14 Architectures for Cross-Enterprise Business Integration

Jörg P. Müller, Stephan Roser, and Bernhard Bauer

CONTENTS

INTRODUCTION

Today, many companies are organized in global networks and outsource activities that can be performed quicker, more effectively, or at lower cost by others [1]. Their competitiveness depends heavily on support systems that can keep up with constantly evolving business relationships and cross-organizational value chains. This requires methodologies, methods, software architectures, and infrastructures to support changes defined at a strategic level and propagate them down to the working levels in terms of business processes and associated information and communication technology (ICT) systems. One way to achieve this is by integrating the ICT systems within an enterprise and across networked enterprises.

The main objective of most integration initiatives is to achieve a new level of interoperability while minimizing the impact on existing ICT environments. According to Erl [2], this means the following:

- Avoiding the creation of a fragmented environment through the introduction of business logic that resides outside of established application boundaries
- Avoiding tightly bound integration channels between applications that are easily broken if either application is modified
- Minimizing redevelopment of applications affected by the integration

A discipline that plays a key role in this task is Enterprise Application Integration (EAI) [3]. The goal of EAI is to make sure that all corporate applications and application systems work together transparently to cover all business activities as if they were designed as one system from the start. Yet, today's EAI have limited robustness because changes in one part of the system are likely to have undesirable impact in other parts, high cost for implementation and maintenance, and the lack of principled end-to-end methodologies and methods to deal with the challenge of cross-enterprise business integration. In fact, EAI solutions face, like the organizations themselves, the challenge of adapting quickly in response to changing requirements.

An essential means to making EAI solutions more robust, less costly, and more adaptive, is to foster and increase the reusability of proven architecture patterns [3] and technologies.

In Chapter 15, the authors consider traditional EAI as an *ex post* integration technology, whereas they regard the standardized service-oriented architecture (SOA)/Web Services technology stack an *ex ante* integration technology. We take a broader view of EAI and believe that with Model-Driven Software Development (MDSD) methods and SOA principles, EAI can be planned and performed *ex ante* rather than after the fact. In our view, SOA is foremost a distributed software architecture that can be used for a flexible and adaptive integration with enhancements such as MDSD and patterns that consider semantics and business functions.

A prerequisite for the seamless integration of cross-organizational business processes (CBPs), whether within the enterprise applications or across enterprises, is to embed EAI into a principled end-to-end overall development approach, taking all levels of abstraction of cross-enterprise collaboration into account. MDSD (see, e.g., [4][6]) is such an approach, consistent with the shift from program-based implementations toward model-driven implementation. MDSD, as a particular case of the new trend in software design of model-driven engineering (MDE) [4], provides techniques to realize and automate the propagation of changes at business level to the technical level.

The structure of this chapter is as follows. We first survey the state of the art in business integration architecture from a service-oriented perspective. Next, we outline the pillars and key architectural components of a model-driven EAI approach, and develop a taxonomy of cross-enterprise EAI approaches. In the following section, by detailing on the different levels of this taxonomy, we develop an MDSD solution that can be used as a reference and basis to realize EAI systems for concrete businesses. In particular, we identify and characterize three

generic service-oriented EAI topologies within this model-driven approach, with a focus on the platform-independent IT level. The chapter ends with conclusions and an outlook to future research challenges.

SERVICE ORIENTATION AND BUSINESS INTEGRATION

While we refer to Chapters 11 and 15 for a comprehensive overview of the state-of-the-art in the area of SOA, we shall discuss some important related work that investigates requirements and approaches for using SOA in the context of EAI, before we cover the related work on MDSD.

Service orientation is based on the concept of service, defined "as a well-defined, self-contained function that does not depend on the context or state of other services" [7]. SOA is an architecture paradigm for IT systems where functions are separated into distinct, loosely coupled units or *services* [8], accessible over a network in order that they can be combined and reused in the development of business applications [9]. The concept of services can be the basis for platform-independent models (PIMs); thus, service orientation can be used to make EAI more effective and adaptive, to provide flexible integration of IT applications and functions. Further down in this section, we introduce categories of service-oriented integration approaches and describe how they can be used to realize a model-driven EAI. Before doing so, however, we briefly survey MDSD.

Model-Driven Engineering

Software engineering is currently witnessing a paradigm shift from programming-based implementation toward model-driven implementation. This carries important consequences on the way information systems are built and maintained [4]. MDE raises the level of abstraction at which developers create and evolve software [6] by treating models as first class artifacts that can be used for representation as well as code generation. This reduces the complexity of software artifacts by separating concerns and aspects of a system [5]. Thus MDE shifts the focus of software development away from the technology toward the problem to be resolved. Largely automated model transformations refine (semi-)automatically abstract models to more concrete models or simply describe mappings between models of the same level of abstraction. In particular, transformation engines and generators are used to generate code and other target domain artifacts with input from both modeling experts and domain experts [12]. MDE is an approach to bridge the semantic gap between domain-specific concepts of applications and programming technologies used to implement them [13]. It provides a technical basis for automation and reuse in terms of generation techniques like model transformation and code generation as well as reusable assets like infrastructures, components, templates, and transformations. Two prominent representatives of MDE are the Object Management Group (OMG)'s Model-Driven Architecture® (MDA) and the software factory initiative from Microsoft.

In MDE, models and model transformations, which can also be treated as models, embody critical solutions and insights to enterprise challenges and hence are seen as assets for an organization [11]. Assets are artifacts that provide solutions to

problems, should be reusable in, and customizable to various contexts. Similarly, in MDSD, architectural and technology patterns are encoded in model transformations, enabling a partial automation and synchronization of modifications (e.g., resulting from process or structure evolutions) across modeling levels, as well as the possibility to extensively reuse assets.

MDSD can be used to provide end-to-end support for the realization of business processes, from the business level (users' view) down to deployed applications (ICT view) on specific platforms via well-defined, largely automated model transformations and refinements. MDSD treats models as primary development artifacts, uses models to raise the level of abstraction at which developers create and evolve software [8], and reduces the complexity of the software artifacts by separating concerns and aspects of a system under development [5].

TOPOLOGIES FOR CROSS-ENTERPRISE EAI

In the following, we introduce three categories of integration solutions on the basis of their topology, as previously described in the literature, see, for example, [2,14]. Our model-driven approach, however, is not restricted to these categories and be extended to new or other classifications. Figure 14.1 depicts the three integration solution topologies.

In a *fully decentralized (peer-to-peer, P2P)* topology, services (in the sense of self-contained functions) of the participating organizations implicitly establish the collaborative process through direct message exchange. Examples are *P2P* networks or multiagent systems. Changing in the business protocol would result in changing one or more peers. Furthermore, the interface and external behavior of the peers are directly exposed to the collaboration space and therefore are directly accessible by entities outside enterprise boundaries.

In a *hierarchical topology*, a controller service defines the steps necessary to achieve the overall goal and maps these steps to services provided by the contributing organizations. Messages exchanged among the services of the collaborating organizations through a central broker component. Typically, a broker realizes a controller service to act as a global observer process that coordinates the partners

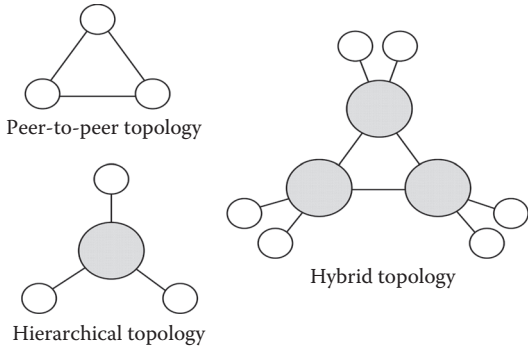

FIGURE 14.1 Integration topologies.

and makes decisions on the basis of data used in the collaboration. Changes to the protocol's messages and semantics would affect only the broker process. Since the broker is not necessarily owned by one of the participating partners, organizations may hide their elementary services from their collaborators. However, they have to reveal them to the broker.

In many cases, a mixture of hierarchical and the fully decentralized topology, that is, a *hybrid topology*, is used to realize complex multipartner collaborations [14]. Elements of the fully decentralized topology are introduced in the hierarchical topology and the controller service is distributed among several controller processes jointly providing the broker functionality. Each participating organization provides one controller service that orchestrates and encapsulates that organization's services. Messages that cross-organizational boundaries go through the controller services. Q1

In our experience, these three general topology patterns are general enough to model the EAI architectures or topologies normally occurring in practice (e.g., tree structures), either by variation or combination of these topologies.

TAXONOMY OF CROSS-ENTERPRISE EAI

The topologies depicted in Figure 14.1 are generic in a sense that they do not yet relate to specific IT platforms and technologies. In this section, we present a model-driven taxonomy of different EAI approaches. This taxonomy, illustrated in Figure 14.2,

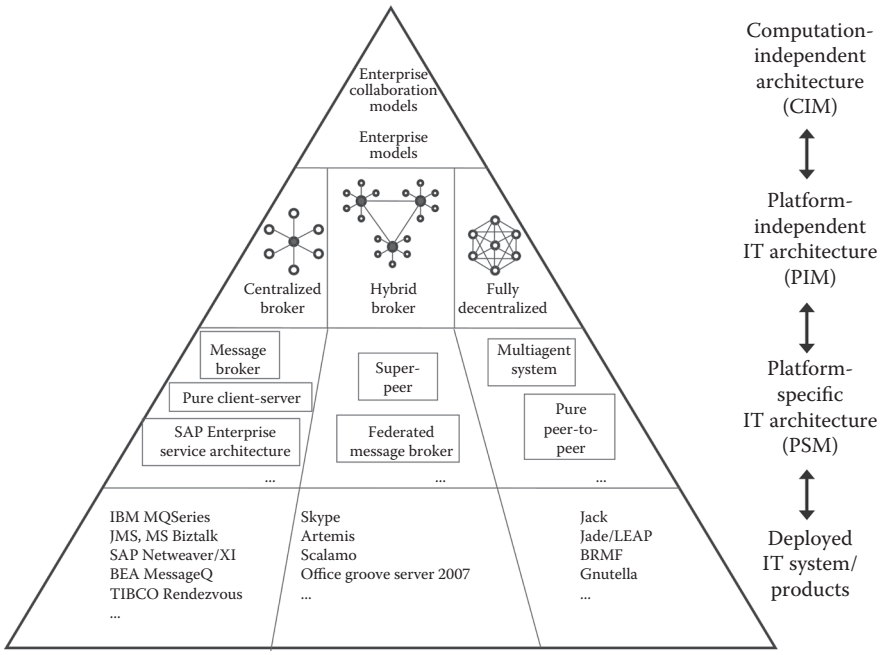

FIGURE 14.2 Classification of architectural approaches according to the model abstractions of MDA.

helps us organize the various topologies in a hierarchy according to the abstractions of the model-driven development. As such, the taxonomy provides a guideline for the structure of the remainder of this chapter.

The top-most level of the pyramid is the Computation-Independent IT Architecture level, also referred to as the Computation-Independent Model (CIM) level, following the MDA terminology. It contains business-level architectures and models of enterprises and their collaborations. There are many available methodologies at this level such as the ARchitecture for integrated Information Systems (ARIS) House of Business Engineering, the Zachman Framework, PERA (Purdue Enterprise Reference Architecture), GRAI (Graphs with Results and Actions Inter-related), CIMOSA (Computer-Integrated Manufacturing Open Systems Architecture), or GERAM (Generalized Enterprise Reference Architecture and Methodology) [15]. At the CIM level, we can represent an enterprise collaboration architecture as a set of enterprise models covering different aspects of the participating enterprises, such as organization, data, services, functions, and process flow plus a conceptual (non-IT-oriented) description of collaboration use cases between the enterprises, such as supply chain relationships. More explanations are available later (pp. 556 and ff.).

The second level of our taxonomy deals with Platform-Independent IT Architectures, abbreviated as PIM as per the MDA conventions, and which exclude the implementation details. In the context of this section, the architectures of interest follow the service-oriented paradigm without being tied down to a specific target platform.

Architectural decisions made at this level account for different communication and coordination topologies. For instance, a procurement process involving original equipment manufacturers (OEMs), suppliers, and potentially purchasing organizations (POs) acting as third-party service providers (see [16]) may be mapped into one of the three generic topologies. In a strictly hierarchical topology, each manufacturer has one purchasing unit that implements and controls interactions with its suppliers, whereas in a hybrid topology, not only competing, but also cooperating POs provide the manufacturer with access to various suppliers and their products, at different conditions and with different underlying business models. Finally, in a decentralized topology, suppliers provide direct access to the manufacturer and manage interactions locally without an intermediate purchasing unit.

Clearly, a business-level model can be mapped into one of these three architectural topologies using a PIM of an IT system even while it considers the communication and coordination patterns that support each of the three architecture paradigms.

The step from conceptual, platform-independent IT level down to specific target platforms is supported by the next lower abstraction level, the platform-specific IT architecture (PSM) level. For example, in the procurement process example above, a hierarchical solution could manifest itself as a message broker-based architecture at the central PO that facilitates the routing of messages and business documents between the manufacturer, the PO itself, and the suppliers. On the other hand, a federated message broker architecture approach [17], for example, based on web services technology, may be used to realize the interactions between different POs in the hybrid topology, that is, to provide a manufacturer partnering with one specific PO access to products available at one supplier only via a different PO. Finally, a structured *P2P* topology, such as described in Ref. [18], may provide a distributed

collaboration space, where OEMs can postrequests, identify potential suppliers, and pursue collaborations without a dedicated central element.

The lowest level describes how platform-specific models (PSMs) are mapped into actual deployed IT systems using dedicated platforms and products. For instance, in the hierarchical case, our central PO could host an SAP Netweaver-based application system using the SAP XI (eXchange Infrastructure) to facilitate the routing of messages and business documents between the OEMs, the PO itself, and the suppliers. In the second case, an IBM Websphere message broker-based federation of Enterprise Service Buses [19] may be used to implement the heterogeneous, federated message broker topology. Finally, a full decentralized information and collaboration space may be built up using the P2P business resource management framework [20] based on distributed hash table lookup protocols such as Chord [17].

METHODOLOGICAL ISSUES IN APPLICATION INTEGRATION WITH MDSD

In EAI systems, similarly structured solutions can be considered as members of the same software systems family (SSF) when they "... share enough common properties to be built from a common set of assets" [21]. This is the case for service-oriented EAI systems. Their shared functionalities are in services that form the common set of assets of the SSF. Other Functionalities that cannot be composed from this common set of assets, whether because they are new or are obtained from other that cannot be ICT systems, are then encapsulated as services and then integrated.

MDSD provides means to realize SSFs. It can be used to realize software product lines (SPLs), where each SPL consists of an SSF [22] whose members share some common functions in addition to having their specific functionalities [23].

MDSD, by itself, does not provide the methods and concepts to build SPLs and model-driven solutions. Such issues can be addressed by SPL engineering techniques [24], which distinguish two phases: building the MDSD solutions, called *domain engineering*, and applying the MDSD solution for EAI development called *application engineering.*

Domain engineering, which consists of domain analysis, domain scoping, and variability analysis, is used to obtain appropriate modeling languages for the CIM level. Reference implementations of the EAI solutions are used to factor out commonalities and variable aspects of the EAI solutions. These commonalities become part of the reusable components, such as transformations, generation templates, or frameworks. Transformations and templates consist of common code that is configured through the content of the models they are applied to. The model content is used to bind the variables of the transformations and templates to concrete values during transformation execution, that is, the execution of templates is triggered by the model content. The languages used to model EAI systems provide means to represent the variable parts of the EAI solutions. Variability is further realized through the choice of certain transformations and generators.

At application engineering time, the MDSD solution is applied. Using the four-level approach presented in Figure 14.2, this means application modeling and specification **Q2**

at the CIM level, the selection of integration architectures, that is, the appropriate CIM to PIM transformations and the application of model transformations and generators from PIM to PSM and further on to code level, takes place. After mapping of high- to lower-level models, the next step is to manually refine the generated models (PIM and PSM) by further narrowing down the variables in the solution.

MDSD solutions also allow the PIM-level model to be directly mapped to a textual representation ("code") without an explicit PSM layer. For instance, a transformation to the Web Services Business Process Execution Language (WS-BPEL) could provide either the WS-BPEL text files as an output (model-to-text transformation), or a model representing the WS-BPEL process description (model-to-model transformation). In the following, we will not distinguish between these two possibilities and use the term PIM-to-PSM to denote either of them.

REALIZATION OF EAI WITH MODEL-DRIVEN ENGINEERING

We now present an MDSD solution that can be used as a basis to realize EAI systems for concrete businesses. We describe and discuss our reference MDSD solution for EAI, its application, and provide insights in the decisions when building the solution.

Our MDSD solution for EAI follows the methodology depicted in Figure 14.2. In this approach, coarse-grained high-level models are transformed and manually refined to more fine-grained lower-level models. When enough information is available, the ICT system is generated. This top-down approach works extremely well when changes in the business model precede the development of the ICT system. Yet, it may be necessary to consider bottom-up development as well, for example, to take into account of legacy systems. Bottom-up design can be supported in one of two ways. The first option is to encapsulate the legacy systems by enlisting the domain engineer to describe their services and add them to the MDSD solution as reusable components or modeling elements in the domain-specific modeling language. The other possibility is for the application engineer to model the legacy systems in PIMs, as in classical reverse engineering (also called: architecture-driven modernization (ADM) [25]).

Another relevant scenario concerns the evolution of ICT systems and platforms. This affects the MDSD solution since transformations, frameworks, infrastructure, and so on may have to be modified or added at domain engineering time. Roser and Bauer [26] described an approach to cope with the evolution of models and model transformations. Further, existing models of systems may have to be adjusted at application engineering time.

In practice, one will mostly find and apply hybrid methodologies that combine top-down and bottom-up development aspects. In the following, we will focus on the top-down development aspects to describe the application of an MDSD-based EAI solution. In such a solution, one can find horizontal and vertical transformations. *Horizontal transformations* represent information via different model types at the same abstraction level. They improve the interoperability and exchange of models among the various enterprise modeling formats. *Vertical transformations* implement mappings for higher- to lower-level models, typically mapping one element in the higher-level model into several elements in the lower-level model.

Changes to one model in the model-driven development framework (Figure 14.2) often percolate to other models and at other abstraction levels than where the change originated. To support the change management and ensure the consistency of models throughout this process with a minimum of human intervention, model transformations should be specified in a bidirectional way whenever possible. Unidirectional model transformations could impose restrictions on changes or adjustments of existing solutions. Unfortunately, for some vertical transformations (e.g., CIM to PIM, PIM to PSM), this may be inevitable because bidirectional transformations may require considerable effort that cannot be expended in practice. Q3

Computation-Independent Model

At the computation-independent level (CIM), our MDSD approach considers two main variability aspects of application development. First, enterprise business processes are represented using a modeling language like ARIS [27]. The modeler can define new processes and functionality on the basis of existing systems, services, and (sub-) processes provided by the MDSD solution. Figure 14.3 provides an example of a cross-organizational strategic sourcing business process modeled with ARIS. The participants of this process are: an OEM whose goal is the sourcing of an engineering service from a supplier (SU) and the PO that ensure the procurement.

The process starts with the OEM issuing a Statement of Requirements (SOR) to one or more suppliers via the PO. Suppliers generate offers, offers are collected and checked by the PO, and then sent to the OEM for evaluation. The example process ends with a selection of a supplier by the PO based on feedback by the OEM. To obtain good-quality models, which can be used for transformation to and refinement at PIM level, it is necessary to provide and enforce appropriate modeling guidelines. Modeling guidelines also support the modeler at business level in formulating his/her solution.

Second, it is crucial to consider further influence factors and constraints in our MDSD solution at CIM. The realization of EAI is not only influenced by the ICT systems to be integrated, but in addition by internal and external influence factors (also called contingencies [28]) of the enterprises. Examples are vertical standardization in the enterprise business segment, legislation and regulation, market

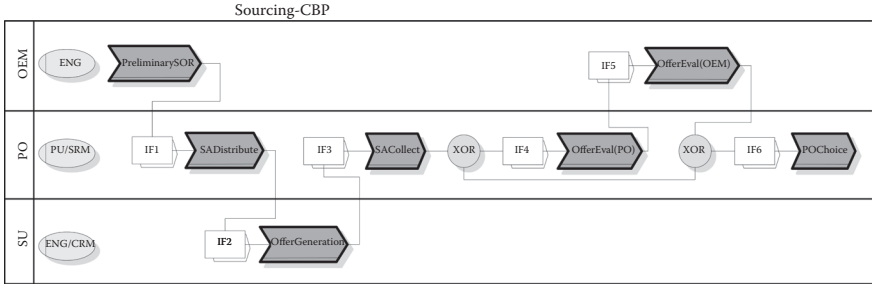

FIGURE 14.3 Example of a model at the CIM.

structure, and topologies (e.g., "star" vs. "net"), domain-specific distribution of data, rights, control, and so on. Other internal factors include competition for access to the departmental ICT and of the integrated ICT systems. Modeling at the CIM requires evaluating the importance of these factors, even through the modeling languages currently available do not yet support the inclusion of such constraints in CIMs. Accordingly, these constraints and additional factors need to be captured in separate decision models, where the constraints are decomposed in a decision tree with factors and subfactors. There is still demand for further research about how to represent this information in "enterprise network models" in order to increase automation and quality in ICT systems development.

CIM to PIM

In the next step, the integration architect has to select an appropriate integration solution architecture and a set of model transformations to implement the architectural topology. This can be carried out via a multicriteria decision model as described in Ref. [29]. Basically, the integration architect compares and evaluates the possible coordination and integration topologies on the basis of desired properties like modifiability, interoperability, or data security for various scenarios [30]. The rating criteria depend on the contigencies of the enterprises involved and the integration scenarios.

In our example, one OEM and a few big first-tier suppliers form a virtual enterprise by establishing a temporary association of independent companies, suppliers, and customers. They are linked by information technology to share costs, skills, and access to each other's markets. In one scenario, we assume that half of the cross-organization business processes are supported by legacy proprietary applications and that these applications would be replaced within the next 5 years.

The integration architect compares the integration topologies in terms of their support of future changes in the organizations' services. In the fully decentralized topology, services need to know the identity of any service that would replace a service they communicated with and adjust the syntax or semantics of the message exchanges accordingly. In the hierarchical topology, changes have to be made only in the controller service. In the hybrid topology, such changes are further restricted to the local controller service of the organization where the new service is introduced. Since in our scenario half of the applications are proprietary, the hybrid architecture would have the highest ranking and the fully decentralized the lowest. The rating can be extended to other scenarios and the overall rating of each integration topology is calculated [29]. The integration architect will choose the integration solution that has the highest overall rating.

In the hybrid topology, we consider that each organization provides one controller service to orchestrate the organization's internal services, called private processes (PPs). We use PIM4SOA (PIM for SOA) as a target model at the platform-independent modeling level. In PIM4SOA, controller services are called view processes (VPs) and the internal services of an enterprise are called PPs.

PIM4SOA was developed during the European project ATHENA. It supports the smooth transformation of business process models into Web Service environments

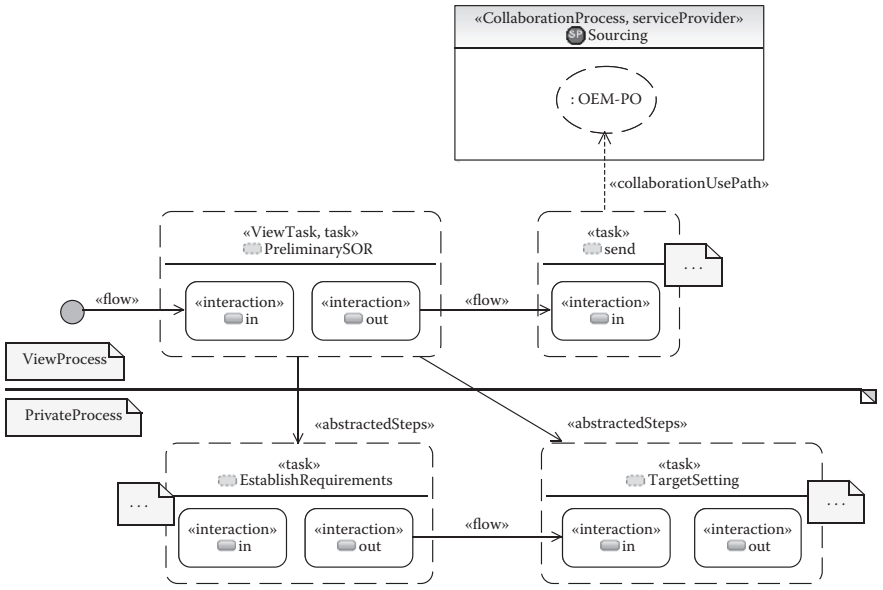

FIGURE 14.4 PIM4SOA model generated for the Sourcing CBP of the hybrid architecture.

and composition standards, like BPEL (see [31] for more details). Its metamodel supports platform-independent modeling through generic constructs, such as collaborations and service providers.

Figure 14.4 depicts a Unified Modeling Language visualization of the PIM4SOA model generated by applying a model transformation to the "Sourcing CBP" in the CIM for the hybrid architecture shown in Figure 14.3. The VP integrates and combines the services offered by the OEM's ICT system. For example, the OEM realizes the task *Preliminary SOR* through the *Establish Requirements* and *Target Setting* services. The model comprises further integration code like sending messages to collaborating partners. For example, the OEM sends a message, which is part of the Sourcing collaboration, to the PO.

To implement model-to-model transformations like our CIM-to-PIM transformation, a variety of transformation approaches exist [32]. The OMG's model transformation standard Queries/Views/Transformations (QVT) [33] supports declarative (QVT Relations) and imperative (QVT Operational Mappings) transformation approaches. Declarative transformation approaches are best applied to specify simple transformations and relations between source and target model elements, while imperative approaches lend themselves for implementing complex transformations that involve detailed model analysis [34]. Since our CIM-to-PIM transformation requires more detailed source model analysis to generate the target model, an iterative approach to implement the model transformation is appropriate. Listing 14.1 depicts an excerpt of the CIM-to-PIM transformation formulated with QVT

```
modeltype ARIS "strict" cimModel ;
modeltype PIM4SOA "strict" uses pimModel ;
transformation CIMtoPIM (in aris:ARIS , out pim4soa: PIM4SOA) {
  main () {
    aris.objectsOfType(EPC)->map generateCBP();
    aris.objectsOfType(EPC)->map mapCollaborations();
    ...
  }
  mapping EPC::generateCBP() : CollaborationProcess {
    when {
      self.swimlanes <> null;
    }
    population {
      result.type = 'ABSTRACT';
      result.name = self.name;
      result.views = self.swimlanes->map generateVP();
      self.controledges->map Collaborations(result);
    }
  }
  mapping Swimlane::generateVP() : ViewProcess {
    population {
      result.name = self.name;
    }
  }
  mapping ControlFlowEdge::mapCollaborations(in
cbp:CollaborationProcess) : Collaboration {
    init {
      result = cbp;
    }
    population {
      if self.source.swimlane = self.target.swimlane
      then result.collaborations = self->object(e) Collaboration {
        ...
        }
      else ...
      endif;
      ...
    }
  }
}
```

LISTING 14.1 Sample ARIS to PIM4SOA model transformation.

Operational Mappings. The rules describe the procedure to transform the structural part of the CBP description into service-oriented models. The notation uses a pseudo-metamodel of ARIS and the PIM4SOA metamodel. More details can be found in Ref. [35].

PIM TO PSM

The selection of a particular integration topology at the platform-independent level reduces the number of possible realizations or PSMs. For instance, a hybrid topology could be realized through a *super-peer appro*ach or by using federated message brokers. A super-peer a *P2P* network is an architecture with two types of nodes:

normal peers or super peers, which are dedicated nodes with specific capabilities. A federated broker consists of several workflow orchestration engines working together.

In our example scenario, we choose to realize a federated broker architecture. The PIM-to-PSM transformation maps the processes and the services onto WS-BPEL [36] and Web Services Description Language (WSDL) [37] descriptions.

A *template* is used for the generation of the WSDL code. A template consists of the target text containing slices of metacode to access information from the source and to perform code selection and iterative expansion. Templates are close to the structure of the code to be generated and are perfectly suitable to iterative development as they can be easily derived from examples. Hence, templates are a good choice for the generation of WSDL code or other complex invocation patterns that can occur in workflow languages and platforms (see Ref. [38]).

When generating the BPEL code, however, the code generation depends also on the control flow of the modeled processes. In our application scenario, the generated BPEL code has a specific sequence of processing steps depending on the control flow of the described process. This sequence is produced through complex graph transformation algorithms and with the invocation of the generation templates in a specific order (see [38]). Because a purely template-based generation of the BPEL code is hard to maintain and to extend, it is beneficial to combine template-based and visitor-based code generations [32,38].

In the following we have look at our 'Sourcing' process to illustrate the BPEL code generation for the example model shown in Figure 14.4. Basically, a visitor mechanism traverses the process flow of the process at the platform-independent level (a more detailed description of the applied workflow code generation framework can be found in [38]). For each node in the PIM the visitor invokes the templates that generate the respective BPEL code. For the PIM process depicted in Figure 14.4, it works as follows:

- The visitor starts by traversing the VP at the start node. When the visitor enters the action "*PreliminarySOR*," it calls the respective templates for entering a model element of the type «*ViewTasks*» (cp. Listing 14.2 depicts some sample generation templates specified using the openArchitectureWare [oAW] framework [39]). The information (the variable parts) reused from the PIM is italicized. Code interpreted by the oAW framework is contained in 'double-brackets' («,»). An attribute is evaluated for the current model element (for which the template is applied). For example, «*name*» for the model element of the type «*ViewTask*» in Figure 14.4 is evaluated to "*PreliminarySOR*." One can use these properties to navigate through the model, for example, «*collaborationUsePath.collaboration.name*». **Q4** **Q5**
- Next, the visitor processes the tasks as a view task abstract from «*abstractedSteps*» and calls the generation templates for «*task*».
- Finally, the visitor processes the «*send*»-message to another integration process (which may run on another workflow engine of the federated integration solution).

```
«DEFINE EnterViewProcess FOR PIM4SOA::ViewTask»
  <scope name=' «name» '>
«ENDDEFINE»

«DEFINE ExitViewProcess FOR PIM4SOA::ViewTask»
  </scope>
«ENDDEFINE»

«DEFINE SendTask FOR PIM4SOA::Task»
  <reply name=' «name» ' partnerLink='
«collaborationUsePath.collaboration.name» ' operation=' «...» ' />
«ENDDEFINE»
```

LISTING 14.2 Sample oAW templates.

Listing 14.3 shows the integration code that a BPEL engine generates using the templates of Listing 14.2 (the templates for ãtaskõ are omitted in the listing). The examples are kept simple for the sake of understandability; in real-world applications, the transformations can be much more complicated (e.g., in the AgilPro project [40], about 80 lines of BPEL code need to be generated for a single task [38]).

CONCLUSIONS AND FUTURE OUTLOOK

In this chapter, we have presented how a combination of service orientation and model-driven engineering can be applied in the area of EAI. This approach provides a structure way for cross-enterprise applications and automates the propagation of changes in business requirements to the design and realization of IT systems.

In process-driven enterprises, enterprise applications need to be developed, managed, and integrated in the context of the business processes that these applications support. Therefore, EAI activities aim to align, harmonize, and coordinate the business processes of co-operating enterprises. Business process-level integration has been accomplished through a variety of point-to-point integration models, most commonly facilitated by broker and orchestration components.

```
<process name='OEM_Sourcing' ...>
  <scope name='PreliminarySOR'>
    <invoke name='EstablichRequriements' partnerLink='OEM_internalLink'
operation='...' />

    <invoke name='TargetSetting' partnerLink='OEM_internalLink'
operation='...' />

  </scope>

  <reply name='send' partnerLink='OEM-PO' operation='...' />
</process>
```

LISTING 14.3 BPEL code generated using the template in Listing 14.2.

In recent years, a new generation of EAI solutions has been developed under the service-oriented paradigm, which lends itself to the development of highly adaptable solutions and to the reuse of existing applications. This is because service-oriented integration adopts the concept of service to establish a PIM that can be used with various integration architectures. As a consequence, in a service-oriented world, sets of services can be assembled and reused to quickly adapt to new business needs. However, service-orientation does not provide an integration solution by itself.

Even though the example realization used BPEL and WDSL, the approach works for other standards in the service-oriented area. The applicability of this approach has been shown in several projects like AgilPro [40] or the European Project ATHENA.

Supporting EAI for service-oriented systems and the enactment of CBPs with MDSD poses a number of challenges. One of these challenges is the development of software architecture coordination patterns that can be applied for the integration of cross-organizational systems and the coordination of the relevant business processes. Another challenge in this context is the automatic derivation ICT system models that are based on the coordination patterns from higher-level CBP descriptions. One of the remaining challenges that this approach would have to meet is to ensure the interoperability of the different systems, possibly by extending it using ontological concepts (see Refs. [40][42][43]).

Four important issues are still open in the field of model-driven engineering. First, MDE introduces additional costs to the development process, since metamodels and model transformations first have to be developed. They only pay off when they can be applied several times. For example, they can automate recurring tasks or encode patterns that can be reused in different projects.

Second, models and model transformation have to track the evolution of standards, applications, and metamodels. Possible approaches have been presented [44][45][46], including the use of Semantic Web technologies [26][47].

A third point of discussion is the number of abstraction levels that are applied in MDE solutions. More abstraction levels provide more possibilities for customization (and more difficulties in uncovering errors) and also have to be accompanied with model transformations between the different abstraction levels. The "model-driven light" idea (see e.g., [48]) using one consolidated metamodel for more than one abstraction layer. Supporting different views for each abstraction layer, for example, by using event-driven process chains, Business Process Modeling Notation, or any other process modeling standards, may simplify MDE and reduces the number of model transformations. **Q6**

Fourth, the reusability of application is another topic of discussion. The ADM of the OMG [25] supports the bottom-up approach, thus complementing MDE. Reuse of models and model transformations in the design are also interested interesting aspects. Here again Semantic Web technologies can help.

So far, the focus of most research on MDSD has been on the computer-supported coordinated management of models at the levels of business process, IT architecture, and IT platform. Models of the strategic vision of the enterprise vision such as the Balanced Scorecard [11] need to be incorporated systematically in an MDSD context. The scope of consideration has to be extended from the business processes to

the enterprise strategy and enterprise goals, and—taking the reverse view—knowledge about long-term strategic objectives of enterprises and groups of enterprises ultimately have to be made available and be used more effectively in order to shape and optimize architectures of cross-enterprise business integration.

REFERENCES

1. Snow, C.C., Miles, R.E., and Coleman, H.J., Managing 21st century network organizations, *Organ. Dyn.*, 20 (3), 5–20, 1992.
2. Erl, T., *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, Upper Saddle River, NJ, 2004.
3. Fowler, M., *Patterns of Enterprise Application Integration*, Addison Wesley Professional, Indianapolis, IN, 2002.
4. Bézivin, J., On the unification power of models, *Softw. Syst. Model.*, 4 (2), 171–188, 2005.
5. Hailpern, B. and Tarr, P., Model-driven development: The good, the bad, and the ugly, *IBM Syst. J.*, 45 (3), 451–461, 2006.
6. Stahl, T. and Völter, M., *Model-Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, Chichester, England, 2006.
7. Alex, B., and Ritsko, J., Preface to service-oriented architecture, *IBM Syst. J.*, 44 (4), 651–652, 2005.
8. Bell, M., *Introduction to Service-Oriented Modeling, Service-Oriented Modeling: Service Analysis, Design, and Architecture*, John Wiley & Sons, Hoboken, NJ, 2008.
9. Erl, T., *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, Upper Saddle River, NJ, 2005.
10. Greenfield, J. et al., *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, Wiley Publishing Inc., Indianapolis, IN, 2004. **Q7**
11. Kaplan, R.S. and Norton, D.P., The balanced scorecard—Measures that drive performance, *Harv. Bus. Rev.*, January–February, 71–80, 1992.
12. Schmidt, D.C, Guest editor's introduction: Model-driven engineering, *Computer*, 39 (2), 25–31, 2006.
13. Booch, G. et al., An MDA manifesto, *MDA J.*, May, 2–9, 2004.
14. Leymann, F., Roller, D., and Schmidt, M.T., Web services and business process management, *IBM Syst. J.*, 41 (2), 198–211, 2002.
15. Bernus, P., Nemes, L., and Schmidt, G., Eds, *Handbook on Enterprise Architecture*, Springer-Verlag, Berlin, 2003.
16. Stäber, F. et al., Interoperability challenges and solutions in Automotive Collaborative Product Development, in *Enterprise Interoperability II: New Challenges and Approaches (I-ESA'07)*, Gonçalves, R.J. et al., Eds, Springer-Verlag, London, 2007, 709–720.
17. Stoica, I., et al., A scalable peer-to-peer lookup service for Internet applications, in *Proceedings of the ACM SIGCOMM'01 Conference*, San Diego, CA, 2001. Also, *IEEE/ACM Trans. Netw.*, 11 (1), 17–32, 2003.
18. Stäber, F. and Müller, J.P., Evaluating peer-to-peer for loosely coupled business collaboration: A case study, in *Proceedings of 5th International Conference on Business Process Management (BPM'07)*, Alonso, G., Dadam, P., and Rosemann, M., Eds, vol. 4714, Lecture Notes in Computer Science, Springer-Verlag, 2007, 141–148. **Q8**
19. IBM Corp. IBM Websphere Message Broker, available at http://www-306.ibm.com/software/integration/wbimessagebroker/, accessed May 3, 2008.
20. Friese, T., et al., A Robust business resource management framework based on a peer-to-peer infrastructure, *Proceedings 7th International IEEE Conference on E-Commerce Technology*, 2005, 215–222. **Q9**

21. Czarnecki, K. and Eisenecker, U., *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley Professional, Indianapolis, IN, 2000.
22. Parnas, D., Designing software for ease of extension and contraction, *IEEE Trans. Softw. Eng.*, 5 (2), 128–138, 1979.
23. Clements, P. and Northrop, L., *Software Product Lines: Practices and Patterns*, Addison-Wesley Professional, Indianapolis, IN, 2001.
24. van der Linden, F.J., Schmid, K., and Rommes, E., *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Springer-Verlag, New York, 2007.
25. OMG. White paper: Architecture-driven modernization—Transforming the enterprise. admtf/07-12-01, December 2007. **Q10**
26. Roser, S. and Bauer, B., Automatic generation and evolution of model transformations using ontology engineering space, *J. Data Semant.*, in press, Springer LNCS. **Q11**
27. Scheer, A.W., *ARIS—Vom Geschäftsprozess zum Anwendungssystem*, 3rd Edn, Springer-Verlag, Berlin, 1998.
28. Donaldson, L., *The Contingency Theory of Organizations*, SAGE Publications Inc., Thousand Oaks, CA, 2001.
29. Roser, S., Designing and enacting cross-organizational business processes: A model-driven, ontology-based approach, PhD thesis, University of Augsburg, 2008.
30. Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice*, Addison-Wesley, Indianapolis, IN, 2003.
31. Benguria, G. et al., A platform independent model for service oriented architectures, in *Enterprise Interoperability: New Challenges and Approaches*, Doumeingts, G. et al., Eds, Springer-Verlag, London, 2007, 23–34.
32. Czarnecki, K., and Helsen, S., Feature-based survey of model transformation approaches, *IBM Syst. J.*, 45 (3), 621–645, 2006.
33. OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification—Final Adopted Specification, ptc/07-07-07, July 2007. **Q10**
34. Gardner, T. et al., A review of OMG MOF 2.0 Query/views/transformations submissions and recommendations towards the final Standard, *Proceedings of 1st MetaModeling for MDA Workshop*, 2003, 178–197, available at http://www.zurich.ibm.com/csc/bit/bpia.html or http://www.omg.org/docs/ad/03-08-02.pdf
35. Bauer, B., Müller, J.P., and Roser, S., A decentralized broker architecture for collaborative business process modeling and enactment, in *Enterprise Interoperability—New Challenges and Approaches*, Doumeingts, G. et al., Eds, Springer-Verlag, London, 2007, 115–125.
36. OASIS, Web Services Business Process Execution Language Version 2.0, wsbpel-primer, May 2007. **Q12**
37. W3C, Web Services Description Language (WSDL) 1.1, available at http://www.w3.org/TR/wsdl, March 2001.
38. Roser, S., Lautenbacher, F., and Bauer, B., Generation of Workflow Code from DSMs. In *Proceedings of 7th OOPSLA Workshop on Domain-Specific Modeling*, Montreal, Canada, Sprinkle, J., Gray, J., Rossi, M., and Tolvanen, J.P., Eds, 7th OOPSLA Workshop on Domain-Specific Modeling, Number 38 in Computer Science and Information System Reports, University of Jyväskylä, 2007.
39. oAW, openArchitectureWare (oAW), available at http://www.openarchitectureware.org
40. AgilPro, AgiPro project: Agile Business Processes with Service Enabled Applications, available at http://www.agilpro.eu **Q13**
41. Elvesæter, B. et al., Towards an interoperability framework for model-driven development of software systems, in *Interoperability of Enterprise Software and Applications*, in *Proceedings of the 1st International Conference on Interoperability of Enterprise Systems and Architecture (Interop-ESA'2005)*, Konstantas, D. et al., Eds, Springer-Verlag, 2005, 409–423.

42. IDEAS—Thematic Network, A Gap Analysis, Deliverables D3.4, D3.5, D3.6, 2003. **Q14**
43. Missikoff, M., Schiappelli, F., and Taglino, F., A controlled language for semantic annotation and interoperability in e-business applications, *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, Sanibel, FL, 2003, available at http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-82
44. Didonet Del Fabro, M. et al., Model-driven tool interoperability: An application in bug tracking, in *5th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'06)*, vol. 4275, Lecture Notes in Computer Science, Meersman, R. et al., Eds, Springer-Verlag, Berlin, 2006, 863–881. **Q15**
45. Didonet Del Fabro, M., and Valduriez, P., Semi-automatic model integration using matching transformations and weaving models, *Proceedings of the 200l ACM Symposium on Applied computing (SAC)—Model Transformation Track*, 2007, 963–970.
46. Wimmer, M. et al., Towards model transformation generation by-example, *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS-40)*, 2007, 285b. **Q15**
47. Kappel, G. et al., Lifting metamodels to ontologies: A step to the semantic integration of modeling languages, in *Proceedings of the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML)*, vol. 4199, Lecture Notes in Computer Science, Springer, Nierstrasz, O. et al., Eds, 2006, 528–542, available at http://www.bioinf.jku.at/publications/ifs/2006/1006.pdf
48. Roser, S., Lautenbacher, F., and Bauer, B., MDSD light for ERP, *Proceedings of the 23rd Annual ACM Symposium on Applied Computing, Track on Enterprise Information Systems—EIS*, 2008, 1042–1047. **Q16**
49. Stiefel, P., et al., Realizing dynamic product collaboration processes in a model-driven framework: Case study and lessons learnt, in *Proc. of 14th International Conference of Concurrent Engineering*, Lisbon, Portugal, 2008. Available at http://www.ice-proceedings/org