Computers in Industry 64 (2013) 951–965

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Model-driven approach to enterprise interoperability at the technical service level

Ravi Khadka [a,*], Bramhananda Sapkota [b], Luís Ferreira Pires [b], Marten van Sinderen [b], Slinger Jansen [a]

[a] Utrecht University, P.O. Box 80.089, 3508TB Utrecht, The Netherlands
[b] University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 12 October 2012
Accepted 25 July 2013
Available online 23 August 2013

ABSTRACT

Enterprise Interoperability is the ability of enterprises to interoperate in order to achieve their business goals. Although the purpose of enterprise interoperability is determined at the business level, the use of technical (IT) services to support business services implies that interoperability solutions at both the business and technical level should be aligned. This paper introduces and demonstrates the suitability of an approach based on model transformations to automate enterprise interoperability. We start by considering that a set of enterprises are willing to interoperate in the context of their individual goals. The interactions necessary for their cooperation are then properly captured in terms of a so-called *choreography*. Our approach allows a choreography to be mapped and transformed to an *orchestration*, which defines the operation of the actual technical services of the interoperating enterprises. The paper discusses the technical challenges of implementing the transformation, and illustrates our approach with two application scenarios.

1. Introduction

Enterprise Interoperability (EI) can be defined as "the ability of enterprises to interoperate in order to achieve their business goals without special effort from the customer or user" [1]. Although the purpose of enterprise interoperability is determined at the business level, the extensive use of technical (IT) services to support business services implies that interoperability solutions at both the business and technical level should be aligned, in order to enable effective and efficient inter-enterprise collaborations [2]. However, problems caused by incompatibilities between technical enterprise systems (e.g., data inconsistencies, inconsistent interfaces) have been obstacles to EI. This holds especially in dynamic business environments, in which enterprises have to quickly adapt to changes in the internal organisation of the enterprise, in market demands and opportunities, in partners and intent of business collaborations (e.g., due to mergers and acquisitions), and in the supporting technologies [3].

Service-Oriented Architecture (SOA) has been introduced as a design principle for business and technical services [2,4–7]. SOA promotes reusability, since it prescribes the use of the service abstraction to decouple business functions from technology platforms, bringing many benefits to EI at both business and technology levels. Model-Driven Architecture (MDA, [8]) is an approach to systems development that prescribes the use of models to capture system abstractions, and the use of model transformations to automate development tasks [9,10]. The potential benefits of applying MDA to SOA at both business and technical service levels have been extensively acknowledged in the literature [4,6,11].

This paper aims at introducing and demonstrating the suitability of an approach based on model transformations to automate EI. This paper assumes that interoperability requirements from the business level are already captured by choreographies of technical services, and proposes an MDA-based transformation approach to move (semi-)automatically from these choreographies to interoperable technical services supported by orchestrations. The paper discusses our approach in detail, by considering the technical challenges of implementing the approach and its limitations. We start by assuming that a set of enterprises are willing to interoperate, i.e., they identify common grounds for meaningful interactions in the context of their individual goals. Based on these interactions, a model of the cooperation is defined, in terms of a so-called choreography. The

* Corresponding author. Tel.: +31 612053704.
E-mail addresses: r.khadka@uu.nl (R. Khadka), b.sapkota@ewi.utwente.nl (B. Sapkota), l.ferreirapires@ewi.utwente.nl (L. Ferreira Pires), m.j.vansinderen@ewi.utwente.nl (M. van Sinderen), slinger.jansen@uu.nl (S. Jansen).

0166-3615/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.compind.2013.07.006

proposed approach allows a choreography to be mapped and transformed to an orchestration, which defines the behaviour of an orchestrator component that coordinates the operation of the actual technical services of the interoperating enterprises. The transformation from a choreography to an orchestration can be automated to a large extent, which enhances the efficiency and accuracy of the development process.

This paper is further structured as follows: Section 2 gives the background of this work, Section 3 introduces and justifies the mappings between choreographies and orchestrations supported in our approach, Section 4 describes our transformation approach in more technical detail, Section 5 explains the implementation of the model transformations developed in this work, Section 6 presents two application examples, Section 7 discusses related work, Section 8 presents our design decisions and the limitations of our approach and Section 9 presents the conclusions and future work.

2. Background

Once enterprises agree to collaborate at the business service level, their technical services have to interoperate in order to support the business goals of this collaboration. The interoperability of technical services needs to be described at different levels, not only to allow the definition of interoperability requirements, but also to precisely prescribe the interactions (message exchanges) that are necessary in order to fulfil these interoperability requirements. The fulfilment of interoperability requirements determines the effectiveness (correctness) of the collaboration, and indirectly implies that the business goals are properly supported. Therefore techniques are necessary to allow the description of interoperability requirements and the derivation of proper technical solutions to fulfil these requirements. In our approach we want choreographies (message exchanges) ultimately to be enforced by an automated system, therefore a transformation from choreographies to orchestrations is necessary. Such transformation enforces the automatic derivation of proper solutions at technical service level, by fulfilling the interoperability requirements defined at business service level.

2.1. Choreographies

In our research, we assume that a collaboration among enterprises is described using a choreography model. A choreography model relates to technical services, but at the highest possible abstraction level. It abstracts from the internal processes of business actors, and constrains the interoperation of business actors in terms of abstract message exchanges. The relationship between enterprise collaboration at the business level and the most abstract technical level has been covered by several authors, and is outside the scope of this paper. For example, in [12,13], the authors describe a method to derive a choreography (a coordination model) from an enterprise collaboration model that focuses on the value exchange between business actors (a value model). In [14], value models and goal models (i.e., models that focus on the goals of the business partners in a collaboration) are considered in combination as a starting point for deriving a web services composition that support the enterprise collaboration.

A choreography defines the common behaviour of interacting parties in terms of ordering constraints, alternatives and time constraints [15]. Choreographies describe public message exchanges, in the sense that they do not reveal how each particular interacting party is internally organised to comply with the prescribed common behaviour. Since they abstract from the contribution of each partner to the collaboration and ignore the internal structure of the participants, choreographies are considered to be at a relatively high-level of abstraction [15–17].

We illustrate the choreography concept with a purchase order scenario, which consists of a customer, a sales department, a manufacturing department (manufacturer in short), a stock department, a shipment department and a billing department. The sales department receives the purchase order from customer and forwards it to the manufacturer, which in its turn requests the stock department service to check the availability of the ordered goods. In response, the stock department provides the information and the status of the ordered goods to the manufacturer. The manufacturer then requests the shipping department for the arrangement of the shipment of the goods, and receives a

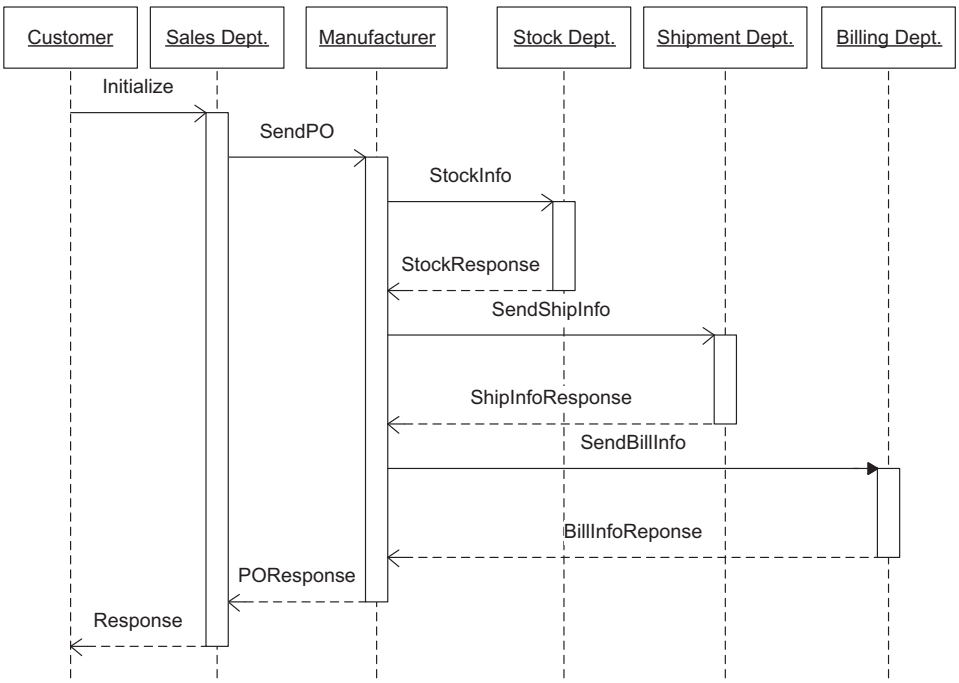

Fig. 1. Sequence diagram of purchase order scenario.

Listing 1
Example of WS-CDL collaboration constructs.

```
<roleType name = “ManufacturerDept”>
  <behavior name ="ManufacturerBehavior"/>

</roleType >
...
<relationshipType name =”ManuToBilling”>
<roleType typeRef=" tns:ManufacturerDept "/>
  <roleType typeRef=" tns:BillingDept"/>
</relationshipType>
...
<participantType name=” ManufacturerDeptParticipant”>
<roleType typeRef=”tns:ManufacturerDept“ />
</participantType>
...
<channelType name=”BillingChannel”>
<roleType behavior= “BillingBehavior” typeRef=” tns:BillingDept” />
<reference>
    <token name=”tns:BillingRef” />
</reference>
</channelType>
```

confirmation from the shipment department as response. Concurrently, the manufacturer invokes the billing department to calculate the total cost. Upon receiving the billing information, the manufacturer sends the purchase order response to sales department, which includes the date of the shipment and the total cost of the goods. Finally the customer receives the purchase order response through the sales department. Fig. 1 shows a sequence diagram that describes these interactions at a high-level of abstraction, representing the choreography of the collaboration between the services of the different departments.

Many languages have been defined in the last years to describe choreographies, such as WSCI [18], WS-CDL [19] and BPEL4Chor [20]. In this paper we consider choreographies described using WS-CDL, which is a W3C recommendation and as such it is widely used. WS-CDL is a declarative, XML-based language to specify choreographies in terms of collaborations, and is supported by (Eclipse) tools, like the Pi4SOA WS-CDL editor.[1]

A collaboration in WS-CDL is specified with the *participantType*, *roleType*, *relationshipType* and *channelType* constructs. A *participantType* represents an entity that plays a particular set of roles in the collaboration. A *roleType* represents a role in terms of a particular observable behaviour performed by a participant. A *relationshipType* defines the relation between *roleTypes*. A *channelType* is a point of communication between the *participantTypes*, and specifies where and how message is exchanged. Listing 1 describes the collaboration between the manufacturer and the billing department in the purchase order scenario presented in Fig. 1.

In addition, WS-CDL offers constructs to handle information and to define activities (behaviours), which are described as actions to be performed by one or more participants. Detailed information on WS-CDL is outside the scope of this paper, for which we refer to [19].

2.2. Orchestrations

An orchestration defines the coordination between services defined from the point of view of an orchestration process that handles this coordination [16,17]. An example of orchestration can be identified in Fig. 1, if we concentrate on the behaviour of the manufacturer service, and define it so that it can be performed by an orchestration engine.

[1] http://pi4soa.sourceforge.net/.

Some languages have been defined to describe orchestrations, like WSFL and XLANG, which have been combined in BPEL4WS [21], and its successor WS-BPEL [22]. In this paper we consider an orchestration defined using WS-BPEL since it is an OASIS standard and is strongly supported by (commercial) tools.

Similar to WS-CDL, WS-BPEL is XML-based, i.e., processes described with WS-BPEL are instances of the WS-BPEL XML schema. WS-BPEL is a language for describing the behaviour of a business process based on interactions between the process and its partners in either an executable or declarative (abstract) way. A business process can use services to invoke business functions, and the process itself can be exposed as a service. A *process* is at the root of a WS-BPEL process definition. A *partnerLink* allows the definition of relations between ports of the WS-BPEL process and ports of other services, and the assignment roles in these relations (*myRole* and *partnerRole*). Process variables are used to hold data that are needed to maintain the state of the WS-BPEL process.

WS-BPEL also allows the definition of basic and structured activities to define process behaviours. Some basic activities can be used to define an interaction with a partner, such as, for example, *invoke* to invoke a partner service, *receive* to receive a service invocation and *reply* to send a response message for a service invocation. Other basic activities are *assign* to update the value of variables, and *throw* to generate a service level fault. Structured activities are normally used to represent the ordering of the activities, and include *sequence* to represent sequential ordering, *flow* to represent activities executed in parallel, and *if*, *while* and *switch* to represent conditional execution, looping and conditional branching, respectively. Listing 2 shows an excerpt of the manufacturer process of the purchase order scenario discussed in Fig. 1.

Detailed information on WS-BPEL is outside the scope of this paper, for which we refer to [22].

3. Mapping choreographies to orchestrations

Since a choreography defines the common behaviour of cooperating services without considering how the individual contribution of these services is performed, a choreography can be considered to be at a higher level of abstraction than the orchestration(s) that implement the choreography. Furthermore, a given choreography can in principle be refined in terms of orchestration(s) in many alternative ways, depending on the responsibilities assigned to each individual orchestration or service. In the literature, two general strategies have been

Listing 2
Excerpt of WS-BPEL process for the manufacturer service.

```
<process name="ManufacturerDeptProcess" targetnamespace="http://www.pi4soa.org/purchaseOrder">
<partnerLink name="ManufacturerDeptParticipantType" myRole="ManufacturerDeptParticipantRole"
                    partnerLinkType="ManufacturerDeptParticipantLT"/>
<partnerLink name="BillingDeptParticipantType" partnerRole="BillingDeptParticipantRole"
                   partnerLinkType="BillingDeptParticipant"/>
...
<variable name="Billinglnfo" messageType="Billinglnfo" />
...
<sequence>
  <flow>
    <sequence>
      <invoke operation="sendBill" inputVariable="BillInfo" partnerLink="BillDeptParticipant"
outputVariable="..."/>
     <receive operation="SendBill" variable="BillInfoResponse" partnerLink="BillDeptParticipant" />
    </sequence>
   ...
  </flow>
</sequence>
</process>
```

identified to map choreographies to orchestrations, namely to use *decentralised orchestrations* or a *centralised orchestration* [3,23,24].

In our approach, we consider the transformation from a choreography to a centralised orchestration, because this strategy is more widely practiced and yields simpler results. In a centralised orchestration, an orchestrator component[2] is responsible for the coordination of all the participating services [23]. Conceptually we defined a mapping from the choreography to the orchestrator, which is the central component that coordinates the cooperation. This mapping embodies the design decision of applying centralised orchestration which is explained in detail in [3,26]. Since we consider that choreographies and orchestrations are defined using WS-CDL and WS-BPEL, respectively, we defined a mapping from WS-CDL language constructs to WS-BPEL language constructs inspired by [27,28] that complies with our conceptual mapping. Table 1 shows the mapping of WS-CDL to WS-BPEL language constructs that we considered in this work.

4. Transformation approach

We aimed at automating the transformation of a choreography specified in WS-CDL into a centralised orchestration specified in WS-BPEL. In order to be able to perform this transformation, we elicited the metamodels of these two languages, and defined a transformation from WS-CDL metamodel elements to WS-BPEL metamodel elements according to the mapping depicted in Table 1. This allows the transformation to be performed by a transformation engine, such that any WS-CDL model (instance of the WS-CDL metamodel) can be transformed into a WS-BPEL model (instance of the WS-BPEL metamodel). Fig. 2 shows our model transformation approach in terms of its MDA artefacts (metamodels, models and model transformation). In Fig. 2 we omit elements like the knowledge and pragmatics used to define the transformation.

This transformation approach has been defined (amongst others) in [8] and characterises (metamodelling-based) model transformation, which is one of the cornerstones of Model-Driven Engineering. In our approach, a source WS-CDL model is transformed into a target WS-BPEL model. The transformation specification is defined using the Atlas Transformation Language (ATL) [29] and defines transformation rules from WS-CDL metamodel elements into WS-BPEL metamodel elements, according to the mappings shown in Table 1. Since concrete WS-CDL and WS-BPEL specifications are represented as XML documents that comply with their corresponding XML schemas, we had to elicit the WS-CDL and WS-BPEL metamodels from their XML schemas in order to define the transformation in terms of metamodel elements. Initially, we automatically generated the WS-CDL and WS-BPEL metamodels from their respective schemas as explained in [30]. Many duplicated or unused model elements were generated in this way, which unnecessarily complicated the transformation process. Hence, we adjusted these metamodels manually from their respective XML schemas by removing the unnecessary elements. These metamodels are defined in Ecore, which is the metametamodel language of the Eclipse Modelling Framework (EMF).[3] This allows us to define and execute these transformations in the Eclipse workbench. Appendix A shows the WS-CDL and WS-BPEL metamodels used in the transformation process.

Furthermore, concrete WS-CDL specifications are represented as XML documents, and the ATL transformation engine expects models in Ecore (serialised using the XMI format, [31]) so we defined a transformation from the WS-CDL XML serialisation format to the Ecore/XMI format. The ATL transformation engine produces then an Ecore/XMI serialisation of the WS-BPEL model, which has to be translated to WS-BPEL XML syntax, e.g., in order to be executed by an orchestration engine. Therefore another transformation is necessary from the Ecore/XMI serialisation of the WS-BPEL model to its XML syntax. Fig. 3 shows the transformation chain that results from the concatenation of these transformations.

Fig. 3 shows the following transformations:

- *Auxiliary transformations*, to transform a WS-CDL specification (XML syntax) to a WS-CDL model in XMI (T1) and to transform a WS-BPEL model in XMI to a WS-BPEL specification (XML syntax) (T3).
- *Core transformation*, to transform a WS-CDL model in XMI to a WS-BPEL model in XMI (T2).

5. Transformation implementation

Fig. 4 shows the tools and languages we applied in order to implement the transformation chain[4] of Fig. 3. We use the Pi4soa

[2] The service offered by an orchestrator component is often called a mediation service [25].

[3] http://www.eclipse.org/modelling/emf/.

[4] The source code and all metamodels used in the transformation chain are available in http://people.cs.uu.nl/ravi/source/source.zip.

Table 1
Mapping of WS-CDL to WS-BPEL language constructs.

WS-CDL constructs	WS-BPEL constructs	Remarks
roleType	*process* per role	*bpel:targetNamespace* attribute is derived from the *cdl:targetNamespace* of *cdl:package*
participantType	*partnerLink*	
relationshipType	*partnerLinkType*	
variable	*variable*	*bpel:messageType* attribute is derived from the *cdl:type* of related *cdl:informationType*
channelType	*correlationSet*	*bpel:properties* is derived from *cdl:name* of *cdl:token* within *cdl:identity*
sequence	*sequence*	
parallel	*flow*	
choice	*if-else*	*bpel:condition* is manually defined by process designer
workunit		
repeat = false, block = false	*if-else*	*bpel:condition* is manually defined by process designer
repeat = true	*while*	*bpel:condition* is manually defined by process designer
block = true	–	No mapping
interaction		
action = request	*invoke*	Current party is mentioned in *cdl:fromRole*
action = request	*receive*	current party is mentioned in *cdl:toRole*
action = respond	*reply*	current party is mentioned in *cdl:fromRole*
action = respond	*receive*	Current party is mentioned in *cdl:fromRole* (synchronous reply)
assign	*assign*	
Finalise	*compensationHandler*	
noAction	*empty*	
silentAction	*sequence* with nested *empty*	*silentAction* is manually defined by process designer

WS-CDL editor to produce a choreography specification. Each transformation is discussed in the sequel.

5.1. Core transformation (T2)

The core transformation T2 has been described using ATL according to the mapping between WS-CDL and WS-BPEL metamodel elements shown in Table 1, and has been executed with the ATL engine for two different source models as explained in Section 6. The ATL engine reads a WS-CDL XMI source model as input, executes the transformation rules, and generates a WS-BPEL XMI target model as defined in the ATL transformation specification.

An excerpt of transformation T2 is shown in Listing 3, in which the *roleType* construct of WS-CDL is transformed to a WS-BPEL *process*. A *roleType* construct is used to specify the observable behaviour of a participant in the collaboration. In our approach we transform a choreography to a centralised orchestration, so we generate a *process* for the centralised orchestrator (manufacturer). The *name* and *targetnamespace* of the process is derived accordingly from the WS-CDL specification. The *variable*s associated with the *process* are derived from the *variable*s of the WS-CDL specification, *partnerlinks* of the process are generated from other *roleType*s of the WS-CDL specification, and the activity constructs of the process are generated from the activity constructs of the WS-CDL specification.

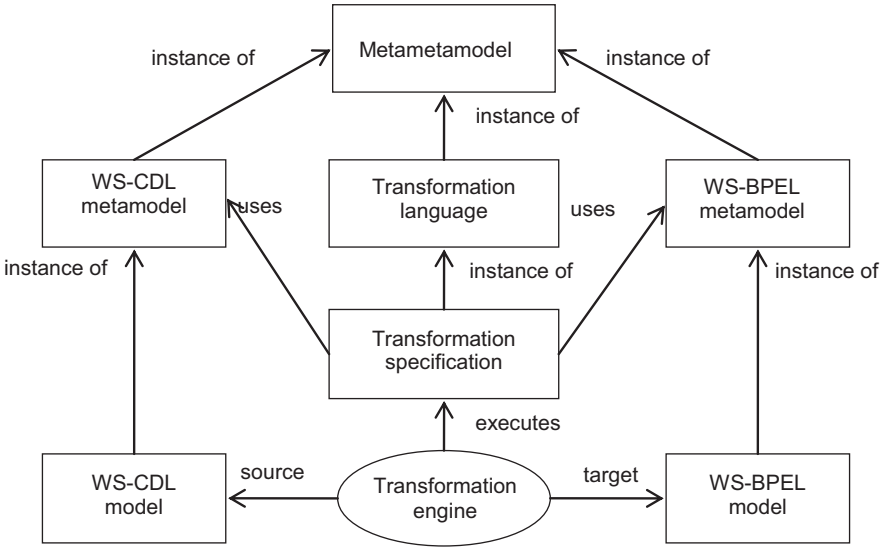

Fig. 2. Transformation approach.

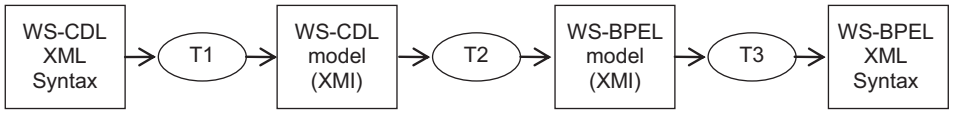

Fig. 3. Transformation chain from a WS-CDL specification to a WS-BPEL specification.

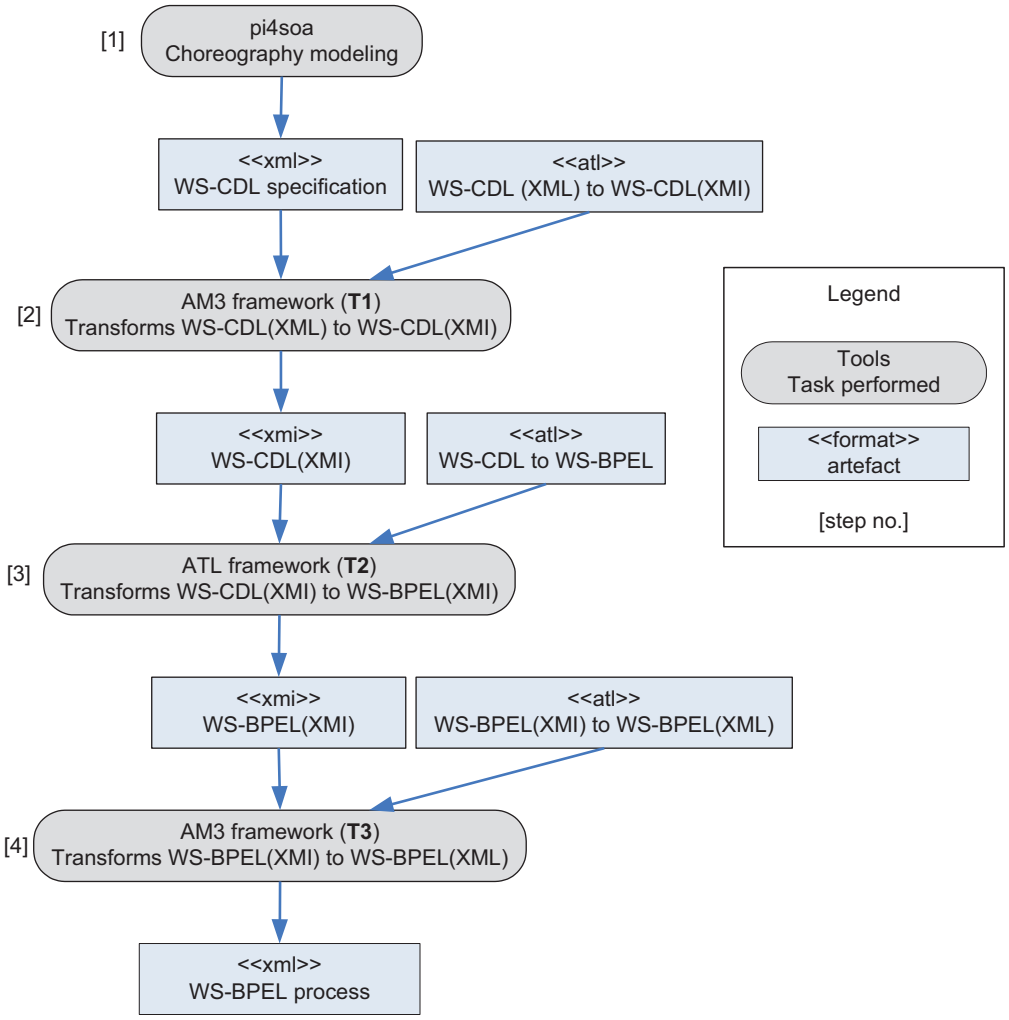

Fig. 4. Implementation steps.

Fig. 5 shows a diagrammatic representation of the transformation depicted in Listing 3 in terms of the mappings between metamodel elements.

5.2. Auxiliary transformations (T1 and T3)

The auxiliary transformations T1 and T3 have been implemented by leveraging the ATL's XML injector and extractor for injecting and extracting XML models into and from the XMI metamodel syntax, respectively. We used the AtlanMod Megamodel Management[5] (AM3) framework to implement the XML injector in transformation T1 and the XML extractor in transformation T3. In order to perform these transformations, we have used the XML metamodel, adapted from the AM3 zoo,[6] as shown in Fig. 6.

The code excerpt in Listing 4 shows part of transformation T1 in ATL between the XML source model and the WS-CDL XMI target model. However, running this code directly in the ATL engine does not result in the desired WS-CDL XMI format, so we use the XML injection mechanism. Similarly, in the transformation T3 we use the XML extraction mechanism to extract WS-BPEL XML code from the WS-BPEL XMI format.

The code excerpt in Listing 5 shows the ATL Ant task that invokes the XML injection of transformation T1. Task *am3.loadModel* loads a model with injectors and task *am3.atl* executes the *cdl2xmi.atl* transformation. Finally, task *am3.saveModel* is used to save the model in XMI format. Similarly XML extraction is also carried out in transformation T3 to transform WS-BPEL XMI model to WS-BPEL XML model.

Transformation T3 successfully generated the WS-BPEL skeleton from the given WS-CDL specification. The choreography has a higher abstraction level than the orchestration, so the choreography lacks the internal details of the participating services of the collaboration. Therefore, we had to manually add some missing details to the generated WS-BPEL process. After these details have been included, the WS-BPEL specification was validated for conformance with the WS-BPEL XML schema. We also imported the WS-BPEL process in the Eclipse BPEL designer tool [32] to check the behaviour of the orchestrator, and tested this behaviour for correctness, with successful results. The details of the auxiliary transformations can be found in [3,33].

6. Application examples

We validated our approach with two application scenarios: a purchase order scenario and a build-to-order scenario. The application scenarios are explained in the sequel.

6.1. Purchase order scenario

The purchase order scenario is used as an example to illustrate the usability of our approach, and is already introduced in Section

[5] http://wiki.eclipse.org/AM3.
[6] http://www.emn.fr/z-info/atlanmod/index.php/Ecore#XML_1.1.

Listing 3
Excerpt of transformation T2.

```
-- @path CDL=/test2Bpel/CDL.ecore
-- @path BPEL=/test2Bpel/BPEL.ecore
module cdlToBpel;
create OUT : BPEL from IN : CDL;

helper def : orchestrator : String = 'Manufacturer';
helper def : isOrchestrator() : CDL!RoleType =
    CDL!RoleType.allInstances() ->
    select(r | not r.isConnectedToChannelType()
    and r.name = thisModule.orchestrator)->first();

rule roleType2BPELprocess{
 from
   s: CDL!RoleType (not s.isConnectedToChannelType()
     and s.name = thisModule.orchestrator)
 to
   t: BPEL!Process(
   name <- s.name+'Process',
   targetNameSpace <- s.getTargetNamespace(),
   variables <- s.getVariables()
   ->collect(v | thisModule.varToVar(v)),
   partnerLinks <- s.getRoleTypes(),
   scopeElementActivity <- CDL!Activity.allInstances(),
   correlationset <- CDL!ChannelType.allInstances()
   )    }
```

2.1. The WS-CDL choreography specification has been produced using Pi4soa WS-CDL editor, and transformations T1, T2 and T3 have been performed on this specification. We obtained a WS-BPEL skeleton for the manufacturer process as result, which was validated against the WS-BPEL XML schema. The generated WS-BPEL specification was then imported in the Eclipse BPEL designer to check the behaviour of the orchestrator, which complies with the behaviour of choreography. Fig. 7 shows the WS-BPEL process of the orchestrator in the Eclipse BPEL designer.

6.2. Build-To-Order scenario

The Build-To-Order (BTO) application scenario, adopted from [27], is also used as an example to illustrate the usability of our approach. We briefly introduce this scenario with the sequence diagram shown in Fig. 8.

The BTO scenario consists of a customer, a manufacturer, and suppliers for CPUs, main boards and hard disks. The manufacturer offers assembled IT hardware equipment to its customers and has

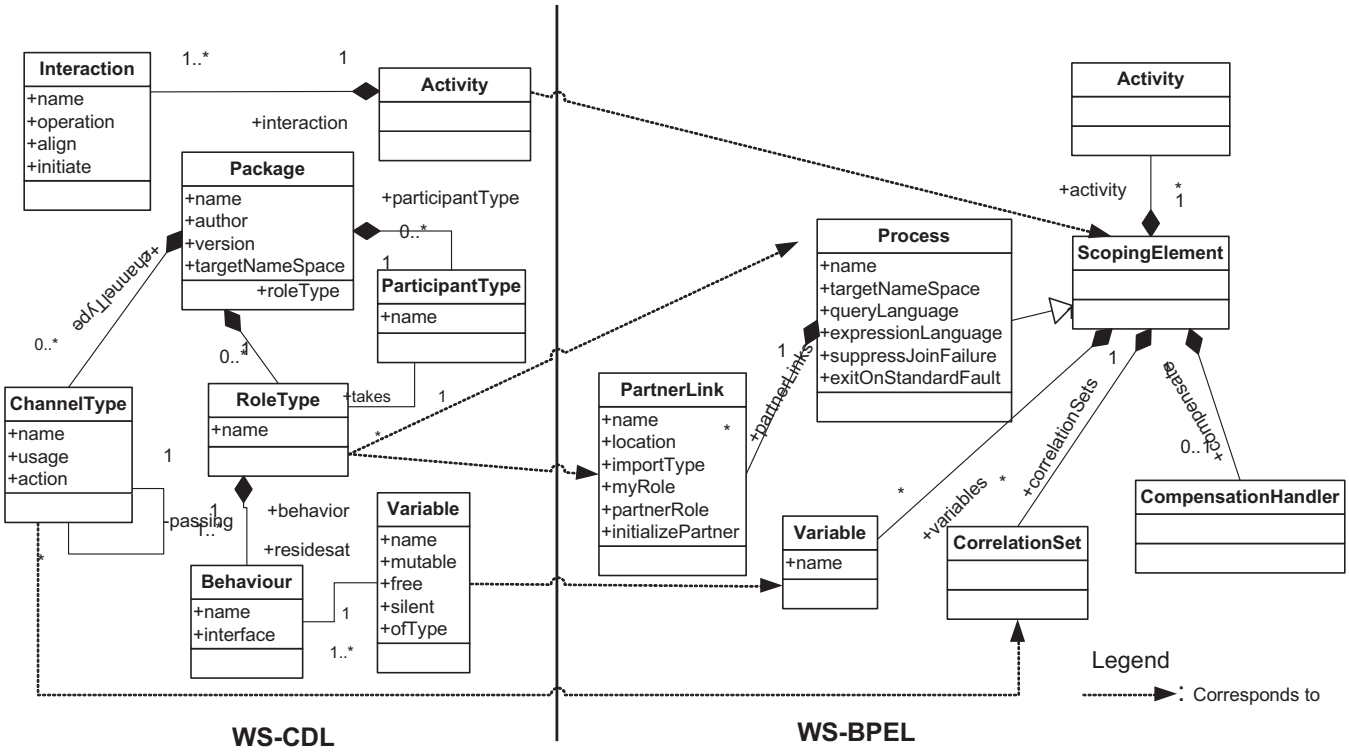

Fig. 5. WS-CDL *RoleType* to WS-BPEL *Process* transformation.

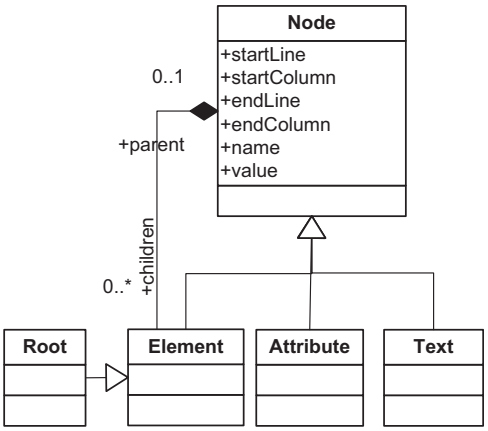

Fig. 6. XML metamodel.

adopted a BTO business model in which it holds a certain part of the individual hardware components in stock and orders missing components if necessary. The customer sends a quote request with details about the required hardware equipment to the manufacturer. The latter sends a quote response back to the customer. As long as the customer and the manufacturer do not agree on the quote, this process is repeated. Upon reaching a mutual agreement, the customer sends a purchase order to the manufacturer. If the manufacturer needs to obtain hardware components to fulfil the purchase order, it sends an appropriate hardware order to the respective supplier. In turn, the supplier sends a hardware order response to the manufacturer. Finally, the manufacturer sends a purchase order response back to the customer.

Initially, the choreography of BTO was modelled using the Pi4soa WS-CDL editor, and subsequently T1, T2 and T3 transformations have been performed. The WS-BPEL skeleton for the manufacturer process was obtained as result, which was validated against the WS-BPEL XML schema. The generated BPEL specification was then imported in the Eclipse BPEL designer to check the behaviour of the orchestrator, which complies with the behaviour of the choreography. Fig. 9 shows the WS-BPEL process of the orchestrator in the Eclipse BPEL designer.

7. Related work

Several approaches [24,25,30] reported in the literature have addressed transformations from WS-CDL to WS-BPEL.

In [28], the transformation from a WS-CDL choreography to a WS-BPEL based orchestration process is presented. The mapping of WS-CDL constructs to WS-BPEL constructs is provided, which is implemented in recursive XSLT as a proof-of-concept to realise the transformation. The transformation is bidirectional, i.e., the XSLT script can generate a WS-BPEL process from a WS-CDL specification, and vice versa. In [27], a WS-CDL to WS-BPEL transformation approach is proposed in which Service Level Agreements (SLAs), which are defined as the annotations of the WS-CDL specification, are also transformed into WS-BPEL policies. The transformation mapping from WS-CDL to WS-BPEL is inspired by [28] and the transformation is implemented using Java.

Weber et al. [34] propose a transformation approach to generate a WS-BPEL process from a WS-CDL specification that

Listing 4
Excerpt of transformation T1.

```
module cdl2xmi;
create OUT : CDL from IN : XML;

rule Root2Package{
 from
  s : XML!Root
 to
  t : CDL!Package(
   name<-'Package',
   children<-Sequence{name, author, tgnsp, version,
    s.informationType, s.relationType,
    s.participantType, s.roleType,
    s.tokenLocator, s.token,
    s.choregraphyPkg, s.channelType}),
...
    }
```

Listing 5
Illustration of XML injection (ant task).

```
<project name="CDL2BPEL" default="transformAll">
<!-- other tasks -->
<!--Inject source model  -->
 <am3.loadModel modelHandler="EMF" name="xmlModel" metamodel="XML" path="/project/po.cdl">
   <injector name="XML"/>
 </am3.loadModel>
<!-- Transform XML model into CDL model -->
 <am3.atl path="/project/cdl2xmi.atl">
  <inModel name="IN" model="xmlModel"/>
  <inModel name="XML" model="XML"/>
  <inModel name="CDL" model="CDL"/>
  <outModel name="OUT" model="cdlModel" metamodel="CDL"/>
 </am3.atl>
<am3.saveModel model="cdlModel" path="/project/cdl.xmi"/>
</project>
```

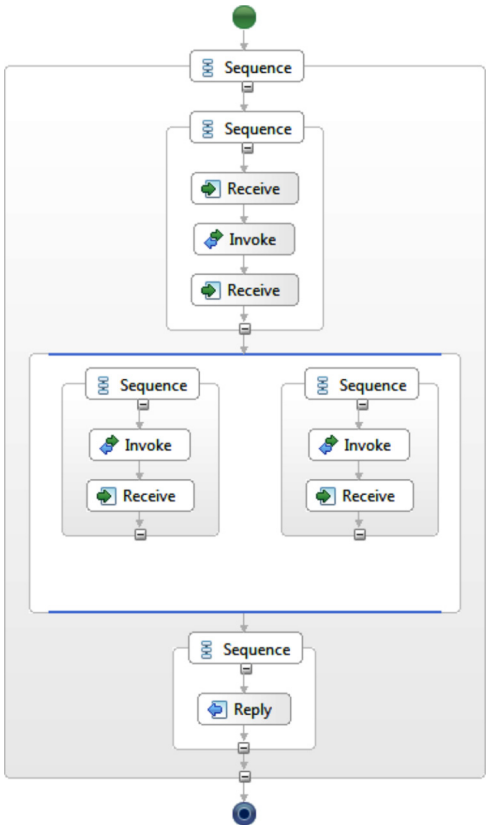

Fig. 7. Manufacturer WS-BPEL process of the purchase order scenario.

introduces the concept of information gap to represent the different levels of details between a choreography and an orchestration. They also indicate that the sum of orchestrations contains more knowledge than the choreography they implement. The proposed approach is implemented in Java. Unlike our MDA based transformation approach, the transformation approaches mentioned above are either based on XSLT or on general purpose programming languages like Java. Such transformations are in general harder to maintain and understand when compared to MDA-based transformations [10].

Recently, Model-Driven Interoperability (MDI) for EI has received significant attention in academia and has resulted in various interoperability frameworks such as the IDEAS Interoperability Framework [35] and the ATHENA Interoperability Framework (AIF) [36]. The IDEAS interoperability framework focuses on structuring interoperability issues into business, knowledge, semantics, and architecture and platform issues. Based on IDEAS, the AIF defines interoperability at the following levels: enterprise/business, process, service, and information/data. The service level of the AIF defines the Platform-Independent Model for Service-Oriented Architecture (PIM4SOA) metamodel and the toolset facilitates the transformation of a PIM4SOA model to a WS-BPEL model, as an alternative to the transformation from WS-CDL to WS-BPEL. The COIN research project aims at studying, developing and prototyping open, self-adaptive, generic ICT solutions for enterprise collaborations and EI [37]. An EI baseline framework has been developed in this project to facilitate EI using MDI and semantic mediation interoperability [38]. Further, the Manufacturing Service Ecosystem (MSEE) research project has also proposed model transformations within MDI to achieve EI [39]. However, the COIN and MSEE projects do not explicitly address transformations from choreographies to orchestrations, so that their work can be considered complementary to ours.

8. Discussion

In the both application scenarios, transformation T3 successfully generated a WS-BPEL skeleton of the orchestrators from their respective WS-CDL specifications. Choreographies have a higher abstraction level than orchestrations, since a choreography does not represent the internal details of the participating services in the collaboration. Therefore, in general it is not possible to generate a complete executable WS-BPEL process, with all the necessary

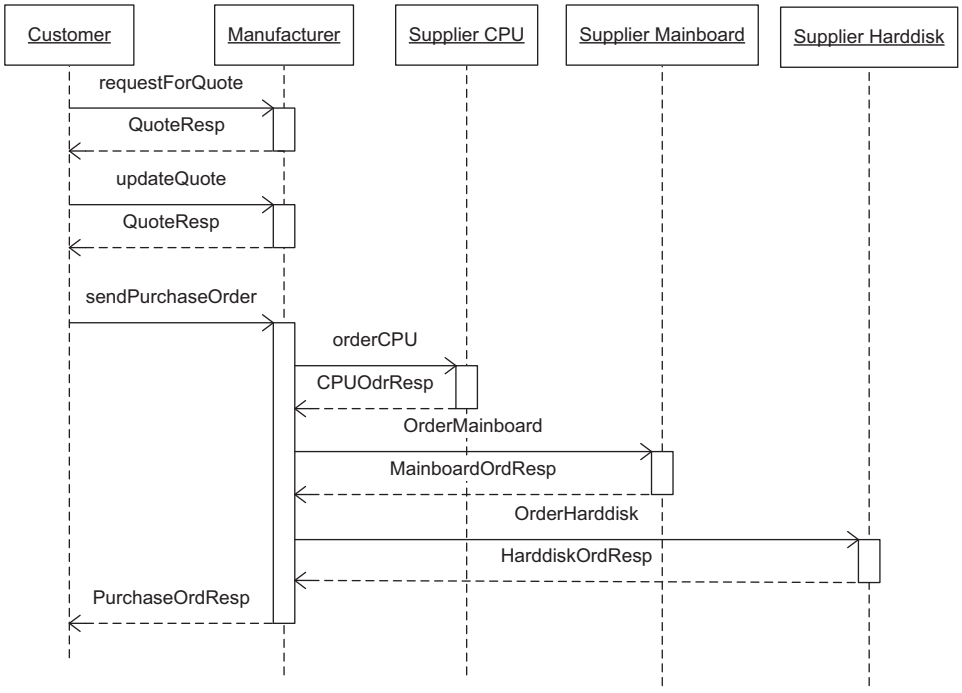

Fig. 8. Sequence diagram of BTO application scenario.

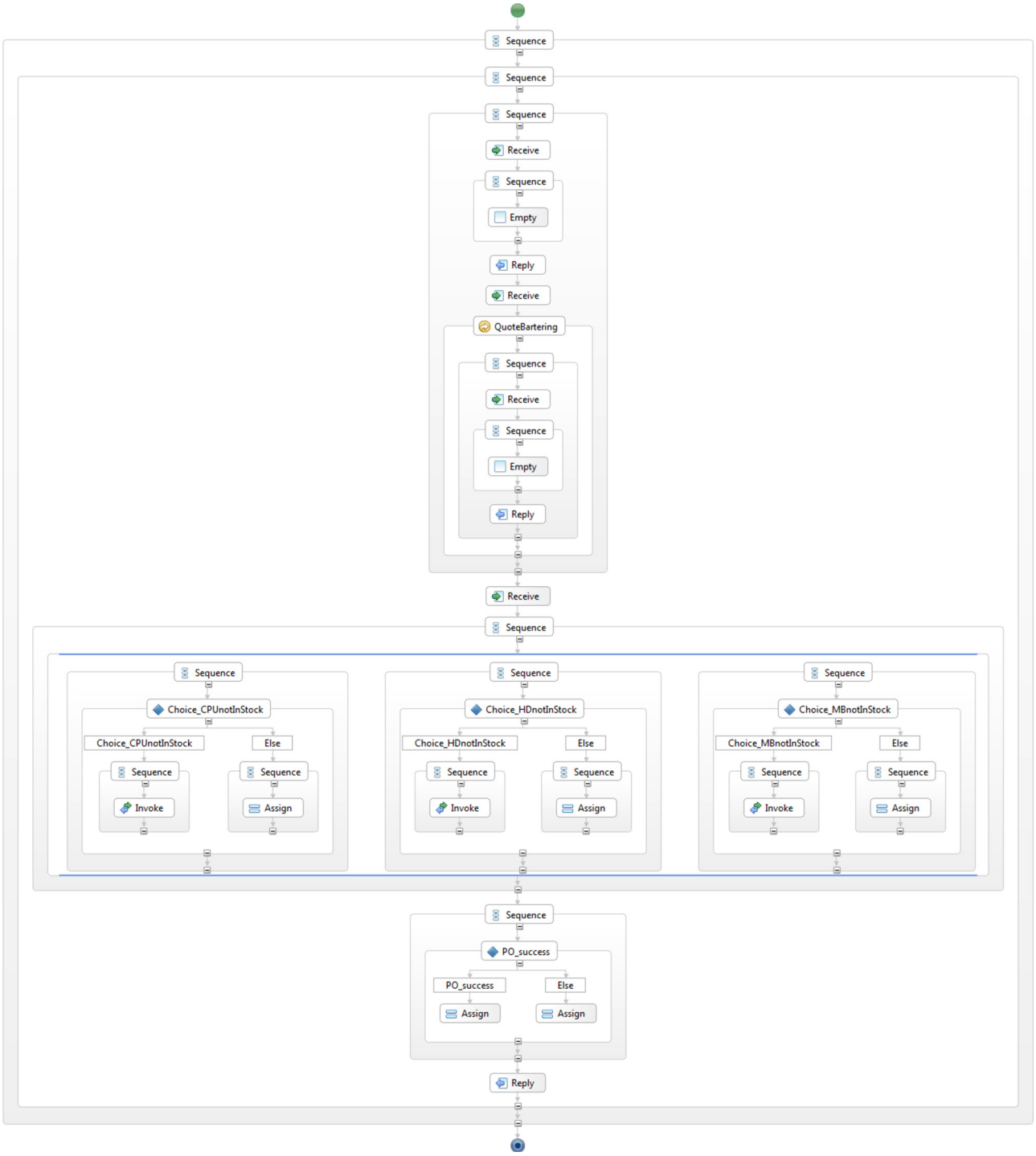

Fig. 9. WS-BPEL process of the manufacturer process of the BTO scenario.

internal details of the participating services, from a choreography. In our approach, the process designer has to manually provide the missing details to the generated the executable WS-BPEL specifications. These missing details are indicated in the transformation mappings depicted in Table 1, i.e., the process designer has to specify the conditions manually. Particularly, a process designer has to add implementation-specific information to the resulting process, which includes the conditions of the WS-BPEL *if-else* constructs and the activities of the related parties in case *silent* activities have been defined in the WS-CDL specification. After these details are added, the WS-BPEL specifications can be validated against the executable schema of the WS-BPEL standard and imported to the Eclipse BPEL designer. For example, Listing 6 depicts an excerpt of the WS-BPEL process skeleton generated for the manufacturer process of the BTO application scenario, in which the process designer has to manually add a condition to replace the empty condition *default* = 0.

The WS-CDL specifications of both application scenarios and the generated WS-BPEL specifications of both the orchestrators are available in [3].

Our choice of generating centralised orchestration in WS-BPEL follows from the popularity and simplicity of this choice. The

Listing 6
Excerpt of the generated WS-BPEL process skeleton.

```
<flow name ="parallel">
<if name ="Choice_CPUnotInStock">
      <condition>"default=0"</condition>
           <sequence>
                <invoke operation ="orderCPU" ../>
           </sequence>
...
...
</flow>
```

generation of decentralised orchestrations introduces various issues, such as distribution and partitioning of responsibilities of each participant in the choreography accordingly in each orchestration process. These issues make it difficult to generalise the transformations, therefore hindering the automation of the transformation process. Furthermore, multiple orchestrations are required to run, possibly in different orchestration engines, so that proper runtime support for error handling in each process is necessary [24].

The approach proposed in this paper uses a metamodelling-based model transformation, in which we represent the abstract syntax of both source and target models in the definition of the WS-CDL and WS-BPEL metamodels, respectively. The relationships between source and target model are presented as transformation mapping, and we implemented this transformation mapping in the ATL transformation language. This approach falls under the "transformation language support" category [10], in which a language is used that provides a set of constructs for explicitly expressing, composing, and applying the transformation. We use metamodelling to express the source and target models at a higher level of abstraction when compared with transformations implemented using a general-purpose programming language, such as Java, C#, or transformations based on scripting languages like XSLT. It has been acknowledged that transformations implemented with general-purpose programming languages tend to be hard to write, comprehend, and maintain while XSLT-based transformations require experience and considerable effort to define and implement [10].

9. Conclusions

In this paper we presented an approach to facilitate enterprise collaboration by addressing EI problems caused by system heterogeneity and business-technology misalignment in the context of continuous change. Our approach uses a largely automated model transformation to obtain a centralised orchestration from a choreography, and, hence, improves the efficiency and accuracy of the enterprise collaboration process. A choreography defines the common behaviour (message exchanges) of the interoperating enterprises, abstracting from the internal details of the participants. In contrast, an orchestration defines the coordination between the interoperating services from the perspective of an orchestrator process that handles this coordination. We used WS-CDL to specify choreographies, WS-BPEL to specify centralised orchestrations and a chain of transformations, implemented in ATL, to transform a given WS-CDL choreography to a WS-BPEL process. The approach has been illustrated with an implementation prototype and its feasibility and general applicability have been demonstrated with two simple yet representative application scenarios. We also discussed the technical challenges of implementing the WS-CDL to WS-BPEL transformation. Our transformation represents an improvement over the currently existing manual transformations. In particular, our transformation facilitates collaboration at business service level by preserving the interoperability requirements captured in choreographies when moving from the choreography to the orchestration level.

With our proposed approach, we demonstrated the suitability of model transformations to (semi-) automate EI. In particular, we claim that our approach offers the following contributions to EI: (i) draws on the important service-orientation trend in business, (ii) links business interoperation requirements (expressed as a choreography of services) to technical interoperability solutions (in terms of a service orchestration process), and, (iii) is suitable for the changing business context for EI, where occasional drastic technology-based changes are replaced by more frequent changes driven by evolving requirements and emerging opportunities for value creation in business networks.

Several directions have been identified for future work. More resilient industrial case studies could be performed to validate the current approach and assess its practical applicability for deployment in industry. The results of these industrial case studies could be compared with similar case studies in which the transformations are manually performed by domain experts. An interesting line of work is to enhance the proposed approach to include in the transformations service level agreements (SLA) defined as annotations in the WS-CDL specification, which should be transformed into WS-BPEL policies. Another possible direction is the automatic generation of WSDL specifications for each participant from the choreography description, which can ease the deployment and testing of the behaviour of the generated WS-BPEL process. A more challenging direction is to investigate the support of some patterns of distributed orchestrations, and to build tools that allow designers to choose the transformation strategy.

Appendix A. Metamodels

See Figs. A.1 and A.2.

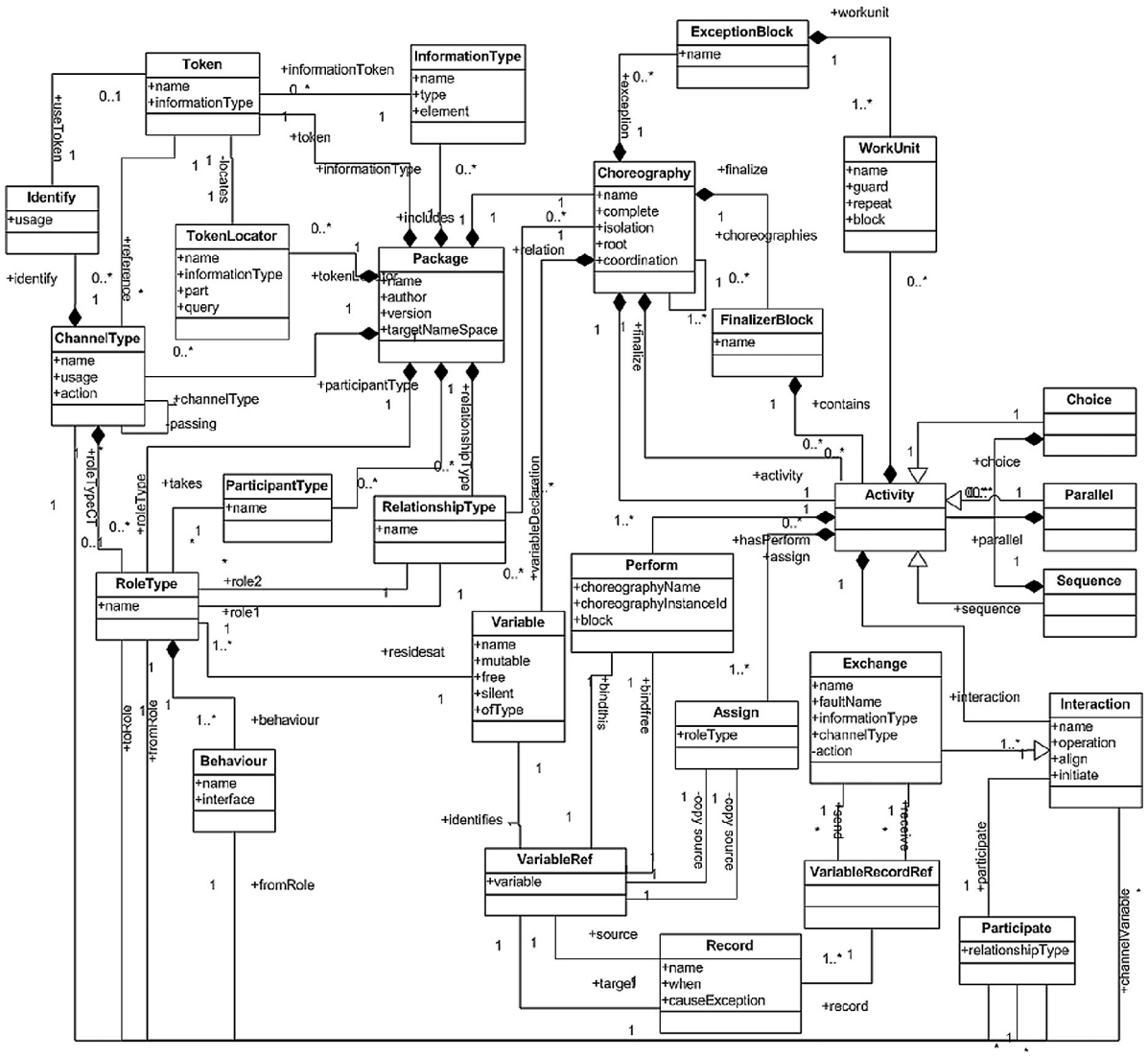

Fig. A.1. WS-CDL metamodel.

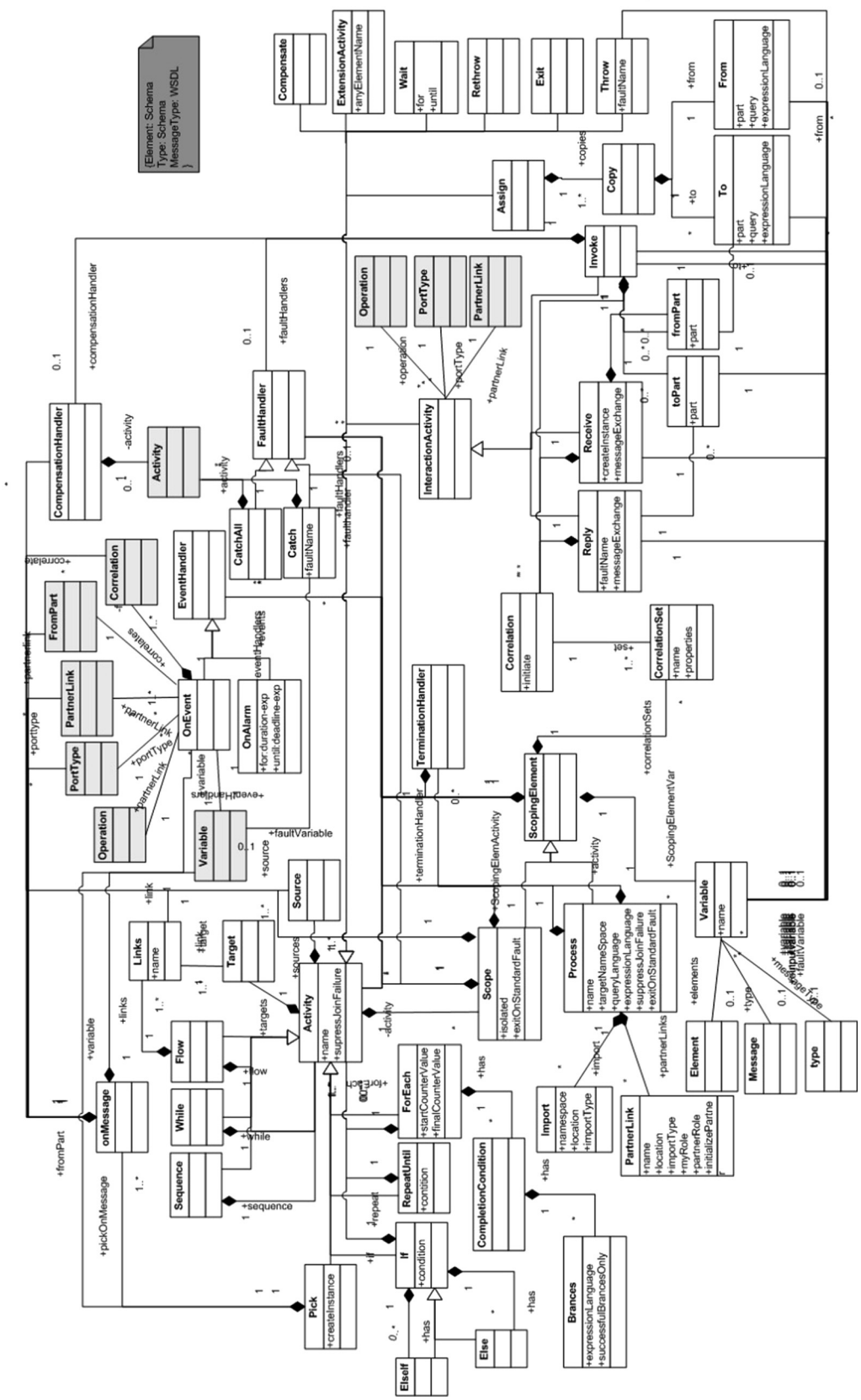

Fig. A.2. WS-BPEL metamodel.

References

[1] D. Konstantas, et al., Interoperability of enterprise software and applications (Introduction), in: Proceedings of the International Conference on Interoperability of Enterprise Software and Applications (I-ESA 2005), Springer Verlag, Genève, 2005.
[2] M. van Sinderen, Challenges and solutions in enterprise computing, Enterprise Information Systems 2 (4) (2008) 341–346.
[3] R. Khadka, Model-Driven Development of Service Compositions: Transformation from Service Choreography to Service Orchestrations, Faculty of Electrical Engineering, Mathematics & Computer Science, University of Twente, 2010 .
[4] R. Jardim-Gonçalves, A. Grilo, A. Steiger-Garcao, Challenging the interoperability between computers in industry with MDA and SOA, Computers in Industry 57 (8/9) (2006) 679–689.
[5] R.M. Pessoa, et al., Enterprise interoperability with SOA: a survey of service composition approaches, in: IWEI'08, IEEE, Munich, Germany, 2008.
[6] M. van Sinderen, M. Spies, Towards model-driven service-oriented enterprise computing, Enterprise Information Systems 3 (3) (2009) 211–217.
[7] M.S. Li, et al., Enterprise Interoperability Research Roadmap in an Enterprise Interoperability Community Document-Work Coordinated by the Enterprise Interoperability Cluster of the Information Society and Media Directorate-General, European Commission, 2006.
[8] OMG, MDA Guide Version 1.0.1, Object Management Group, 2003.
[9] D.C. Schmidt, Model-driven engineering, Computer 39 (2) (2006) 25–31.
[10] S. Sendall, W. Kozaczynski, Model transformation: the heart and soul of model-driven software development, Software, IEEE 20 (5) (2003) 42–45.
[11] J. Touzi, et al., A model-driven approach for collaborative service-oriented architecture design, International Journal of Production Economics 121 (2009) 5–20.
[12] H. Fatemi, M. van Sinderen, R. Wieringa, Value-oriented coordination process modeling, in: 8th International Conference on Business Process Management (BPM2010), Springer, Hoboken, NJ, USA, 2010.
[13] H. Fatemi, M. van Sinderen, R. Wieringa, E 3 value to BPMN model transformation, in: Adaptation and Value Creating Collaborative Networks – 12th IFIP WG5.5 Working Conference on Virtual Enterprises, PRO-VE 2011, Springer, Sao Paulo, Brazil, 2011.
[14] R. Mantovaneli Pessoa, M. van Sinderen, D. Quartel, Towards requirements elicitation in service-oriented business networks using value and goal modelling, in: Proceedings of the 4th International Conference on Software and Data Technologies (ICSOFT 2009), INSTICC, Sofia, Bulgaria, 2009.
[15] G. Decker, et al., Interacting services: from specification to execution, Data and Knowledge Engineering 68 (2009) 946–972.
[16] C. Peltz, Web services orchestration and choreography, Computer 36 (10) (2003) 46–52.
[17] A. Barros, M. Dumas, P. Oaks, Standards for web service choreography and orchestration: status and perspectives, in: C. Bussler, A. Haller (Eds.), BPM Workshops, Springer Berlin/Heidelberg, 2006, pp. 61–74.
[18] A. Arkin, et al., Web Service Choreography Interface (WSCI) 1.0, 2002 http://www.w3.org/TR/wsci/.
[19] N. Kavantzas, et al., Web Service Choreography Description Language (WSCDL) 1.0, in W3C Candidate Recommendation, W3C, 2005.
[20] G. Decker, et al., BPEL4Chor: extending BPEL for modeling choreographies, in: ICWS 2007, IEEE, Salk Lake City, UT, 2007.
[21] T. Andrews, et al., Business Process Execution Language for Web Services v 1.1., 2003 Available from: http://www.ibm.com/developerworks/library/specification/ws-bpel.
[22] A. Alves, et al., Web Services Business Process Execution Language Version 2.0, OASIS, 2007.
[23] W. Binder, I. Constantinescu, B. Faltings, Decentralized orchestration of composite web services, in: ICWS'06, IEEE, 2006.
[24] G.B. Chafle, et al., Decentralized orchestration of composite web services, in: Proceedings of WWW 2004 Conference on Alternate Track Papers and Posters, ACM, NY, USA, (2004), pp. 134–143.
[25] G. Wiederhold, Mediators in the architecture of future information systems, Computer 25 (3) (1992) 38–49.
[26] R. Khadka, et al., Model-driven development of service compositions for enterprise interoperability, in: M. Sinderen, P. Johnson (Eds.), Enterprise Interoperability, Springer, 2011, pp. 177–190.
[27] F. Rosenberg, et al., Integrating quality of service aspects in top-down business process development using WS-CDL and WS-BPEL, in: Proceeding of EDOC 2007, IEEE, MD, USA, 2007.
[28] J. Mendling, M. Hafner, From WS-CDL choreography to BPEL process orchestration, Journal of Enterprise Information Management 21 (5) (2008) 525–542.
[29] F. Jouault, et al., ATL: a model transformation tool, Science of Computer Programming 72 (1-2) (2008) 31–39.
[30] Generating an EMF Model using XML Schema (XSD), 2004 (cited 20.09.12); available from: http://www.eclipse.org/modeling/emf/docs/2.x/tutorials/xlibmod/xlibmod_emf2.0.html..
[31] F. Jouault, et al., ATL: a QVT-like transformation language, in: OOPSLA 2006 Companion, ACM, OR, USA, 2006, pp. 719–720.
[32] BPELDesigner Eclipse BPEL Designer, 2011 Available from: http://eclipse.org/bpel/.
[33] R. Khadka, et al., WSCDL to WSBPEL: a case study of ATL-based transformation, in: MtATL'11, Zurich, Switzerland, 2011 www.CEUR-ws.org.
[34] I. Weber, J. Haller, J.A. Mulle, Automated derivation of executable business processes from choreographies in virtual organisations, International Journal of Business Process Integration and Management 3 (2) (2008) 85–95.
[35] K. Schulz, et al., A Gap Analysis – Required Activities in Research, Technology and Standardisation to close the RTS Gap – Roadmaps and Recommendations on RTS Activities, Deliverables D 3.4, D 3.5, D 3.6, IDEAS Thematic Network-No. IST-2001-37368, 2003.
[36] A. Berre, et al., The ATHENA interoperability framework, in: R.J. Gonçalves, et al. (Eds.), Enterprise Interoperability II, Springer, London, 2007, pp. 569–580.
[37] S. Huber, C. Carrez, H. Suttner, Development of innovative services enhancing interoperability in cross-organizational business processes, in: M. Sinderen, P. Johnson (Eds.), Enterprise Interoperability, Springer, 2011, pp. 75–88.
[38] P. Sitek, et al., The COIN Book: Enterprise Collaboration and Interoperability, Verlagsgruppe Mainz, 2011.
[39] E.M. Silva, C. Agostinho, R. Jardim-Goncalves, Achieving interoperability via models transformation within the MDI, in: M. Zelm, et al. (Eds.), Enterprise Interoperability: I-ESA'12, Wiley, Valencia, 2012.

R. Khadka is a research assistant at the Department of Information and Computer Science at Utrecht University, the Netherlands. His research interests include legacy modernization and model-driven engineering (MDE). He currently focuses in legacy to service-oriented architecture (SOA) migration of financial applications. Mr. Khadka holds a BE degree in Information Technology (2006) from Nepal and an MSc degree in Software Engineering from the University of Twente (2010), the Netherlands.

B. Sapkota is Postdoctoral researcher at the University of Twente, the Netherlands. His research interests include distributed systems architecture, modeling techniques, specification languages, and telematics applications. He has been working on the application of Semantic web and Linked Data technologies to user-centric applications and service personalisation. He contributed to various national and international research projects and co-authored more than 30 research papers in international journals, conferences, workshops and standards working groups. He is a member of programme committee of a number of international conferences and workshops. Dr. B. Sapkota holds an MSc degree in Telematics and a PhD in Computer Science.

L. Ferreira Pires is Associate Professor at the University of Twente, the Netherlands. His research interests include design methodologies for distributed systems, architecture of distributed systems, modelling and specification techniques, middleware platforms and telematics applications. More recently he has been working on the application of Model-Driven Engineering (MDE) and Semantic web technologies to the design of context-aware applications and services, and to (dynamic) service composition. He contributed to various national and international research projects, and co-authored more than 100 research papers in international conferences, workshops and journals. He is a member of the Programme Committee of a number of international conferences and workshops, and he is co-chair of the MODELS-WARD conference. He is currently Education Director of the "Business Information Technology" Education Programme of the University of Twente. Dr. Ferreira Pires holds a BSc degree in Electronics (1983), an MSc degree in Electrical Engineering (1989) and a PhD degree in Electrical Engineering (1994).

M.J. van Sinderen is associate professor at the University of Twente in the Netherlands. There he is also leader of the strategic research orientation "Applied Science of Services" at the Centre for Telematics and Information Technology. His major research interests are design methods and architectures for networked information systems, particularly in the area of next-generation collaborative enterprises and smart consumer applications. He managed and participated in several national and international research projects and coauthored more than 100 research papers. He is a member of the Editorial Board of the Enterprise Information Systems journal, published by Taylor & Francis, and of the Editorial Board of the Service Oriented Computing and Applications journal, published by Springer. He is chair of the Steering Committee of the IEEE EDOC Conference on Enterprise Computing. He is also member of the Managerial Board of IFIP WG5.8 on Enterprise Interoperability. Dr. Van Sinderen holds an MSc degree in Electrical Engineering and a PhD degree in Computer Science.

S. Jansen is an assistant professor at the Department of Information and Computer Science at Utrecht University. He is one of the leading researchers in the domain of software ecosystems and co-founders of the International Conference on Software Business and International Workshop on Software Ecosystems. He is lead editor of the book "Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry" and of several others. Besides his academic endeavors he actively supports new enterprises and sits on the boards of advisors of several start-ups.