

Mapping BPMN 2.0 Choreography to WS-CDL: A Systematic Method



Amir Ebrahimifard¹, Mohammad Javad Amiri², Mostafa Khoramabadi Arani³, Saeed Parsa⁴

¹ Graduate School of Management and Economics

Sharif, University of Technology, Tehran, Iran

² Student, Computer Science Department, University of California

Santa Barbara, Santa Barbara, USA

^{3,4} Software Engineering Group, Iran University of Science and Technology

School of Computer Engineering, Tehran, Iran

ebrahimifard@gsme.sharif.edu, amiri@cs.ucsb.edu, khoram@comp.iust.ac.ir, Iran, parsa@iust.ac.ir

ABSTRACT: Nowadays compositing web services to present capabilities in response to customers' complex requirements has a special importance in service-oriented development. However, service choreography as a form of service composition, still faces many problems. The large gap between business domain and service-oriented domain, especially in service choreography and the lack of systematic method for code generation from the business level models are two significant challenges of this domain. The goal of this paper is to present a stepwise systematic method for mapping an inter-organization business process model to an implementation level code. To this end, choreography business processes are modeled using the interaction view of BPMN 2.0 standard and then, the proposed algorithm maps the BPMN 2.0 model to the WS-CDL code in 16 steps. The resulted code covers all of the required elements of WS-CDL, describes the choreography of services and can be used in the implementation level.

Keywords: Service Composition Choreography BPMN 2.0, WS-CDL

Received: 12 October 2015, Revised 14 November 2015, Accepted 20 November 2015

© 2016 DLINE. All Rights Reserved

1. Introduction

Service-oriented Computing (SOC) has become an important trend in software engineering, exploiting both web services and Service-oriented Architecture (SOA) as fundamental elements in the development of on-demand applications [1]. SOA can be defined as an architectural style for building enterprise solutions essentially including a collection of self-describing interoperable services that are able to communicate with each other. In other words, this architecture enables different organizations to independently implement services that meet their needs, yet can also be combined into higher-level business processes and enterprise solutions[2; 3; 4]. SOA encourages organization to understand, how their information technology infrastructure capabilities can be organized to achieve business goals [5].

Web services are known as the best way of realizing SOA[6] and are the way for effective utilization of services [7]. If implementation of a web service involves the invocation of other web services, it is necessary to combine the functionality of several web services. In this case, we speak of a composite service [8]. Service composition stands as a mechanism

combining multiple services to build up more complex functionalities. This enhances the potential of single services. The aim of Web service composition is to arrange multiple services into workflows supplying complex user needs [9]. Today, with service composition, instead of developing monolithic applications, we build large-scale software applications by composing loosely-coupled services. By doing so, we can reuse and select suitable services as various organizations can provide similar services for the development of different applications [10].

There are two possible ways for service composition: orchestration and choreography which are defined as two aspects of inter- and intra-organization service composition. One of the major challenges in the choreography domain is the lack of a single comprehensive definition for this concept. Every resource presents its own description of choreography; however, most of them are complementary and do not conflict with each other.

Due to the huge amount of web services and the need to supply dynamically varying user goals, it is necessary to perform the composition automatically [9]. Repeatability, reliability, efficiency, testing throughout the development cycle, and versioning are other common benefits of automating development processes [11]. In recent years, many researchers have concentrated on automatic service composition, as it is a remarkable and promising solution to software engineering [12]. Bridging the gap between model and code is one of the most important problems of the service composition process. This problem can be examined in choreography or orchestration domain. In orchestration, there are solutions present for resolving this problem [13; 14], but in the choreography domain there is not any comprehensive solution yet. In other words, there is not a thorough mapping mechanism between business to business(B2B) choreography language and service choreography language. [15] suggests a method for mapping BPMN1.0 to WS-CDL. This method faces two major problems. First, it uses BPMN 1.0 that does not support choreography in interaction approach and uses inter-connection approach that has serious problems. Second, that method does not consider the process as a whole and presents a solution for mapping each part of the process separately and independently, and does not mention a way for integrating these parts, so the resulted code is not applicable.

The main goal of this paper is to present a method for mapping B2B choreography models to the service choreography language.

The rest of the paper is arranged as follows. In Section 2, the definition of choreography, choreography modeling paradigms and choreography modeling languages are discussed. In Section 3, a method for mapping BPMN 2.0 choreography models to the WS-CDL language is proposed. The proposed method is then evaluated through Section 4 and finally, the conclusion of this paper is presented in Section 5.

2. Choreography Definition & Modeling Approaches

This section presents a broad review of the choreography concepts. First, choreography is defined in the domain of service composition. Then, the choreography modeling approaches are described and compared based on different characteristics. Next, the choreography languages are introduced and two languages for modeling choreography in two different levels will be introduced based on these comparisons.

2.1 Choreography Definition

The terms choreography and orchestration describe two aspects of web service composition [16]. The relation between these two concepts is that the interaction between individual behaviors of each involved party, which are defined via the orchestration, results in a collaborative behavior, described by the choreography [17]. W3C defines choreography as the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state [18], but in fact there are numerous different and sometimes incompatible definitions for choreography. So the first challenge in the choreography domain is the lack of a comprehensive definition of the choreography concept.

After studying choreography definitions, the substantial characteristics of this concept are listed as below:

1. Specification of the message-based interaction of participants[19; 20; 21; 22]
2. Corporation of participants to gain a common goal[22; 23; 18]

	Common characteristic IC I		
	Message Sending & Receiving Activities [28; 29]	Separate	One
	Control and Data Flow Dependencies [29]	Role based	Global
	Default Service Communication Type[29]	Async	Sync
	Choreography Modeling[30]	Hard	Easy
	Redundancy in Model[30]	✓	×
	Execution Feasibility[31]	✓	×
	Modeling Anti-Patterns[32]		
Decision-making	Incompatible Branching Behavior	✓	×
	Impossible Data-based Decisions	✓	×
	Inability to Modeling Mixed choices	✓	×
Ordering of interactions	Contradicting Sequence Flow	✓	×
	Incomplete Sequence Flow	✓	×
	Uni-lateral Sequentialization	✓	×
Process creation & termination	Optional Participation	✓	×
	Not-guaranteed Termination	✓	×
	Realization Challenges [32]		
	Unenforceable Sequence	×	✓
	Unrealizable Choreography	✓	✓
	Non-desynchronizable Choice	×	✓

Table 1. Choreography Interaction Modeling(I) vs. Choreography Inter-connection Modeling(IC)

3. Description of interactions from a global perspective[4; 17; 20; 21; 24]
4. Collaboration of independent agents, without a centralized control[21; 18; 24; 25]
5. Description of externally observable peer-to-peer interactions[17; 20; 21; 23]
6. Contract for multi-participant collaboration[23; 18; 26; 27]

2.2 Choreography Modeling Approaches

Inter-connection models and interaction models are two main approaches for choreography modeling. In inter-connection modeling, data flow dependencies are defined within each role, and message sending and reception are in separate activities. In contrast to this, in interaction modeling, atomic message exchanges are the basic building blocks and control and data flow is defined globally [28].

Using interaction modeling results in several advantages for accurate choreography modeling: **(a)** Control flow dependencies are not defined per role, but rather seen from a global perspective. In this way, redundancy in control flow relationships are

avoided and the interactions ordering prevents modeling anti-patterns and deadlock. Having less redundancy in structures leads to faster and easier modeling. **(b)** Global specification of branching structures avoids modeling errors such as decision making anti-patterns, and process instance creation and termination anti-patterns caused by incompatible branching structures [29].

There are some challenges that are caused by interaction modeling which did not exist before, because dependencies no longer belong to individual partners and are defined on a global level. Table 1 compares these two approaches. Based on this comparison, it is obvious that interaction modeling does not suffer from the same problems as inter-connection modeling, so modeling choreography using the interaction modeling approach is more suitable and satisfies all characteristics of a choreography.

2.3 Choreography Modeling Languages

There are many choreography languages which can be compared from different points of view.[33] suggests three classes for categorizing choreography languages: B2B choreographies, services choreographies, and conceptual choreographies.

In this paper, assuming that services have been identified and the main purpose is to realize the choreography of services, systematic method to transform B2B choreography to service choreography is introduced. For this purpose from each domain a representative must be selected. A few standards have been proposed for modeling choreographies, such as the Web Services Choreography Description Language (WS-CDL), a W3C standard candidate proposed in 2004, and, more recently, the OMG Business Process Model and Notation version 2 (BPMN2) [34]. So, BPMN2 as a standard BPM language from B2B choreography languages and WS-CDL as a specification by W3C from service choreography languages are chosen.

3. Mapping Bpmn 2.0 Choreography Model to WS-CDL

In this section a systematic method for mapping BPMN choreography model to WS-CDL code is introduced. The goal of this method is filling the gap between B2B domain and service domain that can help to describe and implement service choreography using a modeling language.

It should be noted that the BPMN model is the base of mapping and due to the differences between approaches of BPMN model as a B2B choreography language and WS-CDL as a service choreography language, supporting all elements of WS-CDL is impossible, nonetheless the proposed method supports all necessary and a large number of optional elements of WS-CDL.

WS-CDL contains two types of elements: static elements and dynamic elements. Static elements are used for declaring roles, participants, relationships, and data types. Determining these elements is necessary for declaring dynamic parts like interaction, sequences and choices. Therefore in following 16 steps, first static elements are extracted which are used during declaration of dynamic parts. Each step is a completion to preceding steps. In every step both BPMN 2.0 and WS-CDL element are shortly introduced and after that a mapping between these elements is proposed.

3.1 Declaring Roles

According to OMG specification “A Choreography defines the sequence of interactions between Participants” and these participants can be PartnerEntity or PartnerRole. In WS-CDL, interactions are occurring between roles which are declared as roleType element. Therefore participants in BPMN 2.0 are mapped to roleType element in WS-CDL.

```
<roleType name="NCName" >
...
</roleType>
```

By checking all BPMN 2.0 choreography tasks, all participants of choreography can be determined. These participants are declared using roleType element in WS-CDL.

3.2 Declaring Behaviors

Behavior element determines a set of operations which the choreography role is required to support using web services. These operations are a subset of service operations which is defined in WSDL files.

Each role can have multiple web services.

```
<roleType name="NCName" >
    <behavior name="NCName" interface="QName"? />+
</roleType>
```

In BPMN 2.0 choreography task the operation which is called is not defined. In other hand, new attributes can be added to elements according to omg specification, Therefore an attribute should be added to choreography tasks for defining the operation which is called in interaction. By grouping the operations according to WSDL files, behavior elements are determined.

3.3 Declaring Participating Organizations

participantType element in WS-CDL groups together those *roleTypes* that must be implemented by a participant which is an organization or a virtual organization. This concept is not modeled using BPMN 2.0 explicitly. Therefore the choreography designer should specify which choreography role is implemented by which organization.

```
<participantType name="NCName" >
    <roleType typeRef="QName" />+
</participantType>
```

3.4 Declaring Relationships

relationshipType element in WS-CDL declares the roles which have interaction. It is a declaration of mutual commitments that must be made for collaborations to be successful. Every interaction in WSCDL uses a *relationshipType* as a part of its declaration.

```
<relationshipType name="NCName" >
    <roleType typeRef="QName" behavior="list of NCName"? />
    <roleType typeRef="QName" behavior="list of NCName"? />
</relationshipType>
```

By checking all BPMN 2.0 choreography tasks, relationships are determined. For every non repeated couple of participants which their interaction is shown using choreography task, a *relationshipType* element is declared.

3.5 Restricting Relationships

In WS-CDL it is possible to restrict a relationship to a subset of operations which the role supports. Therefore after defining relationships, by checking operation attribute of choreography task it is possible to restrict the declared *relationshipType* elements to those operation that is used by this relationship using behavior element in its definition.

3.6 Declaring Channels

A channel is a reference to a participant service that is used during choreography, a token is for extracting data in channels and a *tokenLocator* specifies how to extract the token.

The concept of channels is absent in BPMN 2.0[35] But in WS-CDL it is one of the major part of choreography description. Therefore for being able to do the mapping, choreography designer should specify channels, token and token locator. And for each choreography task it should be defined which channel is used. After that by checking all choreography tasks in BPMN 2, cannels are determined.

3.7 Defining Data Types

Data types are defined using *informationType* element in WS-CDL. It is used for defining type of information exchanged in choreography. In BPMN 2.0 choreography data exchange is not modeled, therefore properties of message data that is

exchanged between roles in interactions should be determined using an attribute in choreography task. This attribute defines data, data type and variables if they are used in message exchange. For variable if the data being exchanged is used later it should be stored in a variable.

```
<informationType name="NCName"
  type="QName" ? | element="QName" ? />
```

Using data types declared in added attribute and data types used in tokens declaration, the set of data types is determined. For every data type an informationType element should be declared.

3.8 Defining variables

In WS-CDL Variables capture information about objects in a choreography.

```
<variableDefinitions>
  <variable name="NCName"
    informationType="QName" ? | channelType="QName" ?
    ...
    roleTypes="list of QName" ? />+
</variableDefinitions>
```

After defining channelType, informationType and roleType elements almost all variables can be defined. Using the attribute added to choreography tasks and channels which is used to connect to each role, big portion of variables can be defined. For every channel that is used there should be a variable declaration by that channelType and the role which have access to it. For every variable determined in attribute of choreography task there should be a variable declaration by informationType part equals to data type associated with that variable and role types having access to it. For each data that is used in decision making a variable should be declared and the value of this variable should be determined in an interaction before that decision making part.

3.9 Declaring Intermediate Events

Event is something that happens during the course of a Process [36]. There are three types of event in BPMN 2.0 which are start event, intermediate event and end event. There are two types of intermediate events in BPMN 2.0:

- **Intermediate Events in Normal Flow**

According to OMG specification none, timer, conditional, link, signal, multiple intermediate event can be used in choreography in normal flow.

None intermediate is just for documentation and should not be used in a choreography diagram that is going to be realized.

Link intermediate event is just for readability and is ignored during mapping process.

Time, condition and signal intermediate events are declared using a silentAction element in WSCDL, because these events are handled internally. silentAction element in WS-CDL is used for nonobservable operation that must be performed to be able to continue the choreography. In WS-CDL there is a timeout element which is defined for determining the deadline of an interaction and cannot be applied here.

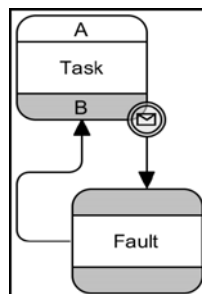


Figure 1. Message intermediate event used in BPMN to show fault

```
<silentAction roleType="QName"? />
```

Multiple intermediate event can only be used as a catch for collection of events in BPMN 2.0 choreography diagram. Therefore multiple intermediate event is one of time, condition and *signal* intermediate events or some of them and it is an internal thing therefore like before it is mapped to a *silentAction* element in WS-CDL.

• Intermediate Events Attached to Activity Boundary

According to OMG specification message, timer, cancel, compensation, conditional and signal intermediate events can be used in choreography diagram when attached to activity boundary. In WS-CDL interaction are atomic and cannot be broken in normal; but when an exception occurs, the interaction breaks and choreography flow changes to exception flow which is declared in exceptionBlock of choreography. When these interactions are used for this purpose the BPMN diagram can be mapped to WS-CDL.

Message intermediate event when used like the Figure 1 can be mapped as a fault message in an interaction. Note that the message type of choreography task comes afterward should be just a request not request-respond.

3.10 Declaring Non-repeatative Interaction

An interaction as the basic building block of a choreography can be resulted when information exchanged between collaborating participants and possibly the synchronization of their observable information changes and the values of the exchanged information[26].

```
<interaction name="NCName"
  channelVariable="QName"
  operation="NCName"
  align="true"|"false"?
  initiate="true"|"false"? >
  <exchange name="NCName"
    faultName="QName"?
    informationType="QName"?|channelType="QName"?
    action="request"|"respond" >
    ...
  </exchange>*
  <timeout... />?
  <record>
    ...
  </record>*
</interaction>
```

After defining roleType, channelType, and variable elements, interaction element can be defined. For every choreography task, there is an interaction element declaration. Like BPMN 2.0 choreography task, WS-CDL interactions used to show observable message exchange between participants. Different part of interaction declaration can be defined as below:

Operation: according to operation that is specified using an attribute of choreography task.

ChannelVariable: according the variable that is defined for the channel used for connecting to the receiver role.

Align: using the method in [29] it is determined if the choreography is desynchronizable, if it's desynchronizable align attribute would be false, for interactions which needs to be synchronized in communication align attribute would be true. So the interaction is executed synchronously.

Initiate: if it is the first choreography task that should be execute first and it is the initiator of choreography, it should be true.

Exchange: its defined using attribute added for specifying data being exchanged during choreography task.

Timeout: it is not supported by BPMN 2.0 Completely. There is a boundary time intermediate event in BPMN 2.0 but it can be used to show that after some time passed if nothing happens an exception occurs and choreography is continued in exception block but it cannot model recording data which is in WS-CDL timeout specification. This part is optional in WS-CDL.

Record: it is not supported by BPMN 2.0 because in BPMN 2.0 storing data is not modeled. This part is optional in WS-CDL.

3.11 Declaring Paths

A path in WS-CDL is defined using sequence element; sequence is an ordering structure in WS-CDL. In BPMN ordering is shown using sequence flow. A path in BPMN 2.0 is a flow that does not include any gateway. Each path of BPMN 2.0 can be declared in WS-CDL using sequence element.

```
<sequence>
...
<interaction>...</interaction>
...
</sequence>
```

3.12 Declaring Repetitive Interactions

A BPMN 2.0 choreography task with loop activity marker means an interaction which is repeated until next interaction happens. In this model the initiator of next choreography task must be as same as immediately preceding choreography task to be realizable; because initiator decides when to finish the loop by not repeating the repeated message and sending a new message.

To be able to map this model to WS-CDL it is needed to use workunit, choice and assign element of WS-CDL. workunit is an element in WS-CDL for creating loops and conditions, choice is an element for exclusive paths and assign element is for assigning new value to variables. It should be noted that the looped interaction is a request-respond message exchange activity. Figure 2 shows the use of choreography task by loop marker in BPMN 2.0

First a boolean variable should be added to variableDefinition part of WS-CDL. If the boolean type is not defined; an informationType should be declared and added to WS-CDL description.

```
<workunit repeat=" cdl:getVariable(CTaskDone,'','')=false()">
  <choice>
    <interaction name="C Task" ></interaction>
    <sequence>
      <interaction name="C Task 2" ></interaction>
      <assign roleType="Bidder">
        <copy name="NCName" >
          <source variable="true()" />
          <target variable="cdl:getVariable(CTaskDone,'','')" />
        </copy>
      </assign>
    </sequence>
  </choice>
</workunit>
```

A BPMN 2.0 diagram with Multi-Instance marker should not be used in choreography diagram which is going to be realized because it is not one interaction and multiple types of messages is sent or received. Therefore instead of using an activity with this marker multiple activities should be used.

Figure 3 shows the parallel multi instance activity and Figure 4 shows the sequential multi instance activity.

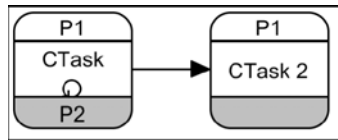


Figure 2. Use of choreography task by loop marker in BPMN 2.0

3.13 Declaring Branches Without Implicit Loop

Gateways are used to control how the Process flows (how Tokens flow) through Sequence Flows as they converge and diverge within a Process. There are five types of gateway in BPMN 2.0 which is used for modeling branches. Table 2 shows these different types.

Parallel gateways are mapped to parallel element of WS-CDL which means covered sequences are executed in parallel.

Inclusive gateways are mapped to parallel and workunit element of WS-CDL. Since the outgoing sequences can be executed in parallel way, parallel element is used and workunit element is used for checking the condition before executing.

Exclusive gateways are mapped to choice and workunit elements of WS-CDL. Choice is used for creating exclusive paths and workunit is used for checking conditions of paths.

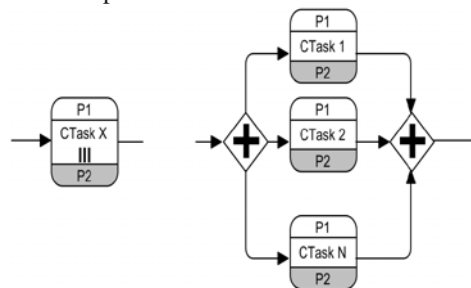


Figure 3. a) a choreography task with parallel multi instance marker. b) the alternative model that shows the same thing as (a)

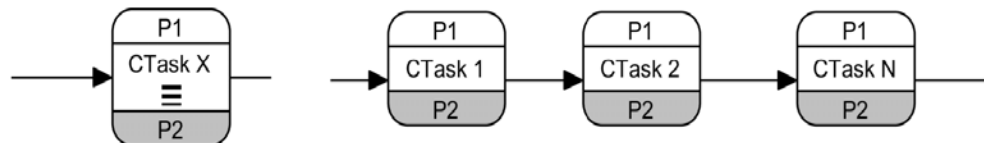
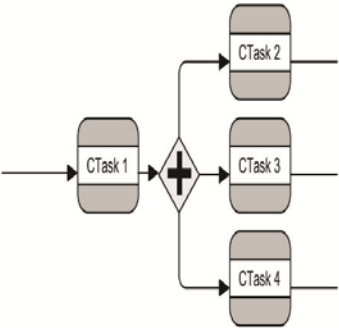


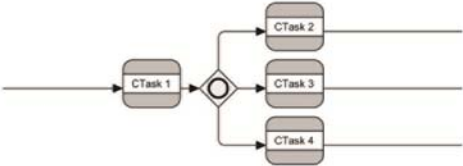
Figure 4. a) a choreography task with sequential multi instance marker. b) The alternative model that shows the same thing as (a)

Gateway	Purpose
parallel gateway	modeling parallel paths
inclusive gateway	modeling data-based conditional inclusive paths
Exclusive gateways	modeling data-based conditional exclusive paths
Event-based gateway	modeling event-based exclusive paths
Complex gateway	modeling complex conditional branches

Table 2. Different types of gateways in BPMN 2.0

Event-based gateways are modeled using choice element of WS-CDL. Choice is exclusive and it chooses the path that its first interaction happens. Complex gateways are used for creating complex condition and it is almost impossible to create a unique way of mapping to WS-CDL. Complex gateways can be modeled using a combination of other gateways. Table 3 shows all different types of BPMN 2.0 gateways and the corresponding WS-CDL code.

	BPMN 2.0 model	WS-CDL code
Parallel Gateway		<pre> ... <interaction name="CTask 1"> ... </interaction> <parallel> <sequence> <interaction name="CTask 2"> ... </interaction> </sequence> <sequence> <interaction name="CTask 3"> ... </interaction> </sequence> <sequence> <interaction name="CTask 4"> ... </interaction> </sequence> </parallel> ... </pre>

Inclusive Gateway		<pre> ... <interaction name="CTask 1"> ... </interaction> <parallel> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 2"> ... </interaction> </sequence> </workunit> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 3"> ... </interaction> </sequence> </workunit> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 4"> ... </interaction> </sequence> </workunit> </parallel> </pre>
-------------------	---	---

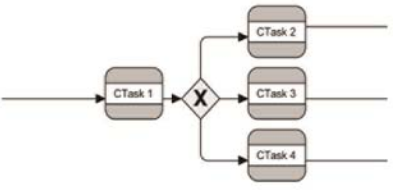
Exclusive Gateway		<pre> ... <interaction name="CTask 1"> ... </interaction> <choice> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 2"> ... </interaction> </sequence> </workunit> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 3"> ... </interaction> </sequence> </workunit> <workunit guard="Condition" block="true" > <sequence> <interaction name="CTask 4"> ... </interaction> </sequence> </workunit> </choice> </pre>
-------------------	---	---

Table 3. Proposed mapping between different types of BPMN 2.0 gateways and WS-CDL

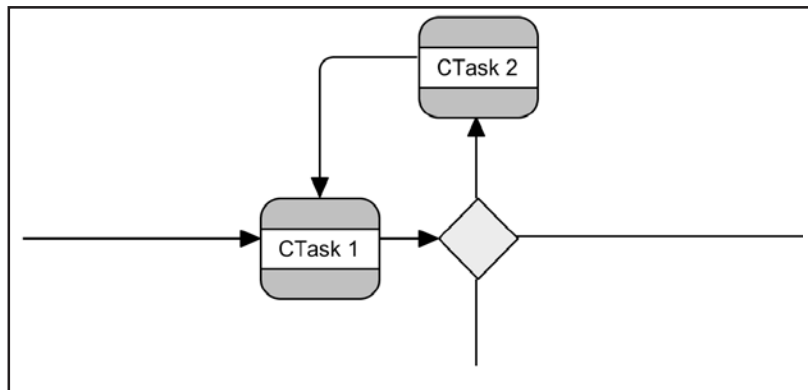
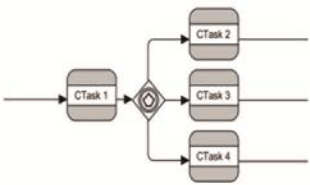


Figure 5. Sequence Flow Looping in BPMN 2.0

3.14 Declaring branches with Sequence Flow Looping

Sequence Flow Looping is created by connecting a sequence flow to an object which has an outgoing sequence flow that leads to a series of other Sequence flows, the last of which is an incoming sequence flow for the original object[36]. In Figure 5, CTask1 and CTask2 create a Sequence flow looping.

In inclusive gateway when there is a loop, a parallel element is used because it inclusive and different paths can be executed in parallel and a workunit for taking care of conditions and loops.

Event-based Gateway	 <pre> ... <interaction name="CTask 1"> ... </interaction> <choice> <sequence> <interaction name="CTask 2"> ... </interaction> </sequence> <sequence> <interaction name="CTask 3"> ... </interaction> </sequence> <sequence> <interaction name="CTask 4"> ... </interaction> </sequence> </choice> </pre>
---------------------	--

In event-based gateway, when there is a loop; a boolean variable needs to be created and added to variableDefinition part of WS-CDL. A workunit element for creating the loop is needed and choice for creating paths of event-based gateway. Workunit element repeat condition became false at end of sequence which is not part of the loop and is one of other outgoing path of gateway.

In exclusive gateways a workunit element with a repeat condition is used for creating loop and after that a choice element and workunit element with repeat attribute set to false for other outgoing paths of gateway. Because of the nature of exclusiveness when one of paths that are not part of loop happens, the loop cannot happen again.

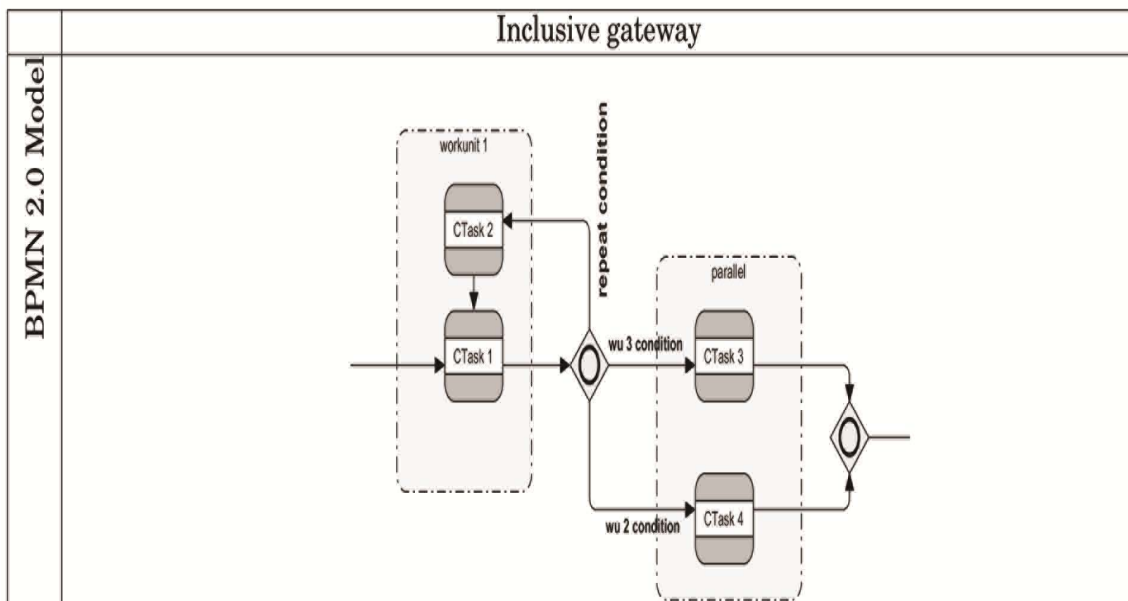


Table 4. Proposed mapping between BPMN 2.0 gateways which involve in a sequence looping and WS-CDL

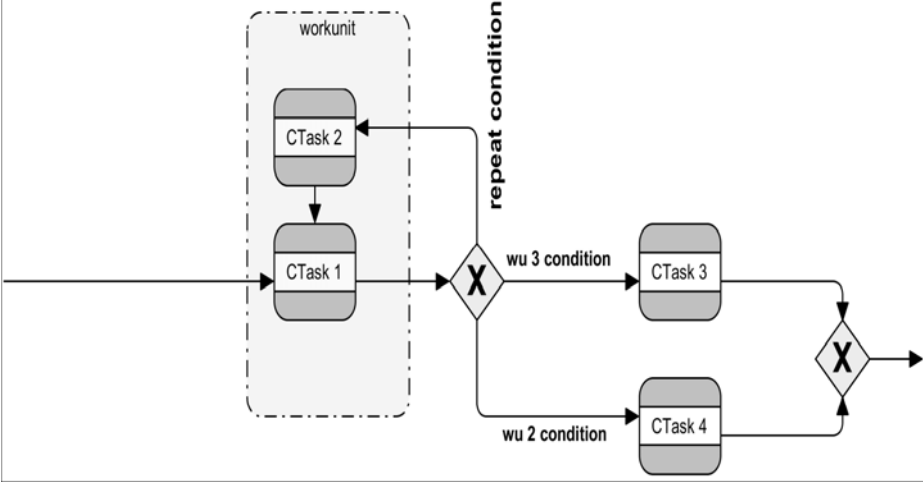
WS-CDL code	<pre> <interaction name="CTask1"> </interaction> <parallel> <workunit name="workunit1" guard="true" repeat="repeat condition" > <sequence> <interaction name="CTask2"> ... </interaction> <interaction name="CTask1"> ... </interaction> </sequence> </workunit> <parallel> <workunit name="workunit3" guard="wu3 condition" repeat="false" block="true"> <interaction name="CTask3"> ... </interaction> </workunit> <workunit name="workunit2" guard="wu2 condition" repeat="false" block="true" > <interaction name="CTask2"> ... </interaction> </workunit> </parallel> -rest of the model- </parallel> </pre>
	Event-base gateway
BPMN 2.0 Model	<pre> graph LR In(()) --> CTask1[CTask 1] CTask1 --> G1{ } G1 --> CTask2[CTask 2] CTask2 --> CTask1 G1 --> CTask4[CTask 4] CTask4 --> G2{ } G2 --> CTask3[CTask 3] CTask3 --> G2 G2 --> Out(()) </pre>

```

<interaction name="CTask1">
...
</interaction>
<workunit name="workUnitName"
    guard="cdl:getVariable("LoopCondition","","")
        = true()" repeat="true()">
<choice>
    <sequence>
        <description type="documentation">
            this sequence will be repeated
        </description>
        <interaction name="CTask2" >
            ...
        </interaction>
        <interaction name="CTask1" >
            ...
        </interaction>
    </sequence>
    <sequence>
        <interaction name="CTask3">
            ...
        </interaction>
        <assign roleType="R">
            <copy name="setBarteringDone">
                <source expression="false()"/>
            <target variable=
                "cdl:getVariable('LoopCondition','','')"/>
            </copy>
        </assign>
    </sequence>
    <sequence>
        <interaction name="CTask4" >
            ....
        </interaction>
        <assign roleType="R">
            <copy name="setBarteringDone">
                <source expression="false()"/>
            <target variable=
                "cdl:getVariable ('LoopCondition','','')"/>
            </copy>
        </assign>
    </sequence>
</choice>
</workunit>

```

Exclusive gateway

<p>Exclusive Gateway</p>	
<p>WS-CDL code</p>	<pre> <interaction name ="CTask1"> ... </interaction> <workunit name="workunit1" guard="true" repeat="repeat condition"> <sequence> <interaction name="CTask2"> ... </interaction> <interaction name="CTask1"> ... </interaction> </sequence> </workunit> <choice> <workunit name="workunit3" guard="wu3 condition" repeat="false" > <interaction name="CTask3"> ... </interaction> </workunit> <workunit name="workunit2" guard="wu2 condition" repeat="false" > <interaction name="CTask2"> ... </interaction> </workunit> </choice> </pre>

3.15 Declaring Choreography and Sub Choreography

Dynamic part of choreography is defined in choreography element of WS-CDL.

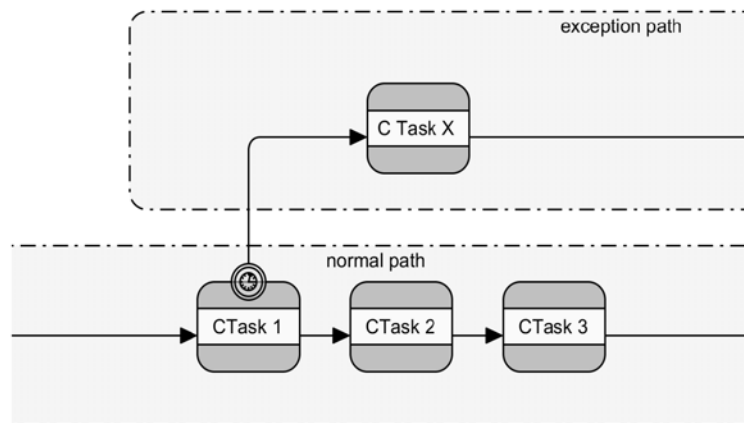


Figure 6. Modeling exception using BPMN 2.0

```
<choreography ... root="true"|"false"? >
  <relationship type="QName" />+
  variableDefinitions?
  Choreography-Notation*
  Activity-Notation
  <exceptionBlock name="NCName" >
    WorkUnit-Notation+
  </exceptionBlock>?
  ...
</choreography>
```

In mapping process, main BPMN 2.0 diagram is mapped to root choreography element in WS-CDL so its root attribute set to true. After that other elements that have been defined in previous parts, are placed within it. For all relationshipType that are used in this choreography one relationship element by type attribute set to the name of used relationshipType is added.

If choreography sub process is used in BPMN 2.0 model; first a choreography definition is added for child diagram and placed in choreography-Notation part of choreography definition or placed aside the root choreography definition and for using it across the WS-CDL perform element is used.

exceptionBlock part of choreography definition in WS-CDL is used for negative flow and choreography failure. BPMN 2.0 does not support this directly, but if intermediate event attached to activity boundary is used and the outgoing path dose not intersect with normal flow of choreography, the outgoing flow of intermediate even t can be declared in exception block. Figure 6 shows modeling exception using BPMN2.0.

```
<choreography name="PurchaseChoreography" >
  ...
  <variableDefinitions>
    <variable name="purchaseOrderAtRetailer"
      informationType="purchaseOrder" roleTypes="tns:Retailer"/>
  </variableDefinitions>
  <interaction name="CTask1" />
  <perform choreographyName="RetailerWarehouseChoreography">
    ...
```

```

</perform>
    <interaction name="CTask2" />
    ...
</choreography>
<choreography name="RetailerWarehouseChoreography">
    <variableDefinitions>
        <variable name="purchaseOrder"
            informationType="purchaseOrder" roleTypes="tns:Retailer"
            free="true"/>
    </variableDefinitions>
    ...
</choreography>

```

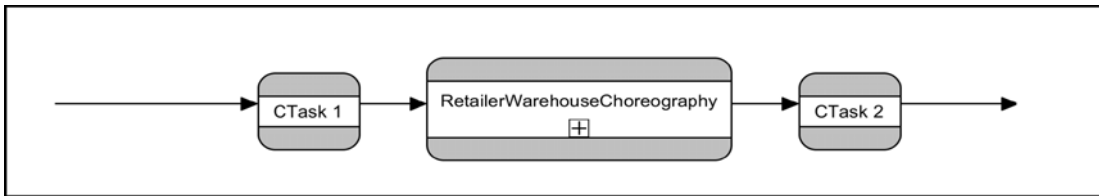


Figure 7. An example of a choreography model with sub choreography

3.16 Defining Overall structure of WS-CDL

For the whole choreography designed, a package element is declared in WS-CDL and *informationType*, *token*, *tokenLocator*, *participantType* and *choreography elements* declaration is placed within it.

```

<package
    ...
        <informationType/>*
        <token/>*
        <tokenLocator/>*
        <roleType/>*
        <relationshipType/>*
        <participantType/>*
        <channelType/>*
        Choreography-Notation*
    </package>

```

As mentioned earlier during this mapping process some attribute are added to BPMN 2.0 choreography task. These are channelVariable for specifying the channel used in interaction, operation for specifying the operation called in interaction and exchange for specifying the data exchanged.

4. Evaluation

To evaluate our mapping, we prepare a 4-level verifying method that can evaluate accuracy of our method. In Birdseye view, the method consists of 4 parts that has been applied on our mapping method.

These 4 parts are the degree of supported patterns and elements, the level of precision with increasing the size of processes, the level of precision with increasing process complexity and the level of precision with increasing ratio of total messages to total participants. Now each of these steps will be described below.

4.1 Degree of Supported Patterns And Elements

This index, measures the number of covered elements and patterns in our method proportional to the total number of existing elements and patterns. Total elements were identified according to [37; ?] and total patterns were identified based

	Items	Number of Items	Supported Items
Elements[37; 36]	Activities	1	1
	Gateways	5	5
	Connectivity Objects	3	2
	Artifacts	3	0
	Swimlanes	2	0
	Events	17	11
Patterns[37]	Basic Controlflow	5	5
	Advanced Synchronisation	4	4
	Structural Patterns	2	2
	Multiple Instances Patterns	4	4
	State-Based Patterns	3	3
	Cancellation Patterns	2	2

Table 5. Elements and Patterns

on the mentioned items in [37]. After identifying total existing patterns and elements, we evaluate our method and measure the coverage level of these total elements and patterns. For these evaluations we did two tasks, first, all existing elements and patterns are categorized in Table 5 and for each category the number of all items and the number of items that are supported with our method are specified.

Second we choose several processes, which belong to various domains, from MIT Process Handbook [38] and count the number of all items in those selected processes and the number of items that we can cover them, finally the coverage ratio is calculated by dividing the number of supported items by all items. The results can be seen in Table 6.

4.2 Level of Precision with Increasing Size of Processes

In the next step, the size of processes are taken into account. Here we consider the number of choreography tasks as the size of process. Again, using MIT Process Handbook and considering the size, several processes are selected and the precision of those processes is measured. To this end precision is defined as follows:

$$Precision = \frac{\text{number of supported patterns} + \text{number of supported elements}}{\text{total number of patterns} + \text{total number of elements}}$$

Figure 8 shows the results.

As demonstrated in Figure 8, when process size grows, the precision of suggested method does not strongly change.

4.3 Level of Precision with Increasing Process Complexity

In this step, the process complexity is measured regarding process complexity. In this context process complexity is defined as the number of gateways. In order to evaluate proposed method several processes with different complexity are selected and the precision is calculated for each one. The results can be seen in Figure 9.

As Figure 9 shows, precision of our method is not heavily changed along with increasing process complexity.

4.4 Level of Precision With Increasing Ratio of Total Messages to Total Participants

Finally, the last index for evaluating the precision of proposed method, is:

$$Precision = \frac{\text{total messages}}{\text{total participants}}$$

this ratio measures the average of messages number which has been transmitted by each participant in a choreography.

Process	Number of Patterns	Number of Elements	Number of Covered Patterns	Number of Covered Elements	Percent of Covered Patterns	Percent of Covered Elements
Replenish inputs	15	74	12	65	80%	88%
Create new market space	24	90	23	87	96%	97%
Optimize the supply chain	31	85	24	74	77%	87%
Manage raw material inventory	10	32	8	30	80%	94%
Recycle and manage product returns	18	55	14	50	78%	91%
Forecast customer demand with suppliers	24	20	20	17	83%	85%
Collect sales data at POS	11	22	11	22	100%	100%
Hire human resources	26	51	20	40	77%	78%
Pay employee	33	69	26	53	79%	77%
Determine vertical strategy	31	74	30	72	97%	97%
Manage risk by outsourcing	34	84	31	78	91%	93%
Manage taxes and duties	40	62	35	55	88%	89%
Determine potential technology	23	61	20	54	87%	89%
Select supplier	19	47	18	42	95%	89%
Buy electronically using Internet	12	32	9	25	75%	78%
Deliver for Internet orders	20	35	20	35	100%	100%
Conduct customer interviews	18	38	18	38	100%	100%
Select human resource automated telephone	15	34	14	30	93%	88%
Benchmark warehouse performance	10	26	8	20	80%	77%

Table 4. Degree of supported patterns and elements in sample processes

Figure 10 demonstrates that ration for several processes.

As can be seen in Figure 10 our method is not sensitive to changes in the ratio of total messages to total participants in choreography processes.

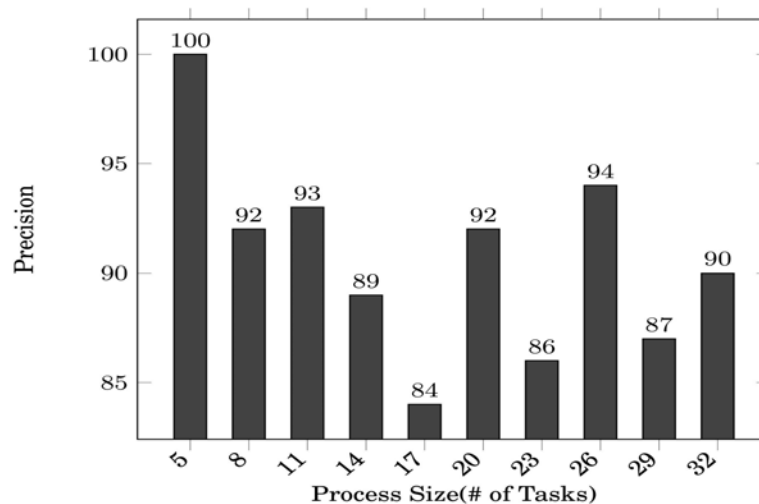


Figure 8. Relation Between Precision and Process Size

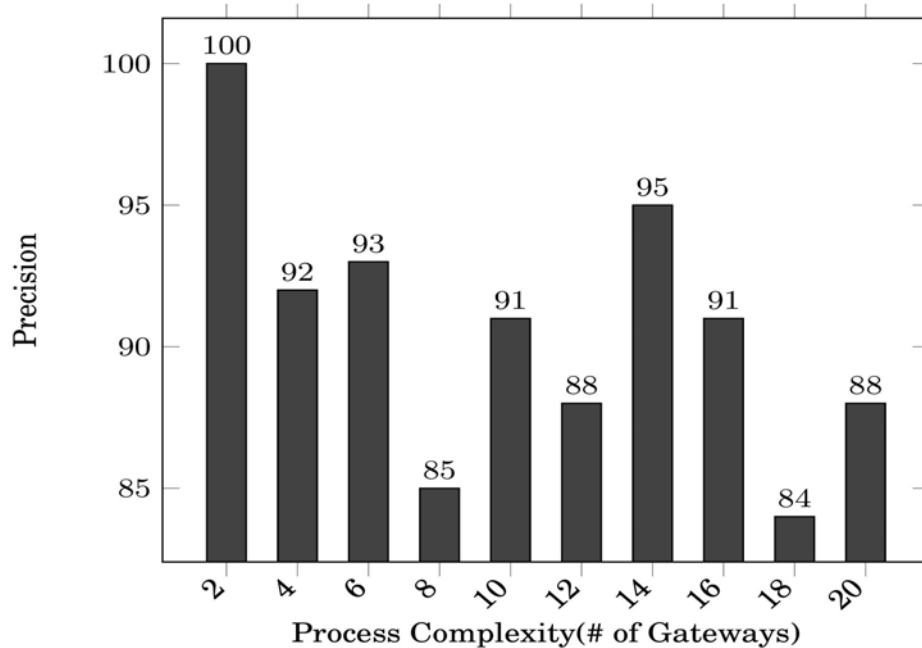


Figure 9. Relation Between Precision and Process Complexity

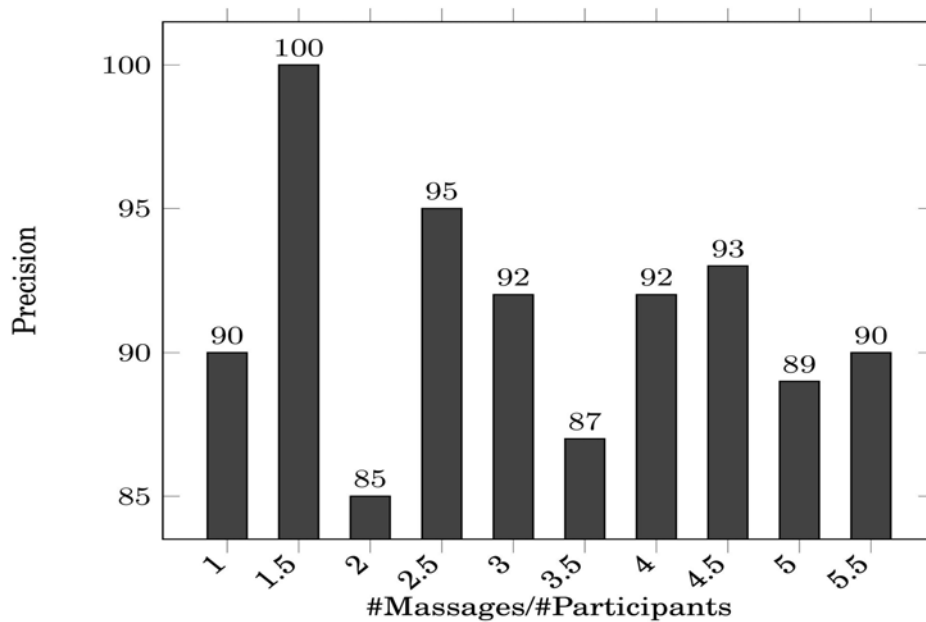


Figure 10. Relation Between Precision and Number of Messages/Participants

4.5 Real Example

Finally, an example of a choreography model for the Purchasing Process is illustrated. In the example process, a buyer corporation sends a registration request. If the type of request is ordinary, the request does not need to be confirmed, but if there is a special request, the registration should be confirmed by the agency. Also, if the type of request is superior, the agency sends the request to the factory and the factory in an iterative process gets the information from the agency and confirms the request. In all three cases, the buyer should start the registration sub-process. After the registration, the buyer can order the request to factory. The factory should confirm this request and while the request is not confirmed, data transfer continues between the factory and the buyer. After request confirmation, the company informs the buyer and the

buyer in contact with the agency, can determine the delivery type.

Then the factory informs the buyer about the cost and the buyer pays this price. If the payment is not successful, the factory informs the buyer and sends a no deliver request to agency. Figure 11 shows the BPMN 2.0 choreography model for this process.

Using the mapping method, the BPMN 2.0 choreography model of purchasing process is mapped to the WS-CDL code.

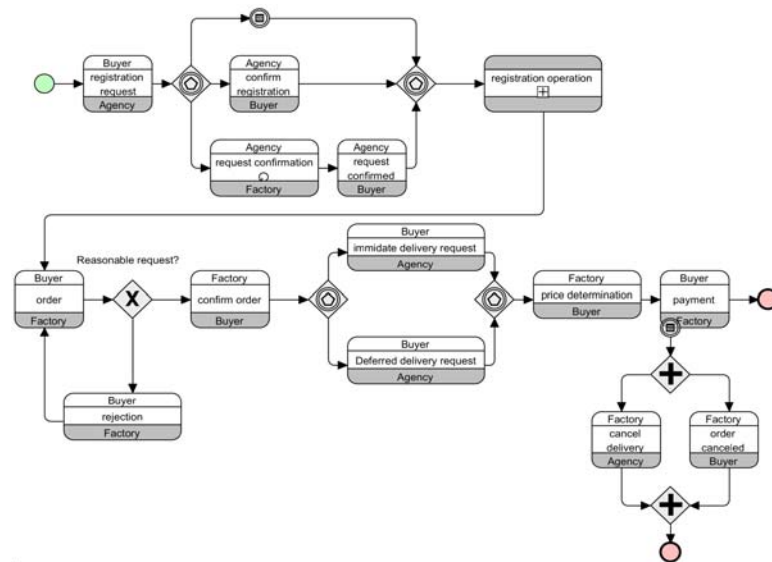


Figure 11. Purchasing process BPMN 2.0 choreography model

5. Conclusion

In this paper a method for mapping a choreography model in business process domain to descriptive choreography language in implementation domain has been proposed. According to various choreography definitions, first of all a comprehensive definition of choreography is presented. For choreography modeling in business domain, due to lower drawbacks and modeling anti-patterns interaction approach was used. Among modeling languages, BPMN 2.0 and among languages, with the aim of service choreography describing, WS-CDL is selected. Afterward 16 steps method for automated mapping BPMN 2.0 choreography diagram to WS-CDL is presented. Providing a systematic way for mapping has led us to overcome differences between current viewpoint of B2B and service choreography and difficulty of interaction between these two fields. Considering all the required structures and presented mapping method caused that final code could be implementable.

References

- [1] Ma, S. P., Fanjiang, Y. Y., Kuo, J. Y. (2014). Dynamic Service Composition Using Core Service Identification, *Journal of Information Science and Engineering*, 30 (4), 957-972.
- [2] Zdun, Uwe, Carsten Hentrich, and Wil MP Van Der Aalst. (2006). A survey of patterns for service-oriented architectures, *International Journal Of Internet Protocol Technology*, 1.3, 132-143.
- [3] Rosen, Michael, et al. (2008). *Applied SOA: service-oriented architecture and design strategies*, John Wiley and Sons.
- [4] Kovac, Damjan, and Denis Trcek. (2011). A Survey of Web services Orchestration and Choreography with Formal Models, Available on <http://www.softec.si/pdf/kovac-damjan.survey.pdf>, Last Visited: 21 Jan 2015.
- [5] Bhuyan, P., Ray, A., Mohapatra, D. P. (2015). A Service-oriented Architecture (SOA) Framework Component for Verification of Choreography. In *Computational Intelligence in Data Mining-Springer India*, 3, 25-35.
- [6] Weerawarana., Sanjiva., Francisco Curbera., Frank Leymann., Tony Storey., Donald F. Ferguson. (2005). *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice

Hall PTR.

- [7] Ding, Z., Liu, J., Wang, J., Wang, F. (2014). An executable service composition code automatic creation tool based on Petri net model. *Computing and Informatics*, 32 (5), 968-986.
- [8] Dustdar., Schahram., Wolfgang Schreiner. (2005). A survey on web services composition, *International Journal of Web and Grid Services*, 1.1, 1-30.
- [9] Bartalos., Peter., Mria Bielikov. (2012). Automatic dynamic web service composition: A survey and problem formalization, *Computing and Informatics*, 30.4, 793-827.
- [10] Truong, Hong-Linh, Schahram Dustdar. (2009). A survey on context-aware web service systems, *International Journal of Web Information Systems*, 5.1, 5-31.
- [11] Tim Kitchens. (2006). Automating Software Development Processes, Published January 12, 2006, Available on: http://www.developerdotstar.com/mag/articles/automate_software_process.html, Last Visited: 6 Jan 2015.
- [12] Syu, Yang, et al. (2012). A survey on automated service composition methods and related techniques, *Ninth IEEE International Conference on Services Computing (SCC)*.
- [13] Milanovic, Nikola, Mirosław Malek. (2004). Current solutions for web service composition, *IEEE Internet Computing*, 8.6, 51-59.
- [14] Schaffner, Jan, Harald Meyer, Cafer Tosun. (2007). A semi-automated orchestration tool for service-based business processes, In Service-Oriented Computing ICSOC 2006, *Springer Berlin Heidelberg*, 50-61.
- [15] Madiesh, Mostafa, Guido Wirtz. (2008). WS-CDL Creator: Modeling WS-CDL using BPMN, The 2008 International Conference on Semantic Web and Web Services (SWWS08), CSREA Press.
- [16] Peltz, Chris. (2003). Web services orchestration and choreography, *Computer*, 36.10, 46-52.
- [17] Autili, Marco, Davide Di Ruscio, Paola Inverardi, James Lockerbie, Massimo Tivoli. (2011). A development process for requirements based service choreography, In IEEE Workshop on Requirements Engineering for Systems, *Services and Systems-of-Systems (RESS)*, 59-62, (2011).
- [18] W3 Glossary, Available on: <http://www.w3.org/Glossary>, Last Visited: 10 Jan 2015.
- [19] Bauer, Bernhard, and Jrg P. Miller, (2004). Mda applied: From sequence diagrams to web service choreography, In Web Engineering, *Springer Berlin Heidelberg*, 132-136, (2004).
- [20] Barros, Alistair, Marlon Dumas, Phillipa Oaks. (2006). Standards for web service choreography and orchestration: Status and perspectives, In Business process management workshops, Figure 10 demonstrates that ration for several processes.
- [21] Laube, Annett, Patrick Winkler. (2010). Generation of choreography skeletons from web service definitions, In Service Computation 2010, *The Second International Conferences on Advanced Service Computing*, 1-6.
- [22] Mellat, Azadeh, Naser Nematbakhsh, Ahmad Farahi, Farhad Mardukhi. (2011). Suitability of UML State Machine For Modeling Choreography of Services, *International Journal of Web and Semantic Technology* 4-2.
- [23] Barker, Adam, Paolo Besana, David Robertson, Jon B. (2009). Weissman, The benefits of service choreography for data-intensive computing, In Proceedings of the 7th ACM international workshop on Challenges of large applications in distributed environments, 1-10.
- [24] Wieczorek, Sebastian, Andreas Roth, Alin Stefanescu, Anis Charfi, (2008). Precise steps for choreography modeling for SOA validation and verification, In *IEEE International Symposium on Service-Oriented System Engineering*, SOSE'08, 148-153.
- [25] Cottenier, Thomas, and Tzilla Elrad. (2005). Executable Choreography Processes with Aspect-Sensitive Services, Computer Science Department, *Illinois Institute of Technology*, (2005).
- [26] Kavantzaz, Nickolas, David Burdett, Gregory Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto, Web services choreography description language version 1.0., *W3C candidate recommendation* 9, (2005).
- [27] Pahl, Claus, Yaoling Zhu. (2006). A semantical framework for the orchestration and choreography of web services, *Electronic Notes in Theoretical Computer Science*, 151-2, 3-18, (2006).

- [28] Pfitzner, Kerstin, Gero Decker, Oliver Kopp, Frank Leymann. (2007). Web service choreography configurations for BPMN, In Service-Oriented Computing Workshops, ICSOC 2007, *Springer Berlin Heidelberg*, 401-412, (2007).
- [29] Decker, Gero, Alistair Barros, Frank Michael Kraft, and Niels Lohmann, Non-desynchronizable service choreographies, In Service-Oriented Computing Workshops, ICSOC 2008, *Springer Berlin Heidelberg*, 331-346, (2008).
- [30] Decker, Gero, and Alistair Barros, Interaction modeling using BPMN, In Business Process Management Workshops, *Springer Berlin Heidelberg*, 208-219, (2008).
- [31] Kopp, Oliver., Frank Leymann. (2008). Choreography Design Using WS-BPEL, *IEEE Data Eng. Bull.*, 31-3, 31-34.
- [32] Decker, Gero, Mathias Weske. (2011). Interaction-centric modeling of process choreographies, *Information Systems* 36-2, 292- 312.
- [33] Schnberger, Andreas. (2011). Do we need a refined choreography notion?, *Services und ihre Komposition*.
- [34] Besson, F., Moura, P., Kon, F., Milojicic, D. (2015).Bringing Test-Driven Development to web service choreographies. *Journal of Systems and Software*, 99, 135-154.
- [35] Cortes-Cornax, Mario, et al. (2011). Evaluating choreographies in bpmn 2.0 using an extended quality framework, *Business Process Model and Notation*, *Springer Berlin Heidelberg*, 103-117.
- [36] OMG: OMG, Business Process Model, Notation (BPMN) 2.01, Object Management Group: Needham, MA 2494, 34, (2013).
- [37] Wohed, P., Dumas, M., Ter Hofstede, A. H., Russell, N. (2005). Pattern-based Analysis of BPMN-An extensive evaluation of the Control-flow, the Data and the Resource Perspectives.
- [38] The MIT Process Handbook Project, Available at: <http://ccs.mit.edu/ph/>