TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Ph.D. Thesis

A UML-based Methodology for Model-Driven B2B Integration: From Business Values over Business Processes to Deployment Artifacts

Conducted for the purpose of receiving the academic title
'Doktor der Sozial- und Wirtschaftswissenschaften'

Advisors

Univ.Prof. Dipl.-Ing. Dr.techn. Hannes Werthner

and

Ao.Univ.Prof. Mag. Dr. Christian Huemer

Institute of Software Technology and Interactive Systems (E188)

Vienna University of Technology

Submitted at the

Vienna University of Technology

Faculty of Informatics

by

Marco Zapletal

0007099

Vienna, 12th June 2009 _____

Kurzfassung der Dissertation

Business-to-Business (B2B) Electronic Commerce adressiert den Geschäftsdatenaustausch zwischen Unternehmen. Dabei müssen zwischenbetriebliche Geschäftsprozesse, die sich über Unternehmensgrenzen erstrecken, abgebildet werden. Der Entwurf und die erfolgreiche Implementierung von zwischenbetrieblichen Geschäftsprozessen unterscheiden sich stark von den Anforderungen normaler, sogenannter innerbetrieblicher Geschäftsprozesse. Zumeist wird jedoch auch für zwischenbetriebliche Geschäftsprozesse ein herkömmlicher Integrationsansatz gewählt, welcher (a) nur die spezifische Sicht eines Unternehmens auf den zwischenbetrieblichen Geschäftsprozess hat und (b) zumeist von der IT Ebene startet. Effiziente B2B Integration erfordert jedoch eine gesamtheitliche Modellierungsmethode, welche (a) zwischenbetriebliche Integration aus einer neutralen Perspektive beschreibt und (b) die Geschäftsmodelle, die Geschäftsprozesse, sowie die IT Implementierung aller beteiligten Partner einbezieht. Geschäftsmodelle beschreiben dabei die Akteure in einem Unternehmensnetzwerk und den im Netzwerk stattfindenden Wertefluss. Geschäftsprozessmodelle spezifizieren im B2B Kontext die zwischenbetrieblichen Abläufe, welche den Wertefluss reflektieren müssen. Auf der IT Ebene werden die Geschäftsprozessmodelle schließlich entsprechend implementiert. Diese Arbeit stellt eine UML-basierte, modellgetriebene Methode zur B2B Integration vor, die diese drei Ebenen umfasst.

Kapitel 1 liefert eine Einleitung in das Thema B2B Integration. Wir motivieren unseren Ansatz aufgrund von bekannten Problemen herkömmlicher B2B Integrationsansätze und aktuellen Errungenschaften, welche einen Einfluss auf diese Domäne haben. Anschließend wird in Kapitel 2 relevante Literatur aus dem Problemkontext vorgestellt. Die Literaturdiskussion ist entsprechend der drei Ebenen, der von uns entwickelten Modellierungsmethode strukturiert. Kapitel 3 stellt ein reales Beispiel aus dem Bereich des Abfallmanagements in der Europäischen Union vor, welches in den weiteren Kapiteln als begleitendes Beispiel für die vorgestellten Konzepte dient.

Kapitel 4 beschäftigt sich mit der Beschreibung von wirtschaftlichen Zusammenhängen mittels Geschäftsmodellen auf Basis von e^{3}value, einem der bekanntesten Ansätze im Bereich Geschäftsmodellmodellierung. Die proprietäre Notation von e^{3}value wird nach UML übergeleitet. Das resultierende UML Profil für e^{3}value fügt sich nahtlos in die Notation auf den folgenden Ebenen ein, welche ebenfalls auf UML basieren.

Die Ebene der zwischenbetrieblichen Geschäftsprozessmodellierung ist Inhalt von Kapitel 5. Dabei bauen wir auf die Konzepte

der UN/CEFACT's Modeling Methodology (UMM) auf, welche wir als aktives Mitglied in der Version 1.0 mitentwickelt haben. In diesem Kapitel werden Schwächen von UMM 1.0 aufgezeigt sowie die entsprechenden Lösungen zu deren Überwindung vorgestellt. In Kapitel 6 wird eine formale Repräsentation der Schnittstellen entwickelt, die jeder Partner in einem zwischenbetrieblichen Geschäftsprozess, der mit UMM entworfen wurde, unterstützen muss. Die Spezifikation der Schnittstellen erfolgt zuerst plattformunabhängig, mittels UML Zustandsdiagrammen. Entsprechend dieser Spezifikation wird in den folgenden Kapiteln 7 und 8 die modellgetriebene Entwicklung in Richtung der IT Ebene vervollständigt. Die Ableitung von Web Services Artefakten aus UMM Prozessen wird in Kapitel 7 beschrieben, während Kapitel 8 die Codegenerierung von Artefakten der Windows Workflow Foundation erläutert.

B2B setzt voraus, dass sich potentielle Geschäftspartner online - auf Basis ihrer Geschäftsmodell- und Geschäftsprozessbeschreibungen - finden können. Für diesen Vorgang spielen e-Business Registries eine zentrale Rolle. Kapitel 9 stellt ein Metamodell für e-Business Registries vor, das die die Registrierung der im vorgestellten Modellierungsansatz entwickelten Artefakte unterstützt. Kapitel 10 gibt eine Zusammenfassung dieser Arbeit und weist zudem auf ungelöste Fragen hin, die als Anhaltspunkte für zukünftige Forschungsthemen dienen könnnen.

Zusammengefasst liefert die vorliegende Arbeit folgende fünf Beiträge zum Stand der Forschung im B2B: (1) eine UML-basierte Notation von e^{3}value zur Beschreibung von Werteflüssen in einem Unternehmensnetzwerk; (2) Erweiterungen zur Verbesserung der UN/CEFACT Modeling Methodology für die Beschreibung zwischenbetrieblicher Abläufe; (3) eine formale Spezifikation für die erforderlichen Schnittstellen der Partner in UMM Prozessen; (4) einen modellgetriebenen Ansatz zur Generierung von Web Services und Windows Workflow Foundation Artefakten; (5) ein Metamodell für eine e-Business Registry zur Unterstützung des vorgestellten Modellierungsansatzes.

Dementsprechend liefert diese Dissertation eine Verbesserung des gegenwärtigen Ansatzes zur B2B Integration durch die Entwicklung von Konzepten auf der ökonomischen Ebene, der Prozessebene und der Implementierungsebene und einer durchgängigen Methode.

Abstract

Business-to-business (B2B) electronic commerce builds upon inter-organizational business processes that cross the borders of enterprises. Their design and implementation presupposes a different approach than intra-organizational processes do. Experience shows that bottom-up approaches starting from the IT layer of a single enterprise - expecting that all other business partners adjust to it - do not work out. Instead, a prolific B2B design approach must consider three layers in a top-down manner: Firstly, the economic perspective identifies the players and their value exchanges within a business network resulting in a business model. Secondly, business collaboration models specify the choreography of inter-organizational business processes in accordance with the business model. Finally, the business collaboration models are transformed to deployment artifacts to be interpreted by IT systems. In this thesis, we propose a design approach for B2B integration based on the Unified Modeling Language (UML) considering all three layers.

This thesis starts by giving an introduction to B2B integration. We motivate our approach by shortcomings of the past and some relevant achievements that have an impact on our domain. A thorough overview of related work is given in section 2, which is structured according to the three layers mentioned above. Section 3 introduces a real-world case taken from the domain of European waste management. This case accompanies the remainder of this thesis as a continuous example.

In section 4, we cover the economic perspective of our design approach. This layer focuses on business modeling in order to explore the economic rationale behind a B2B scenario. On this layer we make use of e^{3}value - an already popular approach for business modeling currently using its own notation. We propose a UML profile for the e^{3}value methodology for creating business models in our approach. Thereby, we reach convergence on the underlying modeling notation towards UML on the different layers.

Subsequently, we cover the process layer in section 5. Since our approach targets at B2B integration, we concentrate on the choreography of inter-organizational business processes - so-called business collaboration models. We build upon UN/CEFACT's Modeling Methodology (UMM), a global standard developed by the United Nations Centre of Trade Facilitation and Electronic Business (UN/CEFACT). We have co-edited the UMM 1.0 specification as active members of UN/CEFACT. In the course of section 5 we identify several shortcomings of UMM 1.0 and provide adequate solutions.

In section 6, we take a first step to bridge the process layer and the deployment layer. We formally specify the behavior of a business service interface that supports the specific UMM semantics. We use UML state machines for a platform-independent specification serving as a blueprint for platform-specific deployment described in section 7 and 8, respectively. Thereby, section 7 covers a model-driven approach to create Web Services artifacts from UMM models and section 8 describes the derivation of Windows Workflow Foundation artifacts.

Finally, a successful approach to B2B relies on the successful discovery of potential business partners. Registries are a promising approach to support the discovery in a B2B scenario. Therefore, we propose an e-business registry model supporting our three-layer approach in section 9. Section 10 concludes this thesis by giving a concise summary of our work. Furthermore, we sketch issues that are still unresolved in order to highlight possible links to future work.

In summary, this thesis provides the following five contributions: (1) a UML notation for value-based requirements engineering based on e^{3}value; (2) improvements to UN/CEFACT's Modeling Methodology to model inter-organization business processes; (3) a formal specification of UMM business service interfaces; (4) a model-driven approach to generate Web Services and Windows Workflow artifacts; (5) an e-business registry meta model supporting our approach. In short, the overall approach enhances current efforts in B2B integration by bridging the economic, the process, and the implementation layer.

Contents

1 Introduction

Are you old enough to remember the dot-com boom? This was the time when stock markets like the NASDAQ had their peak due to the boom of the Internet. The pioneers of that era discovered the Internet as a new communication and distribution channel. The foundation of a huge number of Internet-based companies - the so-called dot-com companies - coined this area. Although most of them lacked a well-defined business model, the dot-com boom encouraged investors to pump enormous venture capital in the emerging market. If you remember the dot-com boom, you know that the dot-com bubble burst in 2001.

e-business vs. e-commerce

All the dot-com companies had one goal in common: They wanted to exploit the Internet for doing business. Two new terms originated in this era: e-commerce and e-business. Although often used synonymously, current literature basically differs between them as follows: (i) e-business - as coined by IBM in the 1990s - refers to the redesign (and thus, IT support) of all business processes - internal as well as external ones - along the supply chain [103]; (ii) e-commerce, as defined in [107], constitutes the trading of physical goods and intangibles such as information and services by electronic means. Thereby, e-commerce also includes electronic support for collaboration between companies.

Classification criteria for e-commerce

The field of e-commerce may be classified according to several criteria [66]: (i) participating actors, (ii) phases of a trading transaction, (iii) the monetary volume of a transaction, (iv) and the economic and technical layers of a transaction. For now, we do not concentrate on the phases and the monetary volume of a trading transaction. In terms of participating actors we may distinguish between three different types: administration, business, consumers - the so-called e-commerce ABC [123]. There may be relationships between all of them - e.g., Business-to-Business (B2B), Business-to-Consumer (B2C), Business-to-Administration (B2A), Consumer-to-Consumer (C2C), etc.

In this thesis we concentrate on Business-to-Business (B2B). Furthermore, we make the following assumption: from a technical perspective administration or a government agency is expected to behave like a company (e.g., in terms of IT support, business process focus, etc.). Since this thesis focus on the technical realization of business processes (and not on legal aspects), we further subsume B2A, A2A, and A2B under the term B2B.

Moreover, corresponding to [66] the economic and technical layers of an electronic trading transaction may be distinguished as follows:

- Market models

- Business models
- Applications
- Software components
- Fundamental technologies

In this thesis we concentrate on the first four layers of this classification and take the fundamental technologies (e.g., XML, HTTP, TCP/IP, etc.) as given. In terms of market models and business models, we consider and describe the business model of a company in subject to its role within a business network or an electronic market. Hence, we further regard these two layers as a merged one. Moreover, we refer to the application layer as the business process layer in order to be independent of specific vendor solutions. Finally, we substitute the term "software components" by "deployment layer".

Motivating trends for this thesis

Having introduced the focus of this thesis, let's explore the motivation behind it. There are some recent achievements in both, business and technology, observable that have driven the approach of this thesis. These drivers on different levels are in line with those identified by Kalakota and Robinson in [39].

- *Measuring the return on investment (ROI) of e-commerce ideas:* Today, investments in innovative e-commerce ideas are hard to gain. The management wants to be convinced that a certain idea returns an investment in a conceivable time. Consequently, entering new business networks in order to conduct e-commerce needs to be motivated accordingly. This can be realized by modeling, formalizing, and simulating the business model of a business network.
- *Business process management (BPM):* Business process orientation has become a major trend since the 1990s. Business process management enables the elicitation, design, execution, monitoring, and optimization of operational sequences in order to realize continuous improvement according to business goals. In short, companies make use of BPM to stay flexible, innovative, and competitive. Effective BPM needs to be supported by corresponding software solutions. In recent years, the market has substantially grown for both, consulting companies and software vendors.
- *Enterprise application integration (EAI):* The need for application integration is a result of the situation that every company runs software products from different software vendors. Typical business processes span different software products. A smooth information flow along business processes hence requires the integration of different software products. In B2B scenarios, business processes cross the borders of companies. Thus, software products distributed across different companies have be integrated for realizing e-commerce transactions.
- *Globalization:* Beside the social, cultural, political, and environmental effects of the ongoing globalization, it dramatically changes economy and trade. The latter has even more been boosted by the impact of the Internet as a distribution and communication channel. The "enablement of a global electronic

market, where companies of any size, in any global region conduct business using the Internet" [44] has already been the aim of the ebXML initiative at the peak of the dot-com boom. The work in this thesis is motivated by the ideas and concepts of ebXML.

- *Services science:* The term information society has been discussed for several decades in literature [55, 4, 12]. In general, an information society is characterized by the degree of services and information that is produced and consumed. The information society is considered as the successor of the industrial society having the majority of employees engaged in the services sector [4]. Recently, service orientation has become an important and popular paradigm in the IT sector. Starting with service-oriented architectures (SOA) and Web Services, the notion of services has been introduced to business process management, and has eventually resulted in a call for a "science of services" [102, 8]. In this proposal, the authors identify the need for "service science" [56] as an interdisciplinary approach for the design and implementation of service systems, where people and technology exchange services in order to create value for others.

Beside these recent trends, there is another, major motivation for the approach proposed in thesis: the shortcomings in traditional B2B approaches, which are addressed in the following sub-section.

1.1 Synopsis about the past of business-to-business electronic commerce

Although the term e-commerce was coined during the dot-com boom, conducting electronic business between enterprises was not an invention of the Internet age, but has existed for decades. However, requirements of B2B electronic commerce have changed. In former days, when B2B electronic commerce was referred to as Electronic Data Interchange (EDI), its focus was document-centric. This means, in order to avoid bilateral agreements on business documents, business partners agreed on business document standards. But, as history has shown, the results of these standardization efforts were overloaded and ambiguous document standards. This led to costly EDI systems and participation in electronic commerce was reserved to large companies that have been able to afford such implementations. As a consequence, only limited circles of acquainted enterprises exchanged business messages electronically in order to reach their business goals and gain financial benefits.

Did XML solve the problem?

With the advent of the Internet, the area of electronic business started to boom. In the field of B2B electronic commerce, small and medium sized companies saw their chance to enter electronic markets. It was envisioned to find new business partners electronically and to dynamically conduct e-business. In addition, with the advent

of XML, there arose hope that the problems of EDI will be solved all of a sudden. However, this was a broad misconception - the pure mapping of a delimiter-based syntax, as used in traditional EDI standards, to a tag-based syntax did not yield a solution to the shortcomings of traditional EDI.

Thinking in business processes instead of documents

At this time, business process management was already in use to implement workflows internal to a company. Enterprises started to adopt business process modeling in order to monitor their procedures and to design process-based solutions. In the context of EDI, the concept of a business process has already existed - but buried in the minds of those people that were responsible for the inter-organizational systems. These people were aware, for example, what to do next when an invoice was received and how to trigger manual compensation if - in case of a failure - a dunning letter was received before an invoice. They were able to resolve the problem by phoning the business partner, because their counterpart was known to them. In this respect, the concept of a business process - as a protocol for specifying the course of business - was already there, but existed only as a mental model.

A collaborative business process is a protocol describing the interactions

According to the idea of modern electronic markets where companies of almost any size conduct business in a dynamic way, business partners are not known to each other as described above. Dynamic B2B e-business involves spontaneous agreements, which might exist just for one economic transaction. There are no offline negotiations and no face-to-face relationships. Instead, agreements are made online, which requires business partners to unambiguously define how to conduct business with them. In other words, business partners must describe what business processes they offer in order to show potential business partners how to interact with them. Such collaborative business process models capture the flow of business information between business partners, which is exchanged to reach a certain business goal. Furthermore, business process models help to elicit the minimum of business information that is required in each step of a collaborative business process. This eliminates redundantly transmitting information, which in turn lowers the risk of semantic differences between exchanged business information. In addition, business processes are also subject to standardization efforts in the future, which allow for a cost-effective implementation of commercial off-the-shelf software (COTS) supporting these standardized processes.

The new SOA paradigm aims for the alignment of business and IT

Regarding software development, we observe a major paradigm shift towards service-based communication - known as service-oriented architectures (SOA). The SOA concept promises reuse of components and allows for an easier implementation of communication across heterogeneous platforms and among enterprises based on open and free specifications. Furthermore, SOA may be parameterized by deployment artifacts such as machine-readable business process specification and business document schemes. Such deployment artifacts enable a more flexible and easier adoption of service-based systems to changing environments - even at runtime.

Contemporary B2B approaches have a focus on business processes and service orientation

Process-centric B2B approaches combine the paradigm shifts on

the business and on the technical level. On the business level, business process models capture business logic independent of the underlying technology platform. Those models may then be implemented using different technology platforms. This approach allows enterprises to quickly adapt to changing and newly emerging technologies, since business logic doesn't change as fast as technology does. This is in line with the services science approach [8] focusing at service driven innovation. But one should note that services are defined differently in management science and in computer science. In the latter one, a service is a simple or complex task executed within an organization on behalf of a customer [86]. In management science, a service is defined as a business economic activity (mostly intangible in nature), offered by one party to another in order to achieve a certain benefit [131, 43]. In either case, a service offering is delivered by executing a business process. This implies that a service corresponds to the interface of a business process by encapsulating its behavior.

Faster changing market conditions entail flexible IT environments

In order to stay in today's business, companies must quickly adopt to faster and faster changing business conditions. Business models must reflect these changes, business processes must be designed supporting the value exchanges, and IT applications must adjust to changing company goals. These requirements are often referred to as business/IT alignment. Management expects this to happen at low cost. Service-oriented architectures have the potential to provide a new level of flexibility in regard to the adaptation of the affected IT systems. Whereas in former days change requests to the IT resulted in a rigorous change of IT system implementations, nowadays, service-oriented IT departments focus the challenge of service alignment.

1.2 Service-oriented architectures and the Open-edi reference model

In order to analyze the potential of SOA, we first have to understand what SOA refers to. According to the OASIS SOA Reference Model [75], SOA stands for a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. This specification continues that in general, entities (people and organizations) create capabilities to solve the problems they face in the course of their business. The main drivers for a SOA are the management of the growth of large-scale enterprise systems, facilitating Internet-scale provision and use of services, and the reduction of costs in inter-organizational cooperations.

SOA is not just an implementation technology like Web Services

From the above excerpts of the SOA Reference Model it becomes evident that SOA is not limited to implementation issues addressed by Web Services - the current technology of choice to implement a SOA. This means that a SOA-based approach to inter-organizational cooperation must also address the business requirements in organizing and utilizing a distributed solution for a business partnership. This is in line with the Open-edi reference model, which became an ISO standard for inter-organizational systems in 1997 [37]. Open-

edi distinguishes between the business operational view (BOV) and the functional service view (FSV). The BOV addresses the business aspects such as business information, business conventions, agreements and rules among organizations. The FSV is related to information technology aspects, which are necessary to support the execution of a business collaboration. Accordingly, the FSV implements the scenarios developed in the BOV.

Successful B2B integration spans over three perspectives

At first sight one might tend to reduce the BOV layer to model the business interactions between the involved business partners. However, it is at least equally important to analyze the value propositions of each of the participating business partners [23]. Consequently, separating the concerns in developing inter-organizational systems results in three different perspectives shown in figure 1.1. The management focuses on the value perspective described by business models. Business people have a process perspective described by business process models that operationalize the business models. The IT people focus on the execution perspective of the deployment artifacts implementing the business process models.

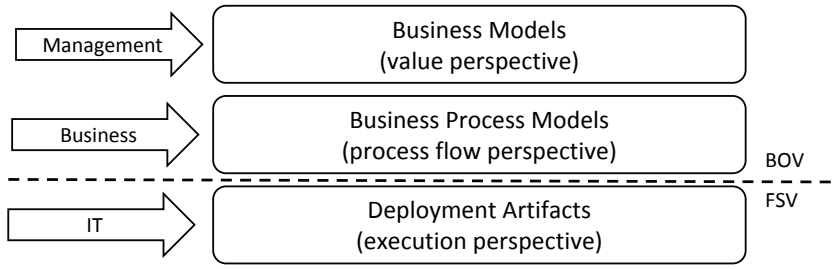


Figure 1.1
The three layers of B2B collaborations

This thesis proposes a modeling approach for B2B integration spanning the three perspectives

For modeling B2B processes, the United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT) standardizes UN/-CEFACT's Modeling Methodology (UMM). UMM is located on the middle layer in figure 1.1.

We have been participating in the standardization efforts of UN/-CEFACT since 2005. Since that time, we contributed to the development of UMM and have been co-authors of the UMM specification. Several concepts that have flown into the standardization of UMM, were subject to our academic research.

This thesis is based on the UMM and extends it to an integrated modeling approach for B2B systems spanning business models, business process models, and deployment artifacts. The developed modeling approach has a clear top-down focus: It starts on the top layer with the definition of business models. Based on a well-defined business model, appropriate collaborative business process models are created. Model-driven development techniques transform business process models to machine-interpretable deployment artifacts.

1.3 Shortcomings of traditional SOA-based B2B approaches

Traditional SOA-based approaches for inter-organizational systems have three major shortcomings. They are identified and detailed in the following sub-section. Then, in sub-section 1.4 we elaborate on the contributions of this thesis, which together compose our three-layered model-driven approach for B2B integration. We are convinced that by addressing the identified shortcomings our approach improves the state of the art in B2B integration.

Complexity goes beyond simple Web Services model

Web Services are certainly an appropriate technology to implement inter-organizational business processes. However, the traditional approach to implement Web Services may not be appropriate as we elaborate in the following: Companies provide access to application components that implement key functionalities via the Internet. These application components are known as Web Services and their interfaces are described by Web Service Definition Language (WSDL) artifacts.

The simple "find-bind-invoke" paradigm is not working as envisioned

A provider of a service is expected to register all the details of accessing the service. Potential service consumers may search the registry for a suitable service and retrieve its access information. By following the access information the service consumer binds its application to the service offered by the service provider. This rather simplistic approach might work for simple services that have no dependencies on other services executed between the service consumer and the service provider. In practice, even this relatively simple case is not working as envisioned - this is underpinned by shutting down the public UDDI directories previously offered by Microsoft, IBM, and SAP in 2005.

A B2B process is a complex choreography of service interactions

An inter-organizational business process is much more complicated than a simple service request and its optional response implemented by a single Web Service. It usually comprises many interactions, i.e. Web Services, between the same set of business partners. An inter-organizational business process cannot be described by a stateless protocol. The flow of interactions (Web Services) between the business partners depends on the business entity states that are a result of the interactions. Inasmuch the interactions must be executed in an agreed order. This flow is described by a choreography of Web Services.

Choreography vs. orchestration

One should note that we distinguish between choreography and orchestration as follows: An orchestration describes a process that is executed internal to a company. A local choreography is a projection on an orchestration - it contains only those tasks of a process that are visible to the outside world. Those tasks as well as the order between them describe the required flow of message exchanges in order to interact with the process. Thus, a local choreography has always a partner-specific viewpoint. A global choreography describes a pro-

cess from a neutral viewpoint by capturing the observable behavior between complementary local choreographies.

The simple Web Service model does not scale for B2B

According to the Web Services model, a business partner must not only register the Web Services for each of the interactions, but also the choreography of the Web Services in a registry. A potential business partner must search the registry for choreographies that are compliant to his own "preferred" choreography. This means that a service requester must first search for business processes that seem to match according to their semantic description. For each of these business processes it must retrieve the choreography description. Next, it has to check that the retrieved local choreography is complementary to its own local choreography. Complementary means that the general flow of interactions specified in the choreography is identical, but the task in each interaction is the corresponding one - i.e., if one business partner invokes a service the other one must retrieve a service call of the same service. Furthermore, the requester has to check its support of each of the individual services within the choreography as specified by the service provider.

It follows that searching for a complementary inter-organizational business process is not a simple query to a registry. Imagine that a SOA approach to inter-organizational systems is successful. This means a registry contains millions of business process descriptions. It becomes, evident that such a retrieval process does not scale.

Bottom-up approach

In the previous sub-section we concluded that the retrieval process does not scale. Furthermore, we believe that it is even unlikely in a bottom-up approach to find complementary business processes in a registry. In general, the term "bottom-up approach" means that (existing) systems are put together in order to realize an ampler system. The former become then sub-systems of the resulting larger system.

Business partners should not develop their part of a business collaboration in isolation

In terms of B2B integration, we refer by "bottom-up approach" to efforts that are commenced by companies in isolation. They use some kind of business process modeling method to describe the orchestration of their internal business processes. As we already know, the interface that describes how to interact with the business process from the outside world is called the local choreography. If business partners develop their local choreographies in isolation from each other it is rather unlikely that the resulting business process models - i.e., the global choreography - will match. The models may differ in the set of activities that compose the local choreography, the flow between them, different activity/service names, and different input/output formats - just to name a few. As a matter of consequence, the business partners will not be able to perform an inter-organizational business process by electronic means.

Business collaboration models are a kind of contract and are modeled from a neutral perspective

In order to overcome these shortcomings, this thesis follows a top-down approach. In general, a "top-down approach" describes a decomposition approach. With respect to B2B, a top-down approach commences with a global view on the integration efforts. The global view is described by an agreed model of an inter-organizational business process, which should be followed by the local choreographies

of each partner. The agreed inter-organizational process serves more or less as a contract between the business partners by taking a neutral perspective on the required flow of interactions. Each business partner is then able to derive its local choreography and to bind its internal processes to the global choreography.

This approach increases the chance for finding complementary business processes. Furthermore, it leads to a retrieval process that will scale, since service requesters will search for business partners that support the same global choreography of an inter-organizational business process in the complementary role.

Missing value perspective

A business model explains the economic rationale behind a business collaboration

The development of an inter-organizational system does usually not consider the value proposition of participating companies in the inter-organizational process. It rather concentrates on the definition of the sequence and structure of the inter-organizational business process. However, before two or more companies will engage in a collaborative process it is important to analyze *how* and *why* these enterprises cooperate from a business perspective. The answer to this questions is given by the value perspective described by means of a business model. A business model - which differs from a business process model - has to define which value objects are created by which partner and which value objects are exchanged between which partners. This reveals the value proposition for each participating partner and indicates to the management whether participation in the inter-organizational business process is economically relevant or not. However, the value perspective is rarely used in traditional approaches.

1.4 Contributions of this thesis

As mentioned before, this thesis builds on UMM in order to create an integrated modeling approach spanning the value perspective, the business process perspective, and finally the deployment perspective. This section discusses the five contributions (c.f., figure 1.2) that constitute the proposed modeling approach.

Problem 1: UMM lacks a value perspective for justifying B2B integration scenarios

UMM concentrates on specifying business process models as well as their requirements, but lacks value-based requirements engineering by means of business models. The value perspective enables the justification of a B2B integration scenario from an economic perspective. Furthermore, the business model approach allows for identifying business processes that have to be supported in order to participate in a business network.

Contribution 1: Integration of value-based requirements engineering in UMM based on e^3value concepts

In order to provide value-based requirements engineering in our UML-based approach, we aim to integrate business modeling concepts in UMM. In section 2, we identify several business modeling approaches. In our approach, we integrate concepts of the e^3value ontology for modeling the economic rationale behind business collaborations. The e^3value ontology currently defines its own notation, which makes it cumbersomely to use in a UMM-based project. An efficient

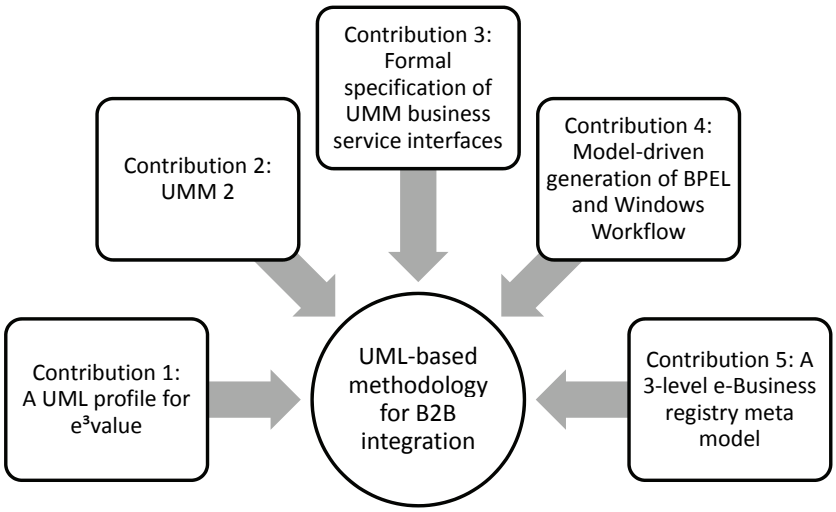


Figure 1.2
The five contributions of this thesis

integration of e^{3}value with UMM presupposes that both modeling approaches are based on a single modeling language. This thesis proposes a UML profile for e3-Value. Consequently, UML as a single underlying modeling language is used throughout our modeling approach. This implies two major benefits for stakeholders: (i) they can stick to a single modeling notation and (ii) the whole modeling process can take place in a single tool environment.

Problem 2: UMM 1.0 is built on UML 1.4, which is outdated. Stakeholders demand a switch to UML 2

The UMM Foundation Module 1.0 [109] became a UN/CEFACT technical specification in 2006 (c.f. the time line in figure 1.3). It was the first UMM version that was formally defined as a UML profile. A UML profile customizes UML for a domain-specific purpose by defining a set of stereotypes, tagged values and constraints. When the UMM 1.0 project was started within UN/CEFACT in 2004, UML 2.0 was not considered as stable enough. Hence, UMM 1.0 is built on UML 1.4, but today UML 2 is considered as the state of the art.

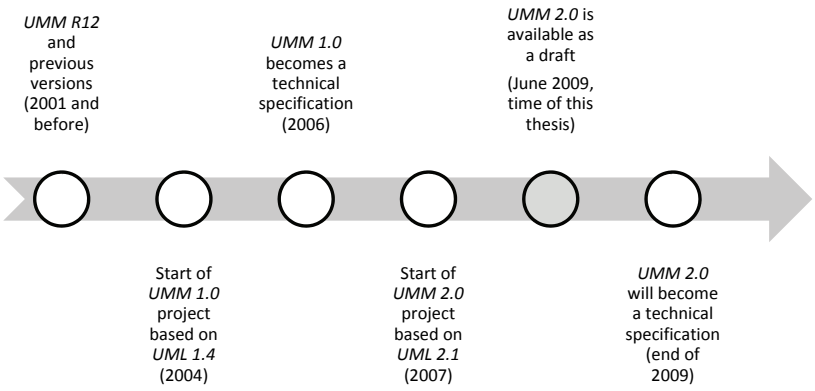


Figure 1.3
The history of UMM

Therefore, UMM stakeholders start asking for an update of UMM - i.e., a "UMM 2.0" that is defined on top of UML 2. Beside the criticism that UMM 1.0 is based on an outdated UML, the meta model of UMM 1.0 is still often bashed as too complex (e.g., UMM 1.0 mod-

els often result in excessive package structures). Furthermore, some workarounds that were necessary in the meta model due to the use of UML 1.4 contribute to often bloated UMM models. UML 2 provides major improvements to key modeling elements of the UMM (e.g., activity diagrams), which render these workarounds unnecessary. As a consequence, the UMM 2.0 project has been started in UN/CEFACT in 2007.

Contribution 2: Development of a "UMM 2.0" that sits on top of UML 2

In June 2009, when this thesis is finished, UMM 2.0 is one step before becoming a technical specification according to the terms of reference for UN/CEFACT standards. The work in this thesis has had a rigorous influence on the development of UMM 2.0. In fact, the contributions outlined in section 5 have been submitted for standardization and have been integrated in UMM 2.0.

The result is a more straightforward UMM that builds on current standards. This ensures further adoption by potential users and fosters the support of tool vendors. The use of UML 2 also eliminates the above mentioned workarounds in the UMM meta model.

Problem 3: An unambiguous specification of business service interfaces is required when implementing collaborative business processes

After gaining an agreement on a common UMM model, the IT systems on each partner's side must be implemented accordingly. For realizing B2B processes, each partner exposes a business service interface (BSI) to the outside world. The BSI is responsible for binding the collaborative processes to the internal ones. In other words, it carries out the inter-organizational information exchange and communicates business information to/from the internal business application. In order to implement a BSI, business partners require unambiguous specifications of their expected behavior in business document exchanges. The specification of a business service interface must cover (i) how to re-act on incoming messages, (ii) how to react on messages expected, but not received, and (iii) how to handle exceptions detected internally or communicated by partners. Currently, such an unambiguous specification does not exist for business service interfaces supporting UMM business transactions.

Contribution 3: Formal specification of UMM business service interfaces by means of state machines

Another contribution of this thesis, which brings the collaborative business process models to deployment, is a formal specification of business service interfaces that support UMM business transactions. Starting from the global UMM model, we analyze which actions must be carried out by which partner in which state of the business document exchange. This may become challenging for the following reason: UMM models describe the business collaboration from a neutral viewpoint, whereas business service interfaces implement the same business collaboration from a partner-specific viewpoint. Thus, the business service interface implements only those message exchanges relevant for a specific business partner. Furthermore, we identify actions that are required to fit a business service interface in an organization's IT landscape. In other words, we describe the actions that are required to bind the business service interface to the internal processes. The resulting formal specification is a UML state machine. It serves as a blueprint for implementing collaborative business processes and guides, therefore, the deployment on concrete IT platforms.

Problem 4: Deployment artifacts should not be created manually

The blueprints for the business service interfaces show that busi-

ness document exchanges according to UMM semantics follow always a set of patterns. Thus, it would be cumbersome to craft every BSI implementation by hand. It is rather desired to apply model-transformation techniques in order to generate concrete BSI implementations according to the blueprints mentioned above. Contemporary deployment platforms like SOA or declarative workflow technologies support the idea of model-driven development. They are not restricted to hard-coded implementations, but allow for the declarative configuration of process engines using machine-interpretable process models - i.e., deployment artifacts that are mostly XML-based. These deployment artifacts are not intended to be created manually. In fact, they should be derived from formal process models by employing model-driven development techniques.

Contribution 4: Model-driven generation of business service interfaces in BPEL and Windows Workflow

This thesis covers model-driven development techniques for deploying UMM choreographies to two implementation platforms for business service interfaces: Web Services and Windows Workflow. Starting from the global perspective provides three major benefits: (i) the business collaboration model serves as a kind of contract partners agree on. (ii) the business collaboration model allows the generation of complementary business service interfaces for each partner. This ensures that the business service interfaces interact in accordance to the global choreography of the business collaboration. (iii) the model-driven approach allows quick and cheap customizations of a B2B system to fast changing business requirements.

Problem 5: A dynamic B2B environment requires an e-Business registry

The vision of dynamic B2B presupposes that business partners find each other electronically based on the business scenarios they are interested in. In order to get into contact with potential partners, companies need to exchange information about their e-business capabilities. Consequently, this scenario exacts the concept of an e-business registry, which is a central site for companies to publish as well as to consume e-business related information. According to our three-layer approach for B2B integration, such an e-Business registry has to be capable of managing business models, inter-organizational business process models, and deployment artifacts.

Contribution 5: a 3-level e-business registry meta model

For publishing artifacts that are created by our approach, we specify a registry meta model supporting the specific requirements of our approach on top of the ebXML registry standard. One major responsibility of the e-business registry is to link artifacts together that describe the same business scenario from different perspectives. Consequently, business partners may query the e-business registry first by economic considerations. Having found a business scenario of economic interest, business partners may then review business processes that support the chosen business model. Finally, deployment artifacts associated to the business process model are retrieved, which may be used to realize local implementations. The proposed e-business registry complements our modeling approach by considering not only technical details, but also the business perspective.

This thesis is located in the research field of business informatics. It combines an analytical and a constructive approach. The analytical part comprises the identification of the three layers of B2B integration as well as the linking between them. The constructive part

includes the findings that have flown into the standardization efforts of UN/CEFACT.

1.5 Structure of this thesis

Section 2: Related work

In section 2 we review related work in respect to business modeling, business process modeling, and deployment platforms.

Section 3: Description of the waste management example

Throughout this thesis, a common example is used to demonstrate our approach. The example is taken from the EU-project EUDIN (European Data Interchange for Waste Notification Systems), which supersedes a paper-based document exchange for waste transports by electronic information exchange. Section 3 introduces the reader into the domain of waste transports within the European Union and details the scenario of our example.

Section 4: Business model layer

Then, section 4 concentrates on the business modeling layer. We stress the importance of business modeling for eliciting the economic rationale of an e-business scenario and briefly present the concepts of e^{3}value. As the main contribution of this section, we propose five alternatives for defining a UML profile for e^{3}value. We conclude with an evaluation, which of the alternatives fits best for our needs. The UML profile for e^{3}value is a milestone for integrating value-based requirements engineering in our UML-based approach for B2B integration.

Section 5: Inter-organizational business process modeling

Section 5 focuses on the path to UMM 2.0. We briefly outline the history of UMM and stress the needs for UMM 2.0 as proposed in this thesis. We do this by emphasizing the limitations of UMM 1.0 and suggest solutions to overcome each of the limitations. The proposed improvements have been contributed to UN/CEFACT and have become part of UMM 2.0.

Section 6: State machines for business service interfaces

Beginning with section 6 we start to plan the deployment of inter-organizational business process models. In fact, we formally specify the behavior of a business partner in order to be compliant to the global choreography. The resulting artifact is a UML state machine, which serves as a blueprint for generating business service interfaces.

Section 7: Mapping UMM to BPEL and WSDL

Section 7 focuses on the deployment of UMM business transactions to the Web Services platform. The resulting business service interfaces are specified by means of WSDL and BPEL, whereby WSDL describes the service interfaces and BPEL the flow of interactions between them.

Section 8: Windows Workflow

In section 8 we focus on an alternative deployment platform - Windows Workflow. Similar to the previous section, we describe in detail the transformation process from the UML-based inter-organizational business process model understandable by humans to a declarative workflow model, which is executable by a workflow engine.

Section 9: The e-business registry

The created artifacts should be made publicly available in order to be found by potential business partners. We begin section 9 by stressing the need for such a discovery process in the field of B2B. Our discovery process is based on the concept of a registry. Hence, this section introduces an e-business registry meta model for manag-

ing the modeling artifacts produced by our approach and shows how it complements our overall modeling approach for B2B integration.

Section 10: Unresolved issues and conclusion

Finally, section 10 concludes this thesis. The focus of this section is two-fold: first, we give a short summary of our contributions. Second, we point out open issues, which were revealed in the course of this thesis, but have not been targeted yet.

2 Related Work

In this section we discuss the state of the art and the related work in the scope of our three layered approach (cf. figure 1.1 in section 1). The findings in this section are an extension to the survey we gave in [11]. The structure of this section - visualized in figure 2.1 - is line with our top-down approach: (i) We start with current literature about business models; (ii) we review related business process modeling approaches before (iii) discussing technologies on the deployment layer; (iv) finally, we cover existing work that is related to our approach in general, but cannot be unambiguously assigned to one of the three layers.

Business Modeling

In the definition of a business model, we follow Timmers [106], who defines it as an architecture for the product, service and information flows, including a description of the various actors and their roles, together with a description of the sources of revenues and potential benefits. Several other definitions can be found in [88], which presents a framework for structuring and analyzing business models.

The focus of business model research has changed due to fast moving market conditions

According to the work in [85], the term business models stands for various things: classifications or types of business models (e.g., distribution over certain channels), concrete real world instances of business models (e.g., of a certain company), or concepts (to describe the elements of a business model and the relationships between them). The authors observed that the term first became prominent in the 1990s, which coincidences with the advent of the Internet. The authors of [25] found out that over time the focus of business model research has changed, ranging from establishing taxonomies of business models to describing elements of business models, and finally to building business model ontologies. Such ontologies assist stakeholders in establishing a common understanding by providing a set of vocabularies and concepts that are used to describe the business logic. In fast moving market conditions with the entrance of new players and a deconstruction of value chains, stakeholders are supposed to form business networks in a flexible way on a plug & play basis. Business models should be expressed formally in order to be unambiguous and to be processable in a machine-readable way: (i) This ensures that they can be easily adapted to changing requirements; (ii) they can be analyzed by tools that are capable of simulating different business scenarios to facilitate the selection of the most sustainable one; (iii) they provide a starting point for business process modeling.

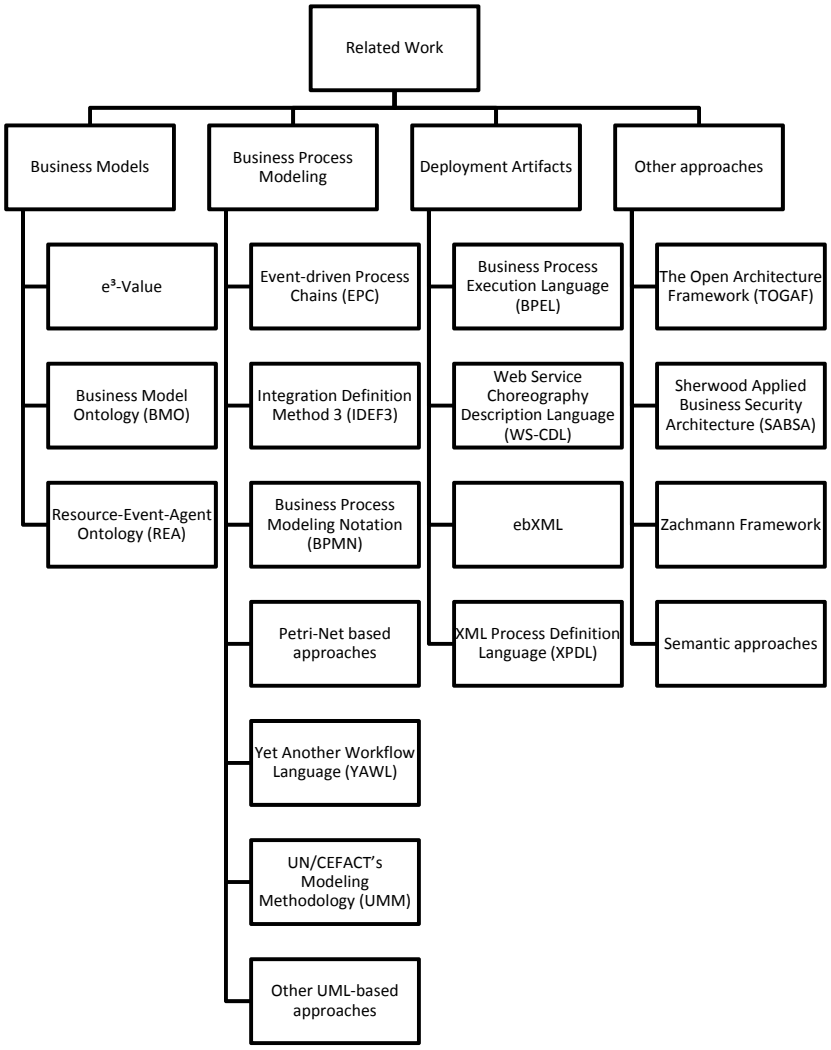


Figure 2.1
Related work structured according to the three layered approach

There exist several methods to capture and model the economics behind the business process: In e^3value [21, 22] a business model is regarded as a value constellation, i.e., a network of enterprises that jointly create and distribute objects of economic value to satisfy a consumer need. Focus is on an economic value proposition, i.e., expressing the objects of values an actor is willing to exchange for other value objects. The model ensures the concept of economic reciprocity, i.e., an actor gives an object of value and gets another object of value in return. Thereby, an object of value may be a physical good, a service, money, or something intangible (e.g., raising the degree of popularity of a certain brand). The model illustrates which actors have economic transactions among each other on an abstract level, but omits the internal processes necessary to create these values. Emphasis is on showing who is doing business with whom.

e^3value is popular approach to describe value constellations in business networks

In the *Business Model Ontology (BMO)* [84], business models are described based on four perspectives, comprising product innovation, infrastructure management, customer relationship and financial aspects and the relationships between them. In contrast to the previous

The Business Model Ontology (BMO) describes a business model from one partner's specific viewpoint

mentioned e^3-value model, which describes the network constellation as a whole, this ontology rather focuses on a specific actor and outlines his position in the business network and how he makes profit. In this way, "product innovation" mainly refers to the value the actor offers to a specific target customer segment whereas "customer relationship" refers to the different distribution channels to deliver the created value. "Infrastructure management" outlines how this value is created by regarding the resources of the actor as well as his business network. "Financial aspect" is influenced by all these three elements and determines the actor's profit model.

In [25], the two business modeling approaches - e^3-value and BMO - are compared based on a framework to identify common characteristics as well as differences. The main difference between the two approaches is their focus. The e^3-value focuses on the network of business partners and on the exchange of values. The focus of BMO is on a specific enterprise thereby expressing its business interactions by pointing out offering- and customer-related aspects.

REA has its origins in the modeling of accounting systems

The *Resource-Event-Agent Ontology (REA)* captures the declarative semantics of the collaborative space between enterprises from an economic viewpoint. The acronym REA is deduced by the three main concepts of the approach: agents (A) exchange resources (R) in order to achieve their economic goals. Thereby, an exchange of resources is driven by economic events (E) - e.g., the need for re-stocking some goods. The REA ontology was developed by McCarty in the 1980's and evolved from a generalized framework for modeling accounting information systems [60] to an ontology for enterprise information systems [33]. The main concepts of REA - resources, exchanges, and agents - are quite similar to the corresponding e^{3}value concepts. The evident benefit of e^{3}value is its intuitive notation and the respective tool implementation provided by its founders [20]. REA currently lacks such a notation, but UML class diagrams may be used. The advantages of REA compared to e^{3}value are its concepts to identify the triggers for economic exchanges (i.e., the events) and, furthermore, the ability to specify commitments and agreements between business partners. The work in [97] investigates how REA and e^{3}value can complement each other.

Business Process Modeling

Business process modeling attracted a lot of attention over the last couple of years. A lot of different approaches to model business processes have been developed. Surveys comparing different types of business process modeling languages are provided in [54, 101, 51]. In this related work section, we use the notation for modeling business processes as a first level of categorization. Some of them are based on UML or Petri-Nets, others have developed their own proprietary graphical notation, and other ones are text based and do not provide a graphical notation at all. Another categorization of business process modeling approaches is based on their scope. Traditionally, the modeling of business processes focuses on business processes internal to an organization fulfilling customer needs. More recent approaches consider inter-organizational business processes. These

approaches describe the inter-organizational business process either from a participant's point of view, i.e. a local choreography, or from a neutral perspective, i.e. a global choreography. A comprehensive survey of inter-organizational business process modeling techniques is given in [93]. Furthermore, we categorize business process modeling approaches according to their binding to the supporting IT infrastructure. Some approaches are mainly used in the requirements specification phase to support the communication with business domain experts. Resulting models are usually on a rather abstract level and tend to hide implementation complexity. Other approaches are more implementation oriented and sometimes seem to be a reverse engineered graphical notation of workflow languages or of Web Services orchestrations and choreographies.

Event-driven Process Chains (EPCs) are utilized in the ARIS framework

First we focus on non-UML based business process modeling approaches. A popular notation are *Event-driven Process Chains (EPCs)*, which is a business process modeling language focusing on control flow dependencies of activities and events in a business process. It is utilized in the ARchitecture of Integrated Information Systems (ARIS) by Scheer [96] as the central method for the conceptual integration of the functional, organizational, data, and output perspective in information systems design. EPCs are usually used for modeling internal business processes on a business level. In recent years many approaches have been developed to migrate the EPC concepts to other notations - e.g. a UML profile for EPC in [42], an XML-based interchange format for EPCs [65], or an extension of the traditional EPC called yEPC (Yet another EPC) proposed in [64].

Another modeling approach is the *Integrated DEFinition Method 3 (IDEF3)* [58]. It is a methodology for modeling business processes and sequences of systems. IDEF3 supports two kind of models: the process flow description and the object state transaction description.

The Business Process Modeling Notations gained strong support from industry and academia

Recently, the *Business Process Modeling Notation (BPMN)* [83] has attracted a lot of attention. BPMN has been developed by the Object Management Group (OMG) in order to enhance a standardized modeling notation, which is readily understandable by different stakeholders - from the business analysts to the technical developers. It may be used for modeling both internal and inter-organizational business processes. BPMN incorporates aspects of already advanced modeling notations (e.g. UML activity diagrams [95], IDEF [59], ebXML BPSS [74], RosettaNet [94], etc.). In order to close the gap between the business process design and the business process implementation, OMG's Business Process Management Initiative (BPMI) standardizes the mapping from BPMN to executable business processes specified in BPEL. BPMN is a graph-based language, whereby BPEL's underlying model is mainly block-structured. Consequently, a mapping between those two language paradigms is challenging [87]. In [38] the authors propose their own notation for a global business process choreography. It focuses on two types of business processes (contract and executable ones) and provides an interface protocol, in order to let these processes collaborate.

Petri-Nets are heavily used for business process modeling

Many approaches to model processes are based on the *Petri-Net theory*, which is both a graphical and mathematical modeling tool

[69]. They are used to model business processes [116] as well as workflow systems [114]. Petri-Nets are also used to model inter-organizational systems. An implementation of the choreography aspects of ISO's Open-edi reference model [37] was contributed by Lee [47]. Similar to this approach other authors used Petri-Nets to define the business processes between organizations [48, 52, 115].

The research in workflow patterns were the inspiration for YAWL

Contemporary business process modeling standards and workflow solutions had been subject to a rigorous analysis based on a set of workflow patterns. Workflow patterns [118] identify a comprehensive set of control flow patterns re-occurring in real-world business processes. Some of the rather complex patterns are not easy to map on existing languages. This inspired the authors to develop their own workflow language called *YAWL (Yet Another Workflow Language)* [117]. YAWL is based on Petri-Nets but adds mechanisms for providing comprehensive support for the identified workflow patterns.

UML-based approaches have become very popular for business process modeling

We continue with *UML-based approaches* for modeling business processes. Most of these approaches are based on UML activity diagrams [95]. They either provide just guidelines on using activity diagrams for this special purpose or they specify a UML profile. A very popular approach to model internal business processes is provided by Penker and Eriksson [90]. They show how to use UML for documenting the entire enterprise. It is outlined how to model businesses, from business architecture to processes, business rules, and goals. Furthermore they define business patterns that provide reusable solutions to common business problems. A UML profile for intra-organizational business processes was proposed in [53]. The authors customize the UML 2 activity diagram for modeling business processes considering business process goals and performance measure. This approach was then refined in [41].

Other approaches for modeling B2B collaborations are also based on the UML

For the purpose of representing and managing B2B business processes considering an inter-organizational perspective, Kim proposes a UML 1.x based modeling approach [40]. He uses activity diagrams for modeling collaborative processes as a flow of transactions in order to create an ebXML compliant business process specification. Kramler et al. [45] use UML 2 for depicting the choreography of Web Services. In comparison to traditional business process modeling, additional requirements must be considered for service collaborations - e.g. security management or transaction management. In their paper the authors split the models into a layered architecture - collaboration, transaction, and interaction level, in order to compare these levels of granularity with the eCo framework [19].

Deployment Artifacts

The third layer of our approach corresponds to the Functional Service View (FSV) of the Open-edi model (c.f. figure 1.1). It comprises technical specifications for creating deployable artifacts, which are interpretable by machines and implement the business logic captured on the business process layer. In contemporary service-oriented architectures (SOA), process or workflow engines consume deployment artifacts in order to adapt their behavior according to the business requirements. Deployment artifacts for business processes are mostly

XML-based. A survey of different XML-based business process languages is provided in [119].

BPEL and Windows Workflow are the target platform considered for our approach

In this thesis we illustrate the deployment of UMM models using the Web Services stack as well as the Windows Workflow Foundation [1]. In terms of Web Services, we employ the *Web Service Description Language (WSDL)* and the *Business Process Execution Language (BPEL)* [77]. The former one describes the interfaces of Web Services, whereas the latter one specifies the flow of interactions between them. We further introduce each of these technologies in the respective section (7 and 8) of this thesis.

In regard to UMM, there already exists some work in the field of deriving deployment artifacts from business process models. In [30] the authors outline a proof-of-concept approach generating BPEL code from UMM. We described several shortcomings of this approach in [50]. In section 7 of these thesis, we introduce a UMM to BPEL binding, which solves these shortcomings.

WS-CDL is the W3C's effort for standardizing global choreography language

Another specification of the Web Services stack for modeling the choreography of business processes is the *Web Service Choreography Description Language (WS-CDL)* [127]. It is based on the pi-calculus and describes a collaboration between two or more peers as a global choreography by capturing the sequence of message exchanges. An agreed choreography description serves as a kind of contract between all participants of a process in order to achieve their respective business goals. Each participant is required to implement its part of the process according to the agreed choreography description. Partner specific abstract process specifications (e.g., in BPEL) may be derived from the global choreography in order to facilitate and verify local implementations. Furthermore, the choreography description can be used at design time to check the compliance of a local service implementation and at run time to determine the current state of a choreography. Therefore, WS-CDL may be characterized as both, as a modeling artifact as well as a deployment artifact. For this overview on related work, we decided to assign it rather to the deployment layer. WS-CDL has a heavy focus on reuse, i.e., the same choreography description might be reused in different geographical and industry contexts and new choreographies might be composed of existing ones.

UMM vs. WS-CDL

There are some similarities between WS-CDL and UMM: (i) both serve as a kind of contract for an agreed choreography between partners; (ii) they are not meant to be executable, but guides and facilitate the implementation of business service interfaces; (iii) both may be used to generate BPEL artifacts describing the respective business service interface by employing model transformation. A WS-CDL to BPEL binding is outlined by the authors of [63]. In terms of realizing B2B systems, WS-CDL has three major shortcomings compared to UMM: Firstly, although it is intended that WS-CDL processes are created by humans it is an XML-based language without a standardized graphical notation. Hence, tool vendors have to develop their own graphical notation for WS-CDL to provide an appropriate user interface. Secondly, it does not consider the requirements and the drivers for business collaborations, which is crucial for imple-

menting inter-organizational systems. Thirdly, WS-CDL is limited to the Web Services stack. Therefore, we opted for building our approach on the UMM instead on WS-CDL.

The XPDL should become the interchange format for BPMN diagrams

Another XML-based process description language is the XML Process Definition Language (XPDL), which is standardized by the Workflow Management Coalition (WfMC). The WfMC positions XPDL as an interchange format between different workflow tools like business process modeling and simulation tools as well as workflow engines. XPDL comes with an XML schema, which is capable of representing the structure of a business process - i.e., its control flow - as well as its corresponding graphical visualization in process modeling tools. In 2004, the WfMC endorsed BPMN and started to align the XPDL with BPMN in order to represent the full set of BPMN concepts. The WfMC envisions XPDL to become the interchange format of BPMN models between different tool vendors.

The ebXML framework was a precursor in the field of contemporary B2B approaches

Beside the Web Services stack, the *ebXML framework* would be a strong candidate for implementing B2B systems. ebXML was a joint initiative between UN/CEFACT and OASIS. Like Web Services, ebXML is a SOA-based approach, but aims to meet the specific requirements of B2B. ebXML is a distinct process-centric approach influenced by lessons learned from traditional EDI standards. The ebXML framework provides a set of five specifications: Messaging (ebMS), Registry (ebRIM/ebRS), Collaboration Protocol Profiles and Agreements (CPP/A), Core Components (CC) and Business Process Specification Schema (BPSS).

The ebXML Messaging [76] is composed of SOAP and other Web Service specifications (WS-Security [71], WS-ReliableMessaging [78], WS-Reliability [73], etc.) for realizing a secure and reliable message exchange. It defines an architecture for a B2B message service handler that may be realized by a different set of these Web Services specifications and also different versions thereof. In order to support interoperability, ebXML Messaging defines a set of conformance profiles. Furthermore, it specifies message exchange patterns for typical B2B communication scenarios. ebXML messaging provides the basic infrastructure for the communication between business partners as well as for communicating with an ebXML Registry.

An ebXML Registry corresponds to a central site for managing B2B related content. Such content may include, but is not limited to, Collaboration Protocol Profiles and Agreements (CPP/A), business processes described using the Business Process Specification Schema (BPSS), and business documents specified by means of Core Components (CC).

Collaboration Protocol Profiles (CPP) [70] allow partners to specify their party information and most notably their capabilities in terms of conducting electronic business. The capabilities denote which business processes are supported in which role (by referring to BPSS artifacts) and by which technical infrastructure. A Collaboration Protocol Agreement (CPA) [70] captures an agreement between two business partners on a certain business process and a covenant technical infrastructure. Thereby, a CPA represents a kind of intersection between the respective CPPs of both partners.

Core Components (CC) [111] represent building blocks of reusable business information in order to assemble business document types. The Core Components (CC) methodology focuses on avoiding overloaded business document types by reusing business information and reducing the number of optional fields.

Business Process Specification Schemes (BPSS) [74] capture the choreographies of collaborative business processes in a machine-interpretable manner. BPSS artifacts are means for configuring e-business systems at runtime in order to execute a certain business collaboration. ebXML does not mandate an approach to create BPSS process specifications, but recommends using UMM for collaborative process modeling. BPSS has been closely aligned to UMM by its specification as an executable subset of UMM. It provides a representation for these parts of the UMM that are required to execute a collaborative process (i.e. collaborations, transactions and exchanged documents), but omits those parts of the model not required for execution (e.g., captured business requirements).

The relationship between UMM and ebXML

It becomes obvious that there has been a strong relationship between UMM and ebXML since their beginnings. UMM was at no time part of the ebXML framework but it has been the recommended modeling framework to design an ebXML-based infrastructure. As outlined above, a BPSS corresponds to an executable subset of a UMM model. Furthermore, a UMM model captures enough information about message exchanges to generate template Collaboration Protocol Profiles (CPP). The relationship between business choreographies defined in UMM and in ebXML is examined in [3]. In addition, we introduced the algorithm of an implementation that transforms UMM models to BPSS artifacts in [36]. In this thesis, however, ebXML is not considered as a potential target platform due two reasons: (i) today, ebXML support by tool vendors is rather low and (ii) all relevant aspects of the UMM to ebXML binding have already been published.

Other Enterprise Modeling Frameworks

The need of integrated methodologies for modeling enterprise systems is not particularly new. Different frameworks have been developed to ensure a comprehensive approach to the planning, analysis, design, implementation, and governance of enterprise information systems. It is the special focus of our approach on modeling inter-organizational systems that it differs from approaches like the Zachman framework [130]. The Open Group Architecture Framework (TOGAF) [27], Sherwood Applied Business Security Architecture (SABSA) [98], the reference architectural styles for service-oriented computing as proposed in [10], as well as from various defense industry and government frameworks. Since all these frameworks are rather similar, we just highlight the concepts of the Zachman Framework which is the most commonly accepted one. It consists of a matrix, where the rows define the different participant's perspectives in building enterprise architecture (visionary, owner, designer, builder, implementer, and worker) and the columns represent the six basic interrogatives (what, how, where, who, when, and why). Each intersection contains a unique model giving an integrated view of the

enterprise which is being modeled. Each model is technology neutral and does not identify a representation language. Therefore, the Zachman approach was rather developed for analytical purposes than for developing concrete business process models. In [13] the authors propose to use a UML profile for the Zachman framework in order to overcome the shortcomings of having no well defined methodology and no concrete modeling notation.

Semantic approaches to B2B Recently, several research approaches strive for applying ideas and concepts of the Semantic Web to service-oriented architectures and business process management. The idea of a Semantic Web [5, 14] emerged from the difficulty that most information on the Internet is only understandable by humans, but not interpretable by machines [16]. This is due to the fact that the information is neither structured nor described accordingly with meta data. The idea of the Semantic Web tries to overcome these limitations by annotating information in an appropriate and machine-processable way. Semantic Web languages for describing those annotations are for example the Resource Description Framework (RDF) [126] or the Web Ontology Language (OWL) [125].

Semantic Web Services

In the field of service-oriented architectures, semantic concepts are utilized to describe functional as well as non-functional properties of services. The semantic description of a service goes beyond existing interface descriptions (e.g., by means of WSDL) and facilitates the discovery, selection, and composition of services. In 2001, the work in [61] coined the term *Semantic Web Services*. In their approach, the authors propose to annotate Web Services using the DAML family of Semantic Web markup languages. The machine-processable markup enables various agent technologies to discover Web Services for performing automated compositions. Other approaches for Web Service compositions relying on semantic descriptions are found in [62, 105, 108, 99, 100]. The idea of Semantic Web Services has been further developed in [7] with a focus on dynamic supply chain management. The authors suggest a conceptual architecture for B2B e-commerce based on the Web Service Modeling Framework (WSMF) [15]. The approach is positioned as an alternative and enhancement to the ebXML framework and the Web Services stack.

The Web Services Modeling Framework (WSMF)

The WSMF is a conceptual model that describes Web Services and their composition. According to the fundamental idea of the framework, scalable and dynamic e-commerce collaboration require two complementary principles: strong decoupling of the various components that realize an e-commerce application and a strong mediation service that enables everybody to speak with everybody in a scalable manner. In order achieve these two principles, the WSMF is based on four pillars: (i) ontologies, (ii) goals to describe the user's objectives, (iii) semantic Web Service descriptions, and (iv) mediators to solve interoperability problems. The Web Service Modeling Ontology (WSMO) [17, 129] builds upon the WSMF and further refines and extends it. Furthermore, WSMO provides a formal language for modeling Web Services - the Web Service Modeling Language (WSML) [9, 128]. Another effort for semantically describing

Web Services is the Web Ontology Language for Services (OWL-S) [57] [124], which emerged from the DAML-S language [2]. A comparison between OWL-S and WSMO is given in [46].

Some research efforts consider the above mentioned semantic approaches for working on B2B integration. In [121], the authors describe a prototype that builds upon the WSMO. The same framework is used in [120] to show how Semantic Web Services can be integrated in SAP R/3. Semantic B2B integration based on dynamic service selection is the focus of the work in [18].

The idea of combining Semantic Web Services with Business Process Management (BPM) has been developed by the authors in [29]. The resulting consolidated approach is called Semantic Business Process Management (SBPM). The authors claim that the degree of mechanization in BPM solutions is still low. SBPM is expected to overcome this limitation. The work in [28] focuses on the representational requirements for SBPM and proposes a set of ontologies to meet those requirements. SBPM has been further developed in the EU project SUPER (IST-026850) as described in [89].

3 The example: Cross-border Waste Movement in the European Union

The modeling approach proposed in this thesis is demonstrated and motivated by a real-world example taken from the domain of waste movement. In the European Union (EU) cross-border waste transports require an exchange of transport documents between the participating companies (i.e., the exporter and the importer of waste) and the local competent authorities (i.e., export authority and import authority) of the respective countries.

EUDIN employs UMM

Today, information exchange is done via paper-based documents. The EU-sponsored project EUDIN (European Data Interchange for Waste Notification Systems) aims to supersede the paper-based information exchange by electronic data interchange. Thereby, EUDIN relies on UN/CEFACT's Modeling Methodology (UMM) for specifying the inter-organizational information exchange [34].

3.1 The problem context of EUDIN

Waste transports in the European Union must be announced to several authorities

Several thousands of cross-border waste transports are conducted each year within, from, or to countries of the European Union. Although the European Union permits the free and open transfer of persons, goods, and services between their member countries, waste movement's are exempted of this settlement. In order to monitor and control waste movements, they have be announced according to the council regulation No. 259/93. The process of announcing waste transport involves the exporter and the importer of the waste, the export authority and the import authority of the respective source/-target countries, as well as the particular competent authorities of potential transit countries.

Today, the information exchange is done via paper-based forms

The monitoring and controlling process of cross-border waste movements includes the information exchange between all participating organizations about the announcement of a waste transport, the notification about the arrival of a waste transport at its designated destination, as well as notifications about the disposal of the waste. Today, this information exchange still happens paper-based. Companies have to fill out the required forms by hand and have to send them via fax or (snail) mail to all involved organizations. This results in an exchange of several thousands of paper-based forms, which must be recorded again electronically on each organization's side. In other words, data which exists electronically at organization (A) is used to fill out a paper-based form. Then, this form is communicated to organization (B), which must enter in turn the information

in it's local information system. Consequently, this cumbersomely process including many manual tasks and much media disruption is time-consuming and erroneous for both, companies and competent authorities. The goal of EUDIN is an end-to-end electronic support of the information exchange.

3.2 Requirements and goals of EUDIN

As outlined in the previous sub-section, the ultimate goal of EUDIN is the continuous electronic administration of waste movement processes spanning the information systems of all participating organizations. As being motivated by the ideas of EDI, EUDIN envisions that information should only be entered once and then conveyed electronically and processed automatically along the supply chain. In regard to the exchange of transport documents/allowances, environmental information, arrival notifications, etc., this results in a faster and less erroneous information exchange, which eventually leads to overall lower costs.

EUDIN also streamlines the announcement process

Beside the general benefits of exchanging information electronically, EUDIN lets organizations also profit from streamlined and harmonized processes. In today's paper-based information handling, organizations are responsible for distributing the filled-out forms to all other participating organizations. In other words, an exporter of waste has to announce a waste transport not only to the responsible export authority, but also to the import authority as well as to all competent authorities in transit. The same applies for the importer of the waste. In the course of switching to electronic information exchange, EUDIN further optimizes the process by having exporter and importer only communicating with an information system provided by the respective competent authority in their home country. This information system is connected to an information broker, which takes further care of the information exchange with all other involved authorities. Moreover, competent authorities of non-transit countries are in charge of forwarding the information to the importing/exporting organizations in their home country.

EUDIN focuses on the global choreography between the involved organizations

In order to be successful, it was evident from the beginning of EUDIN that a process-centric perspective is required in addition to the traditional standardization of exchanged business documents. Standardizing business documents in the sense of traditional EDI efforts is a necessary step, but not enough for realizing continuous inter-organizational information exchange. The need for a process-centric perspective to be successful becomes quickly evident if the integration scenario comprises a lot of variations of the standard communication flow, which have to be supported by electronic means as well. Therefore, the EUDIN scenario requires that all involved organizations agree on a common process choreography and integrate their information systems accordingly. In other words, successful inter-organizational integration requires an agreement on the business logic first. Gaining agreements between organizations on common business logic is facilitated by providing a neutral viewpoint on inter-organizational information exchanges. The neutral (or

global) viewpoint covers only the observable behavior between organizations, but no internal flows.

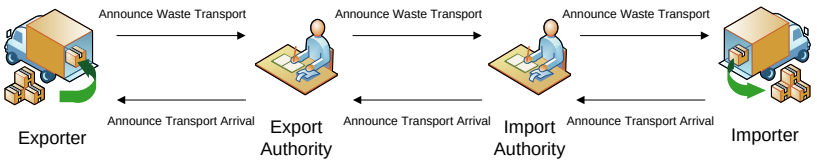


Figure 3.1
High-level overview of the information exchanges in the waste movement process

3.3 The EUDIN example in this thesis

In this thesis, we take the cross-border waste transport scenario of EUDIN as an example to motivate our model-driven approach for inter-organizational integration. For demonstration purposes, we make the following assumptions: (i) there are no transit countries - i.e., export authority and import authority communicate directly; (ii) the example does not comprise the information of waste disposal. The transport of waste is announced from the exporter to the export authority, from the export authority to the import authority, and finally from the import authority to the importer. The notification of waste arrival goes in the reverse direction. Figure 3.1 depicts a high-level overview of the information exchange in waste management process as it is used in this thesis. In section 4 the waste transport scenario is further explained from an economic perspective, whereas section 5 concentrates on the information exchanges.

4 A UML Profile for the e3-Value e-Business Model Ontology

In order to allow a straight-through modeling approach, it is desirable to base the different steps in developing inter-organizational systems on a single modeling notation. Most of the steps described above are already based on UML. This means they customize the general purpose language UML by means of stereotypes, tagged values and constraints for their specific purpose. In e^{3}value, the definition of the value exchanges is not based on UML. Thus, the definition of the UML profile for value exchanges completes our overall UML based approach for inter-organizational systems. Since the e^{3}value approach specifically targets business models in an inter-organizational environment, it is our goal to transform its concepts to a UML profile. In this section we discuss different options for representing e^{3}value in UML. A UML-based e^{3}value notation is a precondition to seamlessly integrate it with the UMM.

The remainder of this section is structured as follows: In sub-section 4.1 we introduce the e^{3}value concepts. This is followed by an e^{3}value model of our waste management example, which is also used to demonstrate the transformation to a UML profile. Sub-section 4.2 is split in five parts - each discussing a candidate solution for mapping e^{3}value to UML. Finally, sub-section 4.3 gives some concluding remarks about a UML profile for e^{3}value.

4.1 e^{3}value at a glance

4.1.1 The e^{3}value concepts

e^{3}value is an ontology-based methodology for modeling and designing business models for business networks [21] incorporating concepts from requirements engineering and conceptual modeling (including a graphical notation). Its main focus is on identifying and analyzing how value is created, exchanged and consumed within a multi-actor network, hence, taking the economic value perspective and visualizing what is exchanged (which kind of economic value) by whom [24]. An economic value exchange, and consequently the e^{3}value ontology as a whole, is based on the principle of reciprocity emphasizing the duality character of business transactions. This “give and take”-approach denotes that every actor offers something of economic value, such as money, physical goods, services, or capabilities, and gets something of economic value in return. The e^{3}value ontology de-

fines a number of concepts that will be briefly outlined in this section. The concepts are described in more detail in [21] as well as in [22].

Actors represent parties engaged in a value exchange. They are considered as independent economic entities that strive for profitability by carrying out *value activities*. These profitable or utility-increasing *value activities* are intended to be directly and unambiguously assigned to the corresponding actors.

Actors

By conducting *value activities* actors exchange *value objects* that are valuable to one or more actors of the business network. As mentioned before, these objects are "things of value" - either material, such as physical goods, products or money, or non-material (e.g. services, capabilities or experience).

Value activities and value objects

Actors signal their will to provide or request *value objects* through interfaces. In the e^{3}value ontology these interfaces are called *value ports*. The rationale of *value ports* is to abstract from an actor's internal processes and, instead, to concentrate exclusively on the external connection to other business partners (i.e. actors) and other components. Two *value ports* are connected to each other via a *value exchange*. The latter depicts one or more potential trades of *value objects* between *value ports*.

Value objects are exchanged through value ports

The values offered to or requested from the environment are represented by so-called *value offerings*, representing sets of equally directed *value ports*. The concept of *value offerings* allows the mapping of "object bundling" in case that objects can be requested or provided only in combination. This means that an offering may consist of several *value object*s to be exchanged.

Value offerings enable object bundling

Usually there are always two *value offerings* - one incoming and one outgoing - that are subsumed by a *value interface*. This typifies the give-and-take principle and shows which *value object* is offered in return for another. Each *actor* may have multiple *value interfaces* grouping individual *value ports*. The exchange of values is atomic - i.e., *value objects* in an offering are exchanged through the *value interface* on all of its *value ports* (each exchanging exactly one *value object*) or none of them.

Value interfaces realize the principle of economic reciprocity

For creating appropriate visual representations of the value models a graphical notation is provided - the stated elements are represented in figure 4.1.

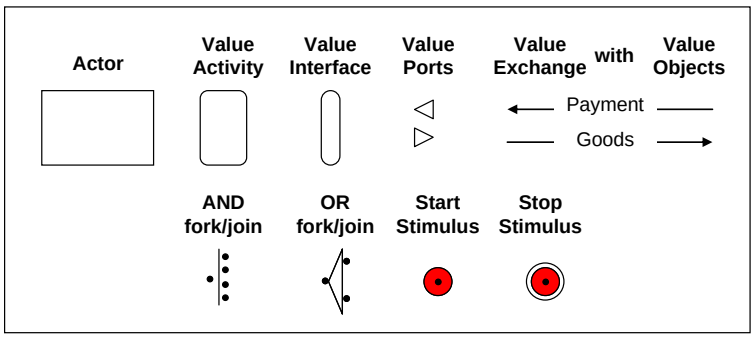


Figure 4.1
e3-Value Concepts

For mapping more complex, multi-step scenarios, components of existing scenario techniques, so-called *use case maps* (UCMs), are deployed [6]. These UCMs add four further modeling concepts: Firstly,

Scenario paths allow to model complex business scenarios

a *scenario path* shows which *value interfaces* have to exchange *value objects* as a result of a start stimulus or as the result of exchanges through other *value interfaces* [21]. Each *scenario path* is subdivided into one or more segments. Individual segments are related to each other by *connection elements*. Similarly to well-known process modeling concepts, *AND forks* as well as *OR forks* (and their corresponding *joins*) can be used to model two or more sub-paths. Furthermore, each *scenario path* starts with a *start stimulus*, representing a specific consumer need, and ends with *stop stimulus* after the last segment of the *scenario path*.

An e3-Value model may be evaluated using profitability sheets

Moreover, e^{3}value can be combined with goal-oriented modeling [26] by mapping strategic business goals of the business actors to the value model, which subsequently shows the value exchanges necessary to realize the business goals. The feasibility of a business model may be evaluated by means of profitability sheets and sensitive analysis as implemented by the native e^{3}value modeling tool. Since those profitability sheets have to coded in a tool implementation, we do not further concentrate on them in developing our modeling approach. A tool implementation of our approach may of course consider to offer a feature for generating profitability sheets.

4.1.2 The e^{3}value model for the waste management example

In this sub-section we demonstrate the e^{3}value concepts by means of our example from the waste management domain. The resulting e^{3}value model is shown in figure 4.2.

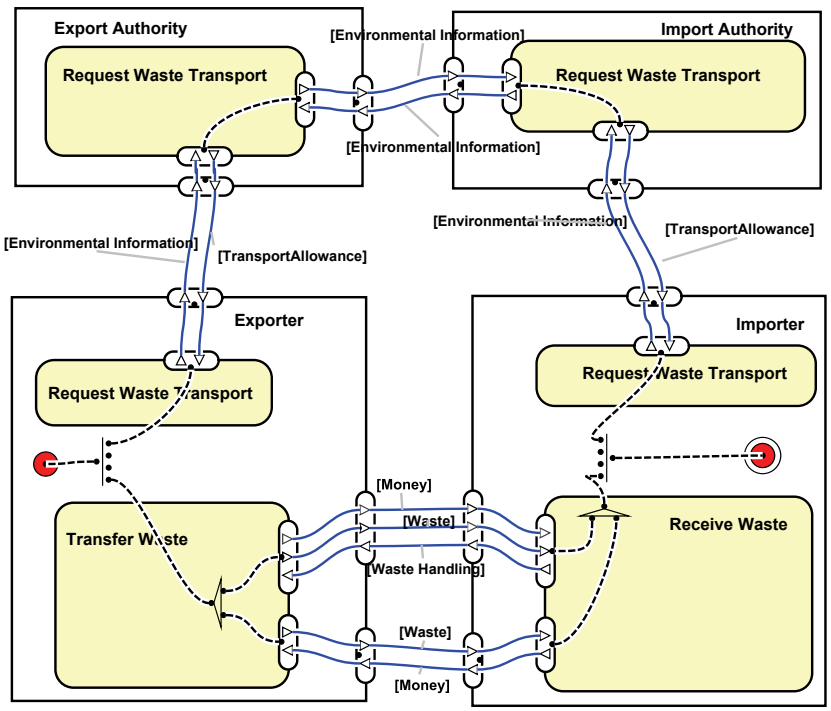


Figure 4.2
Waste management example modeled in e^{3}value

Each party - exporter, export authority, import authority, and importer - is represented by an *actor* in e³value and performs a *value activity* named request waste transport. The *actors* conduct *value exchanges* between their request waste transport *value activities* in order to fulfill the legal implications. The exporter provides environmental information to the export authority in order to get a transport allowance in return. The outgoing *value port* of the exporter's activity indicates that he provides the environmental information. Similarly, the incoming *value port* shows that the exporter demands a transport allowance in return. The atomicity of the *value exchange* is denoted by the *value interface* sitting on the edge of the exporter's request waste transport activity. It binds the two *value ports* together - indicating that either both or none of the two *value exchanges* take place. The *value exchanges* between the export authority and the import authority as well as between the importer and the import authority - as described above - are modeled in a similar way.

Exporter and importer conduct additional value exchanges

Exporter and importer perform additional *value activities* that represent the actual trading of the waste. The *value activity* transfer waste defines two *value interfaces*. The first one provides money and waste and consumes the service of waste handling in return. The second *value interface* trades waste against money. In that case, the waste has a value for the importer, because it gets recycled and then resold. The *value activity* receive waste of the importer provides the complementary *value interfaces* to the *value activity* transfer waste.

The scenario path imposes constraints on the business network

Furthermore, the concept of *scenario paths* - as depicted in figure 4.2 - shows the path by which interfaces values are exchanged. If waste is traded, the exchange of environmental information and transport allowances is mandatory. In contrary, there is no information exchange with the competent authorities required if no waste is going to be traded. It follows, that these two *value exchanges* are interlinked by an AND connector. This is denoted by the *AND fork* following the *start stimulus* located in the area of the exporter. The first path of the *AND fork* connects the *value interfaces* of the request waste transport activities - indicating that all of these *value exchanges* are required in the scenario. The second path starting from the *AND fork* represents the trading of the waste itself. We already outlined the two alternatives: either waste in return for money or waste and money in return for waste handling. These two alternatives are interlinked by an XOR connector. This is denoted by the *OR fork* preceding the two *value interfaces* of transfer waste. Note, an OR fork in e³value has an XOR semantic. The scenario paths are merged by an *OR* and *AND join* on the right hand side of figure 4.2. This means they are merged before the overall scenario is ended with a *stop stimulus*.

4.2 Mapping e³value to UML

This section focuses on mapping the e³value concepts to a UML profile. In addition to a prose description of the concepts, e³value comes with a MOF-like meta model specifying the concepts and the relations between them [24]. Furthermore, e³value defines its own

graphical notation. The MOF-like meta model is significantly different from the UML meta model. Developing a UML profile means to represent the e3 concepts on top of the UML meta model by means of stereotypes, tagged values and constraints.

Five alternatives are discussed

A mapping to a UML profile is not straightforward due to the significant differences between the e^{3}value meta model and the UML meta model. Since UML originates from software engineering, none of the UML standard diagrams has originally been intended to capture e^{3}value semantics. It is necessary to analyze which of the existing UML standard diagrams and corresponding model elements are best suited for a customization towards UML. In the following subsections we discuss five different alternatives by means of the waste management example and state their strengths and shortcomings.

4.2.1 The activity parameter variant

Value activities are a cornerstone of e^{3}value . At a first glance it seems to be consequent to map them to activities and activity diagrams, respectively. A possible solution for the waste management scenario based on activity diagrams is depicted in Figure 4.3.

The first alternative employs activity diagrams, activities, activity parameters, and actors. It is also able to describe scenario paths.

Each e^{3}value *actor* results in a UML activity partition assigned to a corresponding UML actor. The activity partition shows his area of responsibility. In the waste management example the activity diagram includes four activity partitions for the following actors: `exporter`, `export authority`, `import authority`, and `importer`. Each *value activity* is mapped to a UML activity showing the stereotype *value activity*. In the waste management example, each party performs an activity `request waste transport`. Furthermore, the `exporter` performs `transfer waste` and the `importer` performs `receive waste`.

We use UML activity parameters to model *value exchanges*. An activity parameter describes the input or output to/from a UML activity. It follows, that an offering activity carries an output parameter, whereas a consuming activity carries an input activity. The flow of a *value object* is described by an exchange from the output parameter to the input parameter. The *value object* itself is a stereotype based on the UML metaclass class. This *value object* is assigned to both, to the input parameter and to the output parameter.

To illustrate these concepts we take a look at the *value exchanges* between the `exporter` and the `export authority` on the left hand side of figure 4.3. The *value exchanges* are realized between the activities called `request waste transport` on each partner's side. The *value object* `environmental information` is assigned to the output parameter of the `exporter's activity` as well as to the input parameter of the `export authority's` activity in order to realize the *value exchange* from the `exporter` to the `export authority`. The flow of the *value object* `transport allowance` goes the other way round.

This variant is not able to reflect value ports as well as value interfaces

A major drawback of this variant is the fact, that it lacks the concepts of *value ports* and *value interfaces*. These concepts are required to group *value exchanges* for denoting the atomicity of a set of *value exchanges*. There is no concept in UML activity diagrams that corresponds to an e^{3}value *value port*. For representing *value*

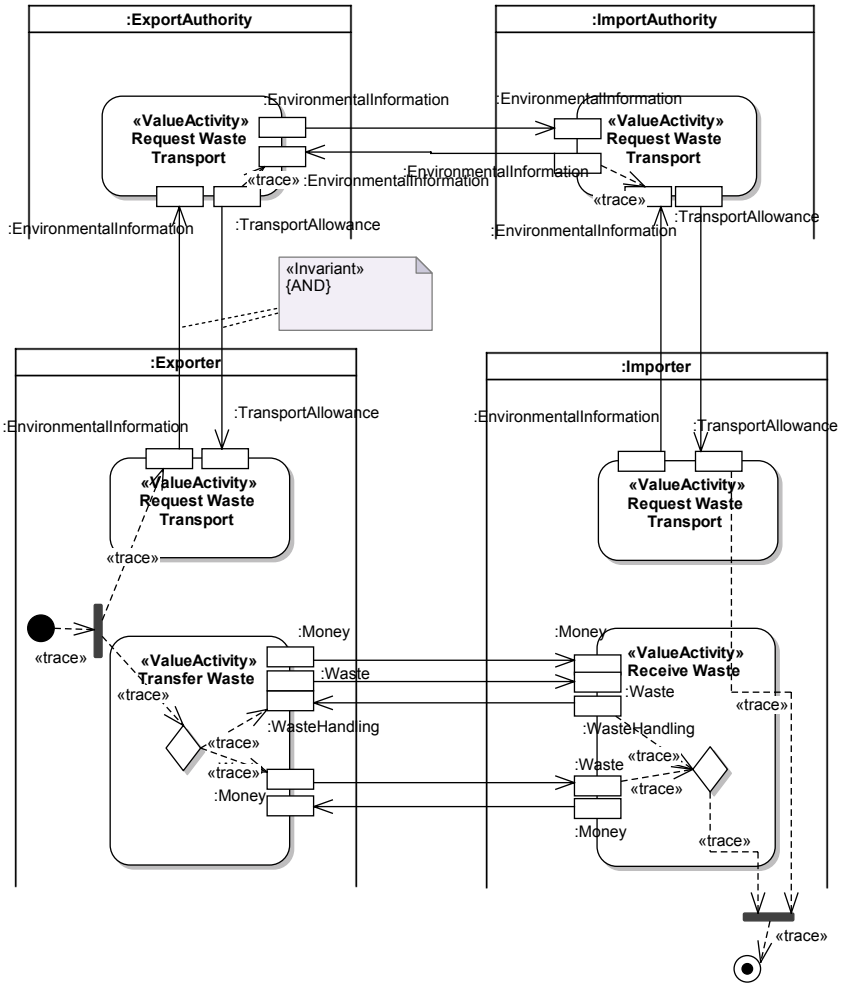


Figure 4.3
The activity parameter variant

interfaces one might think of UML parameter sets to group multiple parameters. However, the semantics of a parameter set allow to group either input or output parameters, but does not allow a mix of input and output parameters. Furthermore, multiple parameter sets of the same activity are in an XOR relationship with each other - which does not correspond to the e^{3}value semantics.

Due to these two reasons, UML parameter sets do not fit the requirements of our mapping. A workaround denoting the atomicity of two or more *value exchanges* is attaching a constraint to the respective object flows. In figure 4.3 we defined such a constraint - mandating an AND relationship between the *value exchanges* of the `exporter` and the `export authority`. For the sake of readability we refrain from showing this kind of constraint between other *value exchanges* in figure 4.3.

For mapping e^{3}value *scenario paths* we utilize the pseudo states for modeling flows in UML activity diagrams. The *start stimulus* and the *stop stimulus* of e^{3}value are mapped to UML initial and final states, respectively. The *AND fork/joins* are mapped to the UML concepts of fork/joins, whereby e^{3}value *OR fork/joins* correspond to UML decisions/merges in our mapping. In figure 4.3, the decision node within the `transfer waste` activity of the `exporter` indicates a

choice of two different scenarios: either exchanging `waste` for `money` or exchanging `waste` and `money` for the service of `waste handling`. Similarly, the fork node immediately after the initial state denotes that the `exporter` must conduct a *value exchange* with the `export authority` (`environmental information` against `transport allowance`) and a *value exchange* with the `importer` as explained above.

UML dependencies are used to model the scenario path in order to avoid the idea that it represents a control flow

It is important to note that a *scenario path* does not specify a control flow amongst the activities nor does it mandate any sequences in time. In order to stress this fact we refrain from using the UML connector type control flow. Instead, we use the concept of a *trace* - a stereotyped UML dependency - to describe the `scenario paths`. Usually a *trace* should lead to a *value interface*. Since the concept of a *value interface* cannot be represented in this variant a *trace* leads to any *value port* being part of a set of *value ports* that are graphically grouped to a *value interface*.

4.2.2 The boundaries variant

The boundaries variant is somewhat similar to the activity parameter variant. It is also based on UML activity diagrams. The representation of e^{3}value *actors* and e^{3}value *value activities* still remains the same. However, as shown in figure 4.4, *value exchanges* are modeled using the UML object node notation instead of the activity parameter notation. In other words, the exchange of a *value object* is modeled by an object flow starting from the offering activity to the *value object* modeled as object node leading to the consuming activity.

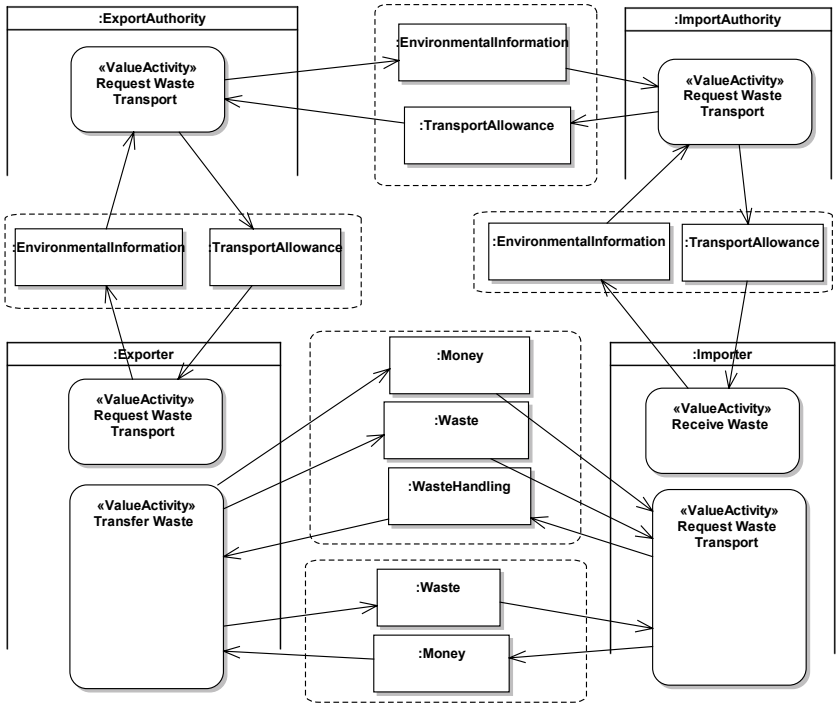


Figure 4.4 *The boundaries variant*

Although the semantics is the same as in the activity parameter variant, the object node notation allows grouping the exchanged

Value objects are represented as UML objects and grouped using interruptible regions

value objects. We use the concept of interruptible regions provided by UML for describing the atomicity of *value exchanges* being part of a value scenario - e.g., the exchange of `transport allowance` in return to `environmental information` between the `exporter` and the `export authority`. In activity diagrams, interruptible regions usually denote boundaries for exception handling. Nevertheless, we are able to use boundaries to express the semantics of economic reciprocity of two or more *value exchanges*.

Drawback: This variant is not able to describe scenario paths

The usage of interruptible regions results in a big drawback. Since interruptible regions must not be the source or the target of any UML connector, we are not able to represent *scenario paths*. Similar to the activity parameter variant, the inability to represent the e^{3}value concepts of *value ports* and *value interfaces* is a major shortcoming. Although the boundary variant benefits from compact and simple diagrams, it is inadequate for describing more complex e^{3}value scenarios.

4.2.3 The interface variant

The interface variant is also based on UML activity diagrams, but provides another solution for *value exchanges*. As shown in figure 4.5 this solution makes use of UML interfaces. A *value interface* is a stereotyped UML interface that is now used to bind *value exchanges* to an atomic unit. A *value activity* may have one to many *value interfaces* - each connected with a UML dependency.

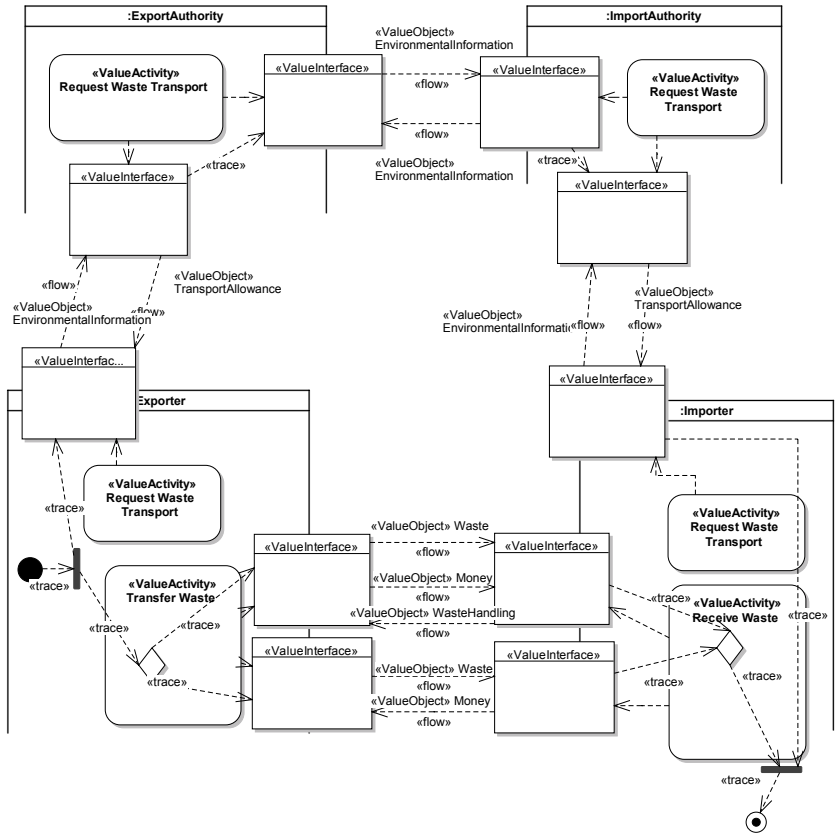


Figure 4.5
The interface variant

Value interfaces are described as UML interfaces

According to our example in figure 4.5, the exporter has a total of three *value interfaces*. One is bound to request waste transport, the remaining two are bound to the *value activity* transfer waste. A *value exchange* between two *value interfaces* is described by the UML concept of an information flow. Information flows describe circulation of information in a system in a general manner [82]. In our mapping, they are used to exchange *value objects*. The *value object* being exchanged is specified as the information flow's classifier. As shown in figure 4.5, the *value interfaces* of exporter and export authority exchange values via two information flows. One information flow sends environmental information from the exporter to the export authority in exchange for another information flow carrying the transport allowance. All classifiers of information flows are stereotyped as *value objects*.

Although there is no explicit representation of value ports this variant reflects all e³value concepts

The interface variant also provides *scenario paths* - described by similar concepts as introduced for the activity parameter variant. Although the definition of this variant lacks the explicit notion of *value ports*, it is the first mapping variant described so far that allows for modeling implicitly all e³value concepts by means of UML. In other words, *value ports* are not visualized by their own specific UML modeling element, but are in some respects represented by the concept of a *connector end* as defined in the UML meta model. However, for the sake of simplicity - and since it is not supported by most UML tools - we refrain from stereotyping the connector ends of an information flow as a *value port*.

4.2.4 The signal variant

Similar to the interface variant, the signal variant (cf. figure 4.6) covers all e³value concepts. However, it differs from the interface variant in terms of modeling *value interfaces* and *value ports*. In the signal variant, *value interfaces* are represented as stereotypes based on UML activities. *Value interfaces* are modeled as child elements of the *value activity* they belong to.

This variant uses UML signals for explicitly modeling value ports

As shown in our waste management example in figure 4.6, we place the *value interfaces* inside of the *value activities*. Unlike in the interface variant, *value ports* are modeled explicitly using the concept of UML signals. We use *send signal actions* to indicate outgoing *value ports* and *receive signal actions* for representing incoming *value ports*. Each signal action - no matter if incoming or outgoing - is stereotyped as a *value port*. As we know from e³value , *value ports* must be part of exactly one *value interface*. Correspondingly, we place the stereotyped UML signals within the *value interfaces* they are part of.

The concepts for representing *value exchanges* and *scenario paths* look alike the ones used in the interface variant. *Value exchanges* are described using information flows. The *value object* that is conveyed through the *value exchange* is assigned as the information flow's classifier. For defining the *scenario path* we apply the concept of the *trace* dependency combined with some elements borrowed from UML diagrams in order to describe *AND* and *OR fork/joins*.

UML purists may disagree with the nested activities to model value ports and value interfaces

The signal variant provides the user with all e³value concepts

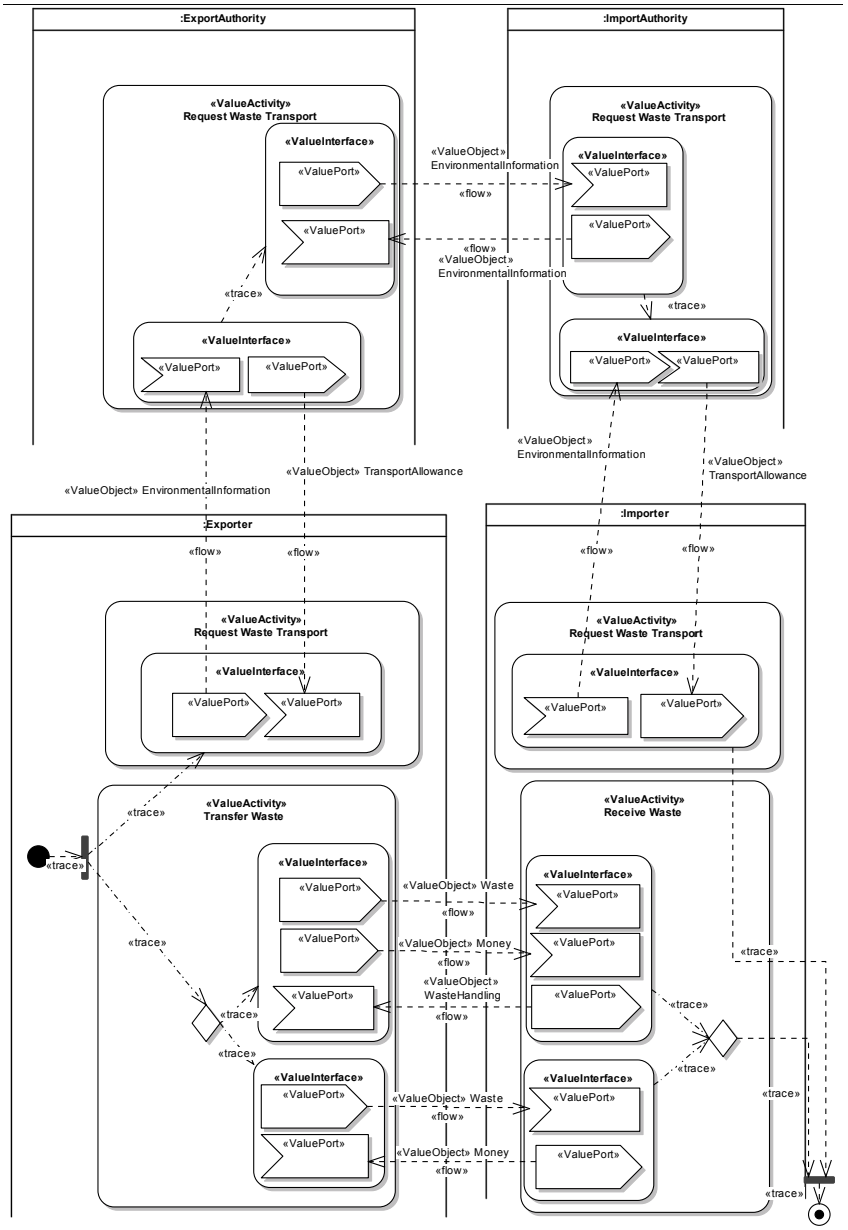


Figure 4.6
The signal variant

when using UML for business modeling. Some UML purists may disagree with the nested UML activities and actions in order to model *value ports* and *value interfaces* of a *value activity*. Those nested activities - missing start and end nodes - do not result in a well defined sequence of activity flows. However, this is a result of the different semantics of flows in e^{3}value and in UML activity diagrams.

4.2.5 The use case variant

In order to overcome the noncompliance in regard to the flow semantics between e^{3}value and UML activity diagrams, we propose another solution based on UML use case diagrams (cf. figure 4.7).

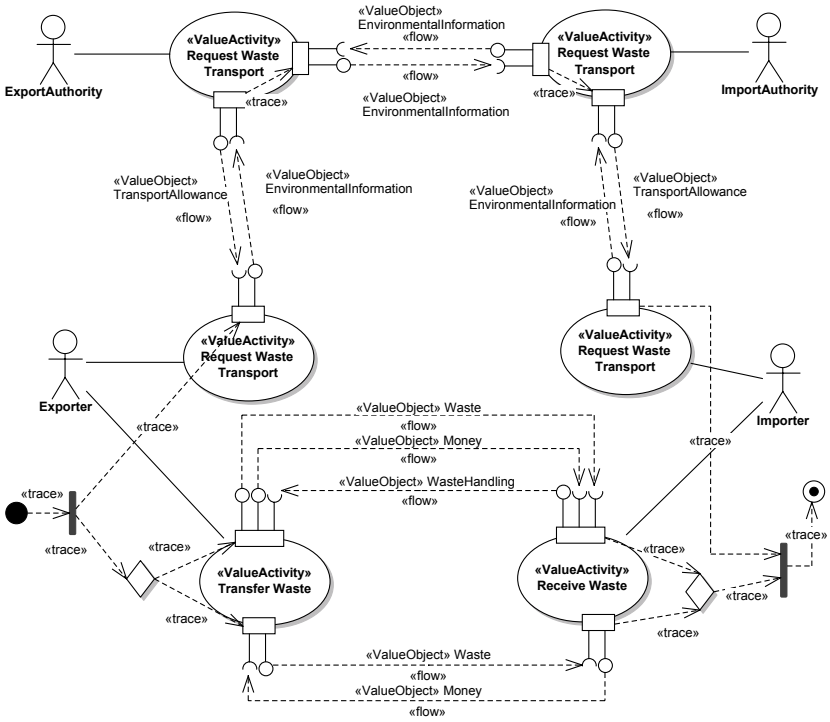


Figure 4.7
The use case variant

An e^{3}value *actor* is again represented by a UML actor. An actor may be connected to one or more use cases representing *value activities*.

The use case variant represent e^{3}value actors as UML actors and value activities as use cases

Thus, each use case gets the stereotype *value activity* assigned. Considering our example in figure 4.7, we model four *business partners* named `exporter`, `export authority`, `import authority` and `importer`. Each *business partner* is connected to his stereotyped *value activities*. For example, the `exporter` is associated with two use cases - `request waste transport` and `transfer waste` - both stereotyped as *value activity*.

Value interfaces are described as UML ports attached to a use case

In order to represent the *value interfaces* of a *value activity* we use UML ports. According to the UML2 specification, ports represent interaction points between a classifier and its environment [82]. Since the concept of a use case inherits from the concept of a classifier, a use case is eligible to have embedded ports. Each port in our mapping is stereotyped as *value interface*. In UML, each port may define zero to many interfaces. Each interface is either providing or requiring objects. Thus, the concept of a UML port interface perfectly fits the needs of an e^{3}value *value port*. Accordingly, outgoing ports become providing interfaces and incoming ports become consuming interfaces. Each port interface gets the stereotype *value port* assigned. Due to readability reasons, we do not show the stereotypes *value interfaces* and *value ports* in our example in figure 4.7.

Consider the *value activity* `transfer waste` of our example: This use case has two ports representing its *value interfaces*. One port shows the exchange of `waste` and `money` in return to `waste handling`. Thus, this port defines three port interfaces - two providing ones indicating that `waste` and `money` is transferred to the *value activity* `re-`

ceive waste and one interfaces consuming the service of waste handling. The other port of transfer waste has two interfaces - one provides waste and the other one consumes money in return.

The scenario path is described as in the signal and interface variant

The *value exchanges* between the ports and their interfaces are described like in the interface and signal variant. We denote them by information flows leading from providing interfaces to consuming interfaces. The classifier of this information flow corresponds to the *value object* sent in this *value exchange*. *Scenario paths* are again described by the concept of a *trace* dependency. We use *trace* dependencies to describe possible paths of *value interfaces* with UML pseudo states representing *AND*/*OR forks* and *joins*.

4.3 Final assessment

When mapping e^{3}value to UML we had to consider two somewhat conflicting aspects: On the one hand side, all e^{3}value concepts should also be present in a corresponding UML profile. On the other hand side, resulting UML diagrams must be easily understood by business experts with limited UML knowledge. Inasmuch these diagrams must not be overloaded. In our research work, we started off with a mapping towards activity diagrams that include value activities with input/output parameters. At first glance, this solution seemed to be the obvious choice, since the resulting diagram results in the same look and feel as e^{3}value diagrams. Unfortunately, not all e^{3}value concepts may be mapped in this solution. So we continued with developing alternative variants based on activity diagrams. However, we learned that all the proposed solutions either fail in mapping all e^{3}value concepts or result in overloaded diagrams that are hard to read for business people.

The use case variant fits best for mapping e^{3}value to UML

Finally, we based the UML profile for e^{3}value on use case diagrams. We strongly prefer this solution to the ones based on activity diagrams. It results in comprehensible business models representing the whole set of e^{3}value semantics. In addition, the use case variant is fully compliant to the UML meta model. Furthermore, e^{3}value is a method for requirements gathering and will be used as such as part of the UMM. In the UML - and also in the UMM - the concept of a use case serves as a mean for eliciting the requirements of a system. Thus, it is reasonable to model e^{3}value concepts by means of use case diagrams rather than by activity diagrams. The UML profile for e^{3}value is a perfect complement to the UMM for capturing the business justification for inter-organizational systems.

5 UN/CEFACT's Modeling Methodology (UMM): From UMM 1.0 to UMM 2.0

In this section we elaborate on UMM, which is located at the business process layer of our three-layered approach. In order to guarantee user acceptance of the UMM, it must be both effective and easy to understand for business process modelers and software architects. Due to the growing tool support of the Unified Modeling Language (UML), the decision in favor of UML as notation of UMM was made in 1998 within UN/CEFACT. In the first years, UMM specified its own conceptual meta model and provided guidelines on creating compliant artifacts using the UML. In late 2004 it was decided to define UMM 1.0 as a formal UML profile [109], i.e. a set of stereotypes, tagged values and constraints - in order to customize the UML meta model to the special purpose of modeling global B2B choreographies. At this time the UML version of choice by UN/CEFACT was UML 1.4 [81]. We lead the editing of the UMM foundation module 1.0, which became a UN/CEFACT standard in October 2006.

This section introduces new concepts for UMM on top of UML 2

We start this section by introducing the main concepts of UMM by means of the UMM 1.0 foundation module (cf. [31, 109]). First experiences in applying UMM 1.0 in real world projects have shown some shortcomings in the methodology. Accordingly, we provide an analysis of these shortcomings. One of the weaknesses of UMM 1.0 is the fact, that it is based on UML 1.4, whereas most tool vendors have moved towards UML 2 [82]. Thus, we further propose in this section how to define UMM on top of UML 2 including new concepts to overcome its current limitations. The work in this section has been contributed to UN/CEFACT and has been included in UMM 2.0.

The remainder of this section is structured as follows: Sub-section 5.2 provides a detailed explanation of UMM 1.0 and the resulting artifacts by means of our waste management example introduced in section 3. In the next sub-section we demonstrate the limitations of UMM 1.0. Finally, sub-section 5.3 tackles these limitations and shows solutions by introducing new concepts based on UML 2.

We recommend reading this section with a close look on the resulting UMM model structure in figure 5.2 and the overview of the most important artifacts in figure 5.3. We have marked both figures with numbers in black circles in order to denote where a certain artifact is sitting in the model structure. Furthermore, we make references to these numbers in the textual description of the artifacts below.

5.1 UMM 1.0 by example

UMM 1.0 comprises three main views: *business domain view* (BDV), *business requirements view* (BRV), and *business transaction view* (BTV). The latter two are split into sub-views. A UMM *business collaboration model* reflects this structure by creating packages for all these views and sub-views (See figure 5.2).

Form: *BusinessProcess*	
General	
Business Process Name	Manage end-to-end waste transport
Definition	A waste transport taking place between an exporter, an export authority, an import authority, and an importer
Description	Subject of the business process is the waste transport between different countries. The exporter pre-informs the export authority about a waste transport. The export authority forwards this announcement to the import authority, which in turn sends it to the importer. Upon receipt of the waste transport the importer informs the import authority. Now the information goes the chain backward to the export authority and to the exporter.
Participants	Exporter, export authority, import authority, importer
Stakeholder	None
Reference	Waste Management
Start/End Characteristics	
Pre-condition	A notification for the planned transport was issued.
Post-condition	- The waste has been moved from the exporter to the importer.
Begins When	Exporter informs the export authority on a waste transport
Ends When	The exporter receives the transport arrival receipt from the import authority.
Actions	- Pre-inform on waste transport - Inform on waste receipt
Exceptions	- No waste transport took place.
Relationships	
Included Business Processes	none
Affected Business Entities	WasteTransport

Figure 5.1
Example: Business process worksheet

5.1.1 Business Domain View

By modeling the business domain view, the business analyst gathers domain knowledge

The BDV is used to gather existing knowledge from stakeholders and business domain experts. In interviews the business process analyst tries to get a basic understanding of the business processes in the domain. The use case descriptions of a *business process* are on a rather high level.

Thereby, worksheets are a popular mechanism to guide the interview and to capture business know-how. Worksheets are structured forms for the elicitation of specific requirements [35]. It is important that the analyst does not influence the business expert. The interview has to take place in the language of the business domain expert; technical and modeling terms should be avoided. The interviews ensure that all involved parties share a common understanding of the business domain. In this step, the analyst discovers intra- and inter-organizational business processes as existing or desired by individual parties. A simplified example for the output of an interview kept in a worksheet is depicted in figure 5.1.

Worksheets are a popular mechanism for requirements engineering

The results of the interviews are transformed into a UML notation. Each worksheet describing a business process results in a *business process*. One or more *business partners participate* in a *business process* and zero or more *stakeholders* have an *interest in*

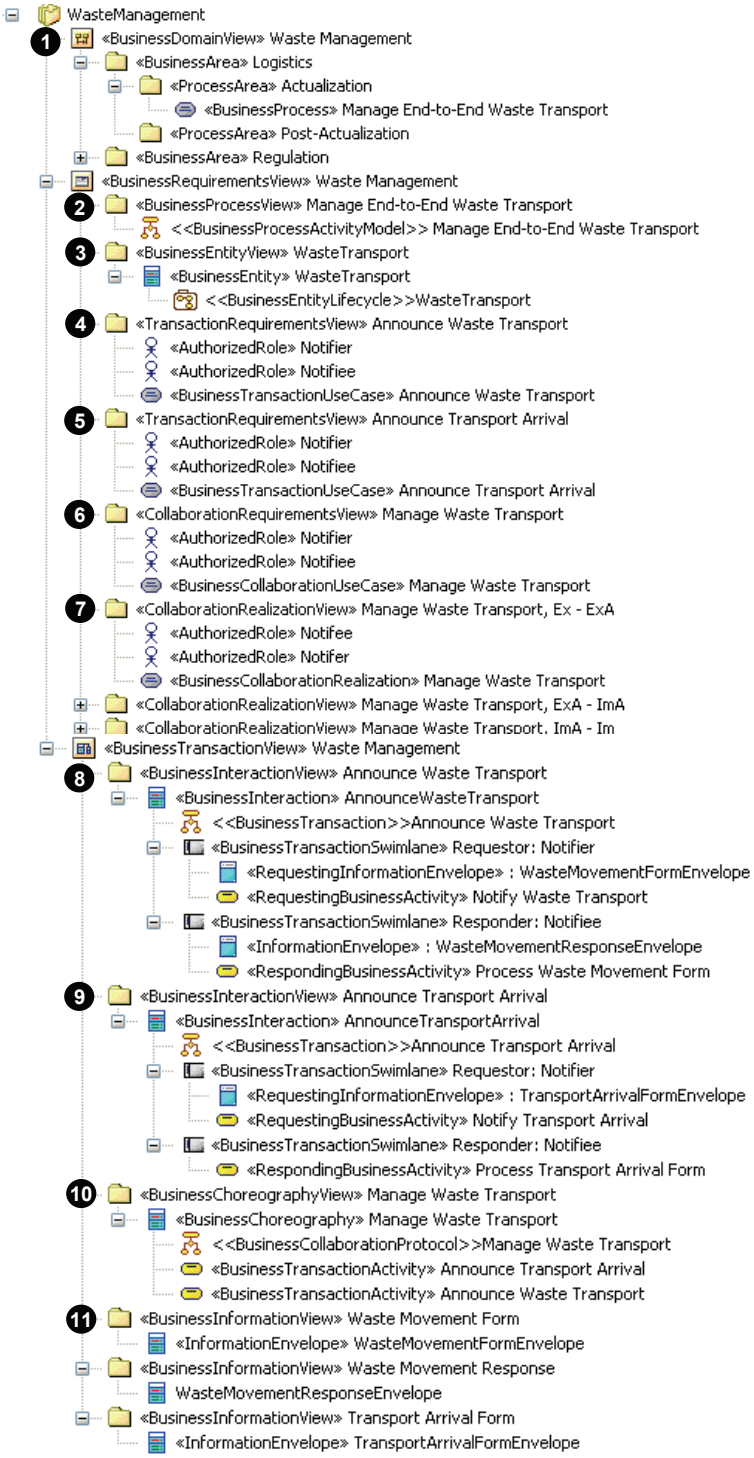


Figure 5.2
Example: Waste Management - UMM 1.0 Model Structure

dependency with the process. The BDV results in a map of *business processes*, i.e. the *business processes* are classified using *business area* sub-packages. UN/CEFACT does not mandate a certain classification schema for business processes. Appropriate taxonomies may be Porter's Value Chain (PVC) [92], UN/CEFACT's Catalog of Common Business Processes [110], or the Supply Chain Reference Model (SCOR) [104].

Each *business area* consists of *process area* packages that correspond to the Open-edi phases (planning, identification, negotiation, actualization, and post-actualization) [37]. Similar to *business areas*, this classification schema for *process areas* is not mandatory.

Example

In our waste management example relevant *business areas* are logistics and regulation, each covering at least the *process areas* of actualization and post-actualization. We do not want to detail here all the processes that may be important to the domain experts and stakeholders in these areas, but show the manage end-to-end waste transport as an example in figure 5.2 (1). This *business process* representing the multi-party process along the chain of business partners is assigned to the *process area* actualization within the *business area* logistics. In order to keep the example simple, we omit to depict other *business processes* as well as *business partners* and *stakeholders*.

5.1.2 Business Requirements View

The business requirements view gathers the requirements for the design of business collaborations

Those *business processes* from the BDV that provide a chance for collaboration are further detailed by the business process analyst in the BRV. The BRV consists of a number of different sub-views. The *business process view* (2) gives an overview about the *business processes*, their activities and resulting effects, and the *business partners* executing them by taking into account the input gathered by the respective worksheets before. The activity graph of a *business process* may describe a single partner's process, but may also detail a multi-party choreography. The business process analyst tries to discover interface tasks creating/changing *business entities* that are shared between *business partners* and, thus, require communication with a *business partner*.

In our example we illustrate the *business process activity model* of manage end-to-end waste transport (2), which further details the corresponding *business process* of the BDV. In this *business process activity model* the exporter pre-informs the export authority about a waste transport and expects an approval of the waste transport in return. In turn, the export authority announces the waste transport to the import authority to get the approval, and the import authority does the same with the importer. Later on when the waste is received by the importer, this information goes uni-directional back the chain from the importer to the import authority to the export authority, and finally to the exporter.

Business entities capture real-world things having business significance

The information exchanged between *business partners* is about the *business entity* waste transport. Firstly, a waste transport entity is created with state announced. Announced is a kind of pending state because it requires a decision by the other *business partner* to set it either to approved or to rejected. Once an approved transport has happened it is set to arrived. These so-called *shared business entity states* must be in accordance with the *business entity lifecycle* of waste transport. This lifecycle is defined in the state chart of the *business entity view* (3).

It is obvious from the requirements described so far that the announcement together with the information of approval/rejection as

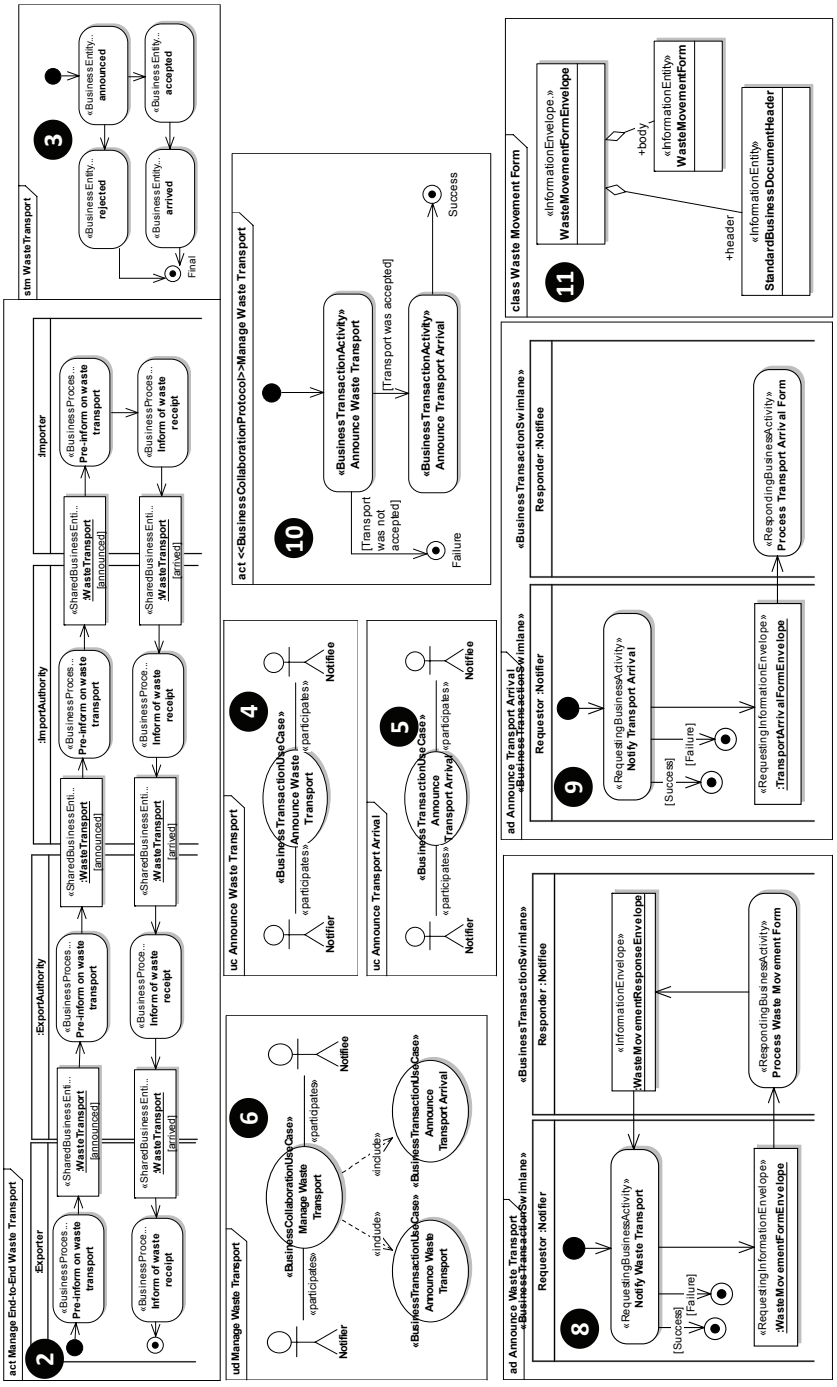


Figure 5.3
Overview of UMM artifacts using the Waste Management example

well as the information of the waste receipt always occur between a different pair of *business partners*. It is not efficient to describe these tasks for each pair again and again. Instead, these tasks are defined between *authorized roles*. A *transaction requirements view* defines the *business transaction use case* for a certain task and binds the two *authorized roles* involved. In our example we have two *transaction requirement views*: announce `waste transport` (4) - which also includes the decision - and announce `transport arrival` (5). The *authorized*

roles are in both cases a notifier who makes the corresponding announcement and a notifiee.

The concept of authorized roles and business collaboration realizations enables to execute the same tasks between different pairs of business partners

The *collaboration requirements view* includes a *business collaboration use case*. The *business collaboration use case* aggregates *business transaction use cases* and/or nested *business collaboration use cases*. This is manifested by *include* associations. In our example the *business collaboration use case* manage waste transport (6) includes the *business transaction use cases* announce waste transport (4) and announce transport arrival (5). Furthermore, the *authorized roles* participating in the *business collaboration use case* must be defined.

Sometimes it is hard to find a good name for an *authorized role*, like in our example. We call the roles again notifier and notifiee. The notifier is the one who initiates the management of a waste transport and the notifiee is the one who reacts on it. A *business collaboration use case* may have many *business collaboration realizations* that define which *business partners* play which *authorized roles*. In our example we require a *business collaboration realization* between the exporter and the export authority (7), one between export authority and import authority, and another one between import authority and importer. A detailed discussion of *business collaboration realizations* is provided in sub-section 5.1.4.

5.1.3 Business Transaction View

The business transaction view specifies a formal, global choreography of information exchanges based on the requirements gathered before

The BTV builds upon the BRV and defines a global choreography of information exchanges and the document structure of these exchanges. The choreography described in the requirements of a *business transaction use case* is represented in exactly one activity graph of a *business transaction*. A *business transaction* is used to align the states of *business entities* in the information systems of the *authorized roles*.

We distinguish one-way and two-way *business transactions*: In the former case, the initiating *authorized role* reports an already effective and irreversible state change that the reacting *authorized role* has to accept. This is the case in the *business transaction* announce transport arrival (9). In the other case, the initiating partner sets the *business entity/ies* into an interim state and the final state is decided by the reacting *authorized role*. It is a two-way transaction, because an *information envelope* flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. In the *business transaction* announce waste transport (8) the *business entity* announce waste transport is set into the interim state announced by the notifier, whereas the notifiee sets the final state of approved or rejected. Irreversible means that returning to an original state requires compensation by another *business transaction*.

A business transaction follows always the same pattern

A UMM *business transaction* follows always the same pattern: A *business transaction* is performed between two *authorized roles* that are already known from the *business transaction use case* and that are assigned to exactly one swimlane each. Each *authorized role* performs exactly one *business action*. The initiating role performs a *requesting business activity*, whereas the respondent executes a *re-*

sponding business activity. An object flow between the *requesting* and the *responding business activity* is mandatory. An object flow in the reverse direction is optional. Both, the two-way transaction announce `waste transport` (8) and the one-way transaction announce `transport arrival` (9) follow this pattern.

UMM employs the six business transaction patterns that have already proven to be useful in RosettaNet

In UMM, we further distinguish business transactions based on their business intent. There are two types of one-way transactions: If the business information sent is a formal non-repudiable notification, the transaction is *called notification*. Otherwise the transaction is known as *information distribution*.

Furthermore, there exist four different types of two-way transactions: If the responder already has the information available beforehand, it is a *query/response* transaction. If the responder does not have the information, but no pre-editor context validation is required before processing, the transaction is a *request/confirm* one. If the latter is required, the next question is whether the transaction results in a residual obligation between the business partners to fulfill terms of a contract. If so, it is a *commercial transaction*. Otherwise it is a *request/response* transaction. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi [37]. They have proven to be useful in RosettaNet [94].

Business transaction patterns are distinguished by the defaults for the quality of service parameters

The different types of business transaction patterns differ in the defaults for the tagged values (i.e., quality of service parameters) that characterize *business actions*: *is authorization required, is non-repudiation required, is non-repudiation of receipt required, time to perform, time to acknowledge receipt*, and *time to acknowledge processing*. The values for *time to perform* and for *retry count* are only defined for the *requesting business activity*.

The *is authorization required* flag mandates that the sender must sign a business document if set true. An *acknowledgment of receipt* is usually sent after grammar validation, sequence validation, and schema validation. However, if the *is intelligible check required* flag is set to false, the acknowledgment is sent immediately after receipt without any validation. An *acknowledgment of processing* is sent after validating the content against additional rules to ensure that the content is processable by the target application.

It should be noted that both kinds of acknowledgments are business signals and are not acknowledgments on the network level, like the ones required for reliable messaging. Furthermore, acknowledgments are not visualized on the activity graph of a *business transaction*. The *time to perform* parameter indicates the agreed time duration within the responder must return the responding business document. If an acknowledgment or a business document is not received in time, a time-out exception occurs. In this case it is the task of the initiator of a *business transaction* to re-initiate the business transaction according to the agreed *retry count*. If the *non-repudiation required* flag is set true, a party must not be able to repudiate the execution of the business action that inputs/outputs a business document. If *non-repudiation of receipt* is required, the receiver of a business document must send a signed receipt. Besides *business*

actions, the *information envelopes* are characterized by three security parameters to signal *confidential*, *tamper proof* and *authenticated* exchanges. Exact definitions of the respective tagged values are given in the specification [109].

According to the UMM *business transaction* semantics, the *requesting business activity* does not end after sending the envelope - it is still alive. This is a re-interpretation of the UML activity semantics, where the execution of an activity finishes when an outgoing transition is activated. The *responding business activity* may output the response which is returned to the still living *requesting business activity*. This interpretation may be curious for UML purists - however, it was already introduced by the RosettaNet [94] modeling approach and is well accepted by the e-business community.

A business collaboration protocol choreographs a set of business transaction activities. Each business transaction activity is refined by a business transaction

The requirements described in a *business collaboration use case* are choreographed in the activity graph of a *business collaboration protocol* which is defined in a *business choreography view*. In our example, the `manage waste transport` requirements (6) are mapped to the homonymous *business collaboration protocol* (10). A *business collaboration protocol* choreographs a set of *business transaction activities* and/or *business collaboration activities*. A *business transaction activity* is refined by the activity graph of a *business transaction*. In our example, the *business collaboration protocol* of `manage waste transport` (10) is a simple sequence of two *business transaction activities*: `announce waste transport` and `announce transport arrival`. Each of them is refined by its own *business transaction* (8,9). *Business transaction activities* have tagged values for a maximum *time to perform* and an indicator whether *concurrent* execution is allowed or not. *Business collaboration activities* - which are not used in our example - are refined by a nested *business collaboration protocol*. The trace between a *business transaction activity* and an underlying *business transaction* as well as between a *business collaboration activity* and an underlying *business collaboration protocol* is realized by a so-called *maps to* dependency. In the *business collaboration protocol* `manage waste transport` (10) the first *business transaction activity maps to* the *business transaction* `announce waste transport` (8) and the second one *maps to* the *business transaction* `announce transport arrival` (9).

UMM 1.0 provides no guidelines on data modeling

Finally, the information exchanged in transactions must be unambiguously defined. Each object in an object flow state is an instance of a class representing an envelope. The aggregates within this envelope are defined in a class diagram. Figure 5.2 includes a very limited extract of the class diagram for the `waste movement form` envelope (11), which is exchanged in the *business transaction* `announce waste transport` (8). UMM only mandates that an *information envelope* aggregates an *information entity* being the *header* and another one being the *body*. Usually, an *information entity* recursively aggregates other *information entities*. UMM itself does not mandate any rules on structuring *information entities*. It only suggest to base *information entities* on its Core Components Technical Specification (CCTS) [111], which is a non-UML based data modeling approach.

5.1.4 Mapping of authorized roles

One of the key goals of UMM is to foster re-use. For this purpose UMM 1.0 provides a concept called "role mapping" to re-use business transactions and nested business collaborations within different business collaborations as well as a similar concept to realize the same business collaboration between different business partners. We assume that a *business transaction use case* may be included in many *business collaboration use cases*. Consider for example, the *business transaction use case* announce transport arrival is part of another *business collaboration use case* in the logistics domain.

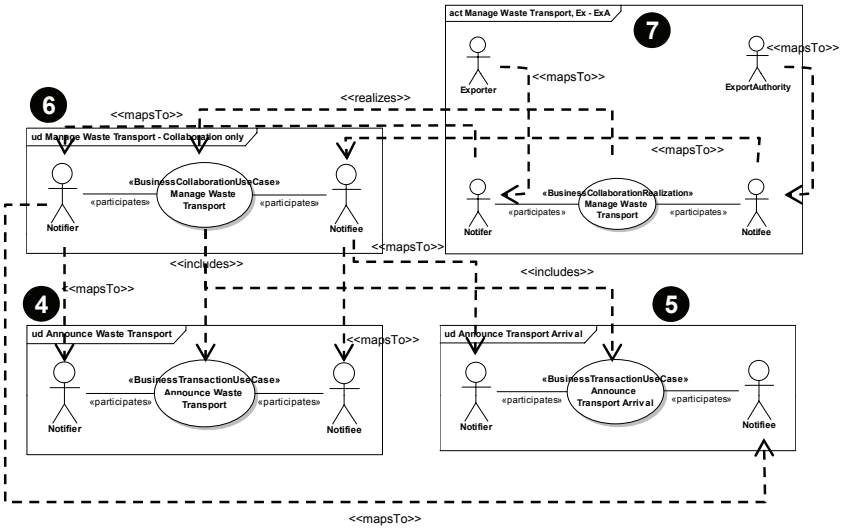


Figure 5.4 *Mapping of Authorized Roles*

UMM aspires the re-use of modeling artifacts

For the purpose of re-use, the *authorized roles* are defined in the very specific context of a *business transaction*. A *business collaboration use case* that includes the *business transaction* also defines participating *authorized roles* in its specific context. It is the peculiarity of UMM that a certain *authorized role* of the *business collaboration use case* must take on an *authorized role* of an included *business transaction use case*. For this purpose UMM uses *maps to* dependencies defining which *authorized role* of a *business collaboration use case* plays which role in an included *business transaction use case* (or nested *business collaboration use case*).

UMM uses maps to dependencies to collate roles of a transaction to roles participating in a collaboration

This concept is demonstrated by our waste management example (see figure 5.4). The *business collaboration use case* manage waste transport (6) includes two *business transaction use cases*: announce waste transport (4) and announce transport arrival (5). By coincidence, the roles of all three use cases are notifier and notifiee. However, this does not mean that a notifier of a *business collaboration use case* always maps to a notifier of a *business transaction use case*. In our example the notifier of manage waste transport (5) plays also the notifier of announce waste transport (4). However, he plays the notifiee in announce transport arrival (5) since the information flows the other way round. For the notifiee of manage waste transport it is just the opposite. Evidently, the notifier must be a different *authorized role* in each of the three use cases, however

with a homonymous name. Accordingly, *authorized roles* are always defined in the namespace of its *transaction requirements view* or *collaboration requirements view*, respectively. This is easy to recognize in the tree view of our example of figure 5.2.

Different pairs of business partners are assigned to the same business collaboration using the concept of business collaboration realizations

In the previous section we already learned that the same manage waste transport *business collaboration* must be realized between different pairs of *business partners*. Thus, we need different *business collaboration realizations* - each defining which *business partner* plays which role in it. Accordingly, our waste example results in three *business collaboration realizations* of manage waste transport - one between exporter and export authority, one between export authority and import authority, and one between import authority and importer. Figure 5.4 depicts the first one as a representative (7). The *business partners* participating in the *business collaboration realization* are the ones already defined in the BDV and, thus, are not re-defined in the namespace of the *collaboration realization view*. However, each *business collaboration realization* defines *authorized roles* which are usually - but not necessarily - homonymously named as the ones of the corresponding *business collaboration use case*. The previously introduced concept of *maps to* dependencies is used to map both the *authorized roles* from a *business collaboration realization* to a *business collaboration use case* as well as *business partners* to *authorized roles* of the *business collaboration realization*. In the manage waste transport realization (7) of figure 5.4 the exporter plays the notifier and the export authority acts as notifiee.

5.2 Limitations of UMM 1.0

After adopting the UMM 1.0 foundation module as described in the previous section, several projects were started to test the specification in real world environments. These first experiences showed a number of shortcomings of the current UMM version, which are outlined in this section.

Limitation 1 is the fact that UMM is rather vague on its guidelines on modeling the business documents. In UMM an *information envelope* being exchanged in a *business transaction* is (recursively) composed of *information entities*. UMM suggests that these *information entities* are based on the UN/CEFACT Core Components Technical Specification (CCTS) [111], which provides an ontological base of re-usable building blocks for creating shared libraries of interoperable business documents. However, the CCTS defines its own very specific meta model that is completely independent from the UML meta model. As a matter of fact, core components must be represented as (a collection of) classes and their attributes to be used in a UMM business document model. If every modeler defines his own way of mapping core components to equivalent UML classes, business documents from different projects will differ significantly even when based on the same concepts. This prohibits re-use and is in contradiction to UN/CEFACT's goal of cross-industry alignment. Conse-

quently, a well-defined representation of core components in UML is needed.

Limitation 2 is simply a result of building up on UML 1.4. We learned that a *business transaction activity* must be refined by a *business transaction*. A *business transaction* may be called by multiple *business transaction activities*. In UMM 2.0, a *call behavior action* has been introduced for exactly this purpose. However, UML 1.4 does not provide a similar concept. The UML 1.4 meta model defines a 1:n relationship between an activity and refining activity graphs. Whereas an activity may be refined by different graphs, one and the same activity graph cannot refine different activities. In order to simulate the *call behavior action* we have introduced the *maps to* dependency from a *business transaction activity* to the refining *business transaction*. This workaround leads to a proper model, but is not natively supported by UML tools and thus provides rather poor usability.

Limitation 3 is UMM's inability to model alternative responses. Currently, UMM *business transactions* follow a very strict pattern. In case of a two-way *business transaction* this pattern requires that exactly one *business information envelope* is sent in return. For example, it is always a `waste movement response envelope` that is returned in the *business transaction* `announce waste transport` (8). Instead of modeling a single `waste movement response envelope` that includes data structures for both positive and negative responses, one might prefer to define the exact data structure for a positive response and another one for a negative response. It is criticized that UMM supports the latter approach only by a workaround that is not explained in the standard's documentation: A positive `waste movement response envelope` and a negative `waste movement response envelope` have to be modeled as subclasses of a common abstract superclass `waste movement response envelope`. Furthermore, this workaround is not immediately recognized when just having a quick look on the *business transaction*. Thus, it is preferred to model different alternative responses in a *business transaction*.

Limitation 4 is the fact that a UMM choreography may be well interpreted by a human, but fails to give an unambiguous machine-processable definition to further derive software artifacts. UMM uses the concept of a *business entity lifecycle* to show the order of possible states of a *business entity*. Nevertheless, the *business entity lifecycle* is currently only used as part of the *business requirements view* in order to analyze possible *business collaborations*. In fact, the states of a *business entity* determine the flow of *business transaction activities* within a *business collaboration protocol*. However, the guards in a *business collaboration protocol* do not formally reflect the *business entity states*, but rather use plain text descriptions. In our example of figure 5.3 we show the lifecycle of the *business entity* `waste transport` (3). The *business collaboration protocol* (10) continues after `announce waste transport` with `announce transport arrival` only if the waste

transport was accepted, if it is rejected the *business collaboration protocol* ends. Unfortunately, this is defined by plain text on the guards, and does not formally reflect the *business entity states* as defined in the *business entity lifecycle*. Thus, a formal reference to the *business entity states* is desired. This requires also, that a *business transaction* properly defines which *business entity states* may be reached after its execution. Currently, the default states `success` and `failure` - like in the `announce waste transport` *business transaction* (8) in figure 5.3 - require human interpretation to what this means for the reached *business entity state*. In addition, a human is needed to decide whether the response leads to the successful or to the failing *business entity state*. It is desired to specify characteristics of a response that lead to one or the other *business entity state*.

Limitation 5 is the concept of mapping *authorized roles* as explained in sub-section 5.1.4. Although, the concept fosters the re-use of *business transactions* and provides a proper allocation between *authorized roles* taking part in a business collaboration and *authorized roles* participating in a *business transaction*, it shaped up as too complex especially for large business collaborations. This is due to the fact that each *business transaction* included in a *business collaboration protocol* requires two additional *maps to* dependencies being modeled in the corresponding use case diagram. Let's consider again figure 5.4 in sub-section 5.1.4. The *business collaboration use case* `manage waste transport` has only two *business transaction use cases* included - `announce waste transport` and `announce transport arrival`. The diagram, however, becomes already quite complex, although the business collaboration is not too large. If we think about a more complex business collaboration comprising several *business transactions* with the mapping of *authorized roles*, UMM 1.0 results in overloaded diagrams. In other words, this approach is not feasible for complex collaborations. As a consequence, stakeholders asked to replace the role mapping concept of UMM 1.0 with a more lightweight and straightforward approach in future UMM versions.

Limitation 6 is the package structure of a UMM 1.0 model as depicted in figure 5.2. It became obvious that modeling artifacts that belong together are spread over different packages. This leads to an unnecessary high number of packages. This is due to the fact that a UMM model is structured according to the steps of the methodology. However, certain artifacts created in one step and being located in the corresponding package are further elaborated in another step and the results are then located in a disconnected package in the hierarchical structure. For example, business processes are described on a rather high level by use cases in the *business domain view* in order to get a basic understanding of the domain. Business processes that seem to have a dependency on the collaboration under development are further elaborated in the *business requirements view*. The resulting activity graph is placed in the *business process view* package and thereby is disconnected from the use case that serves as a starting point. In figure 5.2 we recognize this split by the example

of the *business process* waste transport (1) and the corresponding *business process activity model* (2).

Another obvious example of splitting artifacts that logically belong together is the separation of requirements on a *business collaboration* as well as on a *business transaction* and the corresponding activity graphs of a *business collaboration protocol* / *business transaction*. Looking at figure 5.2 we recognize that the manage waste transport *business collaboration use case* (6) and the corresponding *business collaboration protocol* (10) are located in different packages. The same applies to the announce waste transport *business transaction use case* (4) and its *business transaction* (8) as well as to the announce transport arrival artifacts (5 + 9).

The splitting of UMM artifacts has also negative implications on the registration of UMM models within an e-business registry. In this case, the registry has to ensure that these dependencies are satisfied when artifacts are submitted or retrieved. In case of a submission it must check if all required packages are known - either within the same submission or already within the registry. If this is not the case the e-business registry must reject the registration. On retrieval a registry is likewise required to maintain consistency. A response on a retrieval request must contain the requested artifact plus all dependent parts.

In order to solve these shortcoming we want to avoid an unnecessary complex and overwhelming package structure. We rather intend to group artifacts which belong together into the same package. This facilitates the modeling, the comprehension, and the registration of UMM artifacts.

5.3 UMM 2.0: A proposal for new features and re-packaging

5.3.1 Business documents

In this sub-section we deal with limitation 1. We concentrate on guidelines for modeling the business documents as part of a UMM model. We consider UN/CEFACT's suggestion to base the business documents on the Core Components Technical Specification (CCTS) [111]. For this purpose the proprietary core components meta model is mapped to a semantically equivalent UML profile. We submitted this UML Profile for Core Components (UPCC) to UN/CEFACT [112]. In the following, we first describe the basic concepts of core components and then introduce the profile. For an elaborate discussion on UPCC, we refer the reader to the work in [49].

The Core Components Technical Specification (CCTS)

Core components are re-usable building blocks for assembling business documents

UN/CEFACT's goal is to provide an ontological base of re-usable building blocks for business documents. Such building blocks are referred to as core components. A core component may be re-used in different business documents, or in other words in different business contexts. Accordingly, a core component must be independent

of a business context. It must represent all business semantics that may be required by any business context. We distinguish between three different types of core components namely *basic core components* (BCC), *aggregate core components* (ACC) and *association core components* (ASCC). In order to explain the different types we give an example on the left hand side of figure 5.5. Note, CCTS does not define a graphical notation for core components. For reasons of consistency we already use the notation of UML class diagrams for representing core components as defined in our profile.

On the left hand side of figure 5.5, the *aggregate core component* `person` is shown. It has two exemplary *basic core components* `date of birth` and `first name`. *Basic core components* are atomic values being aggregated in *aggregate core components*. For modeling relationships between different *aggregate core components* so called *association core components* are used. The *aggregate core component* `person` in figure 5.5 has two *association core components* pointing to the *aggregate core component* `address` namely `work` and `private`. `Address` itself has three exemplary *basic core components* namely `country`, `postal code` and `street`. *Basic core components* have a certain data type indicated by the term separated by a colon from the name of the *basic core component*. The *basic core component* `country` for instance is of type `country code`. Data types used for *basic core components* are referred to as *core data types* (CDT).

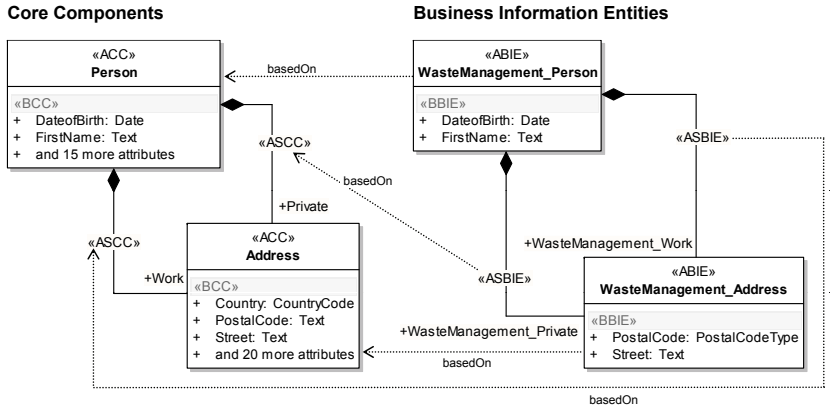


Figure 5.5
Core Components at a glance

By introducing the business context we can qualify and refine core components to the specific needs of a business document in an industry or domain. Core components used in a certain context are referred to as *business information entities* (BIE). In our example the context of the waste management domain is used.

Core Components are refined and customized for a specific purpose using the business context

Similar to the concept of core components we distinguish between three different types of business information entities namely *basic business information entities* (BBIE), *aggregate business information entities* (ABIE) and *association business information entity* (ASBIE) as shown on the right hand side of figure 5.5. The relationship among the three different types is analog to the relationship between *basic core components* (BCC), *aggregate core components* (ACC) and *association core components* (ASCC). The data type

of a *basic business information entity* can either be a *core data type* (CDT) or a *qualified data type* (QDT). Data types in *business information entities* are only refined if necessary. `Postal code type` in `waste management_ address` is an example of a refined and customized data type (QDT) which defines a code list. Since codes are represented as strings this QDT is a valid customization of the non-customized data type (CDT) `text`.

The relationship between core components and information entities is denoted by a based on dependency

Core components and business information entities have a certain interdependency. The *based on* dependencies in figure 5.5 show the relationships among core components and business information entities. It follows that *aggregate business information entities* are based on *aggregate core components*. Implicitly given by this relationship *basic business information entities* are based on *basic core components*. Likewise *association business information entities* are based on *association core components*.

The core components concept requires inheritance by restriction, which is not known in UML

Because UML does not provide a mechanism for inheritance-by-restriction no *generalization* can be drawn between core components and *business information entities*. Instead a modeler creates a new *business information entity* by taking a given core component and leaving out the attributes (BCC) and associations (ASCC) not needed. Hence a generic core component is customized to the specific needs of a business context. For example, `waste management_ address` in figure 5.5 is derived from the *core component* `address`. In order to help defining and differentiating a *business information entity* from its underlying core component and other *business information entities* we use the concept of so called *qualifiers*. In the example shown in figure 5.5 the qualifier `waste management_` is used for the *aggregate business information entities*.

A UML profile for Core Components (UPCC)

The UML Profile for Core Components introduces UML class diagrams as the method of choice for the modeling of core components. In the previous chapter, the UML class based notation of UPCC has already been used. By using a set of stereotypes, tagged values, and OCL constraints UPCC restricts the UML meta model to the specific needs of core components modeling.

The UML profile for core components closes the gap between UMM and information modeling

An example is used in order to further elaborate on the basic concepts of UPCC. As outlined in sub-section 5.1, UMM specifies the exchanged information in a business processes on a very generic basis by defining an *information envelope* with a *header* and a *body*. Underneath the body any data representation method of choice can be used. In the following example the use of UPCC for modeling the exchanged data in the waste transport domain will be shown.

The tree view on the left hand side of figure 5.6 shows the different libraries represented by packages of a core components model. For every package the according diagram is shown on the right hand side of figure 5.6. Furthermore the core components model has been simplified for explanatory purposes, e.g. not all core components are shown in the core components library.

The first package to be examined is the *PRIMLibrary* `primitive types` (A) in figure 5.6. A *PRIMLibrary* contains the very basic data

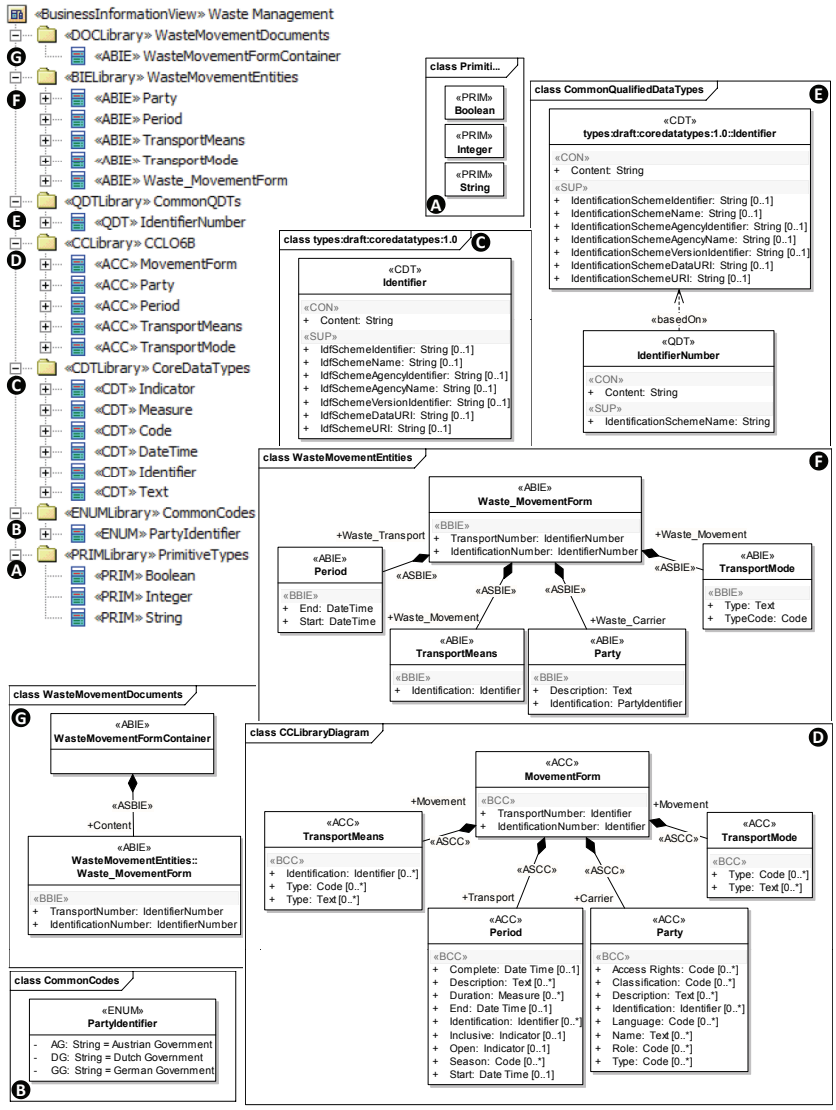


Figure 5.6
A Core Components example

types such as *Boolean*, *Integer* or *String*. In particular useful for representing code lists are so called *ENUM* types which are aggregated in *ENUMLibraries*. The *ENUMLibrary* `common codes` (B) shown in figure 5.6 contains the *ENUM* type `party identifier`. It represents a set of different governments (B).

Above the *ENUMLibrary* (B) in the tree view, the *CDTLibrary* (C) is located. A *CDTLibrary* contains the core data types. Core data types are basic data types used in *core components* and *business information entities*. In order to keep the example simple, the diagram (C) shows only one *core data type* in detail, namely `identifier`. A *core data type* consists of a *content component* stereotyped *CON* and several *supplementary components* stereotyped *SUP*. The *content component* holds the actual data of the *core data type* whereas the *supplementary components* provide additional meta information about the *content component*. The data type of the content component can either be a *PRIM* type or an *ENUM* type. In case of the core data

type identifier shown in the diagram (C), the content component is of type string (*PRIM*).

Within the *CCLibrary* CCL06B (D) the core components are stored. The main *aggregate core component* as depicted in the diagram (D) is movement form consisting of two *basic core components* transport number and identification number. The movement form has four *association core components* namely transport means, period, party and transport mode. All the *core components* in the *CCLibrary* CCL06B will serve as the basis for deriving the *business information entities* for the waste transport domain. By definition all *core components* are independent of any domain or business.

Above the *core components* in the tree view the *QDTLibrary* common QDTs (E) is located. As depicted in the diagram (E) *qualified data types* are derived from *core data types* by restriction. From the initial seven *supplementary components* of identifier only one is actually used in the *qualified data type* identifier number. *Qualified data types* are exclusively used in *business information entities*. Likewise to the concept of core data types, the data type of the content component of a qualified data type can either be of type *PRIM* or of type *ENUM*. In case of the qualified data type identifier number a *PRIM* type is used for the content component (string).

The diagrams (D) and (F) in figure 5.5 show the strong relationship between core components and information entities

Business information entities per se are stored in a package as shown by the *BIELibrary* waste movement entities (F) in the tree view of figure 5.6. If we compare the diagram of the *CCLibrary* (D) and the diagram of the *BIELibrary* (F) we again see the strong relationship between *core components* and *business information entities*. The generic *aggregate core component* movement form in (D) serves as the basis for the *business information entity* waste_ movement form in (F). In order to distinguish the *business information entities* from the *core components*, the qualifier waste_ is used. Note that the *basic core components* in (D) exclusively use *core data types* e.g. identifier for the *basic core component* transport number in movement form. *Business information entities* on the other hand can also use *qualified data types*. The *basic business information entity* transport number in waste_ movement form (F) uses the *qualified data type* identifier number.

The final business document is assembled in the *DOCLibrary* waste movement documents (G). In a *DOCLibrary* self-contained *aggregate business information entities* can be defined - in our case a waste movement form container. As outlined in the diagram (G) it has exactly one *association business information entity* namely content which is of type waste_ movement form. This waste_ movement form is taken from the *BIELibrary* waste movement entities (F) and moved to the *DOCLibrary*. It however physically remains in the *BIELibrary* and only a link is made to the element within the *DOCLibrary*. By looking at the diagram (G) this is indicated by the term waste movement entities:: waste_ movement form. The term before the colons indicates the *BIELibrary* of origin of the *business information entity* - in this case waste movement entities.

The business document as defined in the *DOCLibrary* is then finally taken and attached to the body element within the *business information view* of the UMM 2.0 model.

5.3.2 Migrating to UML 2

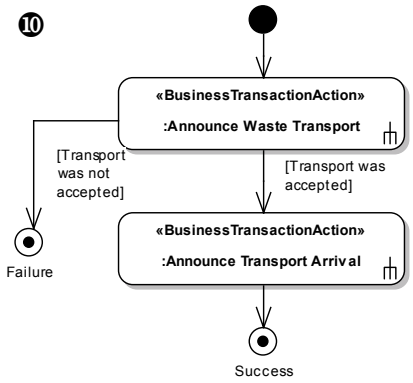
In this sub-section, we tackle limitation 2 concerning call behavior. In UML 1.4 *activity graphs* were specialized *state machines*. In UML 2 they have been replaced by the concept of an *activity*. An *activity* captures user-defined behavior by describing a flow of *actions* and their interaction with *objects* representing data. An *action* is a fundamental unit to describe a step of work within the execution of an *activity*. In the following, we propose a transition of UMM *business collaboration protocols* and *business transactions* to UML 2 using the `waste management` example.

UML 2's call behavior activity eliminates maps to dependencies

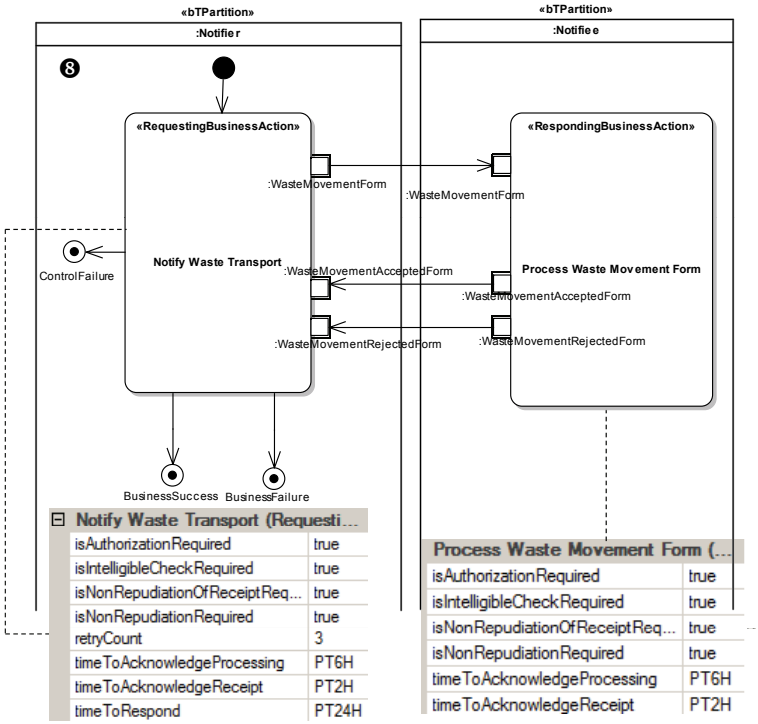
Figure 5.7 shows an excerpt of the waste transport example introduced in section 5.1. However, the example is based on UML 2 concepts now. The *business collaboration protocol* `manage waste transport` in figure 5.8(a) is composed of two interactions - `announce waste transport` and `announce transport arrival`. In UML 2, the *business collaboration protocol* becomes an *activity* and each of the two former UML activities become *actions*. Thus, we also rename the stereotype *business transaction activity* to *business transaction action*. We already know that the concept of a *business transaction action* must be refined by a *business transaction*. A *business transaction* also becomes an *activity* in UML 2. In order to refer from a *business transaction action* to its refining *business transaction* we utilize the predefined action type *call behavior action*. A *call behavior action* - indicated by the rake-symbol in the lower right corner of a *business transaction activity* in figure 5.8(a) - reflects the call of another *activity*. This eliminates the corresponding *maps to* dependency in UMM 1.0 that is based on UML 1.4. In our example in figure 5.8(a), the first *business transaction action* calls the `announce waste transport` *business transaction* and the second one calls the `announce transport arrival` *business transaction*.

5.3.3 Modeling alternative responses in UMM 2 transactions

A *business transaction* is always composed of a *requesting* and a *responding business activity* - each of them becomes a UML *opaque action*. This indicates that the "semantics of the action are determined by each partner's implementation" [82]. Due to the UML 2 nomenclature, we also rename the corresponding stereotypes to *requesting business action* and *responding business action*. Furthermore, we prefer to notate the information flows between these two actions by the new pin notation of UML 2. Figure 5.8(b) shows this new notation for the *business transaction* `announce waste transport`. An output pin of a *requesting business action* and an input pin of a *responding business action* form the flow of the request. Both pins are stereotyped as *requesting information envelope*. In our example, the request is a `waste movement form`. Thus, the `waste movement form` is



(a) *Business collaboration protocol manage waste transport*



(b) *Business transaction announce waste transport*

Figure 5.7 *Manage Waste Transport Example using UML 2*

assigned as a classifier to both pins. In analogy to the request, the combination of an output pin of a *responding business action* and an input pin of a *requesting business action* reflect a response information flow. Accordingly, those pins are stereotyped as *responding information envelope*.

A business transaction may now comprise multiple response document types

Considering the response in case of two-way transactions, we suggest an extension to the UMM transaction concept, which solves limitation 3. The current UMM transaction concept allows only one type of *responding information envelope*. Usually, the type of response differs significantly in case of a positive and a negative response. In UMM 1.0, we must use an abstract parent class for the positive and the negative response. Instead, we propose multiple output pins to *responding business activities* and multiple input pins

to *requesting business activities* in order to show different types of object flows. Alternative object flows are modeled in UML by parameter sets. If more than one parameter set is specified - even if each one includes only one parameter - these parameter sets have an XOR relationship with each other according to the UML 2 specification [82]. As outlined above, the announce waste transport *business transaction* defines the exchange of a waste movement form envelope between the *requesting business activity* notify waste transport and the *responding business activity* process waste movement form. A waste movement acceptance envelope is returned in case of an accepted transport. A waste movement rejection envelope is sent back if the transport is rejected.

5.3.4 Governing UMM Choreographies by means of Business Entity States

This sub-section proposes a solution to limitation 4 concerning no formal concept for business entity states guarding the collaboration. We learned about the interdependencies between *business collaboration protocols*, *business transactions* and *business documents*. The type of a *business document* leads either to a success or a failure of the *business transaction*. The *business collaboration protocol* uses guards on its transitions that depend on a success or failure of *business transactions*. Thus, we recognize three issues to be solved:

- The content of a response document needs to be examined to decide on the positive or negative outcome of a *business transaction*. Let's consider our UMM 1.0 example *business transaction* announce waste transport (8 in figure 5.3). Since a waste movement response envelope is returned for both, a positive or negative response, we have to look inside the document to check for the business intention of the response document. For example, the waste movement response envelope may carry a boolean attribute is accepted that needs to be evaluated by a corresponding OCL constraint (the use of OCL constraints is described later in this sub-section). A second approach to determine the outcome of a *business transaction* is based on the alternative response documents introduced in sub-section 5.3.3. Thereby, each response business document type is used to express a certain business intention. In the following, we prefer the latter approach.
- The business intention leads to a success or a failure of a *business transaction*. This means the actual business document type decides on the end state of a *business transaction*. However, there exists no formal relationship between a business document type and the semantically corresponding final state.
- The *business collaboration protocol* is guarded by the outcome of *business transactions*. However, there is no explicit reference between the transition guards and the business transaction final states.

Implicit State Changes by Business Documents

The first requirement to overcome the ambiguities in UMM is deriving the business transaction end states from the exchanged business document types. In order to bind a final state to a business document type, we use OCL constraints as transition guards.

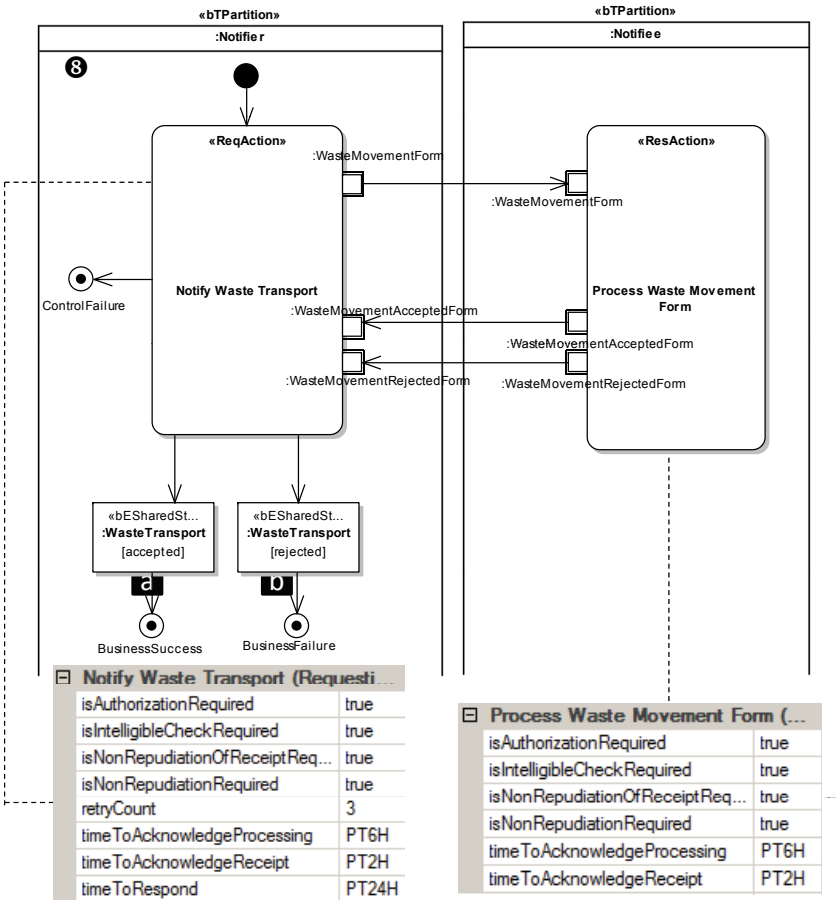


Figure 5.8
Business transaction with business entity states

Again, we demonstrate the incorporation of OCL constraints by the means of the announce `waste transport` *business transaction* and the two possible response types. We use an OCL constraint to formally represent the logic needed to determine, whether a `waste movement accepted form` or a `waste movement rejected form` was received as the *responding business document*. Accordingly, we define in listing 5.1 an invariant that checks if the received document is of type `waste movement accepted form` or not.

```
context NotifyWasteTransport
inv PositiveResponse:
  self.input->one(x | x.isTypeOf(WasteMovementAcceptedForm) and x <> null)
```

Listing 5.1
OCL constraint OCL constraints are used to determine the outcome of the business transaction based on the response document type

The OCL constraint is used to guard the transitions to final states in the announce `waste transport` *business transaction* presented in figure 5.8. The transition denoted by rectangle *a* in figure 5.8 is guarded by the OCL expression in listing 5.1 and the transition denoted by rectangle *b* is guarded by an *else* OCL expression. The *requesting business action* `notify waste transport` receives in return

either a waste movement accepted form or a ,waste movement rejected form. The notify waste transport action checks the received document to determine which of the mutually exclusive conditions holds. If the document is of type waste movement accepted form, the first invariant is valid and the *business transaction* ends with a success. Otherwise, the second condition evaluates true leading to a failure of the *business transaction*.

The mandatory use of mutually exclusive OCL guards on the business document content leads to unambiguous flows that are understood by humans and machines.

Changing Business Entity States in Business Transactions

Next, we have to reflect the results of a *business transaction* in *business entity state* changes. A *business entity* is defined as a real-world thing having business significance that might be shared among two or more business partners in a collaborative business process [109].

The outcome of a business transaction is explicitly bound to business entity state changes

In UMM 1.0, the concept of a *business entity* already exists, but just as a mechanism for requirements elicitation. Part 3 of figure 5.3 shows the lifecycle for the *business entity* waste transport. Currently, this lifecycle is not explicitly referenced by any artifacts of the *business transaction view*. The *business entity lifecycle* defines that a waste transport is at first in state announced. Later on it is set either to the state accepted or to the state rejected. This corresponds to the intention of the *business transaction* announce waste transport. By sending the waste movement form, the *business entity* waste transport is set into the interim state announced. Depending on the type of the response, the *business entity* waste transport is either set to state accepted or rejected. Although this intention seems to be obvious for humans, it is not traceable in UMM 1.0. Thus, we include the resulting final *business entity states* in the concept of a *business transaction*.

The requesting business action sets the final business entity state

In the previous sub-section we introduced an approach that allows the *requesting business action* to decide whether the *business transaction* ends with a success or with a failure. In fact, the *requesting business action* sets the corresponding *business entity state* before it reaches the final state. Consequently, we include an *object flow state* that sets the corresponding object state before reaching the final state.

Figure 5.8 exemplifies this concept by means of the announce waste transport *business transaction*. As already outlined before, the notify waste transport action checks which type of *responding business document* is actually received. If the notifiee answers with a waste movement accepted form, the *business entity* waste transport is set to the state accepted before ending with a success. Otherwise, it sets the state to rejected before ending with a failure.

A Business Collaboration Protocol guarded by Business Entity States

In the previous sub-section we bound the result of *business transactions* to *business entity states*. In this sub-section we use these *business entity states* to govern the flow of a *business collaboration*

protocol. This is realized by guarding the transitions between *business transaction actions* by the actual *business entity states*.

Business entity states govern the flow of business collaboration protocols

In the current UMM there is no formal notation mandated to specify guards. For example, the transition between announce waste transport and announce transport arrival in figure 5.8(a) is guarded by the plain text 'Transport was accepted'. This certainly reflects the result of the announce waste transport *business transaction*, or better the *business entity state* that was set by this *business transaction*. Since *business entity states* are explicitly modeled in our new approach on *business transactions*, we are able to access the actual state. In order to check the state of a *business entity*, OCL provides the function <Object>.oclInState(theState). This function returns true if the object is in the specified state. Accordingly, we mandate this function on the guards of the *business collaboration protocol*.

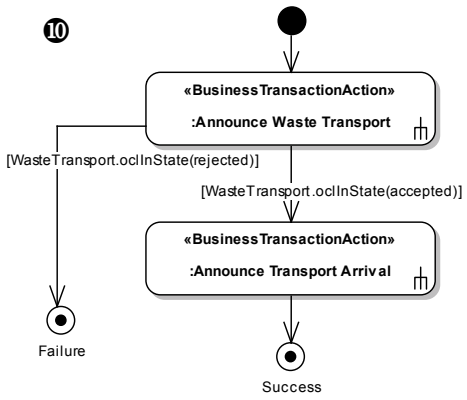


Figure 5.9
Business collaboration protocol with OCL guards

The business entity-centric approach unambiguously defines the control flow of business collaborations

This leads to the *business collaboration protocol* manage waste transport as depicted in figure 5.9. The first action in this choreography is announce waste transport. As outlined in the sub-section before, the announce waste transport *business transaction* sets the *business entity* waste transport either to state accepted or rejected. The two outgoing transitions from the announce waste transport action carry mutually exclusive OCL functions checking whether waste transport has been set to one or the other state. The *business collaboration protocol* ends if waste transport is in state rejected, and continues with announce transport arrival if waste transport is in state accepted. Announce transport arrival is a one-way *business transaction* that is executed to inform about the arrival of the waste transport. As we know, the state change communicated by a one-way transaction is irreversible. Consequently, in the business entity lifecycle of waste transport there is only one subsequent state - arrived - to the state *accepted*. Since there is no decision about the outcome of announce transport arrival, there is only one outgoing transition leading to the successful end state.

5.3.5 Introducing a new modeling approach for business collaboration protocols

This sub-section deals with limitation 5, which addresses the quite complex role mapping mechanism in UMM 1.0. For a more lightweight definition of role mappings, we propose a new modeling approach for business collaboration protocols in UMM 2.0. It is inspired by the modeling of participants in collaborative business processes in BPMN [83]. Figure 5.10 depicts the new business collaboration protocol for the manage waste transport example.

Participants are represented in a business collaboration protocol using partitions

The *business collaboration protocol* is still composed of *business transaction actions* and *business collaboration actions* (not shown in our example). In order to depict the *authorized roles* participating in a *business collaboration*, we use the concept of partitions. Exactly one partition is created for each *authorized role*. The concepts of *initFlows* and *reFlows* are used to describe, which role of the business collaboration initiates and responds in an underlying business transaction, respectively. Thereby, an *initFlow* connects a partition representing an *authorized role* with a *business transaction action* / *business collaboration action*. The same applies to *reFlows*.

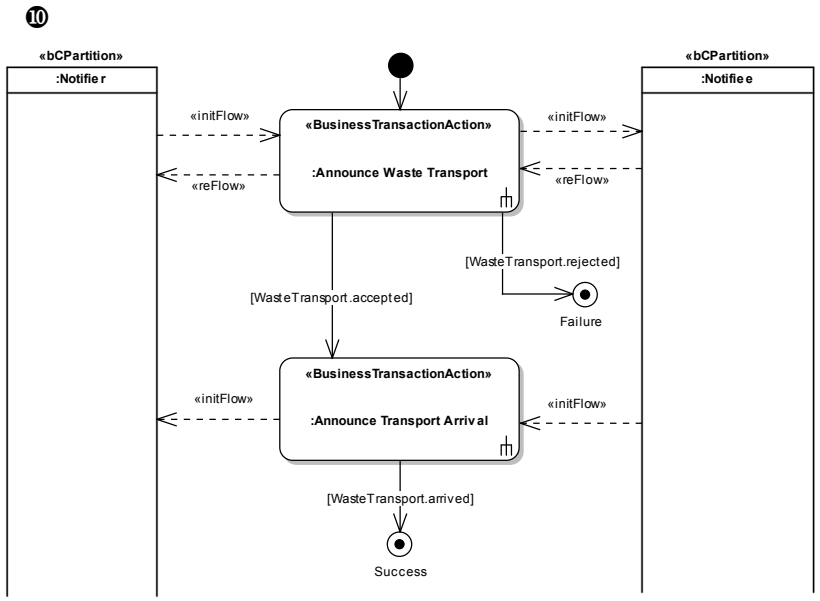


Figure 5.10
Example: Business collaboration protocol in UMM 2.0

The *business collaboration protocol* in figure 5.10 defines the exact choreography of the manage waste transport collaboration. Using the concept of two *business collaboration partitions (bCPartition)* the two *authorized roles* notifier and notifiee participating in the business collaboration are shown. The *business collaboration protocol* starts with the *business transaction* announce waste transport. The *initFlow* dependency from the notifier to the *business transaction action* announce waste transport in figure 5.10 indicates, that the notifier initiates the *business transaction*. Consequently, the notifier of the *business collaboration protocol* plays the notifier in the *business transaction* announce waste transport. The other

initFlow leads from the *business transaction action* announce waste transport to the partition of the notifiee. This declares that he plays the notifiee in the announce waste transport *business transaction*. The *reFlow* dependencies are not required for the unambiguous mapping between collaboration roles and transaction roles. However, they are used to visualize already on the *business collaboration protocol* that an underlying *business transaction* is a two-way transaction (which was a request by stakeholders).

The next *business transactions action* - announce transport arrival - points up that the mapping of roles is not always straightforward as in the case of announce waste transport. There is one *initFlow* leading from the notifiee to announce transport arrival indicating that he initiates the *business transaction*. Consequently, the notifiee of the *business collaboration protocol* plays the notifier in announce transport arrival. The *initFlow* leading from announce transport arrival to the partition of the notifier illustrates he takes up the notifiee role in the *business transaction*. Since announce transport arrival is a one-way transaction, there are no reFlows modeled.

The new approach for business collaboration protocols supersedes the role mapping concept in UMM 1.0

By introducing the new modeling approach for *business collaboration protocols* in UMM 2.0, we greatly facilitate the mapping of roles in UMM. By employing partitions for *authorized roles* in *business collaboration protocols* together with *initFlows* and *reFlows* results in two major benefits: (i) the complex mapping between *authorized roles* of *business collaborations* and *authorized roles* of *business transactions* using the *maps to* dependencies is omitted; (ii) the new *business collaboration protocol* provides a simplified visualization of who is doing what in a business collaboration to stakeholders. On the contrary, the business collaboration protocol may become overloaded compared to the old modeling style in UMM 1.0.

However, please note that there exist still maps to dependencies in UMM 2.0 for modeling *business collaboration realizations*. In other words, there are no changes to map *business partners* to *authorized roles* of a business collaboration as shown in upper right corner (7) of figure 5.4.

5.3.6 Re-packaging the UMM model structure

In this sub-section we focus on the limitation 6 concerning the splitting of related artifacts over different packages. In order to overcome this obstacle, we propose a re-packaging of UMM as outlined in figure 5.11. We argued in sub-section 5.2 that a *business process* sitting in the *business domain view* and its *business process activity model* sitting in the *business process view* of the *business requirements view* belong together. Furthermore, the business entity lifecycle in the *business entity view* is strongly related. Thus, we decided to move all these concerns into the top level package *business requirements view*.

The business process view is merged into the business domain view

The *business requirements view* now covers the *business processes* and their categorization into *business areas* and *process areas*. If a *business process* is further detailed, the resulting *business process activity model* is now specified directly beneath the *business process*. In our waste management example the *business process ac-*

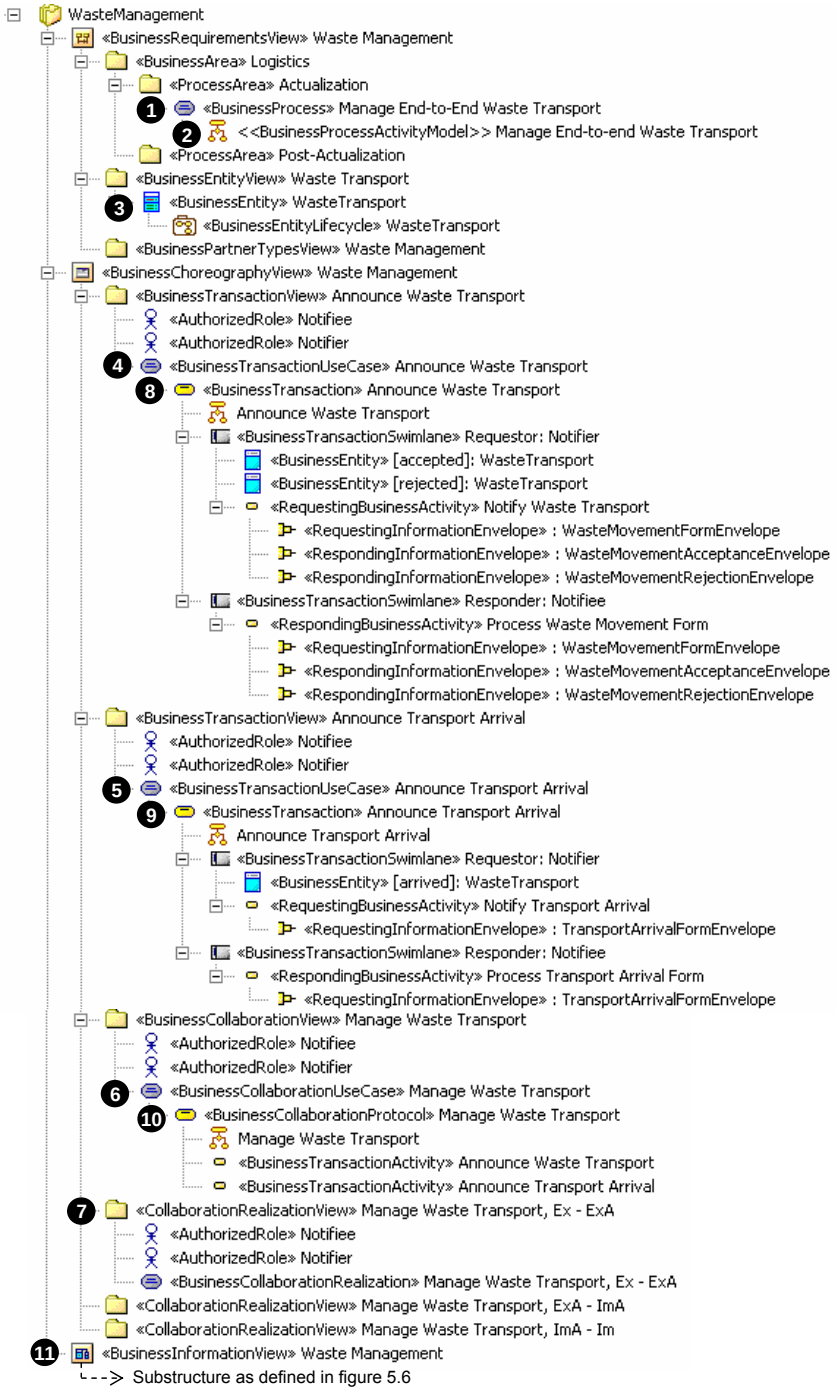


Figure 5.11
Example: Waste Management - UMM 2.0 Model Structure

tivity model `manage end-to-end waste transport` (2), is now directly added beneath the corresponding *business process* (1). Thereby, we reflect the tight binding between *business processes* and their *business process activity models*.

In addition to this structure, the BRV includes the *business entity view* capturing *business entity lifecycles* and the *business partner view* serving as a container for *business partners* and *stakeholders*. The latter was introduced to allow the modeling of organizational

The business requirements view is now composed of the business domain view, the business entity view, and the business partner types view

relationships, which we do not detail in our example. In our example the *business area* `logistics` with the *business process* and its *business process activity model* `manage end-to-end waste transport` (1+2), the *business entity view* `waste transport` (3), and the *business partner types view* `waste management` belong now to the *business requirements view*.

Business transactions and their corresponding business transaction use cases are now kept in one package forming a logical unit

In sub-section 5.2 we complained about the split of *business transaction use cases* and *business transactions*, as well as of *business collaboration use cases* and *business collaboration protocols*. The requirements and their activity model form a logical unit and should stick together. Since these concerns are about modeling a choreography, we call the second top level package *business choreography view*. The *business choreography view* consists now of *business transaction views* (which differ from the concept of the *business transaction view* in UMM 1.0), *business collaboration views*, and *collaboration realization views*.

The requirements in a *business transaction use case* and the related *business transactions* are merged into the 'new' *business transaction view*. Our *business choreography view* `waste management` contains the *business transaction views* `announce waste transport` (4+8) and `announce transport arrival` (5+9). Each *business transaction use case* (4,5) has exactly one activity beneath representing the model of the *business transaction* (8,9). The activity stereotyped as *business transaction* is described by a flow of actions visualized in activity diagrams. The activities - stereotyped as *business transactions* - are referenced by the *business transactions actions* of the *business collaboration protocol* (10) via the concept of *call behavior actions*.

The requirements of a *business collaboration use case* and the underlying *business collaboration protocol* are assembled in the *business collaboration view* based on the same principles. Accordingly, in our `waste management` example the *business choreography view* contains one *business collaboration view* called `manage waste transport` (6+10).

In order to reflect that the *collaboration realization view* gathers no requirements but realizes choreographies between concrete *business partners*, the *collaboration realization view*, originally assigned to the BRV, moved to the *business choreography view* as well. The concept of *business collaboration realizations* did not change. Consequently, the *business choreography view* `waste management` contains the three *collaboration realization views*, which we already know from (7) in figure 5.2.

The business information view becomes the third top-level view

The *business information view* (11), assigned to the BTV in UMM 1.0, becomes the third top-level view accentuating the importance of integrating process modeling and information modeling. All business documents exchanged in *business transactions* are modeled there as detailed in figure 5.6 of section 5.3.1.

By re-packaging the UMM model structure we responded to the criticism of stakeholders arguing against the complex package structure of UMM 1.0. The new suggested structure makes UMM easier to understand and simpler to use.

5.4 Final assessment

In this section we elaborated on the migration of UMM 1.0 to UMM 2.0. We started by introducing UMM 1.0, which we edited. At the time of this thesis, UMM 1.0 is the most recent UN/CEFACT technical specification (cf. section 1.4). It has been shown, however, that UMM 1.0 has some shortcomings. In this section we introduced several concepts to overcome these shortcomings. The proposed findings haven been submitted to UN/CEFACT and have been incorporated in UMM 2.0.

The first criticism of UMM 1.0 is its imprecise definition of the exchanged business documents. It only prescribes the use of an information envelope with a header and a body. However the content of the body is not specified by UMM. In this section we outlined how UN/CEFACT's core components are used to model the business documents. Because core components are specified in a way which is not compatible with common modeling tools we have introduced the UML Profile for Core Components (UPCC). The profile allows the easy integration of core components into a UML modeling tool of choice.

The second major criticism we elaborated in this paper was the refining of a *business transaction activity* by a *business transaction*. The modeling of *business transactions* being called by multiple *business transactions activities* was not possible, due to the 1:n relationship between an activity and its refining activity graphs. The concept of *maps to* dependencies in UMM 1.0 was a workaround solution providing only poor usability. Because we base our new approach on UML 2.0 we can use the concept of *call behavior actions* and abandon the workaround solution of UMM 1.0.

As a third major criticism we mentioned, that UMM does not include a proper concept for modeling alternative responses regarding the exchanged business information. Instead of modeling a single response envelope one might prefer to define two distinctive messages - one for a negative response and one for a positive response. By extending UMM transactions by object pins and parameter sets we provide a solution allowing multiple mutually exclusive response envelopes.

The unambiguous machine-processable definition of UMM choreographies has been identified as a fourth criticism of the current UMM version. The guards in a *business collaboration protocol* do not formally reflect the *business entities states* and, therefore, no automated processing is possible. In this section, we presented a formal definition of the guard conditions that reflect the business entity state changes realized by exchanging documents in a *business transaction*.

The fifth limitation was that the so-called role mapping concept in UMM 1.0 is too complex. This results in overloaded use case diagrams showing the relationships between roles of a *business collaboration protocol* and roles of a *business transaction*. We introduced a new modeling approach for *business collaboration protocols* which makes it possible to drop the current role mapping concept. Furthermore, the new approach to *business collaboration protocols* provides

a comprehensive overview for stakeholders, who is interacting in a *business transaction* and whether a *business transaction* comprises an uni-directional or bi-directional information exchange.

The last criticism identified is the package structure of UMM 1.0 which has turned out to be inefficient. Modeling artifacts logically belonging to each other are spread over different packages. We have introduced a new package concept facilitating the governing of inter-dependent modeling artifacts. In particular the split of requirements on a business collaboration as well as on a business transaction and the corresponding activity graphs of a *business collaboration protocol / business transaction* has been eliminated.

6 A State Machine for UMM Business Transactions

UMM comprises three main views for describing a computation independent model from a neutral perspective. However, UMM is currently missing a platform independent model showing how each partner has to realize the message exchanges to support the agreed choreography. In this section we derive such a platform independent model from a UMM *business transaction* - a key artifact of the computation independent model. The resulting model is based on a state machine describing the local view of a participating business partner. This state machine unambiguously defines how a business partner has to react on incoming messages and on message expected but not received.

The model-driven architecture approach of the OMG comprises three types of models

The model driven architecture (MDA) approach of the OMG distinguishes three different views of a system that results in different kind of models [80]: (1) The computation independent viewpoint focuses on the environment of the system and its requirements. The details of the structure and processing of the IT system are unspecified. It leads to a *computation independent model* (CIM) that is familiar to the practitioners of the domain under consideration who do not need to care how to realize the functionality of an IT system. (2) The platform independent viewpoint focuses on the operation of an IT system while hiding the details necessary for a particular platform. It leads to a *platform independent model* (PIM) exhibiting a certain degree of platform independence in order to be suitable for a number of different platforms of similar type. (3) The platform specific viewpoint extends the platform independent viewpoint with an additional focus on the details of using a specific platform by an IT system. It leads to a *platform specific model* (PSM) limited to a particular platform.

Our approach misses the platform independent model

Traditionally, the MDA approach focuses on the development of application systems and their code. However, a MDA approach may also be used in developing inter-organizational systems describing how autonomous applications of different organizations have to interact. This is also in accordance with the Open-edi reference model [37]. In MDA terms the BOV is both, a computation independent model and a platform independent model. The FSV is clearly a platform specific model. In terms of our approach the global UMM model forms the computation independent model, whereas two implementation platforms of our three-layered approach - Web Services (WSDL and BPEL) and the Windows Workflow Foundation (WF) - make up the platform specific model. It follows that we miss the platform in-

dependent model "in between" to complement our model-driven approach.

6.1 Mapping business transactions to state machines

In the remainder of this sub-section we show the creation of a platform independent model by transforming the global choreography of UMM *business transactions* to a local choreography describing the message exchanges of business documents and business signals.

UMM business transaction abstract from the complexity of a business service interfaces

As we know, UMM *business transactions* are described by a rather simple pattern. This is also due to the fact that the activity graph shows only the exchange of business documents, whereas the business signals representing acknowledgments are described by tagged values. Thus, the local choreography of a partner's interface is not only a sequence of sending and receiving business documents. It requires a more complex choreography that has to reflect an exchange of business documents leading to well-defined business states even in case of failures. In this sub-section we propose to describe this complex choreography by means of a state machine that acts on incoming and outgoing messages. We use a state machine, because it is best suited to describe the valid states of a business partner interface and the events causing state transitions - finally leading to a success or failure of the business transaction.

The actions to be carried out by a business partner interface mainly depend on the instantiation of the tagged values of *requesting/responding business actions* (cf. section 5.1.3). This instantiation depends on the UMM transaction type. We do not elaborate state machine representations for each of the six transaction types. Instead, we demonstrate the business partner interface by the most complex pattern - the commercial transaction. In our waste management example, the commercial transaction pattern is employed for the `notify waste transport` *business transaction*, since it results in a legally binding agreement for the transport of waste. Figure 6.1 depicts again the `notify waste transport` transaction used to demonstrate the state machine transformation.

The state machine of the `notifier` is discussed in the following sub-section. Then, sub-section 6.3 outlines the notifiee's state machine by showing the differences between those two. Finally, subsection 6.4 discusses the composition of multiple state machines to reflect a business collaboration.

6.2 The notifier's state machine (initiating party)

In a UMM *business transaction* each role must evidently implement its own business partner interface. It follows, that a UMM *business transaction* results in two state machines each describing a business partner interface. Figure 6.2 shows the system specification for the

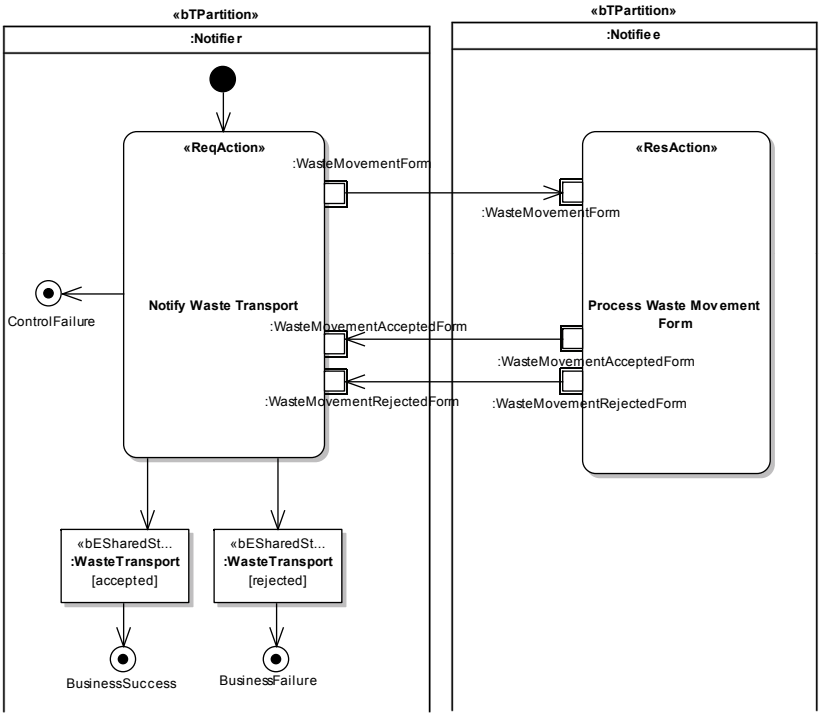


Figure 6.1
Notify waste transport business transaction

notifier's part of notify waste transport. One should note, that due to readability we shortened some of the identifiers in the figure (e.g., *AckReceipt* instead of *acknowledgment of receipt*, etc.).

A *business transaction* is a unit of work represented by a composite state. According to our example, it is started when the business partner interface receives the waste movement form from the business application. It may be re-initiated due to time-out exceptions. This means that the initiator has to restart the *business transaction* if an *acknowledgment of receipt/processing* or a awaited response document is not received in time. The maximum number of restarts is defined by the *retry count*. Thus, the first state is *checkRetryCount*. Its activity *checkIfRetryCountLeft* audits the remaining retries to re-initiate the transaction in case of a previous failure. If the *retry count* is equal or greater than zero the system proceeds with the transmission of the request.

If the no retries are left, the business transaction ends with an exception

In case of re-initiating the *business transaction* with no remaining retries the system transitions to state *sendFailedBusinessControl_Outgoing*. In this state, the system issues a notification of failed business control to the partner's system and exists the transaction with a failure. If retries are left, a signal event *RetryLeft* fires the transition to state *sendWasteMovementForm_Outgoing*. Within this state the notifier transmits the waste movement form by invoking the operation *processWasteMovementForm(form)* offered by the notifiee (note: the parameter *form* of *processWasteMovmentForm* corresponds to an instance of the waste movement form).

The notifier *waits for an acknowledgment of receipt after sending the* waste movement form

After sending the document we reach the state *waitForAckReceipt_Incoming*. In this state the notifier expects that the notifiee acknowledges the receipt of the waste movement form sent be-

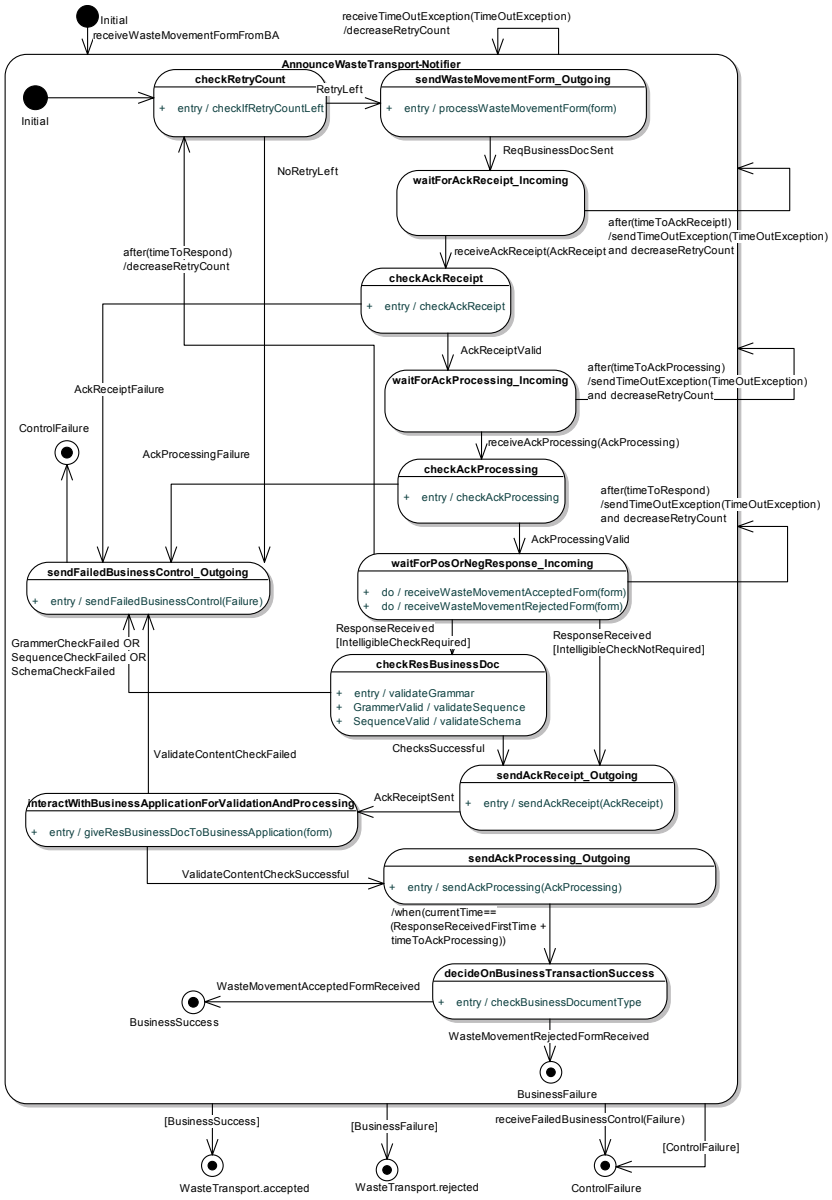


Figure 6.2
State machine showing the notifier's part of `notify waste transport`

fore. The system remains there until either of the following two events occurs: (1) If the `notifier` receives no acknowledgment within the agreed *time to acknowledge receipt*, a time event specified by *after(timeToAcknowledgeReceipt)* eventuates and executes an activity that issues a time-out exception and decreases the available retries. The transition activated by the time event re-enters the composite state of the *business transaction*, which begins again with *checkRetryCount*. (2) The notifier receives the *acknowledgment of receipt*. This results in the call event *receiveAckReceipt(AckReceipt)*.

The business service interface must check received acknowledgments

If the *acknowledgment of receipt* is picked up, the notifier enters the state *checkAckReceipt*. The activity *checkAckReceipt* checks the acknowledgment's content. A failure yields to an *AckReceiptFailure* event and the system transitions to *sendFailedBusinessControl_Out-*

going. Otherwise, a successful check produces an *AckReceiptValid* event that leads to *waitForAckProcessing_Incoming*.

An acknowledgment of processing is handled like an acknowledgment of receipt

This state works like the state *waitForAckReceipt_Incoming* described above, except that the business partner interface waits for an *acknowledgment of processing*. Again, if the notifier's system does not receive the acknowledgment in the agreed time, a time-out exception is issued, the available retries are decreased, and the composite state is re-entered to check the retries. If the acknowledgment is received and the applied checks are successful the notifier enters the state *waitForPosOrNegResponse_Incoming*. Otherwise, if the checks fail an *AckProcessingFailure* is actuated and the system switches to *sendFailedBusinessControl_Outgoing*.

The `notifier` *may either receive a positive or negative response*

In this step of the *business transaction*, the `notifier` waits for either a `waste movement accepted form` or a `waste movement rejected form`. The `notifier's` business service interface offers a corresponding service operation for each possible business document type - indicated by two activities *receiveWasteMovementAcceptedForm* and *receiveWasteMovementRejectedForm*. The activities denote that the service interface is continuously listening in order to receive one of the two business documents. Not receiving either business document within the agreed time (i.e., *time to respond*) causes the initiator to issue a time-out exception, to decrease the available retries, and to re-enter the composite state to check the retries. Otherwise, receiving a business document actuates the signal event *ResponseReceived*.

Received business documents have to pass grammar, sequence, and schema validation

This signal event is specified for two outgoing transitions with mutually exclusive guards concerning an intelligible check of the document. If an intelligible check is required the system enters state *checkResBusinessDoc*. In this state the document has to pass through a grammar, sequence and schema validation prior to issuing a proper *acknowledgment of receipt*. The grammar validation secures that the document's syntax is processable by the system. The sequence validation assures that the document is received in the proper message sequence according to the document flow. Finally, the schema validation checks the received document against its associated schema(s). If any of the above mentioned checks fails, the `notifier's` system goes immediately to state *sendFailedBusinessControl_Outgoing*. If all checks are passed or if the intelligible checks were not necessary, the system switches to *sendAckReceipt_Outgoing*. Being there, the `notifier` confirms the proper pick up of a responding business document by sending an *acknowledgment of receipt* as indicated by the activity (*sendAckReceipt(AckReceipt)*).

The business application checks the business document against further business rules

Subsequently, in the state *interactWithBusinessApplicationForValidationAndProcessing* the business partner interface hands the business document over to the business application. It validates the document against further business rules. If successful, the business document is processed accordingly. If validation of the business document's content fails, the business application issues a *validateContentCheckFailed* signal and the business service interface changes its state to *sendFailedBusinessControl_Outgoing*. Otherwise, in the case of success, the business service interfaces reaches the state *sendAckProcessing_Outgoing* and sends an *acknowledgment of processing* to

indicate the successful content validation. After executing this action, the notifier's business partner interface has to keep the business transaction alive until no failure messages may be received anymore. This time ends when the *time to acknowledge processing* has passed after sending a responding business document. Until this time the notifiee - as the responder within the transaction - may still signal a time-out exception or a failed business control.

The response business document type determines the success or failure of a business transaction

Finally, as the very last step in the *business transaction*, the business service interface checks in state *decideOnBusinessTransactionSuccess*, which of the two response business document types is received in order to determine if the *business transaction* is a business success or not. If a waste movement accepted form is picked up, the *business transaction* is a *business success*, which corresponds to the state accepted of the *business entity* waste transport (cf. figure 6.1). Otherwise, the *business transaction* fails resulting in the *business entity state* WasteTransport.rejected. As we learned in section 5.3.4, the *business entity states* are used to govern the further flow in a *business collaboration protocol* based on the outcome of the conducted *business transactions*.

The business service interface must be able to receive exception messages at any time during the transaction

Both failure messages may not only be received after sending the *acknowledgment of processing*, but at anytime, when the business partner interface resides in the (overall) composite state representing the *business transaction*. A time-out exception is a signal by the responder that he has not received a message within the expected time. This requires to re-initiate the *business transaction*. Accordingly, if the call event *receiveTimeOutException(TimeOutException)* occurs, the business partner interface decreases the *retry count* and re-enters the composite state of the *business transaction*. A failed business control is received if the responder is not able to process a message correctly. Thus, the call event *receiveFailedBusinessControl(Failure)* terminates the *business transaction* with a failure.

6.3 The notifiee's state machine (responding party)

The state machine of the responding party in a *business transaction* must be complementary to the one of the initiating party. The resulting state machine for the notifiee in our example is depicted in figure 6.3. The concepts used in this state machine are very much similar to the ones of the notifier's one. However, the order of receiving and then sending business documents is reversed including the handling of acknowledgments. Due to the similarity, we do not explain all the states in detail.

The responder's business service interface does not need to care about the retry count

The major difference is that the *retry count* is not controlled by the notifiee. This means, when an *acknowledgment of receipt* or *processing* is not received in the expected duration a time-out exception is issued. Then, the composite state is re-entered, but no retry counter is decreased. By re-entering the composite state, the notifiee's business partner interface is waiting for the waste movement form. The notifiee does not need to care about the *retry count*. If

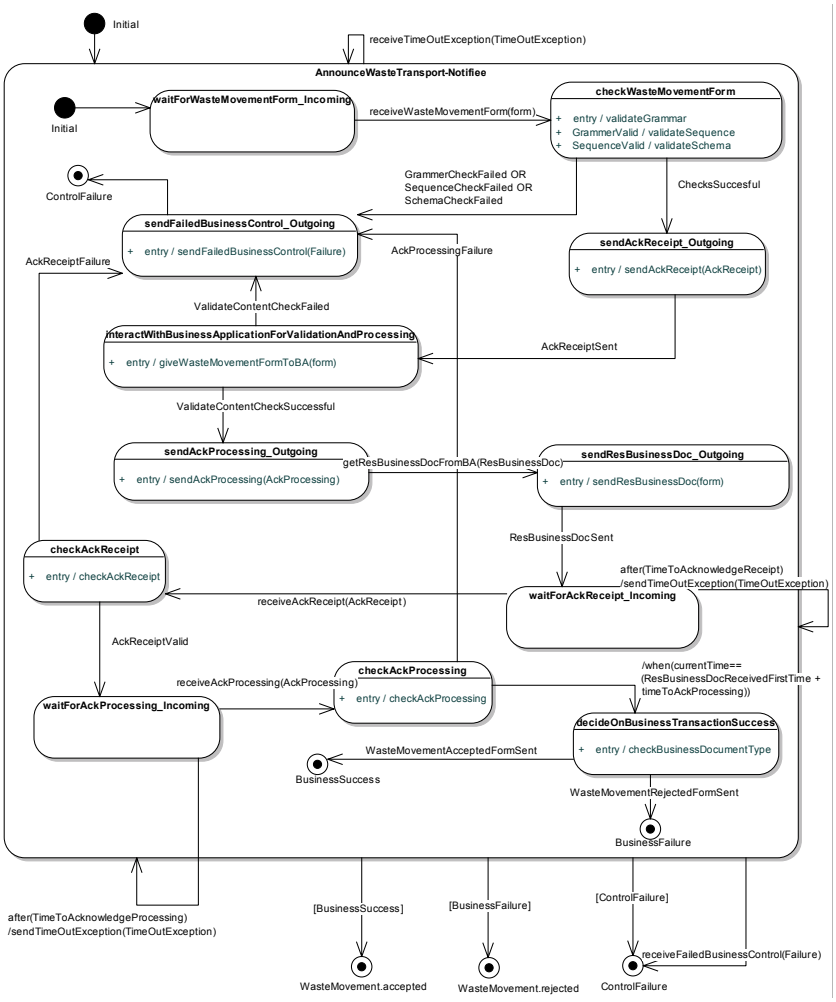


Figure 6.3
State machine of the notifiee's part of `notify waste transport`

no retry is left, no `waste movement form` will be received. However, the `notifier` must issue a failed business control message, which causes the call event *receiveFailedBusinessControl(Failure)* on the `notifiee's` side. This exits the composite state and terminates the transaction by leading to the failure end state.

6.4 Final assessment

A complex business collaboration consists of multiple *business transactions*. As we know, a *business collaboration protocol* is used in UMM to specify the flow of *business transactions*. In order to derive a business partner interface the *business collaboration protocol* is transformed to a state machine. Each *business transaction action* of the *business collaboration protocol* is transformed to a state of this state machine as shown in figure 6.4. It must be decided whether the business partner is the initiator or responder in the corresponding *business transaction*, and the complex state is modeled according to the example state machine we depicted in this section. The flow between the *business transaction activities* must be transformed to

corresponding state transitions guarded by the *business entity states* that are reached in the *business transactions*.

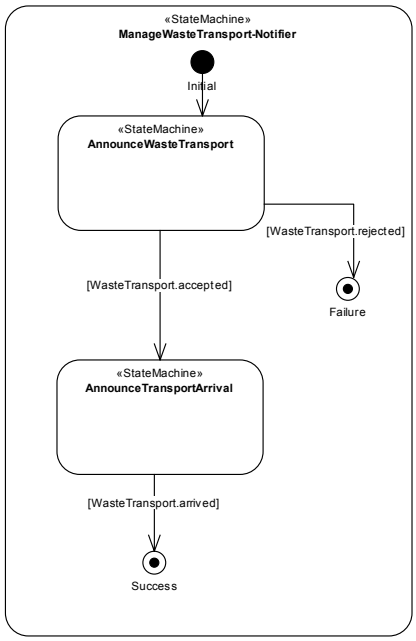


Figure 6.4
Sketch of a state machine representing a business collaboration protocol

A UMM model describes a global choreography of business document exchanges. Realizing the resulting business collaboration requires to implement a derived local choreography supported by a business partner interface on each side of the collaboration. UMM does neither specify how to derive the local choreography, nor does it guide the message handling within the business partner interface. In this sub-section, we used UMM *business transactions* and showed how to derive compliant state machines for implementing the message handling within the business partner interfaces of the initiator and of the responder - illustrated by means of the announce `waste transport` business transaction. It becomes obvious that the rather simple patterns of UMM *business transactions* require a complex message handling mechanism. *Business transactions*, which are easy to understand for business persons, are transformed to system specifications that are useful to the software engineer.

7 Deriving BPEL from UMM Business Transactions.

In the last section, we introduced a state machine representation for business service interfaces supporting UMM *business transactions*. According to OMG's MDA approach, the resulting state machine is a platform independent model, which unambiguously specifies the required behavior of a business service interface regardless of the underlying technology. In order to deploy the model to a specific IT platform, the platform independent model needs to be transformed to a platform specific model. In our approach, the two candidate platforms - i.e., platform specific models - are Web Services and the Windows Workflow Foundation. In this section, we concentrate on the derivation of Web Services artifacts.

BPEL is a promising candidate platform for UMM models

In order to implement a business process using Web Services, we need to map the flow of a business process to a set of Web Service interactions. In the recent past, the Business Process Execution Language (BPEL) [77] has gained strong support from both, industry and academia. BPEL describes a process by a flow of activities representing interactions of Web Services. In general, activities specify the sending or receiving of a message to/from a service. Service interactions are further specified by references to Web Service Description Language (WSDL) artifacts.

BPEL has a partner-specific view on a business process

BPEL describes a business process from a partner-specific viewpoint. If each participant in a collaborative business process, describes the same, shared process from his own view in isolation, the respective BPEL descriptions will most likely not match. Thus, it is required to describe a collaborative process from a global viewpoint first. Having all participants agreed on the business collaboration, complementary partner-specific BPEL description of the respective business service interfaces can be derived.

In the following, we outline how to process a global UMM *business transaction* model for deriving partner-specific business service interfaces in BPEL. Again, we use the `notify waste transport` *business transaction* to illustrate the derivation process. Figure 7.1 shows the example with concrete assignments for the tagged values of the *business transaction*.

There are four entities involved in a business transaction

Considering figure 7.1, we identify four participating entities namely the `notifier`, the `notifier's business application`, the `notifiee` and the `notifiee's business application`. For each of the entities a WSDL is generated. On the `notifier's` and the `notifiee's` side a BPEL process orchestrates the different service invocations.

The remainder of this section is structured as follows: in subsection 7.1 we describe the generation of WSDL artifacts for both

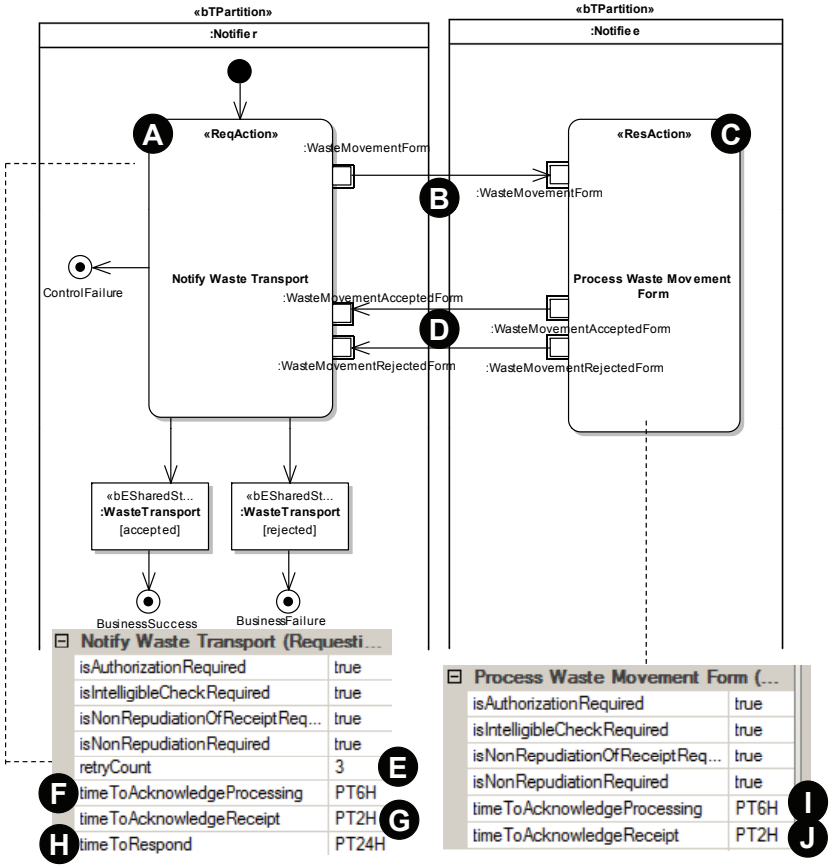


Figure 7.1
`Notify waste transport` *with concrete tagged value assignments*

parties. The derivation of the notifier's business service interface is elaborated in sub-section 7.2 and sub-section 7.3 discusses the differences in terms of the `notifiee`'s process. In order to visualize the resulting processes we use a simple graphical notation since it allows a more thorough understanding than showing the BPEL code listings.

7.1 Generating WSDL

The *business transaction* depicted in figure 7.1 describes how the business process is coordinated between the two business partners. In a Web Service environment, this coordination is done using BPEL specifications where each service interface is described by means of WSDL. In our example the generation of BPEL from the UMM model results in four WSDL files for the following entities: `business application notifier`, `notifier`, `notifiee` and `business application notifiee`.

The WSDL specifies the service operations an entity has to offer

Each WSDL file specifies the operations a party offers, the messages and data types which are exchanged as well as the services the operations are bound to. The WSDL files of the `notifier` and the `notifiee`, respectively, capture also the *partner link types*. Those specify the binary relationship between two participants by referencing the *port type* each participant has to provide. As an example,

listing 7.1 shows the WSDL file of the notifier. Due to readability parts of the WSDL have been left out.

Listing 7.1
WSDL file of the notifier

```
4  <?xml version="1.0" encoding="UTF-8"?>
5  <wsdl:definitions name="Notifier" targetNamespace="http://notifier.at"
6      [...]
7  <wsdl:types>
8      [...]
9  </wsdl:types>
10 <wsdl:message name="WasteMovementForm">
11  <wsdl:part name="parameters" element="xsd1:WasteMovementForm" />
12 </wsdl:message>
13 <wsdl:message name="Acknowledgment">
14  <wsdl:part name="parameters" element="bas:Acknowledgment"/>
15 </wsdl:message>
16     [...]
17 <wsdl:portType name="Notifier">
18  <wsdl:operation name="receiveWasteMovementFormFromBA">
19   <wsdl:input message="tns:WasteMovementForm" />
20  </wsdl:operation>
21  <wsdl:operation name="receiveAck">
22   <wsdl:input message="tns:Acknowledgment" />
23  </wsdl:operation>
24     [...]
25 </wsdl:portType>
26 <wsdl:binding name="NotifierSOAP" type="tns:Notifier">
27     [...]
28 </wsdl:binding>
29 <wsdl:service name="Notifier">
30     [...]
31 </wsdl:service>
32 <plnk:partnerLinkType
33  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
34  name="Notifier-BusinessApplication-PLNKT">
35  <plnk:role name="BusinessApplication"
36  portType="ba:NotifierBusinessApplication" />
37  <plnk:role name="Notifier" portType="tns:Notifier" />
38  </plnk:partnerLinkType>
39 <plnk:partnerLinkType
40  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
41 name="Notifier-Notifiee-PLNKT">
42  <plnk:role name="Notifier" portType="tns:Notifier" />
43  <plnk:role name="Notifiee" portType="wsdl1:Notifiee" />
44 </plnk:partnerLinkType>
45 </wsdl:definitions>
```

In the WSDL file above, the data types used and the messages exchanged during the process are described between line 5 and 15. The different operations of the notifier are specified between line 17 and 25. The bindings for the particular operations as well as endpoint information are specified between line 26 and 31.

We use one operation for handling both types of acknowledgments

The UMM process per se requires different acknowledgment messages exchanged between the participating business partners - i.e., *acknowledgments of receipt* or *acknowledgments of processing*. The operation *receiveAck* specified in line 21 handles the different acknowledgments received by the notifier. The distinction which type of acknowledgment has currently been received is made by examining the acknowledgment's content at the business service interface.

7.1.1 Generating partner link types

Partner link types represent the binary relationship between two roles (i.e., participants) in a process by means of the service they provide. Each role refers to a specific port type of the WSDL file. In our example the notifier interacts with his business application (line 32 of listing 7.1) and with the notifiee (line 41 of listing 7.1). The notifiee has also two partner link types - one for the interaction with the notifier and one for the interaction with his business application.

Partner link types bind two compliant BPEL processes together

The UMM model describes the transaction from an overall point of view. BPEL, however, describes a process from the point of view of a participant. Hence, the UMM model results in two different BPEL files - one for the notifier and one for the notifiee. Partner links are provided in BPEL for specifying the relationship between

compliant BPEL processes. Listing 7.2 shows an excerpt from the BPEL process of the notifier. In line 51 of listing 7.2, the partner link to the notifier is specified by referencing the partner link type *Notifier-Notifiee-PLNKT* (specified in line 41 of listing 7.1). Since the notifier is the owner of the process, the attribute *myRole* refers to the notifier role of the partner link type and the attribute *partnerRole* refers to the notifiee.

The BPEL file of the notifiee specifies the same relationship, however, having the values for *myRole* and *partnerRole* swapped.

```
46 <bpel:partnerLinks>
47  <bpel:partnerLink myRole="Notifier" name="Notifier-BusinessApplication-Link"
48   partnerLinkType="ns1:Notifier-BusinessApplication-PLNKT"
49   partnerRole="BusinessApplication"/>
50  <bpel:partnerLink myRole="Notifier" name="Notifier-Notifiee-Link"
51  partnerLinkType="ns1:Notifier-Notifiee-PLNKT"  partnerRole="Notifiee"/>
52  </bpel:partnerLinks>
```

Listing 7.2
Partner Link example

7.2 Generating the notifier's BPEL process

The notifier's process is shown in figure 7.2. The first activity is a *receive* called receive waste movement form from BA (1 in figure 7.2). It picks up a waste movement form (B in figure 7.1) sent by the business application. Since it is the first activity, it creates a new BPEL process instance upon receipt of the document. In a UMM *business transaction*, this action - performed within the notifier's system - is carried out as part of the *requesting business action* (A).

UMM's retry count *is represented as a loop in BPEL*

As we know, a *business transaction* may be re-initiated due to time-out exceptions. In order to allow possible re-starts of the *business transaction*, we use BPEL's *repeat until* activity - check retry count (2) - containing the actual process flow. If a time-out exception occurs, the check retry count activity starts a new iteration as long as its condition evaluates true. A new iteration is started if the *retry count* is greater than or equals zero and the variable process ended is false. The *retry count* is expressed by a BPEL *variable* that is decremented each time a time-out exception occurs.

According to the *retry count* (E) of the notify waste transport example, the transaction may be re-initiated three times - which corresponds to four runs in total. Within check retry count, a *scope* (3) is used to structure the business process. As shown in figure 7.2 the *scope* comprises the activities carrying out the regular process (4). Furthermore, the scope has *fault handlers* (5) and *event handlers* (6) associated.

7.2.1 Regular process

The notifier starts the actual *business transaction* by an *invoke* (7) calling the notifiee's operation process waste movement form (C). According to our example, the operation consumes a waste movement form (B). Since the notifiee needs some time to process the waste movement form, the invoke is performed asynchronously and the notifier's process is continued immediately. Next, the *assign* activity save timestamp waste movement form sent (8) saves the current time to a BPEL *variable*. Based on the time the waste movement form

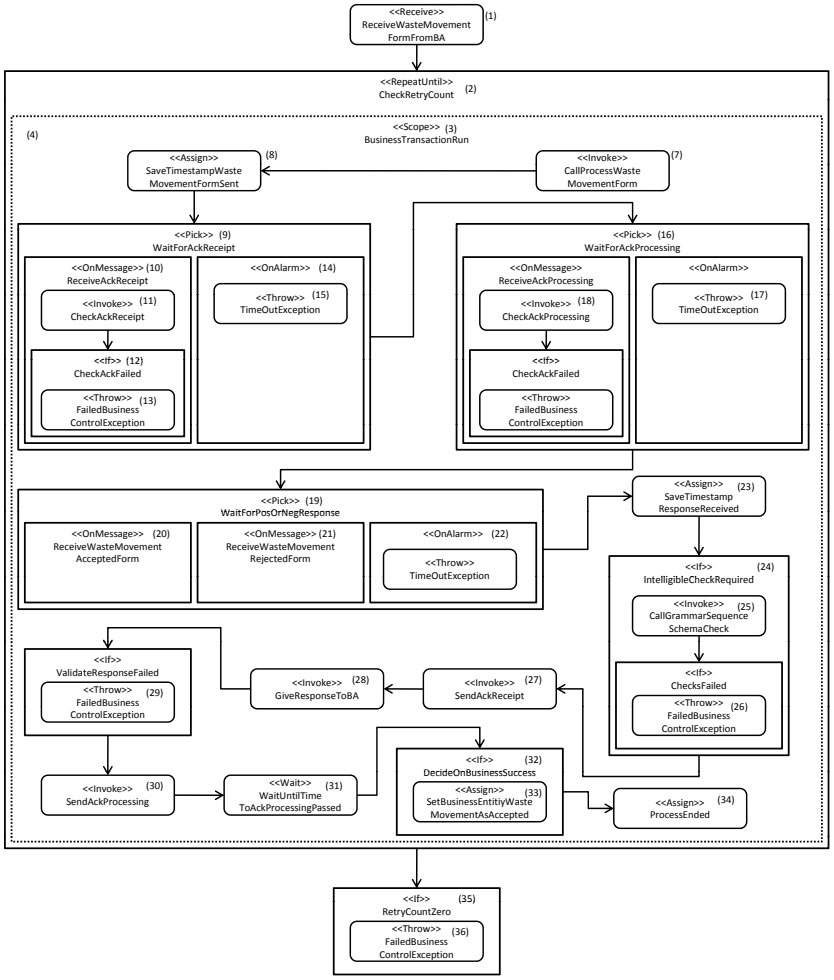


Figure 7.2 *Notifier's BPEL Process*

was sent, we calculate the time limits within acknowledgments and the response document are expected.

A pick represents the waiting for a message under the consideration of a possible time-out

Next, the `notifier's` business service interface waits for an acknowledgment from the `notifiee` confirming the receipt of the `waste movement form`. It must be received within two hours - the agreed *time to acknowledge receipt* (G). Therefore, we use a *pick* activity named `wait for ack receipt` (9) that waits for the occurrence of one event from a set of events. According to the semantics of a UMM *business transaction*, there are two possible scenarios in this step:

An on message element within a pick waits for the receipt of a message

Firstly, the acknowledgment is received, which is modeled using an *on message* branch (10). In this case, the acknowledgment is handed over to an external service that checks the acknowledgment's content. The communication with that service is defined by a synchronous *invoke* called `check ack receipt` (11). Its result is evaluated by the following *if* activity (12). In case the checks fail, the *throw* activity (13) raises a failed business control exception. The handling of exceptions is outlined in sub-section 7.2.2.

An on alarm element within a pick monitors time limits

Secondly, if no *acknowledgment of receipt* is received in time, the *on alarm* event (14) is actuated. *On alarm* corresponds to a timer-based alarm - either specified by a deadline (in terms of a timestamp)

or by a duration. In our example, we define the time limit as a deadline that equals to the variable `timestamp waste movement form sent` (saved in step (8)) plus the agreed *time to acknowledge receipt*. In case the deadline is met, a time-out exception is thrown (15). As outlined later, handling the time-out exception results in decrementing the *retry count* and in terminating the current attempt by re-starting with `check retry count` (2).

Acknowledgments of processing are modeled as acknowledgments of receipt in BPEL

If an *acknowledgment of receipt* was properly received, the `notifier` expects an *acknowledgment of processing*. It indicates that the `notifiee's` business application is able to process the `waste movement form`. Handling an *acknowledgment of processing* (16) is similar to handling an *acknowledgment of receipt*, except that the agreed time limit may differ. According to our example, the `notifiee` must send the acknowledgment within 6 hours (F) - i.e., the deadline corresponds to `timestamp waste movement form sent` plus *time to acknowledge processing*. Again if the message is not received on time (17), a time-out exception is thrown. Otherwise, upon receipt of the *acknowledgment of processing*, the *invoke* activity `check ack processing` (18) sends it to the external service mentioned before in order to validate it. If the check was successful, the process expects one of the two possible response documents (D) - either a `waste movement accepted form` or a `waste movement rejected form` - to be received.

Each possible response document is modeled by its own on message element

Similar to the waiting for acknowledgments a *pick* activity `wait for pos or neg response` (19) is used to specify the time constraint. Since the `notifier` may either receive a `waste movement accepted form` or a `waste movement rejected form`, we nest two *on message* activities within the *pick* activity - one for receiving the acceptance (20) and one for receiving the possible rejection (21). In case the deadline (`timestamp waste movement form sent` plus *time to respond*) (H) is elapsed, a time-out exception is thrown (22).

The BPEL business service interface uses external services for validation purposes

Upon receipt of either document, the current time is stored (23) into a *variable* (`timestamp response received`) for further processing. In case an intelligible check - i.e., grammar, sequence and schema validation - of the document is required (24), it is sent to a validation service via a synchronous *invoke* activity `call grammar sequence schema check` (25) for performing the validation routines. Given an erroneous response document a failed business control exception is reported by the service (26).

If the response document is properly received, the process proceeds to `send ack receipt` (27). This asynchronous *invoke* confirms the successful receipt of the response document to the `notifiee`. Subsequently, the invoke `give response to BA` (28) hands over the response document to the `notifier's` business application for further processing. The business application synchronously returns whether the response can be processed or not. In the latter case, a failed business control exception is thrown (29). In the former case, the `notifiee` is provided with an *acknowledgment of processing* (30).

After sending the response, the `notifier` *has to keep the business transaction alive for some time*

The *business transaction* must be kept alive to allow the `notifiee` to issue a failure if he does not receive the acknowledgment (*time-out exception*) or is not able to process it accordingly (*failed business control exception*). Receiving failures from the `notifiee` is detailed in

the sub-section about *event handlers*. The time until the `notifier` has to wait corresponds to `timestamp response received` plus the *time to acknowledge processing* (I) of the document. Keeping the *business transaction* alive is specified by a *wait* activity (31) with the given deadline.

Having passed the deadline, the process has to check whether the *business transaction* has succeeded in a business sense (32) or not - i.e., whether the response is a `waste movement accepted form` or a `waste movement rejected form`. In case of a business success, a boolean variable is set true by the assign activity `set business entity waste transport as accepted` (33). On the level of a business collaboration, the flow of the process is governed based on setting of the boolean variable. Afterwards, the variable `process ended` is set true (34) to prevent further runs of the *repeat until* activity `check retry count` (2). Finally, an *if* activity (35) checks whether the *retry count* equals zero. A *retry count* of zero indicates that all attempts to execute the business transaction failed. In this case, a failed business control exception is thrown (36).

7.2.2 Fault handlers

Two types of exceptions may occur in a UMM business transaction

During the execution of a UMM *business transaction* abnormal behavior is represented by exceptions. We distinguish between two types of exceptions: *time-out exceptions* and *failed business control exceptions*. The former ones, indicate that a certain message is not received on time. The latter ones signal that a message's integrity is violated or that the maximum amount of attempts to re-initiate the *business transactions* is reached. Exceptions are either raised internally in case abnormal behavior is encountered or are signalized by the business partner.

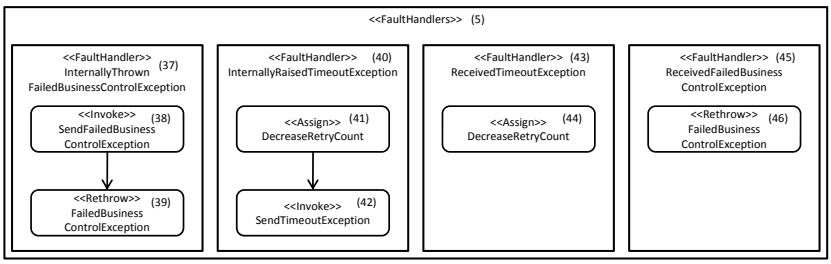


Figure 7.3 *BPEL fault handlers for managing exceptional behavior within the* `notifier's` *regular process flow*

BPEL provides fault handlers for handling exceptional behavior

In BPEL, exceptions are managed by so-called *fault handlers*. The fault handlers of our example BPEL process are sketched in figure 7.3. As figure 7.2 shows, the actual process flow is modeled in a *scope* ((3) in figure 7.2), which is the only activity within the *repeat until* container (2). The *fault handlers* of figure 7.3 are attached to the *scope* in figure 7.2. If an exception is encountered, the current execution of the *scope* is terminated and the corresponding *fault handler* is activated.

In our example, we installed four different *fault handlers*: The first one ((37) in figure 7.3) is activated if a failed business control exception is thrown internally. In this case, the `notifier` communi-

cates the notifiee the failure via the *invoke* send failed business control (38) including a reason of failure. Subsequently, the exception is re-thrown (39) to the enclosing owner of the *business transaction*. The second one (40) handles an internally raised time-out exception. It decreases the available retries by an *assign* activity (41) and communicates the time-out exception (42) referencing the missing message to the notifiee. Activating a *fault handler* skips the current execution of the *scope*. Since the *scope* is contained in the *repeat until* activity, the loop condition is re-evaluated after fault handling. Thus, it is possible to re-initiate a *business transaction*. The third *fault handler* (43) treats time-out exceptions signaled by the notifier. The *retry count* is again decremented (44) and the control flow is handed back to the *repeat until* activity. The last *fault handler* (45) catches a failed business control exception that was received from the notifier. In this case, the exception is handed over to the process owner (46).

A fifth fault handler is attached to the whole process

If the *retry count* is smaller than zero at the end of the process (36), a failed business control exception is raised, too. However, this exception is not handled by the fault handlers (5) attached to the *scope* (3). Instead, we install a *fault handler* - working similar as the first one described above - to the scope of the BPEL process.

7.2.3 Event handlers

As we learned, the notifier may receive a time-out exception or a failed business control exception at any time during the process execution. To implement this requirement, we use the concept of *event handlers* provided by BPEL.

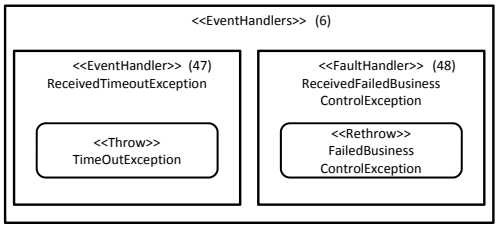


Figure 7.4 *BPEL event handlers for managing occurring events during the* notifier's *regular process flow*

Event handlers listen to events that may occur during the regular process execution. We attach two *event handlers* (6 in figure 7.4) to the *scope* (3 in figure 7.2), which represents the regular flow of a *business transaction*: The first *event handler* (47) deals with a time-out exception signaled by the notifiee. Once it is received, a time-out exception is raised that is managed by the corresponding fault handler (43). The second one (48) is responsible for failed business control exceptions. Similarly, it re-throws the exception to be further treated by the fourth *fault handler* (45).

Event handlers re-act to events that may occur concurrently to the regular process

7.3 Generating the notifiee's BPEL process

As we know from section 6, the concepts used in the notifiee's process are almost analog to those described in the notifier's pro-

cess. Since, the notifier's and the notifiee's process must be complementary, the order of receiving and sending the waste movement form and the waste movement accepted form or the waste movement rejected form, respectively, is reversed. This also includes the handling of the acknowledgments.

The major difference between the two processes is that the notifiee does not control the *retry count*. If the notifiee does not receive an acknowledgment on time, a time-out exception is communicated to the notifier but no *retry count* is decreased. Hence, no loop activity is required in the notifiee's process.

7.4 Final assessment

Simple UMM business transaction become complex BPEL processes

In this section we demonstrated the binding of UMM *business transactions* to the Web Services stack using BPEL and WSDL, respectively. The mapping follows the platform independent model for the notify waste transport example created in section 6. In MDA terms, the resulting artifacts conform to a platform specific model, which may be used for deployment to an orchestration engine. Thereby, a simple UMM *business transaction* becomes a complex BPEL process. In our approach, we employ scopes, event handlers and fault handlers for managing the different types of business signals which are exchanged during the process.

As already outlined in section 6.4, the mapping of whole *business collaboration protocols* additionally requires mapping the flow between *business transaction*. In other words, each *business transaction* of a *business collaboration protocol* is mapped according to the example presented in this section. In order to govern the flow between the *business transactions* in BPEL, corresponding guard conditions need to be set in the BPEL process. Other remaining tasks to execute the generated code comprise: (i) the definition of concrete service endpoints and (ii) the binding of the derived business service interface to the business application.

8 Transforming UMM Business Transactions to Business Service Interfaces in Windows Workflow

Beside the Web Services stack, we use the Windows Workflow Foundation (WF) [68] as an alternative deployment platform in our three-layered approach. The Windows Workflow Foundation (WF) is an extensible framework for building workflow-centric applications in .NET. The WF makes workflows first-class citizens of an application by having a clear business process focus. Workflows may either reside within an application or may communicate with other applications by providing or consuming services.

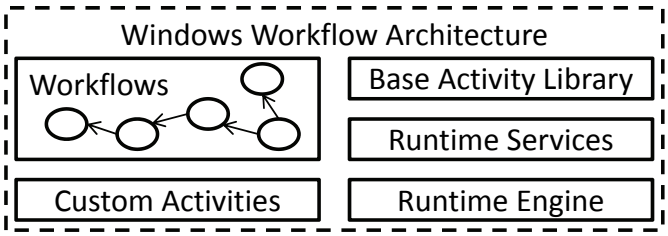


Figure 8.1
Windows Workflow (WF) architecture

Windows Workflow allows developers to build workflow centric applications

The general architecture of WF is shown in figure 8.1. A more detailed introduction to the various components is found in [1]. In general, workflows are executed within the workflow runtime, which may be hosted in any type of .NET application. The runtime engine's functionality is expendable through runtime services. For example, developers may add persistence or tracking capabilities to their workflows by applying the corresponding services, which come out of the box.

WF provides a basic activity library out of the box

The basic building blocks of a workflow are activities. WF provides a set of standard activities for common purposes such as flow control (*parallel*, *if/else*, *sequence*, etc.), event handling (*listen activity*, *event handling scope activity*, etc.), or distributed communication (*send activity*, *receive activity*, etc.). The most primitive activity - the *code activity* - allows to quickly add any custom .NET code. However, in order to re-use custom code across workflows, developers should rather implement specific business logic by the concept of *custom activities*. Custom activities encapsulate a unit of work that serves a specific business purpose.

Sequential workflows implement automated processes, whereas state machine workflow realize user-driven business processes

WF supports two flow paradigms for the definition of workflows - *sequential workflows* and *state machine workflows*: The former define a prescribed series of execution. A sequential workflow, however, may contain loops, branches with conditions and may listen to external

events. The underlying flow model is similar to the one of BPEL. Sequential workflows are typically used in automated processes, where the workflow is in control. On the other hand, state machine workflows are event-driven. The developer creates a workflow as a set of states and transitions between them. Events trigger the execution of activities and/or the transition to other states. Thus, the events determine the execution path of the workflow. The target of a transition may be any other state within the workflow. State-machine workflows are often used for business processes that involve much human interaction, where the execution order heavily depends on user input. Furthermore, state machine workflows are used if the process is mainly driven by external events or if all possible execution paths are hard to describe within a sequential workflow.

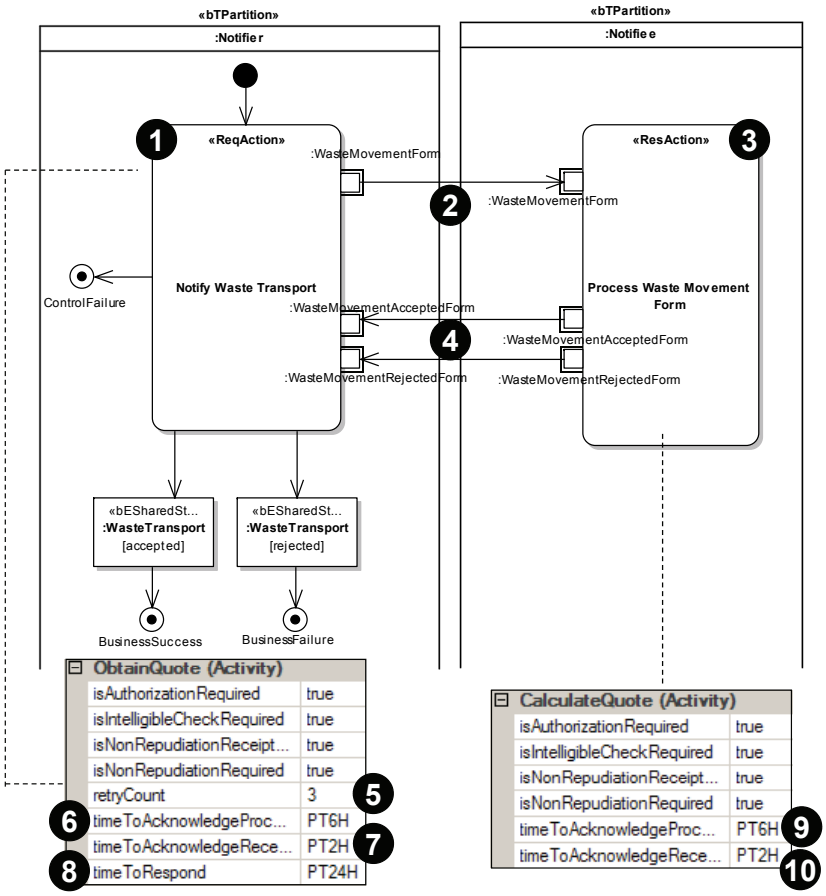


Figure 8.2
Announce waste transport *with concrete tagged value assignments*

Inter-organizational communication of the BSI is based on Web Services

In the remainder of this section we describe in detail the transformation of UMM *business transactions* to business service interfaces (BSI) realized in WF. The flow within the BSI is defined using WF's sequential workflow language. The interface to the workflow, however, is composed of well-defined business services implemented using Web Service specifications. Since examples facilitate understanding, we detail the mapping again by means of the announce waste transport example (cf. figure 8.2). The transformation steps, however, correspond to a general pattern. Consequently, every UMM business transaction may be transformed to WF following the steps

described in this sub-section. As we have found out in the previous sections, the notifier's and the notifiee's business service interfaces are complementary to each other. In order words, the order of sending and receiving business documents is reversed. The same applies for the handling of acknowledgments. Therefore, we concentrate only on the notifier's part to describe the transformation process.

8.1 The regular process flow

In the following, we demonstrate the mapping by means of our example *business transaction* depicted in figure 8.2. The resulting business service interface realized in WF is shown in figure 8.3. The WF process is defined as a sequential workflow resulting in 12 major steps (A-H), whereby steps F to K are composite activities. If not explicitly noted else, all activity types are contained in WF's basic activity library.

Step A: Interacting with the business application

At the very beginning, the `notifier's` BSI receives the `waste movement form` from the business application. Receiving the document is implemented by a *handle external event activity*. This presupposes that the business application is implemented in .NET as well. If this is not the case, the *handle external event activity* may be substituted by a *receive activity* for enabling cross-platform communication - for example realized by Web Service calls.

The Windows Communication Foundation realizes distributed communication workflows

The *receive activity* and the *send activity* integrate the Windows Workflow Foundation with the Windows Communication Foundation (WCF) [67]. The WCF is the .NET approach to unify and simplify distributed communication. The WCF follows the 'ABC' model by separating address, binding, and contract. A *receive activity* is bound to a .NET interface known as service contract. The service contract specifies, which operations a service has to expose. WCF services may be deployed using several bindings including the WS-I Basic Profile 1.1 [122], .NET Remoting, and Microsoft Message Queuing. One or more endpoint addresses may be declaratively specified for a WCF service. Moreover, the same WCF service may be exposed at different endpoint addresses using different bindings.

Step B: Checking the available retries

The available retries are monitored by a while activity

As we already know from UMM *business transaction* semantics, the initiator has to re-start a *business transaction* in case of time-outs according to the agreed *retry count*. In our example, the *retry count* amounts to three (see (5) in figure 8.2). We use the WF *while activity* to repeat the execution of the *business transaction* if required. Consequently, the *while activity* (B in figure 8.3) has to be executed until either the retry count is exceeded or the *business transaction* is flagged as finished. Thus, we define the loop's condition as `retryCount >= 0 && !businessTransactionFinished`, whereby both parameters are defined as normal .NET variables within the work-

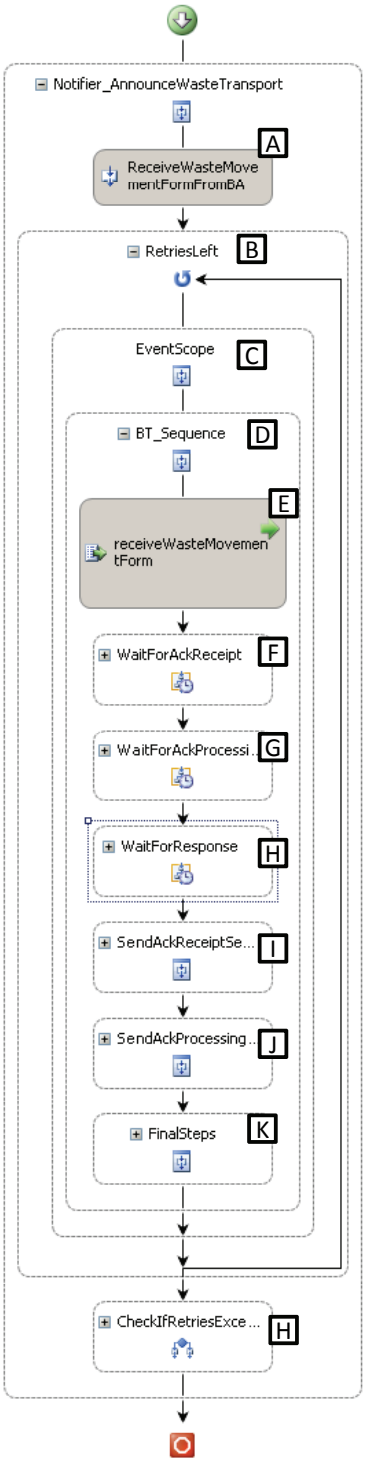


Figure 8.3 *Notifier's business service interface implemented using Windows Workflow*

flow. In case the *business transaction* is finished - either resulting in a business success or business failure - the BSI sets the variable `businessTransactionFinished` to *true* in step K.

Steps C and D: Listening to business signals during the regular process flow

The event handling scope activity re-acts on communicated exceptions

The only activity within the *while activity* is an *event handling scope activity*. This activity type allows to act upon events concurrently to the execution of the regular process flow. Such events may be time-out exceptions or failed business control exceptions received from the `notifiee` at any time during the course of the *business transaction*. Consequently, we use the *event handling scope activity* for receiving and processing business signals concurrently to the regular process flow.

The *event handling scope activity* may have several event handlers attached. Thereby, each event handler listens to a certain type of event (i.e., an incoming business signal). We discuss event handlers and the actions they trigger in sub-section 8.2. The *sequence activity* (D in figure 8.3) within the *event handling scope* serves as a container for the activities realizing the message exchange with the notifiee's BSI (steps E to K).

Step E: Sending the waste movement form

Step E communicates the waste movement form ((2) in figure 8.2) from the notifier's BSI (1) to the notifiee's one (3). On the notifier's side, the service call is implemented using the *send activity*. Note, that `receive waste movement form` (E in figure 8.3) is indeed a *send activity*, which refers to the operation offered by the responder. The call is performed asynchronously (denoted by the single arrow on the right hand side of the *send activity* (E)), which means that the workflow continues immediately. The semantics of an asynchronous operation call by a *send activity* correspond to a truly fire-and-forget behavior. This entails that the client does not even receive a fault message from the service in case of an exception.

Business transactions are implemented by completely asynchronous communication

This behavior is in line with the semantics of asynchronous UMM *business transactions patterns*. Thereby, business document exchanges are completely asynchronous in order to avoid blocking behavior of business service interfaces. Nevertheless, interacting business service interfaces share the same understanding about the state of a business document exchange by communicating business signals as shown in the following steps.

Step F: Waiting for the acknowledgment of receipt

Grammar and schema checks of a message are straightforward for strongly typed XML-based business documents...

After sending the `waste movement form`, the notifier waits for a business signal of type *acknowledgment of receipt* from the notifiee's BSI. According to UMM *business transaction* semantics, an acknowledgment of receipt is issued after a received business document passes grammar-, schema-, and sequence validation. One may argue that these checks may be performed instantly at the receiving BSI and, hence, the receipt of the request document should be acknowledged synchronously. This assumption is valid in a Web Services environment, where service interfaces expect strongly typed messages according to a certain XML schemata. In this case, grammar- and

schema validation is implicitly performed at the receiving service interface by the underlying implementation technology.

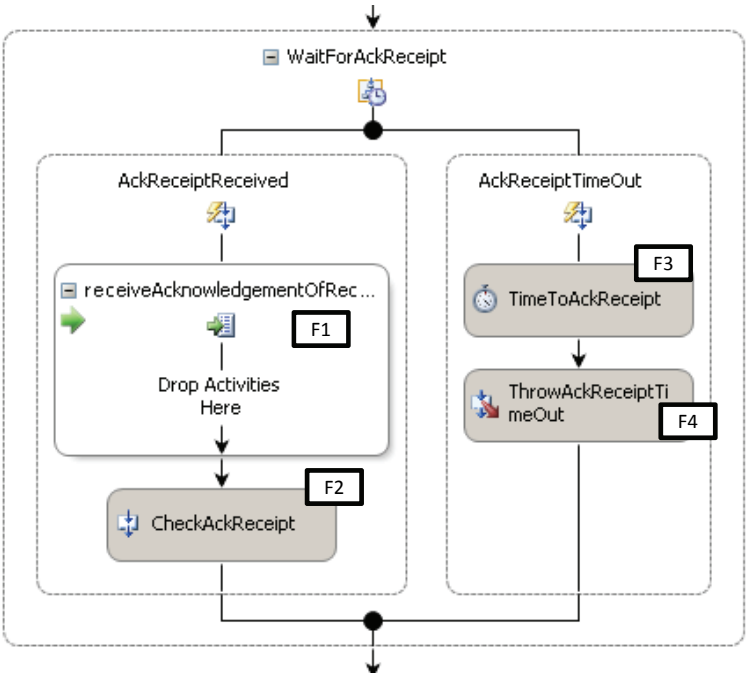


Figure 8.4
Step F: Waiting for the acknowledgment of receipt

...but are not trivial for traditional EDI standards

However, numerous established electronic data interchange (EDI) standards are not defined in XML Schema, but have their own non XML-based syntax. Let's consider a business service interface that must be capable of processing EDIFACT - a traditional, but still widely used EDI standard. In EDIFACT, the content of a message is separated using plus and colon characters as delimiters: In this respect, the business service interface exposes service operations that take strings as input arguments. Consequently, the service interface is not able to validate grammar and schema of the input message upon receipt. Instead, after receiving the message, some internal functionality of the BSI has to parse the business document and has to perform a variety of checks to ensure its validity. Such checks require processing time - particularly if the business document comprises several EDIFACT messages or some received messages are already enqueued. In this respect, synchronous acknowledgments would lead to blocking behavior of the business service interface, which is evidently not desired in an e-business environment. Thus, implementing e-business transactions mostly requires the decoupling of business documents from their receipt acknowledgments.

The listen activity provides deferred choice behavior

Figure 8.4 shows the required activities of step F in detail. The notifier expects the *acknowledgment of receipt* from the notifiee's BSI to confirm that the `waste movement form` passed the syntactical checks. The *listen activity* in step F has two branches. The left branch is activated when the notifier's business service interface receives the *acknowledgment of receipt*. If the acknowledgment, however, is not picked up within the agreed time frame, the right branch is activated. The *listen activity* is responsible for activating that branch, whose trigger event occurs first. The remaining branches are canceled.

In order to expose a service for receiving the acknowledgment, the first activity in the left branch is a *receive activity* (F1). In figure 8.4, the *receive activity* is decorated with an arrow on the left hand side, which visually distinguishes it from the *send activity*. The *receive activity* is bound to an operation called `receive acknowledgment of receipt`. We define this operation in a service contract particularly for business signals. This service contract is not restricted to any business context and may be globally defined for business transactions. In regard to our example, we assume that at least the notifier and the notifiee's bind their BSI's to this service contract for exchanging business signals.

In contrast to BPEL, message checks are performed directly by the business service interface

The *receive activity* is followed by an activity that is responsible for checking the contents of the received acknowledgment (F2). Similar to the service contract for business signals, these checks may be identical for the same type of business signal across different *business transactions*. Thus, we propose to implement the required checks and constraints in a custom activity type. The *custom activity* `check ack receipt` may then be re-used in different Windows Workflow business service interfaces. If, by any reason, checking the acknowledgment of receipt fails, the custom activity throws an exception that is further handled as outlined in section 8.2.

Delay activities realize time-outs

In the right branch, the first activity is a *delay activity* (F3). It monitors the agreed *time to acknowledge receipt*. In our example, this time limit corresponds to a maximum of two hours (see (7) in figure 8.2). If exceeded, the *delay activity* triggers a time event which makes the *listen activity* activating the right branch (and consequently deactivating the left branch).

The throw activity cancels the current run of a business transaction

In this case, the *business transaction* has to be re-started due to a time-out exception. In order to re-start the *business transaction* the current run has to be canceled and the condition of the *while activity*, which monitors the retries, has to be evaluated again. This behavior is accomplished by the *throw activity* (F4) following the *delay activity*. The *throw activity* actuates a *time-out detected exception* that is caught by a fault handler attached to the *while activity*. Handling faults is further discussed in section 8.2.

Step G: Waiting for the acknowledgment of processing

As shown in figure 8.5, the notifier expects an *acknowledgment of processing* in this step. It confirms that the `waste movement form` was successfully handed over to the notifiee's business application for further processing. This implies that the business document was delivered to the business application, where it passed additional validation rules. Validation by the business application covers checks on the semantic level - e.g., the `waste movement form` must only refer to valid ISO country codes.

From an implementation perspective, handling acknowledgments of receipt and processing is very similar

In terms of the activity flow, handling *acknowledgments of processing* and their contingent time-outs is similar to processing *acknowledgments of receipts*. In the left branch, the steps G1 and G2 model the reception and the checks for a received acknowledgment, whereas the right branch (G3 and G4) handles the time-out. The agreed time-out monitored by the *delay activity* (G3) corresponds to

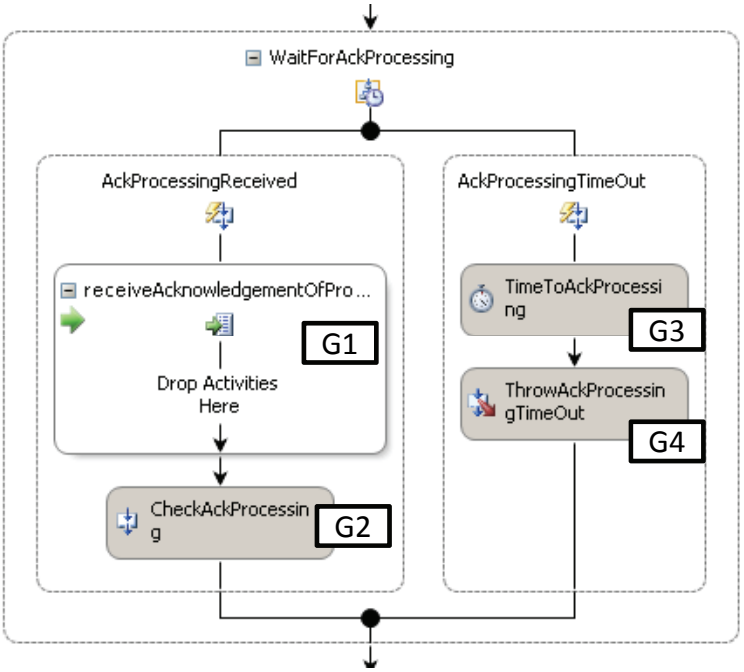


Figure 8.5
Step G: Waiting for the acknowledgment of processing

the time to acknowledge processing as defined in (6) in figure 8.2. The *acknowledgment of processing* affirms the notifier that the notifiee is able to process the `waste movement form` and will respond to it.

Step H: Waiting for the response document

Similar to steps F and G, waiting for the response document is implemented by a *listen activity* (H). According to our example, three branches are required to represent the potential alternatives:

We need one branch in the listen activity for each possible business document type

We define one branch for monitoring the maximum time limit as agreed in the UMM model (see time to respond (8) in figure 8.2). The excerpt in figure 8.6 shows that the right branch keeps track of the time limit. If no response document is received within the agreed time to perform, the *delay activity* (H3) triggers a time event and the *throw activity* (H4) terminates the current cycle.

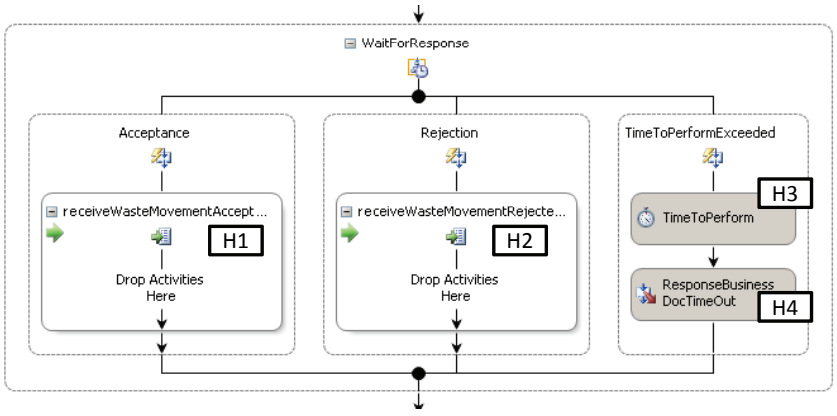


Figure 8.6
Step H: Waiting for the response document

The branch on the left hand side is triggered upon receipt of a `waste movement accepted form`. This branch is composed of the *receive activity* H1. Similarly, the middle branch contains the *receive activity* H2, which listens for a `waste movement rejected form`. As we know, as soon as an event triggers the execution of any branch of the *listen activity*, the remaining branches are canceled.

Step I: Sending the acknowledgment of receipt

Custom activities are employed to re-use validation routines across BSI's

Before the receipt of either response document is acknowledged, the business service interface is required to perform a grammar-, schema-, and sequence validation. Since these are generic validation routines we employ again the concept of custom activities. If the business document passes the checks in step (I1), the *send activity* (I2) confirms the successful receipt by communicating an *acknowledgment of receipt* to the notifiee's BSI. A validation exception would raise a failed business control exception as further outlined in section 8.2.

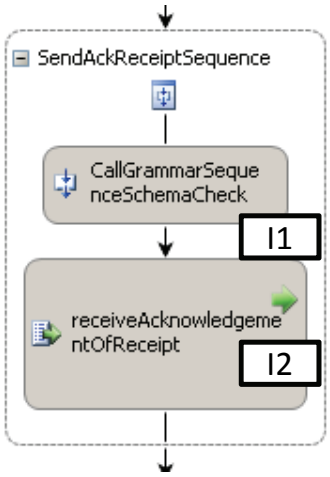


Figure 8.7
Step I: Sending the acknowledgment of receipt

Step J: Sending the acknowledgment of processing

The business application checks the business document against additional business rules

After the proper receipt of the response document is affirmed, the notifier's BSI hands it over to the business application for further processing. As outlined before, our example assumes that the business application hosts the business service interface. Therefore, we implement the communication between those systems using a *call external method activity* (J1). Once the response document is delivered, the business application verifies that the document is processable according to pre-defined business rules.

If no exception is thrown by the business application, the BSI sends an acknowledgment of processing (J2) denoting that the verification was successful.

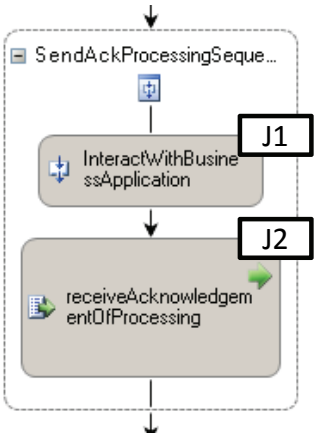


Figure 8.8
Step J: Sending the acknowledgment of processing

Step K: Final steps

Within this sequence some final tasks are performed, which are required before the execution of the *business transaction* can finish. The first activity is a *delay activity* (K1 in figure 8.9), which keeps the *business transaction* alive in order to allow the notifiee to issue a time-out exception or a failed business control exception.

After sending the acknowledgment of processing, the notifiee may still communicate an exception

The former is communicated by the notifiee, if the acknowledgment of processing is not received in time by its business service interface. The latter one is thrown, if an acknowledgment is received, which is not processable. How long the *business transaction* is kept alive is calculated by adding the *time to acknowledge processing* of the response document (I in figure 8.2) to the time when processing the response document was acknowledged (step J2).

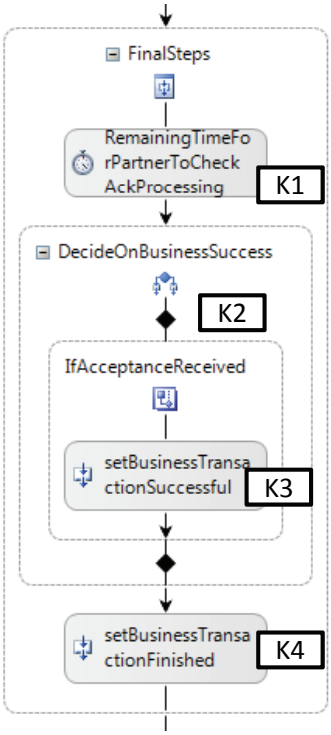


Figure 8.9
Step K: The business service interface performs some final checks

Next, the BSI has to check whether a positive or negative response was communicated by the notifiee in order to determine the further steps in the business collaboration. The *if/else activity* K2 checks if a waste movement accepted form was picked up or not. If so, the custom activity `set business transaction successful` (K3) sets a corresponding variable of the business service interface.

The BSI has to determine if the business transaction was a business success or not

Finally, the custom activity K4 sets the variable `businessTransactionFinished` to true. One should note that this variable is evidently set in either case - a business success or a business failure. It effects, however, that the condition of the *while activity* (B), which is defined by `retryCount >= 0 && !businessTransactionFinished`, is not met any longer and, hence, the while loop stops.

Step H: Checking the retry count

Before the *business transaction* is eventually finished, the business service interface must check if the *retry count* is not exceeded. Note, the loop continues until the *retry count* is equal or greater than zero. If the *retry count* is decremented to -1 at the end of the last attempt, the condition of the *while activity* is not met any longer and the control flow reaches step H.

Is the retry count exceeded, or not?

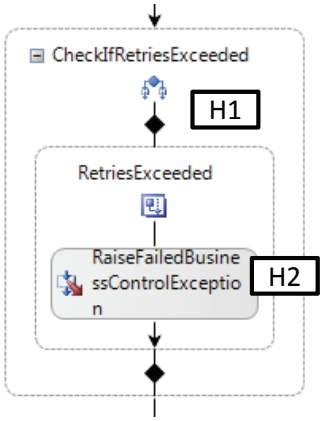


Figure 8.10
Step H: Finally, the business service interface checks if the retry count is exceeded or not

Therefore, the *if/else activity* (H1 in figure 8.10) must query if the *retry count* is greater or equal to zero. If true, the *business transaction* was successful in a business sense and the process continues accordingly. Otherwise, a failed business control exception is thrown (H2) and the *business transaction* failed. Furthermore, this results in a failure of the overall business collaboration, which includes the *business transaction*.

8.2 Event and fault handlers

Aside from executing the regular process flow, the business service interface must be capable of handling exceptional behavior. There are two concepts for managing deviations from the regular process flow: *event handlers* and *fault handlers*. The former ones deal with business signals communicated by the other party in the *business*

transaction, whereas the latter ones handle exceptions that are raised through received business signals or that are detected locally.

Event handlers process occurring events beside the regular process flow

Event handlers listen to events that may occur concurrently to the regular process flow. The occurrence of an event activates the respective event handler and executes the nested activities therein, but it does not cancel the regular process flow. In other words, both activity flows are executed concurrently. An event handler guarantees that participants can dispatch a time-out exception or a failed business control exception at any time. In WF, one or more event handlers may be attached to an *event handling scope activity* to listen for events beside executing the nested process flow.

Fault handlers realize try/catch behavior within workflow

Fault handlers are able to catch any type of .NET exceptions that are raised during the execution of a workflow. Exceptions may either be thrown within the code (i.e., by the execution of an activity) or declaratively modeled by a *throw activity* as part of the workflow. Raising and handling exceptions in WF is similar to common object-oriented programming languages. This means, exceptions may be differentiated by their types and thrown and re-thrown across scopes until a proper fault handler is found. For supporting UMM *business transactions*, we use combinations of event and fault handlers to manage exceptional behavior in business service interfaces. According to the UMM *business transaction* semantics, the following faults may occur from the notifier's perspective:

EX-1: The notifiee sends a time-out exception

If the notifiee detects a time-out for a business document or a business signal on his side, he communicates a time-out exception to the notifier. For receiving time-out exceptions concurrently to the regular process flow, we attach an event handler to the *event handling scope activity* in step C in figure 8.3.

A received time-out exception result is re-thrown as a local time-out exception

The first activity of the event handler defines the event to which the handler is listening. In case EX-1, the first activity is a *receive activity* that is bound to an operation for picking up time-out exceptions. Next, a *throw activity* raises a time-out exception that is caught by a fault handler attached to the *event handling scope activity* in step D. Within the fault handler, there is a custom activity that decrements the retry count. The execution of the fault handler cancels the nested activities within the *event handling scope activity* (i.e., steps E to K). Since the *event handling scope activity* is enclosed by the *while activity* (B), the control flow continues with re-evaluating the loop's condition.

EX-2: The notifiee sends a failed business control exception

A received failed business control terminates the business transaction with an exception

A failed business control exception is communicated by the notifiee if he is not able to process a received business document or business signal. The notifier's BSI handles the receipt of failed business control exceptions similar to time-out exception. However, a failed business control exception does not decrease the *retry count*, but terminates the *business transaction* immediately. Hence, instead of decreasing

the *retry count* within the fault handler, we re-throw the exception beyond the boundaries of the *business transaction*.

EX-3: The notifiee does not answer in time

EX-3 is detected locally and leads to the same behavior as EX-1

The notifier's BSI detects a time-out exception if the notifiee does not provide a business document or business signal within the agreed time frame. Time-outs may be detected by the notifier in steps F3, G3, and/or H3. If so, the *throw activity* following the respective *delay activity* raises a *time-out detected exception* to be caught by a fault handler attached to the *event handling scope* in step D. Within the fault handler, a *send activity* communicates the time-out exception to the notifiee and the *retry count* is decremented. As a consequence of activating the fault handler, the activity flow inside step D is canceled and the condition of the *while activity* (B) gets re-evaluated.

EX-4: Syntactic or semantic document checks fail

Failed document checks result in a failed business control exception

If the notifier's BSI is not able to process a business document or a business signal sent by the notifiee, he has to trigger a failed business control exception. Within the notifier's BSI a failed business control exception may be thrown by the custom activities `check ack receipt` (F2), `check ack processing` (G2), or by validation procedures of the business application invoked by the *call external method* activity in step J1. A failed business control exception is not caught within the *while activity* (B), but by the outermost sequence that encloses steps A to H. This is due to the fact that a failed business control exception terminates all attempts to execute a *business transaction*.

EX-5: The retry count is exceeded

An exceeded retry count results in a failed business control exception

The condition of the *while activity* (B) does not evaluate to true any longer, if either the *business transaction* has ended (indicated by the internal boolean variable `businessTransactionFinished`) or if the *retry count* has been exceeded due to time-out exceptions. In the latter case, the *business transaction* is evidently not successful, but requires the communication of a failed business control exception to the notifiee. Consequently, before the *business transaction* eventually ends, the notifier has to check in step H if the *retry count* is exceeded or not. If exceeded, the *throw activity* H2 raises a failed business control exception, which is handled similar as in case EX-4.

8.3 Final assessment

In this section we introduced a UMM to Windows Workflow binding. The generated business service interface in .NET is in fact ready to compile. Before executing the workflow, the following tasks remain: (i) Declarative configurations of service endpoints; (ii) Hosting the workflow; (iii) Binding its internal interfaces to a business application.

In section 7 we outlined the deployment of UMM *business transactions* to the Web Services stack and BPEL, respectively. Both, the presented WF approach and the BPEL approach result in business service interfaces following the concepts of SOA. BPEL's evident advantage is being a jointly developed industry standard. Software vendors, however, have already started to implement their own platform-specific extensions to BPEL in their tools. This limits the portability of BPEL of being a cross-platform standard. A major advantage of WF is the fact that the BSI may directly execute business logic. BPEL, however, is limited to the orchestration of services. Consequently, business logic - even if not intended to be re-used - has to be exposed as a service. Furthermore, WF supports WS-* specifications like WS-Security or WS-ReliableMessaging out of the box by declaratively adapting endpoint configurations.

9 An e-Business Registry supporting the 3-Layer Approach

In the previous sections we have elaborated on a modeling approach for designing B2B integration solutions. When applying our modeling approach, several modeling artifacts on three layers are created. These artifacts comprise business models, inter-organizational business process models, and eventually deployment descriptions for services, which reflect the process models. In order to enable a successful discovery of potential business partners, these artifacts have to be publicly available.

Taking into account the three layers of B2B integration, one will first look for a partner who offers a required service on the economic level and who supports a complementary role in a choreography, before binding to its IT services. Inasmuch, a registry for inter-organizational systems should cover all three levels and maintain the dependencies between them.

The current registry architecture has not yet proven to be useful

In reality, however, most business partners meet off-line and have to undergo a cumbersome co-ordination process to align their IT-systems. The current registry-based approach does not take up because it is purely IT-driven and follows a bottom-up strategy. Each company develops its own interfaces to its own IT-system and the choreography that fits its proprietary needs. The resulting interfaces described for example by WSDL and BPEL are registered. In order to find a potential business partner one must look up the choreography and perform a difficult match-making process to check whether the other partner's choreography is compliant to one's own. However, if the choreographies have been developed in isolation, it is rather unlikely that they will match. Even, if the choreographies match on a technical level, there is still the risk of incompatibilities on the economic level, because the business functionality is hidden in the technical details.

e-business registries must reflect the three perspectives on e-business partnerships

For these reasons we state that the current main-stream approach to establish business partnerships online must be reconsidered. Therefore, we propose an approach for an e-business registry that takes three perspectives of e-business partnerships into account: (1) the economic perspective (business models), (2) the global choreography perspective (business process models), and (3) the IT perspective (deployment artifacts). We motivate the three perspectives for an e-business in sub-section 9.1. In sub-section 9.2 we introduce the registry meta-model on top of the ebXML registry information model (ebRIM) that manages the artifacts on each level and keeps their dependencies. Finally, we give a final assessment in section 9.3.

9.1 Motivating Business Scenario

Our approach to establish business partnerships online is strongly motivated by ebXML [79]. One may argue that ebXML's relevance is fading due to major industries accepting Web Services as industry standard on the technical level. However, in contrast to Web Services, ebXML had a pure e-business focus from its beginning. Thus, we feel that the basic ebXML scenario in figure 9.1 is still relevant, independent of the technologies used to implement this scenario.

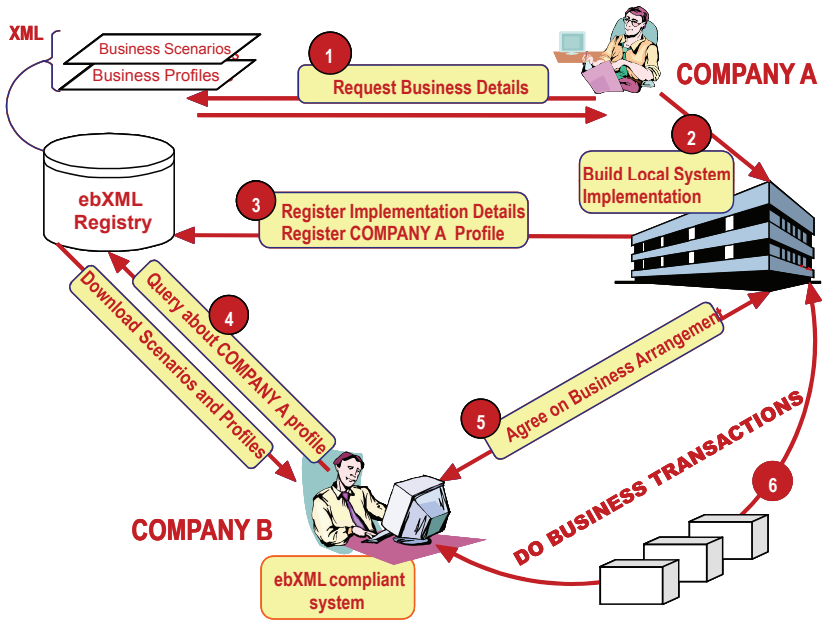


Figure 9.1 *ebXML Scenario [79]*

In this scenario, company *A* wants to perform a certain business scenario. In step 1 it downloads the details of this business scenario. Up to our interpretation as of today, this means it first downloads a computation independent model describing the business functionality and business value associated with a certain scenario. Taking into account our waste management example, we assume that company *A* is a `waste exporter` and looks for scenarios in the corresponding domain. Amongst others it finds the business scenario of our waste movement process.

The idea behind ebXML is still valid for e-business partnerships

If a company recognizes a business value in a certain scenario, it will download a specification of the choreography of business document exchanges that realizes the business functionality. In our example, company *A* sees a value in acting as a `waste exporter`. Thus, it will download the choreographies for interacting with an `export authority`. This completes step 1 of the ebXML scenario.

In step 2 a company builds its local system implementation. This does not necessarily mean that it has to implement a new information system from scratch. However, it has to set up the interfaces to its local information system as required by the global choreography and specifies the local choreography for this interfaces (which may be derived from the global one). Company *A* will specify the inter-

Local implementations are built in accordance with the global choreography

faces required for a waste exporter and the local choreographies to interact with an export authority.

Both, global and local choreographies may be managed within a registry

Having its system ready, the company will register its services in step 3. Company *A* announces in step 3 that it is ready to act as a waste exporter in a waste movement scenario. This means it registers its WSDL files describing the technical details of its services. Next, it registers a BPEL file describing how its services interact with an export authority. So far, company *A* has registered its local service with the export authority. To advertise the fact that its local service is compliant to what is expected from an export authority in the commonly defined global choreography, it establishes a link from its local service to the corresponding global choreography within the registry.

This link facilitate the search in the registry. In step 4 company *B* is looking for a potential business partner. We assume that company *B* is performing the role of an export authority in the waste movement scenario. Thus, it queries the registry for business partners who support the same global choreography (i.e. the one of waste movement) and who are able to act as complementary role (i.e. as waste exporter). Accordingly, company *B* will find company *A* as potential business partner. Since company *A* has established a link in the registry between the global choreography and its local service, company *B* is able to navigate this link in order to get all the technical details for binding its services to the ones of *A* in the following steps.

Standardization efforts are crucial for the success of e-business

This motivating scenario is based on the existence of well accepted business scenarios and global choreographies. Such scenarios and global choreographies may be developed by standardization organizations or industry consortia. It is also supposable that they are being developed by single organizations like market leaders. In fact, it does not matter who is going to develop them. It is up-to the market to decide which scenarios become well accepted and which ones won't. It is also important to note that there will be most likely multiple global choreographies that may realize the same business scenario such as waste movement. Some will get accepted, others won't.

Evidently, our motivating scenario refers to three different levels required in an e-business registry. These levels correspond to the three layers of our UML-based modeling approach for B2B integration. The first one is the business scenario describing the economic value. In our approach we use e^{3}value to describe this level (cf. section 4).

On the second level, the e-business registry has to manage models for global choreographies. Consequently, the proposed registry meta model has to support UMM 2.0 artifacts, which we introduced in section 5.

Finally, the third level specifies the deployment artifacts that implement the business service interfaces in order to support business collaborations. In this thesis, we elaborated on the deployment of UMM models by means of Web Services (section 7) and Windows Workflow (section 8). Since both deployments are managed similarly

in our e-business registry, we only demonstrate the registration on this level by means of Web Services artifacts - i.e., BPEL and WSDL files.

9.2 The e-business registry meta model

The proposed e-business registry is based on ebRIM

In this section we discuss how the artifacts created by our three-layered approach are managed in a registry. For this purpose we propose a registry meta model based on the ebXML registry information model (ebRIM) [72] that supports the specifics of the artifacts on the three layers. Our e-business registry meta model serves the purpose of defining which artifacts are maintained in the registry.

An ebXML registry stores these artifacts as *extrinsic objects*, which are XML files in our case, but may also be binaries in a general case. The content of the *extrinsic object* is encapsulated - this means a query to the registry does not access the content of an *extrinsic object*. Thus, an *extrinsic object* must be associated with appropriate meta data to allow an effective search. Our e-business registry meta model defines appropriate meta data for each of the artifacts on the three levels. Furthermore, the different artifacts and their meta data have dependencies on each other. The e-business registry meta model defines the required links between the *extrinsic objects* of the different artifacts and also between their meta data if appropriate.

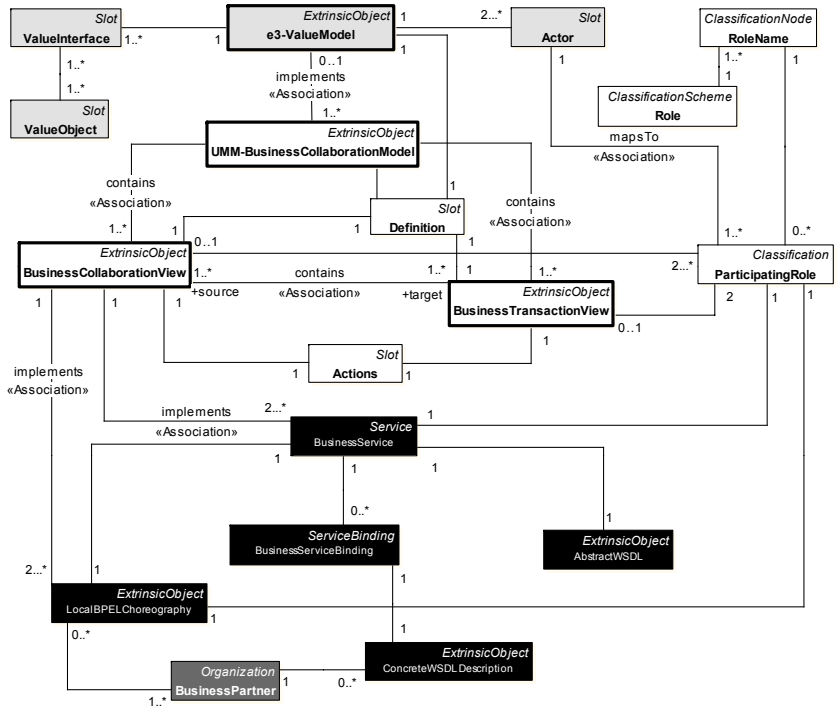


Figure 9.2 *e-business registry meta model*

The resulting meta model is depicted in figure 9.2. *Extrinsic objects* representing the various artifacts are shown with a thick border. The classes referring to the value layer of e^{3}value are presented with gray background. The ones managing the global choreography

of UMM are depicted with white background. Finally, the ones relating to the deployment layer are presented with black background. Each class of our e-business registry model is based on an existing meta class of ebRIM. The corresponding ebRIM meta class is denoted in the upper right corner of each class.

9.2.1 Registering e^{3}value models

The UML profile for e^{3}value enables the interchange of e^{3}value models based on XMI

In section 4 we introduced a UML representation for e^{3}value. Figure 9.3 depicts again the e^{3}value model for the waste movement scenario based on the use case variant of the proposed UML profile. For interchanging UML models, the XML Metadata Interchange (XMI) format is approved by the OMG. This XML-based representation is used to store the e^{3}value model as an extrinsic object in the e-business registry.

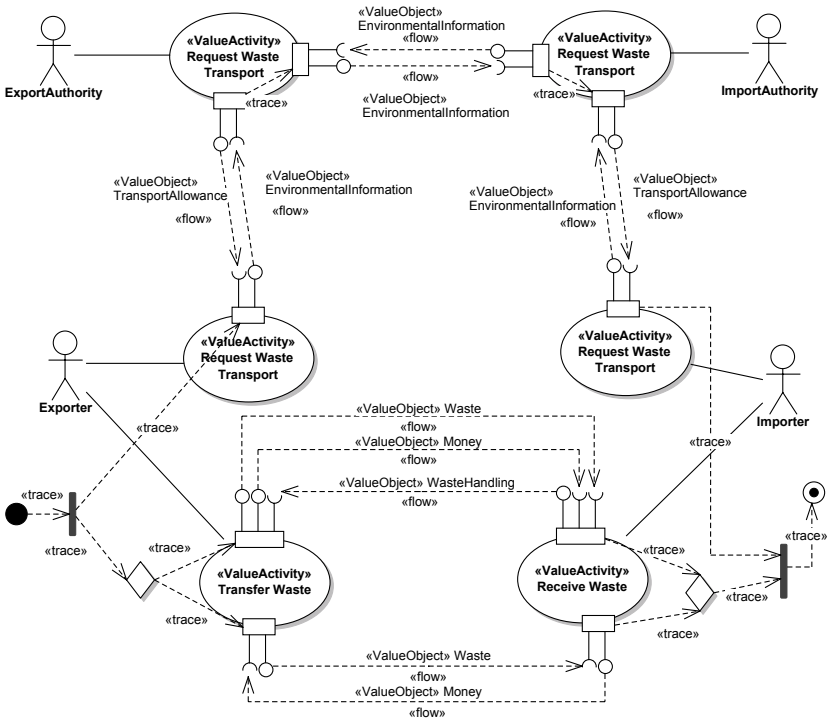


Figure 9.3 *Waste movement business scenario modeled using the UML profile for e^{3}value*

In a next step we have to decide on the meta data to accompany the e^{3}value model. Evidently, the definition capturing the business justification of an e^{3}value model is a kind of meta data. Additionally, the participants of a business network as well as the exchanged value objects may be of interest when searching for potential business partners.

Relevant information is extracted from the model and annotated as meta data

Correspondingly, we extract this information from an e^{3}value model and use it for annotating the `e3-Value model`. The definition as well as each actor of the `e3-Value model` are represented by their own *slots* and connected to the `e3-Value model`. In a similar way, we map value interfaces and the exchanged value objects to slots in our e-business registry meta model. In ebRIM, the concept of *slots* is used to add arbitrary meta data to registry objects.

9.2.2 Registering UMM models

Similar to the e^{3}value model, the UML-based UMM model is represented in XMI, too. In [32], we discuss in detail how to represent a UMM 1.0 model by means of XMI. Representing a UMM 2.0 model in XMI requires only trivial adaptations and is therefore not further detailed. The XMI representation of a UMM model is used to store it as an extrinsic object - `UMM – business collaboration model` - in the registry.

The e-business registry supports the registration of whole UMM models or parts thereof

In order to foster re-use of UMM artifacts, our approach does not only support registration of a whole UMM model, but also of parts thereof. As we outlined in section 5, UMM business transactions and business collaborations may be re-used across different UMM models. Hence, we consider the storage of business collaboration views and business transaction views as self-contained units. Each of them is represented by its own extrinsic object. Figure 9.4 shows sample artifacts for the process level of the e-business registry already known from our waste management example.

Business transactions and business collaborations are extracted from a UMM model and stored as self-contained entities

A business registry that implements the e-business registry meta model has to extract the XMI of the respective parts of a UMM model and has to store them as separate entities. For keeping the relationships between the UMM model and its business collaborations and business transactions, associations between a `UMM – business collaboration model` and its `business collaboration views` and `business transaction views` must be created. A slot named `Actions` is assigned to each `business collaboration view` and each `business transaction view` for textually describing the flow of actions. Furthermore, we introduce a slot called `Definition` to capture the purpose and the business justification of a UMM model.

Role names are indexed as meta data for search purposes

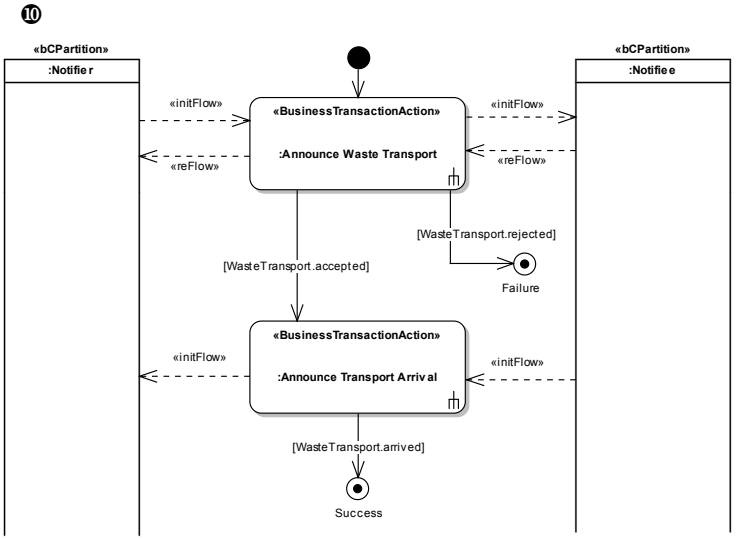
In addition to textual descriptions of process flows, the authorized roles in a business process are candidates for being stored as meta data. Accordingly, we link each `business collaboration view` and each `business transaction view` with their corresponding authorized roles. Since business transactions describe binary relationships, exactly two `participating roles` are connected to a `business transaction view`. Since UMM business collaborations may also capture multi-party processes, two or more `participating roles` may be connected to a `business collaboration view`.

Role names may be standardized by UN/CEFACT in the future

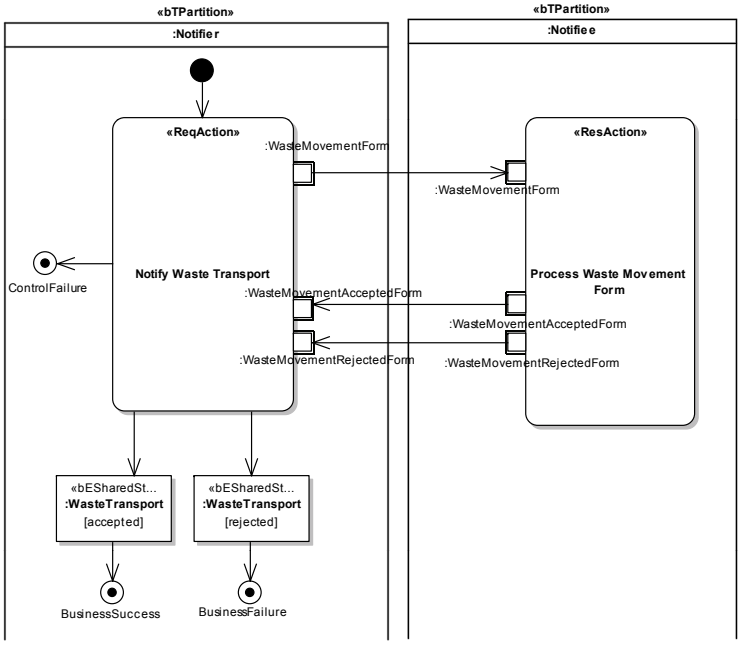
The roles that participate in a standardized UN/CEFACT business collaboration may be named according to a given scheme. In the future, UN/CEFACT may identify a common set of role names for business collaborations. In order to represent such a taxonomy for roles in our e-business registry meta model, we employ the concepts of *classification scheme* and *classification nodes*. The classification `participating role` expresses that a given role takes part in a certain business collaboration or business transaction.

9.2.3 Linking e^{3}value and UMM models

In order to enable business partners to query for a business collaboration model that fulfills a certain business model, these types of models must be linked in a business registry. Accordingly, the `UMM –`



(a) *UMM 2.0 business collaboration protocol* `manage waste transport`



(b) *UMM 2.0 business transaction* `announce waste transport`

Figure 9.4
Examples for UMM 2.0 artifacts, which are stored on the second level of an e-business registry

`business collaboration model` and the `e3-value model` are connected with an *Association*. ebRIM requires that an association between two registry objects must have a certain type that identifies the type of association.

Associations in the meta model are typified

For the purpose of our e-business registry model two generic association types are sufficient: *implements* and *contains*. Since business models are implemented by business processes, we use the *implements* association between the entities `e3-value model` and `UMM - business collaboration model`. A *contains* associations is used to represent "part of" relationships between artifacts. For example, we establish a *contains* relationship between a `UMM - business col-`

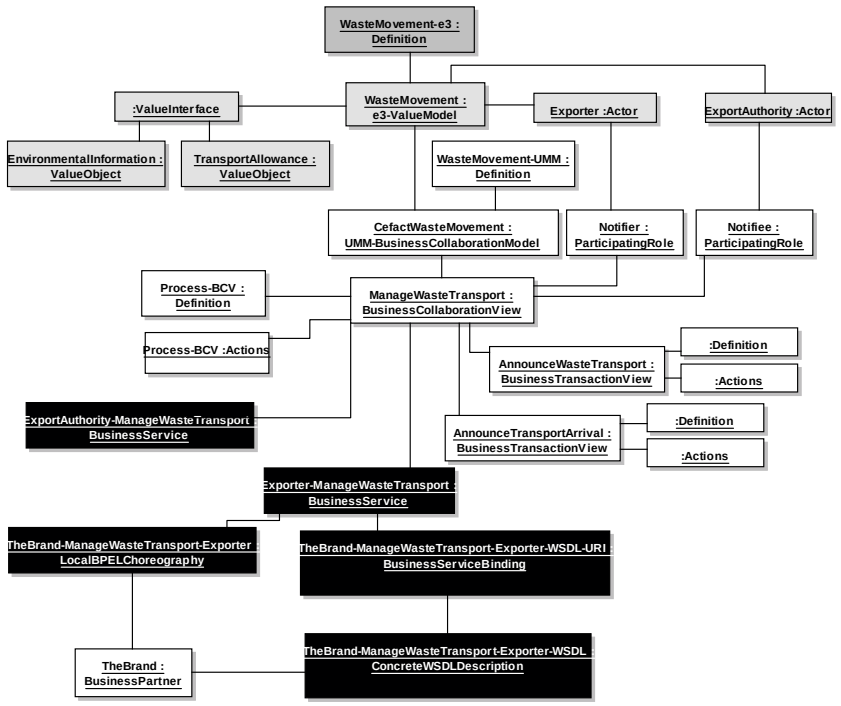


Figure 9.5
e-business registry example - waste management

laboration model and a business collaboration view or a business transaction view, respectively.

Moreover, as shown in figure 9.2 we introduce the association type mapsTo for connecting roles on the choreography level with actors on the value level. The mapsTo association indicates which role in the business collaboration is fulfilled by which actor in the value network - and vice versa.

9.2.4 Registering Web Services artifacts

We map deployment artifacts - demonstrated in this section by local BPEL choreographies and WSDL artifacts - to the e-business registry model and link them to the corresponding UMM models. The implementation of a global UMM collaboration requires commitments of the participating business partners in terms of service offerings. In other words, each participant has to implement his part of the process and has to expose this implementation as a service to his partners.

Collaboration models are linked to abstract WSDL definitions...

Correspondingly, a UMM business collaboration is implemented by two or more business services - one per each participant. ebRIM provides the generic type *Service* for representing business services in our e-business registry meta model. In order to foster the implementation of standardized business services, abstract WSDL definitions may be provided by UN/CEFACT. An abstract WSDL captures the implementation specifics of a service that has to be offered by a participant to support a certain business collaboration. Consequently, we establish a relationship between business service and

abstract WSDL in our e-business registry meta model as shown in figure 9.2.

... and local BPEL choreographies

In addition to abstract WSDL descriptions, UN/CEFACT may supply standardized local BPEL choreographies describing the flow of message exchanges with respect to each participant. We already know that a business collaboration in UMM may be either bi-lateral or multi-party. Thus, two or more local BPEL choreographies may implement the message exchange as described in a global business collaboration view.

9.2.5 Registering companies and their services

As motivated in sub-section 9.1 companies may present themselves within a business registry. In our e-business registry meta model we use the ebRIM type Organization to represent concrete companies. A set of additional ebRIM types like person name, postal address, email, etc. may be used to detail a company's profile. Since the usage of these aspects is considered as straight-forward we do not include them in our proposed e-business registry meta model.

Organizations link themselves to the service - i.e., business collaborations - they support

Once a company is registered it specifies which business collaborations it supports in which role. In our e-business registry meta model, this is simply specified by establishing a relationship between a business partner and a local BPEL choreography. Furthermore, companies may register concrete WSDL descriptions for the business services they support. A partner's concrete WSDL description is linked to a business service via the entity business service binding. The ebRIM concept of a service binding contains the URI, where an implementation of a service may be located. In the e-business registry meta model, the business service binding contains the URI of the business service that is offered by a certain company. The concrete WSDL description holds the required information (e.g., protocol bindings) to access the respective business service and to bind to it at run-time.

9.2.6 e-business registry example

Having introduced all concepts of our e-business registry example, we illustrated these concepts in figure 9.5 by means of our waste movement example scenario. An XMI equivalent of the e^{3}value model in figure 9.3 is stored as an extrinsic object in the registry. The meta data associated to this extrinsic object are the slots for its definition and for the actors exporter and export authority. Furthermore it is associated with a value interface. It covers environmental information in exchange for a transport allowance of waste. Note, the overall example would consist of more actors and value interfaces, but we show only those registry elements that are meaningful for a collaboration between the exporter and the export authority to keep the model simple.

The e^{3}value model is associated with the XMI representation of a UMM business collaboration model for waste movement. The meta data of this business collaboration model includes slots for its definition. Furthermore, the business collaboration view manage waste

transport between the notifier and the notifiee as well as the business transaction views being part of this collaboration are extracted and stored as separate extrinsic objects. Mappings between the e^{3}value actors exporter/export authority and the corresponding UMM participating roles are established. The corresponding relationships are maintained in the registry. Furthermore, slots for the definitions and actions containing the meta data of these views are created.

The business collaboration view manage waste transport covers a global choreography described from a neutral perspective. The deployment artifacts on each business partner's side are local choreographies described from the corresponding partner's perspective. Accordingly, there is a business service for the manage waste transport process for the exporter and another one for the export authority.

A waste exporter registers itself as a business partner. It furthermore registers its local choreography for acting as an exporter. This local choreography is composed of operation calls specified in a (or even more) WSDL file(s). The company also registers the WSDL file. In order to advertise the fact that its interfaces meet the requirements of the overall process it links the BPEL file and the WSDL file via a business service binding to the business service of the exporter. Thereby, its local implementation is also bound transitively to the global choreography of the business collaboration view and, finally, to the e^{3}value model.

9.3 Final assessment

In this section we have shown a three level approach for an e-business registry meta model combining the economic, the process choreography, and the service implementation layers. Based on ebRIM, a meta model for the storage and retrieval of artifacts in an ebXML registry has been developed. Using the concept of ebRIM *slots* we have built a meta model which facilitates the search for modeling and technical artifacts. The modeler can either search for e^{3}value artifacts on the business level or for UMM business transaction and collaboration artifacts on the process model level. On the technical level the user can retrieve WSDL and BPEL information for existing business process models.

The proposed registry approach is in accordance with the three layered modeling approach

The proposed registry model is in line with the overall three-layered model-driven approach for B2B integration described in this thesis. It allows potential business partners to find each other online by taking into account the three perspectives of an e-business partnership. In fact, the e-business registry suggested in this section complements the overall contribution of this thesis in the field of B2B integration.

10 Summary and open Issues

In this thesis, we proposed a UML-based approach for the design and implementation of B2B information systems. Our approach spans the three perspectives of relevance for B2B implementations: (i) the economic perspective captured using business models, (ii) the choreography of interactions described by business process models, (iii) the IT perspective resulting in deployment artifacts.

Traditional bottom-up approaches have not worked out in the past

As the hype about SOA and service-orientation turned into a rational consideration of the new architectures and technologies, it has become evident that a bottom-up approach for inter-organizational systems does not work out. Such approaches start from the IT layer of a single enterprise and expect that all other business partners adjust to it. However, if business partners start to build up their B2B systems in isolation, the resulting implementations are not interoperable. This results from the fact, that each business partners has its own view on the business scenario and, in practice, the various subjective views on the same scenario do not match.

One contribution of this thesis is a UML profile for e^{3}value

The neutral perspective of our top-down approach is expected to deal with this integration problem: Business partners first come to an agreement on the economic perspective using business modeling techniques. In other words, they settle on a business model defining who is doing business with whom, and what objects of value are thereby changing ownership. For designing business models, we started off with e^{3}value since it currently gains the most acceptance in this field. As one of the contributions of this thesis, we proposed a UML profile for e^{3}value in order build our approach on a single notation.

The modeled business network serves as the starting point for the next step in our top-down B2B integration approach: The agreement about the choreography of interactions between the business service interfaces on behalf of the partner's internal business applications. The resulting agreement is stipulated by a business collaboration model. Again, we decided to build upon an already accepted method instead of developing yet another process modeling approach. Since we co-author UMM within UN/CEFACT, the manifest decision was to start off with UMM on the process modeling layer.

Findings of these thesis were contributed to the standardization efforts of UN/CEFACT

As one of the key contributions of this thesis, we make several improvements to the UMM. First, we migrated the UML profile to the recent UML version, which was strongly requested by stakeholders. Thereby, when working on the UMM we discovered some flaws and also several potential advancements, which we framed based on our findings. The result of this work, has flown back into the standardization efforts of UN/CEFACT resulting in version 2.0 of UMM,

which has currently the status of a draft for implementation verification [113].

Another key contribution of this thesis is the mapping to deployable artifacts

Beside on enhancing UMM as a modeling language, which corresponds more or less to changes to the meta model, we concentrated on the deployment of business processes modeled in UMM. For this purpose, the graphical model has to be transformed to a language understandable for machines. In other words, we developed mappings to XML-based process specifications, which are used to configure process engines according to the business scenario.

We showed that the UMM model cannot be transformed to a single process specification since a business collaboration involving two or more business partners cannot be executed by a single engine. Instead each partner's business service interface requires its own process specification specifying only those interactions, which are relevant for the partner. Consequently, the transformation algorithm requires that the global model is sliced to one process specification per business partner.

The transformation algorithm was developed by considering the required behavior of business service interfaces that adhere to UMM business transaction semantics. We defined the formal behavior of UMM business service interfaces by means of UML state machines. The state machine serves as a blueprint for the transformation of UMM models to concrete deployment platforms. In this thesis, we focused on two deployment platforms for demonstrating our approach: Web Services and Windows Workflow.

The contribution of the e-Business registry meta model complements our modeling approach

Finally, the created artifacts must be made publicly available to allow potential business partners to find each other based on their business capabilities. In a B2B environment, capabilities are codified using modeling artifacts. Therefore, we proposed an e-business registry, which is capable of managing the artifacts created throughout our three-level modeling approach. The e-business registry is a central site for finding information about e-business scenarios like a search engine is a central point for getting information about web sites.

We opted for UML instead of BPMN as the underlying notation

In this thesis we concentrated on the UML. Recently, the Business Process Modeling Notation (BPMN) gained a lot of attraction for modeling business processes (cf. section 2) and, thus, would be a candidate for being used on the business process layer. However, we did not consider the BPMN in our approach for the following reasons: Firstly, the BPMN in version 1.2 [83] is not as customizable as required by our approach. For example, there are no concepts in the BPMN to represent the concepts of the business model layer. Secondly, BPMN already provides its own concepts for modeling inter-organizational processes, which we consider as not sufficient to model all the required aspects of B2B collaborations. Thirdly, we selected the UMM as one of the core technologies of our approach. The concepts of UMM are already captured in a UML profile. Thus, the decision to use UML as the underlying modeling language was straightforward.

The availability of UML as an underlying uniform notation facilitates communication among stakeholders, reduces possible incon-

sistencies and conflicts, and hence speeds up the entire development process. Evidently, the IT artifacts of the third layer have to be specified in the language of the corresponding target platforms. Hence, no UML representation was defined for them.

There are still some open issues

Although this thesis proposes an integrated approach for designing B2B systems, there are several issues that are not covered in this work. For the sake of completeness we identify and outline these issues in the following. This summary may serve as a starting point for further research in order to extend and/or complement our work.

(i) Internal processes

This work concentrates on realizing B2B collaborations. In other words, we focus on the collaborative space between business partners only, but not on the internal processes of an enterprise. Deploying business collaborations - i.e., business service interfaces - requires to bind them to the business processes that are internal to a company. In our approach, we do not consider such an integration. Furthermore, an enterprise may not be willing to re-engineer their existing internal business processes or even establish new ones in order to support newly developed business collaborations. Thus, existing internal business processes may have to be considered when developing business collaborations in order to integrate them accordingly. Taking existing business processes into account corresponds to a so-called "middle-out" approach, whereas we propose a "top-down" approach in this thesis. Adapting our work to a "middle-out" approach would require an appropriate method for capturing internal processes during the early phases of the business collaboration design for a proper integration with existing and new processes.

(ii) Business documents

Within this thesis we emphasized on a very process-centric perspective on B2B interactions. This means we focused on the flow of interactions between business partners and how to come to an agreement thereon. However, we only briefly discussed the modeling of business information that is conveyed by these interactions. As stated in the motivation of this thesis, there existed a pure document-centric view on the information exchange between enterprises for a while. For the successful realization of B2B collaborations, both perspectives go together and should be considered as equally important: Business processes are modeled according to the requirements of a business scenario. Then, the business process models (as well as the underlying business scenario) helps to identify the business information that must be communicated in each interaction of the business process. This procedure allows to ascertain the minimum required information in order to reach the next step within the business process. Since business information modeling is a topic as complex as business process modeling, elaborating both perspectives in detail would go beyond the scope of this thesis.

(iii) Transformation of business collaboration protocols

Deriving deployable artifacts by applying model transformation techniques to global UMM choreography models is one of the main goals of this thesis. In this thesis we emphasized on the transformation of so-called business transactions, which are the basic building blocks of global UMM choreographies. In UMM, the concept of a

business transaction implements the requirements and semantics of e-business transactions. Thus, transforming business transactions is the most challenging step in deriving deployable artifacts from UMM models. As we know, however, a global UMM choreography (i.e., a business collaboration protocol) may consist of more than one business transaction. In this case, each business transaction as well as the flow between them has to be transformed to the respective IT platform. Unlike business transactions, business collaboration protocols do not always follow the same pattern, but may be defined more or less as an arbitrary graph. In practice, however, most business collaboration protocols incorporate only basic control flow patterns [118] like sequence, parallel split, synchronization, exclusive choice, or simple merge. In this case, mapping business collaboration protocols as a flow of business transactions can be considered as straightforward. The transformation of business collaboration protocols incorporating more complex patterns like for example "arbitrary cycles" is still subject to research.

(iv) Consistency between business models and business process models

By describing the derivation process of deployment artifacts from business process models we almost closed the gap between these two layers in our approach. We transformed the UML-based process models to executable code and XML-based process specifications. In order to bridge the business model layer with the business process model layer, we proposed a UML profile for e^{3}value. However, we did not emphasize on model transformations from business models to business process models. Furthermore, it is evidently necessary that modeling artifacts on those layers are in consistency with each other, i.e., that a business process model meets the requirements of a business model. However, the definition of consistency constraints between those layers were not part of this thesis. Instead, we refer the interested reader to the works in [132] and [91], respectively, which discuss consistency checking between e^{3}value and UML activity diagrams.

Barring these unresolved issues, we are convinced that the thesis at hand contributes to the research area of B2B electronic commerce. The elaborated approach may support enterprises that conduct B2B as well as tool vendors to produce effective and successful solutions in order to realize B2B transactions as they have been envisioned for years.

List of Figures

Bibliography

[1] P. Andrew, J. Conard, S. Woodgate, J. Flanders, G. Hatoun, I. Hilerio, P. Indurkar, D. Pilarinos, and J. Willis. *Presenting Windows Workflow Foundation*. Sams, 1. Auflage, 9 2005.

[2] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D. Martin, D. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. *DAML-S: Web Service Description for the Semantic Web*. In *The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 348–363. Springer Berlin / Heidelberg, 2002.

[3] Hofreiter B., Huemer C., and Kim J.-H. *Choreography of ebXML business collaborations*. *Information Systems and E-Business Management*, 4(3):221–243, July 2006.

[4] D. Bell. *The Coming Of Post-industrial Society (Harper Colophon Books)*. Basic Books, 7 1976.

[5] T. Berners-Lee and J. Hendler. *Scientific publishing on the semantic web*. *Nature*, 410:1023–1024, 2001.

[6] R. J. A. Buhr. *Use case maps as architectural entities for complex systems*. *IEEE Transactions on Software Engineering*, 24(12):1131–1155, December 1998.

[7] C. Bussler, D. Fensel, and A. Maedche. *A conceptual architecture for semantic web enabled web services*. *SIGMOD Rec.*, 31(4):24–29, 2002.

[8] H. Chesbrough and J. Spohrer. *A research manifesto for services science*. *Commun. ACM*, 49(7):35–40, 2006.

[9] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. *The Web Service Modeling Language WSML: An Overview*. In *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 590–604. Springer Berlin / Heidelberg, 2006.

[10] T.S. Dillon, C. Wu, and E. Chang. *Reference Architectural Styles for Service-Oriented Computing*. In *Network and Parallel Computing*, volume 4672/2008 of *Lecture Notes in Computer Science*, pages 543–555. Springer Berlin / Heidelberg, 2008.

[11] J. Dorn, C. Grün, H. Werthner, and M. Zapletal. *From business to software: a B2B survey*. *Information Systems and E-Business Management*, 7(2):123–142, March 2009.

[12] P.F. Drucker. *The Age of Discontinuity Guidelines to Our Changing Society*. Harper and Row Publishers, 1968.

[13] A. Fatolahi and F. Shams. *An investigation into applying UML to the Zachman framework*. *Information Systems Frontiers*, 8(2):133–143, February 2006.

[14] D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[15] D. Fensel and C. Bussler. *The web service modeling framework wsmf*. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[16] D. Fensel, S. Decker, M. Erdmann, and R. Studer. *Ontobroker: The Very High Idea*. In *Proceedings of the Eleventh International Florida Artificial Intelligence Research Society Conference*, pages 131–135. AAAI Press, 1998.

[17] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[18] A. Friesen and K. Namiri. *Towards semantic service selection for B2B integration*. In *ICWE '06: Workshop proceedings of the sixth international conference on Web engineering*, page 17, New York, NY, USA, 2006. ACM.

[19] R.J. Glushko, J.M. Tenenbaum, and B. Meltzer. *An XML framework for agent-based E-commerce*. *Commun. ACM*, 42(3):106–ff., 1999.

[20] J. Gordijn. *The e3-value toolset*. http://www.e3value.com.

[21] J. Gordijn and H. Akkermans. *Designing and evaluating e-business models*. *IEEE Intelligent Systems*, 16(4):11–17, Jul–Aug 2001.

[22] J. Gordijn and H. Akkermans. *Does e-business modeling really help?* In *Proc. 36th Annual Hawaii International Conference on System Sciences*, page 10pp., January 2003.

[23] J. Gordijn, H. Akkermans, and H. van Vliet. *Business Modelling Is Not Process Modelling*. In *ER '00: Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling*, pages 40–51, London, UK, 2000. Springer-Verlag.

[24] J. Gordijn and J. M. Akkermans. *Value-based Requirements Engineering - Exploring Innovative e-Commerce Ideas*. *Requirements Engineering Journal*, 8(2):114–134, 2003.

[25] J. Gordijn, A. Osterwalder, and Y. Pigneur. *Comparing two Business Model Ontologies for Designing e-Business Models and Value Constellations*. In *Proceedings of the 18th BLED conference (e-Integration in Action)*. University of Maribor, 2005.

[26] J. Gordijn, M. Petit, and R. Wieringa. *Understanding Business Strategies of Networked Value Constellations Using Goal- and Value Modeling*. In *Proc. of the 14th IEEE International Conference Requirements Engineering*, pages 129–138, 11–15 Sept. 2006.

[27] The Open Group. *The Open Group Architecture Framework*. Van Haren Publishing, 2006. TOGAF 8.1.1.

[28] M. Hepp and R. Dumitru. *An Ontology Framework for Semantic Business Process Management*. In *Proceedings of the 8th International Conference Wirtschaftsinformatik 2007*, pages 423–440. Universitaetsverlag Karlsruhe, March 2007.

[29] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. *Semantic business process management: a vision towards using semantic Web services for business process management*. In *Proc. IEEE International Conference on e-Business Engineering ICEBE 2005*, pages 535–540, October 12–18, 2005.

[30] B. Hofreiter and C. Huemer. *Transforming UMM Business Collaboration Models to BPEL*. In *OTM Workshops*, volume 3292 of *LNCS*, pages 507–519. Springer, 2004.

[31] B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, and M. Zapletal. *UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce*. In *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops*, volume 4231 of *LNCS*. Springer, 2006.

[32] B. Hofreiter, C. Huemer, and M. Zapletal. *Registering UMM Business Collaboration Models in an ebXML Registry*. In *Proc. 3rd IEEE International Conference on E-Commerce Technology The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services*, pages 45–45, 2006.

[33] P. Hruby. *Model-Driven Design Using Business Patterns*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[34] C. Huemer. *EUDIN: European Data Interchange for Waste Notification Systems - Ein Fall für UN/CEFACT Standards*. , University of Vienna and Research Studios Austria, 2005. eGov Days 2005 of the Austrian Computer Society, in German.

[35] C. Huemer, P. Liegl, R. Schuster, and M. Zapletal. *B2B Services: Worksheet-Driven Development of Modeling Artifacts and Code*. *The Computer Journal*, 52(1), 2009.

[36] M. Ilger and M. Zapletal. *An Implementation to transform Business Collaboration Models to executable Process Specifications*. In *Proceedings of the conference on Service-Oriented Electronic Commerce at the Multikonferenz Wirtschaftsinformatik 2006, Passau Germany*, pages 9–23. GI-Edition - Lecture Notes in Informatics (LNI), 2006.

[37] ISO. *Open-edi Reference Model*, 2004. ISO/IEC JTC 1/SC30 ISO Standard 14662, Second Edition, `http://standards.iso.org/ittf/PubliclyAvailableStandards/c037354_ISO_IEC_14662_2004(E).zip`.

[38] J.-Y. Jung, W. Hur, S.-H. Kang, and H. Kim. *Business process choreography for B2B collaboration*. *IEEE Internet Computing*, 8(1):37–45, January 2004.

[39] R. Kalakota and M. Robinson. *Services Blueprint: Roadmap for Execution (Addison-Wesley Information Technology Series)*. Addison-Wesley Professional, 6 2003.

[40] H. Kim. *Conceptual modeling and specification generation for B2B business processes based on ebXML*. *SIGMOD Rec.*, 31(1):37–42, 2002.

[41] B. Korherr and B. List. *Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL*. In *Proceedings of the ER-Conference on Conceptual Modeling (Workshop BP-UML'06)*, volume 4231 of *LNCS*, pages 7–18. Springer, November 2006.

[42] B. Korherr and B. List. *A UML 2 Profile for Event Driven Process Chains*. In *Proceedings of the 1st IFIP International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2006)*. Springer, 2006.

[43] P. Kotler and K. L. Keller. *Marketing Management*. Prentice Hall, Englewood Clis, NJ, March 2005.

[44] A. Kotok and David R.R. Webber. *ebXML: The New Global Standard for Doing Business on the Internet*. Sams, 1st. Auflage, 8 2001.

[45] G. Kramler, E. Kapsammer, G. Kappel, and W. Retschitzegger. *Towards Using UML 2 for Modelling Web Service Collaboration Protocols*. In *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05)*, pages 227–238. Springer, February 2006.

[46] R. Lara, D. Roman, A. Polleres, and D. Fensel. *A Conceptual Comparison of WSMO and OWL-S*. In *Web Services*, volume 3250 of *Lecture Notes in Computer Science*, pages 254–269. Springer Berlin / Heidelberg, 2004.

[47] R.M. Lee. *Documentary Petri Nets: A Modeling Representation for Electronic Trade Procedures*. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 359–375, London, UK, 2000. Springer-Verlag.

[48] K. Lenz and A. Oberweis. *Inter-organizational Business Process Management with XML Nets*. In *Petri Net Technology for Communication-Based Systems*, volume 2472, pages 243–263. Springer, 2003.

[49] P. Liegl. *Conceptual Business Document Modeling using UN/CEFACT's Core Components*. In *Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM2009)*. Australian Computer Society, January 2009.

[50] P. Liegl, R. Schuster, and M. Zapletal. *A UML Profile and Add-In for UN/CEFACT's Modeling Methodology*. Master's thesis, University of Vienna, feb 2006.

[51] F.-R. Lin, M.-C. Yang, and Y.-H. Pai. *A generic structure for Business Process Modeling*. *Business Process Management Journal*, 8(1):19–41, 2002.

[52] S. Ling and S.W. Loke. *Advanced Petri Nets for modelling mobile agent enabled interorganizational workflows*. In *Proc. Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pages 245–252, 8–11 April 2002.

[53] B. List and B. Korherr. *A UML 2 Profile for Business Process Modelling*. In *Proc. ER 2005 Workshops*, volume 3770 of *LNCS*. Springer, 2005.

[54] B. List and B. Korherr. *An evaluation of conceptual business process modelling languages*. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1532–1539, New York, NY, USA, 2006. ACM.

[55] F. Machlup. *The Production and Distribution of Knowledge in the United States*. Princeton University Press, 1 1973.

[56] P.P. Maglio and J. Spohrer. *Fundamentals of service science*. *Journal of the Academy of Marketing Science*, 36(1):18–20, March 2008.

[57] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. *Bringing Semantics to Web Services: The OWL-S Approach*. In *Semantic Web Services and Web Process Composition*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42. Springer Berlin / Heidelberg, 2005.

[58] R. Mayer, C. Menzel, M. Painter, P. deWitte, T. Blinn, and B. Perakath. *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report*. AL-TR-1995-XXXX, Knowledge Based Systems, Inc., September 1995.

[59] R. J. Mayer, P. C. Benjamin, B. E. Caraway, and M. K. Painter. *A Framework and a Suite of Methods for Business Process Reengineering*. *Business Process Reengineering: A Managerial Perspective*, pages 245–290, 1995.

[60] W. E. McCarthy. *The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment*. *The Accounting Review*, 57(3):554–578, July 1982.

[61] S.A. McIlraith and T.C.H. Zeng. *Semantic web services*. *IEEE intelligent systems*, 16(2):46–53, 2001.

[62] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid. *Composing Web services on the Semantic Web*. *The VLDB Journal*, 12(4):333–351, November 2003.

[63] J. Mendling and M. Hafner. *From Inter-organizational Workflows to Process Execution: Generating BPEL from WS-CDL*. In *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, volume 3762/2005 of *Lecture Notes in Computer Science*, pages 506–515. Springer Berlin / Heidelberg, 2005.

[64] J. Mendling, G. Neumann, and M. Nüttgens. *Yet Another Event-Driven Process Chain*. In *3rd International Conference on Business Process Management (BPM 2005)*, volume 3649, pages 428–433. Springer, 2005.

[65] J. Mendling and M. Nüttgens. *EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC)*. *Information Systems and E-Business Management*, 4(3):245–263, 2006.

[66] M. Merz. *E-Commerce und E-Business*. Dpunkt.Verlag GmbH, 9 2001.

[67] Microsoft. *Windows Communication Foundation (WCF)*. `http://msdn.microsoft.com/en-us/netframework/aa663324.aspx`.

[68] Microsoft. *Windows Workflow Foundation (WF)*. `http://msdn.microsoft.com/en-us/netframework/aa663328.aspx`.

[69] T. Murata. *Petri nets: Properties, analysis and applications*. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[70] OASIS. *Collaboration-Protocol Profile and Agreement Specification*, September 2002. Version 2.0, `http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf`.

[71] OASIS. *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*, 2004. `http://www.oasis-open.org/committees/wss/`.

[72] OASIS. *ebXML Registry Information Model*, May 2005. Version 3.0, `http://www.oasis-open.org/committees/regrep/`.

[73] OASIS. *WS-Reliability 1.1*, November 2005. Web Services Reliable Messaging TC, OASIS Standard, `http://docs.oasis-open.org/wsrm/ws-reliability/v1.1`.

[74] OASIS. *ebXML Business Process Specification Schema Technical Specification*, December 2006. OASIS Standard, Version 2.0.4, `http://docs.oasis-open.org/ebxml-bp/2.0.4/ebxmlbp-v2.0.4-Spec-os-en.html`.

[75] OASIS. *Reference Model for Service Oriented Architecture*, October 2006. OASIS Standard, Version 1.0, `http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf`.

[76] OASIS. *OASIS ebXML Messaging Services*, July 2007. Version 3.0: Part 1, Core Features, `www.oasis-open.org/committees/ebxml-msg/`.

[77] OASIS. *Web Services Business Process Execution Language*, April 2007. Version 2.0, `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[78] OASIS. *Web Services Reliable Messaging (WS-ReliableMessaging)*, January 2008. OASIS Standard incorporating Approved Errata, Version 1.1, http://docs.oasis-open.org/ws-rx/wsrm/v1.1/wsrm.pdf.

[79] OASIS, UN/CEFACT. *ebXML - Technical Architecture Specification*, February 2001. Version 1.4, http://www.ebxml.org/specs/ebTA.pdf.

[80] Object Management Group (OMG). *MDA Guide*, June 2003. Version 1.01, http://www.omg.org/docs/omg/03-06-01.pdf.

[81] Object Management Group (OMG). *Unified Modeling Language Specification*, April 2005. Version 1.4.2, also available as ISO/IEC 19501, http://www.omg.org/spec/UML/ISO/19501/PDF/.

[82] Object Management Group (OMG). *Unified Modeling Language: Superstructure*, February 2007. Version 2.1.1, http://www.omg.org/spec/UML/2.1.1/Superstructure/PDF/.

[83] Object Management Group (OMG). *Business Process Modeling Notation Specification*, January 2009. Version 1.2, http://www.omg.org/spec/BPMN/1.2/PDF.

[84] A. Osterwalder and Y. Pigneur. *An e-Business Model Ontology for Modeling e-Business*. In *Proceedings of the 15th Bled Electronic Commerce Conference*, June 2002.

[85] A. Osterwalder, Y. Pigneur, and C.L. Tucci. *Clarifying business models: Origins, present, and future of the concept*. *Communications of the Association for Information Systems*, 16(25):1–25, 2005.

[86] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. *Service description: A survey of the general nature of services*. , Centre for Information Technology Innovation, Queensland University of Technology, Australia, 2000. report FIT-TR-2003-02.

[87] C. Ouvans, M. Dumas, A.H.M. ter Hofstede, and W.M.P. van der Aalst. *From BPMN Process Models to BPEL Web Services*. In *Proc. International Conference on Web Services ICWS '06*, pages 285–292, Sept. 2006.

[88] A.G. Pateli and G.M. Giaglis. *A Framework for Understanding and Analysing e-Business Models*. In *Proceedings of 16th Bled Electronic Commerce Conference: eTransformation, Kranj: Moderna Organizacija*, pages 329–348, 2003.

[89] C. Pedrinaci, J. Domingue, C. Brelage, T. van Lessen, D. Karastoyanova, and F. Leymann. *Semantic Business Process Management: Scaling Up the Management of Business Processes*. In *Proc. IEEE International Conference on Semantic Computing*, pages 546–553, August 4–7, 2008.

[90] M. Penker and H.-E. Eriksson. *Business Modeling With UML: Business Patterns at Work*. Wiley, 2000.

[91] V. Pijpers and J. Gordijn. *Consistency Checking Between Value Models and Process Models: A Best-of-Breed Approach*. In *Proceedings of BUSITAL*, volume 336 of *CEUR Workshop Proceedings*, pages 58–72. CEUR-WS.org, 2008.

[92] M.E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, 1. Auflage, 6 1998.

[93] S. Roser and B. Bauer. *A categorization of collaborative business process modeling techniques*. In *Proc. Seventh IEEE International Conference on E-Commerce Technology Workshops*, pages 43–51, 19 July 2005.

[94] RosettaNet. *RosettaNet Implementation Framework: Core Specification*, December 2002. V02.00.01, http://www.rosettanet.org/cms/sites/RosettaNet/Standards/RStandards/rnif/.

[95] N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and P. Wohed. *On the suitability of UML 2.0 activity diagrams for business process modelling*. In *APCCM '06: Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*, pages 95–104, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[96] A.-W. Scheer. *Aris-Business Process Modeling*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[97] R. Schuster and T. Motal. *From e3-value to REA: Modeling multi-party eBusiness Collaborations*. In *Proc. of the IEEE International Conference on Commerce and Enterprise Computing*. IEEE Computer Society, July 2009.

[98] J. Sherwood, A. Clark, and D. Lynas. *Enterprise Security Architecture: A Business-Driven Approach*. cmp, 11 2005.

[99] E. Sirin, J. Hendler, and B. Parsia. *Semi-automatic Composition of Web Services using Semantic Descriptions*. In *In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, pages 17–24, 2002.

[100] E. Sirin, B. Parsia, and J. Hendler. *Filtering and Selecting Semantic Web Services with Interactive Composition Techniques*. *IEEE Intelligent Systems*, 19(4):42–49, 2004.

[101] E. Söderström, B. Andersson, P. Johannesson, E. Perjons, and B. Wangler. *Towards a Framework for Comparing Process Modelling Languages*. In *CAiSE '02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 600–611, London, UK, 2002. Springer-Verlag.

[102] J. Spohrer, P. P. Maglio, J. Bailey, and D. Gruhl. *Steps Toward a Science of Service Systems*. *Computer*, 40(1):71–77, January 2007.

[103] E. Staudt. *Die mobile Gesellschaft*. *Information Age Economy*, 5. Internationale Tagung Wirtschaftsinformatik:15–28, 2001.

[104] Supply Chain Council. *Supply-Chain Operations Reference-model - SCOR*, 2006. Version 8.0, http://www.supply-chain.org.

[105] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. *Automated discovery, interaction and composition of semantic web services*. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):27–46, 2003.

[106] P. Timmers. *Business Models for Electronic Markets*. *Electronic Markets*, 8(2):3–8, July 1998.

[107] P. Timmers. *Electronic Commerce: Strategies and Models for Business-to-Business Trading*. Wiley, 1. Auflage, 4 2001.

[108] P. Traverso and M. Pistore. *Automated Composition of Semantic Web Services into Executable Processes*. In *The Semantic Web - ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*, pages 380–394. Springer Berlin / Heidelberg, 2004.

[109] UN/CEFACT. *UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module*, March 2006. Technical Specification V1.0, http://www.unece.org/cefact/umm/UMM_Foundation_Module.pdf.

[110] UN/CEFACT International Trade and Business Process Group (TBG14). *UN/CEFACT Common Business Process Catalog*, September 2005. Version 1.0, Technical Specification, http://www.uncefactforum.org/TBG/TBG14/TBG14Deliverables.htm.

[111] UN/CEFACT Technologies and Methodologies Group (TMG). *Core Components Technical Specification (CCTS)*, November 2003. Version 2.01, http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf.

[112] UN/CEFACT Technologies and Methodologies Group (TMG). *UML Profile for Core Components (UPCC)*, January 2008. Version 1.0, http://www.untmg.org/upcc/spec/1_0.

[113] UN/CEFACT Technologies and Methodologies Group (TMG). *UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module*, January 2009. Candidate for V2.0, Draft for Implementation Verification, http://www.untmg.org/umm/spec/foundation/2_0.

[114] W.M.P. van der Aalst. *The Application of Petri Nets to Workflow Management*. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[115] W.M.P. van der Aalst. *Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets*. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[116] W.M.P. van der Aalst. *Making Work Flow: On the Application of Petri Nets to Business Process Management*. In *ICATPN '02: Proceedings of the 23rd International Conference on Applications and Theory of Petri Nets*, pages 1–22, London, UK, 2002. Springer-Verlag.

[117] W.M.P. van der Aalst and A.H.M. ter Hofstede. *YAWL: yet another workflow language*. *Information Systems*, 30(4):245–275, 2005.

[118] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Workflow Patterns*. *Distributed and Parallel Databases*, 14(1):5–51, July 2003.

[119] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske. *Business Process Management: A Survey*. In *Business Process Management*, volume 2678/2003 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003.

[120] T. Vitvar. *Advances in e-business collaboration: semantic Web services in SAP R/3 B2B integration*. In *Proc. International Symposium on Collaborative Technologies and Systems*, pages 290–297, May 20–20, 2005.

[121] T. Vitvar, M. Moran, M. Zaremba, A. Haller, and P. Kotinurmi. *Semantic SOA to Promote Integration of Heterogeneous B2B Services*. In *Proc. 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services CEC/EEE 2007*, pages 451–456, July 23–26, 2007.

[122] Web Services Interoperability Organization. *Web Services Interoperability Basic Profile (WS-I)*, April 2006. Version 1.1, http://www.ws-i.org/Profiles/BasicProfile-1.1.html.

[123] H. Werthner and M. Borovicka. *E-commerce und Semantic Web*. In *Semantic Web*, X.media.press, pages 307–319. Springer Berlin Heidelberg, 2006.

[124] World Wide Web Consortium (W3C). *OWL-S: Semantic Markup for Web Services*, November 2004. W3C Member Submission, http://www.w3.org/Submission/OWL-S.

[125] World Wide Web Consortium (W3C). *OWL Web Ontology Language*, February 2004. W3C Recommendation, http://www.w3.org/TR/owl-features/.

[126] World Wide Web Consortium (W3C). *Resource Description Framework (RDF)*, February 2004. W3C Recommendation, http://www.w3.org/RDF/.

[127] World Wide Web Consortium (W3C). *Web Services Choreography Description Language*, November 2005. Version 1.0, W3C Candidate Recommendation, http://www.w3.org/TR/ws-cdl-10/.

[128] WSML Working Group. *The WSML Specification*, August 2008. D16, http://www.wsmo.org/TR/d16/.

[129] WSMO Working Group. *Web Service Modeling Ontology (WSMO)*, October 2006. D2v1.3, http://www.wsmo.org/TR/d2/v1.3/.

[130] J. Zachman. *A Framework for Information Systems Architecture*. *IBM Systems Journal*, 31(3):445–470, 1999.

[131] V. Zeithaml, M. J. Bitner, and D. D. Gremler. *Services Marketing*. McGraw-Hill/Irwin, NY, May 2005.

[132] Z. Zlatev and A. Wombacher. *Consistency Between e3-value Models and Activity Diagrams in a Multi-perspective Development Method.* In *Proceedings of the OTM Workshops 2005*, volume 3760 of *LNCS*. Springer, 2005.

Acknowledgments

Die Verfassung dieser Arbeit wäre ohne die direkte oder indirekte Unterstützung vieler Personen aus meinem Umfeld nicht möglich gewesen.

Meinen beiden Betreuern, Hannes Werthner und Christian Huemer, gebührt besonderer Dank für die Zeit und Geduld, die sie in den letzten 3 Jahren aufgebracht haben. Beide hatten stets in offenes Ohr für meine Anliegen. Besonders betonen möchte ich, dass man bei ihnen nicht “nur ein Mitarbeiter“ ist, sondern dass sie sich stets um meine Belange gekümmert und mir viel ermöglicht haben.

Ein großes Danke auch an Philipp Liegl und Rainer Schuster. Nachdem wir bereits die Masterarbeit gemeinsam verfasst haben, sind wir auch im Doktoratsstudium ein Team geblieben. Sie haben zu dieser Arbeit sowohl direkt wie auch indirekt beigetragen. Die Tage im Büro waren stets erheiternd und manchmal sogar kabarettreif - was uns bei der Arbeit aber nicht gebremst, sondern eher motiviert hat. Danke auch an Birgit Hofreiter für interessante Anregungen, Tips und Hilfestellungen. Die ursprüngliche “UMM star alliance” hat sich in den letzten Monaten und Jahren zum “BSopt Team” entwickelt - danke an dieser Stelle auch an Thomas Motal für interessante Diskussionen. Ebenso danke an Gerti Kappel, ohne deren Zutun ich meine Bewerbung bei Hannes Werthner nicht doch noch in letzter Minute abgeschickt hätte.

Es freut mich an dieser Stelle sagen zu können, dass aus dieser Konstellation in den letzten Jahren auch Freundschaften entstanden sind.

Besonderer Dank gebührt auch meiner Familie - besonders meiner Mutter, meinem Vater, und meinem Stiefvater, die mich während des Studiums nicht nur finanziell sondern auch motivierend und emotional unterstützt haben. Dies gilt auch für meine Großeltern, die mich - soweit es ihnen möglich war - auf meinem Weg unterstützt haben. Zum Abschluss möchte ich meiner Freundin Bernadette danken, die in den letzten Monaten meiner Dissertation viel Verständnis bewiesen hat.

Curriculum Vitae

Curriculum Vitae: Marco Zapletal

Technische Universität Wien
Institut für Softwaretechnik und Interaktive Systeme
Electronic Commerce Group
Favoritenstrasse 9-11/188
A-1040 Wien, Österreich

e-mail: marco@ec.tuwien.ac.at
tel: +43/1/588 01/18822
fax: +43/1/588 01/18899
mobil:+43/676/305 23 21
web: `http://www.ec.tuwien.ac.at/~marco/`

Curriculum Vitae: Marco Zapletal

Akademischer Grad	Dipl.-Ing. Mag.rer.soc.oec Bakk.rer.soc.oec
Persönliche Daten	*Geburtsdatum / Geburtsort:* 27. April 1981, Wien *Familienstand:* ledig

Ausbildung

Technische Universität Wien — April 2006 - Juni 2009

Doktoratsstudium in Wirtschaftsinformatik (Notendurchschnitt: 1,0)

- Titel der Dissertation: *A UML-based Methodology for Model-Driven B2B Integration: From Business Values over Business Processes to Deployment Artifacts*

Technische Universität Eindhoven, Niederlande — Feb. 2009 - April 2009

Forschungsaufenthalt an der Fakultät für Wirtschaftskunde & Informatik, Abteilung Architekturen für Informationssysteme

Technische Universität Wien — Sep. 2004 - März 2006

Doppelstudium:

- Masterstudium *Software Engineering & Internet Computing* (Notendurchschnitt: 1,3; mit Auszeichnung bestanden)
  Abschluss: Dipl.-Ing.
- Masterstudium *Wirtschaftsinformatik* (Notendurchschnitt: 1,3)
  Abschluss: Mag.rer.soc.oec.
- Titel der Masterarbeit: *A UML Profil and Add-In for UN/CEFACT's Modeling Methodology*; Die Arbeit definiert ein formales Vorgehensmodell sowie eine Modellierungsmethode zur Erfassung von inter-organisationalen Geschäftsprozessen (B2B).

Universität Wien / Technische Universität Wien — Sep. 2000 - Juli 2004

Bakkalaureatsstudium *Wirtschaftsinformatik* (Notendurchschnitt gesamt: 2,2; im Schwerpunkt Electronic Commerce: 1)
Titeln der beiden Bakkalaureatsarbeiten:

- *Aufbau einer IT-Organisation unter Berücksichtigung von Sourcing Varianten für verschiedene Firmengrößen und Firmentypen*
- *Mapping Business Documents Independent of their Syntax*

Bundesrealgymnasium Bernoullistraße, Wien — Juni 1999

Matura mit gutem Erfolg bestanden

Auszeichnungen

Siemens Forschungsstipendium — Jan. 2009

Verleihung eines Dissertationsstipendiums der Firma Siemens für einen Forschungsaufenthalt an der Technischen Universität Eindhoven.

Best PhD Proposal Award — Aug. 2008

Auszeichnung für das beste PhD Proposal bei der 10th International Conference on Electronic Commerce 2008 (Innsbruck, Österreich).

INiTS Award 2006 — Okt. 2006

Ausgezeichnet mit dem INiTS Award für die Masterarbeit - 1.Platz in der Kategorie Informations- und Kommunikationstechnologie. Der INiTS Award wird in Österreich für innovative Masterarbeiten mit großer Chance auf wirtschaftliche Verwertbarkeit verliehen.

Weiterbildung

Sun Certified Programmer for the Java 2 Platform — Sep. 2004

University of Oulu, Oulu, Finnland — März 2003

European Intensive Programme on Information and Communication Technologies Security (IPICS 2003), Note 1

Saint Michaels College, Vermont, USA — Juni 1998

School for International Studies, Sprachaufenthalt

Berufliche Tätigkeiten

Universitätsassistent und Doktorand — seit April 2006

Institut für Softwaretechnik und Interaktive Systeme, TU Wien

Forschung für: Schnelle und effiziente Einführung von Business-to-Business (B2B) Systemen in Unternehmen beliebiger Größe mittels modellgetriebener Softwarentwicklung sowie standardisierter Geschäftsprozesse. Publikationsleistung:

- 1 Buchveröffentlichung
- 1 Buchkapitel (in Erscheinung)
- 3 Journalbeiträge
- 21 referierte Publikationen bei anerkannten Konferenzen (davon 4 in Erscheinung)

Enterprise Architect — April 2007 - April 2008

T-Mobile Austria, Wien

Etablierung eines Business Process Management Compentence Centers bei T-Mobile Austria; Ausarbeitung und Weiterentwicklung von Architektur Guidelines; Mitarbeit in IT-Architekturprojekten.

IT-Berater — Mai 2003 - Nov 2006

Paradigma Unternehmensberatung GmbH, Wien

Beratertätigkeit in den Bereichen Geschäftsprozessmodellierung und Service-orientierte Architekturen; unter anderem beratend im EU Projekt GILDAnet (Global Integrated transport Logistic DAta NETwork) tätig.

Projektmitarbeiter und Diplomand — März 2005 - März 2006
Dept. of Distributed and Multimedia Systems, Univ. Wien
Verfassung der Masterarbeit innerhalb des Forschungsprojekts 'eGov-Simple' in Partnerschaft mit den Austrian Research Centers GmbH (ARC). Im Rahmen der Masterarbeit wurde eine Software entwickelt, die seitdem im australischen e-Government im Einsatz ist (Project GovDex).

Analytiker — Juni 2004 - April 2005
Sozialversicherung d. gewerblichen Wirtschaft, Wien
Technische Konzeptentwicklung, Design, Architektur und Implementierung des 'Arzneimittelbewilligungsservices' (ABS) zur effizienten Abwicklung der chefärztlichen Bewilligung in Österreich. Seit Jänner 2005 ist ABS bei allen österr. Sozialversicherungsträgern im Einsatz.

Softwareentwickler — Feb. 2004 - Mai 2004
Tibbet & Britten Austria, Wien
Implementierung einer Schnittstelle zur automatisierten Übermittlung von Überweisungsaufträgen von Tibbet & Britten zu deren Bank. Die Übermittlung der Daten erfolgt mittels des UN/EDIFACT Standards.

Softwareentwickler — April 2002 - Jän. 2003
Institut f. Informatik u. Wirtschaftsinformatik, Univ. Wien
Projekt 'A-XML': Konzept und Implementierung eines ebXML (Electronic Business XML) Prototypen zum inter-organisationalen elektronischen Datenaustausch. A-XML galt als Pionierprojekt im Bereich B2B für das Jahr 2001.

Administrative Tätigkeiten

Mitglied der Studienkommission Wirtschaftsinformatik, TU Wien — Mai 2004 - März 2006

Studienrichtungsvertreter für Wirtschaftsinformatik, TU Wien — Juli 2003 - Juni 2005

Mitgliedschaften

United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT)
- Vollmitglied der 'Technology and Methodology (TMG)' Group
- Chair der 'Business Process Working Group' innerhalb der TMG

OCG (Österreichische Computer Gesellschaft)

IEEE (Institute of Electrical and Electronics Engineers, Inc.)

Fremdsprachen

Englisch - fließend in Wort und Schrift, Publikationsniveau

Französisch - grundlegende Kenntnisse

Latein - fortgeschrittene Kenntnisse

Grundwehrdienst

Jagdpanzerbataillon 1, Wiener Neustadt — Sept. 1999 - April 2000

Persönliche Interessen

Individual- und Städtereisen, Skifahren, Squash, Ausgehen

List of Publications

The up-to-date list of publications is found on `http://www.ec.tuwien.ac.at/~marco`.

Books

- *UN/CEFACT's Modeling Methodology (UMM) 1.0 - A Guide to UMM and the UMM Add-In.* With: Philipp Liegl and Rainer Schuster, Verlag Dr. Müller, ISBN-13: 978-3836467704, 2008

Book chapters

- Service-Oriented Enterprise Modeling and Analysis. With: Christian Huemer, Philipp Liegl, Rainer Schuster, and Birgit Hofreiter. In: Handbook of Enterprise Integration, Auerbach Publications, 2009

Journal Papers

- *B2B Services: Worksheet-Driven Development of Modeling Artifacts and Code.* With: Christian Huemer, Philipp Liegl, and Rainer Schuster. In: The Computer Journal, 52-1, Oxford University Press, 2009 (doi:10.1093/comjnl/bxn076)
- *From Business to Software: A B2B Survey.* With: Jürgen Dorn, Christoph Grün, and Hannes Werthner. In: Information Systems and E-Business Management - Special Issue on Design and management of business models and processes in services science, Springer, 2008
- *Modeling e-Government processes with UMM.* With: Philipp Liegl, Rainer Schuster, Christian Huemer, Birgit Hofreiter, and Robert Mosser. Informatica Journal - An International Journal of Computing and Informatics, XXXII:1, 2007

Conference and Workshop Papers

- *State of the art in electronic business document standards: A survey.* With: Philipp Liegl, Christian Pichler, and Michael Strommer. Submitted to the 43th Hawaii International Conference on System Sciences (HICSS), IEEE CS, 2010
- *An Analysis of Windows Workflow's Control-Flow Expressiveness.* With: Wil M.P. van der Aalst, Nick Russell, Philipp Liegl, and Hannes Werthner. Submitted to the European Conference on Web Services (ECOWS), IEEE CS, 2009

- *Towards a global business document reference ontology.* With: Philipp Liegl and Christian Huemer. Submitted to the Third IEEE International Conference on Semantic Computing (ICSC), IEEE CS, 2009
- *Deriving business service interfaces in Windows Workflow from UMM transactions.* In: Proceedings of Service-Oriented Computing - ICSOC 2008, 6th International Conference on Service-Oriented Computing, Springer LNCS, 2008
- *The Development Process of the UN/CEFACT Modeling Methodology.* With: Christian Huemer, Philipp Liegl, Thomas Motal, and Rainer Schuster. In: Proceedings of the Tenth International Conference on Electronic Commerce (ICEC08), ACM, 2008
- *A Holistic Methodology for Model-Driven B2B Integration.* Doctoral Consortium at the Tenth International Conference on Electronic Commerce (ICEC08), CEUR-WS, 2008 - *awarded as the best Student Paper*
- *A 3-level e-Business Registry Meta Model.* With: Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Proceedings of the IEEE Intl. Conf. on Services Computing (SCC2008), IEEE Computer Society, 2008
- *A UML Profile for the e3-Value e-Business Model Ontology.* With: Christian Huemer, Alexander Schmidt, and Hannes Werthner. In: Proceedings of the 3rd Intl. Workshop on Business/IT Alignment and Interoperability (BUSITAL) @ 20th Intl. Conf. on Advanced Information Systems Engineering (CAiSE'08), CEUR-WS, 2008
- *Inter-organizational Systems: From Business Values over Business Processes to Deployment.* With: Christian Huemer, Philipp Liegl, Rainer Schuster, and Hannes Werthner. In: Proceedings of the 2nd Intl. IEEE Conf. on Digital Ecosystems and Technologies (DEST2008), IEEE Computer Society, 2008
- *Worksheet-Driven UMM Modeling of B2B Services.* With: Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Proceedings of the 2007 IEEE Intl. Conf. on e-Business Engineering (ICEBE), IEEE Computer Society, 2007
- *The UMM Add-In - Demo.* With: Birgit Hofreiter, Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Proceedings of the Intl. Conf. on Services Oriented Computing (ICSOC2007), Vienna (Austria), Springer LNCS, 2007
- *The Web Services-BusinessActivity-Initiator (WS-BA-I) Protocol: an Extension to the Web Services-BusinessActivity Specification.* With: Hannes Erven, Georg Hicker, Christian Huemer, and Marco Zapletal. In: Proceedings of the IEEE Intl. Conf. on Web Services (ICWS 2007), IEEE Computer Society, 2007
- *Deriving executable BPEL from UMM Business Transactions.* With: Birgit Hofreiter, Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Proceedings of the IEEE Intl. Conf. on Services Computing (SCC2007), IEEE Computer Society, 2007
- *Modeling Business Entity State Centric Choreographies.* With: Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Pro-

ceedings of the 9th IEEE Conf. on E-Commerce Technology (CEC07), IEEE Computer Society, 2007

- *A State Machine executing UMM Business Transactions.* With: Christian Huemer. In: Proceedings of the Inaugural IEEE International Digital EcoSystems Technologies Conference 2007 (IEEE-DEST2007), IEEE Computer Society, 2007
- *A Survey of B2B Methodologies and Technologies: From Business Models towards Deployment Artifacts.* With: Jürgen Dorn, Christop Grün, and Hannes Werthner. In: Proceedings the 40th Hawaii International Conference on System Sciences (HICSS), IEEE Computer Society, 2007
- *UN/CEFACT's Modeling Methodology (UMM): A UML Profile for B2B e-Commerce.* With: Birgit Hofreiter, Christian Huemer, Philipp Liegl, and Rainer Schuster. In: Proceedings of the 2nd International Workshop on Best Practices of UML (ER BP-UML'06), Springer LNCS, 2006
- *A Business Collaboration Registry Model on Top of ebRIM.* With: Birgit Hofreiter and Christian Huemer. In: Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06), IEEE Computer Society, 2006
- *Registering UMM Business Collaboration Models in an ebXML Registry.* With: Birgit Hofreiter and Christian Huemer. In: Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), IEEE Computer Society, 2006
- *An Implementation to transform Business Collaboration Models to Process Specifications.* With: Michael Ilger. In: Service-Oriented Electronic Commerce, Proceedings of the Multikonferenz Wirtschaftsinformatik 2006 (MKWI06), GI LNI, 2006

Technical Reports

- *Pattern-based Analysis of Windows Workflow.* With: Wil M.P. van der Aalst, Nick Russell, Philipp Liegl, and Hannes Werthner. University of Eindhoven, The Netherlands, 2009
- *Deriving business service interfaces in Windows Workflow from UMM transactions - long version.* Vienna University of Technology, 2008

Master Thesis

- A UML Profile and Tool Support for UN/CEFACT's Modeling Methodology. With: Philipp Liegl and Rainer Schuster. University of Vienna, 2006. *The thesis was awarded with the INiTS Award 2006 in the category Information and Communication Technology*