

Zeier, A., Hauptmann, S.

Ein Beitrag zu einer Kern-Schalen-Architektur für Supply-Chain-Management (SCM)-Software, Teil II: Anforderungen an die Schalen einer SCM-Software und deren Abdeckung in SAP APO 2.0/3.0

FWN-2001-002

Lohmann, M.

Die Informationsbank ICF – eine wissensbasierte Werkzeugsammlung für die Software-Anforderungsanalyse

FWN-2001-003

Hau, M.

Das DATEV-Komponenten-Repository - Ein Beitrag zu Marktplätzen für betriebswirtschaftliche Software-Bausteine

FWN-2001-004

Schoberth, Th.

Virtual Communities zur Unterstützung von Infomediären

FWN-2001-005

Kronewald, K., Menzel, G., Taumann, W., Maier, M.

Portal für bürgergerechte Dienstleistungen in der Sozialen Sicherheit

FWN-2001-006

Maier, M.

Strukturen und Prozesse im "Netzwerk für Arbeit"

FWN-2001-007

Maier, M., Gollitscher, M.

Überlegungen zum Skill-Matching-Modul eines Leitstands für den regionalen, zwischenbetrieblichen Personalaustausch

FWN-2001-008

Schissler, M.

Unterstützung von Kopplungsarchitekturen durch SAP R/3

FWN-2001-009

Göbel, Ch, Hocke, S.

Simulative Analyse interorganisatorischer Kopplungsdesigns

FWN-2001-010

Thome, R. Schütz, St., Zeißler, G.

Ermittlung betriebswirtschaftlicher Anforderungen zur Definition von Geschäftsprozessprofilen

FWN-2001-011

Mehlau, J.

Ist-Aufnahme von IT-Architekturen bei Finanzdienstleistern

FWN-2001-012

T Horstmann, R., Ottenschläger, S.

Internetstudie: Reisedienstleister

FWN-2001-013

Horstmann, R., Zeller, Th., Lejmi, H.

Anbindung von ERP-Systemen an Elektronische Marktplätze

FWN-2001-014

Robra-Bissantz, S., Weiser, B.

Ein Meta-Framework zur Identifizierung und Beschreibung von Push-Möglichkeiten im E-Commerce

FWN-2002-001
Wiesner, Th. .
Push-Konzepte im E-Commerce: State of the Art

FWN-2002-002
Zeier, A.
Identifikation und Analyse branchenspezifischer Faktoren für den Einsatz von Supply-Chain-Management-Software. Teil I: Grundlagen, Methodik und Kernanforderungen

FWN-2002-003
Zeier, A.
Identifikation und Analyse branchenspezifischer Faktoren für den Einsatz von Supply-Chain-Management-Software. Teil II: Betriebstypologische Branchensegmentierung

FWN-2002-004
Zeier, A.
Identifikation und Analyse branchenspezifischer Faktoren für den Einsatz von Supply-Chain-Management-Software. Teil III: Evaluation der betriebstypologischen Anforderungsprofile auf Basis des SCM-Kern-Schalen-Modells in der Praxis für die Branchen Elektronik, Automobil, Konsumgüter und Chemie/Pharma

FWN-2002-005
Zeier, A.
Identifikation und Analyse branchenspezifischer Faktoren für den Einsatz von Supply-Chain-Management-Software. Teil IV: Anwendungsbeispiel

FWN-2002-006
Weiser, B., Robra-Bissantz, S.
Eine kosten- und nutzenorientierte Typisierung von Push-Ansätzen im E-Commerce

FWN-2002-007
Robra-Bissantz, S., Weiser, B.; Schlenker, C.
Push-Konzepte im CRM von Finanzdienstleistungsbetrieben

FWN-2002-008
Eckert, S., Mehlau, J., Mantel, S., Schissler, M., Zeller, T.
Sichere Kopplung von ERP-Systemen und elektronischen Marktplätzen

FWN-2002-009
Mantel, S., Eckert, S., Schissler, M., Ferstl, O. K., Sinz, E. J.
Entwicklungsmethodik für überbetriebliche Kopplungsarchitekturen von Anwendungssystemen

FWN-2002-010
Mehlau, J. I.
Sicherheitsmuster im Kontext der Anwenderungssystemkopplung

FWN-2002-011
Lejmi, H.; Zeller, A.
Einsatz des Kooperativen Planens in B2B-Abwicklungsplattformen – Konzept und praktisches Beispiel

FWN-2002-012
Mautner, R.; Thome, R..
Einsatz von Produktkonfiguratoren

FWN-2002-013
Voigtmann, P.; Zeller, Th.
Enterprise Application Integration und B2B Integration im Kontext von Electronic Business und Elektronischen Marktplätzen. Teil I: Grundlagen und Anforderungen

FWN-2002-014
Eisenhauer, R.; Robra-Bissantz, S.; Schoberth, Th.; Weiser, B.
Communities zur Unterstützung von Push-Konzepten im E-Commerce

FWN-2003-001
Zeller, Th.
Enterprise Application Integration und B2B Integration im Kontext von Electronic Business und Elektronischen Marktplätzen. Teil II: Integrationssysteme und Fallbespiele

FWN-2003-002
Zeller, A.
Controlling von Unternehmensnetzwerken: Bestandsaufnahme und Lückenanalyse

FWN-2003-003
Robra-Bissantz, S.; Schneider, U.; Weiser, B.
Konzeption von Push-Aktivitäten auf einem elektronischen Marktplatz für Produktionsmaschinen

FWN-2003-004
Lehmann, H.; Lehner, F
Is there a ‘Killer Application’ in Mobile Technology? A Tailored Research Approach

FWN-2003-005

Berger, S.; Lehner, F.

Intra- und interorganisationale Kooperation - Unterstützung der Prozess- und Systemkopplung durch mobile Technologien

FWN-2003-006

Lejmi, H.

Ein Beitrag zur Integration der Tourenplanung auf e-Marktplätzen am Beispiel einer Abwicklungs-Plattform in der Möbelindustrie