Composition rules for aspect-oriented process models with BPMN 2.0

Fernando Pinciroli[1][0000-0003-3270-5147]

[1] Universidad Nacional de San Juan, San Juan, Argentina
fernando.pinciroli@gmail.com

Abstract. The aspect-oriented paradigm poses a solution based on the separation of crosscutting concerns to the problems of scattering and tangling present in the code written with traditional approaches, including object-oriented programming. However, this solution can also be applied at all abstraction levels of the software development life cycle, including the business modeling phase. In this paper, a consistent set of rules for the composition of concerns for the business model is presented. These rules, designed using the BPMN 2.0 standard, seeks to ensure the correct composition of diagrams, preserving the syntax of the standard notation, to substantially improve the semantic integrity of the whole model and, consequently, to reduce the appearance of conflicts that normally arise when crosscutting concerns are composed at the join points.

Keywords: Crosscutting concerns, composition rules, business modeling.

1 Introduction

The aspect-oriented paradigm was developed to improve the modularity of computer systems by mainly solving the inconveniences of scattering and tangling that exist in traditional programming, including developments using the object-oriented approach [1]. The main solution provided is called “separation of concerns”.

Although the proposal was initially intended to be used at the programming phase, it is more than desirable that this separation of concerns could be carried out throughout all phases of the software development life cycle (SDLC) [2], affirmation supported by a systematic mapping study [3].

It has been a long time since business processes modeling began to take on an important role as the starting point of the SDLC [4], but this has been explicitly and strongly materialized with the appearance of the concept of Enterprise Architecture, which currently has numerous proposals and includes the business architecture as the first layer of its model [5][6].

The aspect-oriented paradigm requires that certain aspect-related activities be carried out in each SDLC abstraction level [7], which must also necessarily be performed in the business model. Along with them, bidirectional and end-to-end traceability among concerns must be maintained throughout the SDLC. The specific aspect-oriented activities are the detection, separation and encapsulation of the crosscutting concerns and

those activities necessary to reconstitute the complete models, including the identification of relationships among concerns and their subsequent composition, together with the detection and resolution of possible conflicts that could arise [8][9].

In this work a set of rules for the composition of crosscutting concerns for the business model is presented, applying the BPMN 2.0 standard [10], with the aim of providing a solution to the correct composition of crosscutting concerns from a syntactic point of view (strictly adhering to the BPMN 2.0 standard), which also leads to an improvement from a semantic point of view, with the consequent gain in terms of the reduction of possible conflicts.

In section 2 of this work, the objective sought when using composition rules is explained; in section 3, the composition rules themselves are presented, with some examples; in section 4, an example case is offered to demonstrate how to compose a base diagram with crosscutting concerns, following the syntax included in the set of rules, what allows obtaining a syntactically correct model; section 5 mentions the validations that were conducted to test the relevance and applicability of the rules; and, finally, conclusions and future work are presented in section 6.

2 What the composition rules offer

The activities of detection and separation of crosscutting concerns in business models culminate in the extraction of portions of processes (crosscutting concerns) from the original processes (base processes). This can be performed with aspect mining techniques [11][12][13], by detecting this kind of concern in already elaborated processes. Another alternative is to detect and to model the crosscutting concerns separately while the base processes are produced. In any of the cases, it is necessary to use some mechanism that allows to indicate the point of the base processes where the crosscutting concerns should later be composed and under what conditions this should happen. In the aspect-oriented paradigm, the insertion point is called “join point”, the portion of behavior to insert is named “advice”, these advices are encapsulated into an “aspect”, and the criteria to indicate how to compose is the “pointcut”.

There are different proposals to model crosscutting concerns, join points, pointcuts and advices at the abstraction level of business modeling. In this work, we have used the approach proposed in AOP4ST [14]. However, AO4BPMN can also be partially applied [15]. Another advantage that AOP4ST has over AO4BPMN is that it offers the possibility to designate join points implicitly, in addition to the explicit designation provided by both proposals [16].

The composition of the advices in the base processes is achieved by integrating them in the join points indicated by the pointcuts. As a result of this, we will move from an aspect-oriented business model to a conventional business model.

On the one hand, the aspect-oriented models, which have separate concerns from the base models, offer clean business process views for business domain experts and other specialized diagrams for those who must handle the specific crosscutting concerns. On the other hand, the resulting composed diagrams allow a fully integrated panorama to

those who want to ensure that the composition was correctly done, as well as being a tool to carry out full model validation with all stakeholders.

The way to obtain the composed diagrams, which we will consider here, is achieved by using a static composition technique at design time, although there are studies that describe dynamic composition at run time [2], which can provide a greater flexibility.

The correct composition of concerns in the base processes is imperative for the following reasons:

- From a semantic point of view: because it ensures that the processes describe what the process analyst tried to express with the different diagrams.
- From a syntactic point of view: because, in addition to respecting the BPMN 2.0 notation, a syntactically correct modeling will also contribute to a better semantics of the diagrams.
- To avoid possible conflicts: handling the concerns separately and applying them independently of each other in the base processes could cause unwanted situations (contradictions, redundancies, inconsistencies, null effects, etc.). The correct composition from the syntactic and semantic points of view allows to mitigate these possible conflicts.

3 The set of composition rules

A set of composition rules was developed to reach the goal mentioned in the previous section. These rules were syntactically validated one by one to ensure their validity and adherence to the BPMN 2.0 standard [16]. They were separated into categories according to the type of join point and are presented in the following subsections.

3.1 Rules for the "join point" element

Rule #1: Only activity, event, gateway, process and data elements can be join points.
Rule #2: The chronological order of composition is established by the type of advice: first the "before" type advices and the previous part of the "around" advices are composed; next, the elements represented by the "Join point" element are placed; finally, the "after" advices and the last part of the "around" advices are composed.

In the next figures an example of a base diagram, three crosscutting concerns that must be integrated to the base process according to what is defined in each pointcut, alongside the resulting composed process are presented. Each figure offers:

- A base process, without composing, which contains a join point with three pointcuts (Fig. 1).
- Three encapsulated processes, one for each advice to which the three pointcuts refer and that belong to different types: "before", "after" and "around" (Fig. 2, Fig. 3 and Fig. 4 respectively).
- The final result of the process obtained after the composition of the advices in the base process (Fig. 5).

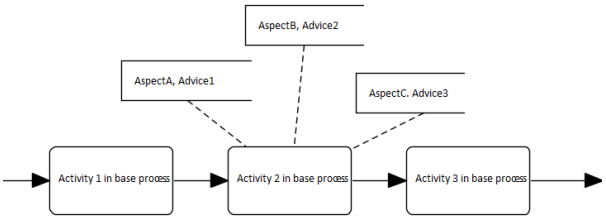

Fig. 1. Join point with three pointcuts

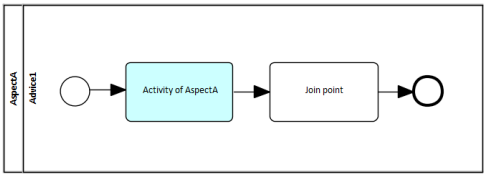

Fig. 2. "Before" type advice.

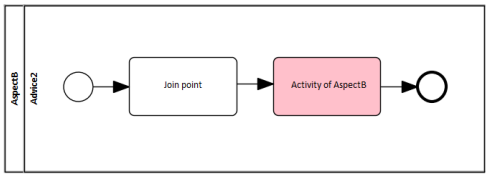

Fig. 3. "After" type advice.

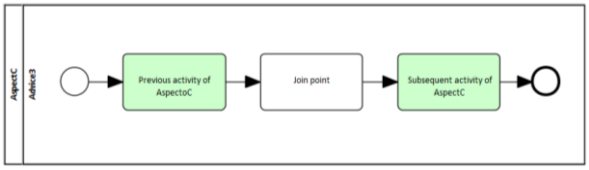

Fig. 4. "Around" type advice.

The composed diagram resulting from the process should look as follows:

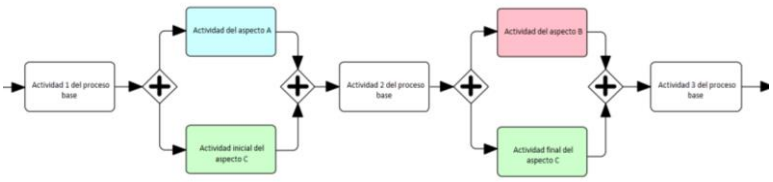

Fig. 5. Diagram resulting from the composition of the three advices at the join point.

The diagram in Fig. 5 shows all the advices integrated in parallel, since in the pointcuts the order number was not indicated (by default, order is 1). Also, all the composed activities will be executed, because a condition was not included in the pointcut; in case of having existed (see pointcut in Fig. 6), an exclusive gateway would have been allocated in the diagram in order to control the execution of the conditional activity. The syntax used to describe the pointcuts can be found in [15].

Rule #3: The advices and parts of advice that must be composed prior to the join point have to be placed according to the chronological order as indicated for the pointcuts: by default, they will be considered to be concurrent, but if a number is indicated, it will be taken into account to establish the order; identical numbers imply concurrency.

Given the base diagram of Fig. 6, Fig. 7 shows the diagram resulting from the composition of the advices of Fig. 2 and Fig. 4 in the base diagram, since one of the pointcuts has a condition and there is an explicit order of precedence in both pointcuts.

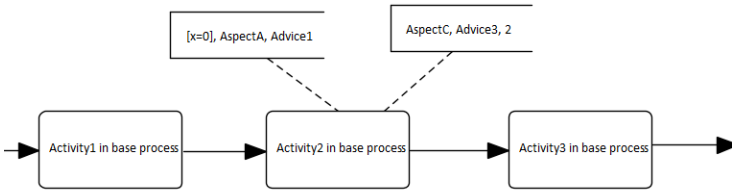

Fig. 6. Process with conditional pointcuts and order of precedence.

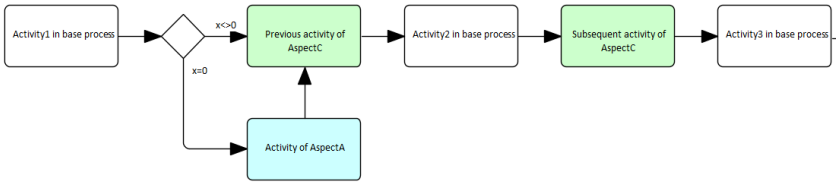

Fig. 7. Diagram resulting from the composition of the two advices at the join point.

Note in the resulting diagram (Fig. 7) that “Advice1” will be executed just in case the condition of its pointcut is met, while the other advice will always be executed, because is unconditional. In addition, if both advices are executed, they will do so according to the order of precedence indicated in their respective pointcuts (by default, order=1 is indicated for “Advice1”).

3.2 Rules for the “activity” element:

Rule #4: Advices applied to transaction type activities or to elements within a transaction will suffer the same fate as any other element of the process in case the transaction is not completed correctly.

Rule #5: Advices applied to sub-processes are integrated before, during or after the sub-process, according to the type of advice.
Rule #6: Advices applied to event sub-processes will be executed only for those threads whose event fires.
Rule #7: Advices applied to reusable sub-processes should be considered a constituent part of these sub-processes.
Rule #8: Advices applied to activities with a loop marker act the same way as in simple activities, since at the end of the activity a single token will continue.
Rule #9: Advices applied to activities with a parallel or sequential multi-instance marker should consider that, at the entrance of the activity, they will be executed only once and, upon exit, as many times as instances are generated in the activity.

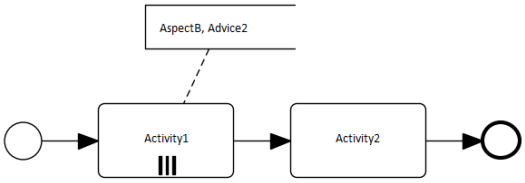

Fig. 8. Multi-instance process.

In the previous diagram, an "after" type advice will be executed as many times as instances are generated in "Activity1", which is initially reached by a single token.

Rule #10: Advices applied to an activity with a compensation marker behave as in a regular activity.

3.3 Rules for the "event" element:

Rule #11: Advices applied to a compensation event attached to an activity should be considered as if they were applied to the corresponding compensation activity.
Rule #12: In an activity with an attached interruptible event, the advices prior to the activity will always be executed, but those after it will not be executed if the activity is interrupted. This circumstance must be analyzed in each particular case to define how to consider the final portions of an advice "around" in case of interruption.

In the diagram of Fig. 9, it can be noted that "Advice3" is applied in a join point that has an attached interruptible event, that is "Delay in shipping" event. The resulting composition is presented in Fig. 10.

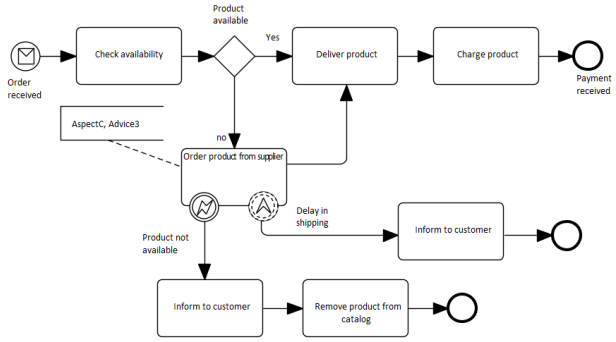

Fig. 9. Join point with an attached interruptible event.

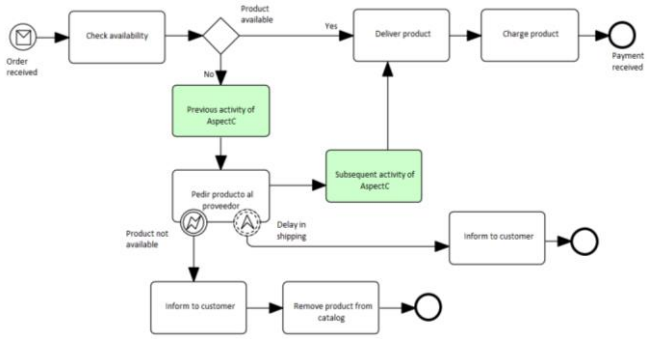

Fig. 10. Diagram resulting from the composition on an attached interruptible event.

In Fig. 10, if the join point activity is completed normally, the final activity of the advice "around" will be executed. However, if the attached interruptible event is fired, the flow will exit from the activity and the final part of the "around" advice will no longer be executed. This happens because the advice applies to the join point that, in this case, is the activity and not the attached event.

This situation must be considered to determine the solution to be chosen, which could be:

- Leaving the model as it is: it is correct for the business process that only the first part of the advice can be executed just in case of interruption of the activity;
- The advice must be completely executed in anyway: for this reason, it will be necessary to also add an "after" type pointcut on the attached event ("Product not available" event in the example in Fig. 10) that inserts the functionality of the second part of the original advice to be fired in case of interruption of the activity;
- Undo what has been done: in case one needs to ensure the complete execution of the whole advice, it will be necessary to include a compensation that undoes what was done by the first part of the advice before the triggering of the "Product not available" event.

Rule #13: In an activity with an attached non-interruptible event, all the advices applied to the join point will always be executed, because the join point is the activity and its execution will not be affected by the firing of the non-interruptible event. At most, it should be analyzed if the subsequent advices –the "after" type advices and the final part of the "around" advices– also have to be executed in the thread exiting the attached event.

The join point in Fig. 9 has a non-interruptible event attached to it. This means that, in case that event is triggered and the activity in which the event is located is also completed normally, the final part of "Advice 3" will be executed only once, because the pointcut is applied to the activity. However, it should be analyzed whether it is necessary to execute it at the exit of the interruptible event. If so, proceed as explained for the case of the attached interruptible event of Rule #12.

Rule #14: The join point is a single element, then, in activities with an attached event, the activity itself and the event must be considered as different elements, so the pointcut exclusively affects the element to which it is applied.

In Fig. 9, the pointcut applies only to the activity, which is the join point. The two attached events to the activity are different elements and they are not considered part of the join point. If necessary, they could have their own associated pointcuts, as mentioned among the possible solutions of Rule #12 and Rule #13.

Rule #15: The existence of a process branch with a termination event should force us to analyze the situation of the possible advices in execution, or to be executed, that should be stopped if a token reaches that event.

The BPMN termination event completely stops the process when the flow reaches it, regardless of what is happening in any of the other branches of the process that may be running. This means that, when there is a termination event, the resulting diagram with the composition of all advices should be drawn up after analyzing the possible impact of the process termination at any time, once the termination event is reached.

Rule #16: Link events cannot be join points, since a pair of link events equal a sequence flow arrow, which is not a valid join point according to Rule #1.

Rule #17: It should be noted that an "after" advice or the final part of an "around" advice is never executed in a final event or in a termination event.

Rule #18: Never assign a "before" advice or the initial part of an "around" advice in an initial event.

Rule #19: Attached events can only have "after" type advices.

3.4 Rules for the "gateway" element:

Rule #20: The "after" advices or the final parts of "around" advices on an exclusive gateway will apply to all output branches of the gateway, but will be executed only once, on the output branch that fires.

Rule #21: The "after" advices or the final parts of "around" advices on a parallel gateway will be applied and executed on all the output branches of the gateway.

Rule #22: The “after” advices or the final parts of the “around” advices on an inclusive gateway will be applied to all the output branches of the gateway that are activated, although it cannot be known, at design time, how many of them will be.
Rule #23: Advices applied on a complex gateway must be analyzed based on how the logic designed for the gateway is.
Rule #24: The “after” advices or the final parts of the “around” advices on an event-based gateway must be applied on all the output branches of the gateway and after the event on each branch but will only be executed on the branch that fires.
Rule #25: The “after” advices or the final parts of the “around” advices on a parallel event-based gateway must be applied in all the exit branches of the gateway and after the event in each branch, and they will be executed in the branches of all events that are fired.
Rule #26: It should be noted that a “before” advice is never executed on an instance-generating event-based gateway.
Rule #27: The events of an event-based gateway can be join points, regardless of whether the gateway is an instance generator or not, and its advices have execution priority over those of the gateway.

3.5 Rules for the “process” element:

Rule #28: The “after” advices and the initial part of the “around” advices applied to a process element are the first activities to be executed, while the “after” advices and the final parts of the “around” advices are the last ones.
Rule #29: The advices applied to an activity with an ad hoc marker, which includes various internal activities, applies only to the entry and exit of the ad hoc activity.

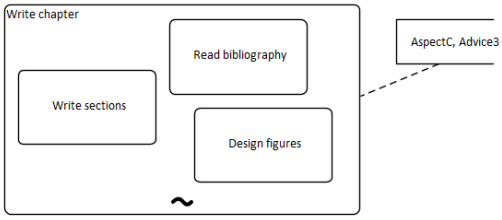

Fig. 11. Ad hoc activity as join point

In the previous example, an “around” advice was applied to an ad hoc activity, whose initial and final parts will be respectively composed before and after that activity, but which will not be applied to internal activities.

3.6 Rules for the “data” element:

Rule #30: Advices applied to data items are executed immediately before storage or after information retrieval, depending on the type of advice applied.

Clearly, the process encapsulated in the advice will not necessarily be as simple as it was proposed in the given examples; they could have an amount and a combination of flow elements that will have to be carefully integrated when composing.

It is important to work very carefully with the sequence flow connectors since the other two types of connectors (associations and messages) will not be affected by the advices. These types of associations will remain unaltered with respect to the elements to which they apply, since the advices are self-contained.

4 The set of composition rules

In this section we present a case of application of some of the composition rules in the surgical process of a hospital. Fig. 12 represents an end-to-end process, from the request for a surgery to the discharge of the patient after the intervention. The figure shows the explicitly indicated join points [16] through pointcuts that use the BPMN annotation element and a syntax that corresponds to the one proposed in AOP4ST [14]. An example of implicit join point designation by composing the advice is also shown in Fig. 15 [16].

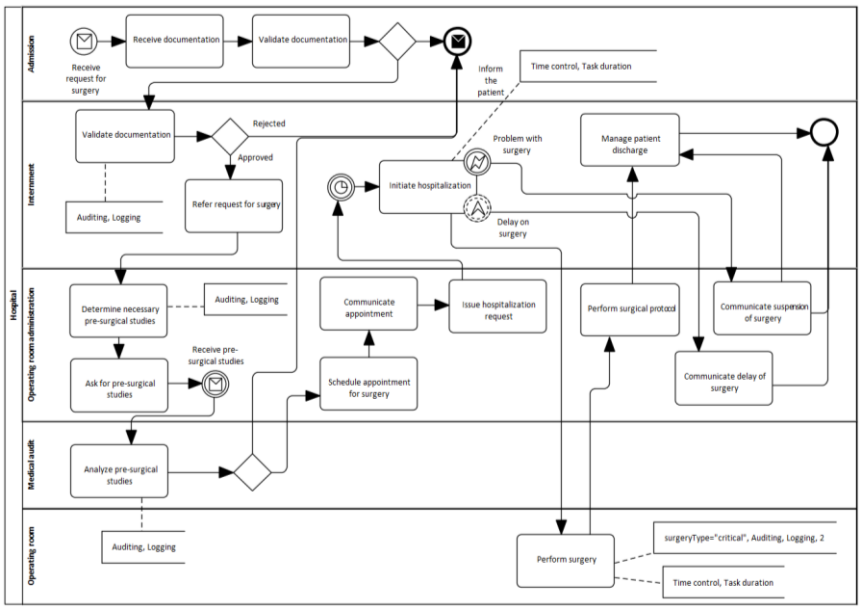

Fig. 12. Base process with explicit join point designation.

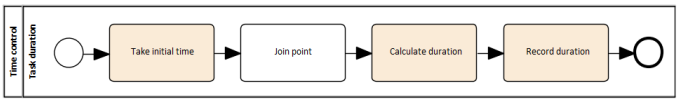

Fig. 13. Concern encapsulating an “Around” type advice.

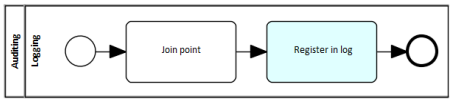

Fig. 14. Concern encapsulating an “After” type advice.

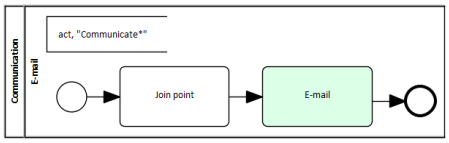

Fig. 15. Concern encapsulating an advice that designates join points implicitly.

The two advices shown in Fig. 13 and Fig. 14 are composed following the information in the pointcuts attached to the join points. In the case of the “Initiate hospitalization”, the subsequent part of the “around” advice is executed just in case the activity is successfully completed, in accordance with Rule #12, Rule #13 and Rule #14. If the edge-mounted “Problem with surgery” event is fired, this part of the advice will not be executed, since it would make no sense to register the duration of an incomplete activity. On the contrary, the uninterruptible “Delay on surgery” will not affect the complete execution of the advice (see Fig. 16).

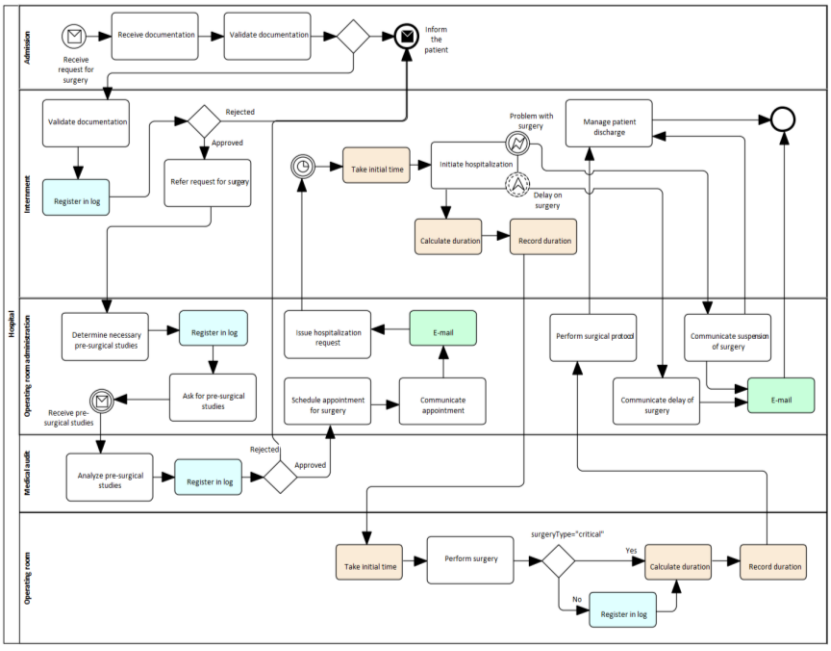

Fig. 16. Resulting process after composition.

In order to compose the advice in the join point activity “Perform surgery”, the chronological order was established following Rule #2 and Rule #3 and from the information offered by the corresponding pointcuts.

Note in the previous diagram, that two activities are composed with the advice of Fig. 15, which uses implicit join point designation, as indicated by the pointcut included in the crosscutting concern.

5 Validation

These composition rules were empirically validated in the research project “Aspect-oriented business process modeling”, which it was carried out at Champagnat University [18], where syntactically well-formed diagrams have always been obtained by using this set of rules. However, it is expected to continue to validate these rules with new cases from real-world setting as they arise.

Within this project and others that followed, three validations were carried out with its implementation in the industry. The first one consisted of the modeling of the corporate processes of a multinational security company, where the concerns of the business model of the “Surveillance” area were detected, separated, encapsulated and composed in the processes belonging to three countries where the company is located: Spain, Brazil and Argentina.

The second case corresponded to the modeling of the processes of a Directorate belonging to the Government of the Province of Mendoza, Argentina. Again, the detection, separation, encapsulation, and subsequent composition of the business processes were successfully carried out.

The third project consisted of the modeling of the internal processes of one of the two largest bank transaction management companies in Argentina, a project that is still ongoing.

With these three experiences, the following conclusions were reached:

- The use of these techniques did not interfere with the achievement of the specific objectives sought with the business model [19].
- Not only did they not interfere, but they also provided a better organization of the models and their identification of the primary and support processes.
- They allowed the possibility that different experts, in the domain of the problem and in technical matters, could validate the models corresponding to their respective areas of knowledge.
- The crosscutting concerns detected in the first experiences could be reused in the following ones.
- The composition rules were sufficient and allowed to correctly compose the process diagrams in BPMN 2.0.
- Although the process analysts had to be trained, there were no major drawbacks, due to the simplicity of the approach and the application of standard concepts, which were already known to the analysts.
- It was possible to compose the BMPN diagrams automatically using the Enterprise Architect tool, widely used in the industry. To this purpose, a plug-in for the tool

was developed and, in all cases, syntactically well-formed diagrams were obtained according to the BPMN 2.0 syntax.

6 Conclusions and future work

With the certainty that this group of rules is not yet complete, we consider that it is possible to continue developing new rules based on an exhaustive analysis of the semantics of the elements of BMMN 2.0 and the combination of them. Without a doubt, they can also be enriched with the analysis of different cases that may arise when model business processes from real-world settings after using this approach. In fact, this article presents 30 rules that expand the proposal that was originally made in 2016 [17], when at that time there were only 18.

Another improvement has been included to enhance the composed diagrams with concern and join point diagrams. They can be used as complementary tools to assist the business analysts with the analysis of composition results. The BPMN annotation elements that are used as instruments for the designation of the join points are the point of integration among the concerns, and it would be very easy to automatically develop the corresponding concern models, as shown in **¡Error! No se encuentra el origen de la referencia.**. In AOP4ST, all concerns (considered from the symmetric and the traditional asymmetric approaches [20]) are modeled as packages.

On the other hand, the join point model presents a greater detail of the relationships among the crosscutting concerns and the specific integration points in the base models. Undoubtedly, this model can also be automatically generated.

Finally, we understand that it is also possible to easily automate the composition of concerns with the criteria previously exposed by using different tools available in the market. For instance, this is already being carried out through the development of plug-ins that extend the functionality of the Enterprise Architect modeling tool, widely present in the industry.

References

1. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming, ECOOP, LNCS, no. 1241. Springer-Verlag, Finland (1997).
2. Jalali, A., Ouyang, C., Wohed, P., Johannesson, P.: Supporting aspect orientation in business process management. In: Software and Systems Modeling, 16(3), 903–925 (2017).
3. Pinciroli, F., Barros Justo, J.L., Forradellas, R.: Systematic Mapping Study: on the coverage of aspect-oriented methodologies for the early phases of the software development life cycle. In press in: Journal of King Saud University – Computer and Information Sciences, Elsevier (2020).
4. Basili, V., Briand, L. Bianculli, D., Nejati, S., Pastore, F., Sabetzadeh, M.: Software Engineering Research and Industry: A Symbiotic Relationship to Foster Impact. In: IEEE Software, 35(5), 44–49 (2018).
5. The Open Group TOGAF Homepage, https://www.opengroup.org/togaf, last accessed 2021/02/18.

6. Rouhani, B.D., ri Mahrin, M.N., Nikpay, F., Nikfard, P.: A Comparison Enterprise Architecture Implementation Methodologies. In: Proceedings of the 2013 International Conference on Informatics and Creative Multimedia, ICICM 2013, pp. 1–6, Kuala Lumpur (2013).
7. Alshareef, S.F., Maatuk, A.M., Abdelaziz, T.M.: Aspect-oriented requirements engineering: Approaches and techniques. In: Proceedings of the First International Conference on Data Science, E-learning and Information Systems, DATA '18, pp. 1–7 (2018).
8. Pryor, J., Marcos, C.: Solving Conflicts in Aspect-Oriented Applications. In: Proceedings of 4th Argentine Symposium in Software Engineering, Buenos Aires (2003).
9. Pryor, J.: Managing Conflicts in Aspect-Oriented Software. In: Proceedings of the V Workshop of Researchers in Computer Sciences, pp. 837-841, Tandil, (2003).
10. Object Management Group BPMN Homepage, https://www.omg.org/bpmn/, last accessed 2021/02/18.
11. Tonella, P., Ceccato, M.: Aspect mining through the formal concept analysis of execution traces. In: Proceedings of the 11th Working Conference on Reverse Engineering, pp. 112–121 (2004).
12. Breu, S., Krinke, J.: Aspect mining using event traces. In: Proceedings of the 19th IEEE international Conference on Automated Software Engineering, 32(9), 310–315 (2004).
13. Tourwe, T., Mens, K.: Mining aspectual views using formal concept analysis. In: Proceedings of the Fourth IEEE International Workshop on Source Code Analysis and Manipulation, pp. 97-106, Chicago (2004).
14. Pinciroli, F.: Aspect-oriented framework process in the early phases of the software development life cycle for a transition in the industry. Ph.D thesis. Universidad Nacional de San Juan (2020).
15. Charfi, A., Müller, H., Mezini, M.: Aspect-oriented business process modeling with AO4BPMN. In: Kühne T., Selic B., Gervais MP., Terrier F. (eds) Modelling Foundations and Applications, ECMFA 2010. LNCS, vol 6138, pp. 48–61. Springer, Heidelberg, (2010).
16. Pinciroli, F.: Explicit and implicit join point designation in aspect-oriented business modeling. In: Proceedings of the XLVI Latin American Computer Conference, CLEI 2020. Loja (2020).
17. Pinciroli, F.: Aspect-oriented business process composition rules in AOP4ST. In: Proceedings of the 35th International Conference of the Chilean Computer Science Society, pp. 1–6. Valparaíso (2016).
18. Pinciroli, F., Barros Justo, J.L., Forradellas, R.: Aspect-Oriented Business Process Modeling Approaches: An assessment of AOP4ST. In: Proceedings of the 46 Argentine Symposium on Software Engineerig, ASSE, JAIIO 2017, pp. 40-47 (2017).
19. Ould, M.: Business Processes: Modelling and Analysis for Re-Engineering and Improvement. Wiley (1995).
20. Bálik, J., Vranic, V.: Symmetric aspect-orientation: some practical consequences. In: Proceedings of the 2012 workshop on Next Generation Modularity Approaches for Requirements and Architecture, NEMARA '12, pp. 7–12. Postdam (2012).