Automatic Service Derivation from Business Process Model Repositories via Semantic Technology

Henrik Leopold[a], Fabian Pittke[b], Jan Mendling[b]

[a] *VU University Amsterdam, The Netherlands*
[b] *WU Vienna, Austria*

Abstract

Although several approaches for service identification have been defined in research and practice, there is a notable lack of fully automated techniques. In this paper, we address the problem of manual work in the context of service derivation and present an approach for automatically deriving service candidates from business process model repositories. Our approach leverages semantic technology in order to derive ranked lists of useful service candidates. An evaluation of the approach with three large process model collection from practice indicates that the approach can effectively identify useful services with hardly any manual effort. The evaluation further demonstrates that our approach can address varying degrees of service cohesion by applying different aggregation mechanisms. Hence, the presented approach represents a useful artifact for enabling business and IT managers to quickly spot reuse potential in their company. In addition, our approach improves the alignment between business and IT. As the ranked service candidates give a good impression on the relative importance of a business operation, they can provide companies with first clues on where IT support is needed and where it could be reduced.

Keywords: Service Derivation, Business Process Model Repositories, Semantic Technologies, Natural Language Analysis

1. Introduction

Service-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 is 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, for instance, the degree of coupling between business and IT architecture is reduced and the effect of changes becomes more transparent.

Email addresses: `h.leopold@vu.nl` (Henrik Leopold), `fabian.pittke@wu.ac.at` (Fabian Pittke), `jan.mendling@wu.ac.at` (Jan Mendling)

In the past, many approaches for identifying services from various input sources have been defined [2]. Among others, these approaches leverage requirements documents [3, 4], different types of conceptual models [5, 6, 7], and source code [8, 9, 10]. Due to their general availability, particularly process models have turned out as a valuable source for identifying services. However, a core problem is that many of the approaches building on process models focus only on single steps of the service derivation process and still require a considerable amount of human effort to derive a promising service candidate. Taking the large size of process model repositories in practice into account [11, 12], this means that these techniques do not scale up to the size of a whole company as the manual inspection of hundreds or even thousands of process models does not represent a feasible solution. Hence, there is a strong need for techniques that can automate the process model-based service identification as far as possible.

In this paper, we follow up on a proposal to address the problem of manual work in the phases of service derivation based on process models [13]. We consider a situation in which an extensive set of process models is available that describes the company's processes at a operational level [47, 11]. To provide a technique that is widely applicable, we do not require any complementary data or artifacts beyond the process model collection. By building on the methodological considerations from [14], we present an approach for the automatic derivation of service candidates, augmented with a set of metrics that provide initial clues about priorities. Recognizing that process models in practice are often suffering from heterogeneous terminology [15], our approach extends the work from [13] by leveraging semantic technology. As a result, we are able to derive services based on semantic relationships and are no longer bound to trivial string comparisons. The 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 service derivation drastically, and to easily scale up to large sets of process models of the whole company. We use three large process model collections from practice to demonstrate the capabilities of our approach.

The rest of the paper is structured as follows. In Section 2, we give an overview of existing service derivation approaches and discuss shortcomings of available approaches. In Section 3, we present the service derivation approach on a conceptual level. Section 4 discusses the results of testing our prototypical implementation on three large process model collections from practice. Section 5 discusses the implications of our work. Finally, Section 6 elaborates on the limitations of the presented approach before Section 7 concludes the paper.

2. Background

In this section, we provide an introduction into the topic of service derivation. First, in Section 2.1, we give an overview of current service derivation approaches. Subsequently, in Section 2.2, we point out the shortcomings of current approaches in order to highlight the need for a novel process model-based service

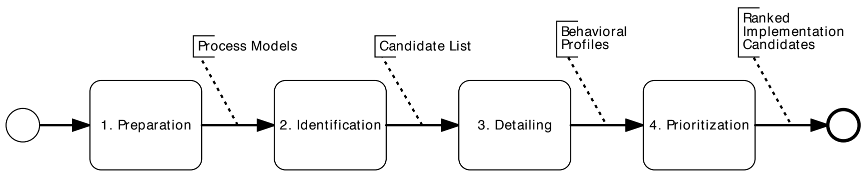


Figure 1: The Four Phases of Service Derivation , adapted from [2]

derivation technique.

2.1. Service Derivation Approaches

For deriving services, various approaches have been proposed in prior research. Many of them explicitly differentiate between business and software services. This distinction is brought forth from different perspectives. A business service is understood as a *specific set of actions that are performed by an organization* [16], 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. Typically, the derivation of business services tends to take more of a top-down approach, i.e., it starts with a rather general analysis of the business by, for instance, investigating the value chain [17, 18, 19, 20]. The derivation of software services, by contrast, is rather bottom-up. It often builds on the analysis of concrete artifacts such as source code or system components. This distinction between business and software services is also apparent in many of the methodological contributions on service derivation.

The methodological contributions can also be considered from the perspective of the overall service derivation process [2]. Figure 1 depicts the service derivation process consisting of four phases: preparation, identification, detailing, and prioritization. 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, organizational structures, or business processes. The subsequent *identification phase* is concerned with identifying capabilities and service candidates. 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 and with which priority.

Table 1 gives an overview of existing service derivation approaches. On the one hand, it shows the main input type, the degree of automation, and whether the approach is targeting software services (SWS) or business services (BS). On the other hand, we also consider the capabilities of the respective approach to support specific phases of the service derivation process. Altogether, we differentiate between three main input types for service derivation techniques: conceptual models, application data, and general requirements

Table 1: Overview of Service Derivation Approaches

Main Input	Approach	Automation	Type	Phases
Conceptual Models				
Process Models	Azevedo et al. [5]	None	BS + SWS	2
Process Models & Appl.	Erradi et al. [21]	None	SWS	2+3
Process & Org. Models	Jamshidi et al. [22]	None	SWS	2
Process Models	Klose et al. [23]	None	BS + SWS	2
Process Models	Sewing et al. [24]	None	BS + SWS	4
Process Models	Zimmermann et al. [25]	None	SWS	2
Process & Ent. Models	Kohlmann and Alt [26]	None	SWS	2
Process Models	Bianchini et al. [27]	Partial	SWS	2,3
Process Architecture	Dwivedi and Kulkarni [28]	Partial	SWS	2
Process Models	Kleinert et al. [29]	Partial	BS + SWS	2
Process Models	Yousef et al. [30]	Partial	SWS	2
Goal-Scenario Models	Kim et al. [31]	None	BS + SWS	2
Data Flow Diagram	Yun et al. [6]	None	SWS	2
Object Models	Jain et al. [32]	Partial	SWS	2
Use Case Models	Kim and Doh [33]	Partial	SWS	2
Feature Model	Lee et al. [7]	Partial	SWS	2
Application Data				
Legacy Code	Aversano et al. [8]	Partial	SWS	2,3
Legacy Code	Chen et al. [9]	Partial	SWS	2,3
Legacy Code	Sneed [34]	Partial	SWS	2,3
Legacy Code	Zhang et al. [10]	Partial	SWS	2,3
Legacy Code	Zhang and Yang [35]	Partial	SWS	2,3
Database Applications	Baghdadi [36]	Partial	SWS	2,3
User Interface Designs	Mani et al. [37]	Partial	SWS	2
General Requirements & Capabilities				
Stakeholder Requirements	Adamopoulos et al. [3]	None	SWS	2
Stakeholder Requirements	Chang and Kim [4]	None	SWS	2
Business Capabilities	Arsanjani and Allam [38]	None	BS + SWS	2
Business Entities	Flaxer and Nigam [39]	None	BS + SWS	2

and capabilities.

In general, service derivation techniques building on *conceptual models* pursue the strategy of clustering the comprised elements in order to obtain a set of service candidates. Depending on the type of the conceptual model, different techniques are employed. One of the most frequently used artifacts in this context is the process model. Against the background of the multitude of automatic process model analysis techniques, it is surprising that many of the identification techniques building on process models suggest a manual analysis (see [5, 22, 23, 26, 25, 24, 21]). Typically, they propose different heuristics on how to derive service candidates from process model activities. For instance, Azevedo et al. [5] suggest the grouping of activities based on the model structure. By contrast, Klose et al. [23] define an evaluation template for considering each activity in detail. Although many techniques build on the manual analysis, there are also techniques available providing automatic support. For instance, Yousef et al. [30] use an ontology to generate a service model from a set of process models. Similarly, Bianchini et al. [27] employ the lexical database WordNet and a reference ontology to identify component services. Kleinert et al. [29] take a different perspective by employing a tree-based structure, a so-called Refined Process Structure Tree (RPST), to identify process regions that represent suitable service candidates. Dwivedi and Kulkarni [28] introduce heuristics that also consider the hierarchical relationships among processes. Even though these approaches make use of automated techniques, the scope of the automation is limited to a particular set of steps. Moreover, they cover the phase of service identification only.

In addition to the numerous derivation techniques focusing on process models, there are also approaches building on other conceptual models. For instance, Lee et al. [7] derive services from feature models by grouping features according to their binding time. In a similar way, Jain et al. [32] employ a spanning tree and group the classes of object models in order to identify web service candidates. An alternative approach is taken by Kim and Doh [33]. They define a rule-based approach to derive service interfaces from use-case diagrams. Similar to process model-based approaches, the latter techniques only partially support the automatic derivation of services with algorithms or tools and solely focus on the service identification phase. Approaches without any automation are introduced by Yun et al. [6] and Kim et al. [31]. Yun et al. [6] propose guidelines to extract flow relations from data flow diagrams. The consolidated flow relations are then used for defining services. Likewise, Kim et al. [31] present guidelines for deriving services from goal models.

Service derivation techniques building on *application data* typically have a high degree of automation. Moreover, this data also leverages capabilities to detail the identified service candidates. One of the most prominent strategies in this context is the automated analysis and service derivation from source code [8, 9, 34, 10, 35]. Although user interaction is required at different stages, all source code based approaches provide support in terms of selecting or grouping algorithms. Besides the multitude of service derivation approaches utilizing source code, there are also other strategies. For instance, Baghdadi [36] analyzes the schemas of relational database applications to define web services. In a quite similar vein, Mani et al. [37] investigate the user interfaces of applications to derive requirements for information services.

In addition to the various approaches building on a particular input artifact such as process models or source code, there are also many service derivation methods that build on *general requirements or capabilities*. These approaches are usually more generic in nature and provide guidelines to identify relevant services. However, these approaches are not automated and mainly focus on the service identification task. For instance, Adamopoulos et al. [3] as well as Chang and Kim [4] propose methods to employ a general analysis of service requirements as starting point for service derivation. Similarly, Arsanjani and Allam [38] propose a three-step approach for deriving service candidates from different business capabilities. Flaxer and Nigam [39] are a bit more specific by asking for the analysis of business entities in order to identify business components and services in the next step.

Altogether, the review of existing approaches reveals that there exists a plethora of different service identification approaches building on different inputs. Also, many methods have been defined for process models . However, current techniques have particular shortcomings that impede their application on large

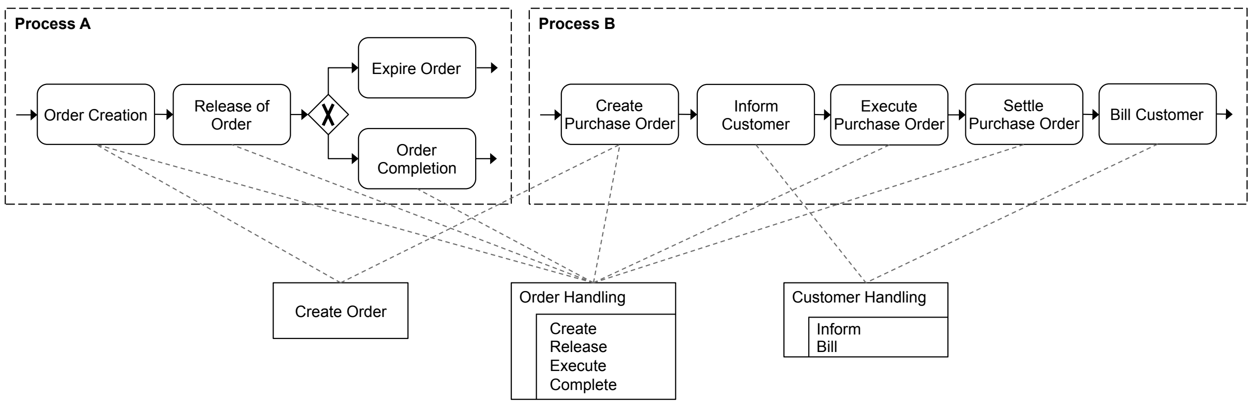


Figure 2: Exemplary Derivation of Service Candidates

repositories of process models. In the following, we elaborate on these shortcomings using an example and highlight the need for a novel service derivation technique based on process models.

2.2. Shortcomings of Process Model-based Service Derivation Approaches

The literature overview has revealed several shortcomings of existing approaches. We elaborate on these shortcomings using the example from Figure 2, which illustrates the service derivation task for two process model fragments. The figure depicts two process model fragments that describe the handling of orders and purchase orders. In Process A, the fragment shows the activity of creating and releasing an order. Then, it has to be decided whether the order has to be completed or to be expired. In process B, we observe a sequential process flow starting with the creation of a purchase order, the informing of the customer, the execution and settlement of the purchase order, and finally the billing of the customer.

Moreover, the example figure shows possible service candidates that can be derived by using the previously mentioned approaches. We highlight the derivation relationship between the process models and the service candidates as a dotted line. For example, the approaches by Azevedo et al. [5] and Klose et al. [23], which concentrate on activity groupings and evaluation templates, would suggest the atomic service *Create Order*. If we employed the approaches by Yousef et al. [30], Bianchini et al. [27], Kleinert et al. [29] or Dwivedi and Kulkarni [28], we would be also able to identify more complex service candidates such as *Customer Handling* or *Order Handling*, which consist of several actions that are performed on a specific object.

However, we identify four main shortcomings with respect to the process-model based service derivation: lack of methodological detail, partial coverage of the service derivation phases, lack of automation, and lack of systematically considering quality issues of process model collections. The first shortcoming, the *lack of methodological detail*, refers to high-level descriptions of many service derivation approaches. Many of the previously introduced approaches do not provide sufficient methodological details. As a result, it remains unclear how these approaches deal with specific characteristics of process models such as structural and linguistic information. Although, for instance, Bianchini et al. [27] employs WordNet for analyzing process model activities, it remains unclear whether they explicitly address the semantic components of activities such as action and business object. Thus, the lack of methodological detail makes it hard to apply these approaches in practice and to reimplement the automated support they provide.

Second, the *partial coverage of the different service derivation phases* refers to the fact that all available approaches only provide a partial solution to the whole service derivation task. Focusing on the process model-based approaches, Table 1 reveals a notable amount of approaches that supports the pure identification of service candidates. However, there are just few approaches that go beyond this step and provide means for detailing these candidates. In fact, only Bianchini et al. [27] provides partial automated support for service detailing by introducing metrics that indicate the relationships among the identified service candidates. Unfortunately, there is no prioritization technique available that supports the user in automatically prioritizing the identified candidates and facilitating the selection. Hence, these steps must be conducted without any guidance from approaches or tools.

Third, the *lack of automation* refers to the extensive amount of manual work that is required by many approaches. In fact, none of the previously discussed approaches considers the potential of fully automating all phases of the service derivation process. In general, a considerable amount of manual work is required to derive a set of services. Particularly, the preparation and the prioritization phase include manual effort. Users are, for instance, asked to align the language of process model activities or to manually annotate activities with necessary information such as input and output [27, 29, 30]. As a consequence, these approaches do not scale up to large organizations when a service-oriented architecture is embraced as a company-wide concept. Moreover, many of the previously discussed approaches do not consider the situation in which large process model repositories are available. However, this is a realistic scenario [11], in which relations between thousands of process models and service candidates as depicted in Figure 2 have to be considered and evaluated by hand.

The fourth shortcoming refers to the *lack of systematical consideration of process model quality issues*. From a general perspective, model quality can be assessed in terms of syntax, semantics, and pragmatics [40, 41]. Syntactical quality is concerned with the correspondence between the process model and the modeling language. Semantic quality is concerned with the completeness and correctness of the process model with respect to the domain. Pragmatic quality ensures correspondence between the model and the interpretation of the audience. In prior research, a plethora of techniques has been proposed to address these issues from a structural perspective (see e.g. [42, 43, 44]) or from a linguistic perspective (see e.g. [45, 15]). Among the discussed service identification techniques building on process models, there is, however, no technique that explicitly takes these quality issues into account. Referring to the example of Figure 2, these techniques do not consider different labeling styles (*Order Creation* vs. *Create Order*) [46], object hierarchies

(*Purchase order* as a specific type of *order*) or terminological inconsistencies (*to settle* as a synonym for *to complete*). As a result, the reliability of the derived service candidates and the validity of the techniques are in question.

Against the background of these findings, we define an approach that addresses the identified weaknesses of existing service derivation approaches. More specifically, we aim at introducing an approach that builds on sound and well-documented concepts (shortcoming 1), covers all phases of the service derivation process (shortcoming 2), provides a large degree of automation (shortcoming 3), and adequately includes quality-preserving measures (shortcoming 4). Our goal is to take a set of process models as input and to automate the steps from preparation to prioritization. To do so, we leverage semantic technology in order to derive non-trivial and useful service candidates. As a result, the manual work in the context of service identification is reduced to the final selection of appropriate services from a ranked list of proposals.

3. Approach for Automatic Semantic Service Identification

This section discusses our approach for the automatic identification and detailing of service candidates from process models. Section 3.1 starts by introducing the formal preliminaries of our approach. Subsequently, Section 3.2.1 through Section 3.2.3 present the details of the introduced service identification approaches.

3.1. Preliminaries

This section introduces the formal preliminaries of the service identification technique. In Section 3.1.1, we discuss formal preliminaries of process models. Then, in Section 3.1.2, we formalize word senses and semantic intersection.

3.1.1. Formalization of Process Models

In this paper, we adapt the canonical process model format from Leopold [48] and introduce a notion of a process model that emphasizes on natural language text. Moreover, we characterize the natural language of process model activities as proposed by Mendling et al. [46]. According to their definition, every label of a process model consists of two components: an action and a business object on which the action is applied. As an example, consider the label *Notify Customer* which contains the action *to notify* and the business object *customer*. It is important to note that activity labels may follow different grammatical structures. For instance, the label *Customer Notification* contains the same action and business object as *Notify Customer* but uses a deviating grammatical structure. Using the technique from [45], action and business object can be automatically extracted, no matter which grammatical structure is used. Capturing actions as verbs and business objects as nouns, we formalize a process model as follows.

Definition 3.1. (Process Model). A process model $P = (A, E, G, F, W_V, W_N, L, \alpha, \beta, \lambda)$ consists of six finite sets A, E, G, W_V, W_N, L, a binary relation $F \subseteq (A \cup E \cup G) \times (A \cup E \cup G)$, a partial function $\alpha : L \nrightarrow W_V$, a partial function $\beta : L \nrightarrow W_N$, and a partial function $\lambda : (A \cup E \cup G) \nrightarrow L$, such that

– A is a finite non-empty set of activities.

– E is a finite set of events.

– G is a finite set of gateways.

– W_V is a finite set of verbs.

– W_N is a finite set of nouns.

– F is a finite set of sequence flows. Each sequence flow $f \in F$ represents a directed edge between two nodes.

– L is a finite set of text labels.

– The partial function α assigns a verb $w_V \in W_V$ to a label $l \in L$.

– The partial function β assigns a noun $w_N \in W_N$ to a label $l \in L$.

– The partial function λ defines the assignment of a label $l \in L$ to an activity $a \in A$, event $e \in E$ or gateway $g \in G$.

As the techniques presented in this paper focus on sets of process models, we further introduce the notion of a process model collection. We denote a process model collection C as being a set of process models P. Moreover, we define the following subsets for a process model collection C:

– The set of all labels: $L^C = \bigcup_{P \in C} L$.

– The set of all actions: $L^C_A = \bigcup_{l \in L^C} \alpha(l)$.

– The set of all business objects: $L^C_{BO} = \bigcup_{l \in L^C} \beta(l)$.

– The set of all actions and business objects $\mathcal{L}^C = L^C_A \cup L^C_{BO}$.

3.1.2. Formalization of Word Senses and Semantic Intersection

In order to automatically identify service candidates based on word semantics, it is necessary to adequately formalize the sense of a word. Thus, we introduce the notion of a *word sense* as being a commonly accepted meaning of a word. For example, we associate the meanings *commercial delivery document* or *request to supply or produce a product* with the single word *order.* We also observe that words typically have a number of different word senses that are dependent on the context of use. For example, it appears to be more appropriate to refer to the word *purchase order* in the sense of a *request for products* than a *commercial document*, if we consider the activity *Execute purchase order* from Figure 2.

For these reasons, we need a comprehensive representation of word senses. In Computational Linguistics, the most widely employed approach is the *enumerative approach* [49]. It includes the identification of discrete word senses and their aggregation into a sense inventory which enables an objective evaluation and comparison of these word senses. Based on this notion, we formalize the connection between words of a dictionary and word senses as follows:

Definition 3.2. (Word Senses). Let D be a dictionary of words. Then, the senses of a word are defined as a function $Senses_D : W \to 2^S$, such that

– $W = W_V \cup W_N$ is the set of words denoted in dictionary D.

– S is the set of word senses that are encoded in the dictionary. 2^S denotes the powerset of senses.

For the extraction of word senses, dictionaries such as the *Oxford Dictionary of English* [50] or the *Longman Dictionary of Contemporary English* [51] are available. What is more, so-called lexical databases can be employed to accomplish this task. A powerful implementation of such lexical databases is *BabelNet* [52]. Similar to WordNet [53], BabelNet organizes nouns, verbs, adjectives, and adverbs into sets of synonyms, so-called *synsets*, that represent a distinct word sense. Moreover, it enriches WordNet senses with Wikipedia concepts which allows a higher coverage of general and domain-specific words [54]. As a result, each word sense of a given word is associated with a synset that can automatically be extracted and processed. For the example of the word *order*, BabelNet retrieves, among others, the following synsets:

– {ordering, order, ordination}

– {order, purchase order}

– {command, statement, instruction, order}

Based on the definition of word senses and BabelNet, we can now compare words on the semantic level and define the concept of *semantic intersection.* Semantic intersection refers to a relation between two words that share an identical word sense. In linguistics, this relation is also known as synonymy. Obviously, two words are considered as semantically intersecting if the sense inventory of the first word overlaps with the sense inventory of the second word. For example, the two business objects *order* and *purchase order* from Figure 2 share the meaning of being a *commercial document.* Thus, we consider these words to be semantically intersecting. Adapting the given definition by Deissenboeck and Pizka [55] to our case, we formally define the semantic intersection as follows:

Definition 3.3. (Semantic Intersection). Let $w_1, w_2 \in W$ be two words and let $Senses_{BN}$ be a function that retrieves all word senses from BabelNet BN. The words w_1 and w_2 are semantically intersecting iff $(Senses_{BN}(w_1) \cap Senses_{BN}(w_2)) \neq \emptyset$.

Building upon the notion of semantic intersection and the techniques to automatically derive actions and business object from activities [48, 45], we now introduce several strategies to automatically identify service candidates.

3.2. Identification and Prioritization 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 information, we introduce three different approaches to derive service candidates. The overall strategy of providing three different techniques is twofold. First, it is useful to exploit the different linguistic aspects that are provided by activity labels. Focusing on the combination of semantically intersecting actions and business objects yields a different result than focusing on syntactically identical business objects only. Second, the approaches allow to address the appropriate level of granularity. Each of the presented techniques increases the aggregation scope such that the first technique identifies the most fine-granular services and the third technique generates the most course-grained services.

3.2.1. Atomic Service Identification

The atomic service identification strategy focuses on single activities and is based on the idea that reoccurring activities are likely to represent relevant service candidates. This approach is in line with the viewpoint that each activity in a process model can be considered as a potential service [56]. Consequently, the number of occurrences of a particular activity throughout the model collection determines its potential of being a suitable and relevant service candidate. In order to operationalize these considerations, we determine the number of semantically intersecting activities in a process model collection. The similarity between two activities is based on the semantic intersection between their actions and business objects as defined in Definition 3.3. As a result, semantically intersecting activities are aggregated into a single service candidate.

The details of the atomic service identification approach are illustrated by Algorithm 1. It starts by defining a new list for the service candidates (line 2). The algorithm then iterates over the set L^C of all activity labels (lines 3-21) and checks whether a semantically identical label is already included in the candidate list (lines 5-16). If this is the case, it is checked whether action or business object of the new candidate represent a syntactically different word. In that case, the respective word is added to the candidate object (lines 8-13). Afterwards, the weight (number of occurrences) of the candidate is increased by one (line 14). If the candidate list does not include a semantically identical label, a new entry is created (lines 17-20). After all activities have been analyzed, the candidates are ordered according to their weight (line 22). As a result, we obtain a list of candidates ordered by their potential of being suitable service candidates.

Algorithm 1: Atomic Service Identification

```
1: computeAtomicServices(ProcessModelCollection $C$)
2: List $candidates$ = new List();
3: for all Label $l \in L^C$ do
4:     boolean $included$ = false;
5:     for all Label $c \in candidates$ do
6:         if isSemanticallyIntersecting($l$.getAction(), $c$.getAction()) $\wedge$
           isSemanticallyIntersecting($l$.getBusinessObject(), $c$.getBusinessObject()) then
7:             $included$ = true;
8:             if $l$.getBusinessObject() $\neq$ $c$.getBusinessObject() then
9:                 $c$.addBusinessObject($l$.getBusinessObject();
10:            end if
11:            if $l$.getBusinessAction() $\neq$ $c$.getAction() then
12:                $c$.addAction($l$.getAction();
13:            end if
14:            $c$.setWeight($c$.getWeight()+1);
15:        end if
16:    end for
17:    if $included$ = false then
18:        $l$.setWeight(1);
19:        $candidates$.add($l$);
20:    end if
21: end for
22: $candidates$.orderByWeight();
```

3.2.2. Composite Service Identification

The Composite Service Identification approach aims at identifying composite service candidates based on business object groups. Hence, it abstracts from single activities and focuses on activity groups having the same or semantically identical business object. For each business object grouping, we determine the weight based on the number of occurrences of the business object among all activities of the model collection.

Algorithm 2 provides an algorithmic description for identifying composite service candidates. In the beginning, a list for the composite candidates is created (line 2). In the subsequent loop, each activity label from the set L^C of all activity labels is inspected (lines 3-19). If the candidate list already contains a candidate with a semantically identical business object, it is checked whether the business object from the current label is syntactically different (lines 6-9). If this is the case, the business object is added to the candidate object (line 10). Afterwards, the weight of the candidate is increased by one (line 12). If the candidate list did not include a semantically identical business object, a new entry is created (lines 15-18). Once all activities were analyzed, simple activities, i.e., candidates with only one action are removed from the list (line 20). Finally, the composite candidates are ordered based on their weight (line 21).

3.2.3. Inheritance Hierarchy Identification

The Inheritance Hierarchy Identification approach is based on the idea of the composite service identification strategy. However, it extends this approach by taking hierarchical relationships between the business objects into account. This is motivated by the design principle of service cohesion [57], which refers to the degree of relatedness between the operations of a service. Assuming that activities with related business

Algorithm 2: Composite Service Identification

```
1: computeCompositeServices(ProcessModelCollection $C$)
2: List $compCandidates$ = new List();
3: for all Label $l \in L^C$ do
4:     boolean $included$ = false;
5:     for all Label $c \in candidates$ do
6:         if isSemanticallyIntersecting($l$.getBusinessObject(), $c$.getBusinessObject()) then
7:             $included$ = true;
8:             $c$.addAction($l$.getAction());
9:             if $l$.getBusinessObject() $\neq$ $c$.getBusinessObject() then
10:                $c$.addBusinessObject($l$.getBusinessObject();
11:            end if
12:            $c$.setWeight($c$.getWeight()+1);
13:        end if
14:    end for
15:    if $included$ = false then
16:        $l$.setWeight(1);
17:        $comCandidates$.add($l$);
18:    end if
19: end for
20: $compCandidates$.removeTrivialCandidates();
21: $compCandidates$.orderByWeight();
```

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 *customer invoice*. Apparently, the word *customer* is a clarification of the main word *invoice* at the end. Hence, a hierarchy can be constructed by relating different parts of the business objects. For computing the weight of such a hierarchy group, we count the occurrence of the main word among all business objects. The identified hierarchy groups can then be used for constructing composite services which explicitly respect the notion of service cohesion. Note that we still include the semantic perspective. Hence, the business object *customer invoice* is not only related to *invoice* but also to the business object *bill*.

Algorithm 3 illustrates the details of this approach. It starts by defining a list for the hierarchical candidates (line 2). Subsequently, each activity label is analyzed in the context of a loop (lines 3-12). In this loop, it is first checked whether the candidate list already contains a label with a semantically identical head word. Therefore, the head words of the current label and the candidate are extracted (lines 5 and 7) and then checked for semantic intersection (line 8). If a semantically identical head word is already part of the list, the remaining words of the current label are incorporated into the tree hierarchy of the candidate (line 9). In case the head words are syntactically not identical, the head word of the current label is added to the candidate (line 12). Afterwards, the weight of the candidate is increased by one (line 14). If the list does not contain a semantically identical head word, a new hierarchy candidate with a weight of one is added to the list (lines 17-18). Once all activities were analyzed, simple activities, i.e., candidates with only one action are removed from the list (line 22). Finally, the hierarchical candidates are ordered according to their weight (line 23).

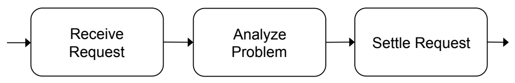
Algorithm 3: Inheritance Hierarchy Identification

```
1: computeHierarchyServices(ProcessModelCollection $C$)
2: List $hierarchyCandidates$ = new List();
3: for all Label $l \in L^C$ do
4:     boolean $included$ = false;
5:     String $l_{head}$ = $l$.getBusinessObject().getHeadWord();
6:     for all Label $c \in candidates$ do
7:         String $c_{head}$ = $c$.getBusinessObject().getHeadWord();
8:         if isSemanticallyIntersecting($l_{head}$,$c_{head}$) then
9:             $c$.addTreeComponents($l$);
10:            $included$ = true;
11:            if $l_{head} \neq c_{head}$ then
12:                $c$.addHeadWord($l_{head}$);
13:            end if
14:            $c$.setWeight($c$.getWeight()+1);
15:        end if
16:        if $included$ = false then
17:            $l$.setWeight(1);
18:            $hierarchyCandidates$.add($l$);
19:        end if
20:    end for
21: end for
22: $hierarchyCandidates$.removeTrivialCandidates();
23: $hierarchyCandidates$.orderByWeight();
```

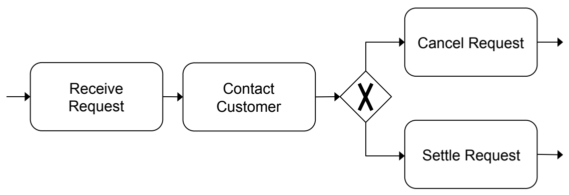
3.3. Detailing of Service Candidates

Service detailing refers to the definition of the structure and behavior of a service, or a set of services. To this end, we adopt an approach for mining action patterns. In general, action patterns define recurring behavior in process models [58]. The conceptual foundation for action patterns are so-called behavioral profiles. Our approach takes a collection of process models as a starting point for deriving action patterns of a specific business object. As an example, consider the two process model fragments from Figure 3 and assume we identified the business object *request* as promising service candidate. Apparently, both process model fragments contain the business object *request* and partially also overlap with regard to the actions that are applied to it. The idea is now to employ synthesis techniques to arrive at a single process model that details the lifecycle of this candidate. Therefore, this section defines the notion of a behavioral profile. Moreover, it explains how action patterns can be identified and how a process model showing the service lifecycle can be found.

Behavioral 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 [59]. 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:



(a) Process Model Fragment 1



(b) Process Model Fragment 2

Figure 3: Exemplary Process Model Fragments for Service Detailing

Figure 4: Behavioral Profiles for Process Model Fragments from Figure 3

	Receive Request	*Analyze Problem*	*Settle Request*
Receive Request		$\rightsquigarrow$	$\rightsquigarrow$
Analyze Problem			$\rightsquigarrow$

	Receive Request	*Contact Customer*	*Cancel Request*	*Settle Request*
Receive Request		$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
Contact Customer			$\rightsquigarrow$	$\rightsquigarrow$
Cancel Request				$+$

- strict order relation $\rightsquigarrow_p$ (x is always executed before y): if $x \succ_p y$ and $y \not\succ_p x$;
- exclusiveness relation $+_p$ (either x or y occurs): if $x \not\succ_p y$ and $y \not\succ_p x$;
- interleaving order relation $||_p$ (x and y can occur in any order): if $x \succ_p y$ and $y \succ_p x$.

Figure 4 shows the derived behavioral profiles for the two process model fragments from Figure 3. Based on such behavioral profiles, we can define the behavioral constraints of a service candidate, i.e., in which order the actions of the service candidate occur.

Action Patterns. Once we have derived the behavioral profile relations from a set of process models, we can analyze how often each of the introduced relations occurs for a given activity pair. If, for instance, an activity x occurs before y in 95% of the analyzed models, this could be considered as a strong indicator for a strict order relationship between x and y. To automatically derive such relations, we build on standard procedures from association rule mining and compute the support and confidence of actions patterns. For each pair of activities that co-occur in one of the process models, we derive their behavioral profile relation. Accordingly, a behavioral action pattern can be defined as a rule R with a minimum support and confidence value [58] such that:

- R is a rule $X \Rightarrow Y$, where $X, Y \subset L_A^C \times \{\rightsquigarrow, \rightsquigarrow^{-1}, +, ||\} \times L_A^C$, 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. As an example consider the two activities *Receive Request* and *Settle Request* in both models from Figure 3. We observe a strict order pattern between the actions *receive* and *settle* in both models. 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 by Smirnov et al. [60]. 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 [60]. Figure 5 illustrates the lifecycle process model for the business object *request* as a result from synthesizing the two process model fragments from Figure 3.

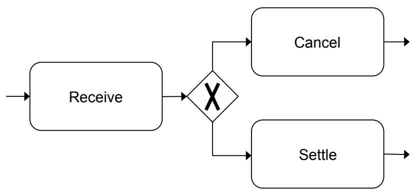


Figure 5: Lifecycle for *Request*, derived from the Process Model Fragments from Figure 3

4. Evaluation

To demonstrate the applicability of the introduced techniques, we conduct an evaluation with real-world process models [14]. The overall goal of the evaluation is to investigate whether the techniques are useful for automatically identifying service candidates from process models. Therefore, we adopt a quantitative as well as a qualitative perspective.

By taking a *quantitative perspective*, we investigate the frequency distribution of the service candidates. In particular, we hypothesize that the number of identified services decreases when we move from the atomic service identification towards the hierarchy service identification. At the same time the weight of the services should increase, i.e., hierarchical services should on average have more instances than composite and atomic services.

Table 2: Details of Used Model Collections

	SAP	**CH**	**TC**
Number of Models	604	328	3,338
No. of Activities	2,433	4,414	35,501
Average No. of Activities per Model	4.03	13.45	4.61
Average No. of Words per Label	3.50	5.59	4.01
Minimum No. of Words per Label	1	1	1
Maximum No. of Words per Label	12	19	24
Notation	EPC	EPC	BPMN

By taking a *qualitative perspective*, we aim at giving the reader an impression of the usefulness of the identified candidates. Hence, we present the most frequent service candidates from each collection and discuss their potential of representing suitable service candidates.

The evaluation section is organized as follows. Section 4.1 presents the test collection of our experiment. Section 4.2 investigates the performance of the service derivation technique. Section 4.3 discusses the results from the service identification by adopting the aforementioned perspectives.

4.1. Test Collection Demographics

For the evaluation of the service candidate derivation technique, we build on model collections from industry. Although the technique can be applied to any given set of process models, particularly industry collections can provide us with insights on the practical usefulness of the identified candidates. Table 2 summarizes the main features of the considered process model collections. They include:

- *SAP Reference Model (SAP)*: The SAP Reference Model represents the business processes of the SAP R/3 system in its version from the year 2000 [61, pp. 145-164]. It contains 604 Event-driven Process Chains (EPCs) which are organized in 29 functional branches such as sales and accounting.
- *Insurance Model Collection (CH)*: The insurance model collection contains 328 business processes from a large Australian insurance company. The contained EPC process models deal with insurance-specific tasks such as the rejection or acceptance of insurance claims and the associated communication with the customer. The insurance model set contains rather large processes with a high density of events.
- *Telecommunication Collection (TC)*: The telecommunication collection contains 3,338 process models covering diverse aspects of the organization. Examples include the handling of customer complaints or contract-related issues. Due to its extensive size, it is well-suited for complementing the test collection.

4.2. Performance Results

In general, the generation of services from process models does not represent a time critical task. A typical application scenario for automatic service derivation is, for instance, an initiative that aims at consolidating

Table 3: Performance Results

	Scope	Annotation	AS	CS	IHS	Total
SAP	Total Time (s)	26.2	111.8	99.3	88.6	325.9
	Time per Activity (s)	0.01	0.05	0.04	0.04	0.13
CH	Total Time (s)	42.1	187.65	153.1	109.5	492.4
	Time per Activity (s)	0.01	0.04	0.03	0.02	0.11
TC	Total Time (s)	344.4	280.51	249.9	189.4	1064.2
	Time per Activity (s)	0.01	0.01	0.01	0.01	0.04

and bundling software functionality. As such an initiative is not likely to be conducted regularly, the derivation of service candidates is a rather non-time-critical task. Nevertheless, we consider computation time to be an important factor, especially to demonstrate the scalability of the technique with respect to large process model collections. We tested the text generation on a MacBook Pro with a 2.4 GHz Intel Core Duo processor and 4 GB RAM, running on Mac OS X 10.7.5 and Java Virtual Machine 1.7. To exclude distortions due to one-off setup times, we ran the derivation twice and considered the second run only. Table 3 summarizes the computation times for the annotation, the atomic service derivation (AS), the composite service derivation (CS), and the inheritance hierarchy service derivation (IHS). In addition, it shows total processing time including annotation and all service identification techniques.

The results demonstrate that the identification of semantic service candidates is associated with a manageable computation effort. The average time for processing a single activity ranges from 0.04 to 0.13 seconds. Depending on the size of the collection, the entire process from annotating the activities to identifying the inheritance hierarchy services hence may take several minutes. The reason for the bigger computation time is the semantic intersection comparison between the activities. However, using BabelNet which provides efficient indexing techniques for quick word sense retrieval, the comparison of word senses is reduced to simple comparison of strings which is executed in few milliseconds. Thus, even for the unusually large and complex TC collection, the total computation is still manageable and can be accomplished within 17 minutes. Since the service derivation is an activity that is not conducted regularly and hence does not represent a time critical task, we consider the computation times to be appropriate for a business scenario.

In addition to the computation time for service derivation, we also investigated the performance time of service detailing. The average time for the detailing of a business object group consumed 0.89 seconds for the SAP collection, 1.45 seconds for the CH collection, and 17.29 seconds for the TC collection. Considering the fact that detailing is typically only conducted for individual service candidates, these run times represent a satisfying performance.

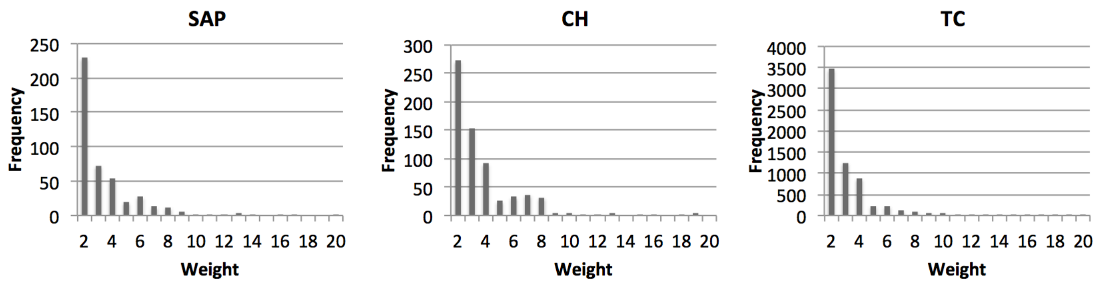


Figure 6: Frequency Distribution of Atomic Services

4.3. Service Derivation Results

This section discusses the results from the service derivation and detailing. Sections 4.3.1 through 4.3.4 present the results from the service derivation. Section 4.3.5 gives an example for the detailing of an identified service candidate.

4.3.1. Atomic Service Results

The quantitative results of the atomic service identification are illustrated in Figure 6. For each collection, the diagrams show the number of services with a weight between 2 and 20. The diagrams illustrate that the collections contain many candidates that occur more than once. Among all investigated process model collections at least 15% of the candidates occur twice or more. This means that at least 15% of the activities are semantically not unique, i.e., there are two or more syntactically or semantically intersecting activities. The large differences in the absolute numbers are the result of the varying size of the process model collections. Thus, the number of services with a weight of 2 amounts to 230 for the SAP collection, 273 for the CH collection, and 3281 for the TC collection. Considering the development from a weight of 2 to a weight of 20, we can see that the number of identified atomic service candidates decreases rapidly. In fact, this effect can be observed for all tree collections. Still, the share of candidates with a weight of 10 or higher amounts to 0.5% for the SAP collection, 0.7% for the CH collection, and 1% for the TC collection. Depending on the complexity of the underlying task, such candidates may represent suitable services.

Table 4 presents the top five atomic service candidates of the considered model collections. For each candidate, we provide the respective action and business object. In case the algorithm identified semantically identical activities, the identical actions and business objects are clustered into a set denoted by curly brackets.

The results illustrate two main aspects. First, they demonstrate that the identified candidates give a good impression of the overall scope of the model collections. This actually emphasizes the relevance of the identified services with respect to the modeled domain. As an example, consider the SAP collection which focuses on goods processing and the associated financial aspects. The top five service candidates for the

Table 4: Results for Atomic Service Identification

	Rank	Candidate Action(s)	CandidateBusiness Object(s)	Weight
	1	Process	Goods Issue	20
	2	{Evaluate, Value, Assess}	-	17
SAP	3	Calculate	Overhead	17
	4	{Transfer, Ship}	-	17
	5	Bill	-	13
	1	Contact	Customer	84
	2	Create	Transaction	63
CH	3	Determine	Claim	36
	4	{Advise, Notify}	Customer	36
	5	Inform	Customer	35
	1	Inform	Customer	187
	2	Request	-	150
TC	3	Identify	Customer	90
	4	Cancel	Order	84
	5	{Manage, Handle}	Incident	83

SAP collection accordingly include related services such as *Process Goods Issue* or *Bill*. For the CH and the TC collections, we observe a similar tendency. As these collections are primarily concerned with the interaction with customers, we observe top candidates like *Contact Customer* or *Inform Customer*. Second, the results indicate the importance of considering word senses. Apparently, *Handle Incident* and *Manage Incident* may refer to exactly the same operation. Hence, it is beneficial to consolidate these semantically identical activities into one atomic candidate. Other examples for semantic clusters which cannot be found in the top five list include *Send E-Mail* and *Send out E-Mail* or *Solve Incident* and *Resolve Incident*.

Although the identified candidates represent important activities within the process model collections, it is still necessary to decide 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*, and *Contact Customer* are more likely to represent business services as they do not represent activities that are likely to be automated. As they are conducted quite frequently, it might be beneficial to consider their standardization and encapsulation as a business service. Candidates that have the potential for automation are, for instance, given by *Bill*, *Cancel Order*, and *Inform Customer*. Hence, they represent good candidates for software services. For example, the billing of a customer could be supported by an application that requires a set of input data and automatically generates an according invoice. Similarly, the cancellation of an order or the notification of a customer can be supported by software services.

4.3.2. Composite Service Results

The quantitative results of the composite service identification are illustrated in Figure 7. It shows the number of identified composite services with a weight between 2 and 30. The diagrams illustrate that the business object grouping significantly reduces the frequency of the identified services. Taking an example

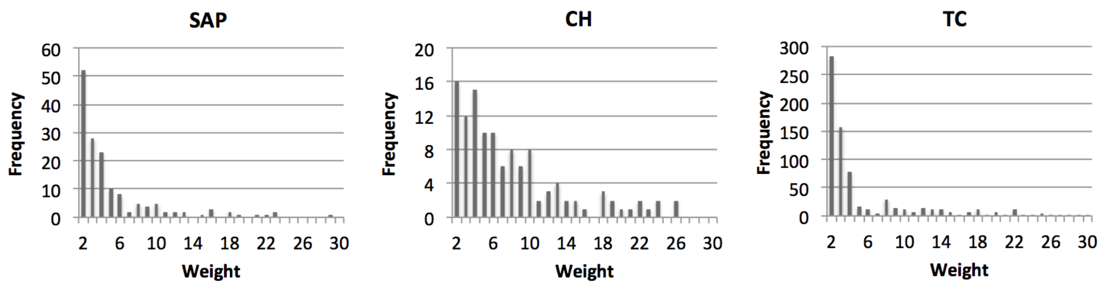


Figure 7: Frequency Distribution of Composite Services

Table 5: Results for Composite Service Identification

	Rank	Object Cluster	Top Action Clusters	Weight	NoA
SAP	1	Order	Execute, Settle, Archive	54	14
	2	{Invoice, Account}	Release, Verify, Process	29	11
	3	Time Sheet	Report, Permit, Process	23	5
	4	Budget	Release, Plan, Update	23	10
	5	{Information, Data}	Transfer, Systematize, Model	22	7
CH	1	Customer	Contact, Inform, Determine	218	12
	2	Claim	Determine, Review, Finalize	123	22
	3	Assessment	Decline, Cancel, Submit	85	10
	4	Transaction	Create, Stop	63	2
	5	{Action, Activity}	Complete, Create, Determine	44	4
TC	1	{Order, Suborder, Purchase Order}	Cancel, Deliver, Complete	742	34
	2	Customer	Inform, Identify, Create	684	35
	3	Case	Create, Dispatch, Accept	359	23
	4	Request	Log, Send, Handle	190	16
	5	Incident	{Handle, Manage}, {Solve, Resolve}, Analyze	159	7

of the SAP collection, we can see that 230 atomic service candidates with a weight of 2 are aggregated to 52 composite service candidates with a weight of 2. Since the algorithm only considers candidates with at least two different actions (i.e., simple activities are excluded), this is an expected result. Despite the overall decrease of the number of candidates, we observe a shift towards candidates with a higher weight, i.e., there are significantly more composite candidates with a weight greater than 10 than atomic services.

Table 5 shows the top five ranked business object groups for each model collection. In addition to the rank and the weight of the business object group, the table also provides the most frequent action clusters and the total number of unique actions (NoA) that are applied on the business objects. The results illustrate that the composite services include business objects and business object groups that are not part of the atomic service list (see Table 4). As examples, consider *budget*, *case*, and the cluster {*action, activity*}. This emphasizes the importance of considering semantic business object groups as service candidates since important business objects might be outnumbered by single activities. The potential of business object groups for representing relevant service candidates is illustrated by the comprised actions. Thinking of the life cycle of a business object such as *order*, it becomes apparent that *execute*, *settle*, and *archive* are core

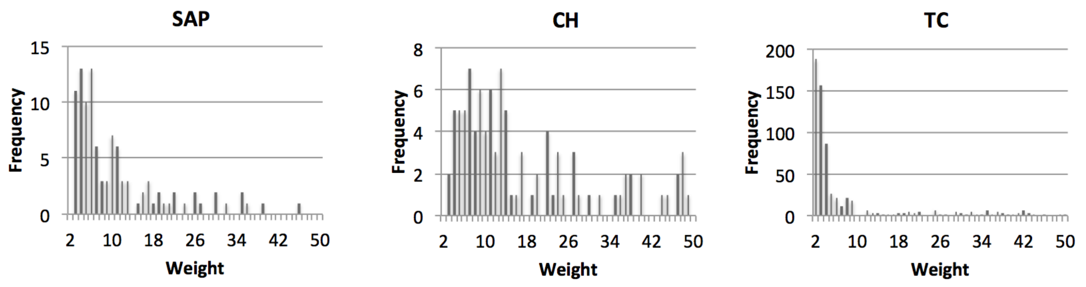


Figure 8: Frequency Distribution of Inheritance Hierarchy Services

tasks in this context. As the composite service derivation automatically derives this information, the user can efficiently evaluate all tasks related to a business object and establish a composite service.

4.3.3. Inheritance Hierarchy Service Results

The quantitative results of the inheritance hierarchy derivation are illustrated in Figure 8. It shows the number of identified hierarchy services with a weight from 2 to 50. In comparison to atomic and composite services, we again observe a shift with respect to the number and the weight of the identified candidates. While the number of candidates with a low weight is decreasing, the number of candidates with a higher weight is increasing. This shift highlights that the aggregation on the main word of the business objects is a valuable strategy for identifying semantically-related candidates subsuming multiple activities.

Table 6 gives an overview of the top-ranked candidates for each collection. It includes the main word, a set of sub node examples, the weight of the candidate, and the total number of distinct sub nodes (NoN). The results show that there are some dominant business objects that are also included in the candidate lists of the other service identification techniques. Examples include the objects *order*, *claim*, and *case*. Apparently, these objects are of central importance for the operations of the investigated processes. However, we also observe that highly-ranked business objects from the other approaches are no longer included. For example, the business object *customer* is no longer among the top-ranked service candidates. This emphasizes the importance of different perspectives. As business objects such as *document* are almost exclusively used in a more specific context (e.g., billing document or customer document), they do not achieve high values in the context of the previously discussed techniques. Despite the reduction to the top node, we still encounter synonym clusters as for instance {information, data} and {task, action, activity}. This again emphasizes the importance of considering semantically related terms.

4.3.4. Comparison of Service Derivation Techniques

The frequency distributions of the different service identification techniques in the previous sections already indicated the effect of the techniques on the number of identified services and their weight. However, in order to get a better impression of this effect, a more detailed comparison is necessary. Figure 9 and 10

Table 6: Results for Inheritance Hierarchy Service Identification

	Rank	Object	Top Nodes	Weight	NoN
	1	Order	Business Order, Sales Order	127	26
	2	{Information, Data}	Plan Data, Benefits Information	53	14
SAP	3	Document	Measurement Doc., Billing Doc.	39	9
	4	{Price, Cost}	Plan Cost, Process Cost, Material Price	36	10
	3	Plan	Budget Plan, Invoicing Plan	35	19
	1	Details	Loss Details, Claim Details	234	83
	2	Claim	Flood Claim, BAU Claim	160	13
CH	3	Assessment	Onsite Assmt., Specialist Assmt.	137	11
	4	Status	Job Status, Investigation Status	121	21
	5	{Task, Action, Activity}	Recovery Activity, Work Activity	93	21
	1	{Order, Suborder}	Purchase Order, HW Order	907	314
	2	Request	Customer Request, Porting Request	806	292
TC	3	{Information, Info}	Customer Info., Collection Info.	370	180
	4	Details	Customer Details, Contact Details	233	117
	5	Case	Customer Case, Business Case	218	71

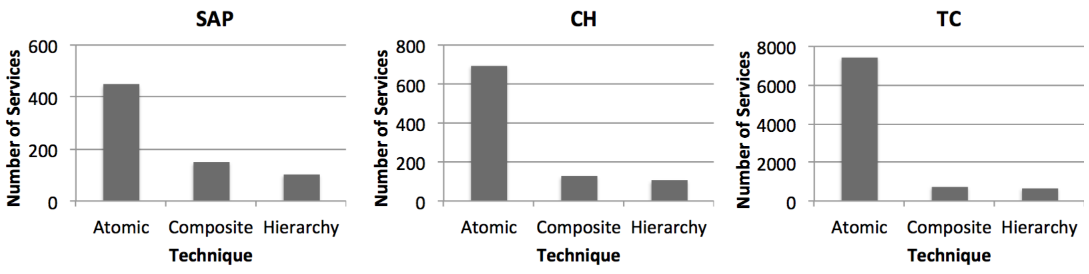


Figure 9: Comparison of Total Number of Services

show the development of the total number of services and the average service weight when moving from the atomic service identification towards the inheritance hierarchy identification. The figures clearly confirm our quantitative hypotheses: While the total number of services is decreasing, the average weight of the services increases. Interestingly, the strength of this effect is comparable among all three collections. This emphasizes the suitability of the presented techniques for identifying services with a differing degree of cohesion. While the atomic services represent candidates with a low cohesion, the composite and inheritance hierarchy service represent candidates with a higher cohesion, i.e., they represent clusters of activities that all contribute to the lifecycle of a specific business object.

To illustrate these quantitative effects, consider the business object *incident*. In the context of the atomic service identification, the activities *Handle Incident* and *Manage Incident* are clustered into a single service candidate with a weight of 83. The composite service identification increases the weight of the service related to the business object *incident* to 159. The reason is the inclusion of additional actions such as *analyze* and *resolve*. The inheritance hierarchy service identification further increases this weight to a total of 202. This increase is caused by the additional inclusion of compounds such as *customer incident* and *information*

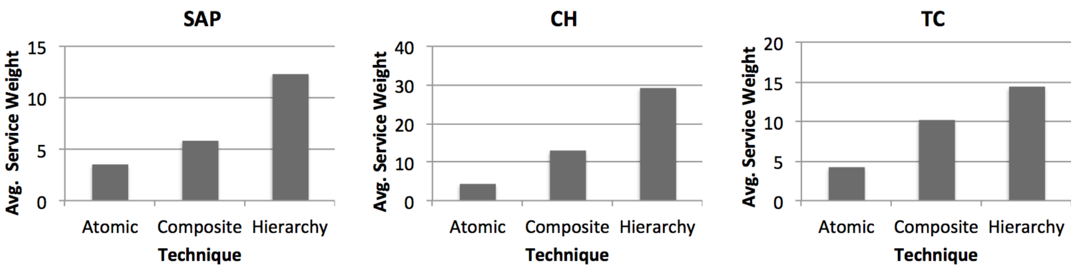


Figure 10: Comparison of Average Service Weight

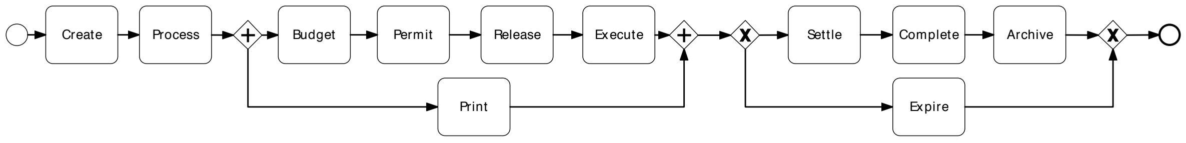


Figure 11: The Process Model for the Composite Service *order*

incident.

This example does not only illustrate the quantitatively described aggregation effect, but also one of the core challenges that is associated with it: the proper association of compound nouns. A detailed analysis of the results reveals that BabelNet has almost a perfect coverage of non-compound words. Non-covered words are either spelled incorrectly (e.g. *harship* instead of *hardship*) or represent non-standard abbreviations (e.g. *EAA* or *AWP*). However, specific compound nouns from the business domain as, for instance, *information incident* are often not covered. For such words, the hierarchy service identification plays an important role. By decomposing compound nouns into their constituents, the business object *information incident* can be successfully related to *incident* as well as to its synonyms such as *event.*

4.3.5. Detailing of a Service Candidate

To demonstrate the derivation of the internal service structure, we use the top-ranked business object *order* from the SAP collection.

In order to determine the internal structure of such a composite service, we compute the behavioral profile for the comprised activities. Table 7 shows the corresponding behavioral profile for the composite service *order*. Recalling that the behavioral relation of a strict order results in a sequence and the interleaving relation in a parallel path, 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 between the activity *process* and the activity *execute*, the order is strict. The synthesis of this profile yields the process model depicted in Figure 11.

The generated process model is the result of synthesizing the information from 20 different process models

Table 7: The Behavioral Profile for the Composite Service *order*

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

from the SAP collection. Although, in total, 23 process models contain the business object *order*, only 20 of them include two or more activities with this business object. As it is not possible to compute a behavioral relation from a single activity, we excluded these models from the life cycle computation. The associated processing time of the detailing amounts to 978 milliseconds. Considering that *order* is the most frequent business object, this illustrates the practical applicability of the life cycle computation.

From a general perspective, this example illustrates that the information from a process model collection can also help to support the internal design of a service. By configuring the minimal support *minsup* and the minimal confidence *minconf*, the automatically generated internal structure can be adapted to the needs of the user, i.e., less frequent activities can be excluded or included. Although this structure may not represent the final implementation of the service, it represents valuable information about the behavior and interaction of service activities, which is presented to the user in a fully automated fashion.

5. Implications

In general, the results from the evaluation demonstrated that we can successfully derive useful service candidates. As opposed to existing techniques, we emphasize three important differences: degree of automation, usage of linguistic information and semantic technology, and different service perspectives.

The presented approach *fully automates* the four phases of service identification from preparation to prioritization of the candidates. The most important difference to previous service identification techniques can be found in the preparation and the prioritization phase. By building on a linguistic parsing and annotation technique, we are able to automatically identify heterogeneous labeling structures and thus to relieve users from the manual preparation. In the prioritization phase, we support users by ranking the derived service candidates. Although the final selection is still left to the user, the automation significantly reduces the manual effort. The application of the technique on three large process model collections, one including more than 30,000 activities, demonstrates the need for automation. Up to the final selection, the proposed technique requires no manual work. Hence, it can be scaled up to extremely large process model collections.

As opposed to existing techniques, we *semantically analyze the natural language* of process model activities. Thus, we are able to use action and business object to obtain different service candidates and to compute the internal structure of a service. Moreover, the detailed consideration of the natural language allows us to group semantically similar behavior. By exploiting the capabilities of BabelNet, we no longer depend on the syntactic similarity of terms, but we can successfully cluster semantically intersecting terms such as *handle* and *manage* or *action* and *activity.* In general, the semantic analysis represents an important delta as it facilitates the automated yet reliable and useful clustering of model behavior. While some approaches suggested similar strategies, none of them provided support for automatically implementing them.

As a consequence of the possibility to individually consider actions and business objects, the presented approach includes *different service perspectives.* The benefits of having different perspectives are, for instance, reflected by the commonalities among the different service lists. We observed that some business objects such as *order* are included in the top five list of all identification techniques, while other candidates can be found only in the list of a single technique. On the one hand, this highlights the importance of these candidates, on the other hand, this emphasizes the benefits of taking different perspectives, because additional candidates are also provided by each technique. Another benefit of the different perspectives is the coverage of different levels of granularity. While atomic services represent fine granular services, inheritance hierarchy services represent rather course-grained services. Depending on the needs of the user, this represents an important feature.

Altogether, we consider the presented technique to be an important step towards the fully automated derivation of service candidates. Hence, from a practical perspective, the technique can be effectively employed in companies with large process model repositories. However, application scenarios do not only include the automated identification of service candidates. Our technique may also help to improve the alignment between business and IT. As the ranked service candidates give a good impression on the relative importance of a business operation, they can also provide companies with first clues on where IT support is needed and where it could be reduced.

6. Limitations

Despite the interesting findings presented in this paper, we have to discuss the results from the perspective of some limitations.

First, we have to state that the employed process model collections are not representative in a statistical sense. Process models in practice are used for various purposes and may describe operations at different levels of detail. In this paper, we consider a situation in which process models describing processes at a operational level are available. While this is typically the case for large organizations, it is an important requirement of our approach. More generic process descriptions, such as value chains, will only be of limited use for our approach. Besides this requirement, it is, however, important to note that we do not make any assumptions on the grammar or the content of the collections. Hence, we are are confident that our technique can effectively identify services from any English process model collection that describes the processes at a sufficient level of detail.

Second, we build on the lexical database BabelNet. As a result, we can only identify semantic relations between concepts that are part of this dictionary. Particularly for hierarchical relationships between words, so-called hypernomy relations, and for more specific vocabulary, other knowledge bases and ontologies may represent useful sources for further improvement of our technique. Nevertheless, frequently occurring terms will, even if they are not part of BabelNet, still appear as service candidates. What is more, the hierarchy service identification technique is able to associate compounds. This way, we mitigated the limitations of using BabelNet.

Last, we would like to highlight the assumption of semantic quality of the analyzed process models. If the process models do not adequately represent the company's operations, the service identification is not likely to create useful results. However, we believe that if an company invests time and money for maintaining a large process model repository, it is a viable assumption that the semantic quality of these models is appropriate.

7. Conclusion

In this paper, we addressed the problem of manual work in the context of identifying services from business process models. From a literature review we learned that current approaches often lack methodological detail, only partially cover the different phases of service derivation, often require an extensive amount of manual work, and do not systematically address process model quality issues. The approach presented in this paper addresses these problems by providing the possibility to automatically derive a list of ranked service candidates from business process models. In particular, it generates a ranked list of atomic services, composite services, and inheritance hierarchy services. An evaluation with three large process model collections from industry demonstrated that the approach can successfully generate useful service candidates. Further, it illustrates the benefits of the delta to previously defined techniques. The sophisticated analysis of natural language facilitates the semantic comparison of action and business object to generate different service candidate lists. In addition, it enables the computation of an internal service structure. The automation provides users with the possibility to apply the technique on huge process model collections. As a result, services can also be derived from extensively large collections with hardly any manual effort.

In future research, we plan 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] J. O'Sullivan, D. Edmond, A. H. M. ter Hofstede, What's in a Service?, Distributed and Parallel Databases 12 (2/3) (2002) 117–133.

[2] T. Kohlborn, A. Korthaus, T. Chan, M. Rosemann, Identification and Analysis of Business and Software Services - A Consolidated Approach, IEEE Transactions on Services Computing 2 (1) (2009) 50–64.

[3] D. Adamopoulos, G. Pavlou, C. Papandreou, Advanced service creation using distributed object technology, Communications Magazine, IEEE 40 (3) (2002) 146–154.

[4] S. Chang, S. Kim, A Systematic Approach to Service-Oriented Analysis and Design, in: Product-Focused Software Process Improvement, vol. 4589 of *LNCS*, Springer Berlin Heidelberg, 374–388, 2007.

[5] L. G. Azevedo, F. Santoro, F. Baião, J. Souza, K. Revoredo, V. Pereira, I. Herlain, A Method for Service Identification from Business Process Models in a SOA Approach, in: Enterprise, Business-Process and Information Systems Modeling, vol. 29 of *LNBIP*, Springer Berlin Heidelberg, 99–112, 2009.

[6] Z. Yun, S. Huayou, N. Yulin, Q. Hengnian, A service-oriented analysis and design approach based on data flow diagram, in: International Conference on Computational Intelligence and Software Engineering, IEEE, 1–5, 2009.

[7] J. Lee, D. Muthig, M. Naab, An Approach for Developing Service Oriented Product Lines, in: Proceedings of the 12th International Software Product Line Conference, SPLC '08, IEEE Computer Society, Washington, DC, USA, 275–284, 2008.

[8] L. Aversano, L. Cerulo, C. Palumbo, Mining candidate web services from legacy code, in: 10th International Symposium on Web Site Evolution, 37–40, 2008.

[9] F. Chen, Z. Zhang, J. Li, J. Kang, H. Yang, Service Identification via Ontology Mapping, 2012 IEEE 36th Annual Computer Software and Applications Conference 1 (2009) 486–491.

[10] Z. Zhang, R. Liu, H. Yang, Service identification and packaging in service oriented reengineering, in: Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering, 241–249, 2005.

[11] M. Rosemann, Potential Pitfalls of Process Modeling: Part A, Business Process Management Journal 12 (2) (2006) 249–254.

[12] C. Houy, P. Fettke, P. Loos, W. M. van der Aalst, J. Krogstie, Business process management in the large, Business & Information Systems Engineering 3 (6) (2011) 385–388.

[13] H. Leopold, J. Mendling, Automatic Derivation of Service Candidates from Business Process Model Repositories, in: Business Information Systems, 84–95, 2012.

[14] P. Sanders, Algorithm engineering–an attempt at a definition, in: Efficient Algorithms, Springer, 321–340, 2009.

[15] F. Pittke, H. Leopold, J. Mendling, Spotting Terminology Deficiencies in Process Model Repositories, in: Enterprise, Business-Process and Information Systems Modeling, 2013.

[16] G. Feuerlicht, Design of service interfaces for e-business applications using data normalization techniques, Information Systems and E-Business Management 3 (4) (2005) 363–376.

[17] M. Bell, Service-oriented modeling. Service Analysis, Design and Architecture., John Wiley and Sons, Hoboken, NJ, 2008.

[18] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[19] E. Ramollari, D. Dranidis, A. J. H. Simons, A Survey of Service Oriented Development Methodologies, in: Proceedings of the 2nd European Young Researchers Workshop on Service Oriented Computing, 2007.

[20] Q. Gu, P. Lago, Service Identification Methods: A Systematic Literature Review, in: E. Nitto, R. Yahyapour (Eds.), Towards a Service-Based Internet, vol. 6481 of *LNCS*, Springer Berlin Heidelberg, 37–50, 2010.

[21] A. Erradi, N. Kulkarni, P. Maheshwari, Service Design Process for Reusable Services: Financial Services Case Study, in: Proceedings of the 5th international conference on Service-Oriented Computing, ICSOC '07, Springer-Verlag, Berlin, Heidelberg, 606–617, 2007.

[22] P. Jamshidi, M. Sharifi, S. Mansour, To Establish Enterprise Service Model from Enterprise Business Model, in: Proceedings of the IEEE International Conference on Services Computing, SCC '08, IEEE Computer Society, Washington, DC, USA, 93–100, 2008.

[23] K. Klose, R. Knackstedt, D. Beverungen, Identification of Services - A Stakeholder-Based Approach to SOA Development and its Application in the Area of Production Planning, University of St. Gallen, 2007.

[24] J.-H. Sewing, M. Rosemann, M. Dumas, Process-Oriented Assessment of Web Services, International Journal of E-Business Research 2 (1) (2006) 19–44.

[25] O. Zimmermann, P. Krogdahl, C. Gee, Elements of service-oriented analysis and design, IBM developerworks .

[26] F. Kohlmann, R. Alt, Business-Driven Service Modelling - A Methodological Approach from the Finance Industry, in: Proceedings of the 1st International Working Conference on Business Process and Services Computing, 180–193, 2007.

[27] D. Bianchini, C. Cappiello, V. Antonellis, B. Pernici, P2S: A Methodology to Enable Inter-organizational Process Design through Web Services, in: Proceedings of the 21st International Conference on Advanced Information Systems Engineering, CAiSE '09, Springer-Verlag, Berlin, Heidelberg, 334–348, 2009.

[28] V. Dwivedi, N. Kulkarni, A Model Driven Service Identification Approach for Process Centric Systems, in: Proceedings of the IEEE Congress on Services Part II, SERVICES-2 '08, IEEE Computer Society, Washington, DC, USA, 65–72, 2008.

[29] T. Kleinert, S. Balzert, P. Fettke, P. Loos, Systematic Identification of Service-Blueprints for Service-Processes - A Method and Exemplary Application, in: M. Rosa, P. Soffer (Eds.), Business Process Management Workshops, vol. 132 of *LNBIP*, Springer Berlin Heidelberg, 598–610, 2013.

[30] R. Yousef, M. Odeh, D. Coward, A. Sharieh, BPAOntoSOA: A generic framework to derive software service oriented models from business process architectures, in: Proceedings of the 2nd International Conference on the Applications of Digital Information and Web Technologies, 50–55, 2009.

[31] S. Kim, M. Kim, S. Park, Service Identification Using Goal and Scenario in Service Oriented Architecture, in: Asia-Pacific Software Engineering Conference, vol. 0, IEEE Computer Society, Los Alamitos, CA, USA, 419–426, 2008.

[32] H. K. Jain, H. Zhao, N. R. Chinta, A Spanning Tree Based Approach to Identifying Web Services., in: L.-J. Zhang (Ed.), ICWS, CSREA Press, 272–277, 2003.

[33] Y. Kim, K.-G. Doh, The Service Modeling Process Based on Use Case Refactoring, in: W. Abramowicz (Ed.), Business Information Systems, vol. 4439 of *LNCS*, Springer Berlin Heidelberg, 108–120, 2007.

[34] H. Sneed, Integrating legacy software into a service oriented architecture, in: Proceedings of the 10th European Conference on Software Maintenance and Reengineering, 11 pp.–14, 2006.

[35] Z. Zhang, H. Yang, Incubating services in legacy systems for architectural migration, in: Proeedings of the 11th Software Engineering Conference, 196–203, 2004.

[36] Y. Baghdadi, Reverse engineering relational databases to identify and specify basic Web services with respect to service oriented computing, Information Systems Frontiers 8 (5) (2006) 395–410, ISSN 1387-3326.

[37] S. Mani, V. Sinha, N. Sukaviriya, T. Ramachandra, Using User Interface Design to Enhance Service Identification, in:

IEEE International Conference on Web Services, 78–87, 2008.

[38] A. Arsanjani, A. Allam, Service-Oriented Modeling and Architecture for Realization of an SOA, in: Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society, Washington, DC, USA, 2006.

[39] D. Flaxer, A. Nigam, Realizing Business Components, Business Operations and Business Services, in: Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference, CEC-EAST '04, IEEE Computer Society, Washington, DC, USA, 328–332, 2004.

[40] J. Krogstie, O. I. Lindland, G. Sindre, Defining quality aspects for conceptual models, in: Proceedings of the international working conference on Information system concepts: Towards a consolidation of views, Chapman & Hall, Ltd., 216–231, 1995.

[41] J. Krogstie, G. Sindre, H. Jorgensen, Process models representing knowledge for action: a revised quality framework, European Journal of Information Systems 15 (1) (2006) 91–102.

[42] D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, K. Wolf, Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models, Data & Knowledge Engineering 70 (5) (2011) 448–466.

[43] W. van der Aalst, H. de Beer, B. van Dongen, Process Mining and Verification of Properties: An Approach Based on Temporal Logic, in: OTM Conferences (1), vol. 3760 of *LNCS*, Springer, ISBN 978-3-540-29736-9, 130–147, 2005.

[44] W. M. P. van der Aalst, A. K. A. de Medeiros, Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance, Electron. Notes Theor. Comput. Sci. 121 (2005) 3–21, ISSN 1571-0661, doi: 10.1016/j.entcs.2004.10.013.

[45] H. Leopold, S. Smirnov, J. Mendling, On the refactoring of activity labels in business process models, Information Systems 37 (5) (2012) 443–459.

[46] J. Mendling, H. A. Reijers, J. Recker, Activity Labeling in Process Modeling: Empirical Insights and Recommendations, Information Systems 35 (4) (2010) 467–482.

[47] R. Dijkman, M. La Rosa, H. A. Reijers, Managing large collections of business process models-current techniques and challenges, Computers in Industry 63 (2) (2012) 91–97.

[48] H. Leopold, Natural Language in Business Process Models: Theoretical Foundations, Techniques, and Applications, vol. 168 of *LNBIP*, Springer, 2013.

[49] R. Navigli, Word sense disambiguation: A survey, ACM Computing Surveys (CSUR) 41 (2) (2009) 10.

[50] A. Stevenson, Oxford Dictionary of English, Oxford University Press, 2010.

[51] P. Procter, Longman dictionary of contemporary English .

[52] R. Navigli, S. P. Ponzetto, BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network, Artificial Intelligence 193 (2012) 217–250.

[53] G. Miller, C. Fellbaum, WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.

[54] S. P. Ponzetto, R. Navigli, Knowledge-rich word sense disambiguation rivaling supervised systems, in: Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, 1522–1531, 2010.

[55] F. Deissenboeck, M. Pizka, Concise and consistent naming, Software Quality Journal 14 (3) (2006) 261–282.

[56] S. Inaganti, G. K. Behara, Service Identification: BPM and SOA Handshake, BPTrends .

[57] M. P. Papazoglou, W. V. D. Heuvel, Service oriented design and development methodology, International Journal of Web Engineering and Technology 2 (2006) 412–442.

[58] S. Smirnov, M. Weidlich, J. Mendling, M. Weske, Action Patterns in Business Process Model Repositories, Computers in Industry 63.

[59] M. Weidlich, J. Mendling, M. Weske, Efficient Consistency Measurement Based on Behavioral Profiles of Process Models, IEEE Trans. Software Eng. 37 (3) (2011) 410–429.

[60] S. Smirnov, M. Weidlich, J. Mendling, Business process model abstraction based on Synthesis from consistent behavioural profiles, International Journal of Cooperative Information Systems 21.

[61] G. Keller, T. Teufel, SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping, Addison-Wesley, 1998.