User Assistance For Business Process Model Decomposition

Agnes Koschmider
Institute of Applied Informatics and
Formal Description Methods
Universität Karlsruhe (TH), Germany
Email: koschmider@aifb.uni-karlsruhe.de

Emmanuel Blanchard
Labortoire d'Informatique de Nantes Atlantique
Site École polytechnique de l'université de Nantes, France
Email: emmanuel.blanchard@univ-nantes.fr

Abstract— **Petri nets are a suitable language for modeling business processes with complex flow structures. Processes with a number of alternative, concurrent or sequential flow structures on the same process abstraction level may result in complex processes. Such complex and poorly unstructured processes are particularly difficult for users to understand. Process decomposition can significantly improve the comprehensibility and consistency of process models and enables faster reuse of process models. To maintain homogenous process decomposition demands a significant amount of modeling experience. In this paper, we will present our approach for (semi-)automatic detection of non-uniformly specified process element names on the same process decomposition level. Our detection system validates process element names regarding reasonable hierarchical specifications. The aim of our approach is to (semi-)automatically highlight non-uniformly specified process elements in order to improve process model consistency. The approach is based upon an OWL-based description of Petri nets.**

I. Introduction

Business process models serve as a modeling communication foundation for all persons involved in the process modeling, execution, and implementation. Process models make it possible to analyze alternative process designs in order to ameliorate implemented functionalities of information systems. Non-uniformly modeled process models (e.g., process element names have heterogeneous abstraction on the same abstraction level) return falsified analysis results, may lead to costly erroneous judgements or disregarding of relevant information. Furthermore, different modeling purposes and processes with alternative, concurrent or sequential flow structures on the same process level make processes complex. Such complex and poorly structured process models are particularly hard for users to understand.

Consistent process decomposition can significantly improve the comprehensibility and model consistency of processes [1]. It may help to avoid misjudgements and disregarding relevant information. Decomposable processes (process/subprocess models) allow systems to be compact. They enable the modularization of large business process models and faster reuse of process models. They make processes more easy to read and to understand.

However, the level of process decomposition depends on the modeler and on the purpose and scope of the process modeling. Each decomposition level describes process elements from a different abstraction level. Top level process models formulate an overview of process activities and bottom models provide more detailed descriptions. However, users have to maintain particular modeling requirements such as homogenous abstraction of process element names on the same decomposition level in order to improve model consistency. But, model consistency demands to identify an appropriate point to stop the decomposition and to avoid overly granularity for process element names (e.g., datatypes for process element names). Consequently, to maintain this particular modeling requirement demands a significant amount of experience in the field of process engineering and may result in extra analyzing efforts.

Within this context we will present an approach that highlights non-uniformly specified process element names on the same process model decomposition level. Our analysis system validates process element names regarding their hierarchical specification. In case of heterogeneous specifications the system (semi-)automatically detects the non-uniformly specified process elements. Our approach has been applied for business processes modeled with Petri nets [2]. In order to support a (semi-)automatic detection requires an unambiguous description of Petri nets, which enables automatic manipulations. We describe traditional Petri nets using the Web Ontology Language OWL [3]. These so-called semantic business process models [4] are represented in same syntax as numerous (background) ontologies. Additional mapping efforts between background ontologies and our ontology-based Petri net description are not required. Furthermore, semantic business process models promise to solve ambiguity issues caused by the use of different terms for element names [5].

Figure 1a shows "inconsistently" modeled business processes with one process decomposition where the element names in **Process Level 2** have a non-uniform abstraction for element names. The preparation of products is modeled in detail (elements from *order product* to *products prepared*) in contrast to the preparation of documents, which is described in a more abstract way (element *prepare documents*). In order to maintain the same level of abstraction for process element names it is recommended first to model an abstract view of how to send an order (**Process Level 0** in Figure

1b). The sending procedure is modeled in detail by refining the transition *send order* to a subprocess (**Process Level 1**). Product preparation is described in detail with a subprocess modeled in **Process Level 2**.

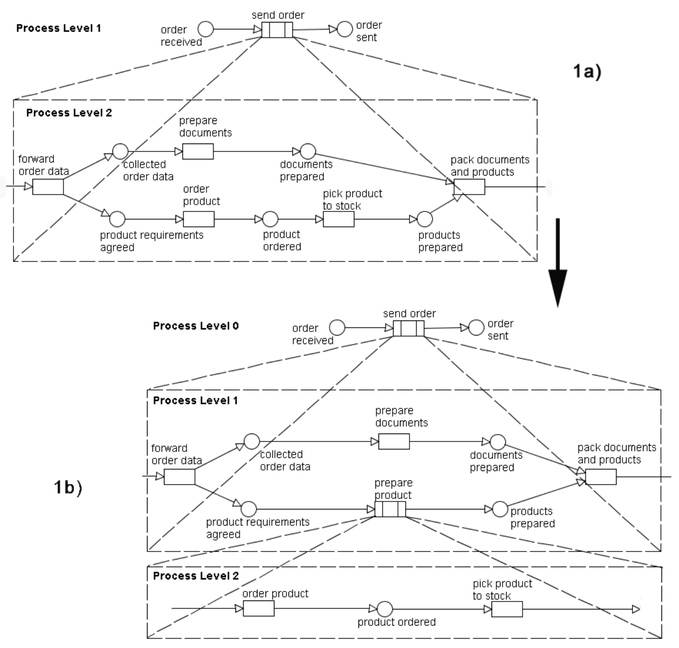


Fig. 1. "Inconsistent"(1a) and "consistent"(1b) process modeling

This paper is structured as follows. In Section II we will recall the main notions of Petri nets and semantic business process models. Section III introduces process modeling methodologies which enable process decompositions. Semantic business process model analysis requires to measure linguistic specificities for element names, which we will explain in Section IV. Section V presents three refinement patterns and algorithms that promise to discover non-uniform process decomposition. Another application area of our approach is to support (unfamiliar) users in process modeling by providing an autocompletion mechanism that we will sketch in Section VI. Some practical experiences and implementation work will be illustrated in Section VII. Section VIII surveys related work and Section IX concludes the paper with a summary of our approach.

II. Foundation

The following section gives an overview of Petri nets and an OWL-based description of Petri nets.

A. Petri net

Petri nets have been established as a language for modeling business processes [6] and in context of workflow management to verify the correctness of workflow procedures [7]. The mathematical foundation of Petri nets provides simulation and analysis techniques, which can be used to verify the correctness of system behaviors. Formally, a Petri net is a directed bipartite graph with two disjoint sets of nodes (places and transitions) and a set of arcs (flow relations). Places are passive components and are drawn as circles. Transitions are active components that change the state of the system and are drawn as rectangles. Places and transitions are connected by directed arcs. Petri net examples are given in Figure 1. We will always consider a net with P being the set of places, T being the set of transitions, F_r being the set of arcs connecting P with T and F_w as arcs connecting T and P.

Numerous Petri net variants have been proposed, which can be subsumed in elementary or high-level Petri nets. In elementary Petri nets (place/transition nets), the flow of tokens representing anonymous objects (tokens drawn as black dots) defines the process flow. To make tokens distinguishable, variants of high-level Petri nets have been proposed such as Predicate/Transitions nets [8]. In this paper, we focus on Predicate/Transitions nets (Pr/T nets) where places are interpreted as predicates representing relation schemes. Transitions occur according to a logical expression which may be attached to them.

B. Semantic Business Process Models

By describing Petri net elements in OWL we combine process modeling methods with semantic technologies to achieve automatic processing of process models instead of manual processing. Each Petri net element has a corresponding element in the semantic business process model (SBPM). The set of places corresponds to the concept **Place**, the set of transitions to the concept **Transition**, the set of arcs connecting places with transitions to **FromPlace** and arcs connecting transitions with places correspond to **ToPlace**. Places following transitions are described via the property **transRef** and transitions following places are expressed through the **placeRef** property. The concepts and properties of SBPMs are shown in an UML-based description in Figure 2[1]. Instances or identifiers of the concept **Place** may be *documents prepared*, *product ordered* or *products prepared*. Instances of **Transitions** can be *prepare documents* or *order product*.
A SBPM corresponds to the instantiation of the Petri net ontology (as described in Figure 2), which can be represented in OWL syntax. The extraction of identifiers of business process element names and the mapping to the Petri net ontology is being carried out automatically and is not directly visible to the modeler. The result of mapping efforts are OWL files, which can afterwards be (semi-)automatically manipulated.

The next section surveys top-down and bottom-up modeling methodologies. Formal net transformations for SBPMs are introduced as well.

III. Methodologies for Process Modeling

When modeling different abstraction levels of business process models, processes can be modeled top-down or bottom-up. In top-down modeling, the top level process formulates an overview of process elements, without providing more detailed descriptions of process elements. Consequently, the top view

[1]Concepts such as IndividualDataItem, Attribute, Value, Delete, Insert and LogicalConcept correspond to specific high-level petri net notation

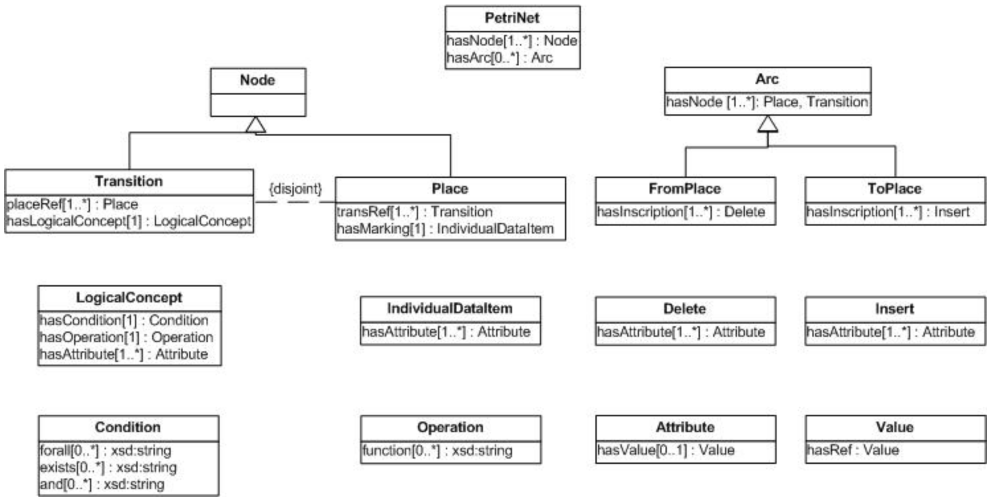

PetriNet
hasNode[1..*] : Node
hasArc[0..*] : Arc
Node
Arc
hasNode [1..*]: Place, Transition
Transition
placeRef[1..*] : Place
hasLogicalConcept[1] : LogicalConcept
{disjoint}
Place
transRef[1..*] : Transition
hasMarking[1] : IndividualDataItem
FromPlace
hasInscription[1..*] : Delete
ToPlace
hasInscription[1..*] : Insert
LogicalConcept
hasCondition[1] : Condition
hasOperation[1] : Operation
hasAttribute[1..*] : Attribute
IndividualDataItem
hasAttribute[1..*] : Attribute
Delete
hasAttribute[1..*] : Attribute
Insert
hasAttribute[1..*] : Attribute
Condition
forall[0..*] : xsd:string
exists[0..*] : xsd:string
and[0..*] : xsd:string
Operation
function[0..*] : xsd:string
Attribute
hasValue[0..1] : Value
Value
hasRef : Value

Fig. 2. Petri net Ontology

of the process is structured in a more fine-grained way by refining transitions to subprocesses.

By using the bottom-up approach, in contrast to the top-down approach, modelers start modeling more specific process elements, which are subsequently linked together to coarse-grained processes. The linking or coarsening is done until a complete abstract view of the process model is determined. The main advantage of the bottom-up approach are a-priori simulation and analysis capabilities of process fragments before they have been integrated to a complete process model. The middle-out approach can be seen as an amalgamation of the bottom-up– and the top-down approach.

The applicability of either both approaches depends on the quantity of process information and modeling purposes. A higher amount of information favors the bottom-up approach. The top-down approach should be used if the modeler has initially only approximate information of processes to be modeled.

For (high-level) Petri nets several formal net transformations such as refinement and coarsening were proposed to realize a top-down and a bottom-up modeling [9]. Using refinement in SBPMs a transition[2] will be refined to a subprocess. The preset (set of all input places) and the postset (set of all output places) of the transition to be refined will be inherited to the subprocess. In our Petri net ontology we have denoted the preset with the property **antPlaceRef** and the postset with the property **placeRef** of the concept **Refinement**. Coarsening is reversal to refinement where a subprocess is replaced by a transition.

IV. Measuring Linguistic Specificity

This Section illustrates how to evaluate a linguistic specificity between instance names of SBPMs using a background ontology. First a classification scheme of instance names is described in order to analyze abstraction homogeneity and heterogeneity on different decomposition levels.

A. Classification scheme

Instance names – depending on their process decomposition level – have different specificities. Linguistic structures of instance names can be classified based on WordNet [10], which is an English online lexical reference system. The base of WordNet consists of nouns, verbs, adjectives or adverbs linked by relationships such as hyperonymy/hyponymy and meronymy/holonymy as listed below:

- synonymy: two terms have an identical meaning (e.g., "merchandise" and "product" are synonym),
- hyperonymy/hyponymy: two terms have an *is-a* relationship (e.g., "product" is a "commodity").

The given structure of WordNet can be extended by homonyms (two terms have same pronunciation but different meaning) in order to classify different identifiers of semantic business process model concepts. For similarity calculation of synonyms and homonyms we refer to [5]. For our work we have to measure the similarity of identifiers with different abstraction levels (hyperonyms/hyponyms). Our measurement is based upon the hypothesis that in fine-grained processes instance names are more specific than in coarse-grained. Specificity of instance names indicates their process decomposition level and thus their abstraction level.

B. Ontology exploitation

First, we have to find a possibility to evaluate (semi-) automatically the specificity of each identifier. To exploit linguistic structure of names we decided to use the (freely available) WordNet ontology, which is widely used in automatic natural language processing especially for linguistic measures [11]. To determine specificities, we will consider research works on semantic measures, which aim at exploiting ontologies.

To describe proximity relationships between domain concepts by a hierarchy, or more generally a graph, remains a

[2]In the following, we will only consider transitions and not places that are refined to subprocesses.

common principle of current knowledge representation systems, namely ontologies, associated with new languages of the Semantic Web – in particular OWL. When considering ontologies, the concepts are not necessarily described by their extension; the internal organization of a domain ontology is often the result of a consensus of domain experts [12]. Hence, defining a semantic similarity $\sigma(c_i, c_j)$ between a pair c_i, c_j of concepts poses specific problems depending on the quality of the information at our disposal. Some measures depend only on concept structuring – often taxonomic – ; others additionally require textual corpus of the domain.

Formally, an ontology can be modeled by a graph where nodes represent concepts and arcs represent labeled relationships. In the following we restrict concept relationships by hyperonymy (generalization) and hyponymy (specialization) relationships[3] associated to the subsumption relationship (is-a). Regarding relationships between concepts in this way is common to every ontology, and several research works have confirmed that this is a reasonable way (e.g., [13]). In fact, we consider a subsumption hierarchy in which for instance "product" is a "commodity" noted $product \sqsubseteq commodity$ ("product" is the hyponym of "commodity" and "commodity" is the hyperonym of "product"). Let $\mathcal{O}$ be this hierarchy defined on a set $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ of concepts. Moreover, to maintain the connectivity, it is common to add a virtual concept c_0 (e.g., "thing"). To simplify this presentation, we assume that each concept c_i has no more than one hyperonym. Consequently, $\mathcal{O}$ can be represented by a rooted tree $T_{\mathcal{O}}(\mathcal{C}, \mathcal{A})$ with the root c_0 and thus each arc $(c_i, c_j) \in \mathcal{A}$ corresponds to $c_j \sqsubseteq c_i$ (see Figure 3).

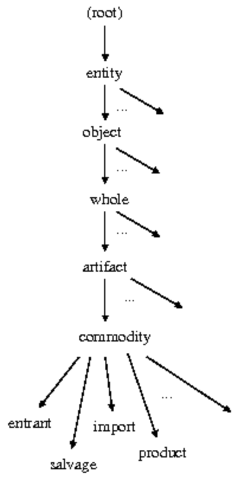


Fig. 3. Hyponym/Hyperonym relationships extracted from WordNet

C. Specificity definition

The definition of subsumption link induces that the deeper a concept is in $T_{\mathcal{O}}(\mathcal{C}, \mathcal{A})$, the more specific it is. Intuitively, the specificity of a concept is an increasing function of the depth in the hierarchy. A dissimilarity, proposed by Rada et al. [13], is $\delta_r(c_i, c_j) = len(c_i, c_j)$ the length (in number of arcs) of the path between the two concepts c_i and c_j. Despite its simplicity, experiments in information retrieval have shown that, when the paths are restricted to is-a links as in our consideration, this measure returns satisfying results.

[3]such often in the literature [13] [14] [15].

The Wu and Palmer's similarity [14] $\sigma_{wp}(c_i, c_j) = \frac{2*len(c_0, mscs(c_i,c_j))}{len(c_0,c_i)+len(c_0,c_j)}$ considers the depth of the concepts c_i and c_j to the root concept ($len(c_0, c_i)$ and $len(c_0, c_j)$). This measure also takes into account the depth of the most specific common subsumer of c_i and c_j ($mscs(c_i, c_j)$). For instance, on Figure 3, "product" and "import" have six common subsumer ("commodity", "artifact", "whole", "object", "entity" and the virtual root) and the most specific one is "commodity".

Finally, we can conclude that the depth of a concept is a possible estimator of its specificity and the similarity between two concepts could be computed with the Wu and Palmer's similarity:

$$specificity(c_i) = len(c_0, c_i) \tag{1}$$

$$similarity(c_i, c_j) = \frac{2 * specificity(mscs(c_i, c_j))}{specificity(c_i) + specificity(c_j)} \tag{2}$$

It is possible to normalize the specificity considering the most specific concept of the hierarchy:

$$specificity(c_i) = \frac{len(c_0, c_i)}{max_{c_x \in \mathcal{C}} len(c_0, c_x)} \tag{3}$$

Now, we investigate the existence of further information, which could be extracted from the structure of the hierarchy. Another approach has been proposed to exploit an ontology that uses an additional textual corpus. This approach introduces a measure of the "information content" of a concept which is the amount of information associated to a concept. The required additional notion is the probability $P(c_i) \in [0, 1]$ of encountering an occurrence of c_i. In practice, this probability is estimated from a textual corpus S by the occurrence frequency of c_i in S $num(c_i)/num(c_0)$. To compute $num(c_i)$, Resnik [16] proposes to count not only the number of occurrences of c_i, but also the number of occurrences of the concepts which are subsumed by c_i. The main result of this approach is the calculation of the "information content" of a concept by $-\log P(c_i)$.

Our recent study [17] on these semantic measures proposes a new measure called the proportion of shared specificity. We suggest a definition of the specificity of a concept: $-\log P(c_i)$ where $P(c_i)$ is the probability of a generic instance to be a member of the instance class defined by the concept c_i. As we do not have the instances, we can estimate the probabilities thanks to the following hypothesis: the instance number in the sons (specialized concepts) of each concept is distributed uniformly. The interests of this new information content calculation is that we do not need any corpus and the semantic of the obtained measure is easier to understand.

To calculate the probability of an instance to be a member of the class defined by a concept, we divide the probability of an instance to be a member of the class defined by the root ($P(c_0) = 1$) by the number of sons of the root. We obtain an estimation of the probability for each child element of the root. We make the same for each son and so on. Finally, we obtain the estimated probabilities for all the concepts of the hierarchy $\mathcal{O}$. For instance, to calculate the specificity of the term "product" of Figure 3 first, the term "root " has been added to obtain a single root. Any generic instance is a member of the class defined by the root concept. Consequently, the probability of the root is $P(root) = 1$. The root concept is specialized in 9 concepts and "entity" is one of them. Following the uniform distribution, the probability of "entity" is $P(entity) = \frac{1}{9}$. With the same reasoning, $P(product) = \frac{1}{9 \cdot 10 \cdot 36 \cdot 3 \cdot 42 \cdot 12}$. The negative logarithm of the probability returns as concept specificity for "product": $-\log(P(product)) = 15.4$. Finally, the similarity definition is the same as the previous definition and we redefine the specificity as follows:

$$specificity(c_i) = -\log P(c_i) \tag{4}$$

where,

$$P(c_i) = \begin{cases} 1 & \text{if } c_i = c_0 \\ \frac{P(c_j)}{|\{c_x, c_x \sqsubseteq c_j\}|} & \text{if } c_i \sqsubseteq c_j \end{cases} \tag{5}$$

We can do the same previous normalization considering the most specific concept of the hierarchy:

$$specificity(c_i) = \frac{-\log P(c_i)}{max_{c_x \in \mathcal{C}}\{-logP(c_x)\}} \tag{6}$$

This specificity helps to detect heterogeneously specified identifiers Place and Transition concepts.

V. Analyzing Hierarchical Process Decompositions

To reach our objective we identified three different so-called refinement patterns "Identical Nouns", "Heterogenous Nouns", and "Specialization of Nouns". We called them refinement patterns as our detection system isolates instance name sequences of one process regarding abstraction homogeneity and heterogeneity and proposes refinement in case of non-uniformly specified instances. Interrelated instance names with heterogenic abstraction as its pre– and postsets will be suggested as refinement candidates as well.

Generally, process element names are designated using a "verb and noun" expression, which should illustrate an unambiguous functionality of the process activity. Depending of the modeling scope users name elements differently. They may name elements in subprocesses with an identical noun as of the transition to be refined. Additionally, they may label elements with different verbs in order to describe in detail the flow semantics. Figure 4 shows the first refinement pattern where the term "product" occurs in all subprocess element names in connection with additional verbs[4]. Sometimes modelers utilize only verbs or only nouns mutually exclusive to describe elements. In this case we suppose that the last utilized noun/verb is applicable for all subsequent elements until another noun/verb is used.

```
<petri:Transition rdf:ID="order_product">
  <petri:placeRef>
    <petri:Place rdf:ID="product_ordered">
     <petri:transRef>
      <petri:Transition rdf:ID="pick_product_to_stock"/>
     </petri:transRef>
    </petri:Place>
  </petri:placeRef>
</petri:Transition>
```

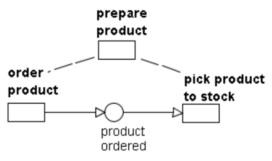


Fig. 4. Refinement Pattern 1 "Identical Nouns"

As a second pattern we have identified that subprocess elements are named with heterogenous nouns as shown in Figure 5.

```
<petri:Transition rdf:ID="check_assignment">
  <petri:placeRef>
    <petri:Place rdf:ID="correct assignment">
     <petri:transRef>
      <petri:Transition rdf:ID="find_delivery_demand">
       <petri:placeRef>
        <petri:Place rdf:ID="scheduled_delivery_demand">
         <petri:transRef>
          <petri:Transition rdf:ID="obtain_subscription">
           <petri:placeRef>
            <petri:Place rdf:ID="confirmation">
             <petri:transRef>
              <petri:Transition rdf:ID="order_documents"/>
             </petri:transRef>
            </petri:Place>
           </petri:placeRef>
          </petri:Transition>
         </petri:transRef>
        </petri:Place>
       </petri:placeRef>
      </petri:Transition>
     </petri:transRef>
    </petri:Place>
  </petri:placeRef>
</petri:Transition>
```

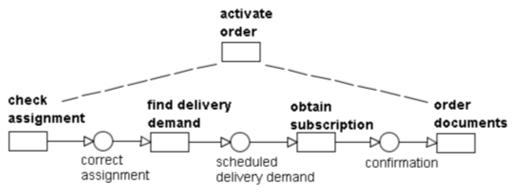


Fig. 5. Refinement Pattern 2 "Heterogenous Nouns"

These two explained patterns can be combined where subprocess elements are named with heterogenous and identical nouns.

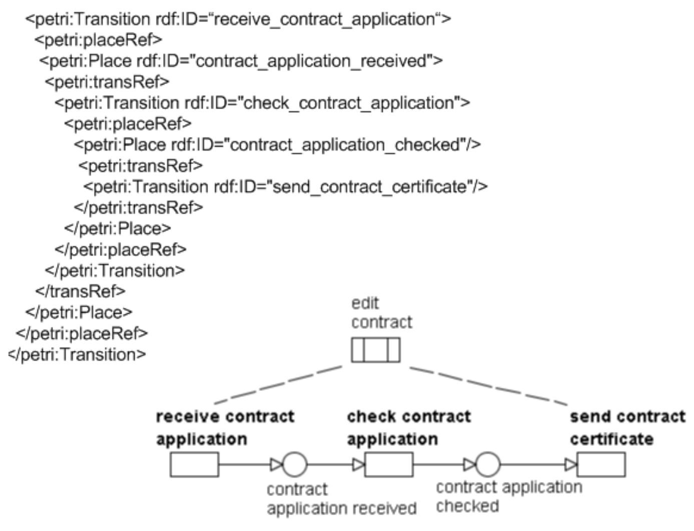
In the third refinement pattern the subprocess nouns can be regarded as a specialization of the specific transition to be refined. In Figure 6 *contract application* and *contract certificate* represent a specialization of the noun *contract*.

In a lexical taxonomy such as WordNet verbs and nouns with same radical (e.g. find, finding) use to have different depth from the root element. A suitable possibility to make calculation results of such verbs and nouns comparable is the substantiation of verbs[5], e.g. the verb *find* has been substantiated by the noun *finding*, *assign* by *assignment*, *obtain* by *obtainmen*t and so on.

The three defined patterns and the linguistic specificity for each concept instance name promise to detect non-uniformly specified process elements. But, such (semi-)automatic detection requires that the system autonomously subsumes elements to one of the patterns. The subsumption of elements can be particularly performed by first integrating all processes/subprocesses into one process as shown in Figure 7

[4]The textuell syntax of OWL is used for exchange between tools and is not directly readable for humans. Therefore, we will illustrate all patterns in Petri net notation as well.

[5]Elements named additionally with adverbs are initially not considered.



```xml
<petri:Transition rdf:ID="receive_contract_application">
 <petri:placeRef>
  <petri:Place rdf:ID="contract_application_received">
   <petri:transRef>
    <petri:Transition rdf:ID="check_contract_application">
     <petri:placeRef>
      <petri:Place rdf:ID="contract_application_checked"/>
       <petri:transRef>
        <petri:Transition rdf:ID="send_contract_certificate"/>
       </petri:transRef>
      </petri:Place>
     </petri:placeRef>
    </petri:Transition>
   </transRef>
  </petri:Place>
 </petri:placeRef>
</petri:Transition>
```

edit contract
receive contract application
check contract application
send contract certificate
contract application received
contract application checked

Fig. 6. Refinement Pattern 3 “Specialization of Nouns”

and then analyzing the linguistic specificity of each name. The integration approach is adopted from [18]. For the integration procedure we assume that processes are modeled top-down. If a transition to be refined has a cycle, then the cycle is broken into two subprocesses. For the Petri net in Figure 1 we have assumed that the pre– and postset of transition **send order** will be inherited to the subprocess. Depending on the tool implementation pre– and postsets may differ. In order to enable efficient analysis of processes modeled with Petri nets we allow only identical pre– and postset inheritances. To find a solution in case of different inheritance of pre– and postsets, [18] proposes to find relationships between element names that are expressible through aggregation, association or generalization. For instance, *contract application* and *contract certificate* can be aggregated to *contract*.

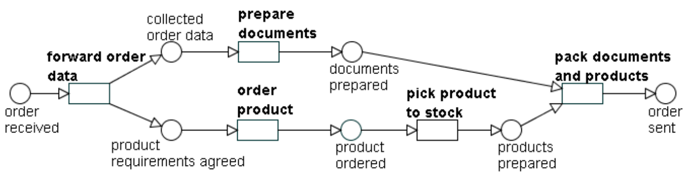
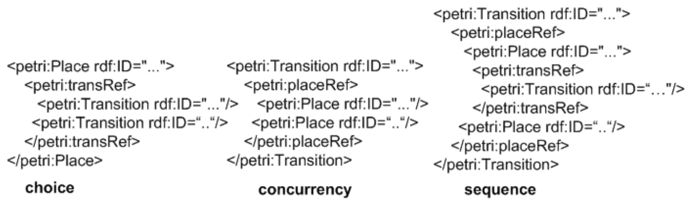


Fig. 7. One composed process for Process Levels of Figure 1

Besides linguistical aspects of names (by measuring the linguistic specificity) we have to consider process flow structures of SBPM in our analysis in order to isolate appropriate sequences. Figure 8 shows the flow structures **choice**, **concurrency** and **sequence**. The flow structure choice allows to model alternative branching (place or a transition has at least two postsets). By the use of concurrency tokens are distributed to two places. The two parallel branches are again integrated by a synchronization (place or transition has at least two presets).

After integration all processes from the top-down to one process, the algorithm traverses the process regarding choice or



```xml
<petri:Place rdf:ID="...">
  <petri:transRef>
    <petri:Transition rdf:ID="..."/>
   <petri:Transition rdf:ID=“..“/>
  </petri:transRef>
</petri:Place>
```

```xml
<petri:Transition rdf:ID="...">
  <petri:placeRef>
    <petri:Place rdf:ID="..."/>
   <petri:Place rdf:ID=“..“/>
  </petri:placeRef>
</petri:Transition>
```

```xml
<petri:Transition rdf:ID="...">
  <petri:placeRef>
    <petri:Place rdf:ID="...">
     <petri:transRef>
      <petri:Transition rdf:ID=“…"/>
     </petri:transRef>
   <petri:Place rdf:ID=“..“/>
  </petri:placeRef>
</petri:Transition>
```

choice
concurrency
sequence

Fig. 8. Process Flow Structures

concurrency branches. The algorithm regards process branching as a breakpoint and tries to find one of the three refinement patterns in a branch. In our consideration only a branch $B(\in B_1, .., B_n)$ having more than two transitions or at least four elements $e_1, .., e_n$ from E_B is a reasonable analyzing sequence.
First of all, the algorithm traverses as long as a branching occurs. This (branching) sequence B will be analyzed considering refinement patterns. Subsequently, the algorithm traverses to the next sequence and so on. If more than half of the elements of a branch are interrelated elements with higher specificity (as the remaining elements), then these elements might be subsumed to one of the three refinement patterns.

(Semi-)automatically discovering refinement pattern 1 (Identical Nouns) works more easily as for pattern 2 (Heterogenous Nouns). The algorithm looks first for an interrelation of nouns with identical linguistic specificity. A pseudo-algorithm for pattern 1 can be implemented as follows (see Algorithm 1). For each sequence B the algorithm subsumes a set of elements $e_1, .., e_n$ from E_B to Pattern 1 if the linguistic noun specificity for the isolated sequence with a set of interrelated elements is identical. If the first noun of the interrelation is the first element of the branch, then the algorithm tracks back as long nouns differ.

Algorithm 1 Refinement Pattern 1

Input: $B_1, ..., B_n$ shall be a set of sequences with individual sets of elements E_{B_i}; $i \in 1...n$.
$\sigma(e_1), .., \sigma(e_n)$ shall be the linguistic specificities for all elements in E_{B_i} where the first element of the sequence has a preset e_{1_p} in a set of elements E_B.
Output: Elements belonging to refinement pattern 1
Method: Perform the following steps:

```
for all $E \in E_{B_1}, .., E_{B_n}$ do
  if $|E| \geq 4 \wedge \sigma(e_1) = ... = \sigma(e_n), e_1, .., e_n \in E$ then
    track back as long $\sigma(e_1) \neq \sigma(e_{1_p})$ and highlight inconsistent names
  else
    continue
  end if
end for
```

The process in Figure 7 has a parallel branching, which is again integrated by transition **pack documents and products**. The first breakpoint for the algorithm is the branching at transition **forward order data**. As the sequence has only two elements, the algorithm traverses to the next branching. The algorithm regards elements from *order received* till *order sent* (the upper branch included) as one sequence and *order received* till *order sent* (the bottom branch included) as a second one. In the upper branch the algorithm finds no refinement possibility[6]. A consistent hierarchical decomposition would return none refinement possibilities in the bottom branch as well. But, in the bottom branch the algorithm finds an identical specificity for interrelated elements (**order product**, **product ordered**, **pick product to stock**[7], **products prepared** with similarity of 0.362) and detects this elements as non-uniformely specified (in case this processes are modeled as in Figure 1a). Specificity results for processes in Figure 1a are depicted in Table I.

TABLE I
LINGUISTIC SPECIFICITY RESULTS FOR PROCESS IN FIGURE 1A

name	noun	composed nouns	verb
order received	0,352	–	0,392
forward order data	–	0,352; 0,235	0,407
collected order data	–	0,352; 0,235	0,204
prepare documents	0,406	–	0,260
documents prepared	0,406	–	0,260
pack documents and products	0,406; 0,362	–	0,482
order sent	0,352	–	0,381
order received	0,352	–	0,392
forward order data	–	0,352; 0,235	0,407
product requirements agreed	–	0,362; 0,347	0,444
order product	0,362	–	0,352
product ordered	0,362	–	0,352
pick product to stock	0,362; 0,423	–	0,203
products prepared	0,362	–	0,359
pack documents and products	0,406; 0,362	–	0,482
order sent	0,352	–	0,381

To discover refinement pattern 2 requires more calculation efforts as for pattern 1. In Figure 9 we integrated the subprocess from Figure 5 with its subsuming processes. If the algorithm can not find refinement pattern 1, then it looks for pattern 2 and 3. The process in Figure 9 has only one branching at the place *order activated*. In our consideration the algorithm does not regard this as a branching (each branch consists only of one element - **coordinate small assignment** and **coordinate big assignment**), but each branching represents a breakpoint for the algorithm. The algorithm compares first the linguistic specificity of all elements between *production order* and *order activated* and then continues till *components tested*.

In Section 2 we have already mentioned that transitions are active components that change the state of the system. A "suitable" transition name expresses tasks of its pre– and postsets. In the following we assume suitable transition names and will only consider linguistic specificities of transitions instead of places. For (semi-)automatic discovery of refinement pattern 2 we have identified two influences. First, the average of transitions' linguistic specificities μ, and second, higher specificities for interrelated elements as the transitions' pre– and postsets. A pseudo-algorithm for pattern 2 can be implemented as follows (see Algorithm 2). For each sequence B the algorithm detects inconsistent hierachical specification for a set of elements $e_1, .., e_n$ from E_B if the specificity of interrelated transitions is higher as the transition average σ or as the transitions' pre– and postsets.

[6]even no pattern 2 or 3 as explained below

[7]nouns linked with nouns via prepositions are regarded as two mutual exclusive nouns

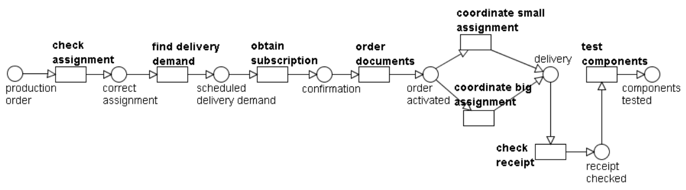


Fig. 9. One composed Process for Process Levels in Figure 5

Algorithm 2 Refinement Pattern 2

Input: $B_1, ..., B_n$ shall be a set of sequences with individual sets of elements E_{B_i}; $i \in 1...n$.
$\sigma(e_1), .., \sigma(e_n)$ shall be the linguistic specificities for all elements in E_{B_i}; e_{1_p} and $e_{n_{po}}$ shall be the pre– and postset of the first and last element in a set of elements in E_B.
$\mu(\sigma(e_{t_i}))$ shall be the average linguistic specificity of all transitions in $E_{T_i} \in E_{B_i}$; $i \in 1...n$.
Output: Elements belonging to refinement pattern 2
Method: Perform the following steps:

```
for all $E \in E_{B_1}, .., E_{B_n}$ do
  if $|E| \geq 4$ and $(\sigma(e_{t_i}) > \mu(\sigma(e_{t_i})) \vee \sigma(e_{t_i}) > (\sigma(e_{1_p}) \vee \sigma(e_{n_{po}})))$ then
    highlight non-uniformed names
  else
    continue
  end if
end for
```

The average $\sigma(e_{t_i})$ of transitions specificities of the process depicted in Figure 9 is 0.434. The linguistic specificity of **check assignment**, **find delivery demand**, **obtain subscription**, and **order documents** are higher than the average (0.455; 0.488; 0.445; 0.508). In the second sequence the system finds no instance name with a higher linguistic specificity as the average and thus no refinement candidate. If a modeler would model a process in that way as illustrated in Figure 9, then the system would detect an inconsistently hierarchical decomposition for these four elements. Calculation results are depicted in Table II.

Ideally, element names in subprocesses can be regarded as a specialization of the refined transition name (pattern 3). As depicted in Figure 6 the transition to be refined is named *edit order* and the elements in the subprocess are named *contract application* and *contract certificate*. Then, these two objects

TABLE II

LINGUISTIC SPECIFICITY RESULTS FOR PROCESS OF FIGURE 9

name	noun	composed nouns	verb
production order	–	0,515; 0,360	–
check assignment	0,560	–	0,350
correct assignment	0,560	–	0,430
find delivery demand	–	0,307; 0,225	0,444
scheduled delivery demand	–	0,307; 0,225	0,444
obtain subscription	0,498	–	0,392
confirmation	0,432	–	0,392
order documents	0,501	–	0.515
order activated	0.515	–	0.515
coordinate small assignment	0,560	–	0,337
coordinate big assignment	0,560	–	0,337
delivery	0,307	–	0,337
check receipt	0,415	–	0,350
receipt checked	0,415	–	0,350
test components	0,261	–	0,342
components tested	0,342	–	0,261

can be aggregated to the term *contract*, which has a lower specificity as the nouns to be aggregated. Coarsening pattern 3 will be discovered via a higher specificity than the transitions' pre– and postsets.

VI. APPLICATION: USER SUPPORT FOR PROCESS MODELING

Consistent process modeling requires a couple of modeling experiences. The measuring of process element abstraction can support users during business processes modeling (the initial idea of this support functionality is presented in [19]). The idea of an autocompletion mechanism is illustrated in Figure 10.

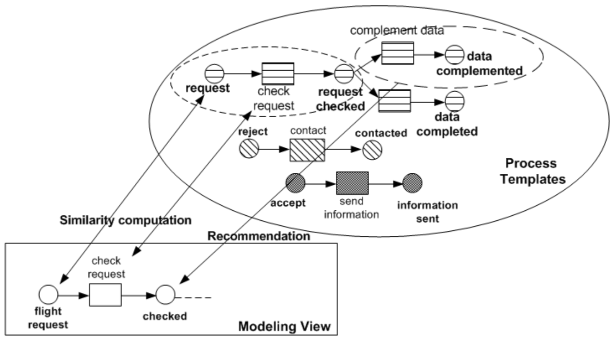


Fig. 10. Overview of the autocompletion process

Manual modeling of business processes is a time consuming task. Typos and structural modeling errors make it particularly error prone to model business processes manually. Therefore, it would be useful to assist the user in modeling business processes by providing an autocompletion mechanism during process modeling. Before proposing appropriate process elements, a recommendation mechanism has to suggest appropriate completions to initial process fragments which may be based on business rules and structural constraints. Thus, the mechanism has to compare process templates with process elements, which are currently modeled, by computing their abstraction homogeneity. In this scenario the name of the initial process element will be compared with all process elements (stored in a process repository) and the similarity between this names is computed as introduced in Section IV. Instance names with different linguistic specificity as the initial process element will not be proposed as fitting elements.

VII. IMPLEMENTATION AND EVALUATION

The computation of linguistic specificities have been performed with the Perl modules of Pedersen et al. [20]. Several seconds are needed to load WordNet and after that the specificity calculations for any concept are barely immediate. The result list of linguistic specificities returns for each element name several linguistic specificities as depicted in Table III. The semantic correspondence (description) of a term is chosen manually.

The presented algorithms have been implemented as an integrated tool suite based on the Eclipse workbench framework[8]. Processes used in our evaluation are real use cases collected in several practical modeling projects. After several manual evaluations we are aiming to implement the presented algorithms in a way to enable (semi-)automatical detection of inconsistencies of process element names on the same decomposition level.

VIII. RELATED WORK

The decompositon of subsystems through hierarchic classification of process models has been proposed for modeling languages such as Petri nets [2], Event Driven Process Chains [21] or more novel orchestration and choreography languages such as BPEL [22]. Several tools for process modeling make it possible to insert process hierachies via refinement concepts [23], [24].

In Section IV we introduced several related works for specificity calculation. Furthermore, we mentioned arguments in favor of our calculation approach.

To the best of our knowledge there is no other approach that has described high-level Petri nets with OWL.

Top-down and bottom-up theories have been proposed for programming languages in order to understand the complexity of software systems [25]. [26] has developed a software from the bottom-up that extends industrial machinery with Semantic Web technology to enable automated service discovery, customization, and semantic translation. [27] describes how patterns from other areas of computer science can be used as "templates" for creating ontology design patterns for automatic construction of enterprise ontologies.

To the best of our knowledge our work is the first theoretical approach to support the feature of (semi-)automatic detection of non-uniformely specified process elements on the same abstraction level.

IX. CONCLUSION AND FUTURE WORK

Subprocesses enable to modularize large business processes and to faster reuse process models. In general process decompositions appear to be more intuitive and easier to understand.

[8]http://www.eclipse.org/

TABLE III

RESULTS OF LINGUISTIC SPECIFICITY

name	type	$\frac{-\log P(c_i)}{max_{c_x \in \mathcal{C}}\{-logP(c_x)\}}$	description
decision	n	0.3372927	(psychological_feature, cognition, content, belief, opinion, judgment, decision)
decision	n	0.2488159	(act, action, choice, decision)
decision	n	0.2919773	(event, happening, ending, result, decision)
decision	n	0.289124	(abstraction, attribute, trait, resoluteness, decision)
final_decision	n	0.3165337	(act, group_action, due_process, judgment, final_decision)
provide	v	0.3098497	(transfer, give, support, provide)

Maintaining same level of abstraction guarantees business process consistency and comprehensibility. To maintain same abstraction for each hierarchy level and to check process consistency require significant modeling experience. The aim of our approach is to automate the procedure of process decomposition analysis to a significant extent. In this paper we presented an approach that supports the detection of non-uniformely specified process elements. In order to enable the analysis of abstraction homogeneity and heterogeneity we described how linguistic specificities for element names can be computed. Three so-called refinement patterns were illustrated to help subsuming elements to one of the patterns and then detecting modeling inconsistencies due to the height of its linguistic specificity. Processes with consistent hierarchical specifications guarantee correct process analysis and workflow views. We are planning to use the implementation of our approach to analyze further real-world business processes.

REFERENCES

[1] P. Huber, K. Jensen, and R. M. Shapiro, "Hierarchies in Coloured Petri Nets," in *Proceedings of the 10th International Conference on Application and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, vol. 483. Bonn: Springer, 1990, pp. 313–341.

[2] W. Reisig and G. Rozenberg, *Lectures on Petri Nets: Basic Models*, 1st ed., ser. Lecture Notes in Computer Science, 1998, vol. 1491.

[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," 02 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[4] A. Koschmider and A. Oberweis, "Ontology based Business Process Description," in *Proceedings of the CAiSE-05 Workshops*, ser. CEUR Workshop Proceedings, J. Castro and E. Teniente, Eds., vol. 160. Porto, Portugal: CEUR-WS.org, June 2005, pp. 321–333.

[5] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring Similarity between Semantic Business Process Models," in *Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)*, ser. Australien Computer Science Communications, J. F. Roddick and A. Hinze, Eds., vol. 67, Ballarat, Australia, 2007.

[6] W. M. van der Aalst, J. Desel, and A. Oberweis, Eds., *Business Process Management: Models, Techniques, and Empirical Studies*, ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1806.

[7] W. M. van der Aalst, "The Application of Petri Nets to Workflow Management," *The Journal of Circuits, Systems and Computers*, no. 8, pp. 21–66, 1998.

[8] K. Jensen, "An Introduction to the Theoretical Aspects of Coloured Petri Nets," in *A Decade of Concurrency – Reflections and Perspectives*, ser. Lecture Notes in Computer Science, J. de Bakker, W. P. de Roever, and G. Rozenberg, Eds. Springer, 1994, vol. 803, pp. 230–272.

[9] D. G. Stork and R. van Glabbeek, "Token-controlled place refinement in hierarchical petri nets with application to active document workflow," in *Proceeding of the 23rd International Conference on Applications and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, J. Esparza and C. Lakos, Eds., vol. 2360. Adelaide, Australia: Springer, 2002.

[10] C. Fellbaum, Ed., *WordNet: An electronic lexical database*. MIT Press, 1998.

[11] A. Budanitsky, "Lexical semantic relatedness and its application in natural language processing," Computer Systems Research Group - University of Toronto, Tech. Rep., 1999.

[12] N. Guarino and P. Giaretta, "Ontologies and Knowledge Bases: Towards a Terminological Clarification," in *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, N. Mars, Ed. IOS Press, 1995, pp. 25–32.

[13] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 17–30, 1989.

[14] Z. Wu and M. Palmer, "Verb semantics and lexical selection," in *Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics*, 1994, pp. 133–138.

[15] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, 1998, pp. 296–304.

[16] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 1, 1995, pp. 448–453.

[17] E. Blanchard, P. Kuntz, M. Harzallah, and H. Briand, "A tree-based similarity for evaluating concept proximities in an ontology," in *Proceedings of the 10th Conference of the International Federation of Classification Societies*. Ljubljana, Slovenia: Springer, July 2006, pp. 3–11.

[18] G. Lausen, "Modeling and Analysis of the Behavior of Information Systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 11, pp. 1610–1620, 1988.

[19] T. Hornung, A. Koschmider, and A. Oberweis, "Rule-based autocompletion of business process models," in *CAiSE Forum 2007 Proceedings at the 19th Conference on Advanced Information Systems Engineering (CAiSE)*, Trondheim, Norway, JUN 2007, (to appear).

[20] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::similarity - measuring the relatedness of concepts," in *Proceedings of the Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 2004, pp. 1024–1025.

[21] A.-W. Scheer and M. Nüttgens, "ARIS Architecture and Reference Models for Business Process Management," in *Business Process Management, Models, Techniques, and Empirical Studies*, ser. Lecture Notes in Computer Science, vol. 1806. Springer, 2000, pp. 376–389.

[22] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, and I. Trickovic, "WS-BPEL Extension for Subprocesses BPEL-SPE," IBM and SAP," Joint White Paper, September 2005.

[23] Get process, "Income process designer," http://www.get-process.com/de/, 2006.

[24] IDS Scheer, "Aris toolset," http://www.ids-scheer.de, 2006.

[25] C. A. R. Hoare, "Theories of Programming: Top-Down and Bottom-Up and Meeting in the Middle," in *Correct System Design, Recent*

Insight and Advances, ser. Lecture Notes in Computer Science, vol. 1710. Springer, 1999, pp. 3–28.

[26] D. J. Mandell and S. A. McIlraith, “Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation,” in *Proceedings of the Second International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 2870. Springer, 2003, pp. 227–241.

[27] E. Blomqvist, “Fully Automatic Construction of Enterprise Ontologies Using Design Patterns: Initial Method and First Experiences,” in *OTM Confederated International Conferences*, ser. Lecture Notes in Computer Science, vol. 3761. Agia Napa, Cyprus: Springer, 2005.