Supporting Multi-layer Modeling in BPMN Collaborations

Flavio Corradini, Andrea Polini, Barbara Re, Lorenzo Rossi(✉), and Francesco Tiezzi

School of Science and Technology, University of Camerino, Camerino, Italy
{flavio.corradini,andrea.polini,barbara.re,
lorenzo.rossi,francesco.tiezzi}@unicam.it

Abstract. In recent years, BPMN has acquired a clear predominance among the notations for modeling business processes. This is mainly due to its capability to close the communication gap between business and IT people. As a consequence, the quality of produced models is more and more important and, among the others, understandability plays a relevant role to permit to properly convey information in such a heterogeneous context. To improve understandability, it is generally recommended to not overwhelm models with to many details, and to use instead sub-process modeling elements to split collaborations into layers. However, the BPMN standard does not provide precise specifications on how the details, hidden at the given layer, should be included in the model, in particular considering message exchange. In particular, the consistency checking between collapsed sub-processes and their detailed representation is left to the modeler, and there is not much support to help him/her in this activity. In this paper, we analyze BPMN modeling tools with respect to their actual capabilities to support multi-layer collaborations. From the analysis we observed a general lack of support in the modeling environment. Then we propose a design methodology providing a set of guidelines to ensure consistency in multi-layer collaborations. These guidelines have been implemented in a stand alone tool, which enables their automated checking in any BPMN modeling tool.

Keywords: BPMN · Modeling guidelines · Messages exchange · Sub-processes

1 Introduction

Business process modeling is an important activity in order to understand and reason on how the work is performed within an organization. In order to support process modeling, several notations have been proposed and are currently available. This paper focus on Business Process Modeling Notation (BPMN) [1], an OMG standard that nowadays is one of the most used notations both in academic and in industrial contexts. This success is mainly due to its capability to close the communication gap between business and IT people. Its wide usage

R. Pergl et al. (Eds.): EOMAS 2017, LNBIP 298, pp. 53–67, 2017.
DOI: 10.1007/978-3-319-68185-6_4

is also testified by the availability of more than 50 tools (for further details see www.bpmn.org) supporting the editing, and often other business process lifecycle phases (e.g., enactment and maintenance).

A largely used diagram of the notation is the *Collaboration* that, among the other aspects, permits to represent the message exchange between different participants collectively cooperating to reach specific goals. Involved participants in a collaboration diagram need to agree on the different aspects of the communication (message orders, message formats, etc.), so that they can effectively reach the objectives of the collaboration. In particular, the involved organizations will have to reconcile their internal processes to properly support the communication.

Collaboration diagrams can be fruitfully exploited for different purposes, that however can have contrasting needs. On the one hand, the diagram conveys relevant information for the involved stakeholders that need to understand and reasons on the impact of the collaboration on their organization. In general, this aspect, which relates to understandability of models, is favored when irrelevant details are hidden in the model and the dimension is kept to a manageable size. On the other hand, collaboration diagrams can be fruitfully exploited, given enough details, to set and deploy supporting software systems, applying for instance model-driven engineering techniques. The automatic derivation of software requires instead to include in the model a high degree of details.

Independently of the purpose of the models, their qualities must be ensured. In particular, to increase models understandability, modeling guidelines are proposed and used in practice. Among the others, it is recommended to split the collaboration into layers with a different focus on the process [2]. In order to do that, BPMN proposes sub-process elements to broke down a model from an abstract layer to a more detailed one (layer nesting is allowed). Indeed, in large and complex models sub-process elements are often used to abstract some part of the behaviour. In such a way it will be possible to achieve the desired trade off between the needs of understandability and precision. Nevertheless, modeling communications in collaboration diagrams could be tricky in multi-layer scenarios when sub-process elements are used.

The usage of multi-layer structures may lead to consistency issues concerning elements that do not represent the same concept in different layers. The OMG standard does not provide any detailed specification of what concerns this kind of situation, leading to modelling environments that behave differently. In particular, they do not support automatic consistency checking, leaving this cumbersome task to modelers. This is a major issue, considering that a manual check is obviously costly and error prone.

In this paper we want to give a solution to such an issue, supporting modelers to design consistent communications in multi-layer BPMN collaborations. More specifically, the contributions of the paper are:

- an analysis of the BPMN modeling tools regarding multi-layer consistency;
- a design methodology for ensuring multi-layer consistency;
- a stand alone tool for checking multi-layer consistency.

The rest of the paper is organized as follows. Section 2 discusses about multi-layer modeling approaches, and compares the most common BPMN modeling tools in terms of supporting mechanisms for multi-layer modeling and related consistency. Section 3 introduces the proposed methodology, providing the list of the defined guidelines. Moreover, it introduces the tool we developed. Section 4 presents most significant related works. Finally, Sect. 5 concludes by also touching upon directions for future work.

2 Multi-layer Modeling: Background Notions and State-of-the-Art

Modeling business processes is not a simple activity and several issues can arise. Business processes have to be considered in relation to size and complexity of the resulting models that in most of the cases need to be handled by introducing sub-process elements. In Sect. 2.1 we list the modeling approaches suitable for using sub-processes according to the OMG standard. Then, in Sect. 2.2, we show how modeling environments manage this kind of multi-layer modeling.

2.1 Multi-layer Modeling Approaches

According to the BPMN standard, expanded or collapsed sub-processes can be used [1]: (a) on the abstract layer (i.e., the main layer) when the sub-process is expanded, (b) on the abstract layer when the sub-process is collapsed using an independent model describing sub-process behaviour.

In practice, it is usually not recommended to use option (a) representing a sub-process in expanded form in the abstract layer, since collapsed sub-processes make the model more understandable. On one hand, the adoption of solution (a) presents issues related to consistency among layers, since all the elements are explicitly represented in the same diagram. Indeed, applying such an approach it is easy to see which task is sending or receiving the message and it is also possible to consider if each message is sent or received. On the other hand, keeping the sub-process expanded at a higher level makes the model and the working space more confusing.

Option (b), in which the sub-process is collapsed in the abstract layer and its specification is provided in an external model, can be a solution in term of modeling. Indeed, this approach solves the issues related to overcrowded models but it can pose issues in relation to the sub-process implementation and to the consistency of the message flows. For instance, inconsistency derive from a wrong naming or a missed specification of the same message in different layers.

The "correct" way of showing messages inside a sub-process is to include on the sub-process model the participants involved in the communication. Moreover, the messages exchange should be consistent among the different layers of the model. In this paper, we consider option (b) thanks to their ability to define multi-layer models giving the possibility to improve understandability.

We present now a multi-layer collaboration scenario, used as running example in the paper. We consider a BPMN collaboration combining the activities of three participants, A, B and C, organized into four layers (see Fig. 1). This example is intentionally kept simple, as it just aims at illustrating the main contributions of the paper. The abstract layer provides, in expanded way, each

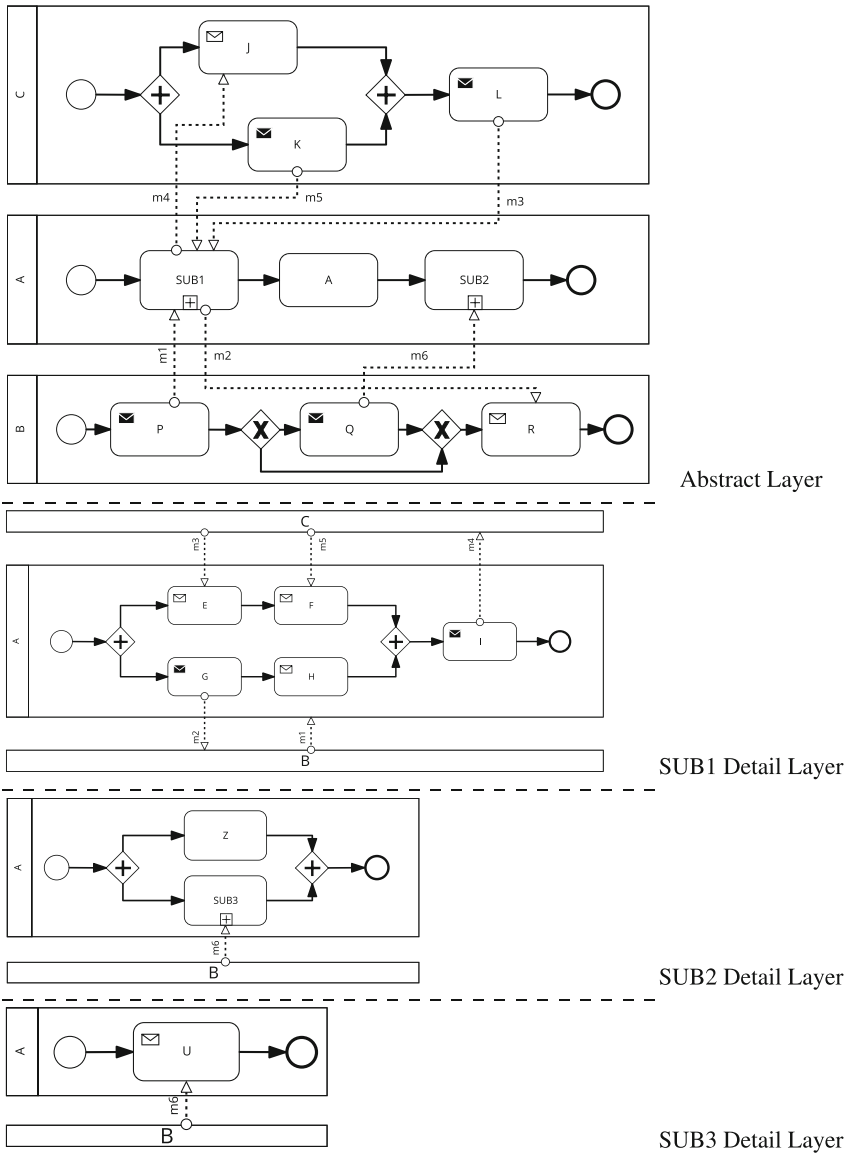

Fig. 1. Example of a multi-layer collaboration.

participant pool. This layer contains two sub-processes, *SUB1* and *SUB2*. Their specification processes are also provided, as well as the behavior of sub-process *SUB3* contained in *SUB2*.

2.2 Comparison of Modeling Environments

Considering multi-layer scenario, option (b), we made an assessment of 8 modeling environments widely used in order to check how they work in practice. In particular, the analysis of these tools relies on the features provided for modeling multi-layer collaborations. The analyzed tools are: ADOxx (www.adoxx.org), Aris Express (www.ariscommunity.com), Bizagi (www.bizagi.com), Camunda (www.camunda.com), Eclipse BPMN (www.eclipse.org/bpmn2-modeler), Magic Draw (www.nomagic.com), Signavio (www.signavio.com) and Visual Paradigm (www.visual-paradigm.com).

Assessment results are provided in Table 1. The table shows, for the abstract layer, if a modeling tool introduces constraints on the number of message flows linkable to the sub-process. It results that AdoXX, Bizagi and Signavio limit the number of message flows that the designer can attach to the sub-process while the other tools do not set any limitation. Table 1 also shows the complexity of modeling tools of linking the abstract layer with the lower layers, hence their suitability to support a multi-layer approach. In this respect, Camunda is the only one that denies this possibility. For this reason, we do not consider it further in the detailed layer analysis. For those modeling environments having the possibility to consider the detailed layer, we compare the tools by analyzing the type of process that can be used in it. The process type can be private, without the possibility to include pools and communication, or public. All the modeling environments refer to public model including pools and messages, except Bizagi. Differently from the other environments, Visual Paradigm gives the possibility to add in the detailed layer pools, tasks and gateways modeled in the abstract

Table 1. Modeling environments comparison.

	Abstract Layer		Detailed Layer	
	Message flow constraints	Multi-layer support	Process type	Consistency check
ADOxx	Yes	Yes	Public	No
Aris Express	No	Yes	Public	No
Bizagi	Yes	Yes	Private	No
Camunda	No	No	-	-
Eclipse BPMN	No	Yes	Public	No
Magic Draw	No	Yes	Public	No
Signavio	Yes	Yes	Public	No
Visual Paradigm	No	Yes	Public	No

layer. Moreover, the tool maintains consistent names and types for the elements, even if, this feature does not consider messages. Thus, consistency is not guaranteed. Finally, Table 1 presents the capability to perform consistency checking focusing on multi-layer communications. We observe that none of the considered tools enables multi-layer consistency (e.g., see the case of Signavio in Fig. 2).

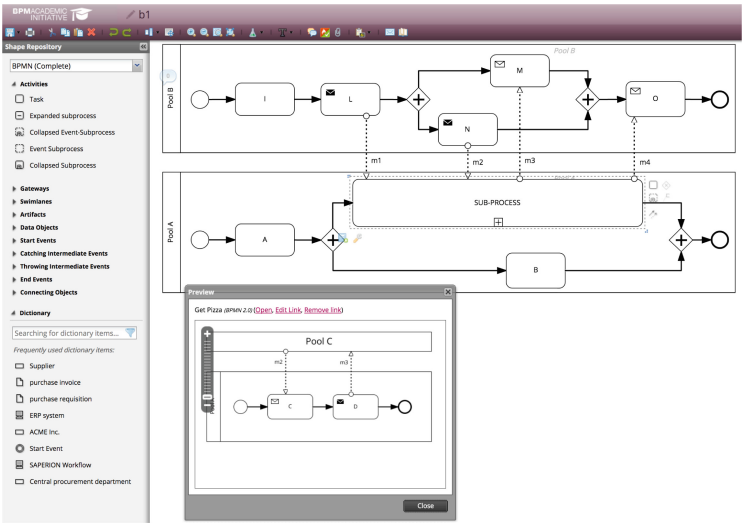

Fig. 2. Lack of consistency in signavio.

3 Methodology and Tool for Multi-layer Consistent Modeling

As we already said, the most effective modeling approach is the option (b), consisting of a collapsed sub-process in the abstract layer and its specification in the detailed one, to limit humans error in modeling. This allows to focus on details via external model view. We believe that in this approach, the designer should be assisted in modeling multi-layer collaboration, without the need to manually check consistency issues. Hence, what is expected from BPMN modeling environments is a reference practice in modeling multi-layer collaborations. Following, Sect. 3.1 discusses the proposed methodology and guidelines, and Sect. 3.2 introduces our consistency checking tool.

3.1 Design Methodology and Guidelines

We propose a top-down modeling approach in which the designer starts to model the abstract layer with collapsed sub-processes and then continue in the nested layers that will be linked to the abstract one. In terms of messages, the modeler should be allowed to attach more than one message flow to a sub-process element,

in order to specify the number of communication tasks in the lower layers. Hence, going deeper in the detailed layers the modeler needs to be assisted by providing all the participant pools, and the related message flows, so that errors are limited. Finally, layer by layer, the designer fill the model by adding elements in the pool containing the sub-process and linking the provided messages.

Here are the proposed modeling steps that we propose to be supported by modeling environments in order to assist the model designer. The following four steps have to be applied in an iterative way for each sub-process and recursively for each layer.

S1: *In the abstract layer, the collaboration is provided including all the involved pools. If the process requires a sub-process specification, the designer has to use the collapsed sub-process element and each message exchanged by this element has to be specified in the message flow (Fig. 3).*

S2: *In the abstract layer, collapsed sub-processes have to be linked to their specification by using an external model.*

S3: *For each detailed layer, the modeling environment automatically includes pools and messages to be consistent with the abstract layer. By default, messages have to be attached to the relative pools (Fig. 4).*

S4: *For each detailed layer, the designer has to refine the model detailing the behavior of the sub-process and attaching the message flows to elements within the specified pool.*

Afterwards, a consistency check is expected. This relies on eight guidelines, detailed in the following, able to guarantee consistency at each level of abstraction. Each guideline is a necessary condition for consistency, but they do not represent a complete set of rules to fully ensure this property. It is worth noticing that the multi-layer consistency checking based on these guidelines is performed in the syntactic definition of the collaboration model, without resorting to any formal definition of its semantics. In fact our guidelines have been derived by referring to the semi-formal semantics of BPMN provided by the standard specification [1].

G1: **Message Propagation.** *Each incoming/outgoing message flow attached to a collapsed sub-process has to appear in the relative detailed layer with the same label.*

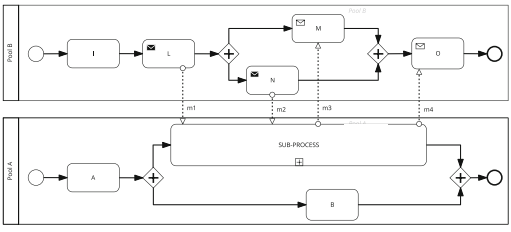

Fig. 3. Abstract Layer.

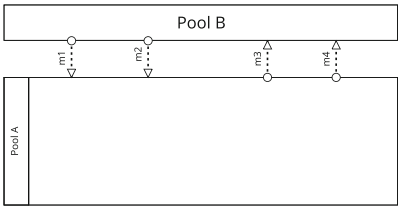

Fig. 4. SUB-PROCESS Detailed Layer - Step 3.

G2: **Message Link.** *Each incoming/outgoing message flow attached to a collapsed sub-process has to conclude its propagation in a message task or message event.*

G3: **Message Number.** *Further messages cannot be added to the ones in the abstract layer.*

G4: **Message Direction.** *Each message must keep the same sending and receiving participant in each layer.*

G5: **Message Ordering.** *For each couple of participants exchanging messages, if one of this performs a receive and after some steps a send, the other participant has to respect this order, by sending and than receiving the same messages independently from the layer in which are included (Fig. 5).*

G6: **Optional Message.** *For each couple of participants exchanging messages, each message sent by one of this participant in a non mandatory*[1] *branch has to be received by the other one in a non mandatory branch (Fig. 6).*

G7: **Mandatory Message.** *For each couple of participants exchanging messages, each message received by one of this participant in a mandatory branch has to be sent by the other one in a mandatory branch (Fig. 7).*

G8: **Looping Message.** *For each couple of participants exchanging messages, each message received by one of this participant in a loop branch has to be sent by the other one in a loop branch (Fig. 8).*

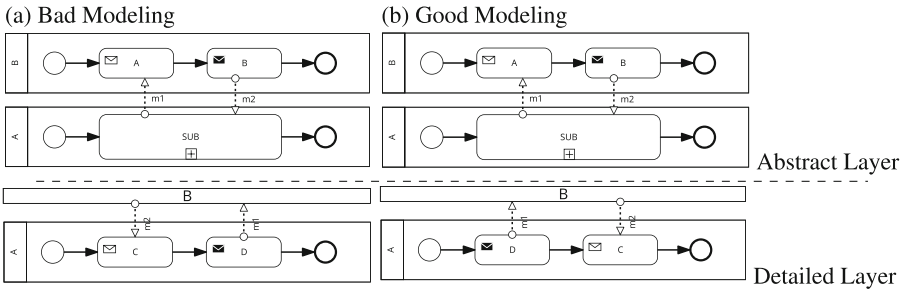

Fig. 5. Message Ordering example.

Considering the example we present in Sect. 2, here in the following we show the guidelines checking results. First of all we observe an issue referring to the message *m1* of the abstract layer that is linked in SUB1 to the border of the pool A. This is the result of the **Message Link - G2** check. We also underline the error of **Message Ordering - G5**. This regards to messages *m3* and *m4* following the order *m3 m4* in SUB1. Considering the abstract layer we can observe that the order is backward. Another problem impacts on message *m6*, in the abstract layer it is not mandatory, while we observe an issue referring to

[1] A non mandatory branch is a path of the process that starts from an exclusive, inclusive or event-based split gateway.

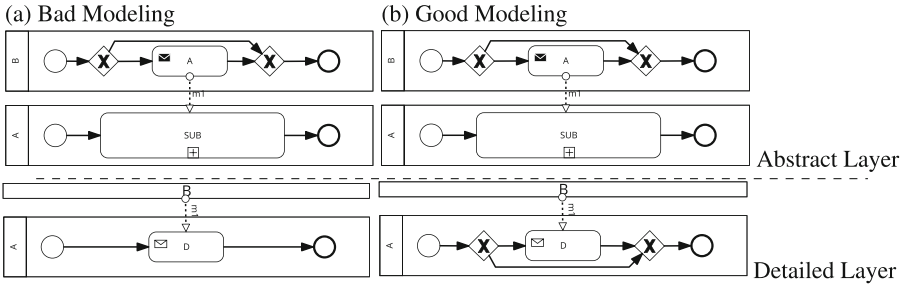

Fig. 6. Optional Message example.

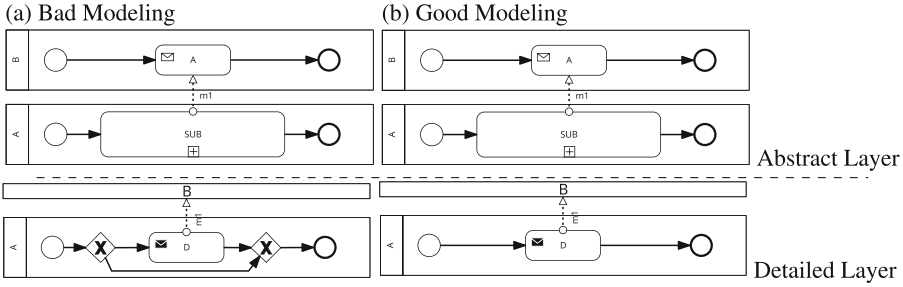

Fig. 7. Mandatory Message example.

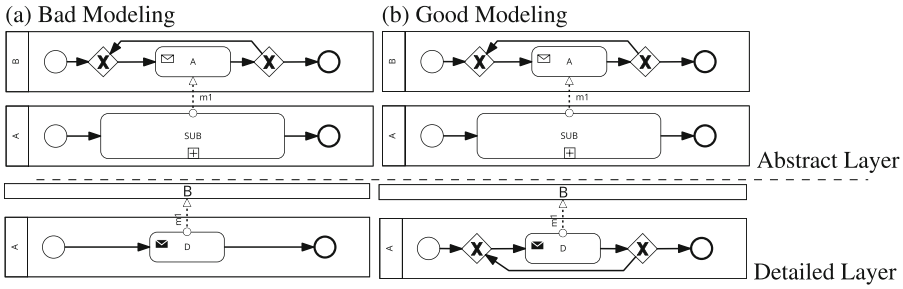

Fig. 8. Looping Message example.

the **Optional Message - G6** in SUB3 where the receiving task is mandatory. According to the suggested guidelines, in Fig. 9 we present the corrected multi-layer collaboration.

3.2 Consistency Checking Tool

In order to support the defined guidelines, we propose a *Java* based tool supporting designers in establishing whether their models are consistent. The tool is freely available[2]. It is independent from any modeling environment, hence can be used as an external service that can be integrated as a plug-in in other existing

[2] https://github.com/lorenzorossiunicam/ConsistencyChecking.

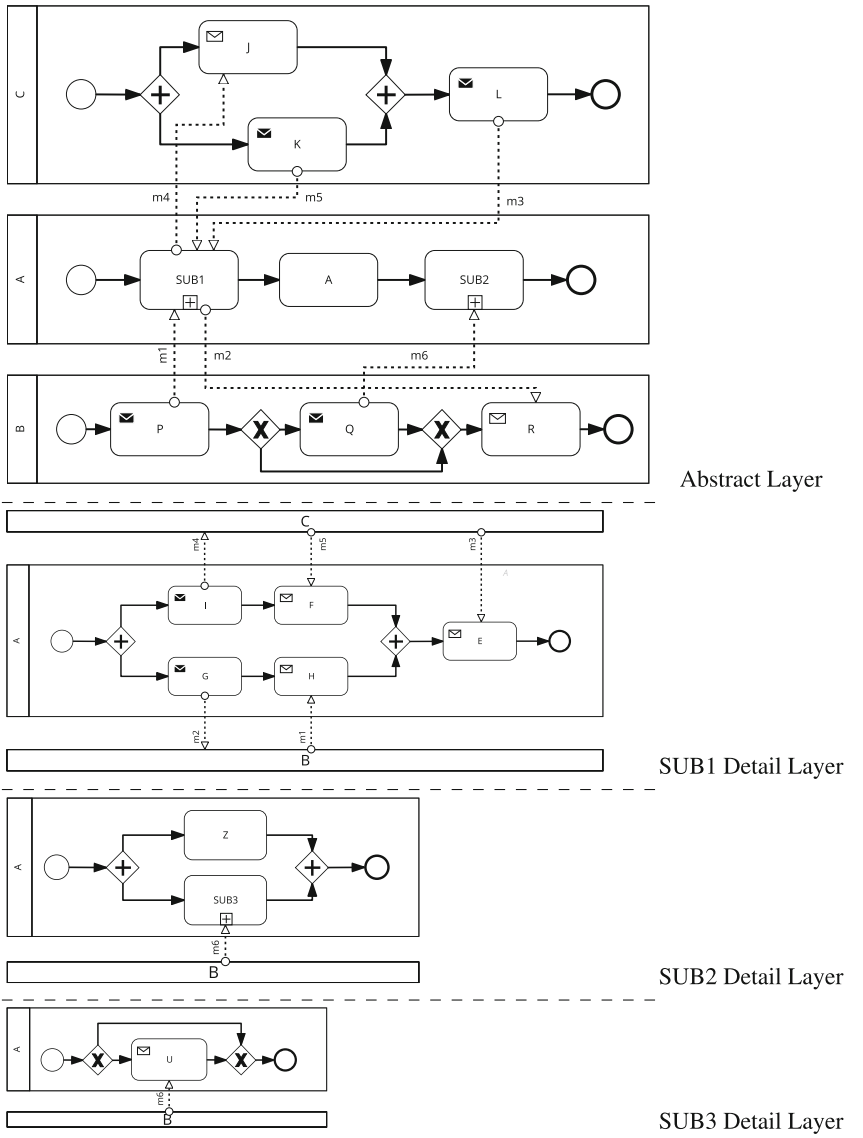

Fig. 9. Multi-layer collaboration (corrected version).

modeling tools, and eventually extended. The tool works with .bpmn files compliant with the OMG BPMN 2.0 standard. The input has to be provided as a set of BPMN models organized with a tree structure, in which the root represents the abstract layer and each leaf refers to a collapsed sub-process of the parent model. Tree structure provides hence a link from each collapsed sub-process to its process definition.

The consistency checking algorithm implemented in the tool, reported in Fig. 10, starts considering the abstract layer contained by the root of the dependency tree, then the other layers are analyzed using a depth first search navigation.

```
checkConsistency(Tree<Collaboration> tree){
    Collaboration root = tree.getRoot();
    for(Pool p : root.getPools())
        for(Lane lane : p.getLanes())
            lane.updateMessageSequence();
    for(CollapsedSubProcess sub : root){
        toCheck.addAll(sub.getMessages());
        msgInRoot = toCheck;
        goDeep(root.getChildrensOf(sub));
    }
    for (Message m : toCheck()){
        //Guideline 1 and 2 Violation
    }
    checkMessageSequences();
    //Guideline 5, 6, 7 and 8 violation
}

goDeep(Leaf l){
    for(Pool p : l.getPools())
        for(Lane lane : l.getLanes())
            lane.updateMessageSequence();
    for(Message m : l.getCollaboration().getMessages()){
        if(!msgInRoot.contains(m))
            //Guideline 3 Violation else
        if(isWrongDirection(m))
            //Guideline 4 Violation else
        if(isSourceOrTarget(m))
            toCheck.remove(m);
    }
    for(CollapsedSubProcess sub : l)
        goDeep(l);
}
```

Fig. 10. Consistency checking algorithm

At each step of the navigation the messages attached to a collapsed sub-process element are added in a global set of messages considering its name, the communicating pools and if the source or the destination have to be checked. In addition, to this, for each participant guidelines G5, G6, G7 and G8 are checked. Then, the procedure removes elements in this set if the missing source/destination is found in the correct layer, the remaining messages suggest errors of missed source/destination. At the same time, messages that are further connected to a sub-process are kept into the set. Otherwise, if new message names are found, the tool notices the absence of their definition in the root model.

Consistency checking has to be done quickly in order to be used in real contexts. The computational complexity of this algorithm clearly depends on the number of layers and messages that have to be checked. Given a collaboration split into L layers, in which are exchanged M messages, the algorithm visits each layer with a computational complexity derived by the depth first search. This complexity is $O(V + E)$, where V is the number of nodes and E the number of edges. In our dependency tree the nodes number is equal to the number of layers while the number of edges is equal to the number of node minus one. Hence, the visit complexity is $O(V + E) = O(L + L - 1) = O(2L) = O(L)$. In addition to this, in each layer the algorithm controls each message. The number of messages in each layer is, in the worst case, equal to the number of messages in the abstract layer. Consequently, the overall computational complexity of the algorithm is $O(L \times M)$.

4 Related Works

Multi-layer consistency, which has been identified in the literature as a relevant issue [3], is still an open field of study. There is a lack of works both in methodological and in formal approaches. Here we first refer to modeling guidelines used into practice and then to those approaches discussing formal verification to support consistency.

Regarding modeling guidelines, valuable contributions can be found in the literature published before the release of BPMN 2.0 [2,4,5]. These works focus on other graphical languages for business process modeling, but many recommendations can also be applied to BPMN 2.0. Regarding BPMN 2.0, a relevant work that specifically focuses on guidelines is provided by Silingas and Mileviciene [6]; the authors analyzed six BPMN models, and identified the *bad smells* – i.e., modeling approaches negatively impacting on model quality – that they contained. On the application of guidelines, an interesting contribution is given by Leopold et al. [7]. The authors focus on quality issues of 585 BPMN 2.0 process models from industry, highlighting which guidelines (collected from specific works, [8–10]) are not followed. Another relevant work is provided by [9], who suggests the use of an approach called *method and style* to help the model designers. More generally on process model quality, the most complete overview is given by de Oca et al. [11]. The authors collected 72 papers addressing different aspects of modeling quality, e.g., understandability, readability, maintainability, correctness, modularity, perceived ambiguity, perceived usefulness, completeness, etc. Starting from this review, the authors provided a set of 27 problems and unified quality guidelines [12]. Summing up, these contributions have a larger scope than ours, since they consider multiple quality attributes. However, our work provides a deeper insight for what concerns the messages exchange in multi-layer scenario. Another difference with our work is that most of the authors do not always suggest a way to verify the application of those guidelines, that is needed to automatically check if the model fits with the guidelines or not.

Regarding the formalization and verification of BPMN model consistency, the notion of sub-process and multi-layer specification has not been extensively studied yet. Among the others, Christiansen et al. [13], Corradini et al. [14], Falcioni et al. [15] El-Saber and Boronat [16] and, Borger and Thalheim [17] provide a direct formalization for a minimal subset of BPMN elements. Others contributions provide a mapping toward well known formal specifications (e.g., process algebras and petri-nets). In particular, Van Gorp and Dijkman define a formalization using visual transformation rules [18]. Differently, Kossak et al. present a sub-process semantics. The paper skips the problem of messages flow saying that "*semantics, however, does not change with the graphical depiction, that is, a collapsed sub-process must have the same semantics as when it is expanded*" [19]. Dijkman et al. propose a mapping from BPMN to Petri nets. The paper introduces also sub-processes saying that "*the behavior of such a process is however not clear in the BPMN specification*" [20]. It has been used in practice in different application domains [21]. Relevant is the work of Conforti et al. [22]. It aims to present a technique for multi-layer discovery of BPMN models without considering issues derived by messages exchange. These papers do not take into account sub-processes in terms of multi-layer structures, hence it is clear that sub-process semantics are developed without taking into account consistency in message exchange.

Finally, consistency is not a specific matter of business process modeling. There exists several research largely focuses on checking consistency of individual model and of relationships between pairs of models [23].

5 Conclusions and Future Works

In this paper we provide the results of an analysis we conducted on eight BPMN modeling tools regarding their capabilities to support multi-layer collaborations. We observe that most of the tools support the multi-layer modeling, some of them do not implement any consistency check. To solve such an issue, we provide a design methodology based on a set of eight consistency guidelines for multi-layer collaborations. Moreover, we develop a stand alone Java tool for checking the proposed guidelines.

As a future work, we plan to investigate more in detail the notion of compliance in order to give a wider set of guidelines suitable to ensure process models correctness by design. We also plan to extend modeling tools to implement the methodology and to integrate the checking tool in the design process [24]. Moreover, we aim to address the problem in a more formal way, by using and, if necessary, extending formal semantics of BPMN collaborations.

Acknowledgments. The authors would like to thank Elisa Ballini for her support in the benchmarking of modelling environments.

References

1. OMG: Business Process Model and Notation (BPMN V 2.0) (2011)
2. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7 pmg). Inf. Softw. Technol. **52**(2), 127–136 (2010)
3. Wong, P.Y.H., Gibbons, J.: A process semantics for BPMN. In: Liu, S., Maibaum, T., Araki, K. (eds.) ICFEM 2008. LNCS, vol. 5256, pp. 355–374. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88194-0_22
4. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75183-0_4
5. Mendling, J., Sanchez-Gonzalez, L., Garcia, F., La Rosa, M.: Thresholds for error probability measures of business process models. J. Syst. Softw. **85**(5), 1188–1197 (2012)
6. Silingas, D., Mileviciene, E.: Refactoring BPMN models: from 'Bad Smells' to best practices and patterns. In: BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Management Notation, p. 125 (2011)
7. Leopold, H., Mendling, J., Günther, O.: Learning from quality issues of BPMN models from industry. In: Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures, Vienna, Austria, 3–4 October 2016, pp. 36–39 (2016)
8. Allweyer, T.: BPMN 2.0 - Business Process Model and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung. Books on Demand (2009)
9. Silver, B.: BPMN method and style: with BPMN implementer's guide, 2 edn. (2011)
10. White, S.A.: BPMN modeling and reference guide: understanding and using BPMN. Future Strategies Inc. (2008)
11. de Oca, I.M.M., Snoeck, M., Reijers, H.A., Rodríguez-Morffi, A.: A systematic literature review of studies on business process modeling quality. Inf. Softw. Technol. **58**, 187–205 (2015)
12. Moreno-Montes de Oca, I., Snoeck, M.: Pragmatic guidelines for business process modeling. Technical Report 2592983, KU Leuven, Faculty of Economics and Business, November 2014
13. Christiansen, D.R., Carbone, M., Hildebrandt, T.: Formal semantics and implementation of BPMN 2.0 inclusive gateways. In: Bravetti, M., Bultan, T. (eds.) WS-FM 2010. LNCS, vol. 6551, pp. 146–160. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19589-1_10
14. Corradini, F., Polini, A., Re, B., Tiezzi, F.: An operational semantics of BPMN collaboration. In: Braga, C., Ölveczky, P.C. (eds.) FACS 2015. LNCS, vol. 9539, pp. 161–180. Springer, Cham (2016). doi:10.1007/978-3-319-28934-2_9
15. Falcioni, D., Polini, A., Polzonetti, A., Re, B.: Direct verification of BPMN processes through an optimized unfolding technique, pp. 179–188. IEEE, August 2012
16. El-Saber, N., Boronat, A.: BPMN formalization and verification using Maude. In: Proceedings of the 2014 Workshop on Behaviour Modelling-Foundations and Applications. BM-FA 2014, pp. 1:1–1:12. ACM, New York (2014)
17. Börger, E., Thalheim, B.: A method for verifiable and validatable business process modeling. In: Börger, E., Cisternino, A. (eds.) Advances in Software Engineering. LNCS, vol. 5316, pp. 59–115. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89762-0_3

18. Van Gorp, P., Dijkman, R.: A visual token-based formalization of BPMN 2.0 based on in-place transformations. Inf. Softw. Technol. **55**(2), 365–394 (2013)
19. Kossak, F., et al.: A rigorous semantics for BPMN 2.0 process diagrams. A Rigorous Semantics for BPMN 2.0 Process Diagrams, pp. 29–154. Springer, Cham (2014). doi:10.1007/978-3-319-09931-6_4
20. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
21. Corradini, F., Polini, A., Re, B.: Inter-organizational business process verification in public administration. Bus. Process Manage. J. **21**(5), 1040–1065 (2015)
22. Conforti, R., Dumas, M., García-Bañuelos, L., La Rosa, M.: BPMN miner. Inform. Syst. **56**(C), 284–303 (2016)
23. Sabetzadeh, M., Nejati, S., Liaskos, S., Easterbrook, S., Chechik, M.: Consistency checking of conceptual models via model merging. In: 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 221–230. IEEE (2007)
24. Flavio, C., Alberto, P., Barbara, R., Damiano, F.: An eclipse plug-in for formal verification of BPMN processes. In: 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service, pp. 144–149, June 2010