Volker Gruhn, Ralf Laue

Detecting Common Errors in Event-Driven Process Chains by Label Analysis

In this article, we discuss several classes of error patterns that can frequently be found in Event-Driven Process Chains (EPC). Instances of these patterns can be detected by using a pattern-matching approach: The model is translated into a set of Prolog rules and potential modelling problems are located by querying the Prolog fact base for certain problem patterns. In particular, this article presents patterns for problems that can be detected by analysing the labels of events and functions in EPCs. To make reasoning about the contents of the labels possible, the labels are transformed into a normal form. By taking synonyms, antonyms and negating words (no, not) into account, we locate labels that contradict each other. This leads to the detection of some classes of errors like modelling of an event and its negation occurring at the same time. Our method has been applied to 1253 EPC models in German language. We have been able to detect a large number of errors in those models that remain undetected using traditional approaches for EPC validation.

1 Introduction

According to Esswein et al. (2004), two stances of semantics of a model must be separated: First, semantics includes the rules for automated interpretation of a model (for EPC models represented by rules defining how the state space of all possible executions is calculated). Second, each modelling construct represents some real-world entities which is another aspect of model semantics. In EPC models, this second aspect of semantics is expressed by the labels of functions and events.

Pfeiffer and Niehaves (2005) call the graphical arrangements of model elements and their connections among each other *model element structure*. The actual meaning of the modelling constructs in the application domain is called *terminological structure*. Pfeiffer and Niehaves stress that an essential part of the semantics of a conceptual model descents from a reference to the language of the application domain. Hence, model validation should take into account both aspects of semantics.

If we look at approaches that have been published for the validation of EPC model semantics (for example van der Aalst 1997, Rump 1999, and Mendling 2007), we realise that the majority of them focuses on the first aspect of semantics. They allow to find control-flow errors like deadlocks but abstract away from the actual meaning of the events and functions in EPCs.

Only a few authors (for example Awad et al. 2008a whose approach will be discussed in Sect. 5) tackled the problem of detecting modelling problems related to the actual meaning of function and event labels.

When reviewing real-world EPCs from various sources, we categorised some frequent error patterns that can only be detected by analysing the labels of functions and events. The aim of our work was to automate this analysis such that a modelling tool can instantly give feedback about the occurrence of a problem. This helps to improve the model quality at the time when a model is created.

Our method has been implemented into the open-source modelling tool *bflow* Toolbox*. Now this tool gives feedback not only about syntactical and control-flow problems as described in Gruhn et al. (2009), but also about contradictions found in the usage of labels.

The remainder of this paper is structured as follows: The error pattens are categorised in Sect. 2. In Sect. 3, we describe our algorithm for locating instances of those patterns in a model. The results of a validation of our approach, using 1253 EPC models from various sources, can be found in Sect. 4. In Sect. 5, we discuss the results and compare our approach with previously published methods.

2 Error Patterns

When analysing EPC models, we identified several error patterns that remain undetected by current control-flow analysis tools. Table 1 gives an overview about the error patterns.

While we use the term 'error patterns' for all patterns in this article, we have to stress that not every occurrence of such a pattern necessarily indicates an actual error in the model. The severity of the occurrence of the patterns is discussed in Tab. 1 and in the subsections of this section. Some patterns indicate actual modelling errors (for example pattern B and J, see below), while other patterns (for example patterns C and D) show a possibility to improve the readability of the model.

In general, the occurrence of a pattern alerts the modeller to double-checked the part of the model where the pattern has been found.

In this section, we discuss the patterns and give examples of the occurrence of those patterns in real-world models. For the purpose of this article, these model fragments have been translated from German into English language.

2.1 Pattern A: Identical Events Before a Join

In an EPC, the semantics of events that are directly connected to a join is that the events model *different* conditions that can occur during the execution of the business process. For this reason, two events with the same meaning both having an arc to the same join can be an indication of a modelling problem.

In some cases, the reason for the occurrence of this pattern is that the business process has been modelled in a wrong way. For example, in the model fragment in Fig. 1 (taken from Grohmann et al. 2007) the two events with the same label 'mark as 'unsuccessful' ' are the result of a modelling error: Most likely, the left branch should depict the situation 'to be marked as successful'.

However, in most cases, the multiple occurrence of the same event just means that the EPC can be simplified. An example for such a case is shown in Fig. 2. The left model does not fulfill the 'minimality' property as required by Weber (1997), i.e. the requirement to restrict the number of modelling elements to a minimum in order to improve readability.

When searching for identical events, we have to disregard trivial texts such as 'OK' or 'processed successfully'. As there is no description *what* has been processed successfully, both events can depict different managerial situations. Our algorithm disregards such trivial texts. Anyway, this pattern can cause occasional 'false alarms' (see Fig. 3 (taken from König 2006) for an example).

Furthermore, we have to consider that events which are associated to different organisational units always have to be considered as different - even if they have the same label. For example, an event 'audit successful' associated to the accounting department is something else as an event 'audit successful' associated to the legal department.

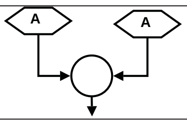
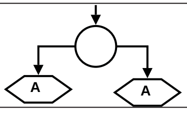
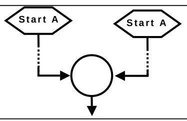
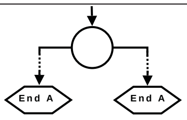
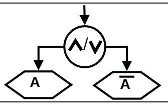
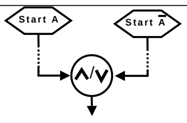
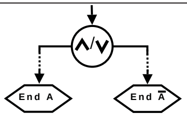
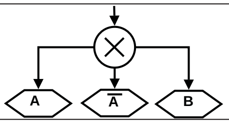
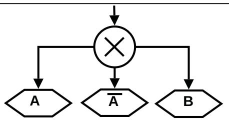
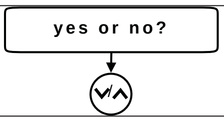
Pattern A	Identical Events Before a Join
	*Definition:* A join is preceded by two events with the same meaning. *Consequences:* In the most cases the model can be simplified by placing the event *after* the join.
Pattern B	Identical Events After a Split
	*Definition:* A split is followed by two events with the same meaning. *Consequences:* Such a construct contradicts the semantical meaning of events which should show *different* postconditions that can occur after the split.
Pattern C	Identical Start Events Precede Same Join
	*Definition:* There are two start events S_1 and S_2 with the same meaning. There is a path p_1 from S_1 to a join J and a path from S_2 to the same join J such that the only common node of p_1 and p_2 is J. *Consequences:* The modeller should double-check whether the model can be simplified.
Pattern D	Identical End Events Follow Same Split
	*Definition:* There are two end events E_1 and E_2 with the same meaning. There is a path p_1 from a split S to E_1 and a path from the same split S to $E2$ such that the only common node of p_1 and p_2 is S. *Consequences:* The modeller should consider closing the control-flow block started by S by adding a corresponding join node in order to make the model easier to read.
Pattern E	Contradicting Events at an AND/OR-Connector
	*Definition:* An AND- or OR-connector has two events which contradict each other as direct successor or predecessor. This is a logical error in the model. *Consequences:* The modeller should consider to replace the connector by an XOR-connector.
Pattern F	Contradicting Start Events Precede Same AND/OR-Join
	*Definition:* There are two start events E_1 and E_2 which contradict each other. There is a path p_1 from S_1 to a join J of type AND or OR. There is also a path from S_2 to the same join J such that the only common node of p_1 and p_2 is J. *Consequences:* This pattern can indicate a logical error in the model.
Pattern G	Contradicting End Events Follow Same AND/OR-Split
	*Definition:* There are two end events E_1 and E_2 which contradict each other. There is a path p_1 from a split S of type AND or OR to E_1. There is also a path from the same split S to E_2 such that the only common node of p_1 and p_2 is S. *Consequences:* This pattern can indicate a logical error in the model.
Pattern H	XOR Connector Preceded or Followed by Events A, $\neg A$ and B
	*Definition:* An XOR-connector is directly followed or preceded by at least three events. For two of these events (A and $\neg A$) it holds that one is the negation of the other one. *Consequences:* The purpose of a third event is questionable.
Pattern I	Comparison Between Two Values Does not Include the Case "Equality"
	*Definition:* Events after a split describe a comparision of values like "$x < y$" / "$x > y$" or changes of a value like "value x has increased" / "value x has decreased" *Consequences:* It could be the case that the modeller has forgotten the case "Equality" ("$x = y$") or "Constancy" (value x remained unchanged).
Pattern J	AND- or OR-Split After a yes/no-Question
	*Definition:* A question that can be answered by either "yes" or "no" is followed by an AND- or OR-split. *Consequences:* The modeller should consider to replace the split by an XOR-split.

Table 1: List of error patterns discussed in this article

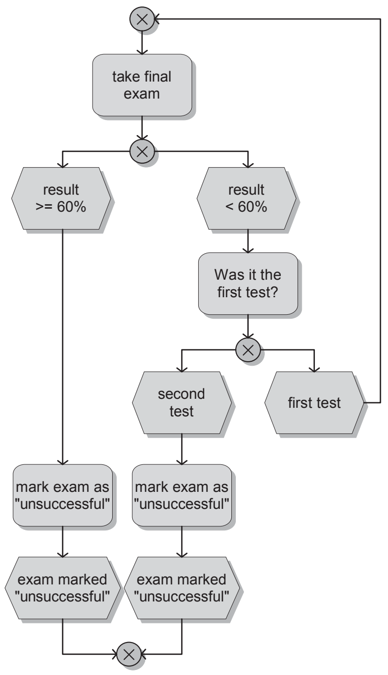

take final exam
result >= 60%
result < 60%
Was it the first test?
second test
first test
mark exam as "unsuccessful"
mark exam as "unsuccessful"
exam marked "unsuccessful"
exam marked "unsuccessful"

Figure 1: This model needs to be corrected.

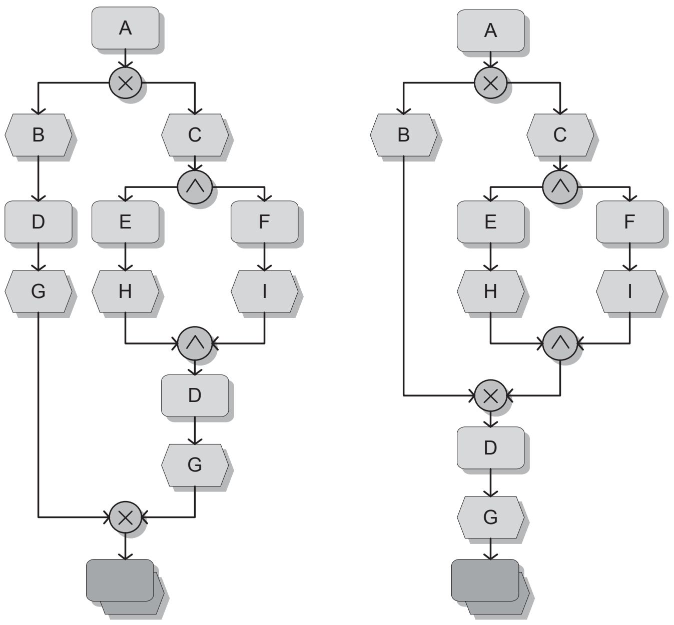


Figure 2: Left: original EPC; Right: improved EPC

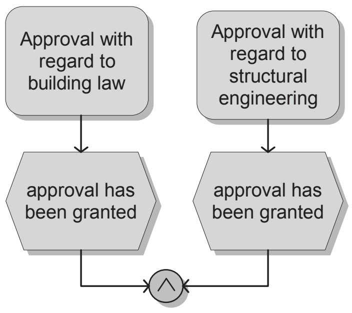

Approval with regard to building law
Approval with regard to structural engineering
approval has been granted
approval has been granted

Figure 3: False alarm for pattern A

2.2 Pattern B: Identical Events After a Split

In an EPC, events that follow a split are used to model the different postconditions that can occur after the processing of the split. There is no good reason for having modelled the same event twice after a split. If such a situation is found anyway in a model, this indicates an error that should be corrected. An example model we have identified with this problem is shown in Fig. 4 (which is a model fragment from the EPC Financial Accounting-Consolidation-Preparations for Consolidation from the SAP R/3 reference model).

2.3 Pattern C/D: Identical Start Events Precede Same Join/Identical End Events Follow Same Split

These patterns are similar to pattern A and pattern B: The purpose of start events and end events in an EPC is to model *different* situations that can occur before starting/after finishing the execution of a business process. If the same start event/end event is modelled multiple times before the same join/after the same split, this often means that the EPC can be modelled in a more structured way as shown in Fig. 5

2.4 Pattern E: Contradicting Events at an AND/OR-Connector

If an AND- or OR-split has an arc both to an event E_1 as to another event E_2, the semantics of the split means that both events can take place at the same point of time. If E_1 and E_2 contradict each other, we can assume that there is an error

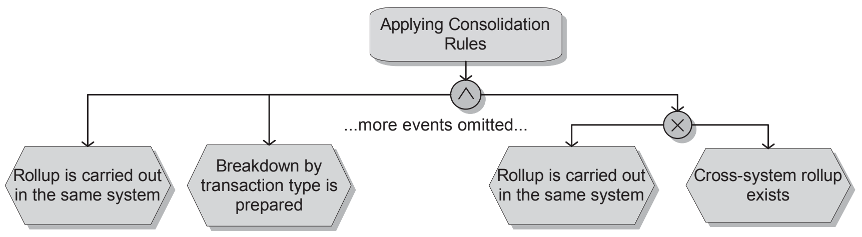

Applying Consolidation Rules
...more events omitted...
Rollup is carried out in the same system
Breakdown by transaction type is prepared
Rollup is carried out in the same system
Cross-system rollup exists

Figure 4: The event 'Rollup is carried out in the same system' seems to be modelled wrongly.

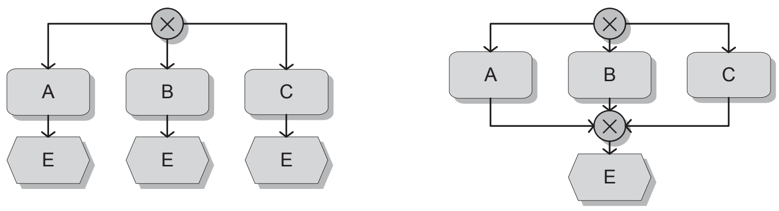


Figure 5: EPC fragment with pattern D (left) and improved EPC fragment (right)

in the model. Analogously, we can assume an error being in an EPC where contradicting events E_1 and E_2 both have an arc to the same AND- or OR-join.

Such modelling errors are quite common for novice modellers. In particular, they may be misled by a natural-language specification such as 'After the processing of X, E_1 **or** E_2 can occur'. While described by the word 'or' in natural language, the correct connector to use in an EPC diagram is the XOR connector. An example of such a modelling error (found in a diploma thesis) is shown in Fig. 6.

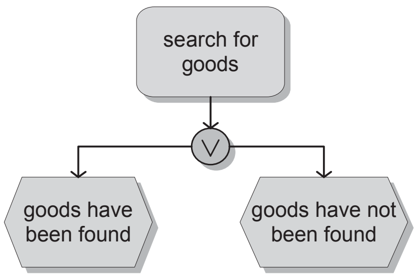


Figure 6: OR should be replaced by XOR here.

2.5 Pattern F/G: Contradicting Start Events Precede Same AND/OR-Join/Contradicting End Events Follow Same AND/OR-Split

These patterns refer to models where an event E_1 and its negation $E_2 = \neg E_1$ are modelled such that they can occur together either as start events or as end events. Such a construction seems to be erroneous. As we did not find a remarkable number of instances of these patterns in our model repository, we do not discuss them in detail. The formal definition of the patterns can be found in Tab. 1.

2.6 Pattern H: XOR Connector Preceded or Followed by Events A, $\neg A$ and B

In this pattern, an XOR-connector is directly followed or preceded by an event A, its negation $\neg A$ and at least one additional event B. As A is the negation of $\neg A$, either A or $\neg A$ should become true. For this reason, the purpose of a third event is questionable (*Tertium non datur*).

A modelling construct like the one shown in Fig. 7 (taken from Giesa and Kopfer 2000), will be at least difficult to understand[1]: Obviously, one of

[1]We are unable to judge about the correctness of the modelling fragment, because we are not familiar with the modelled domain.

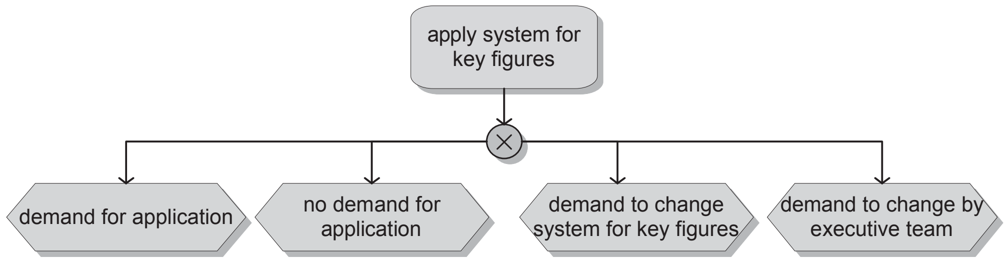

apply system for key figures
demand for application
no demand for application
demand to change system for key figures
demand to change by executive team

Figure 7: One of the both leftmost events will always occur.

the leftmost events has to occur so that there seems to be no good reason for having the other events in the model.

Interestingly, this pattern gave raise to some false alarms in cases like the one shown in Fig. 8. In this model, the statement 'CR is available and (in)complete' has been abbreviated by 'CR is (in)complete'. Although this is easy to understand by a human reader, such a construct caused our algorithm to report a possible modelling problem.

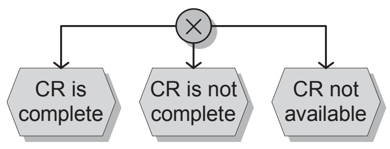


Figure 8: False alarm will be caused by pattern H

While it is in fact impossible to prevent false alarms in cases like the one shown in Fig. 8, we can prevent false alarms in another case that can frequently be found in EPCs: Often A, $\neg A$ and B depict the results of a decision such as 'yes'/'no'/'possibly'. For this reason, our algorithm does not signal the occurrence of this pattern if the label of event B contains restricting words such as 'partially', 'possibly', 'conditionally', etc.

2.7 Pattern I: Comparison Between Two Values Does not Include the Case 'Equality'

Often, the label of an an event in an EPC is based on the comparison of two values.

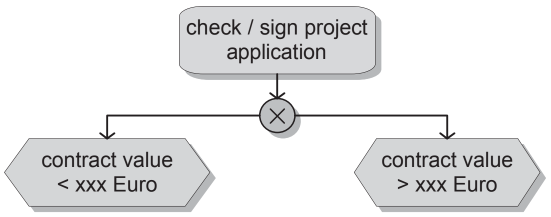

check / sign project application
contract value < xxx Euro
contract value > xxx Euro

Figure 9: It could be the case that 'value= xxx EUR' has been forgotten here.

In the example shown in Fig. 9 (taken from Becker et al. 2008, p. 634), the project application will be processed immediately if the contract value is less than a given amount of money. Otherwise, additional assessments are required. In such a case, it looks as the case 'contract value is exactly xxx Euro' has been forgotten (see Ambler 2003, rule 233).

Our algorithm identifies such situations and alerts the modeller. In the same way, we deal with events that describe the increase/decrease of some value. For example when a split is followed by two events 'demand has increased' and 'demand has decreased', we alert the modeller to consider modelling the case 'demand remains unchanged' as well.

2.8 Pattern J: AND- or OR-Split After a yes/no-Question

Usually, a function that is labelled with a question that can be answered by either 'yes' or 'no' should be followed by an XOR-split and two events (one for the positive reply, another one for the negative reply). The same holds for functions with labels of the form 'Test whether x or y'. Our

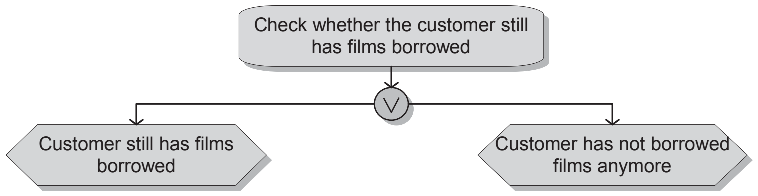

Check whether the customer still has films borrowed
V
Customer still has films borrowed
Customer has not borrowed films anymore

Figure 10: After such a decision, there should be an XOR-split.

algorithm identifies this kind of questions that are wrongly followed by an AND- or OR-split instead of an XOR-split.

An example where our algorithm can locate an error is shown in Fig. 10. In this construct, the OR should be replaced by an XOR. Note that in this case, pattern E will be found as well, but this has not to be necessarily the case.

3 Algorithm for Detecting Error Pattern Instances

Our algorithm for detecting instances of the described patterns uses the approach published in Gruhn and Laue (2007): The information contained in an EPC model is transformed into a set of Prolog facts, and Prolog queries are used for locating errors in the model. In previous papers we have shown how this method can be used for checking syntactical correctness (Gruhn and Laue 2006), absence of control-flow errors (Gruhn and Laue 2009a) and unnecessarily complicated modelling structures (Gruhn and Laue 2009b).

The necessary steps for analysing an EPC using this Prolog-based approach are as follows:

1. The information contained in the EPC model is translated into Prolog facts like `event(i_3)` (there is an event with an ID i_3) or `elementname(i_3,"Create Invoice")` (the node with ID i_3 has the label 'Create Invoice').
2. The labels are transformed into a textual normal form such that labels with the same meaning will be transformed into the same normal form.
3. The patterns described in this article are searched by querying the Prolog system.

3.1 Step 1: Translating the EPC into Prolog Facts

The translation of the information within an EPC model into Prolog facts is done by a simple XSLT transformation. Details can be found in Gruhn and Laue (2006).

3.2 Step 2: Transforming the Labels into a Textual Normal Form

In the second step, the labels of events and functions are transformed into a textual normal form. The purpose of this normal form is that labels with the same meaning should be transformed into the same normal form. For this purpose, we take into account a list of stop words, synonyms and antonyms. In our prototypical implementation, we have included 210 synonyms and 70 antonyms of German terms that can frequently be found in business process models.

For the purpose of populating our catalogue of synonyms and antonyms, we counted the frequency of words in 1154 EPC models. These models contained 66,088 words (among them 7,599 different ones). Among the most frequently found words we grouped terms with the same meaning into classes of synonyms.

For generating the normal form of a label, our algorithm repeatedly substitutes substrings of the label such that terms that belong to the same class of synonyms will be replaced by the same string that symbolises the normal form for this class. For example, in the labels 'The claim is allowed' and 'Claim has been accepted', at first stop words such as *the*, *is* and *has been* will be replaced by the empty string. Afterwards, the terms 'allowed' and 'accepted' (which both belong to the

Listing 1: Prolog rules for identifying pattern A

```
findPatternA(E1,E2) :-
split(C),type(C,xor),          % C is an XOR-split...
arc(C,E1),arc(C,E2),           % which has an arc to E1 and an arc to E2
event(E1),event(E2),           % E1 and E2 are events
E1 @< E2,                      % in particular this means E1 is not equal to E2
elementname(E1,NameE1),        % E1 has the label NameE1
elementname(E2,NameE2),        % E2 has the label NameE2
equivalent(NameE1,NameE2),     % NameE1 and NameE2 have the same meaning
not(trivial(NameE1)),          % NameE1 is not a trival label (like "ok")
not(trivial(NameE2)).          % NameE2 is not a trival label (like "ok")
```

same class of synonyms) will be replaced by the same normal form 'NF_ACCEPTED'. As the result, both labels will be transformed into the same normal form 'claim NF_ACCEPTED'.

In the same way, we deal with terms that are included into the list of antonyms. In this case, we also set a flag that signalises the fact that the normal form expresses the opposite of the original label. This way, the label 'claim dismissed' will also be transformed into the normal form 'claim NF_ACCEPTED'. The mentioned flag tells us that the label expresses the negation of the normal form.

Labels that have the form $x \diamond y$ where $\diamond \in \{<, >, =, \leq, \geq, \}$ and labels such as 'x has increased', 'x has decreased', 'x has been incremented', 'x has been decremented', 'x remained constant', etc. are processed in a special way. Instead of describing the details here, we give a few examples. Our algorithm realises that

- '$x > 1000$' and '1000 is greater than x' contradicts to '$x < 1000$' or 'x is equal to 1000'.
- 'x has increased' contradicts to 'x remained unchanged' and 'x has decreased'.
- The propositions 'x is smaller than y' and 'x is greater than y' describe two of the three possible cases 'smaller than/greater than/equal to'.

3.3 Step 3: Querying the Prolog System

For identifying the patterns, we use Prolog rules. An example for such a rule which queries for pattern A looks as shown in listing 1.

From the description of the problem patterns in Sect. 2 we see that we need Prolog rules for reasoning about the following situations:

1. **An event is a direct predecessor or a direct successor of a connector**:
   This fact is represented by the existence of an arc in the EPC. In the Prolog representation of the EPC, this is represented by a fact `arc(x,y)`.
2. **Two events have the same meaning**:
   As discussed in the explanation to step 2, this will cause both events to have the same normal form.
3. **Two labels A and B contradict each other**:
   This is the case if A and B have been transformed into the same normal form, but a pair of words that is included in the list of antonyms has been used in the process of replacements. This would be signalised by setting the corresponding flag to true.
   Another case for labels A and B contradicting each other is when A results from B by inserting a negating substring ('non-' or 'not').
   Finally, label A contradicts label B if they represent two of the three cases 'smaller than/greater than/equal to' or 'increased/decreased/unchanged'.

4. **A label expresses the fact that an event occurs only partially or only under certain circumstances**:
   This situation is assumed if the label contains a word that restrict its proposition. Such words are 'possibly', 'partially', 'potentially', etc.
5. **The label is a question that can be answered by either 'yes' or 'no'**:
   This situation is assumed if the label contains the substring 'whether' or if it ends with a question mark but does not start with an interrogative pronoun. This way, the question 'Has the claimant the authority to withdraw the claim?' is regarded as a yes/no-question while 'Which products should be offered' is not.

4 Validation

We have searched for the pattern described above in a repository of 1253 EPC models in German language that we have collected from various sources:

- 591 models from the SAP R/3 reference model, a widespread business reference model
- 127 models from textbooks
- 48 models from PhD thesises
- 70 models from published scientific papers
- 252 models from bachelor and diploma thesises and term papers
- 84 models from real-world projects
- 22 models from university lecture notes
- 13 models from technical manuals

The remaining 37 models were EPCs published on the Internet for which we were unable to assign them into one of the above categories. Among the models in our repository, there is a great variation in size of the models, purpose of modelling, business domain and experience of the modelers. For this reason, we think that the models represent a reasonable good sample of real-world models.

Because of the large number of models and the great variation in the modeled domain, we have been unable to perform a manual analysis of all models. For this reason, we are unable to decide whether there were any instances of the discussed error patterns that have not been found by our algorithm, i.e. we cannot quantify the *Recall* value for our algorithm.

However, we have analysed the *Precision* value as follows: All problems that have been signalised by our tool have been checked manually in order to decide whether the model in fact should be improved with respect to this problem pattern. In some cases, such a decision was impossible for us due to lack of domain knowledge (in particular for pattern H). If there were any doubts, we categorised a reported problem as '(possible) false alarm'.

This way, we got the following numbers of legitimate problem alerts and false alarms:

Altogether, our tool has located 129 cases in 99 EPCs where indeed an improvement of the model seemed to be advisable. On the other hand, there were 46 false alarms in 44 models. The large percentage of false alarms for the patterns C, D, F, G and H gives raise to the assumption that it might be less useful to look for these patterns. For the patterns A, B, I and J, almost all instances found of these patterns pointed to a possibility to correct or improve the model. It would be useful for a modeller to run these checks at modelling time in order to avoid the occurrence of the problem patterns.

There is a remarkable coherence between the origin of the EPCs and the classes of problems found in them. For instance, the error pattern E is very typical for novice modellers: Instead of an XOR-split, they use an OR-split which would be suggested by the specificaton of a business process in natural language. While error pattern E was found very often in students' papers (and as well in models from a modelling project in the media domain), there was no instance of this pattern in the SAP R/3 reference model. This shows

Pattern	pattern instances found	(possible) false alarms
Pattern A	70 in 54 models	1 in 1 model
Pattern B	1 in 1 model	none
Pattern C	22 in 21 models	12 in 11 models
Pattern D	21 in 21 models	17 in 17 models
Pattern E	31 in 19 models	1 in 1 model
Pattern F	2 in 2 models	1 in 1 model
Pattern G	2 in 2 models	2 in 2 models
Pattern H	21 in 18 models	12 in 11 models
Pattern I	3 in 3 models	none
Pattern J	2 in 2 models	none

Table 2: Patterns found in the 1253 EPCs of our repository

that the detection of problems by our approach would be more useful for beginners.

5 Comparison of our Approach with Related Work

The results in the previous chapter show that by analysing EPC labels, we are able to identify a remarkable number of modelling problems.

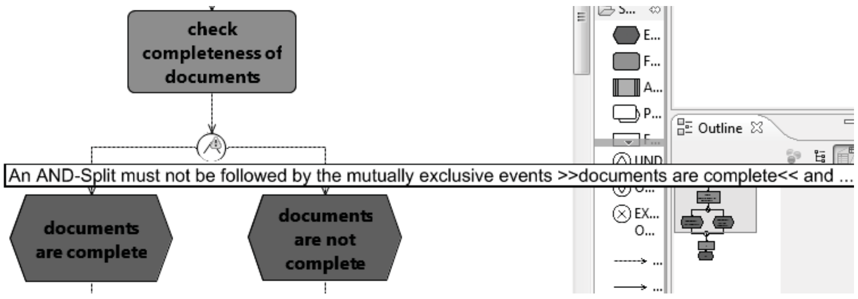
While the method has been implemented for German-language models, we see no reason why it should be difficult to adapt it to models in other languages as well. The main part of the work to achieve this adaption would be to build the synonym/antonym catalogue.

Our approach does not force the modeller to restrict the language used for the labels. Labels of events and functions can contain any phrase in natural language which is common practice in EPC modelling. By restricting the language for describing events and functions, it should be possible to improve the effectiveness of the algorithm considerably.

Such a consolidated usage of words that can occur in labels can be achieved by using the tool Semtalk (Fillies and Weichhardt 2005) which makes use of ontologies. By using such an ontology-based concept, it is even possible to assess the correctness of a business process model with regard to the application domain. Fillies and Weichhardt (2003) give an example where two business process models 'incoming order' and 'order processing' are analysed. An example for a business rule that can be checked for those models is that 'no order should be processed unless it is confirmed'. A very similar example is used in Thomas and Fellmann (2007) where EPC models are used together with ontologies of the business domain.

We are convinced that combining business process modelling and ontologies can lead to very powerful tools. Such tools can be used for checking consistency between models, for validating the compliance with business rules and for querying model repositories. Current work on this topic is described in Weber et al. (2008) and Governatori et al. (2008).

However, we are also convinced that such ontology-based methods will become very difficult to implement in practical modelling projects. The reason is that (at least during the introduction stage), the cost and complexity of such a method will be much higher than with traditional modelling. The approaches described in the literature demand to build an ontology of the business domain in addition to the usual modelling task (see for example the description on how to construct a the so-called semantical model in Filipowska et al. 2008).



An AND-Split must not be followed by the mutually exclusive events >>documents are complete<< and ...

Figure 11: Immediate feedback on errors in the labels of an EPC in the bflow Toolbox*

Our method works without the need to construct an ontology of the application domain. We only have to fill the lists of synonyms and antonyms which can be regarded as a very rudimental part of a domain ontology. In its current form, the catelogue of synonyms and antonyms is yet far from being complete. For this reason we cannot claim to detect all occurrences of the problem patterns described in this article.

While this incompleteness is clearly a disadvantage, we see an advantage in the fact that our method is easy to implement and easy to use. Our validation has shown that even a rather small catalogue of synonyms and antonyms is sufficient for finding a remarkable number of modelling problems. Other than it is the case for ontology-based approaches, our approach delivers information about the modelling problems without adding to the complexity of the modelling method.

The pattern-based analysis of EPC models has already been integrated into the open-source modelling tool *bflow* Toolbox* (see Fig. 11). The analysis can be started 'at the push of a button'. This adds more power to the business process modelling with immediate feedback in the *bflow* Toolbox* as described in Gruhn et al. (2009).

Awad et al. (2008b) use information retrieval techniques for defining a measure of similarity between labels within a BPMN diagram. This approach uses the general-purpose lexical database WordNet (Fellbaum 1998), which makes it possible to compare the similarity of labels without having to restrict the use of the language. Combining Awad et al. (2008b) with previous work on BPMN-Q (Awad et al. 2008a) makes it possible to validate business rules like 'An account must not be opened until certain checks have been completed'.

A similar approach is described in Koschmider and Oberweis (2007). However, the main purpose of this work is not the validation of business process models but to detect model variants.

An important difference between the method described in our article and the approaches given by Awad et al. (2008b) and Koschmider and Oberweis (2007) is that we are able to locate a substantial number of modelling problems with a very small catalogue of synonyms and antonyms that can be used very efficiently. It will be an interesting subject for further studies how the results of our pattern search can be improved by using more advanced algorithms (as described in Awad et al. 2008b, Koschmider and Oberweis 2007 or Gervasi and Zowghi 2005) for detecting labels with the same meaning or labels that contradict each other.

It is another interesting research field to validate whether naming conventions for events and functions have been followed. Bögl et al. (2008) show how lexical databases and semantical patterns for labels can be used for this purpose. Other authors have suggested somewhat different solutions for the same problem: Both Leopold et al. (2009) and Becker et al. (2009) make use of linguistic syntax parsing methods for assuring the compliance to naming conventions.

An approach to measure the label quality of EPC models has been suggested by Friedrich (2009). In particular, this method can find labels with a high chance of ambiguity.

It is our intention to research more patterns that can signalise the possiblity to improve a model. Researchers and practitioners are invited to use and to improve the tests in the *bflow* Toolbox*. The most current release can be downloaded from `www.bflow.org`[2].

References

Ambler S. W. (2003) The Elements of UML Style. Cambridge University Press

Awad A., Decker G., Weske M. (2008a) Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Proceedings of the 6th International Conference on Business Process Management. Springer, Milan, pp. 326–341

Awad A., Polyvyanyy A., Weske M. (2008b) Semantic Querying of Business Process Models. In: 12th International Conference on Enterprise Distributed Object Computing EDOC

Becker J., Kugeler M., Rosemann M. (2008) Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 6th ed. Springer, Berlin

Becker J., Delfmann P., Herwig S., Lis L., Stein A. (2009) Formalizing Linguistic Conventions for Conceptual Models. In: Procceedings of the 28th International Conference on Conceptual Modeling (ER 2009). Lecture Notes in Computer Science 5829. Springer, Gramado, pp. 70–83

Bögl A., Schrefl M., Pomberger G., Weber N. (2008) Semantic Annotation of EPC Models in Engineering Domains by Employing Semantic Patterns. In: ICEIS 2008 – Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume AIDSS, Barcelona, pp. 106–115

[2]The Prolog rules that are responsible for the tests described in this article can be found in the plugin org.bflow.toolbox.mitamm.prolog. The list of synonyms and antonyms we have used for the purpose of our validation can be found at the same place.

Esswein W., Gehlert A., Seiffert G. (2004) Towards a Framework for Model Migration. In: Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, June 7–11, 2004, Proceedings. Lecture Notes in Computer Science Vol. 3084. Springer, pp. 463–476

Fellbaum C. (ed.) WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press, Cambridge

Filipowska A., Kaczmarek M., Stein S. (2008) Semantically Annotated EPC within Semantic Business Process Management. In: Ardagna D., Mecella M., Yang J. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing Vol. 17. Springer, Berlin, pp. 486–497

Fillies C., Weichhardt F. (2003) Towards the Corporate Semantic Process Web. In: Berliner XML Tage. XML-Clearinghouse, pp. 78–90

Fillies C., Weichhardt F. (2005) On Ontology-based Event-driven Process Chains. In: EPK 2005, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten

Friedrich F. (2009) Measuring Semantic Label Quality Using Word Net. In: EPK 2009, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. CEUR Workshop Proceedings

Gervasi V., Zowghi D. (2005) Reasoning about inconsistencies in natural language requirements. In: ACM Trans. Softw. Eng. Methodol. 14(3), pp. 277–330

Giesa F., Kopfer H. (2000) Management logistischer Dienstleistungen der Kontraktlogistik. In: Logistik Management 2(1), pp. 43–53

Governatori G., Hoffmann J., Sadiq S., Weber I. (2008) Detecting Regulatory Compliance for Business Process Models through Semantic Annotations. In: 4th International Workshop on Business Process Design. Springer, Berlin

Grohmann G., Kraemer W., Milius F., Zimmermann V. (2007) Modellbasiertes Curriculum-Design für Learning Management Systeme: Ein Integrationsansatz auf Basis von ARIS und IMS Learning Design. In: Wirtschaftsinformatik Proceedings 2007. Universitätsverlag Karlsruhe, pp. 795–812

Gruhn V., Laue R. (2006) Validierung syntaktischer und anderer EPK-Eigenschaften mit PROLOG. In: EPK 2006, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, 5. Workshop der Gesellschaft für Informatik e.V., pp. 69–84

Gruhn V., Laue R. (2007) Checking Properties of Business Process Models with Logic Programming. In: Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS) 2007. INSTICC Press, Funchal, Madeira, pp. 84–93

Gruhn V., Laue R. (2009a) A Heuristic Method for Detecting Problems in Business Process Models. In: Business Process Management Journal 16(5), pp. 806–821

Gruhn V., Laue R. (2009b) Reducing the Cognitive Complexity of Business Process Models. In: IEEE International Conference on Cognitive Informatics, Hong Kong

Gruhn V., Laue R., Kühne S., Kern H. (2009) A Business Process Modelling Tool with Continuous Validation Support. In: Enterprise Modelling and Information Systems Architecture 4(2), pp. 37–51

König M. (2006) Workflow-Management in der Baupraxis. In: 4. Tag des Baubetriebs 2004 – Tagungsbeiträge Nachtragsmanagement in Praxis und Forschung, Schriften der Professur Baubetrieb und Bauverfahren. Bauhaus-Universität Weimar

Koschmider A., Oberweis A. (2007) How to detect semantic business process model variants? In: Proceedings of the 2007 ACM symposium on Applied computing. ACM, Seoul, pp. 1263–1264

Leopold H., Smirnov S., Mendling J. (2009) On Labeling Quality in Business Process Models. In: EPK 2009, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. CEUR Workshop Proceedings

Mendling J. (2007) Detection and Prediction of Errors in EPC Business Process Models. PhD thesis, Wirtschaftsuniversität Wien, Wien

Pfeiffer D., Niehaves B. (2005) Evaluation of Conceptual Models – A Structuralist Approach. In: Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy, ECIS 2005, Regensburg

Rump F. J. (1999) Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten. B. G. Teubner Stuttgart

Thomas O., Fellmann M. (2007) Semantic EPC: Enhancing Process Modeling Using Ontology Languages. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management, Innsbruck. CEUR Workshop Proceedings Vol. 251

van der Aalst W. M. P. (1997) Verification of Workflow Nets. In: Application and Theory of Petri Nets 1997, Procceedings of the 18th International Conference, ICATPN '97, Toulouse, pp. 407–426

Weber I., Hoffmann J., Mendling J. (2008) Semantic Business Process Validation. In: 3rd international workshop on Semantic Business Process Management at Extended Semantic Web Conference 2008

Weber R. (1997) Ontological Foundations of Information Systems. 4. Coopers and Lybrand Accounting Research Methodology monograph

Volker Gruhn

University of Duisburg-Essen, Germany,
PALUNO – The Ruhr Institute for Software Technology
volker.gruhn@uni-due.de

Ralf Laue

University of Leipzig, Germany, Chair of Applied Telematics / e-Business
laue@ebus.informatik.uni-leipzig.de