Bridging the Gap between Business Processes and Service Composition through Service Choreographies

Mario Cortès Cornax, Sophie Dupuy-Chessa, Dominique Rieu

▶ **To cite this version:**

Mario Cortès Cornax, Sophie Dupuy-Chessa, Dominique Rieu. Bridging the Gap between Business Processes and Service Composition through Service Choreographies. IFIP WG8.1 Working conference on Method Engineering (ME'2011), Apr 2011, Paris, France. pp.190-203, 10.1007/978-3-642-19997-4_18 . hal-00953442

HAL Id: hal-00953442

https://inria.hal.science/hal-00953442v1

Submitted on 17 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Bridging the Gap between Business Processes and Service Composition through Service Choreographies

Mario Cortes Cornax, Sophie Dupuy-Chessa, and Dominique Rieu

University of Grenoble, CNRS, LIG
SIGMA Team, BP 72, 38402 Saint Martin d'Heres, Cedex France
{Mario.Cortes-Cornax,Sophie.Dupuy,Dominique.Rieu}@imag.fr
http://sigma.imag.fr/

Abstract. Inter-organizational business processes implementations using service composition approaches are being more and more used . We want to reduce the semantic gap that exists between both worlds (business processes and services) through service choreographies, a composition approach that we think it is semantically close to multi-party business processes. We rely on modeling techniques as abstraction layers and view separation to achieve our goal. Our start point is a web service choreography meta-model presented in three abstraction layers where each layer is divided in a structural and a behavioral view. The meta-model can be used in a top-down or a bottom-up approach to make a progressive transition between the business process and the service world.

Keywords: Choreography, Business Processes, Modeling Techniques

1 Introduction

Today, organizations are moving towards inter-organizational business processes. Therefore, they depend on other organizations. To model their business processes, analysts and designers use graphical languages that allow an intuitive and easy reading such as Business Process Management Notation (BPMN) [11]. Modular solutions based on service composition [16] are increasingly found to implement business processes. Service composition languages are mainly based on XML and they do not have a graphical standard notation. We observe a semantic gap between the way of describing business processes and the way of implementing them with service composition, what can create ambiguities and unexpected results as described in [14].

A big effort has been done to bring BPMN closer to web services. In [1] Wil M.P. van der Aalst et al. survey several papers issue of the interest to bridge the gap between these two worlds. BPMN and BPEL [10] alignment is exposed in the standard BPMN. However, we observe that mapping efforts are centered in orchestrations i.e. in two by two relationships between the different external entities of a single process. As business process complexity increases and depends

on other organizations, this pair relationships remains insufficient to manage the complexity when several organizations share common goals. This constraint can negatively influence a global understanding of the process limiting its potential capacity of optimization [14]. A global viewpoint besides a set of individual viewpoints is required to better understand, build, survey and optimize the global process.

In this context, an interesting concept is the *web service choreography* [6]. Web service choreography is a service composition approach focused on message exchanges between different services from a global viewpoint. It can be understood as a multi-party communication protocol where there is no central coordinator. This composition approach seems to be close to inter-organizational business processes. We will therefore seek to better understand service choreographies to bridge the gap between the two worlds that are business processes and service composition when working in multi-party scenarios.

We rely on modeling techniques to propose a meta-model that aims at defining the semantics of choreographies. The meta-model reduces ambiguity and clarifies the main elements of a language. Our approach is based on the concepts of *views* and *abstraction levels*. We remark the necessity of several abstraction layers to provide adapted viewpoints to the different actors that contribute to set up an inter-organizational business process that relies on SOA. Providing these abstraction levels and defining progressive transitions between them is the way to achieve the approach between the global process design and the executable processes. In [18] authors argue the necessity of three abstraction layers when vertical model alignment is targeted.

The rest of paper is organized as follows. Section 2 describes the meta-model approach and compare our proposal with related works. The meta-model overview based on a scenario and a brief description of each layer are presented in Section 3. Section 4 presents a discussion that summarizes our work and future perspectives.

2 Overview of the Approach

Our meta-model construction focuses on three main axis illustrated with arrows in Fig. 1. It starts in a deep analysis of the Web Service Choreography Description Language (WS-CDL) [17] that brings us the knowledge about choreography to build a design meta-model. WS-CDL has been criticized in [3] by Barros et al. as it does not have a clear semantic model and it presents a difficult alignment with BPEL[10]. The Web Services Choreography Working Group [19] stopped the development of this language in July 2009 but it is still a reference as choreography language. There has been other language proposals, mostly developed in research projects. In [8] Decker et al. introduce BPEL4Chor [7] and survey some other choreography language proposals as Let's Dance [20] or iBPMN [5], an extension of BPMN adapted to choreographies. But this concept is still far to be standardized and adopted by the industry. It seems that all proposals are converging to the new choreography representation presented by the OMG in

it's new BPMN version 2.0 [12], but the future of choreography still remains uncertain.

Our design meta-model is based on the one presented by Barros et al. in [3] but enriched by dividing it in different packages. The analysis meta-model is an abstraction of the design meta-model where the syntax dependencies of the specific language were removed. The domain meta-model is an abstraction of the latter one where the fundamental choreography concepts are represented. One of our next objectives is to present in the lowest abstraction layer a language meta-model that is closer than WS-CDL to executable process like BPEL4Chor. Choreography language proposals like BPEL4Chor or the Multi Agent Protocol (MAP) language presented in [2] are well aligned to final executable processes but they lack of a more abstraction viewpoint to reach the business level. We find the necessity to present the choreography concept in three traditional abstraction layers (domain, analysis and design) to bring closer both worlds.

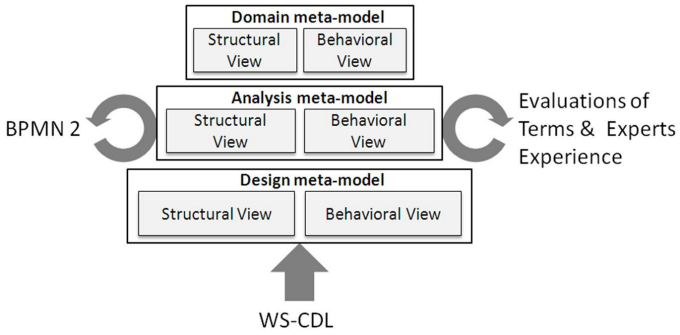

Fig. 1. The construction approach of the meta-model

We observe in Fig 1 that settling the analysis meta-model has been our main target until today. Several iterations at this level were done. An empiric evaluation was performed where business and technical experts examined the suitability of the terms as well as relationships between them through a scenario. We also surveyed some ideas from the choreography diagrams of BPMN's new version [12]. As result, we obtained a simple but representative implementation-independent choreography meta-model. As it is defined in [6] an *implementation-independent level* is where "fundamental decisions about interactions are made", avoiding concrete message formats or security issues. In future works we envisaged to validate in a more formal way our approach against the *Service Interaction Patterns* [4] which are the better known benchmark when working in multi-party collaborative environments. We would like to introduce pattern concepts in our meta-model.

Fig. 1 also shows a division in two different points of view of each layer: the structural and the behavioral views. This separation makes the models easier to

read and more understandable for both designers and developers. The structural view connects all fundamental components displaying them in a static way. The behavioral view defines the reactions of the different elements to the actions of the others. When analyzing WS-CDL, we identified the elements corresponding either to one view or the other. We represented this separation from the first layer (the design meta-model) and we maintained it, in the consecutive abstractions. In this paper we focus on the meta-model and the modeling techniques.

3 The Choreography Meta-Model

This section presents the choreography meta-model layers. We use a scenario describing a computer purchase by a customer and manufacturer's delivery protocol. We will therefore present our meta-model following the scenario taken from [12].

3.1 Scenario

John is a *customer* that orders a custom computer to an Internet's *manufacturer* called ComputerSeller.com. ComputerSeller.com is part of a computer manufacturer's network which they have established a protocol to deliver a purchase order in an efficient and speedy way:

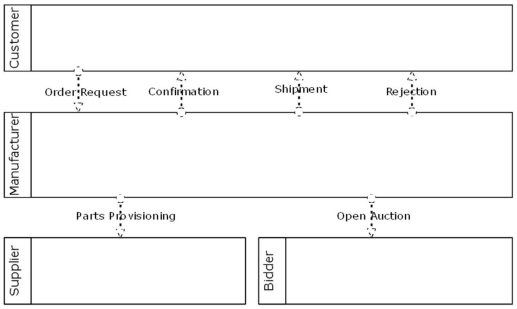

Fig. 2. Scenario interactions represented through a BPMN's collaboration diagram

- If the manufacturer can provide the order, it sends a confirmation and then performs the delivery.
- If the manufacturer can not respond to this command, it rejects the order, explaining the denial reasons.
- If a manufacturer can satisfy the order but it does not have all the pieces available, it should contact a *supplier* so that it provides the missing pieces. The supplier could then procure the missing pieces.

– After contacting the supplier, if the manufacturer has all the parts available to provide the customer's order, it sends a confirmation to the customer and then it delivers the order.
– The manufacturer could not be in possession of all the parts necessary to deliver after contacting the supplier. In this case, it must open an auction with a *bidder* to obtain the missing pieces.
– If the manufacturer has finally obtained all the parts needed, the same actions as before are done: it sends to the customer a confirmation, and then delivers the order.
– If still missing pieces it should reject the order.

Fig. 2 illustrates in BPMN's collaboration diagram the scenario where the public interactions (with no sequencing order) between the actors are presented.

3.2 The Domain Meta-Model

Fig. 3 shows the domain meta-model where the fundamental elements of a choreography are presented.

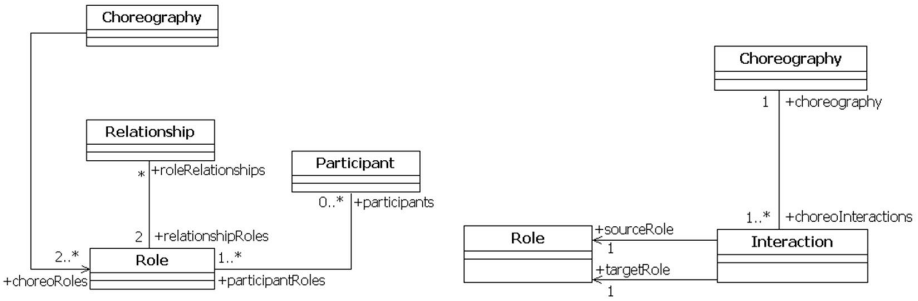

Fig. 3. The domain Meta-Model

From a structural point of view, let us consider the communication protocol (*Choreography*) that has a set of roles (*Role*) linked in two by two relationships (*Relationship*). The relationship represents the existence of a previous knowledge between both roles. A role is an abstract entity (e.g. manufacturer and supplier) played by a participant (*Participant*) that is the concrete entity (e.g. "John" or "ComputerSeller.com"). A participant may play multiple roles in a choreography and a role can be played by several participants (at design time).

From a behavioral point of view, a choreography is defined as a set of interactions (*Interaction)* between two roles (e.g an interaction can be the request of some missing pieces from the manufacturer to the supplier). In an interaction there is a role transmitter (*sourceRole*), in this case the manufacturer, which is the one that starts the interaction and a role receiver (*targetRole*), in this case the supplier.

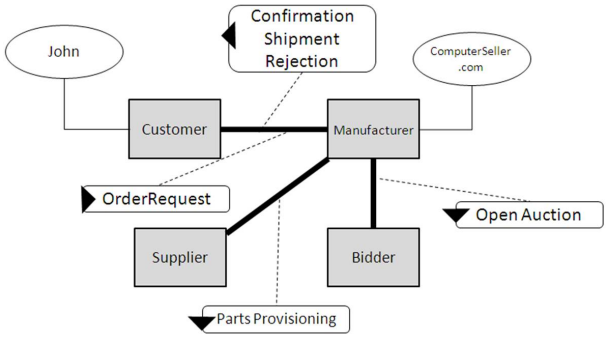

Fig. 4. Structural view representing the scenario + Interactions

The modeled scenario illustrating the domain meta-model concepts is presented in Fig. 4. It represents the structural view but we add interactions. Arrows indicate the "sense" of the interactions i.e the source role and the target role within the interaction. This example is only an illustrative representation that helps us to understand the meta-model. We could also consider the possibility of separating both views in different models but due to the simplicity of the example we decided to illustrate all concepts in the same representation. Graphical notation is an issue of future work and this is not discussed in this article

We retain from this layer that the first elements to be identified in a choreography are the roles and optionally the participants playing that roles. Then, a set the interactions between roles (behavioral) that implies defining their relationships in a static way.

3.3 The Analysis Meta-Model

The analysis meta-model presented in Fig. 5 is the layer in which are represented the elements that must appear in every choreography implementation. In particular, we introduce the control flow mechanisms and the messages exchanged between the choreography participants via the service operations.

Analysis meta-model is explained through the same example but extended. A participant that plays a role must provide the services (*Service*) defined for that role to respect the choreography. Note that we consider a service as a logical entity i.e the way to access the set of defined operations (*Operation*). Each operation defines a request message (*request*) and optionally, a response message (*response*) and error messages (*errorMessages*).

In the behavioral view, we introduce some new elements. As a choreography is a set of ordered interactions between roles, we need mechanisms to compose and order them. We introduce the activity class (*Activity*) as a generalization of control flow activities (*ControlFlowActivity*), and interactions (*Interaction*). A *ControlFlowActivity* can be choice (*Choice*), parallel (*Parallel*) and sequence (*Sequence*) activities. When an operation is invoked in an interaction, it has to be

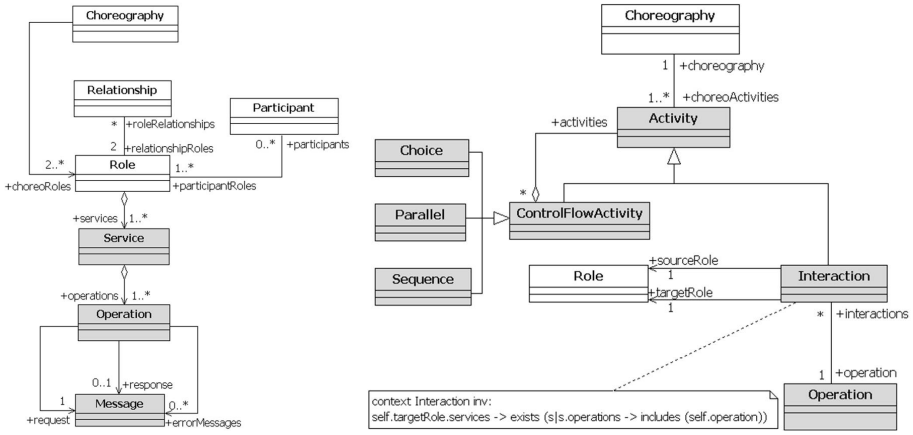

Fig. 5. The Analysis Meta-Model

an operation defined within the *targetRole*'s services. For example, the customer could invoke an operation "manageOrderRequest" provided by the manufacturer in the service "CustomerToManufacturerService". A request message defined in this operation could be for example "requestOrderMessage" and a response message "AcknowledgmentMessage".

The domain meta-model and the analysis meta-model respectively defined in Fig. 3 and Fig. 5 are close. The domain meta-model is included in the analysis meta-model (classes in white) where some domain classes are refined to go into details in the analysis meta-model as the *Role* or the *Interaction*.

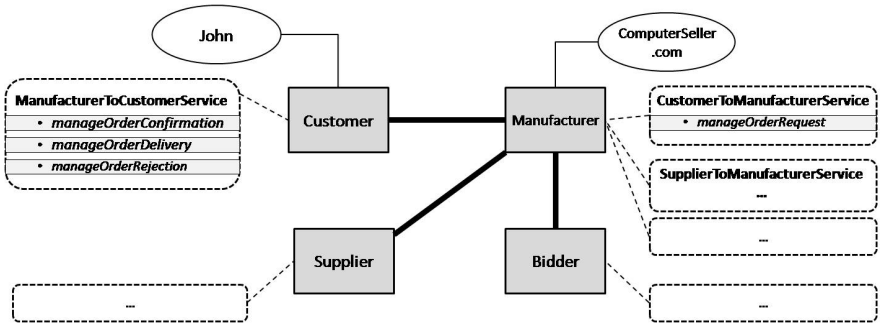

Fig. 6. Structural view of the scenario - Analysis Model

Fig. 6 and Fig. 7 partially model the scenario presented in section 3.1. Fig. 6 represents the structural view where the services provided by each role might be defined. Operations are also defined but messages are not specified as we want to stay simple. Our behavioral model in Fig. 7 is inspired in UML's sequence

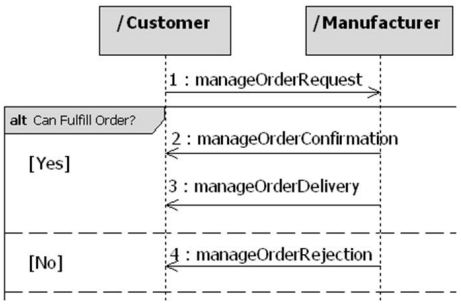

Fig. 7. Behavioral view of a portion of the scenario - Analysis Model

diagrams. Interactions (arrows) between roles (rectangles), operation calls (text above the arrows) as well as sequencing (life line and numbers) or choice (alt) can be represented. This example illustrates in an easy manner our analysis meta-model concepts.

We observe that the analysis meta-model can help users to understand the communication between roles, the interactions sequencing, and the operations required for each role depending on the implemented service.

3.4 The Design Meta-Model

Our design meta-model is based on WS-CDL syntax. Fig. 8 shows the defined packages and it's dependencies. These packages are the starting point of our bottom-up approach. In general, the main package classes have a *corresponding meta-class* in the higher abstraction levels as for example *Interaction*, *Role* or *Choreography*. The complete design meta-model is presented in the Appendix.

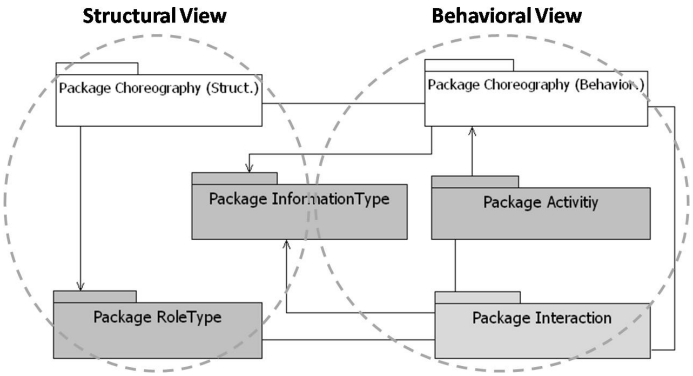

Fig. 8. Design meta-model package overview and dependencies

Code 1.1 and Code 1.2 represent simple WS-CDL's pieces of code that correspond to the scenario. Code 1.1 is what we consider the static definition of WSC-DL's choreographies where roles, relationships, participants and message types are defined. Code 1.2 illustrate the behavioral definition of the choreography where sequencing of interactions and variable declarations are depicted. This behavioral section is identified by "< *choreography* >" label in WS-CDL syntax. More detailed WS-CDL's examples are found in [17].

These examples show that a WS-CDL skeleton could be generated from the analysis model. Regarding the static section, roleTypes (lines 1-9) , participantTypes (lines 11-16) and relationshipTypes (lines 19-22) could be completed. In the behavioral section, interaction sequencing and role's communication could also be completed almost entirely. However, a very simple example is shown. To fully complete a WS-CDL file, we might need the design meta-model. For example, we might complete the message's type exchanged between roles (lines 25-26) and the channelTypes (lines 28-36) to define the way of accessing roles. In the behavioral section, we might also define variable's declaration and treatment (lines 2-7) or the XPath queries to reference variables within the XML file (lines 16-17).

```
1 <roleType name="Costumer">
2     <behavior name="CostumerService" interface="ManufacturerToCostumerService">
3     </behavior>
4 </roleType>
5 <roleType name="Manufacturer">
6     <behavior name="ManufacturerService" interface="CostumerToManufacturerService">
7     </behavior>
8     ...
9 </roleType>
10
11 <participantType name="John">
12     <roleType typeRef="Costumer"/>
13 </participantType>
14 <participantType name="ComputerSeller.com">
15     <roleType typeRef="Manufacturer"/>
16 </participantType>
17 ...
18
19 <relationshipType name="CostumerToManufacturerRel">
20     <roleType typeRef="Costumer"/>
21     <roleType typeRef="Manufacturer"/>
22 </relationshipType>
23 ...
24
25 <informationType name="OrderRequestType" type="OrderRequestMsg">
26 </informationType>
27
28 <channelType name="CostumerToManufacturerChannel">
29     <roleType typeRef="Manufacturer"/>
30     <reference>
31         <token name="tns:URI"/>
32     </reference>
33     <identity type="primary">
34         <token name="tns:id"/>
35     </identity>
36 </channelType>
```

Code 1.1. Example of WS-CDL's code (structural)

```
1 <choreography name="computerPurchaseChoreography">
2 <variableDefinitions>
3    <variable name="orderRequest" informationType="tns:OrderRequestType"
4              roleTypes="tns:BuyerRole tns:SellerRole">
5    </variable>
6  ...
7 </variableDefinitions>
8
9 <sequence>
10   <interaction name="OrderRequest" operation="manageOrderRequest"
```

```
11                channelVariable="tns:Buyer2SellerC">
12   <participate relationshipType="CostumerToManufacturerRel"
13                fromRoleTypeRef="Costumer" toRoleTypeRef="Manufacturer"/>
14
15      <exchange name="OrderRequest" informationType="OrderRequestType">
16         <send variable="cdl:getVariable()"/>
17         <receive variable="cdl:getVariable()"/>
18      </exchange>
19
20      <exchange name="ErrorExchange" informationType="tns:ErrorConfirmationType">
21         <send variable="cdl:getVariable()"/>
22         <receive variable="cdl:getVariable()"/>
23      </exchange>
24    </interaction>
25    <choice>
26       <sequence>
27          <interaction name="Rejection" operation="manageRejection"
28                       channelVariable="">
29       ...
30          </interaction>
31       </sequence>
32       <sequence>
33          <interaction name="Confirmation" operation="manageConfirmation"
34                       channelVariable="">
35             ...
36          </interaction>
37          <interaction name="Shipment" operation="manageShipment"
38                       channelVariable="">
39             ...
40          </interaction>
41       </sequence>
42    </choice>
43 </sequence>
44 </choreography>
```

Code 1.2. Example of WS-CDL's code (behavioral)

We can imagine the huge quantity of code that should be managed when defining complex choreographies by regarding this simple (and not completed) pieces of code. Abstraction is needed to manage choreography complexity in a progressive way. To achieve the goal of approaching business world and service composition world, abstraction layers and different viewpoints should be a main concern to make this concept understandable and exploitable for all the actors participating in an inter-organizational business process set up.

4 Conclusion and Discussion

In this paper we have presented a service choreography meta-model based on modeling techniques as abstraction layers and views separation. Fig. 9 shows a global overview of our approach and future work. A bottom-up approach helps to understand and validate a choreography. A top-down approach helps to implement the final executable processes for each party avoiding ambiguities. We locate the domain and analysis layers into business processes and design layer into the service composition world. We represent the separation as a wavy area as the separation is fuzzy.

By transformation rules, we could move from one layer to another. Therefore, transitions between the abstraction layers manage in a gradually way the gap between business processes and service composition implementations. Formalization of this transformations are envisaged for future works so the passage between layers could be done the more automatically as possible.

We want to extend our approach creating a graphical notation corresponding to the meta-model. As in the Pi4SOA tool [13], we think that the two-views separation is an important aspect to manage the complexity of a service composition

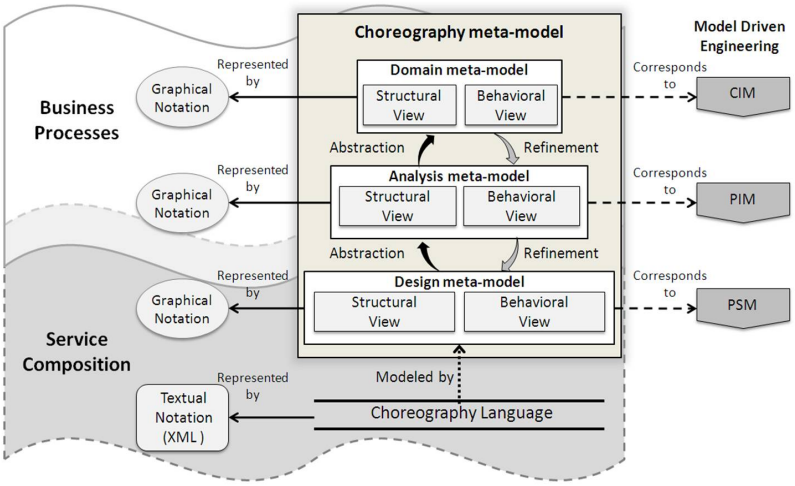

Fig. 9. MDE correspondences and future graphical notation work

model. More generally, the formalization of the three abstraction layers and the views split will help us in respecting the *Principle of Complexity Management*, the *Principle of Cognitive Integration* and the *Principle of Cognitive Fit* defined by Moody for the physics of graphical notation [9]. This part of our work could be clearly compared to the OMG's BPMN 2 [12] work, which incorporates a new graphical notation for choreographies. It means that choreography could be an important concept for business process community.

We also observe in Fig. 9 an obvious correspondence with the paradigm of Model Driven Engineering (MDE) [15]: our domain meta-model layer is equivalent to Computation Independent Model (CIM), the analysis meta-model layer correspond to a Platform Independent Model (PIM), and the design meta-model layer based on WS-CDL is an example of Platform Specific Model (PSM) in MDE. Between the different layers, refinement and abstraction relationships will be implemented by MDE transformations. So this meta-model approach is our first step to help designers and developers in bridging the gap between business process and service composition implementations.

5 Acknowledgments

We would like to thank Gabriel Pedraza, German Vega, Agnes Front and Aurelien Faravelon for their helpful comments. We also thank the international scholarship program FARO Global for their support.

References

1. Van der Aalst, W., Benatallah, B., Casati, F., Curbera, F., Verbeek, E.: Business process management: Where business processes and web services meet. Data and Knowledge Engineering 61(1), 1–5 (2007)
2. Barker, A., Walton, C., Robertson, D.: Choreographing Web Services. IEEE Transactions on Services Computing 2(2), 152–166 (2009)
3. Barros, A., Dumas, M., Oaks, P.: A critical overview of the web services choreography description language. BPTrends Newsletter 3 (2005)
4. Barros, A., Dumas, M., Ter Hofstede, A.: Service interaction patterns: Towards a reference framework for service-based business process interconnection. Faculty of IT, Queensland University of Technology FIT-TR-2005-02 (2005)
5. Decker, G., Barros, A.: Interaction modeling using BPMN. In: Proceedings of the 2007 international conference on Business process management. pp. 208–219. Springer-Verlag (2007)
6. Decker, G., Kopp, O., Barros, A.: An introduction to service choreographies. Information Technology 50(2), 122–127 (2008)
7. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for modeling choreographies. In: Web Services, 2007. ICWS 2007. IEEE International Conference on. pp. 296–303. IEEE (2007)
8. Decker, G., Kopp, O., Leymann, F., Weske, M.: Interacting services: from specification to execution. Data & Knowledge Engineering 68(10), 946–972 (2009)
9. Moody, D.: The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering pp. 756–779 (2009)
10. OASIS: Web services business process execution language v2.0. "`http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`" (2007)
11. OMG: Business process management notation (v1.2). "`http://www.omg.org/spec/BPMN/1.2/`" (2009)
12. OMG: Business process management notation (beta 2). "`http://www.omg.org/cgi-bin/doc?dtc/10-06-04`" (2010)
13. Ross-Talbot, S., Brown, G., Honda, K., Yoshida, N., Carbone, M.: Pi4soa technologies fundation. "`http://sourceforge.net/apps/trac/pi4soa/wiki`"
14. Ross-Talbot, S., Brown, G., Honda, K., Yoshida, N., Carbone, M.: Soa best practices: Building an soa using process governance (2009)
15. Soley, R., et al.: Model driven architecture. OMG white paper 308, 308 (2000)
16. Srivastava, B., Koehler, J.: Web service composition-current solutions and open problems. In: ICAPS 2003 Workshop on Planning for Web Services. vol. 35. Citeseer (2003)
17. W3C: Web services choreography description language version 1.0 - w3c candidate recommendation. "`http://www.w3.org/TR/ws-cdl-10/`" (2005)
18. Weidlich, M., Barros, A., Mendling, J., Weske, M.: Vertical Alignment of Process Models–How Can We Get There? Enterprise, Business-Process and Information Systems Modeling pp. 71–84 (2009)
19. WSC: Web services choreography working group. "`http://www.w3.org/2002/ws/chor/`" (2002)
20. Zaha, J., Barros, A., Dumas, M., ter Hofstede, A.: Lets dance: A language for service behavior modeling. On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE pp. 145–162 (2006)

6 Appendix

The UML class diagrams in Figs. 10 and 11 provide a high-level overview of WS-CDLs meta-model. Fig 10 describes the structural view while 11 describe the behavioral view. These meta-models are based upon the WS-CDL meta-model in [3]. They represent our today's design meta-model.

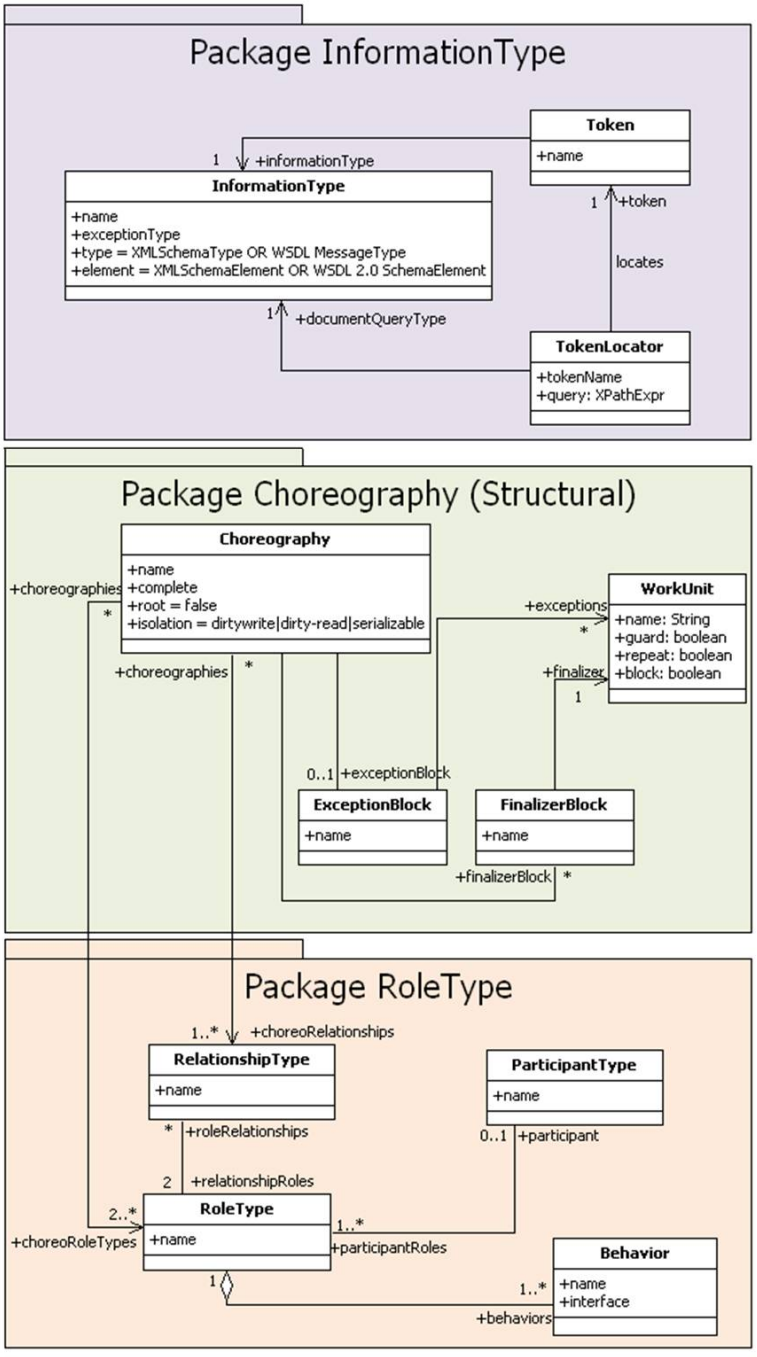

Fig. 10. WS-CDL design meta-model - structural view

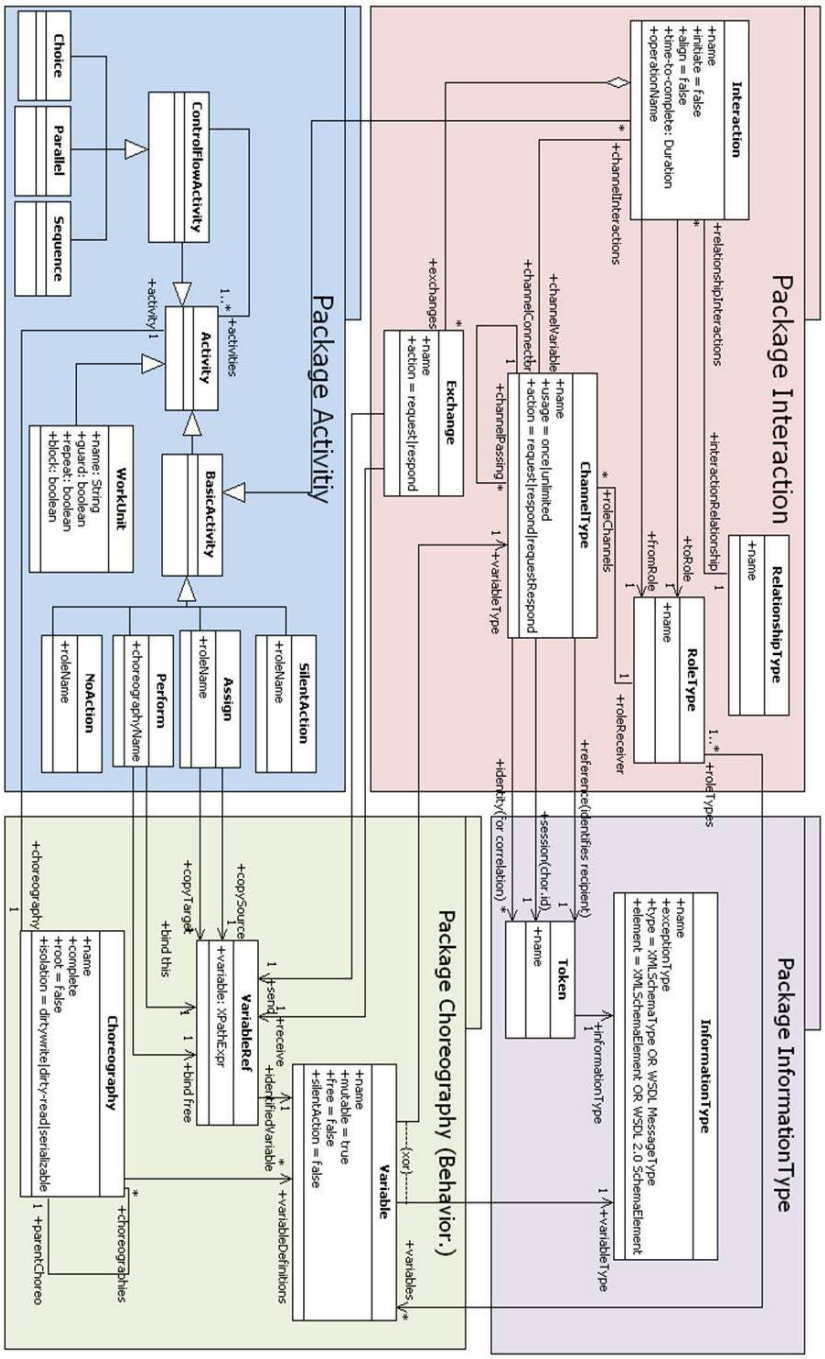

Fig. 11. WS-CDL design meta-model - behavioral view