Automatic User Support for Business Process Modeling

Stefanie Betz, Stefan Klink, Agnes Koschmider, and Andreas Oberweis

Institute of Applied Informatics and Formal Description Methods
Universität Karlsruhe (TH), Germany
{betz,klink,koschmider,oberweis}@aifb.uni-karlsruhe.de

Abstract. The main purpose of business process modeling is the representation and analysis of alternative process designs by formal or semiformal process models. Manual modeling of business processes is a time-consuming task. Typos and structural modeling errors make it particularly error prone to model business processes manually. Users can be assisted in modeling business processes by providing an autocompletion mechanism during the modeling process. In this paper we will describe on-going work for autocompletion of business process models. This approach is based upon an OWL DL description of Petri nets. Our autocompletion mechanism requires validation methods to check process properties of the automatically completed business process, which we will introduce as well. Consequently, we aim to improve manual process modeling by automating process modeling to a significant extent.

1 Introduction

The main purpose of business process modeling is the representation and analysis of alternative process designs by formal or semiformal process models. Many modeling languages – most of them being based on textual programming languages or graphical notations such as Petri nets [20], EPCs [23] or BPMN [27] – have been proposed for process modeling. Novel orchestration- and choreography languages such as BPEL [1] focus on tracking and executing business processes by business applications. To enable verification of BPEL [8] proposes a Petri net semantics. Petri nets have been established as a suitable language for modeling business processes with intuitive graphical notation. Furthermore, Petri nets have a mathematical foundation, which enables simulation and analysis of system behavior.

In this paper we will describe on-going work for autocompletion of Petri net based business process models. Manual modeling of business processes is a time-consuming task. Typos and structural modeling errors make it particularly error prone to model business processes manually. Users can be assisted in modeling business processes by providing an autocompletion function during the modeling process. However, process element names might differ in syntax even when they have the same meaning (homonyms) or one process can be modeled in different ways even when utilizing the same modeling language. It is possible that these process elements will not be suggested as fitting elements to provide autocompletion.

To solve ambiguity issues caused by the use of different names for describing the same tasks a machine readable and interpretable format, which might be used for machine reasoning, is required for Petri nets. Business processes modeled with Petri nets can be translated into the Web Ontology Language OWL [17], an unambiguous format which allows ontological reasoning. So-called semantic business process models combine process modeling methods with semantic technologies to achieve automatic processing of business process models instead of manual processing. We will use a semantic description of Petri nets to make it easier to find appropriate process templates (reference processes), which can be proposed for autocompletion. During the modeling process, a recommendation mechanism determines possible subsequent fragments of all templates by computing similarities. If the system detects a high similarity between one element of a template and a modeling element, then subsequent elements of this element template are proposed for autocompletion. To ensure correct process flow behavior the system has additionally to check properties such as deadlock-freeness.

The structure of this paper is as follows. Firstly, we will recall the main notions of Petri nets and a semantic description of Petri nets with OWL DL. In Section 3 we will describe an approach for measuring similarity between semantic business process models by utilizing *syntactic-*, *linguistic-* and *structural* similarity measurements. To validate process behavior properties we will illustrate analysis methods in Section 4. Modeler's behavior can be observed and learned with machine learning techniques, which we will briefly survey in Section 5. Section 6 concludes the paper with an outlook on future research.

2 Foundations

Next subsection introduces Petri net notation and Petri net modeling of business processes.

2.1 Petri nets

Petri nets are a graphical language and a formalism used to model business processes and verify system behavior. Formally, a Petri net is a directed bipartite graph with nodes and arcs. It can be described by the triple $N = (P, T, F)$, where P is a set of places, T is a set of transitions (which is disjoint from P) and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation. Elements of P are graphically represented as circles, elements of T as boxes and elements of F as directed arcs between places and transitions. A place p is an input place of a transition t, if there exists a directed arc from p to t. A place p is an output place of a transition t, if there exists a directed arc from t to p. The set of all input places of a transition t is denoted by $\bullet t$ and is called *preset*. The set of all output places is denoted by $t\bullet$ and is called *postset*.

Numerous Petri net variants have been proposed, which can be subsumed in elementary or high-level Petri nets. In elementary Petri nets places contain tokens, which represent anonymous objects. A transition t is enabled to change the marking m of the net, if each place p in the preset contains at least one token. If such a transition t occurs, then t consumes tokens from the preset and inserts tokens in the postset. The occurrence sequence, which leads from m to m' is defined by $m \overset{t}{\rightarrow} m'$. A marking m' is called *reachable* from m, if there exists an occurrence sequence from m to m' ($m \overset{\delta}{\rightarrow} m'$) with $\delta = t_1, t_2 \dots t_n$.

For modeling system behavior or business processes with Petri nets we distinguish several process flow structures [26] as depicted in Figure 1. The flow structure OR-split allows to model alternative branching. The two alternative branches are again integrated by a so-called OR-join or a so-called AND-join (synchronization). By the use of AND-split (concurrency) tokens are distributed to two places.

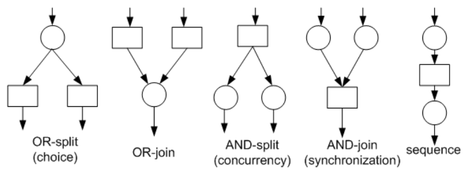

Fig. 1. Overview of Process Flow Structures

The formal foundation of Petri nets allows to verify whether a modeled business process meets certain properties such as deadlock-freeness. If a marking is reached, which is not an intended final process state and which doesn't enable any transition, then the process is in a deadlock, as depicted in Figure 2.

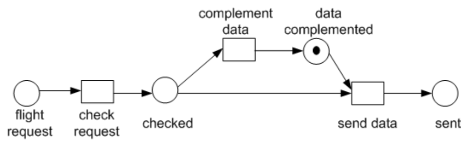

Fig. 2. Deadlock in a Petri net

To make tokens distinguishable, variants of high-level Petri nets have been proposed such as Coloured Petri nets [6] or Predicate/Transitions nets [10]. In this paper, we focus on Predicate/Transitions nets (Pr/T nets) where places are interpreted as predicates representing relation schemes. Transitions occur according to a logical expression which may be attached to them.

2.2 Semantic Business Process Models

To describe business processes modeled with Petri nets in an unambiguous format we have proposed a translation of Pr/T nets to OWL DL [13]. For our work we will refer to OWL DL (Description Logic) in order to be able to use available off-the-shelf reasoning technologies. OWL DL can be considered as a syntactic variant of the $\mathcal{SHOIN}(D)$ Description Logic which is known to be decidable [9]. In Table 1 the main constructs of $\mathcal{SHOIN}(D)$ are shown.

Table 1. Constructs of $\mathcal{SHOIN}(D)$ Syntax

subClass	$A_1 \sqsubseteq A_2$
intersection	$A_1 \sqcap ... \sqcap A_n$
union of	$A_1 \sqcup ... \sqcup A_n$
allValuesFrom	$\forall$ P.A
someValuesFrom	$\exists$ P.A
maxCardinality	$\leq$ nP
minCardinality	$\geq$ nP

In our Pr/T net ontology each Petri net element corresponds to an OWL concept. *Places* are described by the concept *Place*, *transitions* by *Transition* and *arcs* by $FromPlace$ $(P \times T)$ and $ToPlace(T \times P)$. For instance, the concept PetriNet is defined by at least one transition, place and arc.

Table 2. Pr/T net Ontology

$PetriNet \equiv$	$\geq 1 hasNode.(Transition \sqcap Place)$
	$\sqcap \geq 1 hasArc.(FromPlace \sqcup ToPlace)$
$Transition \equiv$	$placeRef.Place \sqcap = 1 haslogicalConcept.LogicalConcept$
$Place \equiv$	$transRef.Transition \sqcap = 1 hasMarking.IndividualDataItem$
$FromPlace \equiv$	$\geq 1 hasInscription.Delete \sqcap \exists hasNode.Place$
$ToPlace \equiv$	$\geq 1 hasInscription.Insert \sqcap \exists hasNode.Transition$
$LogicalConcept \equiv$	$= 1 hasConditon.Condition \sqcup = 1 has.Operation.Operation \sqcap$
	$\exists hasAttribute.IndividualDataItem$
$IndividualDataItem \equiv$	$\geq 1 hasAttribute.Attribute$
$Delete \equiv$	$\forall hasAttribute.IndividualDataItem$
$Insert \equiv$	$\exists hasAttribute.IndividualDataItem$
$Atrribute \equiv$	$\leq 1 hasValue.Value$
$Value \equiv$	$hasRef.Value$
$Condition \equiv$	$forall(string) \sqcup exists(string) \sqcup and(string)$
$Operation \equiv$	$function(string)$

To the best of our knowledge, there exists no other approach that transforms high-level Petri nets into OWL DL. In the next section we will explain how to measure similarity between a pair of semantic business process model elements.

3 Measuring Similarity between Process Elements

Before proposing appropriate process fragments, the recommendation mechanism has to compare modeling elements with process templates and has to find similar elements, which will be proposed as fitting subsequent elements. Thus, the mechanism has to compare process templates with process elements, which are currently modeled, by computing their similarities. The autocompletion system observes and learns from the modeler's behavior. Figure 3 gives an overview of our recommendation process[1]. Instead of representing OWL syntax, which is not readable for modelers, we have depicted business processes in graphical Petri net notation. However, the automatical similarity computation between process elements is based upon the OWL serialisation of Petri nets.

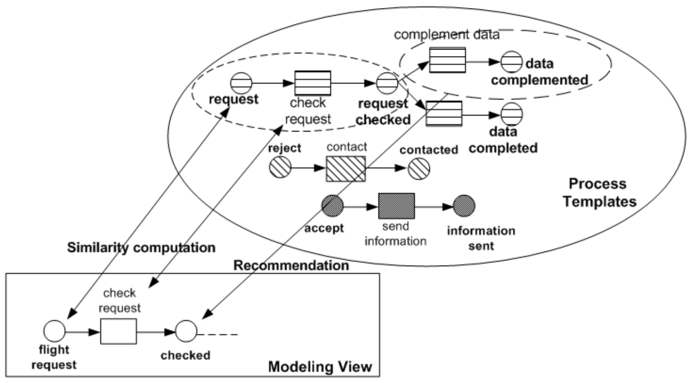

Fig. 3. Overview of the Auto Completion process

Our approach for automatic process element propositions is similar to the autocompletion function in mobile phones. The installed system observes what the user is currently typing, and tries to complete words automatically.

We propose to compute similarities between two process element names by utilizing the similarity measures *syntactical*, *linguistic* and *structural* measurements. Syntactical similarity measures take as input two character strings and compare them. A well-established string similarity measurement has been proposed by Levenshtein [14] which takes into account the amount of operations (insertion, deletion and substitution) needed to transform one string into another. For example, to turn *flight_request* to *request* requires seven deletions. Based on the Levenshtein method [16] has proposed a syntactic similarity measurement which returns for the similarity calculation similarity degrees between 0 and 1. But, syntactical similarity measurements alone are not sufficient since they do not regard the semantics of words. In addition to the syntactical similarity measurement we are utilizing background knowledge in terms of ontologies.

[1] different shading of process templates visualizes independent process models.

Petri nets obey an operational semantics that describes the control flow. However, a missing semantic description of Petri net components hampers the automated processing of process elements. To uncover synonyms or homonyms of process element names we need a description of Petri nets in an umambiguous format such as OWL DL. With the background ontology – numerous background ontologies are modeled with OWL – we compute linguistic similarity measures. Ontologies have paved the way for standardized formal conceptualizations of all kinds of knowledge. To compute linguistic similarity degrees we have worked with WordNet[2] via the JWNL API [5] and with a specific UML Profile [4]. The benefit of the UML-based background ontology is that a converting tool provides an automatic translation from the visual UML modeling to OWL DL syntax. WordNet is in contrast to the UML Profile predefined and fix.

However, to improve the aggregation of syntactical and linguistic similarity measurements we take into account the context of element names by considering structural information of names. For instance, the structure of place names is influenced by places attributes, values and the subsequent transitions as depicted in Figure 4 exemplarily for the place *flight request*. This place has a marking with the name *R_flight_request* and the corresponding attributes *Name*, *Destination*, *Date* and *Quantity* where *Destination* has the values *PAR* (corresponding to the destination Paris) and *FRA* (corresponding to the destination Frankfurt). The subsequent transition of *flight request* is *check request*.

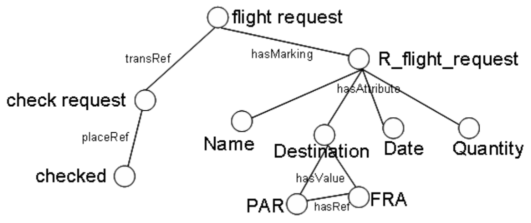

Fig. 4. Context of the Place *flight request*

However, each concept influences place names with different weights. Therefore, we have determined different weights for the concepts as shown in Table 3. In business process models with a lot of places, attributes and transitions weights play a less important role than in small processes with less process elements. The more instances are modeled in a SBPM the more extensive is the context of instances. The processes modeled in this paper have only few places and transitions[3]. Furthermore, depending on the features the similarity measure might differ. To determine structural similarity between place names we compute the syntactical similarity degree of names (measure for names is syn-

[2] an english online lexical reference system, which provides synonym and hypernym sets consisting of nouns, verbs, adjectives, and adverbs [18].

[3] To make the business processes readable we did not assign values for attributes

tactical similarity) and the linguistic similarity degrees of their attributes, values and subsequent transitions.

Table 3. Features and Similarity Measures for Petri nets

Comparing	Feature	Measure	Weight
Places	name	synt sim.	0.2
	Attribute/Value	str sim.	0.5
	successor	ling sim.	0.3
Attributes	name	synt sim.	0.2
	sibling Attribute	ling sim.	0.3
	Values	ling sim.	0.3
	Place	ling sim.	0.2
Values	name	synt sim.	0.2
	Attribute	ling sim.	0.5
	Value reference	ling sim.	0.3
Transitions	name	synt sim.	0.2
	ToPlace	sling sim.	0.4
	FromPlace	ling sim.	0.4

By aggregating these three similarity measures to a combined similarity measure we consider syntactical, linguistic, and structural properties of elements and can compute a more significant similarity between two elements. Table 4 shows some combined similarity degrees (sim_{com}) for process names ($name_{BP}$) and fragments ($name_{BF}$). If the algorithm computes a combined similarity degree > 0.5 between a template element and a process element, which is currently modeled, then the recommendation system proposes it for autocompletion. The user can then decide if (s)he accepts the proposition or rejects it. To learn a threshold θ instead of using a fix value we will present in Section 5 several machine learning techniques.

Table 4. Results of Combined Similarity

$name_{BP}$	$name_{BF}$	sim_{com}
flight request	request	0.8
check request	check request	1.0
request checked	checked	0.9
...	...	...

In the following section we will sketch a method for validating behavior properties of autocompleted business processes.

4 Analysis Methods for Petri nets

To check if a process model meets certain properties, several analyzing methods have been proposed [25]. In our approach we utilize such methods to validate that the insertion of the proposed process fragments does not cause deadlocks and synchronization errors. Thus, our automatic user support includes more than the recommendation of process fragments.

In the following we are focusing on validation of process properties by checking if the autocompleted business process is deadlock free and without lack of synchronization. While utilizing validation algorithms for Pr/T nets we are not considering any inscriptions of places, transitions, or arcs. Instead we are focusing on the net structure of the modeled process by identifying all possible flows based on instance subgraphs [22] (see Figure 5 where we have modeled the instance subgraph for Figure 2). We assume, that the analysis of a specific Pr/T net satisfies the following requirements: the Pr/T net has only one source place and one sink place, every place p and every transition t is on the path between the source and the sink [24]. Additionally, cycles in the net structure must be regarded as single execution units.

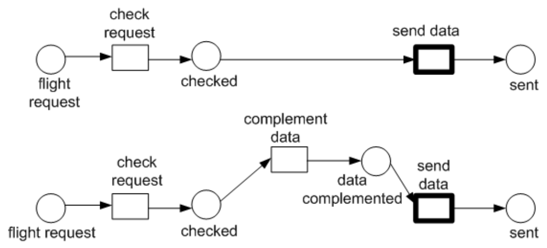

Fig. 5. Instance Subgraphs of Petri net in Figure 2

A deadlock can occur, if branches of an OR-split (such a split can be easier identified by an instance subgraph) are synchronized by an AND-join, as depicted in Figure 2. For instance, if the user inserts the transition *complement data* and its subsequent place *data complemented* and intends to synchronize these elements by inserting a connection from *checked* and *data complemented* to *send data*, then a deadlock occurs.

A lack of synchronization occurs, if an AND-split is synchronized by an OR-join, as depicted in Figure 6. For instance, if the user inserts the transition *travel request* and intends to synchronize the concurrent branches by inserting the place *transport request*, then a lack of synchronization occurs. To facilitate manual modeling we aim to develop a recommendation mechanism, which advices only validated fragments.

We have sketched how deadlocks and lack of synchronization can be discovered and now we will introduce reduction rules to validate both process properties [22], [15], [25]. According to the reduction rules a process is free of structural

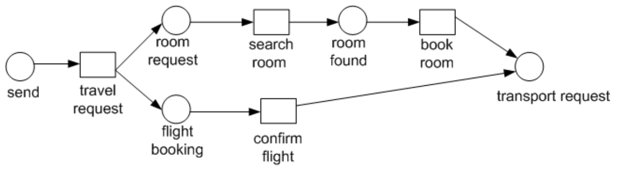

Fig. 6. Example for a Lack of Synchronization

errors if the reduction results in an empty graph. Because of the simplification of the presented processes, we applied only two of the rules, the terminal reduction rule (trr) and the sequential reduction rule (srr) as shown in Figure 7. At the end there are two nodes left over, so the modeled process is not free of deadlocks.

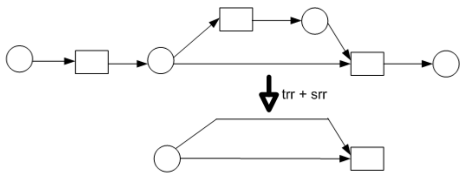

Fig. 7. Execution of the Reduction

5 Learning User Behavior

As described in Section 3 our system is capable of recommending similar elements and the user is asked to select appropriate templates. For further recommendations the users behavior could be observed, i.e., logging which decisions have been made in which situation/context and at which process position or template.

To recommend appropriate elements, a classification method must be utilized to classify the set of stored elements into partitions. These partitions can then be compared with the current user-behavior and the best one(s) concerning context and other criteria as described above will be recommended for selection. Several methods can be utilized to learn user's behavior. The following list gives an overview of established learning techniques.

Support Vector Machines The goal of a support vector machine (SVM) is a classifier in a high-dimensional feature space which can handle even not-separable problems. SVMs can be used to find hypotheses which guarantee a minimal error [11]. SVMs determine those coefficients which separate the training set with the shortest distance vectors. Only the so-called *support vectors* are of interest and are spanning the separating hyperplane. One advantage of SVMs is that they can learn non-linear hypotheses by using a mapping function which maps the original feature space into a (commonly) higher dimensional feature space or by using learning techniques like polynom classifiers, radial-basis classifiers [3] or two-hidden layer (2HL) networks which can form even more complex decision boundaries [2]. Another advantage of SVMs is that they can handle noise-(error-)reduction by deleting those examples – in our case the observed user-behavior – which cause the non-separability. SVMs are robust and do not need any parameter optimization, i.e., they can work on raw data.

Neural Nets Neural nets are a well-established and a successfully constituted technique in the area of soft computing to process information in a vague and tolerant manner. But the most important feature of neural nets is their learning capability which increases the adaptivity of the system and supports the complexity of heterogenous data. A variety of models are available to choose. For recommender systems in general backpropagation algorithms are especially suitable to handle the users feedback and selections to train the net. In our case, neural nets can be used for example to learn a threshold θ in Section 3 for the combined similarity – instead of using a threshold with a fixed value of 0.5.

Bayesian Networks Alternative names are *belief networks*, *probabilistic independence networks*, *influence diagrams*, or *causal nets* [19]. A Bayesian network is a directed acyclic graph (DAG) which represents probabilities of dependencies between a set of random features [7]. The nodes of the graph represent the set of random variables $\mathbb{X} = \{X_1, \ldots, X_n\}$ and the directed edges represent the dependency between these variables. The second part of a Bayesian network is a set of conditional probabilities $P(X_{ij}|X_i)$ according to the associated graph. One advantage of Bayesian networks is that the representation of a probability distribution as a directed graph enables the analysis of complex conditional user events with a graph theoretical approach and ensures the consistency of the system [21]. With Bayesian networks it is possible to calculate conditional probabilities, i.e., to estimate the users future behavior in case of observations made in the past.

Information Filtering Information Filtering is well known from shopping web sites. When the customer clicks on an item I_1, then the system proposes: "Customers who have bought this item I_1 also have bought another item I_2 too…". The advantage of this technique is its simplicity. It can be implemented very easily (only log files have to be parsed) and it can learn incrementally. In our case, Information Filtering can be used to learn behaviors like: "Customers who have inserted this process template at this position also have done this and that…". Furthermore, with Information Filtering techniques the system is able to generate and to compare user-profiles which help to categorize users, e.g., in beginners or specialists [12].

Content-based Information Filtering Content-based Information Filtering is a new extension which also takes the content of the underlying items into account, i.e. I_1 and I_2 is also compared and their similarity is regarded while ranking. With this method the combined similarity of process template elements described in section 3 can be used to calculate the similarity and to rank the recommendations retrieved by the Content-based Information Filtering technique.

Summarizing the upper list, especially each variant of Information Filtering is a promising technique for recommending process templates, which will be further considered.

6 Conclusion

In this paper we have presented on-going work for assisting users in business processes modeling. Compared to manual process modeling we aim to improve the reusability of business processes by an autocompletion mechanism. We propose to use a recommendation system, which observes the user's behavior and suggests possible subsequent elements. Furthermore, the system facilitates validated process properties of the automatically completed process to avoid deadlocks and lacks of synchronization.

Modeling languages such as EPCs, BPMN or BPEL have no direct formal foundations and thus do not enable analysis methods. Hence, if the process is modeled with these languages the process to be automatically completed can not be directly validated regarding deadlocks or lacks of synchronization.

Currently, we are developing an algorithm to automatically analyze hierarchical specifications of process elements and to detect errors in process models. Processes with hierarchical specifications generally appear to be more intuitive and easier to understand.

References

1. A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K. Liu, S. Thatte, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. wsbpel-specification-draft-01, OASIS, September 2005.
2. R. Beale and T. Jackson. *Neural Computing: An Introduction.* Institute of Physics Publishing, Bristol, U.K. and Philadelphia, PA, 1990.
3. C. M. Bishop. Neural networks for pattern recognition. Technical report, Oxford, Clarendon, P., 1995.
4. S. Brockmans, M. Ehrig, A. Koschmider, A. Oberweis, and R. Studer. Semantic Alignment of Business Processes. In *Proceedings of the 8th International Conference on Enterprise Information Systems*, Paphos, Cyprus, May 2006. to appear.
5. J. Didion. *JWNL 1.3*, November 2003. http://www.codezoo.com/pub/component/196?/category=5.
6. H. J. Genrich and K. Lautenbach. System modelling with high level petri nets. *Theoretical Computer Science*, (13):109–136, 1981.
7. D. Heckerman. A Tutorial on Learning with Bayesian Networks. Technical report MSR-TR-95-06, Microsoft Research, March 1995.
8. S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri Nets. In *Business Process Management*, pages 220–235, 2005.

9. I. Horrocks. Applications of Description Logics: State of the Art and Research Challenges. In *Proceedings of the International Conference on Conceptual Structures*, Lecture Notes in Computer Science, pages 78–90. Springer, 2005.
10. K. Jensen. An Introduction to the Theoretical Aspects of Coloured Petri Nets. In J. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency – Reflections and Perspectives*, Lecture Notes in Computer Science, pages 230–272. Springer, 1994.
11. T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In C. Nedellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, Lecture Notes in Computer Science, pages 137–142, Chemnitz, Germany, 1998.
12. S. Klink. Query reformulation with collaborative concept-based expansion. In *Proceedings of the First International Workshop on Web Document Analysis (WDA 2001)*, pages 19–22, Seattle, Washington, USA, 2001.
13. A. Koschmider and A. Oberweis. Ontology based Business Process Description. In J. Castro and E. Teniente, editors, *Proceedings of the CAiSE-05 Workshops*, Lecture Notes in Computer Science, pages 321–333, Porto, Portugal, June 2005.
14. V. I. Levenshtein. Binary Code capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, (8):707–710, 1966.
15. H. Lin, Z. Zhao, H. Li, and Z. Chen. A Novel Graph Reduction Algorithm to Identify Structural Conflicts. In *Proceedings of the 35th Hawaii international Conference on System Sciences*, pages 536–550. IEEE Computer Society Press, 2002.
16. A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proceedings of the European Conference on Knowledge Acquisition and Management*, Lecture Notes in Computer Science, 2002.
17. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3c recommendation, World Wide Web Consortium, 2004.
18. G. A. Miller, C. Fellbaum, and R. Tengi. WordNet – a lexical database for the English language, 2006. `http://wordnet.princeton.edu/`.
19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.
20. W. Reisig and G. Rozenberg. *Lectures on Petri Nets: Basic Models*. Lecture Notes in Computer Science. Springer, 1 edition, 1998.
21. B. A. Ribeiro-Neto and R. Muntz. A belief network model for IR. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 252–260, Zurich, Switzerland, August 18-22 1996. ACM Press.
22. W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Inf. Syst.*, (2):117–134, 2000.
23. A.-W. Scheer and M. Nüttgens. ARIS Architecture and Reference Models for Business Process Management. In *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806, pages 376–389. Springer, 2000.
24. W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *Business Process Management*, pages 161–183, 2000.
25. W. M. P. van der Aalst, A. Hirnschall, and H. M. W. E. Verbeek. An Alternative Way to Analyze Workflow Graphs. In *CAiSE*, pages 535–552, 2002.
26. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
27. S. A. White. Business Process Modeling Notation. Specification, BPMI.org, 2004.