

Transforming Multi-role Activities in Software Processes into Business Processes

Juan Pulgar and María Cecilia Bastarrica^(✉)

CS Department, Universidad de Chile, Santiago, Chile
jpulgar1978@gmail.com, cecilia@dcc.uchile.cl

Abstract. Software processes usually include activities involving several people playing different roles. SPEM provides primitives for defining all the roles involved in each activity. Software process specification notations are not executable and thus supporting tools cannot provide this functionality. Therefore, even having a formal software process specification we cannot achieve all the potential benefits: people have difficulties in following their responsibilities, resulting in a low productivity. The business process domain provides notations that can be executed on a BPMS. There have been attempts to transform SPEM specifications into BPMN. However, there is no natural way to model multi-role tasks in BPMN, and therefore none of these proposals has solved this issue. In this paper we discuss two promising alternatives for modeling multi-role software activities in BPMN: defining compound roles and modeling multi-role tasks as independent processes. We provide an XSLT transformation for automatically generating each of these solutions from a software process specification. We use a real world running example to illustrate the approach.

Keywords: Software process · Multi-role tasks · Model transformation

1 Introduction

Software processes are business processes in the software development domain, so most characteristics of the latter also hold for the former. They allow companies to better organize software development, and more specifically support management of project schedule, scope, resources, risks and expected software quality increasing the probability of success. Process definition is also a requirement if the company intends to achieve an ISO certification or a CMMI evaluation.

Software development is mainly an intellectual activity, so the most relevant resources to be managed in these processes are human resources. Tasks are the simplest work units in a software process, and they may be performed by one or more roles within a process.

From an enterprise point of view, a business process is a collection of activities that, given one or a set of inputs, creates a valuable output for the customer [15]. An executable process is one that may be run in a business process management system (BPMS). Specifying processes in an executable notation enables several other benefits as well:

- continuous process improvement based on empirical data
- reduced time invested in changes derived from business logic, and therefore more agility and flexibility
- reduced risks due to the possibility of simulating processes before executing them

Software Process Engineering Metamodel (SPEM) [9] and Business Process Model and Notation (BPMN) [14] are the standard notations proposed by the OMG for specifying software processes and business processes, respectively. On the one hand, SPEM provides rich primitives for specifying elements that are relevant for software processes, e.g. input and output work products associated with tasks, or roles assigned to each task. SPEM organizes these elements in two parts: the method content where each process element is defined, and the process where defined process elements are combined in order to configure a process. In particular, SPEM allows the definition of tasks with several roles assigned, but SPEM is not executable. On the other hand, BPMN 2.0 is potentially executable, but it does not have inherently a clear way for specifying tasks where several roles participate, and this is a serious limitation when using BPMN for software processes [16].

Once the company has invested in specifying its software process, it is appealing to transform it into BPMN so that it can be executed. There are several proposals for automatically transforming software processes into executable notations but, to the best of our knowledge, none of them addresses the specification of multi-role tasks.

In this paper we analyze two ways of specifying multi-role tasks in BPMN: assigning the task to a compound role formed by the set of roles involved, and specifying the multi-role task as a *meeting* that is fully specified separately. We provide two XSLT transformations that take the software process specification and automatically generate each of the analyzed alternatives. We illustrate the whole procedure with a real world bookstore software process and we discuss the pros and cons of each approach. Even though the resulting business processes are not directly executable as they are, it is a significant step towards this goal.

The rest of the paper is structured as follows. Section 2 presents some related work concerning software and business process notations, and discusses former attempts to transform SPEM processes into executable business process notations. Section 3 presents existing approaches for modeling multi-role activities. Section 4 presents the proposed approach using the running example for illustrating each alternative. Section 5 concludes.

2 Related Work

In this section we discuss approaches for modeling software and business processes, as well as attempts to transform from the former to the latter.

2.1 Software Process Specification

The OMG¹ has recommended SPEM [9] as the standard for software process specification. SPEM has been designed for describing processes and their components, following an object oriented modeling approach based on UML by extending its mechanisms for software process modeling.

Formally specifying a software process is a complex activity, but it allows for using a supporting tool that eases analysis and evolution. There are several tools available for this purpose. Some of them are: Eclipse Process Framework Composer (EPFC), Objectteering Modeler, MagicDraw UML (SPEM plug-in), Rational Method Composer and Enterprise Architect. Among these we can highlight EPFC [8] provided that it is free and therefore it is an appealing option. It internally implements UMA (Unified Method Architecture), an extension of SPEM that includes an UML profile for specifying process behavior, a feature not covered by SPEM [2].

2.2 Business Process Specification

There are several notations for specifying business processes such as: Petri Nets, UML activity diagrams, business process modeling notation (BPMN) [14], XML process definition language (XPDL) [5], integration definition (IDEF) [11], business process execution language (BPEL) [13], and event-driven process chain (EPC) [12]. BPMN 2.0 is widely used in industry mainly because [17]:

- is the OMG standard and thus there are several supporting tools
- provides a graphical notation easy to understand by stakeholders
- reduces the gap between the process design and its implementation
- it is potentially executable

A business process management system (BPMS) is a platform that allows companies to coordinate the realization of business processes based on process representations as models.

2.3 Transforming Software Processes into Business Process

There are several proposals that take SPEM 2.0 software process specifications and automatically generate executable processes in notations such as XPDL or BPEL. In [6] they propose a transformation from SPEM to BPMN using RSL as a transformation language, while [18] uses QVT. None of these proposals addresses multi-role tasks. MOSKitt4ME [4] takes processes specified in EPFC as we do, and transforms them into BPMN 2.0, but it can only transform a few kinds of process elements. In [1,10] they present a formal transformation from UML AD to BPMN 1.0 using MOLA and QVT, respectively, but these transformations do not address role assignment. In [7] we have proposed an XSLT transformation from software processes specified in EPFC into BPMN.

¹ Object Management Group - <http://www.omg.org/>.

However, that work did not address multi-role tasks either. In this work we address this issue and also upgrade the transformations into BPMN 2.0 so that they will eventually be executed in a BPMS; for this purpose we use Bonita².

3 Multi-role Tasks in Software Process Specifications

3.1 Running Example

We have defined the software process followed in a medium size bookstore in Chile. Figure 1 depicts the incidences process for this company specified using EPFC. In order not to overload the figure, EPFC includes neither roles nor work products in the activity diagram, but internally these relationships can be specified and checked. For example, the *Developer* is responsible for *Incidence Analysis*, *Solution Design* and *Solution Programming*, while *Incidence Reproduction* and *Testing* are the responsibility of *Developer* together with the *Business Analyst*, as shown in Table 1

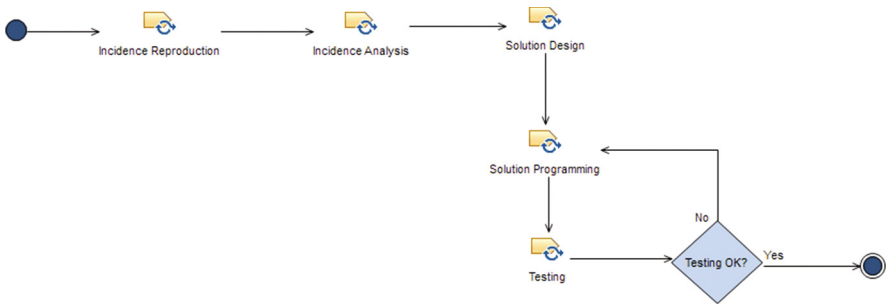


Fig. 1. Incidence process for the bookstore.

Table 1. Roles assigned to each task.

Task	Developer	Business analyst
Incidence reproduction	X	X
Incidence analysis	X	
Solution design	X	
Solution programming	X	
Testing	X	X

² <http://www.bonitasoft.com/products>.

3.2 Modeling Multi-role Tasks in BPMN

A pool is a modeling element in BPMN that sets the boundaries of a business process. It may be divided into lanes for organizing activities. It is a common practice to use lanes for assigning activities to roles, but there is no natural way to represent collaborative activities performed by different roles. Shapiro [16] has addressed this issue with different techniques; we describe them in what follows.

Tasks Duplicated in Each Lane. A first option is to duplicate the task in the lane of each role involved as shown in Fig. 2. A clear problem with this solution is that tasks specified in this way are not multiple copies of the same task, but different tasks with different identifiers, although sharing the same label. Therefore, the diagram shows multiple tasks performing in parallel. There may also be coherence problems when properties among replicated tasks do not match, or due to coevolution inconsistencies. This solution presents problems from a graphical point of view as well. When the task is performed by two roles appearing in consecutive lanes, the meaning may be clear, but when the diagram involves a lot of lanes, the diagram becomes messy and difficult to understand. A parallel gateway should also be added for coordination.

Different Tasks Assigned to Roles. In order to address some of the problems of the previous solution, we can divide the multi-role task into smaller coordinated tasks, assigning each of them to the participating roles as shown in Fig. 3. Modeling this solution may be quite complex because it may be required

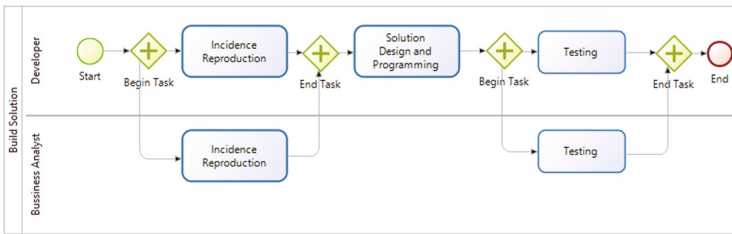


Fig. 2. Replicated multi-role task.

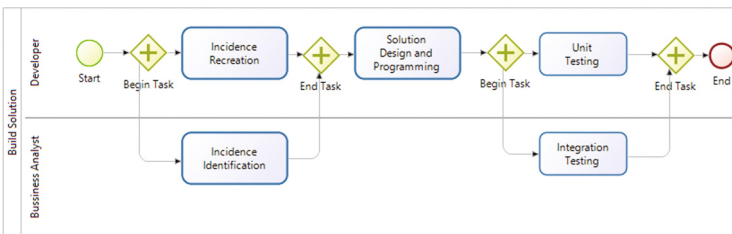


Fig. 3. Disaggregate multi-role task.

to invent several tasks that do not actually reflect the intended action, e.g., all actors that attend a meeting should perform a similar but different task instead of having just one unique activity representing the meeting in itself. Moreover, collaborative activities are not apparent just looking at the diagram.

Separate Lanes Representing Roles as Groups. Yet another proposal is to create a new lane within the pool that represents the set of roles involved in the collaborative task. This avoids task repetition and reduces the number of activities. In this case, the resulting model is more complex since it has more lanes and this situation is even worse when there are several multi-role activities with different sets of roles involved.

Shared Activities Modeled in Separate Processes. This approach uses call activities calling a different process where the multi-role activity is modeled in a pool with a single lane. The activity is assigned to the lane of the role responsible for leading the meeting, and then the meeting is detailed in separate processes, one for each participating role. The main drawback of this approach is that each multi-role activity will have a separate process.

4 Transforming UMA Specifications into BPMN

In this paper we consider the last two options from the previous section because Shapiro [16] recommends them as the most promising ones, and we show how they can be automatically generated from the EPFC specification.

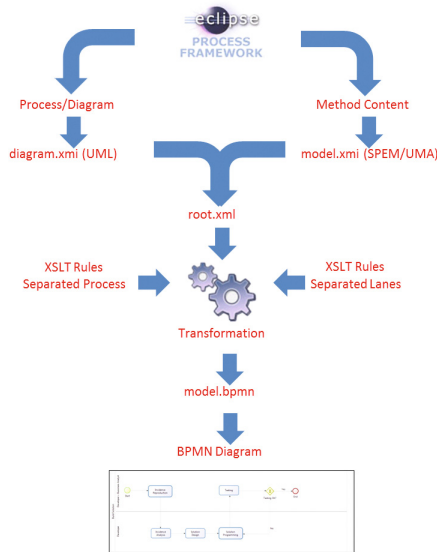


Fig. 4. Solution general structure




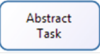

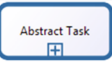







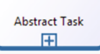

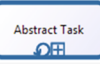

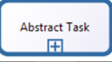


Description	SPEM Element	BPMN Element
Start		
Task, Collaborative Task (using Separated Lanes)		
Task, Collaborative Task (using Separated Process)		
Decision		
Fork, Join		
Merge		
Activity		
Iteration		
Phase		
End		

Fig. 5. Translation of UMA elements into BPMN elements.

Figure 4 describes the structure of the proposed solution. The software process specified in EPFC includes both, the method content and the process diagram. Both are combined into a single `root.xml` model that is taken as input for each XSLT transformation. The table in Fig. 5 indicates the way process elements are converted between notations in the transformations. In Sect. 4.1 we show the transformation of the software process into BPMN where multi-role tasks are assigned to compound roles specified as separated lanes. In Sect. 4.2 we show the transformation to automatically generate multi-role tasks represented as BPMN meetings that are then specified as independent processes.

4.1 Separate Lanes for Compound Roles

The idea is that each role has its own lane containing all the activities where he is the only responsible and, for those activities that have more than one responsible role, a separate lane will be created whose responsible is the compound role. Algorithm 1 shows the pseudocode for the XSLT transformation that takes the `root.xml` model as input and generates this BPMN diagram.

Algorithm 1. XSLT transformation for compound roles

```

Require: XML file containing  $\langle diagram \rangle$  and  $\langle model \rangle$  nodes (root.xml)
Ensure: XML file containing  $\langle definitions \rangle$  node for BONITA BPM, modeling collaborative activities in separated lanes for compound roles
1: Create a  $\langle definitions \rangle$  node
2: for all  $spemActivity \leftarrow \langle uml : Activity \rangle$  node in  $\langle diagram \rangle$  do
3:   Create a  $\langle process \rangle$  node in  $\langle definitions \rangle$ 
4:   Set the  $\langle process \rangle$  attributes (id, name) from the values of the  $spemActivity$  in  $\langle model \rangle$ 
5:   Create a  $\langle laneSet \rangle$  node in  $\langle process \rangle$ 
6:   for all  $spemTask \leftarrow \langle node[@xmi : type = ' uml : ActivityParameterNode'] \rangle$  node in  $spemActivity$  do
7:      $spemRole \leftarrow spemTask$ 's role(s) defined in  $\langle model \rangle$ 
8:     if  $spemRole$  is empty then
9:       Create a  $\langle lane \rangle$  node in  $\langle laneSet \rangle$ , setting the name attribute as Undefined
10:    else
11:      Create a  $\langle lane \rangle$  node in  $\langle laneSet \rangle$ , setting the name attribute as  $spemRole$ 's name
12:    end if
13:    Create a  $\langle flowNodeRef \rangle$  node in  $\langle lane \rangle$ , setting its value as  $spemTask$ 's id
14:  end for
15:  Group all  $\langle flowNodeRef \rangle$  by  $\langle lane \rangle$  node
16:  Create elements (as specified in the Node Transformation Table) appending them to the  $\langle process \rangle$  node
17: end for
18: return  $\langle definitions \rangle$  node
  
```

The XSLT transformation takes each activity in the EPFC `diagram.xmi` and generates a node in the BPMN `definitions` assigning the activity’s attributes, as stated in lines 3–4 in Algorithm 1. Then, it creates a `laneSet` (line 5) and, for each task in `model.xmi`, creates a lane (lines 6–14). Finally, lanes assigned to the same role are combined by role in line 16. In this way, when creating lanes for multi-role activities, an independent lane will be created, and it is not grouped afterwards with any of its participating roles.

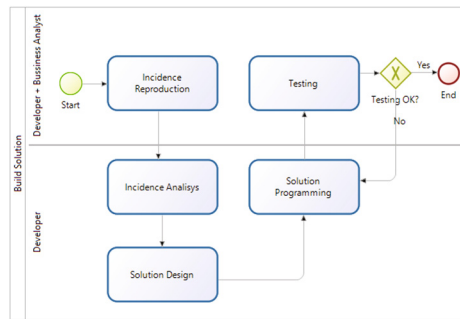


Fig. 6. Multi-role tasks assigned to compound roles

Figure 6 shows the visualization in Bonita BPMN of the result of applying this algorithm to the process in Fig. 1. We can see that *Incidence Analysis*, *Solution Design* and *Solution Programming* are assigned to the *Developer*'s lane, while *Incidence Reproduction* and *Testing* are assigned to the lane of the compound role *Developer+Business Analyst*. By default, the *Start* and *End* activities

are assigned to the lane of the role that performs the first and last activities, respectively; in this case this lane is that of the compound role.

This solution is quite clear in the sense that there is no redundancy, each activity appears only once in the BPMN diagram, and the flow of control is apparent. However, for a particular role, it is not obvious that he may be involved in other activities out of those that appear in his lane. In the example, the *Developer* does not only take part in his lane's activities, but also in *Incidence Reproduction* and *Testing* that appear in another lane. This situation may be even worse since the same role may collaborate with several other roles in different activities and thus his responsibilities may be split across the whole diagram.

4.2 Independent Processes

The second solution consists of a direct translation of each activity defined in the process in EPFC into a BPMN node assigned to the lane of the first role defined in the `model.xmi`, as shown in lines 3–7 in Algorithm 2. Then, for each node where there are more than one role assigned, the node is marked as a *meeting* and a new lane is created for each role involved and a process describing the activities that role must perform is defined with its own *Start* and *End* nodes.

Figure 7 shows the result of applying the algorithm to the process in Fig. 1. We can see that *IncidenceReproduction* and *Testing* are assigned to the *Developer* and marked as *meetings*. Then, in Fig. 8, the lanes corresponding to the specification of each of these activities is shown in a lane assigned to both roles *Developer+Business Analyst*.

Even though this solution seems to be clean and clear due to non duplicated activities, it may be somewhat inconvenient when multi-role tasks are defined in EPFC with its roles in different order. For example, if in the *Incidence Reproduction* tasks roles are define as *Developer* and *Business Analyst*, this activity will appear as a meeting in the *Developer's* lane. But, if on the contrary, if the roles are assigned as *Business Analyst* and *Developer*, then it will appear in the *Business Analyst's* lane.

4.3 Discussion

Although we do not think that any of the solutions is better than the other, we envision situations that favor one or the other. For example, if the same group of roles are usually assigned together to different tasks as in [3], then the first solution is clearer. On the other hand, if the EPFC software process specification is organized so that some of the roles are defined first in all the tasks they are assigned, then the second solution would be better.

Our initial goal was to transform SPEM processes into BPMN in order to make them executable in a BPMS. However, in order to make this possible there are certain other steps that should be performed before executability may be fully reached. First, an application for managing the process should be built

Algorithm 2. XSLT transformation for separate process

Require: XML file containing $\langle diagram \rangle$ and $\langle model \rangle$ nodes (root.xml)
Ensure: XML file containing $\langle definitions \rangle$ node for BONITA BPM, modeling collaborative activities in separated processes

- 1: Create a $\langle definitions \rangle$ node
- 2: **for all** $\text{spemActivity} \leftarrow \langle \text{uml} : \text{Activity} \rangle$ node in $\langle diagram \rangle$ **do**
- 3: Create a $\langle process \rangle$ node in $\langle definitions \rangle$
- 4: Set the $\langle process \rangle$ attributes (id, name) from the values of the spemActivity in $\langle model \rangle$
- 5: Create a $\langle laneSet \rangle$ node in $\langle process \rangle$
- 6: **for all** $\text{spemTask} \leftarrow \langle \text{node}[@xmi : type = 'uml : ActivityParameterNode'] \rangle$ node in spemActivity **do**
- 7: $\text{spemRole} \leftarrow \text{spemTask's first role}$ defined in $\langle model \rangle$
- 8: **if** spemRole is empty **then**
- 9: Create a $\langle lane \rangle$ node in $\langle laneSet \rangle$, setting the name attribute as *Undefined*
- 10: **else**
- 11: Create a $\langle lane \rangle$ node in $\langle laneSet \rangle$, setting the name attribute as spemRole's name
- 12: **end if**
- 13: Create a $\langle flowNodeRef \rangle$ node in $\langle lane \rangle$, setting its value as spemTask's id
- 14: **end for**
- 15: Group all $\langle flowNodeRef \rangle$ by $\langle lane \rangle$ node
- 16: Create elements (as specified in the Node Transformation Table) appending them to the $\langle process \rangle$ node
- 17: **end for**
- 18: **for all** $\text{spemTask} \leftarrow \langle \text{node}[@xmi : type = 'uml : ActivityParameterNode'] \rangle$ node in $\langle diagram \rangle$ **do**
- 19: $\text{spemRole} \leftarrow \text{spemTask's roles(s)}$ defined in $\langle model \rangle$
- 20: **if** spemRole has more than one element **then**
- 21: Create a $\langle process \rangle$ node in $\langle definitions \rangle$
- 22: Set the $\langle process \rangle$ attributes (new id; name as the concatenation of 'Meeting' and spemTask's names)
- 23: Create a $\langle laneSet \rangle$ node in $\langle process \rangle$
- 24: Create a $\langle lane \rangle$ node in $\langle laneSet \rangle$, setting the name attribute as spemRole's names
- 25: Create a task, a start node and an end node
- 26: Create three $\langle flowNodeRef \rangle$ nodes in $\langle lane \rangle$, setting the value as the new task, start node and end node id's
- 27: **end if**
- 28: **end for**
- 29: **return** $\langle definitions \rangle$ node

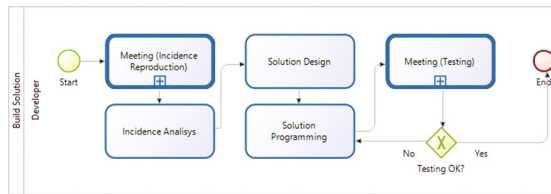


Fig. 7. Multi-role tasks assigned to the first role and marked as a meeting

together with designing the business data. Then, the process itself is built. In this stage, the first step is to create the BPMN process, and this is the step we have automated with our proposal. Then, still other information should be provided such as the business variables, contracts, and actors that will carry out the process steps.

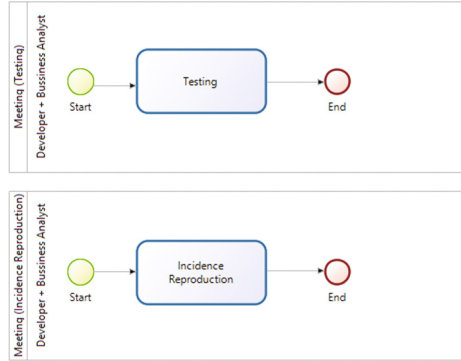


Fig. 8. Meeting tasks assigned to each role taking part

5 Conclusion

In this paper we proposed a strategy for transforming software processes specified in EPFC -conforming to the UMA metamodel- into BPMN processes. We were able to automatically generate the two most promising solutions for multi-role activity specification suggested by Shapiro et al. [16]. We provided two XSLT transformations, one for obtaining each solution, where multi-role tasks are modeled as activities in lanes assigned to compound roles or meetings that are specified in detail as independent processes. Being able to clearly identify responsibilities allows people involved in software project execution to optimize their time, and therefore the whole project productivity. Moreover, being able to automatically transform the software process already specified into a business process, reduces the effort by reusing the specification already available. Even though the resulting diagrams are not directly executable, they constitute an important step towards executing a software process that is consistent with that specified in EPFC.

We were able to transform the software process of the running example. However, and even though it is a real world software process, it is still necessary to apply the approach to larger processes in order to identify some limitations that may be hidden because of the small size of the example. So far, we were also able to transform a more complex process of a much larger organization, that also includes process patterns and iterations. These results are encouraging. However, we will be able to obtain the most out of our transformations when we build a tool that helps choosing the most appropriate approach in each case. It is also necessary to count on objective measurements of each solution quality such as understandability, complexity and completeness, among others.

Acknowledgments. This work is partly funded by Project Fondef IT13I20010, Conicyt, Chile.

References

1. Argaaraz, M., Funes, A.M., Arístides, J.: An MDA approach to business process model transformations. *Electron. J. SADIO (EJS)* **9**, 24–48 (2010)
2. Bendraou, R., Combemale, B., Crégut, X., Gervais, M.: Definition of an executable SPEM 2.0. In: 14th Asia-Pacific Software Engineering Conference (APSEC 2007), 5–7, Nagoya, Japan, pp. 390–397. IEEE Computer Society, December 2007
3. Cabanillas, C., Resinas, M., Mendling, J., Cortés, A.R.: Automated team selection and compliance checking in business processes. In: Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24–26, 2015, pp. 42–51. ACM (2015)
4. Cervera, M., Albert, M., Torres, V., Pelechano, V.: The MOSKitt4ME approach: providing process support in a method engineering context. In: Atzeni, P., Cheung, D., Ram, S. (eds.) *ER 2012*. LNCS, vol. 7532, pp. 228–241. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-34002-4_18](https://doi.org/10.1007/978-3-642-34002-4_18)
5. W. M. Coalition. XML Process Definition Language (XPDL) (2012). <http://www.xpdl.org/>
6. Cota, M.P., Riesco, D., Lee, I., Debnath, N.C., Montejano, G.: Transformations from SPEM work sequences to BPMN sequence flows for the automation of software development process. *J. Comput. Meth. Sci. Eng.* **10**(3–6), 61–72 (2010)
7. Cruz, D.E., Bastarrica, M.C., Duarte-Amaya, H.: De procesos SPEM a procesos BPMN. Un enfoque basado en MDE. In: *CIBSE*, pp. 41–52 (2014)
8. E. Foundation. Eclipse Process Framework Project (EPF) (2015)
9. O. M. Group. Software & Systems Process Engineering Meta-Model Specification (2008). <http://www.omg.org/spec/SPEM/2.0/>
10. Kalnins, A., Vitols, V.: Use of UML and model transformations for workflow process definitions. *CoRR*, abs/cs/0607044 (2006)
11. I. K. Knowledge Based Systems. IDEF. Integrated DEFinition Methods (2016). <http://www.idef.com/>
12. Korherr, B.: Business Process Modelling - Languages, Goals, and Variabilities. Ph.D. thesis, Vienna University of Technology, January 2008
13. OASIS. Web Services Business Process Execution Language Version 2.0 (2007). <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
14. OMG. Business Process Model and Notation, Version 2.0 (2011). <http://www.omg.org/spec/BPMN/2.0/>
15. Sánchez-González, L., García, F., Ruiz, F., Velthuis, M.P.: Measurement in business processes: a systematic review. *Bus. Process Manage. J.* **16**(1), 114–134 (2010)
16. Shapiro, R., White, S.A., Bock, C., Palmer, N., zur Muehlen, M., Brambilla, M., Gagné, D. (eds.): *BPMN 2.0 Handbook*. Workflow Management Coalition, Lighthouse Point (2012)
17. Wohed, P., Aalst, W.M.P., Dumas, M., Hofstede, A.H.M., Russell, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) *BPM 2006*. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006). doi:[10.1007/11841760_12](https://doi.org/10.1007/11841760_12)
18. Zorzan, F.A., Riesco, D.: Transformation in QVT of software development process based on SPEM to workflows. *IEEE Latin Am. Trans.* **6**(7), 655–660 (2008)