Collaborative Activities Inside Pools

Michele Chinosi, European Commission Joint Research Centre, Italy

Introduction

Choreographies and Conversations, introduced with BPMN 2.0, will make modelers able to describe interactions among different Participants as well as messages exchange. Often enough different Participants have to accomplish the same task. This can be now easily and clearly represented using BPMN 2.0. BPMN 2.0 does not specify the usage of Lanes neither their meaning. However, Lanes are sometimes used to specify internal roles or departments.

In this context it could happen that modelers want to represent an Activity performed by different roles or offices together (e.g., attending the same meeting, collaborative writing of a document). Such situation has been modeled so far by using merging Gateways placed before the activities, but this patch does not solve a related problem. BPMN forces to draw elements within Lanes boundaries. This means that, at least conceptually, one Activity is lead by the subject which the containing Lane is linked to, which is not necessarily true. Some experiments revealed how much the means to model such inner collaboration is a desirable feature.

Analysis of the Problem

BPMN allows people to model many different scenarios involving one or more participants, simple or complex activities, splitting and merging paths, messages exchange, different points of view, and so forth. With the advent of BPMN 2.0 a very big interest has been put on how to better represent the relationships among different participants (also known as Collaborations), in particular through the introduction of Conversation diagrams and Choreographies. Now it is possible to model a complex message exchange among multiple participants focusing on messages and relationships among the involved actors/roles. However, one of the most common scenarios both in private and in public sectors is represented by shared activities, or, to put it better, the need to model one single Activity involving more than one participant at the same time. Common examples of such shared or collaborative activities are meetings, conferences, working groups with same goals to achieve, as well as new scenarios faced out in the last years, like cloud computing, collaborative real-time document writing, collaborative modeling. Probably one of the most well-known example which excited many users in 2009/2010 was the launch of the Google Wave collaborative platform[1].

BPMN offers different ways to model a business process. It is possible to model a private process, a public process, a collaborative process as well as the already mentioned Choreographies and Conversation diagrams. The usual way (actually, it is a common practice inherited from BPMN 1.2) to model an activities flow performed by different actors/roles is describing the business process from the point

[1] http://wave.google.com/about.html

of view of public or collaborative diagrams (i.e., highlighting the messages exchange among the participants). An explicative example on how to model a collaborative Activity by using BPMN 2.0 is shown hereinafter.

A meeting with regional partners

Figure 1 shows an example diagram in which there are two participants (the Project Manager and Regional Partners). The depicted scenario focuses on a general project management procedure for a company with many regional partners worldwide. This Business Process Diagram (BPD) also takes advantage of the possibility introduced in BPMN 2.0 to mark a Pool as Multi-Instance if the Participant defined for the Pool is a Multi-Instance Participant, as it is in this example.

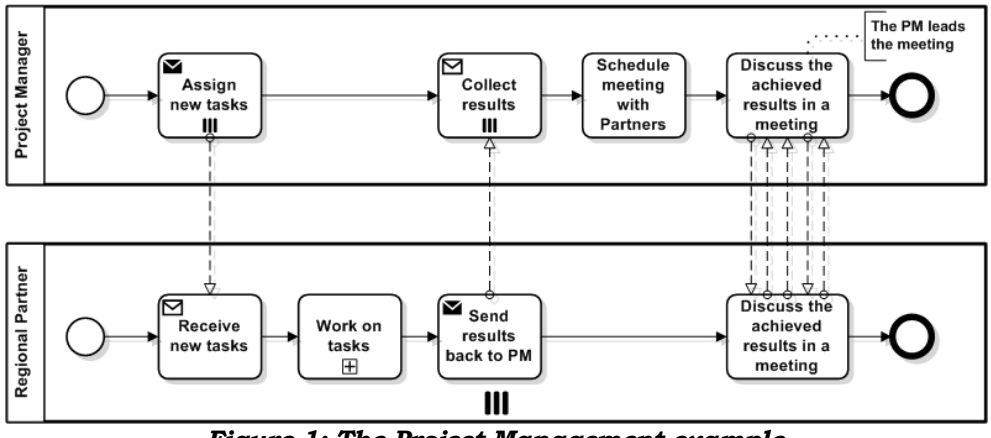

Figure 1: The Project Management example

The last Task in both Pools represents a meeting between the two Participants leaded by the Project Manager. It is possible to have the same Task in different Pools because each Task belongs to a different Participant even if Participants are physically in the same place at the same time doing the same things.

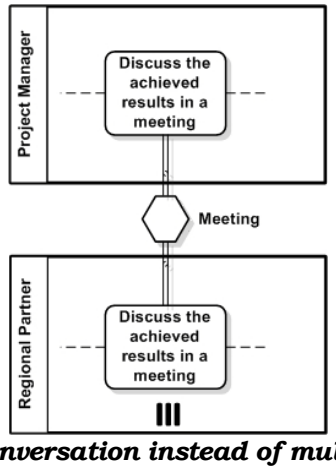

Figure 2: Using a Conversation instead of multiple Message Flows

Message Flows can represent in this case the speeches given by attendees. It is not mandatory to show them, unless there is the need to specify the subject of the speeches. Nevertheless modelers have to keep in mind that they are asked to model an abstract process, not a particular instance of a process. To use Message Flows with no labels or messages just to represent a discussion happening at a

certain point should be avoided. Instead, the same process can be modeled more concisely and neat using a Conversation, as shown in Figure 2. By using a Conversation it is possible to model a meeting avoiding the risk to model only one particular instance of a process rather than a general process. Furthermore, the same BPD can be also represented from a different point of view using a Choreography, as in Figure 3. The biggest advantage of using a Choreography in this case is that all the private details can be left hidden focusing only on relationships among Participants.

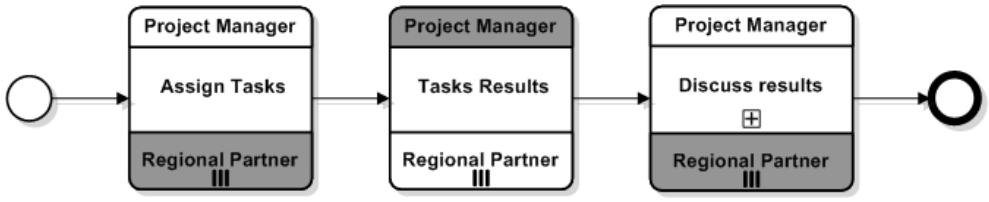

Figure 3: The Choreography associated to the example process

Finally, there is another way to deal with collaborative activities in a Collaborative Process context, especially if it is not important to show the messages exchange. BPMN 2.0 introduces the concept of Global Tasks and Call Activities. It is possible to define a Task or a Sub-Process as Global Task referencing it from any other Process using a Call Activity (even if under certain constraints). Thus, once the main Participant of a shared Activity has been identified, modelers can put the Global Task element in its Process while putting relating Call Activities in all the other involved Processes.

But so far nothing new has been shown, apart from giving few hints on using new BPMN 2.0 characteristics to model collaborative scenarios including shared activities. However, we dealt with multiple Participants represented with different Pools, where most common troubles regard representation or synchronization issues. To model shared activities in an internal context is a more complex problem, somehow concerning meaning (*semantic* is probably a too strong term in this context) often imposed to Lanes.

DO LANES HAVE A MEANING?

The definition of Lanes as it is written in BPMN specifications is somehow misleading. “BPMN does not specify the usage of Lanes” neither their implicit or explicit meaning. In fact, “the meaning of the Lanes is up to the modeler”. Lanes have to be treated just as “sub-partitions within a Process (often within a Pool)”, and used “to organize and categorize Activities within a Pool”. In practice, “Lanes are often used for such things as internal roles, systems, an internal department, etc”[2].

It is clear that Lanes do not have a *semantic* and that *meaning* refers just to the name modelers give to Lanes, even if the difference between “semantic” and “meaning” is not always clear. In computer science, the term *semantic* is applied to certain types of data structures specifically designed and used for representing information content[3], and this definition is coherent with the definition of Lanes provided by BPMN specifications. In semantics, the term *meaning* refers to the ob-

[2] Business Process Model and Notation (BPMN), Version 2.0, OMG Document Number dtc/2010-06-05

[3] http://en.wikipedia.org/wiki/Semantics

jects or concept that a word or phrase denotes, or that which a sentence says. In other words, the meaning is inferred from objects or concepts expressed by words, phrases or sentences[4]. Furthermore, Lanes do not have a Participant associated to them, but instead they inherit the Participant from the owning Pool. However, the statement "the meaning of the Lanes is up to the modeler" leads modelers to believe that giving Lanes a meaning is permitted, thus pretending to use Lanes as they were nested Pools.

It is difficult in general to see a Lane only as a graphical container intended to categorize and organize Activities in a Process. By giving names to Lanes, modelers tend to implicitly associate also a meaning (or at least a behavior) to the associated roles. This practice becomes more marked considering the goals most modelers have in mind when they start modeling a process in BPMN. Many conducted experiments, along with a web sentiment analysis on this topic, revealed how the 90% of modelers loves BPMN as a graphical mean to document processes, while only the remaining 10% is also really interested in simulating, deploying, executing, exporting business processes using BPMN. By considering BPMN only from a graphical point of view, modelers cannot be aware of the big differences in terms of properties, attributes and relationships that lie behind the definitions of Pools and Lanes, especially if they choose to use BPMN editors with only graphical shapes support. Moreover, common practices misguide modelers to give Lanes a name representing roles or functions implicitly applying to them also a semantic, thus modeling business processes having in mind a separation of duties more than a mere graphical arrangement of the Activities.

In this context, it could happen that at a certain point in the middle of a process diagram modelers want to represent an Activity performed by different roles or offices together (e.g., attending the same meeting, collaborative writing of a document), thus changing the granularity level and the point of view on the process. Unfortunately, it is not always possible to change from a private to a collaborative level in the middle of a process (at least, not in an intuitive way).

Summing up, it is possible to outline the problem using a general question: when in the midst of a Swimlane diagram you need to model a shared Task, how do you model it if a Task can only belong to one Swimlane? Before BPMN, many would just extend a Task across Swimlanes. How do we handle this in BPMN?

Such situation has been tackled so far by using different techniques like Gateways placed before and/or after the activities. In such cases, where it becomes important to model interactions among different Participants, Pools should be used instead. A big debate is still ongoing, and during the last two years many attempts have been proposed through printed books[5] and web communities. One of the longest discussions on this topic took place in the *BPMN-Community*'s forum[6]. Some valuable proposals are reported in the following.

[4] http://en.wikipedia.org/wiki/Meaning

[5] See for instance: Silver B., "BPMN Method & Style", Cody-Cassidy Press, 2009; Debevoise T and Geneva R., "The Microguide to Process Modeling in BPMN", Booksurge Publishing, 2008; Sharp A. and McDermott P., "Workflow Modeling", Artech House Pub., 2001.

[6] See for example http://en.bpmn-community.org/forum/78/ and http://en.bpmn-community.org/forum/17/, last accessed 25th of October, 2010.

COLLABORATIVE ACTIVITIES INSIDE THE SAME POOL

The example scenario is the one depicted in Figure 4, where a single report for a project is prepared by multiple actors/roles working together in real time (e.g., in meetings) or near-real-time (communicating continuously). To fill the report is an internal process for a company, involving different roles performed by colleagues.

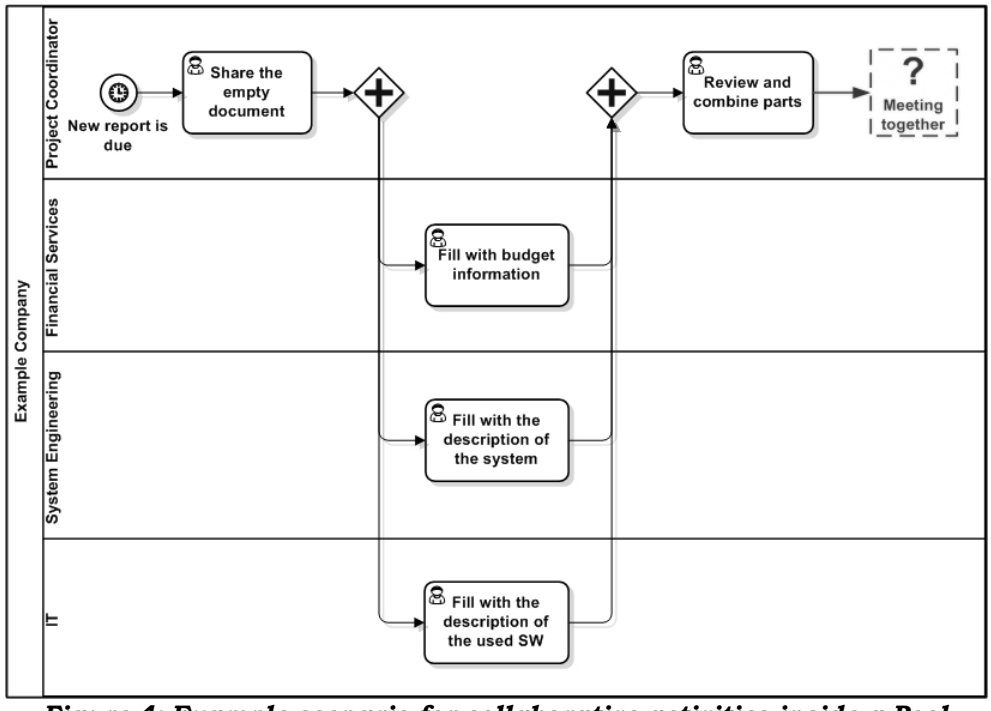

Figure 4: Example scenario for collaborative activities inside a Pool

From this point of view it is fair to model the process using one single Pool with multiple Lanes, one for each actor/role, because there is no messages exchange among different Participants but just an internal documental workflow. Quite at the end of the process a final meeting is foreseen, attended by all of the contributing authors. But how to model (or, better, graphically represent) at that point an Activity performed together by all the roles? As a matter of facts, most users would prefer seeing in an intuitive way this kind of collaborative activity on a model (considering that BPMN aims to be an easily readable notation understandable by all users from business users to technicians) rather than sifting through textual properties and attributes values, like instead technicians or computer scientists love to do.

Solution #1: Duplicate Tasks in each Lane

The first solution consists in duplicating the Task for each involved Lane, like in Figure 5, maybe also grouping all the Tasks together within a Group to improve the readability of the diagram (Figure 6).

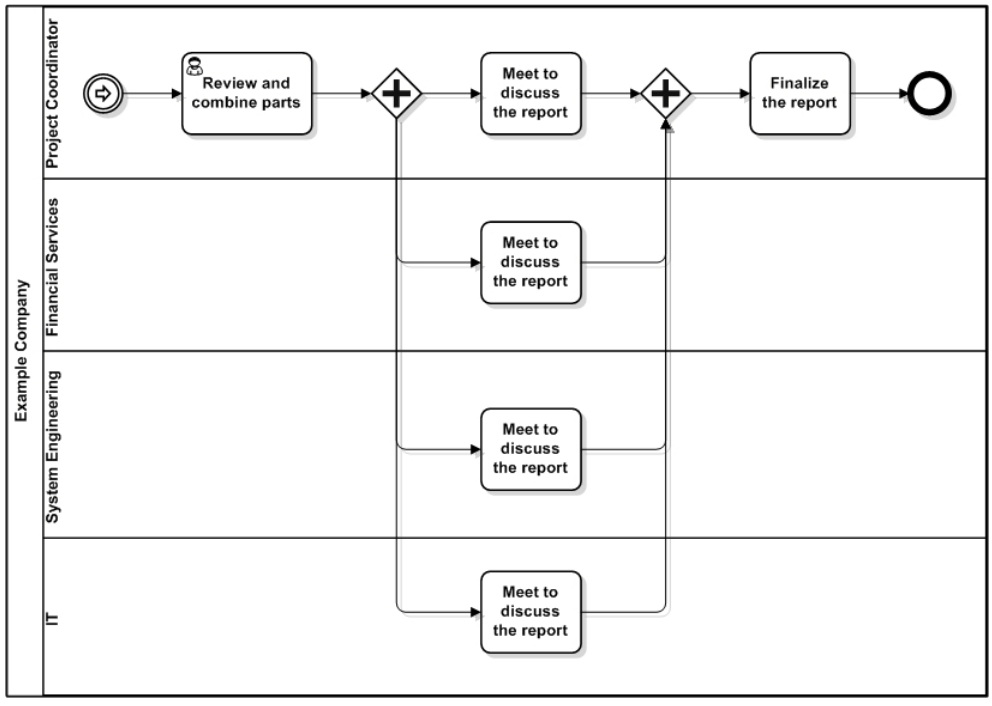

Figure 5: Solution #1 - Duplicate Tasks

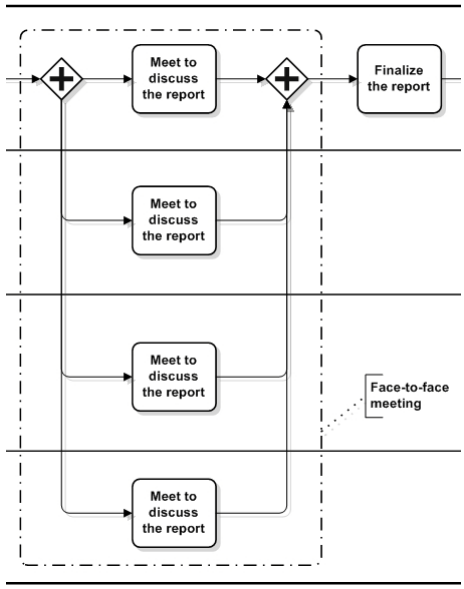

Figure 6: Solution #1 - Duplicated Tasks grouped together

But this solution introduces many problems. Duplicated Tasks are not multiple copies of the same original Task, otherwise the diagram will not be valid with respect to the BPMN specifications, even if, at least conceptually, it could seem so (and probably this is the meaning the author intended to pass to the reader). Instead, all the Tasks have different IDs, even if they share the same label, thus resulting in a diagram with multiple separate Tasks performing the same action in parallel (they are put in between two Parallel Gateways). Moreover, there may oc-

cur coherence problems, because modelers have to guarantee that all the properties values match among the Tasks, as well as refactoring problems due to the necessity to reflect each modification affecting one Task to all the others. Even from a graphical point of view this solution should be avoided. In the shown example there are only four Lanes, and they are contiguous. If the number of the Lanes is higher, or if Lanes are not contiguous but separated by other Lanes, the diagram will become complex and messy. In fact, each involved role/actor should have the same Activity in its Lane while BPMN does not allow to have multi-instance Lanes to represent multiple subjects sharing the same Activities set as it does for multi-instance Pools. Besides, obviously, it is always necessary to synchronize all the Activities with a Parallel Gateway.

Solution #2: Assign different Tasks to actors/roles

To solve some of the problems of the first solution, supposing it is feasible to slightly change the description of the modeled scenario, it would be possible to assign different Tasks to each actor/role, by decomposing the original Task in smaller actions, like in Figure 7. All the roles/actors keep on collaborating to the same shared Task, even if in this diagram all the single parts have been explicitly represented.

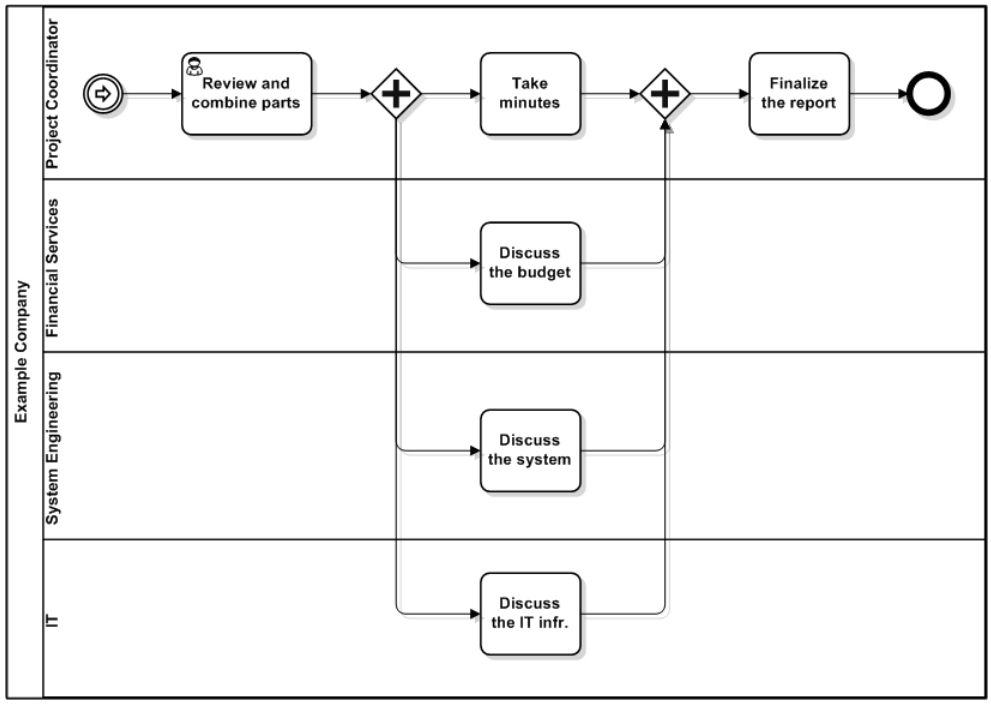

Figure 7: Solution #2 - Assign different Tasks

The depicted solution has fewer problems than the first one, but nevertheless other difficulties face out, especially affecting the modeler more than they were only syntactical or validation issues. In fact, there could be too many actions to invent: each actor/role should have its own action to perform but it is not necessarily true that for each actor/role there exists a particular action to perform. In case of a meeting, many actors/roles have just to attend the meeting, doing nothing special other than listening to the speeches. Another issue concerns the semantic aspect of the diagram whose meaning became weaker by adopting this so-

lution: looking at the diagram, the first impression is not of a meeting, but of different activities taking place at the same time.

Solution #3: Create a separate Lane to represents actors/roles as group

Another approach introduces a new Lane within the Pool to represent the group of actors/roles involved in a common Task. This would permit avoiding Tasks repetition and reducing the activities number. The solution is presented in Figure 8.

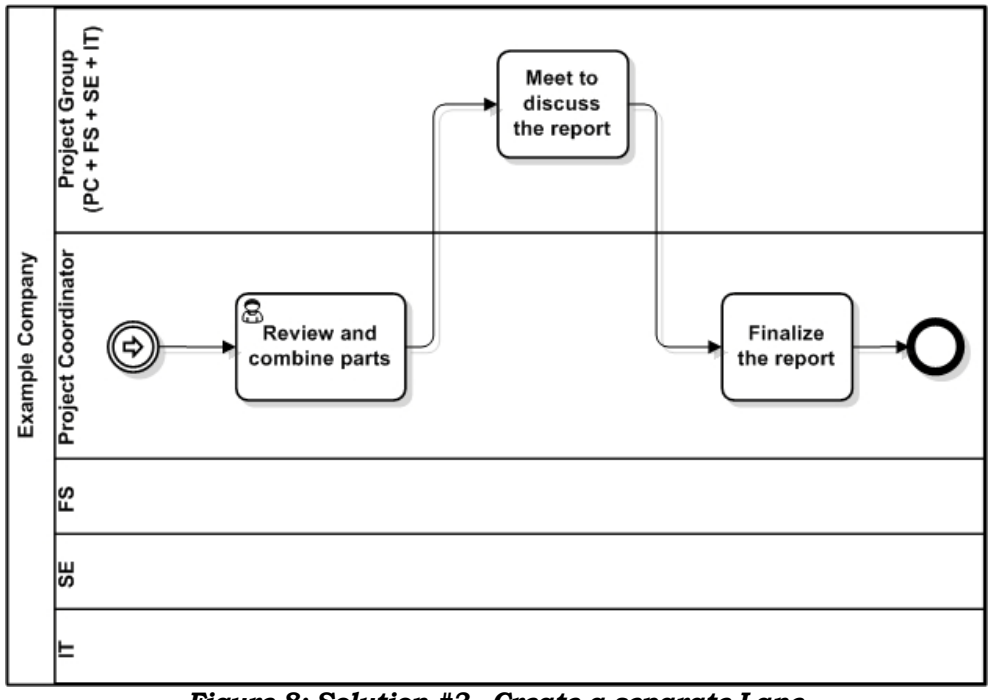

Figure 8: Solution #3 - Create a separate Lane

But also in this case there could be problems. First of all, the complexity of the model increases because a new Lane has been added, perhaps for just one Task. Another difficulty for modelers will come out if in the process there are different groups of actors/roles, for example if after the plenary meeting in the example another meeting between the Project Coordinator and Financial Services takes place. By adopting this solution, each time a shared Activity has to be modeled for a new group of actors/roles, a new Lane will be added to the process.

A slight variation of this solution, presented in Figure 9, uses the nested Lanes mechanism. Although the diagram may looks better, this variation prevents modelers from adding now other groups if needed. Moreover, multiple nesting, although "legal", often confuses the business users.

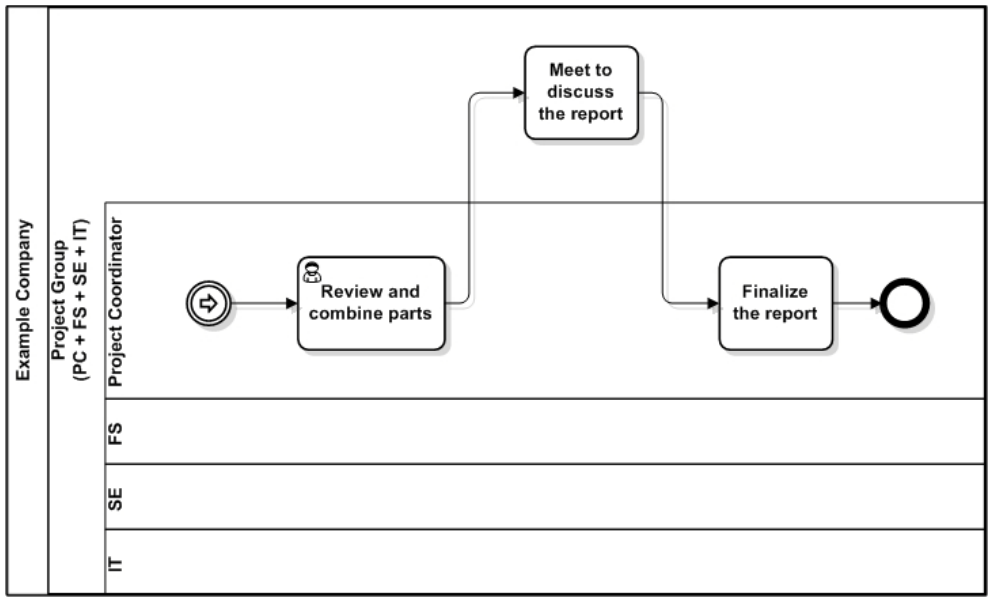

Figure 9: Solution #3 - Create a separate Lane for groups nesting all the other involved Lanes

Solution #4: Shared activities are modeled in a separate Process

The last considered approach, completely different from the others mentioned so far, uses Call Activities calling a different Process where the shared activity has been modeled in a Pool holding a single Lane representing the group of actors/roles performing the action. An example is shown in Figure 10.

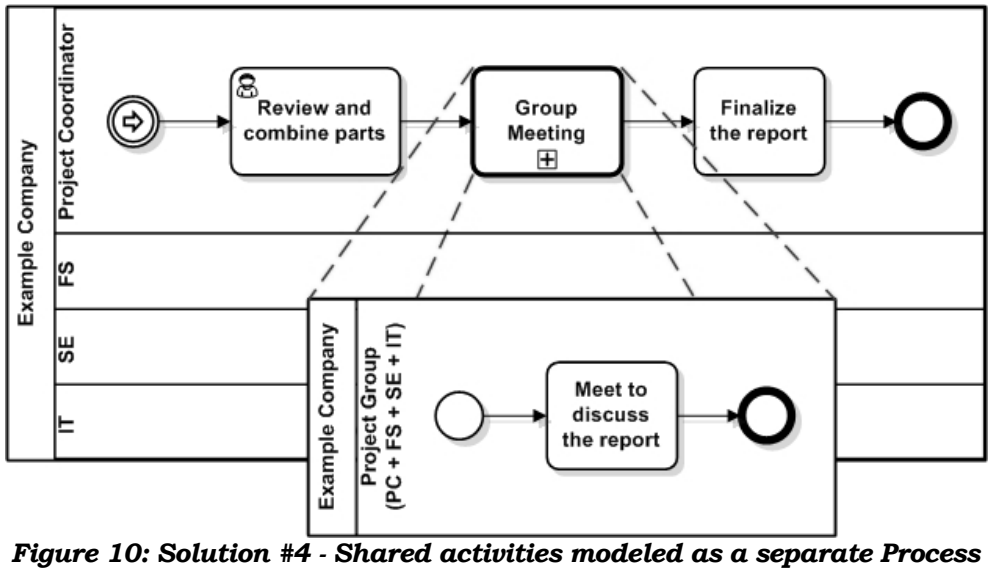

Figure 10: Solution #4 - Shared activities modeled as a separate Process

It is worth noting that the Call Activity is contained in the Lane pertaining to the actor/role responsible for leading the meeting. In other words, modelers could model the shared activities involving multiple actors/roles in a separate Process (one for each Activity and for each group of actors/roles) placing the Call Activity

inside the Lane of the actor/role which is supposed to be the main performer of that activity. The biggest disadvantage is that each simple Task representing a shared or collaborative activity will have a related Process, compelling modelers to define every property and attribute to be BPMN compliant. Thus, the complexity of the whole BPD will increase quite exponentially.

Representing shared Artifacts

The depicted problem of representing shared tasks or collaborative activities becomes even more complex at the time that Data Objects should be added to the diagram, especially if the process describes a collaborative editing of the same document. Among the solutions presented so far, only the third and the fourth solutions can be used to represent a single Data Object edited by different actors/roles at the same time. The second solution can be used only in the context of a common document whose different sections are edited independently by different actors/roles (see Figure 11 on the left), while the first solution would require to add a single Data Object with multiple Data Associations, one for each Task. A trivial solution in this case would be to link through a Data Association one Data Object to a common incoming Sequence Flow of each Task, doing the same for the output of the editing Task (like in Figure 11 on the right), but thus skipping the representation of the editing process of the document.

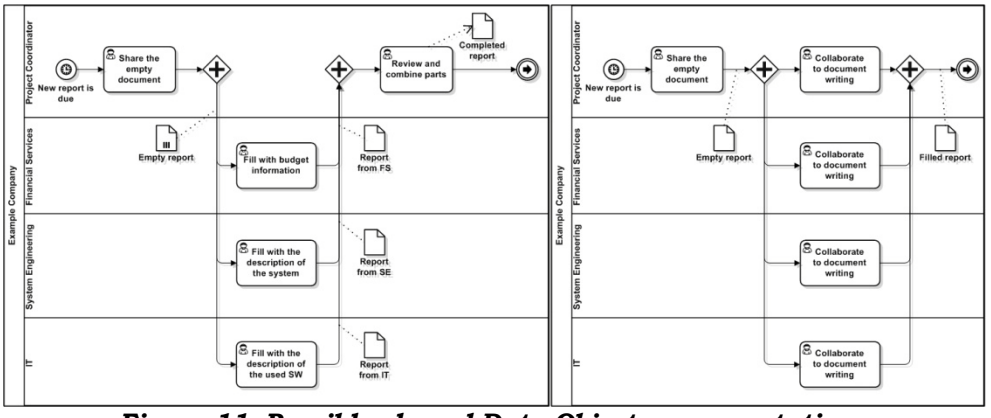

Figure 11: Possible shared Data Objects representations

PROPOSED SOLUTIONS

In this section some other solutions will be delineated. Unlike solutions discussed so far, arisen from common practices, the following originate from BPMN 2.0 specifications, in order to provide alternatives on one hand to fulfill as much as possible users' requirements, while on the other hand being compliant with BPMN specifications as well. While the first two proposals presented here will follow as much as possible the BPMN specs, the last one will suggest an extension to BPMN to meet both the requirements of representing collaborative activities and keeping the diagram easily readable.

Use the Performer class

BPMN 2.0 introduces the Performer class, which defines the resource that will perform or will be responsible for an Activity. The performer can be specified in the form of a specific individual, a group, an organization role or position, or an

organization. The Performer element does not have attributes or model associations other than ones inherited from the BaseElement.

This solution is fully BPMN compliant, thus preserving the "no semantic" property of Lanes. A Performer can also be a "group". If Lanes have no meaning, then one Lane can refer to a single subject but also one of its contained Activities can have a "group" as Performer.

The main disadvantage of this solution is that the Performer element is not shown in a diagram, forcing users to look through the properties of the Process to look for the Performer of an Activity checking if it is a collaborative or a single-user Activity.

Use Pools instead of using Lanes

Global Tasks and Call Activities have been defined in BPMN to allow modelers recall already defined (often common) Tasks in different Processes. In a way almost similar to the fourth solution presented above, it is possible to use a Call Activity calling a Collaboration process, instead of calling another Private process as depicted there. The called Collaboration Process can also be the Collaboration view of a corresponding Choreography modeling the shared Activity.

The main disadvantage of this solution lies in the complexity of the process to be modeled. To avoid such complexity, modelers can put much more effort during the planning phase before starting modeling the process, considering whether to model the main process as a Private process calling a Collaboration process or starting directly modeling a Collaboration process.

Extend Global Tasks and Call Activities

Even if Global Tasks and Call Activities have been defined in BPMN to allow modelers recall already defined Tasks in different Processes, it is possible to use this mechanism also inside the same Pool. This is quite a strained interpretation of the given definitions. "A Global Task is a reusable, atomic Task definition that can be called from within any Process by a Call Activity". "A Call Activity identifies a point in the Process where a global Process or a Global Task is used. The Call Activity acts as a 'wrapper' for the invocation of a global Process or Global Task within the execution. The activation of a Call Activity results in the transfer of control to the called global Process or Global Task". The problem with Call Activities lies in the activation of a Call Activity, which results in the transfer of control to the called Global Process or Global Task. This means that a Call Activity can override properties and attributes of the element being called, potentially changing the behavior of the called element based on the calling context. This is the main reason because it is not possible to use Global Task and Call Activities together inside the same Pool to execute the same Task in parallel in a collaborative environment.

This solution suggests instead to extend the Global Task/Call Activity definitions (and their graphical notation as well) by adding new attributes to the Performer class, as in Table 1.

Attribute Name	**Description/Usage**
`cardinality: string = Single { Single | Group | Open }`	This attribute defines the number of Performers for a collaborative Activity. The default value is Single, meaning an Activity performed by a single subject. Group identifies collaborative Activities executed by a defined group of subjects,

	while Open is for open activities like conferences, meetings, public discussions or events.
`name: string [0..1]`	This optional attribute defines the name of the Performer, whether it is a single subject or a group.
`leadingPerformer: string`	This attribute defines the subject leading the collaborative Activity.

Table 1: New attributes for the Performer class

It is worth noting that the *name* attribute is optional. The Performer class inherits *BaseElement* attributes without adding any other details, thus the name of a Performer can be specified through the inherited *id* and *documentation* attributes, even if it is strongly suggested to use a separate name qualifier to identify the Performer. From a graphical point of view, a new icon for Tasks and Sub-Processes is proposed. It represents a little chain to highlight the concept of an Activity performed together by a number of different subjects, like in Figure 12. The name given to this new Activity is Chain Task or Chain Sub-Process.

Figure 12: An example of Chain Task

Two different use cases are foreseen. First, it is possible to use the Chain Activity by replicating the same Activity in all the involved Lanes. In this case the Activity related to the `leadingPerformer` should be drawn with a thin border (like Global Tasks) while all the other Activities should have a thick border also displaying the same marker (as for Call Activity definition), as it is possible to see in Figure 13.

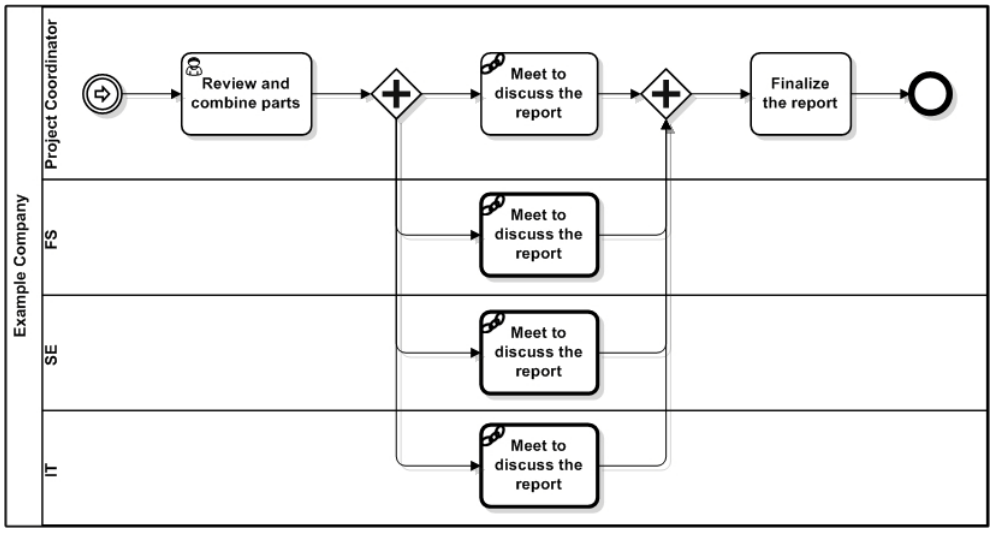

Figure 13: Chain Activities applied in the running example

The main difference from Call Activities is that the semantic of a Call Activity of type Chain acts only as a bookmark for the Global Chain Task, which is the only element responsible for the execution of the Task. That is a Call Activity of type Chain cannot override properties and attributes of the element being called, neither change the behavior of the called element.

In the second use case, instead, it is enough to add a Chain Activity in the Lane related to the `leadingPerformer`. By using the new attributes of the Performers it is possible to specify the cardinality of the involved subjects and the subject leading the Activity. A simplified example of this use case can be seen in Figure 14.

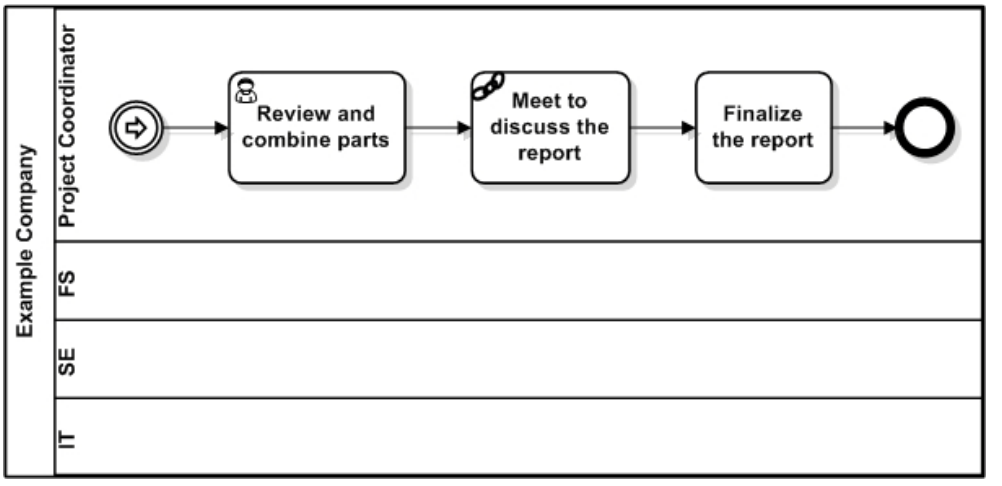

Figure 14: Only one Chain Task in the `leadingPerformer`'s Lane

CONCLUSIONS

BPMN allows modelers to choose the proper granularity level and the more appropriate design methodology to fulfill their representation needs. But BPMN is not a palette of graphical symbols. Instead, BPMN is first of all a leading standard for business process modeling, and even more if considering the high number of new functionalities and features introduced in the version 2.0. Therefore, even if the majority of modelers are approaching BPMN purely as a nice graphical way to document business processes, specifications should be accurately followed. One key point of BPMN concerns the freedom given to the modeler not to follow predefined schemas, but rather to use the notation as a handy but powerful tool.

Unfortunately, this freedom sometimes drives modelers to unforeseen scenarios, as for the need to model collaborative activities inside one Process leaded by only one Participant, thus looking for a solution to use Lanes as sub-Pools with their own “sub-Participants”. By looking into literature, academic papers and web communities it is possible to see many examples, common practices and design methodologies, but real scenarios are often much more complex than the examples reported there, and the specifications themselves are a hard reading for BPMN beginners. The first four solutions discussed in this paper come from this context. The other three proposed solutions are tentative attempts to fulfill modelers needs while keeping the diagrams as simple as possible and also valid. Clearly, it is possible to adopt some of the solutions presented in the first part of

the paper, at least those ones not contradicting the BPMN specifications. Best candidates are solutions number 3 and 4, followed by solution number 2 which is formally correct, although too complex, while the first solution should be discarded. On the other hand, two of the suggested solutions presented in the second part of the this paper are specifications-driven proposals, while the last one, although complex and hard to implement, could be quite easily understood and used by business users and modelers.