Automatic Derivation of Service Candidates from Business Process Model Repositories

Henrik Leopold[1] and Jan Mendling[2]

[1] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
henrik.leopold@wiwi.hu-berlin.de
[2] WU Vienna, Augasse 2-6, A-1090 Vienna, Austria
jan.mendling@wu.ac.at

Abstract. Although several approaches for service identification have been defined in research and practice, there is a notable lack of automatic analysis techniques. In this paper we take the integrated approach by Kohlborn et al. as a starting point, and combine different analysis techniques in a novel way. Our contribution is an automated approach for the identification and detailing of service candidates. Its output is meant to provide a transparent basis for making decisions about which services to implement with which priority. The approach has been implemented and evaluated for an industry collection of process models.

1 Introduction

Services-Oriented Architecture has been discussed for roughly a decade as a concept to increase the agility of a company in providing goods and services to external partners and organizing internal operations. In this context, a service can be understood as an action that is performed by an entity on behalf of another one, such that the capability of performing this action represents an asset [1]. The focus on services is supposed to improve business and IT alignment, as it establishes principles like abstraction, autonomy and reuse [2].

A plethora of approaches to service derivation have been defined in the past. A core problem is though that many of these approaches lack methodological detail, and that none of them builds on automatic analysis techniques, cf. [2]. The problem is that a manual approach does not scale up to the size of a whole company. Indeed, several approaches recommend the manual specification of capabilities, among others based on interviews [3,4,2]. The benefits of reusing information artifacts, e.g. process models, has been recognized, among others in [5], but not in an automatic way. However, the entirety of a company can hardly be taken into account as long as models have to be manually created and inspected.

In this paper, we address the problem of manual work in the phases of service derivation. We consider the situation where an extensive set of hundreds of process models is available, which is realistic for many medium-sized and big companies [6]. Our contribution is an approach for the automatic derivation of service candidates, augmented with a set of metrics giving first clues about

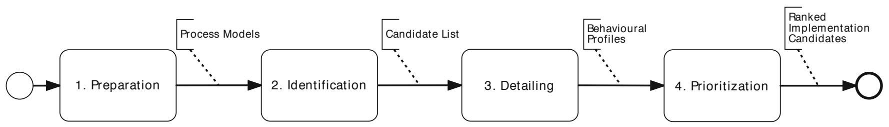

Fig. 1. The four phases of service derivation

priorities. This approach is meant as a decision support tool for business and IT managers to quickly spot reuse potential in their company. In this way, the approach aims to speed up derivation drastically, and it can easily scale for involving large sets of process models of the whole company.

The paper is structured accordingly. Section 2 introduces the procedure of service derivation as it is summarized in related research. Section 3 introduces our approach, which builds on automatic techniques for parsing activity labels in process models. Section 4 presents the results of testing our prototypical implementation on a set of roughly 600 process models from practice. Section 5 discusses related work before Section 6 concludes the paper.

2 Background

This section introduces the theoretical background of service derivation. Several of the existing approaches explicitly distinguish between business services and software services. This distinction is brought forth by different perspectives. A business service can be understood as a *specific set of actions that are performed by an organization* [7], while a software service describes a part of an application system that is utilized by several entities independently [2]. The concept of a business service puts more emphasis on the economic perspective, as the software service is more related to information technology. This divide is also apparent in many of the methodological contributions on service derivation [8,9,10]. Typically, the derivation of business services tends to take more of a top-down approach, and the software service derivation is rather bottom-up. It has been shown though that both derivation types have many commonalities. For both service types, business and software services, many authors consider the analysis and evaluation of business process models to be a central step [11,12,13]. Therefore, we will focus on the reuse of process models and abstract from differences between both service types here. Accordingly, we describe service derivation as a four phase approach involving preparation, identification, detailing, and prioritization, similar to the integrated approach of [2]. Figure 1 illustrates this approach.

The derivation of services usually starts with a *preparation phase.* In this phase, an information base for the service analysis is established. This information base may include different types of business documents such as enterprise architectures, business processes or organizational structures. In this paper, we assume that a collection of process models is already available. This is a viable assumption since big and medium-sized companies typically possess hundreds of process models [6]. The subsequent *identification phase* is concerned with identifying capabilities. In process models, these capabilities can be closely related to actions. If required, the available processes have to be further decomposed in order to arrive at a suitable level of detail. In the following *detailing phase*, the relationships and interactions between services are identified. This includes the detection of overlaps with existing services and the proper incorporation of new services into the existing SOA landscape. Finally, the *prioritization phase* is utilized to decide which services should be considered for implementation with which priority.

This four-phase process shows that the issue of scalability is hardly discussed. That is a significant problem when a service-oriented architecture is embraced as a company-wide concept. When starting from a process perspective, this means that dozens of processes have to be modeled. Often, it takes weeks to document only a single process. Even if a big number of process models already exist, it is hardly possible to inspect them manually in a systematic way. Against this background, it is striking that none of the service derivation approaches from the extensive list collected in [2] considers the potential for automation. In the following, we will define an approach that assembles analysis techniques in an innovative way towards this end.

3 Automatic Service Identification and Detailing

This section discusses our approach for the automatic identification and detailing of service candidates from process models. The basis for our algorithm is a set of process models P where each process model p is characterized by set of activities A. An activity a is further defined as a combination of an action an and a business object bo on which this action is performed. As an example consider the activity *Verify Invoice* which contains the action *verify* and the business object *invoice.* The union of all activity sets A from the model collection P is denoted with A_P. In order to *identify* an ordered list of service candidates S from all activities A_P we introduce a two-phase approach. In the first phase we parse all activities contained in A_P and annotate them with their according action and business object. In the second phase, we employ different strategies to identify a list of service candidates from these activities. Finally, we use behavioral profiles for a *detailing* of the service. The following sections introduce both phases in detail.

3.1 Annotation of Process Model Activities

The goal of this phase is the precise annotation of activity labels with action and business object. In order to accomplish this, we employ a technique developed in prior work [14]. This technique builds on the insight that activity labels follow regular structures, so called label styles. The most frequent label styles are the verb-object and the action-noun style. The verb-object style is characterized by an imperative verb in the beginning which is followed by the business object.

Examples are *Notify Customer* or *Print Document.* In activities belonging to the action-noun style the action is not given as verb, but captured as a noun. As examples consider *Order Shipment* or *User Registration.* These examples illustrate that the structural knowledge about the label styles enables the proper extraction of action and business from activities. Accordingly, the annotation phase is further subdivided into two main steps: recognition of activity labeling style and derivation of action and business object from activity labels.

Recognition of Label Style: The first step is the correct recognition of the activity label style. Thereby, it is important to appropriately cope with the typical challenges of activity labels. This includes the lack of a rich sentence structure and also the zero-derivation ambiguity. The latter refers to misinterpretations due to the fact that one syntactic word can be interpreted as verb or noun (e.g. *the plan* and *to plan*). In order to adequately determine the label style, we designed a recognition algorithm which analyzes different stages of the label context [14]. As an example, consider the activity label *Plan Data Transfer* from the SAP Reference Model. By solely analyzing the label, it is not possible to decide about the label style. The activity could either instruct to *plan* a *data transfer* or to *transfer* a record of *plan data.* However, by broadening the context and considering the whole process model collection, we can learn that many other processes from the collection deal with the business object *plan data.* Accordingly, the label is classified as an action-noun label. In cases where the context of the process model collection is not sufficient, we use a word frequency list to decide whether the first word in the label is more likely to be a verb or a noun. Accordingly, it is categorized as a verb-object or action-noun label.

Derivation of Action and Business Object: The second step in the annotation phase is the actual derivation of action and business object from the activity label. Therefore, we make use of the structural knowledge about the label styles. Thus, we know that a verb-object label begins with an imperative verb which is followed by a business object. Accordingly, the verb-object label *Contact Customer* can be easily decomposed into the action *contact* and the business object *customer.* In the same vein, we derive action and business object from action-noun labels. As an example consider the activity *Credit Status Analysis.* Being aware that this is an action-noun label, we know that the action is given as a noun at the end of the label. By using the lexical database WordNet [15] we can derive the verb *analyze* from the nominalized action *analysis.* The business object is respectively specified with *Credit Status.*

3.2 Identification of Service Candidates

At this stage the action and business object from all activity labels of the considered process model collection are adequately determined. Building on this annotation information, we introduce three different approaches to identify service candidates. The following paragraphs introduce each approach in detail.

Atomic Service Identification: The atomic service identification strategy focuses on single activities and is based on the notion that reoccurring activities

Algorithm 1. Atomic Service Identification

```
1: **List** $candidates$ = $new\ List()$;
2: **for each** activity $a \in A_P$ **do**
3:     $F_A$ = countFrequency($a$,$A_P$);
4:     **if** $F_A \geq 2 \wedge candidates$.contains($a$) = **false then**
5:         $a$.setFrequency($F_A$);
6:         $candidates$.add($a$);
7: $candidates$.orderByFrequency();
```

are likely to represent relevant service candidates. This approach is in line with the viewpoint of [16] that each activity in a process model can be considered as a potential service. Consequently, the frequency of a particular activity throughout the model collection determines its potential of being a suitable service candidate. In order to capture these considerations we introduce the activity frequency metric F_A, which determines the number of similar activities in a process model collection for a given activity. Thereby, the similarity between two activities is based on the congruence between their actions and business objects. Accordingly, activities following a different label style, such as *Notify Customer* and *Customer Notification*, are still considered as equal activities.

The details if the atomic service identification approach are illustrated in Algorithm 1. In order to identify services candidates for a whole process model collection, we compute F_A for each activity in the collection P (lines 2-3). If the frequency F_A of an activity is equal or greater than two and the activity has not been considered in a previous iteration, the activity is added to the candidate list (lines 4-6). After all activities have been analyzed the candidates are ordered according to their frequency (line 7). As a result, we obtain a list of candidates ordered by their potential of being suitable service candidates.

Composite Service Identification: The Composite Service Identification approach aims for identifying composite service candidates based on business object groups. Hence, it abstracts from single activities and focuses on activity groups having the same business object. For each business object grouping we introduce the frequency F_{BO} that determines the relevance of that group based on the occurrence of the business object among all activities of the model collection.

Algorithm 2 provides an algorithmic description for identifying composite service candidates. First, for each activity the frequency of the business object is determined (lines 2-4). In case the frequency of a considered business object is equal or greater than two and the activity - business object combination has not been stored in previous iterations, the combination is added to the group candidate map (lines 5-7). After ordering the business object groups according to their frequencies (line 8), we obtain a list of composite services candidates ordered by their relevance.

Inheritance Hierarchy Identification: The Inheritance Hierarchy Identification approach is based on the considerations of the Composite Service Identification strategy. However, it extends this approach by taking hierarchical

Algorithm 2. Composite Service Identification

```
1: **Map** $groupCandidates = new\ Map()$;
2: **for each** activity $a \in A_P$ **do**
3:     $bo = a$.getBusinessObjectFromAnnotation();
4:     $F_{BO}$ = countFrequency($bo$,$A_P$);
5:     **if** $F_{BO} \geq 2 \wedge groupCandidates$.contains($bo$,$a$) = **false then**
6:         $bo$.setFrequency($F_{BO}$);
7:         $groupCandidates$.add($bo$,$a$);
8: $groupCandidates$.orderByFrequency();
```

relationships between the business objects into account. This is motivated by the design principle of service cohesion [17] that refers to the degree of relatedness between the operations of a service. Assuming that activities with related business objects may also lead to related services, we aim for identifying business object hierarchies. In order to identify relationships between business objects, we decompose the business object terms. As an example consider the business object *purchase order*. Apparently, the word *purchase* is a specification of the main word *order* at the end. Hence, a hierarchy can be constructed by relating different parts of the business objects. For computing the relevance of such a hierarchy group we introduce the metric F_{IH} which is based on the occurrence of the main word among all business objects. The identified hierarchy groups can then be used for constructing according composite services which explicitly respect the notion of service cohesion.

Algorithm 3 illustrates the details of this approach. The basis of the hierarchy consideration are business objects which contain more than one word (line 4). If such a business object is identified, we determine the frequency of its main word among all activities (line 5). In case the frequency of the main word is equal or greater than two and no respective hierarchy tree exists, a new tree with the main word as a root node is created (lines 7-9). Afterwards, all possible business object parts are computed (lines 10-12). This is accomplished by iteratively complementing the first word of the business object until we finally obtain the original business object. Each business object part having a frequency greater or equal than two is inserted as a node on the according hierarchy level (lines 13-14). Finally, the hierarchy trees are sorted according to the frequency of the main word (line 15).

3.3 Detailing of Service Candidates

Service detailing refers to the definition of the structure and behaviour of a service, or a set of services. To this end, we adopt an approach for mining action patterns. Action patterns define recurring behaviour [18]. The conceptual foundation for action patterns are so-called behavioural profiles. Our approach takes a collection of process models as a starting point for deriving action patterns of a specific business object. From these patterns, we can use synthesis techniques in order to arrive at a process model that details the lifecycle of a service candidate. Therefore, this section defines the notion of a behavioural profile, explains how

Algorithm 3. Inheritance Hierarchy Identification

```
1: **TreeList** $hierarchies$ = $new\ TreeList()$;
2: **for each** activity $a \in A_P$ **do**
3:     $bo = a$.getBusinessObjectFromAnnotation();
4:     **if** $bo.getWordCount() > 1$ **then**
5:         $mainWord = bo.words[bo.getWordCount()]$;
6:         $F_{IH}$ = countFrequency($mainWord$,$A_P$);
7:         **if** $F_{IH} \geq 2 \wedge hierarchies$.containsTree($mainWord$) = **false then**
8:             $mainWord$.setFrequency($F_{IH}$);
9:             $hierarchies$.createNewTree($mainWord$);
10:        **for** $i$= 1 **to** $bo.getWordCount()$) **do**
11:            $term = bo.words[1] + \ldots + bo.words[i]$;
12:            $F_{IH}$ = countFrequency($term$,$A_P$);
13:            **if** $F_{IH} \geq 2$ **then**
14:                $hierarchies$.getTree($mainWord$).addNode($term$, $i$);
15: $hierarchies$.orderByFrequency();
```

action patterns can be identified, and how a process model showing the service lifecycle can be found.

Behavioural Profiles: With our approach, we aim to identify service candidates that are utilized in various processes in a company. In order to detail such a service, we have to extract its behavioral constraints from different process models, and consolidate them in an appropriate way. So-called *behavioral profiles* capture such constraints on the level of pairs of activities. A behavioral profile builds on trace semantics for a process model, namely the weak order relation [19]. It contains all pairs (x, y) if there is a trace in which x occurs before y. For a process model p, we write $x \succ_p y$. The behavioral profile then defines a partition over the cartesian product of activities, such that a pair (x, y) is in one of the following relations:

- strict order relation $\rightsquigarrow_p$, if $x \succ_p y$ and $y \not\succ_p x$;
- exclusiveness relation $+_p$, if $x \not\succ_p y$ and $y \not\succ_p x$;
- interleaving order relation $||_p$, if $x \succ_p y$ and $y \succ_p x$.

Based on this behavioral profile, we can define the behavioral constraints of a service candidate.

Action Patterns: Once we have derived the behavioral profile relations from a set of process models, we can determine the support and confidence of action patterns. This works similar to association rule mining. For each pair of activities that co-occur in one of the process models, we map them to their behavioral profile relation. Accordingly, a behavioral action pattern can be defined as a rule R with a minimum support and confidence value [18] such that:

- R is a rule $X \Rightarrow Y$, where $X, Y \subset V \times \{\rightsquigarrow, \rightsquigarrow^{-1}, +, ||\} \times V$, such that X and Y are pairs of actions for which behavioral relations are specified;
- *minsup* is the value of the required minimal support;
- *minconf* is the value of the required minimal confidence.

Such a pattern typically captures the relationship between actions, i.e. verbs mentioned in the activity labels. We can define object-sensitive action patterns if we only consider actions of the same business object. In our context, such object-sensitive action patterns provide the basis for detailing the lifecycle of a service candidate.

Synthesis of Service Lifecycle: The remaining challenge is to define a process model that matches the behavioral relationships of the service candidate. A corresponding synthesis technique has been defined in [20]. The idea is to identify the consistent set of behavioral relations. From these relations, we can construct a process model. The strict order relation defines the skeleton of a corresponding process model. Activities that are not in an order relation are organized in nested XOR- or AND-blocks depending on whether they are exclusive or interleaving. The notion of profile consistency guarantees that such a nesting exists [20].

4 Evaluation

To demonstrate the capability of our service identification approach, we conduct an evaluation with real-world data. In particular, we designed a test collection that contains the activity labels of the SAP Reference Model. The SAP Reference Model is a collection of Event-Driven Process Chains (EPCs) and captures the business processes supported by the SAP R/3 system in its version from the year 2000 [21, pp. 145-164]. It is organized in 29 functional branches as for instance sales and accounting and contains 604 process models with in total 2433 activity labels. In the following paragraphs we present the results for each of the in subsection 3.2 introduced concepts.

4.1 Activity Frequency Results

For obtaining atomic service candidates we computed the metric F_A for each activity in the process model collection. As a result, we identified 464 activities with a frequency of at least two. Twelve of these activities even had a frequency of equal or greater 10. Table 1 gives an overview of the top 5 ranked results.

The results show that it is still necessary to evaluate whether an identified candidate is suitable for being established as a service. In addition, we must decide whether a candidate can be established as a business or as a software service. For instance *Process Goods Issue*, *Billing* and *Planning* are more likely to represent business services, while *Calculate Overhead* and *Difference Processing* could be automatable activities and are hence candidates for software services.

4.2 Business Object Frequency Results

Based on the consideration of the metric F_{BO}, we identified 378 business object groups. Thereby, each business object appeared at least twice among all activities of the model collection. In 27 of these groups the business object was used ten times or more. Table 2 shows the top 5 ranked business objects groups. In

Table 1. Results for Atomic Service Identification

Rank	**Activity**	F_A
1	Process Goods Issue	20
2	Calculate Overhead	17
3	Billing	13
4	Planning	13
5	Difference Processing	13

Table 2. Results for Composite Service Identification

Rank	**Business Object**	**Example Actions**	F_{BO}
1	Order	Execute, Settle, Archive, Release, Print	48
2	Time Sheet	Report, Permit, Process, Approve, Create	23
3	Invoice	Release, Verify, Process, Receive, Reverse	23
4	Budget	Release, Plan, Update, Allocate, Return	23
5	Posting	Perform, Release, Direct	19

addition to the rank and the value of F_{BO}, it also provides the most frequent actions which are applied on these business objects in the analyzed models.

4.3 Inheritance Hierarchy Results

To extract a business object hierarchy, we derived the different parts for each business object and computed their frequencies. In this way, for instance the business object *Service Product Order* was first reduced to *Product Order* and then to *Order*. Whenever a part term was identified twice, a new node in the business object hierarchy was created. Taking the given example, only a new node for *Order* is introduced as the term *Product Order* only appeared once among all activities of the model collection. By computing the metric F_{IH} for each main word, we obtain a ranked list of business object hierarchies.

In total, we identified 362 business object hierarchies where the main word was at least found twice among all models. In 70 hierarchies the root term was used 10 times or more. Table 3 shows the top 5 ranked business object hierarchies including the main word and three frequent example nodes.

4.4 Determination of Internal Service Structure

In order to determine the internal structure of a composite service, we computed the behavioural profile for the comprised activities. Table 4 shows the behavioural profile for the composite service *order*. This profile illustrates that there exists a well-defined order in which the activities must be executed. Apart from the activity *print*, which can be performed at any time after the activity *process*, the order is strict. The synthesis of this profile yields the process model depicted in Figure 2.

Table 3. Results for Composite Service Identification

Rank	**Main Word**	**Example Nodes**	F_{IH}
1	Order	Business Order, Sales Order, Service Order	112
2	Data	Plan Data, Transaction Data, Time Sheet Data	51
3	Cost	Plan Cost, Process Cost, Shipment Cost	46
4	Request	Payment Request, Recruitment Request	30
5	Document	Billing Doc., Customer Doc., Customer Doc.	29

Table 4. The behavioural profile for the composite service *order*

	create	*process*	*print*	*release*	*execute*	*permit*	*settle*	*complete*
create		$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
process			$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
print				$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
release					$\rightsquigarrow$	$\rightsquigarrow^{-1}$	$\rightsquigarrow$	$\rightsquigarrow$
execute						$\rightsquigarrow^{-1}$	$\rightsquigarrow$	$\rightsquigarrow$
permit							$\rightsquigarrow$	$\rightsquigarrow$
settle								$\rightsquigarrow$

5 Related Work

The work presented in this paper relates to approaches for service identification and the application of natural language processing for process models.

Service identification has been considered for business services and software services [2]. The identification of business services is mainly discussed by papers from practice [22,23]. Those typically build on the analysis of business entities and components. By contrast, the derivation of software services is widely discussed in research. Some authors propose a bottom-up approach by analyzing existing legacy systems and building on their functionality [24,25]. Others suggest a top-down strategy [9,26]. However, most approaches proposes to strike a balance between the two extremes. As a result, plenty of software service identification approaches include the analysis and evaluation of business process models [11,12,13].

Techniques for natural language processing have been applied on process models in various contexts. One important application scenario is the assurance of linguistic quality aspects in process models. With this intention NLP tools were employed for identifying semantic errors in activity labels [27] and for constructing

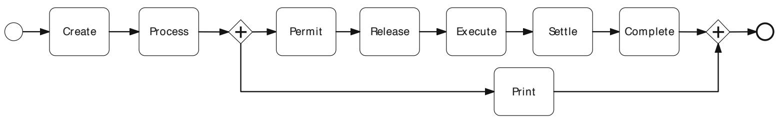

Fig. 2. The process model for the composite service *order*

a glossary from process model collections [28]. NLP tools were also used to refactor whole process model collections in order to ensure the understandability of the comprised activity labels [14]. As the latter requires the decomposition of the activity label into its components, this technique also constituted the basis for the approach presented in this paper. Other prominent applications of NLP techniques are the identification of similarities between process models [29,30] and the derivation of process models from natural language texts [31,32,33].

6 Conclusion

In this paper we presented an approach for the automatic derivation of service candidates from process models. We built on analysis techniques for process models and proposed three different techniques for deriving service candidates. We tested our approach on a process model collection from practice including 600 EPCs with 2433 activities. The evaluation illustrates the capability of our algorithm to provide useful information for service derivation. Our technique does not only enable companies to take an extensive number of process models into account, but also to efficiently analyze them. Considering the results it is important to emphasize that our technique cannot completely automate the service derivation procedure, as the final decision making remains a human task.

In future research, we plan to extend our technique by incorporating lexical relationships such as synonymy and homonymy. Further, we aim to apply our technique in the context of an industrial case study. In this way, we aim for determining the applicability and the significance of each of the identification strategies. In response to the findings, the proposed approach could then be adapted to the specific needs from practice.

References

1. O'Sullivan, J., Edmond, D., ter Hofstede, A.H.M.: What's in a service? Distributed and Parallel Databases 12(2/3), 117–133 (2002)
2. Kohlborn, T., Korthaus, A., Chan, T., Rosemann, M.: Identification and analysis of business and software services - a consolidated approach. IEEE T. Services Computing 2(1), 50–64 (2009)
3. Hafeez, K., Zhang, Y., Malak, N.: Determining key capabilities of a firm using analytic hierarchy process. Int. J. of Production Economics 76(1), 39–51 (2002)
4. Homann, U., Tobey, J.: From capabilities to services: Moving from a business architecture to an it implementation (2006)
5. Zimmermann, O., Krogdahl, P., Gee, C.: Elements of service-oriented analysis and design. IBM developerworks (2004)
6. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal 12(2), 249–254 (2006)
7. Feuerlicht, G.: Design of service interfaces for e-business applications using data normalization techniques. Inf. Syst. E-Business Management 3(4), 363–376 (2005)
8. Bell, M.: Service-oriented modeling. Service Analysis, Design and Architecture. John Wiley and Sons, Hoboken (2008)
9. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)

10. Ramollari, E., Dranidis, D., Simons, A.: A survey of service oriented development methodologies. 2nd europ. young researchers WS on service oriented comp. (2007)
11. Azevedo, L.G., Santoro, F., Baião, F., Souza, J., Revoredo, K., Pereira, V., Herlain, I.: A Method for Service Identification from Business Process Models in a SOA Approach. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 99–112. Springer, Heidelberg (2009)
12. Klose, K., Knackstedt, R., Beverungen, D.: In: Identification of Services - A Stakeholder-Based Approach to SOA Development and its Application in the Area of Production Planning. University of St. Gallen (2007)
13. Erradi, A., Kulkarni, N., Maheshwari, P.: Service Design Process for Reusable Services: Financial Services Case Study. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 606–617. Springer, Heidelberg (2007)
14. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems (forthcoming, 2012)
15. Miller, G.: WordNet: a Lexical Database for English. CACM 38(11), 39–41 (1995)
16. Inaganti, S., Behara, G.K.: Service identification: BPM and SOA handshake. BPTrends (2007)
17. Papazoglou, M.P., Heuvel, W.V.D.: Service&#45;oriented design and development methodology. Int. J. Web Eng. Technol. 2, 412–442 (2006)
18. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process model repositories. Computers in Industry 63 (2012)
19. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE T. Softw. Eng. 37(3), 410–429 (2011)
20. Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on synthesis from consistent behavioural profiles. Int. J. Coop. Inf. Sys. 21 (2012)
21. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley (1998)
22. IBM: Component business models (2005)
23. SAP: Enterprise service design guide (2005)
24. Sneed, H.M.: Integrating legacy software into a service oriented architecture. In: IEEE Conference on Software Maintenance and Reengineering, pp. 3–14 (2006)
25. Belushi, W.A., Baghdadi, Y.: An Approach to Wrap Legacy Applications into Web Services. In: Int. Conf. on Service Systems and Service Management, pp. 1–6 (2007)
26. Flaxer, D., Nigam, A.: Realizing business components, business operations and business services. In: Proceedings of IEEE CEC-EAST, pp. 328–332 (2004)
27. Gruhn, V., Laue, R.: Detecting Common Errors in Event-Driven Process Chains by Label Analysis. Enterprise Modelling and Inf. Systems Arch. 6(1), 3–15 (2011)
28. Peters, N., Weidlich, M.: Automatic Generation of Glossaries for Process Modelling Support. Enterprise Modelling and Inf. Systems Architectures 6(1), 30–46 (2011)
29. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36, 498–516 (2011)
30. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: APCCM 2007, vol. 67, pp. 71–80 (2007)
31. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011)
32. de AR Gonçalves, J.C., Santoro, F.M., Baião, F.A.: Business Process Mining from Group Stories. In: CSCWD 2009, pp. 161–166. IEEE Computer Society (2009)
33. Sinha, A., Paradkar, A.: Use Cases to Process Specifications in Business Process Modeling Notation. In: IEEE Int. Conference on Web Services, pp. 473–480 (2010)