Model-Based Development with SoaML

Brian Elvesæter, Cyril Carrez, Parastoo Mohagheghi, Arne-Jørgen Berre, Svein G. Johnsen and Arnor Solberg

1 Introduction and Overview

Our MDSE methodology aims to integrate with existing business modelling practices within an enterprise, allowing building upon and extending existing modelling practices rather than replacing them. The methodology assumes that modern business modelling practices take advantage of business modelling tools that adopts the OMG MDA specifications Business Motivation Model (BMM) [1] and Business Process Modeling Notation (BPMN) [2]. From these models we will drive the specification of services as a set of SoaML model artefacts.

The MDSE methodology provides guidelines for how to use SoaML to define and specify a service-oriented architecture from both a business and an IT perspective. The methodology prescribes building a set of model artefacts following the iterative and incremental process paradigm. Figure 1 depicts the overall process and identifies the set of model artefacts to specify. The figure shows the set of work products prescribed by the methodology and the overall workflow. The icons indicate the associated BMM, BPMN or SoaML diagram(s) for each work product and

Brian Elvesæter
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: brian.elvesater@sintef.no

Cyril Carrez
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: cyril.carrez@sintef.no

Parastoo Mohagheghi
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: parastoo.mohagheghi@sintef.no

Arne-Jørgen Berre
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: arne.j.berre@sintef.no

Svein G. Johnsen
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: svein.g.johnsen@sintef.no

Arnor Solberg
SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway, e-mail: arnor.solberg@sintef.no

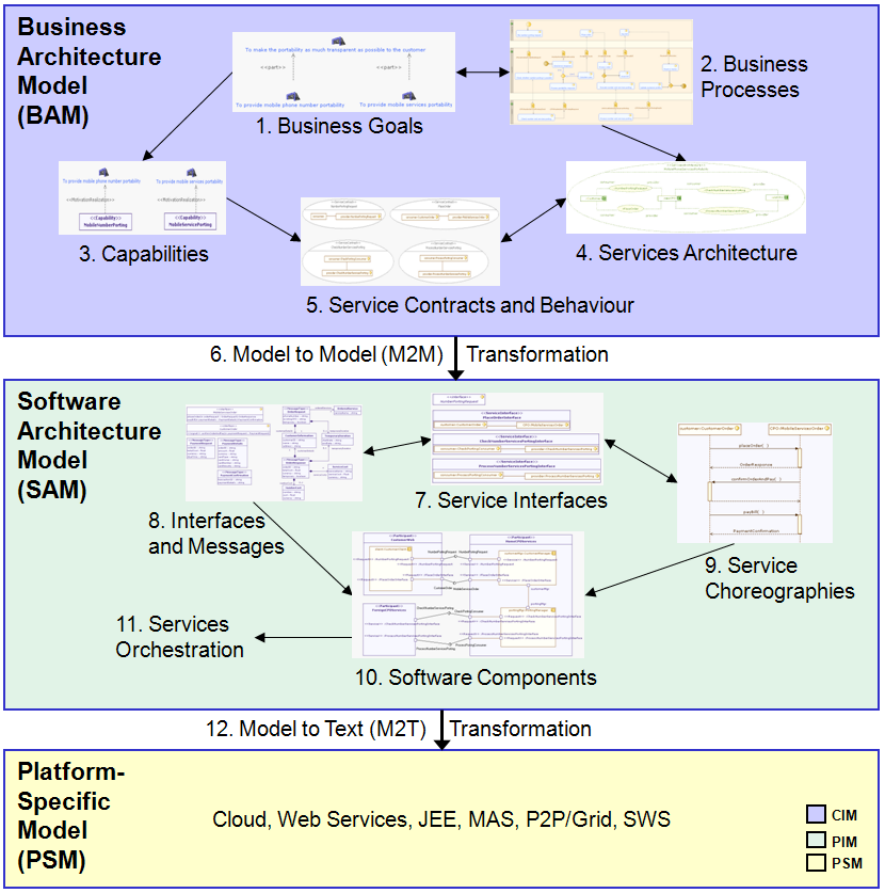

Fig. 1 The overall model-driven process

the arrows show the most common path through the set of work products within an iteration.

Starting from the upper left we have the *Business Architecture Model (BAM)* which includes the business goals, business processes, capabilities, services architecture, and service contracts and behaviour. The *Software Architecture Model (SAM)* specifies the interfaces and message types, services interfaces and behaviour, and components and ports. The *model-to-model (M2M) transformation* consists of transformation rules and procedural guidelines to support a semi-automated mapping from BAM to SAM.

The *Platform-Specific Model (PSM)* contains the design and implementation artefacts of the specified service-oriented architecture in the chosen technology platforms, e.g. cloud, Web Services, Java Enterprise Edition (JEE), multi-agent systems (MAS), peer-2-peer (P2P), grid and Semantic Web Services (SWS). We consider PSM-level modelling guidelines out of scope for the presentation of the methodol-

ogy in this chapter and focus on the business architecture and service architecture modelling. PSM-level extensions to the methodology would involve defining further modelling guidelines and *model-to-text (M2T) transformation* rules for the technology platforms.

We use and extend the Eclipse Process Framework (EPF)[1] for implementing the methodology. EPF is a process framework that allows to define methodology and process content that can be customized and integrated with other engineering methodologies. The methodology presented here is dependent on modelling tools that supports the OMG specifications BMM, BPMN and SoaML. In the example presented in this chapter we have used the UML modelling tool Modelio[2].

2 Illustrative Scenario

We have developed a methodology that takes advantage of existing business modelling practices and provides a step-to-step guide for specifying services uses SoaML concepts. The methodology is illuminated using the telecommunication scenario as introduced for the book. The scenario is about mobile phone services portability and has the following business goals:

- To provide mobile phone number portability.
- To provide mobile services portability.
- To make the portability as much transparent as possible to the customer.

In the following sections we provide guidelines for how to specify the services and their contracts starting from the business processes descriptions. Specifically, we focus on the first goal addressing phone number portability. The starting point is a BPMN process description involving a customer and a number of cell phone operators (CPOs). The scenario assumes that cooperation between different CPOs as well as internal cooperation between the different departments of a CPO. The process diagram shown in Figure 2 describes the request and assignment of mobile services portability. When a customer requires a telephone number portability, the CPO has to provide not only the porting of the number, but also the porting of the services enabled on it, when possible. After checking the portability of the number the CPO executes the porting. At the end of the process the number is activated and bound to the new CPO.

3 Business Architecture Modelling

This part of the methodology covers selected areas of CIM-level modelling resulting in a *Business Architecture Model (BAM)* that describes the business perspective

[1] Eclipse Process Framework, http://www.eclipse.org/epf/

[2] Modeliosoft, http://www.modeliosoft.com/

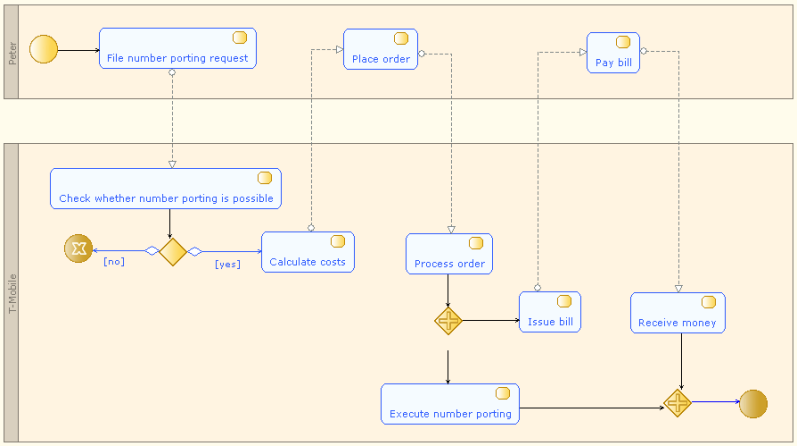

Fig. 2 Request and assignment of mobile phone services portability (BPMN diagram)

of a service-oriented architecture. The BAM is used to express the business operations and environment which the service-oriented architecture is to support. The BAM includes business goals (Subsection 3.1), business processes with associated organisation roles and information elements (Subsection 3.2), and capabilities (Subsection 3.3) that are relevant for capturing business requirements and identify services within a service-oriented architecture. The BAM further describes the services architecture (Subsection 3.4) of the business community and the service contracts (Subsection 3.5) between the business entities participating in the community. Figure 3 depicts an activity diagram that shows the modelling tasks involved in the specification of the Business Architecture Model.

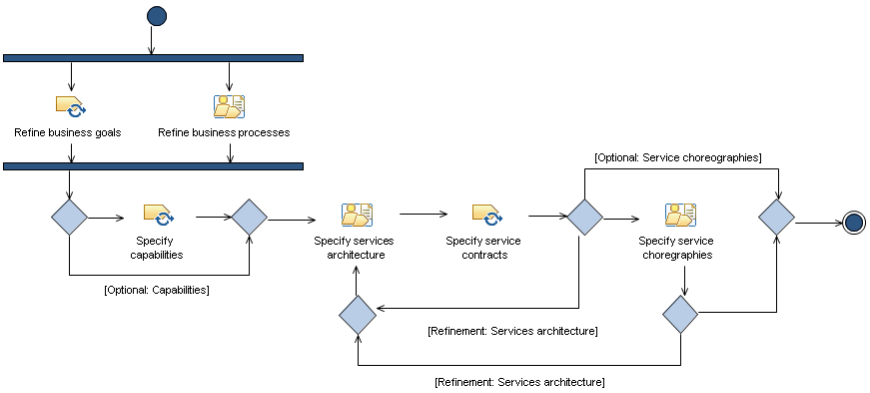

Fig. 3 Business architecture modelling activities (EPF activity diagram)

3.1 Business Goals

Business Motivation Model (BMM) [1] is a business-level modelling technique that supports the modelling of the business goals and objectives, the means and policies to achieve them, and the influencing factors that drive and control the work involved. This is typically used to define the overall business strategy in early phases of an engineering project.

The purpose of the BMM is to agree with the business stakeholders the business goals that will be met by implementing the service-oriented architecture, so that a set of required high level business processes can be identified for further analysis. BMM provides a scheme or structure for developing, communicating, and managing business plans in an organized manner. In particular, the BMM supports the following:

- It identifies factors that motivate the establishment of business plans.
- It identifies and defines the elements of business plans.
- It indicates how all these factors and elements inter-relate.

Among these elements you find business policies and business rules that provide governance for and guidance to the business. Once produced and agreed, the BMM serves as a reference that ensures that a full assessment may be made of all the business implications of any proposed changes to the service-oriented architecture.

3.1.1 Goal Modelling Steps

The specification of the goals and company's motivations are defined in a business goals diagram. The goals can be structured by applying containers within the BMM model. The containers contain one or more goals diagrams in which you can define the goals and their relationships.

- **Step 1: Identify goals.** Business goals are discovered by a process of workshops and interviews involving relevant stakeholders. The BMM describes a loose hierarchy of goals of the business within the particular area of concern, from the goals of a business stakeholder in developing a product to the business goals met by the product or its users.
- **Step 2: Specify goals and their relationships.** The goals are created as classes containing motivation related information. The name is expressed in a natural language and has properties such as scope, quantitative/qualitative, value, etc. Relationships between goals such as "part of", "positive influence", "negative influence", "guarantee" and "measure" can be specified. Figure 4 shows the goals specified for the telecommunication example.
- **Step 3: Perform goal analysis to link the identified goals to existing or potential new business processes.** Goals may be thought of as high level statements of the things that have to be achieved in a business, each expressed as an outcome, but without specifying the "how". As goal analysis proceeds, the analyst

describes an enabling behaviour as the means by which a higher level outcome is to be achieved. Initially this is done through a brain-storming process. This activity results in a list of enabling behaviours that is further refined and consolidated into a set of enabling behaviours that supports all the business goals. This is the starting point for business process modelling in a BPMN tool.

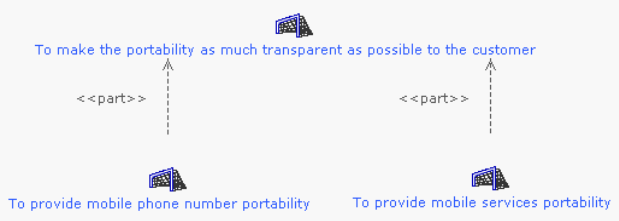

Fig. 4 Business goals diagram (BMM diagram)

3.2 Business Processes

Business Process Modeling Notation (BPMN) [2], is designed to communicate a wide variety of information on business processes to a wide variety of audiences, providing a standard notation that is readily understandable by all business stakeholders. Typically, BPMN is used to define business processes on the CIM level. The definitions are then mapped to more technical models on the PIM and PSM level.

In the BAM, a BPMN model defines the business processes of the domain which are relevant to the service-oriented architecture, and which will enable the goals to be met, and the roles of the resources that perform those processes. It includes a full definition of the roles in the business, focusing on those fulfilled by the system or component to be developed. Business process descriptions bring real business vision and constitute an excellent formalization and analysis tool when constructing systems. In the context of a development project they are used in business-oriented activities related to requirements, specifications and analysis.

Business processes may be at a number of levels of detail, from a high level description of the business processes down to task flows which is a set of detailed specifications for the business services that the service-oriented architecture will provide. BPMN is designed to cover many types of modelling and allows the creation of end-to-end business processes. It allows the specification of private processes (both non-executable and executable), public processes, choreographies and collaborations.

The BPMN modelling in the BAM consists of process, information and organization modelling steps.

3.2.1 Process Modelling Steps

Modelling in BPMN is done by simple diagrams with a small set of graphical elements. It should make it easy for business users as well as developers to understand the flow and the process.

Our methodology does not address the full scope of business modelling at the CIM level, but rather assumes that some kind of business process models or descriptions exist that have been developed using existing BPMN methods. However, typically these models must be further refined and related to our SoaML concepts for describing the business perspective of an SOA. We propose to refine business processes in separate BPMN diagrams that are linked to SoaML concepts. Ideally, this should be done in a modelling tool that allows us to create integrated models using both BPMN and SoaML language constructs. This is the case for Modelio which allows to integrate BMM, BPMN and SoaML models as used in our example. This ensures a tool-supported consistency between the modelling of the business processes, information and organization as outlined here. In the case of using separate tools for modelling BMM, BPMN and SoaML, manual consistency checks must be applied.

- **Step 1: Identify business processes.** Identify the relevant business processes for the service-oriented architecture following these guidelines:
  - Identify public and collaborative business processes that involve interactions and potential usage of software services between different business organizations. These processes are candidates for public *community-level services architectures* in SoaML that describe the service contracts between the business organizations.
  - Identify private business processes for the business entities under your ownership control that are involved in the services architecture under consideration. These processes are candidates for private *participant-level services architectures* in SoaML that describe the service contracts between the internal organizational roles or units within the business organization.
  - The business goals resulting from the modelling steps outlined in section 3.1.1 can be used to scope the selection, in particular if an existing goal analysis has been made from which you can select the business processes linked to the business goals.
- **Step 2: Refining business processes.** Each of the selected business processes identified is a candidate for a SoaML services architecture. Create new business process diagrams and refine the selected business processes following these guidelines:
  - Identify the business entities and model them as either pools (if different business organizations) or swimlanes (in the case of different organizational roles or units within the same organization). Pools and swimlanes are associated with participants in SoaML.

– Focus on the tasks that describe the interaction points between the business entities. These interaction points will be associated with service contracts in SoaML.
– Identify the data flows between these tasks. These data objects will be associated with message types or data entities in SoaML.

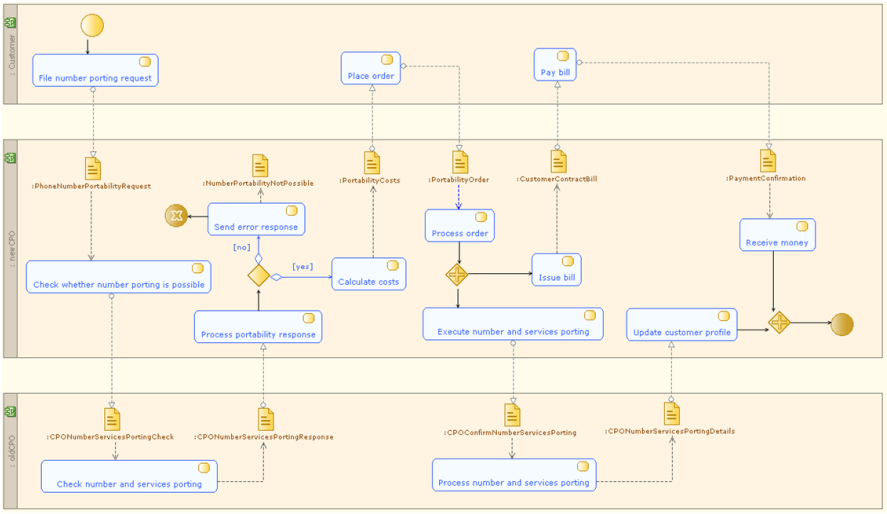

Fig. 5 Request and assignment of mobile phone services portability (BPMN diagram)

Figure 5 shows the refined business process model for the request and assignment of mobile phone services originally depicted in Figure 2. Three different participants take part in this process: *Customer*, *newCPO* and *oldCPO*. The interaction points between the participant tasks have been further revised by modelling data objects that are typed as specific message types, e.g. *PhoneNumberPortabilityRequest*. We assume that the *Customer* and *newCPO* are under our ownership control, thus we focus on the details in these two participants. The tasks in the *oldCPO* participant are considered private, thus from the perspective of the *newCPO* we only model the public information exchange.

3.2.2 Information Modelling Steps

A traditional requirement of process modelling is to be able to model the items (physical or information items) that are created, manipulated, and used during the execution of a process. This requirement is realized in BPMN through various constructs:

- **Data objects:** Represent the data and documents in a process. Data objects usually define the inputs and outputs of activities.

- **Data inputs and outputs:** Data requirements are captured as data inputs and input sets. Data that is produced is captured using data outputs and output sets.
- **Data associations:** Used to move data between data objects and inputs and outputs of activities and processes.
- **Messages:** Represent messages/communications between two separate participants.

BPMN is constrained to support only the concepts of modelling that are applicable to business processes. This means that other types of modelling done by organizations for business purposes such as data and information models is out of the scope of BPMN. We therefore introduce an extension to BPMN that uses the SoaML message type to detail the information elements. Class diagrams will be used for describing these SoaML message types. If the business model contains many business processes we suggest to create a class diagram describing the information for each business process.

- **Step 1: Identify the information items.** Go through your business process diagrams and identify the information items. Each of these information items maps to a message type or a data entity in SoaML. Message types and data entities are defined as stereotypes on a class. So a first step would just be to create these information items as regular UML classes, and then we will refine them to either message types or data entities as part of the service architecture modelling.
- **Step 2: Refine the information classes.** The information modelled here does not need to be complete. It may be sufficient to just link the class to a particular information standard or just describe the most important properties of the data objects.

Figure 6 shows a partial view of the data objects identified in the BPMN process (Figure 5). These data objects are represented as SoaML message types and have been linked with the BPMN data objects.

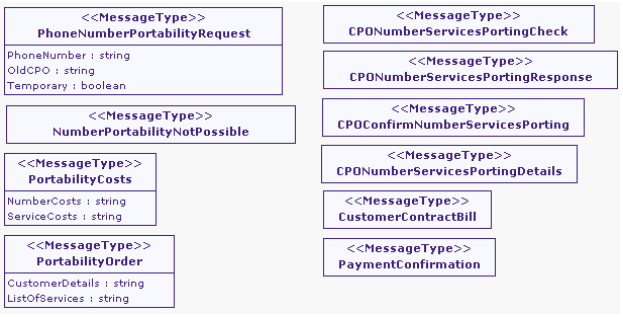

Fig. 6 Message types (UML class diagram)

3.2.3 Organization Modelling Steps

Another traditional requirement of process modelling is to be able to model the organization roles or units that are involved in the execution of a process. This requirement is realized in BPMN through various constructs:

- **Pools:** A pool acts as a container for a process, each one representing a participant.
- **Participants:** Participants define a general business role, e.g., a buyer, seller, shipper or supplier, or they can represent a specific business entity.
- **Lanes:** Lanes create sub-partitions for the objects within a pool and often represent internal business roles within a process.

However, since BPMN is constrained to support only the concepts of modelling that are applicable to business processes, also definition of organizational models and resources is out of the scope of BPMN. We therefore introduce an extension to BPMN that uses SoaML participants to detail the organization roles used in the business process diagrams. Class diagrams will be used for defining the SoaML participants.

- **Step 1: Identify the participants.** Define your main organization units and their internal roles. Go through your business process diagrams and map the pools, participants and lanes to SoaML participants. It is recommended to group the organization unit and any associated internal roles within a UML package. Within this package create UML class diagrams where you define the SoaML participants. These participants should be linked with the pools and lanes in the BPMN diagrams in order to maintain consistency between the BPMN and SoaML models.
- **Step 2: Refine the participants.** Indicate whether these participants represent an organization unit (e.g. company or department), a organization roles (e.g. chief engineer or secretary), a specific software system or undecided. In some cases we see that business process models define a specific lane as interacting with a planned or existing software system. In the business perspective of an SOA we focus on the participants that use and require software services in service contracts, while in the IT perspective of an SOA we focus on the specifications of the software components that realize the software services. In SoaML the concept of a participant is used to represent both of these aspects.

Figure 7 shows the pools identified in the BPMN process (Figure 5). These pools are represented as SoaML participants and have been linked with the BPMN pools.

Fig. 7 Participants (UML class diagram)

3.3 Capabilities

Capabilities identify or specify a cohesive set of functions or resources that a service provided by one or more participants might offer. Capabilities can be used by themselves or in conjunction with participants to represent general functionality or abilities that a participant must have. Capabilities are used to identify needed services, and to organize them into catalogues in order to communicate the needs and capabilities of a service area, whether that be business or technology focused, prior to allocating those services to particular participants. Capabilities can have usage dependencies on other capabilities to show how these capabilities are related. Capabilities can also be organized into architectural layers to support separation of concern within the resulting service architecture.

3.3.1 Capability Modelling Steps

The specification of capabilities should be considered optional in our methodology. Capabilities represent high-level services. For large business architectures it may be a good idea to start with capability modelling to help identifying needed services, whereas for more technical architectures it might be easier to focus on the IT services directly.

- **Step 1: Identify capabilities.** Once you have defined the capabilities, these can later be used to identify candidate services. In order to identify the capabilities in the first place there exists different techniques:
  - Goal-service modelling, which identifies capabilities needed to realize business requirements such as strategies and goals
  - Domain decomposition, which uses activities in business processes and other descriptions of business functions to identify needed capabilities
  - Existing asset analysis, which mines capabilities from existing applications
- **Step 2: Refine the capabilities.** Define abstract operations on the capabilities. At this specification level it is not required to specify the operation signature, but rather an operation that indicates a planned or existing business operation. If possible in the tool, you should also link the capabilities to the goals, business processes and participants (both business organizations and software components) from which the capabilities were derived.

Figure 8 shows a simple example of two capabilitites that were derived from the business goals (Figure 4).

Fig. 8 Capabilities (UML class diagram)

3.4 Services Architecture

A *services architecture* is a high level description of how participants work together for a purpose by providing and using services expressed as *service contracts*. The services architecture defines the requirements for the types of *participants* and service realizations that fulfill specific roles. A *role* defines the basic function (or set of functions) that an entity may perform in a particular context. In contrast, a participant specifies the type of a party that fills the role in the context of a specific services architecture. Both service contracts and participants can be reused when composing different services in other services architectures.

A services architecture has components at two levels of granularity: The *community services architecture* is a ”top level” view of how independent participants work together for some purpose. The services architecture of a community does not assume or require any one controlling entity or process. A participant may also have a *participant services architecture*, which specifies how parts of that participant (e.g., departments within an organization) work together to provide the services of the owning participant. Participants that realize this specification must adhere to the architecture it specifies.

3.4.1 Services Architecture Modelling Steps for Community

A services architecture for a community is modelled as a UML collaboration. We recommend to create a UML package for each services architecture.

- **Step 1: Create a services architecture.** Create a UML collaboration with the stereotype <<ServicesArchitecture>> which specifies the services architecture.
- **Step 2: Identify the participants.** Identify the different participants involved in the services architecture. Participants are components stereotyped <<Participant>>. This was initially done as part of the BPMN modelling described in Section 3.2.3.
- **Step 3: Identify the service contracts.** Identify the possible interactions between the different participants. The interactions are represented as service contracts. A service contract is a UML collaboration with the stereotype <<ServiceContract>>. In this step you will only specify an empty UML collaboration. The detailing of the service contracts will be further elaborated in Section 3.5.1.

- **Step 4: Specify the services architecture.** Once all service contracts and participants have been identified, the service designer can use them to build the service architecture. This step is quite straightforward, as it consists of just putting together the two previous steps. Roles in the UML collaboration are typed by the identified participants, while UML collaboration uses are linked to the service contracts. The modeller has only to bind the different roles to the appropriate collaboration uses, hence specifying how participants will interact.

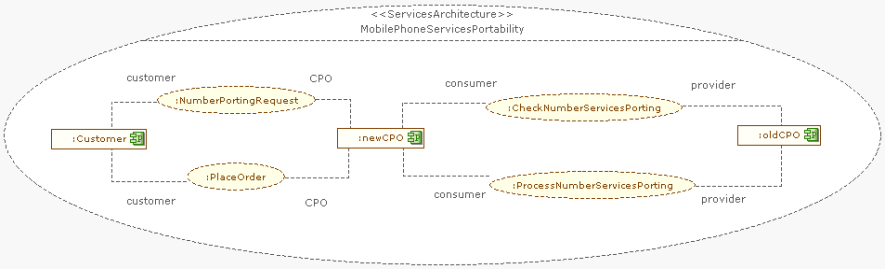

Fig. 9 Services architecture (UML collaboration diagram)

Figure 9 shows the services architecture for the mobile phone services portability scenario. The modelling of the services architecture is closely linked with the refinement of the BPMN business process (Figure 5). The services architecture defines the three same roles: *Customer*, *newCPO* and *oldCPO* linked to their respective participants. Also shown in the figure are collaboration uses that are linked to the service contracts: *NumberPortingRequest*, *PlaceOrder*, *CheckNumberServicesPorting* and *ProcessNumberServicesPorting*.

3.4.2 Services Architecture Modelling Steps for Participant

A services architecture for a participant is modelled as a UML collaboration. We recommend to create a UML package for each services architecture. The modelling follows the same steps as those outline in Section 3.4.1 only now you focus on defining service contracts between the internal organizational roles or units of the participant. and service contracts for the participant. These participants and service contracts should be regarded as private compared to the public community-level services architecture. A starting point for defining these internal service contracts are private business processes.

3.5 Service Contracts and Behaviour

SoaML allows different approaches to specifying services. In our methodology, we have chosen to combine two different approaches, as we see one fits more at the business level (service contract approach) and the other at the IT level (service interface approach). In the business architecture modelling we suggest to use service contracts that are further refined to service interfaces in the service architecture modelling.

A *service contract approach* defines service specifications (the service contract) that define the roles each participant plays in the service (such as provider and consumer) and the interfaces they implement to play that role in that service. The service contract may also own a behaviour that specifies the sequencing of the exchanges between the involved participants as well as the resulting state and delivery of the capability.

The service contract represents an agreement between the involved participants for how the service is to be provided and consumed. This agreement includes the interfaces, choreography and any other terms and conditions. Service contracts are frequently part of one or more services architectures that define how a set of participants provide and use services for a particular business purpose or process. The Service contract approach is most applicable where an enterprise, community SOA architecture or choreography is defined and then services built or adapted to work within that architecture, or when there are more than two parties involved in the service.

3.5.1 Service Contract Modelling Steps

The specification of service contracts can be seen as the refinement of a services architecture. Service contracts are specified as UML collaborations in UML collaboration diagrams.

- **Step 1: Identify service contracts.** Analyse the BPMN diagrams. See Section **??** for advise on identifying SoaML concepts from the BPMN models. For each identified service contract create a UML collaboration with the stereotype <<ServiceContract>>.
- **Step 2: Identify interfaces.** Identify the providers and consumers of the service contract. They are specified as roles typed by an interface. At this modelling step we only identify the names and possibly some high-level operations in the interfaces. These interfaces will be further elaborated and refined as part of the software architecture modelling.

Figure 10 shows the four service contracts specified as part of the services architecture (Figure 9). The first one, *NumberPortingRequest*, represents a simple service where any consumer can use the service without any contractual obligations. It has two roles: *customer* and *CPO*, but only the *CPO* role has an interface type, namely

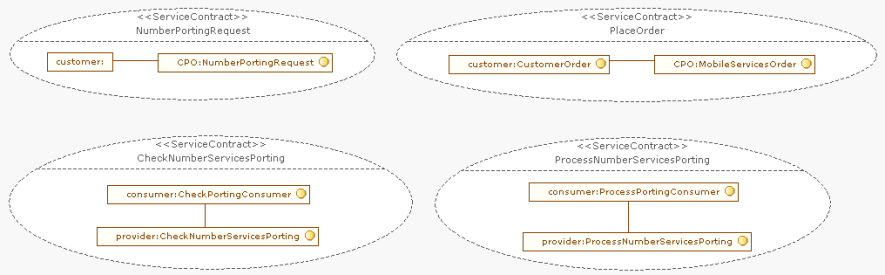

Fig. 10 Service contracts (UML collaboration diagram)

NumberPortingRequest. The three other service contracts specify that there are contractual obligations on both the consumer and provider side, which means that both roles must have an interface type. The service contract *PlaceOrder* specifies the interface *CustomerOrder* for the *consumer* role and the interface *MobileServicesOrder* for the *CPO* role. The two other service contracts specify the contract between the role *consumer* bound to the participant *newCPO* and the role *provider* bound to the participant *oldCPO* in services architecture (Figure 9).

3.5.2 Service Contract Behaviour Modelling Steps

An important part of the service contract is the choreography. A choreography is a specification of what is transmitted and when it is transmitted between participants to enact a service exchange. The choreography defines what happens between the provider and consumer participants without defining their internal processes - their internal processes do have to be compatible with their service contracts.

The behaviour of a service contract can be described at the business level using any UML behaviour diagram, e.g. state machine diagram, activity diagrams and sequence diagrams. We recommend to model the behaviour of any complex service contract in order to get a better understanding of the interaction between the roles. This model will later be refined at the IT level.

3.6 Mapping from Business Architecture Model to Service Architecture Model

This section describes mapping rules from BPMN to SoaML (Table 1) which can be used to align the business perspective and IT perspective of a service-oriented architecture. The mapping rules here are based on the BPMN 1.x specification. The upcoming BPMN 2.0 standard includes constructs for modelling services at both the BAM and SAM-level.

Furthermore, since our methodology prescribes the usage of SoaML concepts at both the business and software model levels, we also provide mapping rules from BAM concepts to SAM concepts (Table 2). The mapping rules can be used to guide the model refinements following our methodology or supported by a semi-automated approach based on a model-to-model (M2M) transformation.

Table 1 BPMN to SoaML mapping rules

BPMN concept	**SoaML concept**	**Description**
Task	Interface	A task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process.
Process	Services Architecture	A process in BPMN describes the interactions between different participants.
Pool / Role / Lane	Participant	A pool in BPMN stands for a business entity or a participant of a process. A lane represents a participant or a department in BPMN and is situated in a pool.
Annotation (Message)	Message Type	The annotation construct in BPMN is connecting two participants, from which one is providing and another is consuming the service in question. As the service provided is not described extensively in BPMN, it is not feasible to get to a ServiceContract message content exactly, so the next abstraction would exactly refer to the MessageType, thus providing the mechanism for reflecting the "loose" messages between different participants.
Pattern(roles, tasks, messages)	Service Contract	The service contract construct is a complex one even in SoaML itself. There is also no single construct at the CIM-level representing this entity, but rather a certain pattern of objects. Example of pattern: Two single tasks following one after another in a container, connected with a sequence flow and associated with one data object.

Table 2 BAM to SAM mapping rules

BAM concept	**SAM concept**	**Description**
Task	Interface	As a task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process.
Pool / Role / Lane / Participant	Participant	A BPMN pool, role or lane can represent an IT artefact. In this case it will map to a SoaML participant representing a software component. The SoaML participant identified in the BAM will need to be mapped to a supporting software component modelled as a SoaML participant.
Message / Message Type	Message Type	A BPMN message can be mapped to a SoaML message type. Message types at the BAM-level may be further refined as separate message types at the SAM-level.
Service Contract	Service Interface / Interface	The service contracts identified in the BAM map almost one-to-one to service interfaces. The only exception is in the case of simple interfaces, where the service contract maps to an interface.

4 System Architecture Modelling

The *System Architecture Model (SAM)* describes the overall architecture of the system at the PIM level. It partitions the system into components and defines the components in terms of what interfaces they provide, what interfaces they use, and how these interfaces should be used (protocol). Two aspects of component collaborations are described: the static model (structure) and dynamic model (behaviour). The structural model describes the components, their dependencies, and their interfaces; the dynamic model describes the component interactions and protocols.

We follow a top-down approach that progresses by refinement, starting from a high level specification of the system (the *services architecture*, a composite collaboration representing the service), to a fine grained specification (components, interfaces and data). Figure 11 depicts an activity diagram that shows the modelling tasks involved in the specification of the System Architecture Model.

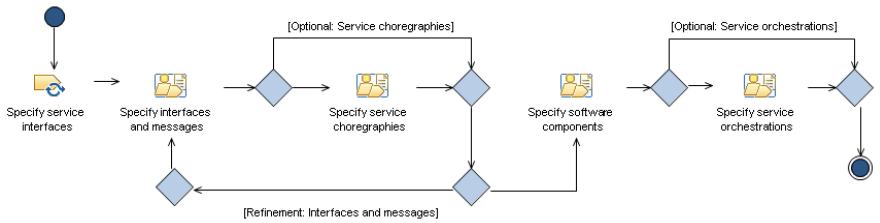

Fig. 11 Software architecture modelling activities (EPF activity diagram)

4.1 Service Interfaces

The service contracts specified in the business architecture modelling are refined to service interfaces in the services architecture modelling. SoaML allows for two types of interfaces:

- **Simple interfaces:** A *simple interface* focuses on a one-way interaction provided by a participant on a port represented as a UML interface. The participant receives operations on this port and may provide results to the caller. This kind of one-way interface can be used with ”anonymous” callers and the participant makes no assumptions about the caller or the choreography of the service. The one-way service corresponds most directly to simpler remote procedure call (RPC) style Web services.
- **Service interfaces:** A *service interface* based approach allows for bi-directional services, those where there are ”callbacks” from the provider to the consumer as part of a conversation between the parties. A service interface defines the

interface and responsibilities of a participant to provide or consume a service. It is used as the type of a service or a request port. A service interface may include specific protocols, commands and information exchange by which actions are initiated and the result of the real world effects are made available as specified through the functionality portion of a service.

4.1.1 Service Interface Modelling Steps

Interfaces and service interfaces are modelled as UML interfaces and classes and specified using UML class diagrams.

- **Step 1: Define the service interfaces.** Create the service interface as a UML class with the stereotype <<ServiceInterface>>. The service contracts can be used to identify the service interfaces.
- **Step 2: Define the provided and required interface.** Define the provided and required interface for the service interface. These required and provided interfaces are modelled as UML interfaces and represented in the service interface as parts with a connection. The required and provided interfaces are refined from the interface types of the consumer and provider roles defined in the service contracts. However, in the service contracts these interfaces were primarily defined as business-level interfaces. These interfaces can be refined directly, or possibly mapped to a new set of interfaces that will be used to detail the IT-level artefacts.

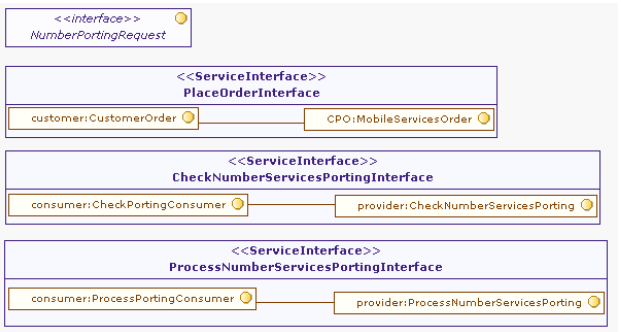

Fig. 12 Service interfaces (UML class diagram)

Figure 12 shows the service interfaces for the four service contracts presented earlier. Recalling that one of the service contracts represented a simple interface, this has been mapped to a new interface *NumberPortingRequest*. The three other service contracts have been mapped to the corresponding service interfaces *PlaceOrderInterface*, *CheckNumberServicesPortingInterface* and *ProcessNumberServicesPortingInterface*. All of these service interfaces contains a new set of interfaces that will be used to further detail the specification for the software components.

4.2 Interfaces and Messages

There are several SOA interaction paradigms in common use including document centric messaging, remote procedure calls (RPC), and publish-subscribe. The decision depends on cohesion and coupling, state management, distributed transactions, performance, granularity, synchronization, ease of development and maintenance, and best practices. SoaML supports both document-centric messaging and RPC-style service data.

- **Document-centric messaging:** *Message types* specify the information exchanged between service consumers and providers. Message types represent ”pure data” that may be communicated between parties – it is then up to the parties, based on the SOA specification, to interpret this data and act accordingly. As ”pure data” message types may not have dependencies on the environment, location or information system of either party.
- **RPC-style service data:** *Service data* is data that is exchanged between service consumers and providers. The data types of parameters for service operations are typed by a data type, primitive type, or message type.

4.2.1 Interface Modelling Steps

- **Step 1: Decide on document-centric or RPC-style.** Decide for each interface whether the operations should follow a document-centric or RPC-style approach. The choice affects how to model the operation signatures (parameters and responses). The choice may differ from interface to interface within the same services architecture, depending on, e.g. technology platform choices and whether we are specifying public external services or private internal services.
- **Step 2: Define the message types and service data.** Message types are represented as UML classes with the stereotype <<MessageType>>. Service data are represented as UML classes with the stereotype <<entity>>. The specification of message types and service data are closely linked to the specification of the operations in the interfaces. For a document-centric approach you will typically only specify one input parameter and one response parameter that are types as message types. A message type is a kind of value object that represents information exchanged between participants. This information consists of data passed into, and/or returned from the invocation of an operation or event signal defined in an interface. A message type is in the domain or service-specific content and does not include header or other implementation or protocol-specific information.
- **Step 3: Detail the structure and properties of the message types and service data.** Both message types and service data may have properties that can be either modelled as UML properties or associated UML classes. Define the necessary properties and associated classes to store the information to be exchanged between the consumer of the service and the provider of the service.

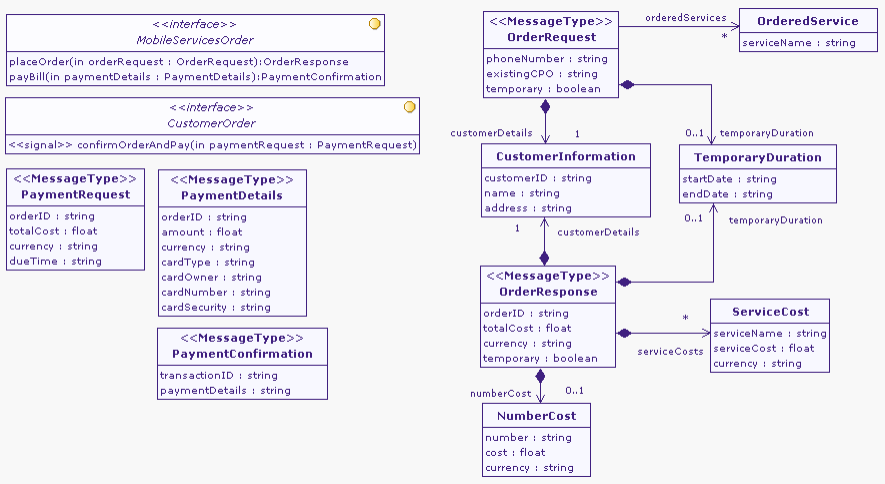

Fig. 13 Interfaces and message types (revised UML class diagram)

Figure 13 shows the interface and message types for the service interface *PlaceOrderInterface*. Three operations have been specified using with their own separate message types as input parameters and responses. The message types *OrderRequest* and *OrderResponse* have both properties and associated classes that contain additional information to support the customer in browsing the cost of the services before selecting the ones to be ported and enabling whether a temporary or fixed porting should be enabled.

4.3 Service Choreographies

The behaviour of a service interface expresses the expected interaction between the consumers and providers of services. It can be specified as any UML behaviour, the most common ones being activity, interaction or state machine. In the modelling steps below we outline the use of interactions in the form of sequence diagrams.

4.3.1 Service Choreography Modelling Steps

- **Step 1: Create a service interface behaviour diagram.** The behaviour of a service interface can be described using a UML sequence diagram. Create an interaction diagram to model the sequence of the service choreography.
- **Step 2: Define the service choreography between the interfaces.** Define the message sequence between the consumer and provider and interfaces. The sequence should be linked to operations defined in the interfaces. The modelling

of the service choreography is an iterative process that often leads to revising the interfaces and messages (Subsection 4.2.1). Iterate over this process of modelling steps until a complete service choreography can be specified.

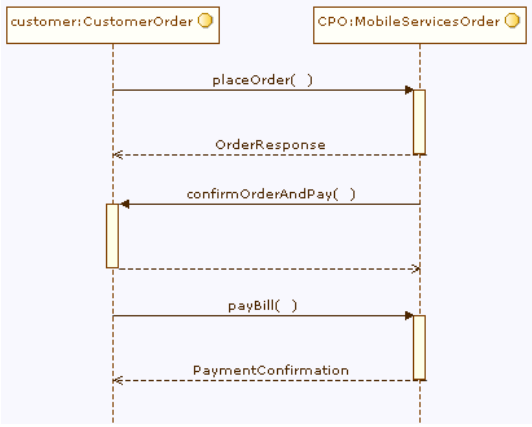

Fig. 14 Service choreography or service interface behaviour defined as a UML interaction (UML sequence diagram)

Figure 14 specifies the behaviour of the service interface *PlaceOrderInterface* that describes a bi-directional service with a call-back at the client-side. The *customer* invokes the operation *placeOrder* on the interface *MobileServicesOrder*. A message type *OrderResponse* is returned to the customer. Then the *CPO* invokes the operation *confirmOrderAndPay* on the interface *CustomerOrder* at the customer-side which triggers a signal forcing the customer to confirm and pay for the order and invoke the operation *payBill*.

4.4 Software Components

The component model focuses on specifying the involved software components that realizes the specified services architecture, either for a community or a participant. Once the components are defined a composite structure is used is show how implementations of these components form a composite service oriented application.

4.4.1 Component Modelling Steps

Software components are modelled as components with composite structures using class diagrams in UML.

- **Step 1: Identify components.** For each participant in the community services architecture you define a SoaML participant which represents a software system that realizes the service contracts specified for the business organizations (specified as SoaML participant) in the Business Architecture Model. For each participant in the participant services architecture you define a UML component that represents an internal software component of the corresponding software system. (In the case of complex participant services architecture you may specify these components as SoaML participants with the intent of specifying additional and more detailed service interfaces at software-level. For simpler services architectures going to traditional component software design should be considered instead.)
- **Step 2: Define the internal component structure.** Based on the private services architecture that you have defined for the participant, the internal components of the software component are specified. The software components are connected through the service interfaces that you have already defined.
- **Step 3: Define the component architecture for the community services architecture.** After the internal component architectures have been defined, you are able to assemble the component architecture that realizes the community services architecture using the public interfaces as defined by the components and connect these using service channels.

Figure 15 shows the assembly of a component architecture that corresponds to the community services architecture specified in Figure 9. Three system-level components (specified as SoaML participants) have been specified that correspond to the participants at the business-level. These three components are connected using *request* and *service* ports that are typed with matching service interfaces and connected with the corresponding required and provided interfaces. For the system component *HomeCPOServices* we have introduced two internal software components, *CustomerManager* and *PortingManager*, that could possibly be linked to a participant-level services architecture for the CPO under study.

References

1. Business motivation model (BMM), version 1.0. Object Management Group, OMG Document formal/08-08-02 (2008)
2. Business process modeling notation (BPMN) 1.2. Object Management Group, document formal/2009-01-03 (2009)

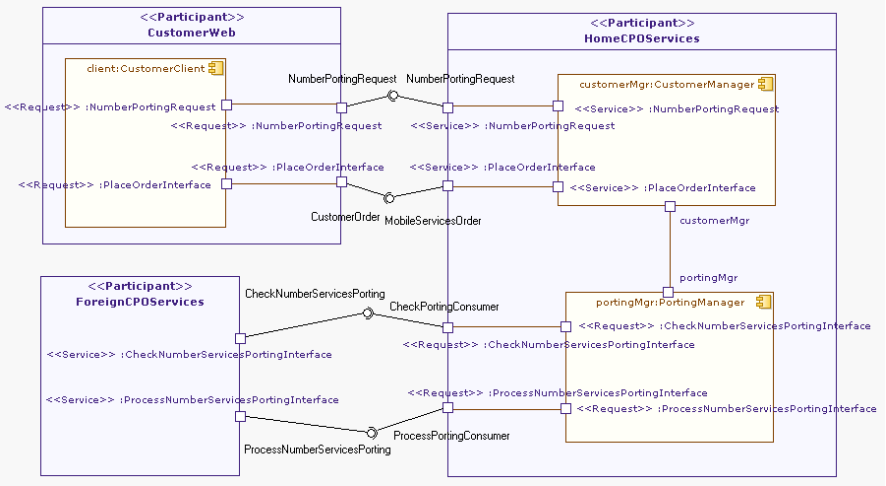

Fig. 15 Software components (UML class diagram)