Model- and Architecture-Driven Development in the Context of Cross-Enterprise Business Process Engineering

Stephan Roser[1], Bernhard Bauer[1], Jörg P. Müller[2]
[1]*Institute of Computer Science, University of Augsburg, D-86135 Augsburg*
[2]*Institute of Computer Science, Clausthal University of Technology, D-38678 Clausthal*
[bauer | roser]@informatik.uni-augsburg.de
mueller@in.tu-clausthal.de

Abstract

Modelling and enacting Cross-Enterprise Business Processes (CBPs) is a key ability for successfully setting up and managing virtual organizations, e.g. supply chains. In this paper we present and compare approaches for modelling CBPs based on the service-oriented architecture paradigm. By embedding our overall approach into a model- and architecture-driven development perspective, we show how service-oriented systems realising CBPs can be derived from business-level modelling.

1. Introduction

Enabling enterprises to keep up with constantly evolving business relationships and cross-organizational value chains, business systems need to become more adaptive and service-oriented. In order to achieve this, methodologies, methods, and infrastructures are required to support changes to business processes being defined at the business level and providing well-defined (and possibly largely automated) model transformations and refinements down to the level of service-oriented implementations. The main objective of our research is to improve business interoperability by developing architecture and tools to provide end-to-end support for the design of business processes, from the business level down to deployed service-oriented applications. The approach we follow in order to achieve this end is model-driven software development (MDSD) [2], a generalization of OMG's Model-Driven Architecture paradigm (MDA™) [14] in combination with a software architecture driven approach (see Section 3).

Within the context of the ATHENA IP [1], we have extended the MDA paradigm to fit the needs of modelling CBPs; in order to realize such CBPs in a service-oriented environment based on a software architecture centric approach. The goal is to develop executable models of cross-enterprise collaborations by applying MDSD techniques based on software architectures. In previous work [4][6], we described conceptual MDSD-based architectures for modelling CBPs with sets of model transformations enabling semi-automated mapping of a computation-independent description, down to a platform-independent model representation. We investigated architectures for realizing business level processes (in our case starting from a CBP model expressed using ARIS[1]) into an information and communication technology (ICT) architecture at the platform-independent level based on Service Oriented Architecture (SOA).

Totally decentralized architectures without broker can very well be applied to eBusiness scenarios with restricted size and complexity like online shopping or auctions in the B2C or C2C market. Those scenarios are characterised by the fact that they describe routine processes with low specificity. The centralized broker architecture described in [4] was found to be useful in a scenario where the collaborative process was largely designed and its execution controlled by one partner in a cross-enterprise relationship, i.e., corresponding to a star topology of a business relationship including one large and powerful player and multiple smaller players, as we can observe it today e.g. in the automotive and aerospace domains. However, it reveals limitations in flexibly supporting more symmetric business relationships, focusing on scenarios supporting collaboration

[1] We realize that some readers will feel alienated by the fact that we call ARIS computational independent level. However, ARIS is a de-facto industry standard for the business level modelling of business processes, and thus is the natural starting point for a top-down model-driven development approach.

between Small and Medium Enterprises, as well as Virtual Enterprise scenarios, where partners that compete in other sectors join together temporarily to provide a product or service. These scenarios will benefit from a less centralized architecture (see [6]), enabling looser coupling, more modular and flexible process modification strategies, and a higher degree of autonomy of the individual partners (including better support for encapsulating enterprise-internal information).

The main contributions of this paper are twofold: first, we introduce three architectures for controlling and enforcing CBPs in a model-driven development context; second, we describe and compare transformation procedures for deriving service-oriented platform-independent models of a CBPs based on different architectures described in Section 3.

After summarizing the technological context of the work in Section 2, Section 3 introduces and aligns model-driven software development with software architecture centric approaches. Section 4 describes three architectures for realizing CBPs in a service-oriented environment. In Section 5, we present instances of model transformations for ARIS to a SOA. Section 6 discusses the described conceptual modelling architectures related to interoperability, relationships to related work, and areas of future research.

2. Context

2.1. Business Process Modelling Terminology

[9] distinguishes between an internal and an external view of business processes. Depending on the viewpoint, a process is described either as an executable, abstract, or collaborative process: **Executable Process:** The internal view models the 'how' of a business process from the modeler's view. Processes that model process flows as a set of partially ordered tasks, are called executable processes [10]. As the flow of an executable processes is described from the viewpoint of a single process coordinating its sub-processes, this is often referred to as process orchestration. **Abstract process:** The external view models the 'what' of a business process. Each process specifies its roles in the collaboration with other processes, but hides the way it is realized. The interfaces of such business processes components are called abstract processes describing the public interactions they perform in relation to their roles in collaborations. **Collaborative process:** describes the collaboration between abstract processes in the case of process choreography. Collaborative processes use abstract processes to model the sequence of the message exchange from the viewpoint of an external observer. The collaborations between the involved parties are modelled as interaction patterns between their roles.

In order to coordinate inter-organizational workflow Liu and Shen introduced the concept of views, as they are used in database systems, to provide abstract information about internal processes. In [13] they extend their work to CBPs. Chiu et al. introduce workflow views to control visibility of internal processes and to enable interoperability of e-services, focusing on combining views of different partners to composite e-services (CBPs). Schulz et al. use the concept of views, and formalize the dependencies between private processes, process views and CBPs [17].

Adopting the general approach of [17], we distinguish between private processes, view process and CBPs (according to [12]): **Private Processes** are internal to an organization. They contain data not be revealed by default. Views on processes provide an abstraction of private processes, which is sufficient to coordinate internal actions with activities of external trading partner(s) [17]. A particular interaction may require involved partners to adapt for the purpose of the communication. This adaptation may not necessarily be reflected in the partners' private (internal) business processes without inflicting their ability to interact with other partners in a different context. **View Processes** combine private processes to an abstract level that enables companies to hide critical information from unauthorized partners. The view process connects the private process with the abstract process an organization provides to a CBP. Based on one private process, different views can be generated and reflecting the specific requirements of multiple interactions. **CBPs** define the interactions between two or more business entities. These interactions take place between the defined abstract processes and are defined as a sequence of message and/or other material input/output exchange. Using different views of the same internal processes, organisations are able to interact in a different context without changing the internal process.

2.2. PIM4SOA

A major result of the ATHENA IP [1] is a set of metamodels and tools called PIM4SOA (Platform-Independent Model for Service-Oriented Architecture, see [7]), supporting the smooth integration into Web Service Composition standards, in particular WS-BPEL [11]. The PIM4SOA metamodel has essentially two main concepts for describing services and their

collaborations at a platform-independent level (see Figure 1). **Collaborations** specify patterns of interaction between participating roles. They specify the involved roles and their responsibilities within the collaboration. **Service providers** take on roles through which they participate in collaborations and realize roles in collaborations. The participation of service providers in a collaboration is modelled via *Collaboration Uses*. The bindings of a collaboration use specify which roles of the collaboration are realized by the roles of the service provider.

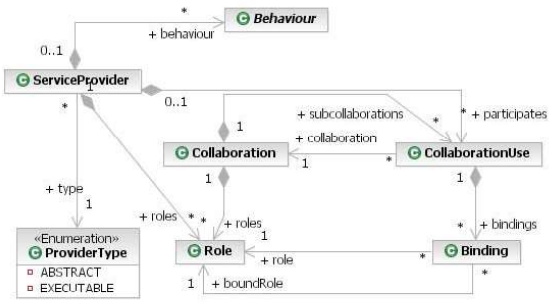

Fig. 1: PIM4SOA: structure

The communication behaviour as well as the activities, that together realize the provided services, can be described by the service provider's behaviour, i.e. process (see Figure 2). The process of a service component specifies the externally observable activities independent of the realization. The process flow defines sequencing constraints and data flow on the related activities. A task is either an internal task that is not further specified, or an interaction task through a service. In the latter case the service is referenced by specifying a collaboration use path.

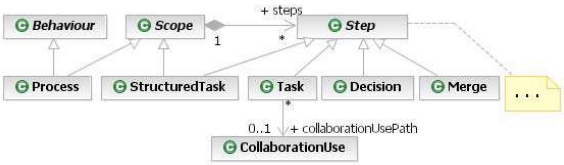

Fig. 2: PIM4SOA: behaviour

3. Architecture for Business Process Modelling Methodology

We introduce, describe and compare model transformations (from business level ARIS models to platform-independent ICT system models) encoding knowledge about a service-oriented ICT architecture.

3.1. MDSD Approach

The architecture of business process modelling can be aligned to architectures for model-driven software development and to an architecture of eBusiness systems. Starting from the business viewpoint of analysts, where all issues related to the organization and the operations of an enterprise are addressed, business process models are semi-automatically refined to specification views of system architects (e.g. service and interface models).[2] Their viewpoint focuses on the main components of a software system before it is refined to the realizations viewpoint of system developers (e.g. interaction and data models). Both specification and realization viewpoint focus on ICT solutions allowing an enterprise to operate, make decisions, and exchange information internally and externally. They deal with composition and flexible execution of services.

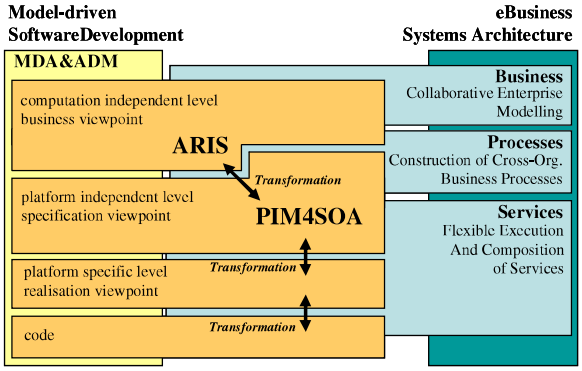

Fig. 3. MDSD and e-Business architectures

Changes of viewpoints and modelling methods are a crucial point in the development of an ICT system, since errors made at this stage are rarely found before the system is deployed. Thus a main concern is that business models are consistent with the ICT systems models (which a grounded on service-oriented architecture in our case).

3.2. SW-Architecture Centric Approach

The typical views in the context of software architectures are (for details see [3]): *context view* showing the interaction and interworking of the system under development with its environment from a high level of abstraction. Interfaces to neighbour systems, and the interaction with the main stakeholders as well as the main points of the infrastructure are shown; *Building*

[2] For a more detailed description of the viewpoints see [8].

block view showing how the system is internally structured, defining the static structures of the system, subsystems, components and its interfaces, which usually developed in a top-down approach from the context view; *Run-time view* showing the dynamics of the system, i.e. which building blocks exist and interwork during run-time; *Deployment view:* in which environment runs the system, i.e. HW component, processors, net topology and protocols.

A software architecture centric approach has to go along with the model-driven approach to support and document the software views as well as the models at different levels of abstraction within an eBusiness systems architecture. E.g. the system architect acts particularly on the context and building block views. A methodology will provide necessary information to allow automatic generation, semi-automatic refinement and deployment of processes to a service-oriented environment.

4. Modelling Architecture for CBPs

Many people and organizations participate in the construction of software systems, and impose different concerns and requirements on the system. Business considerations determine non-functional qualities that must be accommodated in the system architecture. Quality attributes like availability, modifiability, performance, security, testability, usability or business qualities are orthogonal to functional attributes describing the system's capabilities, services, and behaviour. Since quality attributes are critical to the success of systems, they must be considered throughout design, implementation and deployment [3].

In our work we describe, how service-oriented architecture variants of software systems for CBPs can be derived from business level descriptions. By investigating how architecture variants satisfy various quality attributes, we observed two interoperability-related challenges for CBP architectures. Thus our focus is on *modifiability* and *privacy of internal data*.

Modifiability: is about the cost of change [3]. Thus the quality attribute mostly depends on how *flexible* and *modular* a system is. The granularity of the architectural concepts should be sufficiently fine that changes to a participant's private implementation do not necessarily result in changes of other participants private processes. Modularity allows participants to a CBP to be able to change processes without affecting other participants.

Privacy of internal data: The modelling architecture should enable participants of a CBPs to preserve privacy of their internal data, interfaces and processes. The information provided to participate in a CBP generally not allow insights into the participant's internal realization of the functionality.

CBPs can be realized in multiple ways, differing in how CBP's conversation flow is coordinated. In a brokerless approach, private processes use their abstract processes to directly exchange messages over enterprises' boundaries. In an architecture relying on a central broker, private processes exchange messages with an intermediary acting as a global observer process coordinating the partners as well as making decisions on the basis of data used in the CBP. The decentralized broker architecture finally divides the broker process into several view processes jointly realizing the broker. The view processes are provided by organisations participating in the CBP.

In a brokerless architecture control flow logic of CBPs is realized by the private processes of the participants. Due to the mutual exchange of messages these processes depend on one another. Changing the business protocol would result in changing multiple executable processes. This is only flexible and modular to a small extent: if the abstract process of one private process significantly changes, other private processes or even the protocol description need to be adjusted. The same restriction holds for privacy.

The application of a broker pattern has several advantages. When changing protocol description of a CBP, only the broker process needs to be modified, not the multiple private processes of the participating organizations. Organizations can hide their internal processes from their collaborators, but instead have to reveal them to a third party, the centralized broker.

In decentralized broker approach, the single broker component is replaced by several view processes jointly providing the broker functionality (note the boundary in Figure 4). The view process behaviour, which is relevant to the CBP, is defined by public abstract processes. An abstract process is realized by the executable process of the respective view process. A view process also provides internal abstract processes in order to use the private processes' functionality. The enterprise boundary in Figure 4 shows which private processes are used by a view process and vice versa.

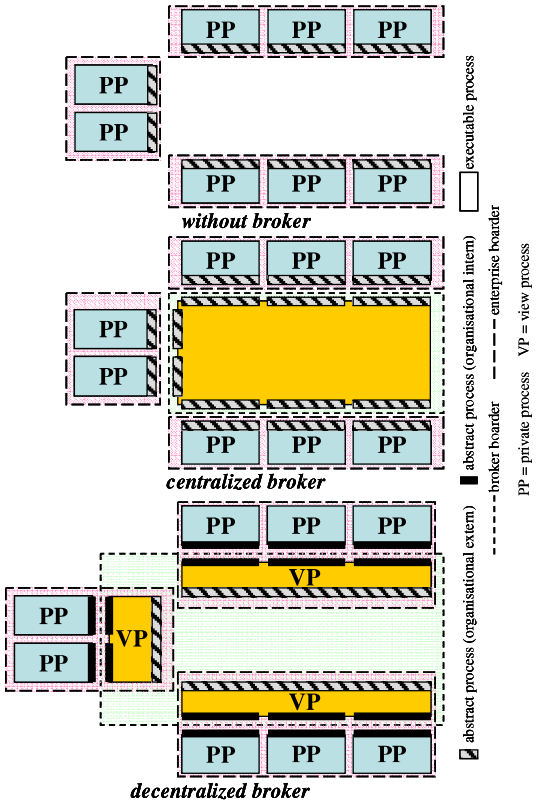

Fig. 4: ICT architecture for CBPs

From a runtime point of view there are two alternatives depending on whether the broker is hosted by a third party or not. In case the logical enterprise boundary in the architecture is also a physical boundary in the runtime architecture, a view process is realised and executed by the participating enterprise owing also the private processes.

Thus, the decentralized broker architecture satisfies the requirement of *flexibility and modularity* at the conceptual level. The view processes are preserved in any runtime architecture derived from this conceptual architecture. The *privacy of internal data* depends on the realization of decentralized broker at runtime. This requirement can be met, if view processes are not hosted by a third party, but rather implemented and executed by the enterprises participating in the CBP.

We are aware that our conceptual ICT architecture does not mention a directory service. The focus of this paper is on the architectural issues for CBP modelling and enactment in order to satisfy quality attributes. It is easy to image a directory service used to dynamically search process partners interacting in CBPs at runtime.

5. Model-Driven Design of CBPs

Business process models at a computational-independent level of abstraction are not affected by the adoption of a certain conceptual architecture. However, ICT models at the platform-independent level may differ considerably depending on whether an architecture with centralized or decentralized brokers, or a brokerless architecture is used. Thus, transformations have to be adjusted to the various architectures. This section shows, how service-oriented ICT models for realizing collaborating components of architectures with and without broker[3] can be derived from high-level ARIS description of a business process.

5.1. Example CBP

The CBP example comprises the solicitation of quotations and the choice of component suppliers by an automotive manufacturer. Three roles are involved in the cross-organizational business process: *OEM (Original Equipment Manufacturer)* is the automotive manufacturer planning to produce a new automobile type; *PO (Purchasing Organization)* is an independent company or department of the OEM conduction the solicitation of quotations and the final selection of the suppliers; *SU (Supplier)* is a component supplier for the automotive industry aiming to place contracts with the OEM via the PO.

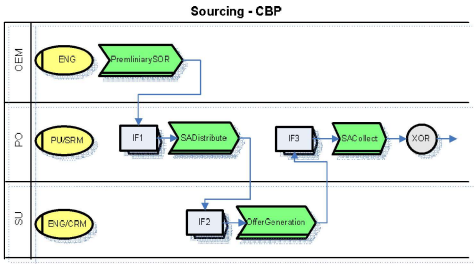

Fig. 5: Case study – process overview

As shown in Figure 5 the three roles OEM, PO and SU are modelled as swimlanes in the process description. The CBP starts with the OEM conducting *PreliminarySOR* where the requirements are gathered and summarised in a 'Statement of Requirements' (SOR). In *SADistribute* PO generates an ActionPlan from the SOR, containing information about the parts to be provided by the suppliers. OfferRequests are derived from the ActionPlan and sent to appropriate SUs. An SU evaluates in the process *OfferGeneration* at which price it can supply certain parts. The SU creates an

Offer and sends it back to the PO. After a PO has collected all incoming offers in *SACollect,* the evaluation of these offers starts.

In the first column of the EPC diagram we can see the departments responsible for executing the roles of the CBP. The OEM role is realized by the engineering department (ENG), the PO role by purchasing department (PU) and the Supplier Relationship Management (SRM), and the SU role by the engineering department (ENG) and the Customer Relationship Management (CRM) of the supplier.

5.2. Refining CBPs for PIM4SOA

The metamodel for specifying services and collaborations presented in Section 2.2 is to be extended for a decentralised broker architecture. We introduce concepts for describing private processes participating in collaborations through their view processes.

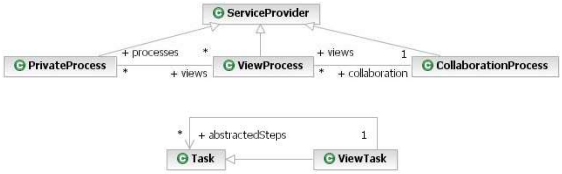

Fig. 6: PIM4SOA extension

In Figure 6 we see private processes, view processes and collaboration processes as service providers. A private process is an executable service provider who references view processes that enact its participation in external collaborations. Its behaviour is modelled by an executable process. A view process is an executable service provider whose behaviour is a process flow model that may include view tasks. A view task is an activity that abstracts a set of activities of the realizing private processes into a single task. A view process realizes roles in a single collaboration and view tasks are visible in the collaboration. A collaboration process is an abstract service provider whose behaviour is a process flow model. The collaboration process may specify the view processes that together enact the collaboration.

We regard a view process as an *executable process* that realizes several abstract processes - one for the collaborations it participates in and the others to participate in the implicit collaborations with the private processes it supports. A view process connects the abstract process an organization provides to a CBP with realizing private processes of the organization.

[3] For more details about the various approaches see [4] and [6].

5.3. Brokerless architecture

Using a brokerless architecture, we can derive private processes as services of a service-oriented environment. The business-level CBP-description enables the composition of binary collaborations patterns to more sophisticated collaborations and protocols.

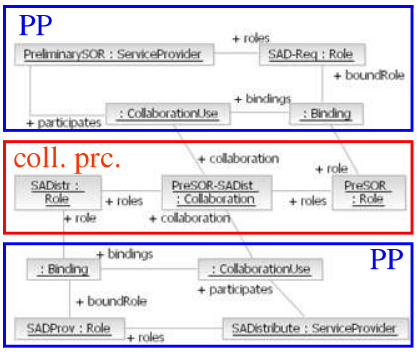

Fig. 7: PIM4SOA instance without broker

From each private process of the 'Sourcing'-CBP description, one service provider is derived. The service providers of the private processes directly collaborate with each other. Therefore collaborations are instantiated for each pair of private processes communicating.

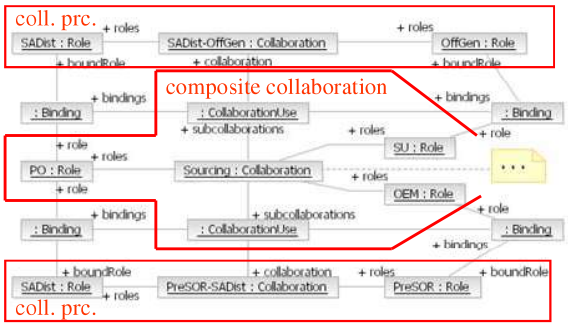

Fig. 8: Instantiation of composite collaboration

The CBP described at business level is divided in multiple binary collaborations in the brokerless ICT architecture. To avoid loss of information (especially about the complete CBP), the collaborations can be grouped to composite collaborations (see Figure 8).

In the example, the composite collaboration derived from the 'Sourcing'-CBP is composed by the binary collaboration patterns between the private processes. This concept is analogous the UML2 specification. [15], p.164ff.

5.4. Centralized broker

A centralized broker architecture is realized by a coordinating executable broker process and several private process. As described in [4], the broker process coordinates the executable private processes via collaboration protocol descriptions.

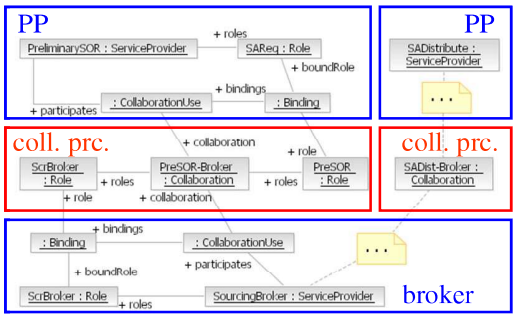

Fig. 9: PIM4SOA instance central broker

For the 'Sourcing'-CBP a service provider is instantiated to provide the centralized broker functionality. Each private process is transformed to one service provider and communicates with the broker process over separate collaboration processes instantiated as collaborations. In Figure 9 the Role *SAD-Req* of the *PreliminarySOR* (`+roles`) is bound to the *PreSOR* Role of the collaboration with the broker process. The broker process implements the coordination of the message exchange described by the protocol description.

5.5. Decentralised broker

For a decentralised broker architecture we are able to generate view processes, being an abstraction from more detailed private processes, and their links to the private process implementation by means of the service-oriented PIM4SOA metamodel and its CBP-extension at platform-independent level.

In Figure 10, the target PIM4SOA model is created by applying transformation to the sample ARIS model introduced in Section 5.1. A collaboration process is derived for the CBP 'Sourcing'. The collaboration process is an abstract service provider. One view processes is derived for each organisation that takes part in the CBP, i.e. the engineering department of the first participating organisation (*Org1-ENG*) and so on. An association connects the view process (`+views`) to the collaboration (`+collaboration`). For each pair of roles that collaborate in the 'Sourcing'-CBP, one collaboration process and one role for each of the collaborating roles are instantiated for the PIM4SOA model; in Figure 10 these are the roles *OEM* and *PO*. Participation of the departments in the collaboration is represented by collaboration uses connecting the respective view processes with the collaboration. A binding is used to specify with which role (`+boundRole`) a service provider realizes a role (`+role`) in the collaboration. Considering the architecture described in Figure 4, the collaboration process represents the protocol description between the publicly visible abstract processes. The CBP is an abstract service provider (not executable) and groups the view processes belonging to one CBP.

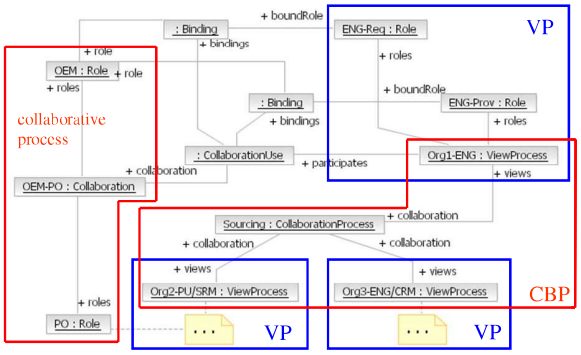

Fig. 10: PIM4SOA instance decentral broker

Figure 11 shows the generation of view processes' behaviour description at the example of the *Org1-ENG* view process. The behaviour of the view process, i.e. the service provider, is described by process. This process consists of steps which are derived from ARIS-CBP. Two view tasks, *PreliminarySOR* and *OfferEvaluation(OEM)*, are instantiated for the corresponding process modules of the ARIS-CBP.

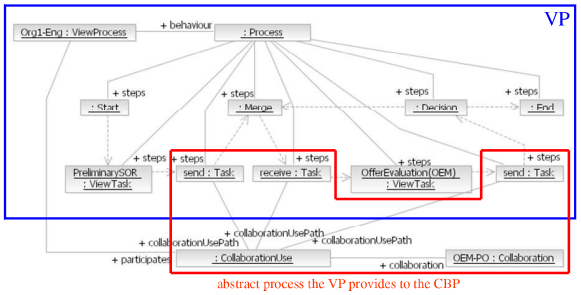

Fig. 11: PIM4SOA instance connecting view process to private process behaviour

For the *Org1-ENG* view process two 'send' and one 'receive' tasks (two times it invokes another process and one time it is invoked) are instantiated and added to the control flow. Those tasks refer to the collaboration uses over which the view process participates in collaborations. In Figure 11 the control flow is depicted in a simplified way as arrows with dashed lines. It shows the complete description of the view process'

executable process. Those parts of the executable process relevant to a publicly visible abstract process, are bound to the respective collaborative process (i.e. collaboration).

6. Discussion

In this paper we have introduced and compared architectures being feasible target platforms of a model-driven transformation of cross-enterprise business processes from the computational independent down to platform-independent level. We showed how the information necessary to automatically create platform-independent ICT-level models for a service-oriented environment can be derived from business process models. We identified a number of modelling constructs allowing us to derive platform-independent architectures that can be mapped to different ICT architectures.

Having implemented both central, decentral and brokerless approaches, the major insights we gained are as follows: While all three approaches can be derived from a CIM description without an explicit description of the CBP, we found it important that CBPs be explicitly modelled; otherwise, model transformation results are likely to be of poor quality. The decentral broker architecture relies on the existence of a CBP model to a higher degree than the central broker architecture and the brokerless architecture do: the latter can be derived more easily from the process flow; in the former, the appropriate grouping of processes to view processes in a decentralized broker must be specified explicitly.

The feasibility of the decentralised broker approach, i.e. generation of services and BPEL-processes out of ARIS-process descriptions as well as the execution of those service collaborations, has been shown and implemented in a prototype of the ATHENA IP [1] project. A more detailed description of the transformation rules can be found in [6].

Future work comprises developing model transformations and mechanisms for deriving runtime ICT-models from the conceptual models described in this paper. Also, it is yet unclear, which constraints on a runtime infrastructure of a brokerless approach can satisfy quality attributes already guaranteed by platform-independent decentralized broker architectures.

Part of the work reported in this paper has been funded by the ATHENA IP under the European grant FP6-IST-507849. It does not represent the view of E.C. nor that of other consortium members, and authors are fully responsible for the paper's content.

References

[1] ATHENA IP project web site. www.athena-ip.org.
[2] ATHENA A6. Specification of a basic architecture reference model. Deliverable D.A6.1, ATHENA IP, 2005. (downloadable from [1], Public Documents).
[3] Bass L., Clements P., Kazman R.: Software Architecture in Practice. Addison Wesley, 2003.
[4] Bauer B., Müller J.P., Roser S.: Adaptive design of cross organizational business processes using a model-driven architecture. 7. International Conference on Business Information Systems, Physica-Verlag, 2005.
[5] Bauer B., Müller J.P., Roser S.: A model-driven approach to designing cross-enterprise business processes. Volume 3292 of LNCS, Springer, 2005.
[6] Bauer B., Müller J.P., Roser S.: A Dezentralized Broker Architecture for Collaborative Business Process Modelling and Enactment. I-ESA'06, 2006.
[7] Benguria G., Larrucea X., Elvesæter B., Neple T., Beardsmore A., Friess M.: A platform-independent model for service-oriented architectures. I-ESA'06, 2006.
[8] Elvesæter B., Hahn A., Berre A.-J., Neple N.: Towards an Interoperability Framework for Model-Driven Development of Software Systems. I-ESA'05, 2005.
[9] Frank, Gardner, Johnston: Business Process Definition Metamodel – Concepts and Overview. IBM, 2004.
[10] Frankel D.S.: Model Driven Architecture – Applying MDA to Enterprise Computing. Wiley, 2003.
[11] IBM, Business Process Execution Language for WS, http://www.ibm.com/developerworks/library/ws-bpel/.
[12] Lippe, Greiner, Barros: A Survey on State of the Art to Facilitate Modelling of Cross-Organisational Business Processes. In Proc. of XML4BPM 2005, 2005.
[13] Liu D.-R., Shen M.: Modeling workflows with a process-view approach. Proc. 7th Internat'l Conf. on Database Systems for Advanced Applications, 2001.
[14] Model-Driven Architecture homepage. Object Management Group, www.omg.org/mda.
[15] OMG: UML2 superstructure, formal/05-07-04.
[16] Scheer A.W.: ARIS – Business Process Modeling, 3rd edition. Springer, 2000.
[17] Schulz K.A., Orlowska M.E.: Facilitating cross-organisational workflows with a workflow view approach. Data & Knowledge Engineering 51(1), 2004.