When Language meets Language: Anti Patterns Resulting from Mixing Natural and Modeling Language

Fabian Pittke[1], Henrik Leopold[1], and Jan Mendling[1]

WU Vienna, Welthandelsplatz 1, A-1020 Vienna, Austria
fabian.pittke|henrik.leopold|jan.mendling@wu.ac.at

Abstract. Business process modeling has become an integral part of many organizations for documenting and redesigning complex organizational operations. However, the increasing size of process model repositories calls for automated quality assurance techniques. While many aspects such as formal and structural problems are well understood, there is only a limited understanding of semantic issues caused by natural language. One particularly severe problem arises when modelers employ natural language for expressing control-flow constructs such as gateways or loops. This may not only negatively affect the understandability of process models, but also the performance of analysis tools, which typically assume that process model elements do not encode control-flow related information in natural language. In this paper, we aim at increasing the current understanding of mixing natural and modeling language and therefore exploratively investigate three process model collections from practice. As a result, we identify a set of nine anti patterns for mixing natural and modeling language.

Keywords: Mixing of Natural Language and Modeling Language, Anti Patterns, Business Process Models

1 Introduction

Nowadays, business process modeling is an essential part of organizational design. Many organizations document their operations in an extensive way that involves several modelers and may result in more than thousand separate process models [1]. The increasing number of process models gives raise to automated quality assurance techniques since the consistency in such large-scale modeling initiatives can be hardly assured in a manual way [2]. Indeed, process models from practice often suffer from inconsistencies with respect to layout, level of detail, terminology, and labeling [2, 3, 4, 5].

Recognizing this, many techniques for automatically assuring the quality of process models have been introduced. There are techniques for checking structural properties such as deadlocks [6], techniques for checking the correctness of the data flow [7, 8], and techniques for automatically refactoring the model structure [9, 10].

Recently, also linguistic issues have been addressed. More specifically, available techniques recognize labeling styles [11, 4] and rework them according to desired naming conventions [12]. However, in particular semantic issues caused by natural language have not been investigated in much detail. As an example, consider the activity label *Consult expert and prepare report*. Apparently, this label contains two separate activities, i.e., *consult expert* and *prepare report*, which are linked by the conjunction *and*. The problem is that the execution semantics between these separate activities is not clearly defined. In fact, the word *and* could imply a parallel as well as a sequential execution. The reason for this confusion is the usage of natural language for expressing control-flow related aspects. Since natural language is often ambiguous, the precise intention of the modeler is not fully transparent to the reader.

In this paper, we investigate the problem of mixing natural and modeling language in process models. As there is, to the best of our knowledge, no research that addressed this problem, we take an explorative approach and manually analyze three process model collections from practice. Our contribution is a classification of anti patterns, which summarizes and groups cases in which natural language is used for expressing semantics of modeling language constructs. For each anti pattern, we identify possible interpretations and describe the characteristics in detail. The overall goal is to provide the knowledge base for automatically detecting, resolving and preventing these cases in the future.

The rest of the paper is structured accordingly. Section 2 illustrates the problem and discusses related work. Section 3 explains our methodology of approaching the problem. Section 4 presents the anti patterns we identified as well as an overview of their occurrence in the investigated model collections. Finally, Section 5 concludes the paper.

2 Problem Statement

In prior research, different aspects of process model quality have been addressed. In particular, structural and behavioral problems are well-understood and can automatically be resolved using different techniques. Structural problems refer to the elements of a process model and their interconnection. Available techniques can automatically transform unstructured process models into structured ones [10] or automatically detect deadlocks [6, 13]. Behavioral problems refer to control flow-related aspects of process models. Available techniques detect control-flow errors by using formal techniques [14] or check control-flow related properties of process models [15]. Additionally, the quality of natural language in process models has been addressed in current research. Existing techniques include, for instance, the refactoring of the activity label grammar [4] or the detection of ambiguous terminology [5]. In [16], the authors also investigate whether the natural language in activities violates the logic that is imposed by control flow splits. For example, an application cannot be rejected or accepted in the same process instance.

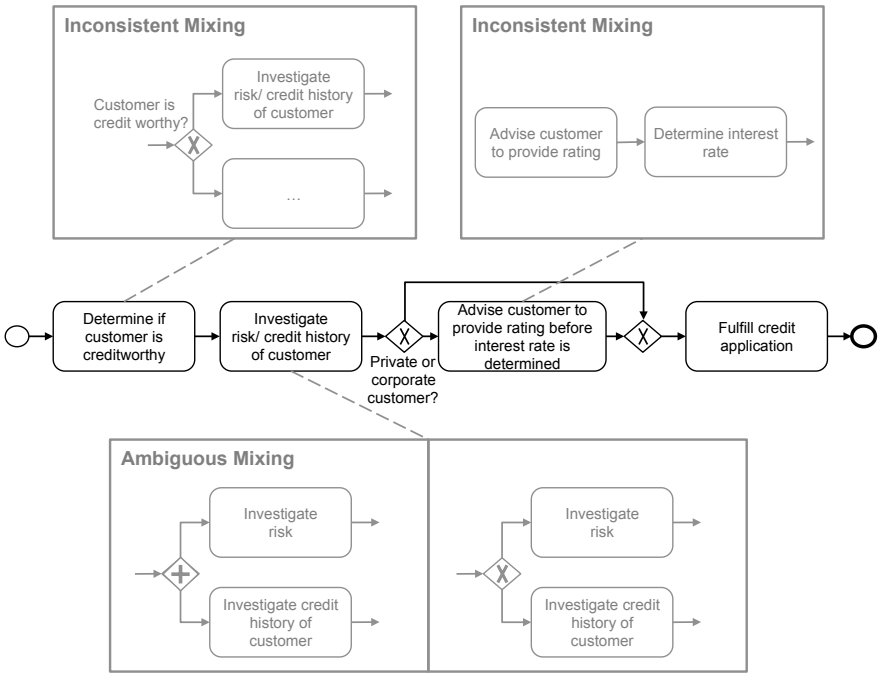


Fig. 1. Process with Elements Mixing Natural and Modeling Language

One aspect that has not been addressed in prior research are inconsistencies resulting from mixing natural language and modeling language in a single model element. Figure 1 illustrates a number of typical problems which occur when the natural language is used to express semantics that are supposed to be communicated with constructs from the modeling language. The figure shows a short process model from a bank representing a credit application. The process starts by determining credit worthiness of the customer. Therefore, possible risks as well as the credit history are analyzed. In the next step, it is determined whether the customer is a private or a corporate customer. Depending on the status, he is advised to provide a rating. Finally, the credit application is fulfilled.

The figure illustrates different cases of mixing natural language and modeling language. The activity *Determine if customer is credit worthy* requires the model reader to evaluate the credit worthiness. However, since this activity implies a decision, it would be more consistent to model it as a gateway. The activity *Advise customer to provide rating before interest rate is determined* uses the word *before* for implementing a sequence of activities. Here, it would be more consistent to model two separate activities. Although both cases represent an inconsistent mix of natural and modeling language, it has to be noted that their are not ambiguous, i.e., the intention of the modeler is clear. Nevertheless, there are also less understandable cases. For example, the activity *Investigate risk/ credit history of customer* is highly ambiguous. Although we know that the activity

involves an investigation of certain objects, it is unclear if the person is requested to investigate both the risk and the credit history or only one of these objects. The semantics of the slash symbol is simply not clearly defined and often used in different ways in practice.

The implications of mixing natural and modeling language are considerable. It may affect the ability of a reader to properly understand the model or to develop a solid understanding of the underlying process. Moreover, the performance of different analysis techniques might be affected. If the control flow semantics of a process are partially encoded using natural language, a structural check for deadlocks or other issues may erroneously evaluate the process as correct. Typically, such techniques simply assume that each activity contains a single piece of information that is not subject to additional conditions.

In order to address the problem of mixing natural and modeling language, techniques are needed that can detect and resolve affected process model elements. This, however, requires a precise understanding of these cases in the first place. So far, current research lacks a deeper understanding of such cases, their characteristics as well as their qualitative and quantitative extent. In order to close this gap, this paper investigates three model collections from practice. We use these collections to detect and classify linguistic anti patterns and to learn about their qualitative and quantitative extent. The overall goal is to provide the necessary knowledge for automatically detecting and resolving such cases in the future.

3 Research Design

The aim of this paper is to detect cases where natural language and modeling language are mixed. To achieve this goal, we adopt an explorative approach as conducted by Weber et al. for identifying refactoring opportunities in process models [9]. In particular, we perform an extensive manual analysis of industry process models to derive a list of generic anti patterns. Section 3.1 introduces our data set. Then, section 3.2 gives an overview of our analysis methodology.

3.1 Selection Criteria and Data Collection

In order to maximize the external validity of our results, we select process model collections that vary with respect to different dimensions such as modeling language, domain, and the degree of standardization. The characteristics of the selected process model collections are summarized in Table 1. Our data set includes:

- **SAP Reference Model Collection**: The SAP Reference Model Collection (SRM) captures the business processes of the SAP R/3 system in its version from the year 2000 [18][145-164]. It includes 604 Event-driven Process Chains with in total 2433 activities. Since the SRM is a reference model collection, it has a relatively high degree of standardization.

Table 1. Demographics of the Test Collections

Characteristic	SRM	IMC	AI
No. of Models	604	349	1,091
No. of Labels	2,433	1,840	8,339
Modeling Language	EPC	EPC	BPMN
Domain	Independent	Insurance	Academic Training
Standardization	High	Medium	Low

- **Insurance Model Collection**: The Insurance Model Collection (IMC) contains 349 EPCs dealing with the claims handling activities of a large insurance company. It includes a total of 1840 activities and is less standardized than the SRM as the models were created for internal purposes only.
- **AI Collection**: The models from the BPM Academic Initiative (AI) stem from academic training (see http://bpmai.org). The selected English subset includes 1,091 process models with in total 8,339 activity labels. As the model have been mainly created by students, we expect the lowest degree of standardization in this collection.

3.2 Data Analysis

To analyze the model collections, we choose an incremental approach that consists of two separate steps: anti pattern extraction and anti pattern classification.

In the *anti pattern extraction phase*, we manually scanned the process model collections for linguistic constructs implying control flow semantics. By independently analyzing the collections, we made sure that we did not miss relevant anti patterns and reduced the probability of biased results.

In the *anti pattern classification phase*, we analyzed each anti pattern in detail and derived possible interpretations. As a result, we received a set of nine anti patterns. Based on the number of possible interpretations, we classified each anti pattern as *inconsistent* or *ambiguous*. Inconsistent anti patterns mix natural and modeling language in an inconsistent way, but still have only one possible interpretation. Ambiguous anti patterns, by contrast, mix natural and modeling language in such a way that two or more interpretations are possible.

4 Findings

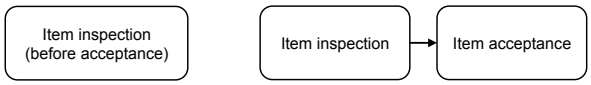
This section presents the findings of our explorative study. Section 4.1 presents the anti patterns which inconsistently mix natural and modeling language, but still only have a single interpretation. Section 4.2 introduces the anti patterns which are ambiguous and, hence, allow for multiple interpretations. In Section 4.3, we give an overview of the quantitative extent of the identified anti patterns.

4.1 Anti Patterns with Inconsistent Mixing

In the following, we introduce the anti patterns having a single interpretation. In total, there are four anti patterns: *Logical Extra Information*, *Iteration*, *Skip*, and *If Evaluation*.

Anti Pattern 1 (Logical Extra Information)

The *Logical Extra Information* anti pattern incorporates logical information into the label. This information imposes additional conditions on the task and, hence, has direct impact on the control flow. Typically, this anti pattern uses temporal prepositions such as *before* or *after* to clarify the order of activities. Figure 2 shows an example of this anti pattern and its corresponding consistent solution.

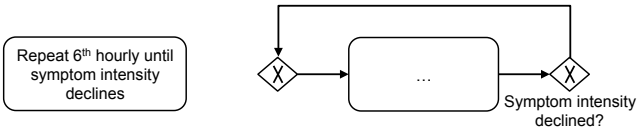


(a) Example (b) Consistent Solution

Fig. 2. Anti Pattern 1 (Logical Extra Information)

Anti Pattern 2 (Iteration)

The *Iteration* anti pattern is arranged in such a way that the natural language fragment asks for an iteration or a loop construct. In most of the cases, the iteration is expressed by the language pattern *repeat ... until* or a statement such as *per item*. In many cases, the label also contains the iteration condition. Figure 3 provides an example of this anti pattern and its corresponding consistent solution.



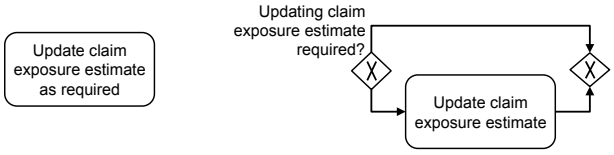
(a) Example (b) Consistent Solution

Fig. 3. Anti Pattern 2 (Iteration)

Anti Pattern 3 (Skip)

The activity of a *Skip* anti pattern generally implies a decision about an activity that must only be conducted under specific conditions. If the conditions are not

met, the activity is skipped and the process continues without executing this activity. Our analysis showed that activity labels that follow this anti pattern combine prepositions with adjectives or the past participle of the verb *to require*. Thus, examples include *if necessary*, *if required*, or *as required*. Figure 4 shows an example and its corresponding consistent solution.

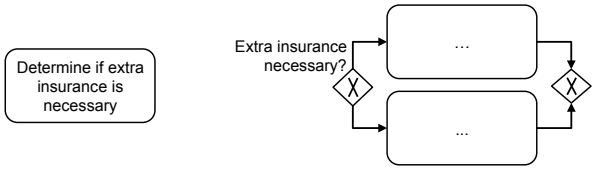


(a) Example (b) Consistent Solution

Fig. 4. Anti Pattern 3 (Skip)

Anti Pattern 4 (If Evaluation)

The *If Evaluation* anti pattern also implies a decision in the process flow. By contrast to the previously mentioned anti patterns, the If Evaluation anti pattern explicitly specifies the condition that has to be checked. In most cases, activities of this anti pattern contain a verb asking for the verification or investigation of certain conditions and the conditional word *if*. Examples of this anti pattern include *determine if*, *validate if*, *check if*, and *confirm if*. Figure 5 shows an example and its consistent solution.



(a) Example (b) Consistent Solution

Fig. 5. Anti Pattern 4 (If Evaluation)

4.2 Anti Patterns with Ambiguous Mixing

In the following, we introduce the anti patterns that allow for multiple interpretations. In total, they are 5 ambiguous anti patterns: *Wrong Label Class*, *Multiple Activities*, *Decision*, *Content-based Extra Information*, and *Temporal*

Extra Information.

Anti Pattern 5 (Wrong Label Class)

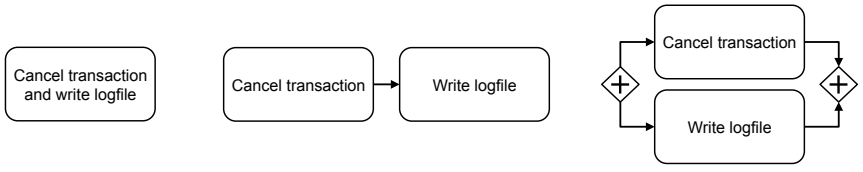
Process model elements suffering from the *Wrong Label Class* anti pattern erroneously combine labeling style and modeling construct, i.e., activity, event, or gateway. As an example, consider an activity that is labeled using an event label or vice versa. As a result, it remains unclear whether the natural language or the modeling language determines the meaning of the construct. As shown in Figure 6, the activity *Tick box invoice entered* might refer to the event *Tick box invoice entered* or to the activity *Enter a tick box invoice*.

(a) Example (b) Consistent Solutions

Fig. 6. Anti Pattern 5 (Wrong Label Class)

Anti Pattern 6 (Multiple Activities)

Activities suffering from the *Multiple Activities* anti pattern combine several actions, business objects, or combinations of these in a single activity element. Hence, a single activity element instructs people to perform multiple streams of action. Typically, this anti pattern includes the conjunction *and*, or special characters such as *+*, or *&*. The interpretation of this pattern is ambiguous. It may refer to a sequence of activities as well as to a parallel execution. Figure 7 illustrates this anti pattern using the activity *Cancel transaction and write logfile*.

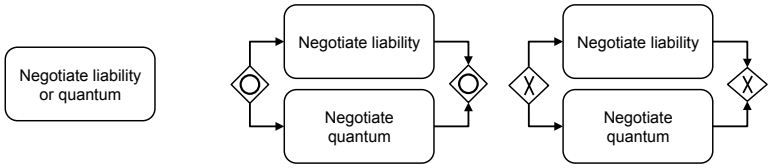


(a) Example (b) Consistent Solutions

Fig. 7. Anti Pattern 6 (Multiple Activities)

Anti Pattern 7 (Decision)

The *Decision* anti pattern implies a control flow split leading to several exclusive or inclusive streams of action. Similarly to the previous anti pattern, this anti pattern may use multiple actions, business objects, or combinations of these. Typically, this anti pattern occurs when two alternatives are linked with the conjunction *or*. Alternatively, the special character / may represent an indicator for this anti pattern. As shown by the activity *Negotiate liability or quantum* in Figure 8, we cannot infer whether this anti pattern expresses an exclusive or an inclusive decision.

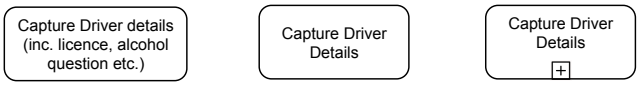


(a) Example (b) Consistent Solutions

Fig. 8. Anti Pattern 7 (Decision)

Anti Pattern 8 (Content-based Extra Information)

The *Content-based Extra Information* anti pattern refers to activities that ambiguously incorporate additional information into the label. This may include the specification of business objects or the refinement of entire activities. The most prominent examples are the use of brackets and the separation of information using a dash. As an example for this anti pattern, consider the activity *Capture Driver details (inc. licence, alcohol, questions etc.)* from Figure 9. Here, the business object *driver details* is further specified in brackets. However, the interpretation of this label is unclear. This anti pattern may refer to a single activity as well as to multiple activities in form of a subprocess that are specified elsewhere. Figure 9 illustrates the possible interpretations of this activity.

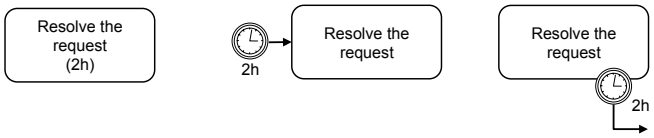


(a) Example (b) Consistent Solutions

Fig. 9. Anti Pattern 8 (Content-based Extra Information)

Anti Pattern 9 (Temporal Extra Information)

The *Temporal Extra Information* anti pattern is similar to the latter anti pattern. It, however, incorporates temporal instead of content-based information. This may include temporal prepositions that clarify the duration (e.g. in minutes, hours, or days) of an activity or other time-related constraints. Typically, the additional information is provided in brackets and, in many cases, unclear. The temporal information may represent waiting time, i.e., time that must pass before the process continues normally, or the temporal information could be interpreted in the sense of an attached intermediate event. The latter implies that the execution of the activity is canceled as soon as the time limit is reached.



(a) Example (b) Consistent Solution(s)

Fig. 10. Anti Pattern 9 (Temporal Extra Information)

4.3 Quantitative Findings

To get an impression of the quantitative extent of the previously introduced anti patterns, Table 2 gives an overview of the number of occurrences of the anti patterns for each investigated collection. The numbers reveal that all three collections particularly suffer from the anti patterns *Content-based Extra Information* and *Multiple Activities*. For the other patterns, we observe a more heterogeneous distribution. While the IMC and the AI collection also frequently suffer from the anti pattern *Wrong Label Class* and *If Evaluation*, the SRM collection is almost free from these issues. Reasons for such differences include a differing degree of standardization and a differing experience of the involved modelers. Especially, less experienced modelers may tend to use natural language for expressing more complex control-flow structures such as loops, skips, and decisions.

In conclusion, we can state that the phenomenon of mixing natural and modeling language can be frequently encountered in practice. A detailed investigation also revealed that models containing one anti pattern tend to include further anti patterns. As pointed out earlier, this can significantly affect the understandability of the models as well as the performance of automated analysis techniques. Hence, the introduced classification represents an important step towards automatically cleaning process model repositories from this quality issue and preventing it in the future.

Table 2. Anti Pattern Frequencies

	Anti Pattern	SRM	IMC	AI
Inconsistent Anti Patterns	AP 1 - Logical Extra Information	0	10	7
	AP 2 - Iteration	0	0	15
	AP 3 - Skip	0	49	0
	AP 4 - If Evaluation	0	156	70
Ambiguous Anti Patterns	AP 5 - Wrong Label Class	0	18	104
	AP 6 - Multiple Activities	217	329	606
	AP 7 - Decision	2	125	52
	AP 8 - Content-based Extra Information	285	63	112
	AP 9 - Temporal Extra Information	0	1	68

5 Conclusion

In this paper, we investigated the problem of using natural language for expressing modeling language constructs. We therefore manually analyzed three process model collections from practice in order to identify anti patterns. In total, we identified a set of nine different anti patterns, which we classified into *inconsistent* and *ambiguous* cases. While the latter category is particularly a problem for the understandability of humans, all these anti patterns may negatively affect the results of automated analysis techniques as they often assume that activities do not contain additional conditions encoded in natural language. A quantitative evaluation of the findings demonstrated that anti patterns for mixing natural and modeling language can be frequently found in process models from practice and that there is also a tendency for models to contain multiple anti patterns.

Altogether, this paper provides the foundations for automatically detecting and resolving issues related to mixing natural and modeling language. Furthermore, the identified anti patterns foster the creation of a canonical process model, i.e., a process model in which each activity refers to only one stream of action. Such a canonical process model would be highly beneficial for several process model analysis techniques as, for instance, the matching of activities [19], the matching of process models [20, 21], and the calculation of process behavior [22]. Against this background, it is our goal to implement a technique for detecting and resolving the identified anti patterns in the future. In addition, we plan to validate our anti pattern classification against additional process model collections.

References

1. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal **12**(2) (2006) 249–254
2. Becker, J., Rosemann, M., Uthmann, C.: Guidelines of Business Process Modeling. In: BPM'00. Springer, Berlin (2000) 30–49
3. Davis, R.: ARIS design platform: advanced process modelling and administration. Springer (2008)

4. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems **37**(5) (2012) 443–459
5. Pittke, F., Leopold, H., Mendling, J.: Spotting terminology deficiencies in process model repositories. In: Enterprise, Business-Process and Information Systems Modeling. (2013)
6. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models. Data & Knowledge Engineering **70**(5) (2011) 448–466
7. Sun, S., Zhao, J., Nunamaker, J., Liu Sheng, O.: Formulating the Data-Flow Perspective for Business Process Management. Information Systems Research **17**(4) (2006) 374–391
8. Sidorova, N., Stahl, C., Trcka, N.: Soundness Verification for Conceptual Workflow Nets with Data: Early Detection of Errors with the Most Precision Possible. Information Systems **36**(7) (2011) 1026–1043
9. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry **62**(5) (2011) 467–486
10. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. In: BPM'10. Springer (2010) 276–293
11. Leopold, H., Smirnov, S., Mendling, J.: Recognising activity labeling styles in business process models. Enterprise Modelling and Information Systems Architectures **6**(1) (2011) 16–29
12. Leopold, H., Eid-Sabbagh, R.H., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems **56** (2013) 310–325
13. Dehnert, J., Rittgen, P.: Relaxed soundness of business processes. In: Advanced Information Systems Engineering, Springer (2001) 157–170
14. van der Aalst, W.M.P.: Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In: BPM'00. Springer (2000) 161–183
15. van der Aalst, W., de Beer, H., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: OTM'05. Springer (2005) 130–147
16. Gruhn, V., Laue, R.: Detecting common errors in event-driven process chains by label analysis. Enterprise Modelling and Inf. Sys. Architectures **6**(1) (2011) 3–15
17. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features–enhancing flexibility in process-aware information systems. Data & knowledge engineering **66**(3) (2008) 438–466
18. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley (1998)
19. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Information Systems **36**(2) (2011) 498–516
20. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: BPM'09. (2009) 48–63
21. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R.M., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: BPM'12. (2012) 319–334
22. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. **37**(3) (2011) 410–429