Ontology Based Business Process Description

Agnes Koschmider and Andreas Oberweis

Institute of Applied Informatics and Formal Description Methods, Universität Karlsruhe,
76187 Karlsruhe, Germany
{koschmider,oberweis}@aifb.uni-karlsruhe.de

Abstract. Coupling of cross-organizational business processes in electronic markets is a difficult and time-consuming task. In practice business processes are geographically distributed which makes it particulary difficult for business partners to coordinate their supply chains and customer relationship management with business units. By using formal description languages such as Petri nets for modeling inter-organizational business processes, purely syntactic composition problems of distributed business environments can be solved. However, the missing semantic representation of Petri nets can hamper the interconnectivity of business processes. Usually, several business partners, even if they share similar demands, have their own specific vocabularies. By representing business processes with Petri nets in combination with the Web Ontology Language (OWL) our approach provides flexibility, ease of integration and a significant level of automation of loosely coupled business processes even if they do not share their respective vocabularies.

1 Introduction

Coupling of cross-organizational business processes in electronic markets is a difficult and time-consuming task. The integration of different business partners into one single value creation chain demands enormous coordination activities. Business processes of different companies have to fit in another organizational environment and they have to complement each other. By using Petri nets [26] for modeling inter-organizational business processes, purely syntactic composition problems of distributed business environments can be solved. Moreover, Petri nets obey an operational semantics that facilitates composition, simulation, and validation of business processes. However, a missing semantic representation of Petri net components can hamper the interconnectivity of business processes. Usually, several business partners, even if they share similar demands, have their own specific vocabularies. Furthermore, the rapid growth of electronic markets´ activities demands flexibility and automation of involved systems in order to facilitate the interconnectivity of business processes and to reduce communication efforts. Semantic markup of business process models and automated reasoning is required. An effective approach for improving distributed systems communication can be provided by metadata-descriptions of the related business objects. In order to reduce negotiation efforts, these metadata-descriptions should be interpretable by machines. A necessary prerequisite for ma-

chine-interpretable metadata and (semi-)automated system cooperation is the availability of detailed knowledge about the underlying business process. Furthermore, not only the syntax but also the application semantics of business process describing metadata must be considered. The syntax defines the structure of data and can be represented in XML notation. The Petri Net Markup Language (PNML) [28] is a popular proposal of an XML based interchange format for Petri nets. Semantic Web languages such as the Resource Description Framework (RDF) [30] and the Web Ontology Language (OWL) [29] were proposed to make it particularly easy to model data in a machine-interpretable form. Based upon RDF, a resource description language for modeling metadata, OWL aims to describe semantic metadata in a computer-interpretable markup. Thus, OWL may enable automation of a variety of tasks currently being performed "manually" by human agents [2].

To make data computer-interpretable has become ever more important since recent Web Services standards have paved the way for discovery and matching of semantically enriched data and services. Process modeling languages such as BPML [1], WSFL [18] and more recently BPEL4WS [5] enable users to compose and orchestrate services to perform certain tasks. But these modeling languages do not yet support analysis methods to verify that business processes meet certain requirements. In order to allow flexible automation and composition of semantic representations of web services, OWL-S (OWL for Services) was proposed [24]. Due to the lack of formal semantics in the OWL-S 1.0 specification, McIllraith and Narayanan use Petri nets to test and verify the composition of Web Services based on OWL-S [21]. A lot of research is currently being done on automated provision and reasoning of Web Services [4, 21, 27]. Petri nets can be used to concisely represent and analyze distributed business processes and are utilized to model inter-organizational processes. Moreover, Petri nets are suitable both for modeling business processes, which are to be implemented as web services, and their coordination [17]. But for interconnectivity and business process coupling executed by machines, semantic representation of business units remains a challenge and has to be addressed by research. In summary, our objective is to provide flexibility, ease of integration and a significant level of automation of loosely coupled business processes even if they do not share their respective vocabularies.

The structure of this paper is as follows. Firstly, we recall the main notions of Petri nets. Secondly, we present a novel process ontology for Petri nets. Thirdly, we elucidate how the petri net ontology can be realized with OWL elements and introduce shortly into the area of ontology mapping techniques and their task to work around ambiguity issues caused by the use of different ontology elements. The development of a tool for modeling ontology based Petri nets is described in the next section. Finally, we discuss open problems and give an outlook on future work.

2 Distributed Business Processes

To concisely represent and analyze distributed business processes different variants of Petri nets have been proposed [26]. Moreover, Petri nets can be utilized to model inter-organizational processes [16]. Formally, a Petri net is a bipartite graph consist-

ing of places (drawn as circles) and transitions (drawn as rectangles). Places and transitions may be connected by directed arcs. Transitions are interpreted as dynamic elements and represent actions or activities of a process. Conditions for the execution of activities are described by places. In elementary Petri nets (place/transition nets) tokens representing anonymous objects define the process flow. When a transition fires tokens are removed from its input places and tokens are inserted into its output places.

For modeling business processes and workflows with identifiable objects high-level Petri nets such as predicate-transition nets (Pr/T nets) [10], Coloured Petri nets (CPN) [15] or XML nets [16] have been proposed. In high-level Petri nets tokens in places represent objects with individual properties. In Pr/T nets places are regarded as relation schemata which define admissible markings of the respective place. A marking of a place is given as a relation of the respective schema, i.e. a set of tuples. When a transition fires, tupels are removed from the transition's input places and inserted into the transition's output places according to the respective arc inscriptions. Figure 1 shows the Pr/T net representation of product order and delivery processes of two business partners.

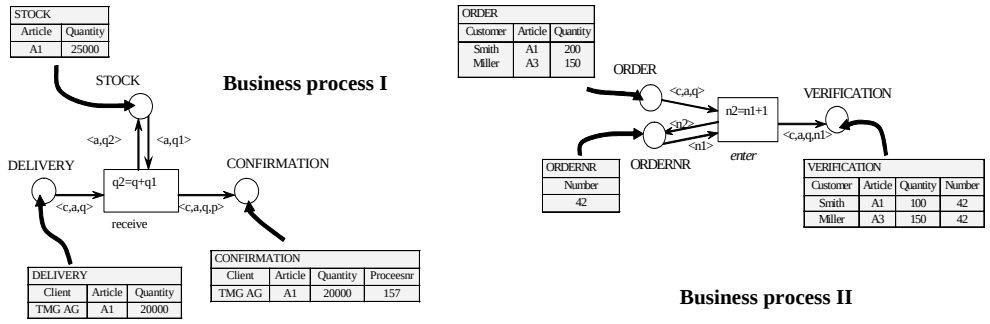


Fig. 1: Pr/T net representation of business processes (excerpts)

Petri nets comprise an operational semantics for processes based on a formal interpretation of the net components and their dynamic behavior. However, collaboration between business partners requires that there is a common understanding of the real world meaning of the places and transitions. Furthermore, to facilitate semantic interconnectivity between business processes (semi)automated system cooperation is demanded. For this reason we describe an ontology based extension for business process models in the following section.

3 An Ontology for Business Processes

Our approach is based on defining semantic metadata for business processes modeled with Petri nets. This makes it particularly easy to automate the communication among process-implementing software components. Our starting point is a concise specification of Petri net elements with the OWL elements *Classes*, together with the taxonomic construct *SubClassesOf* and *Properties*. Every individual in the OWL world is a member of the class `owl:Thing`. Thus each user-defined class is implicitly a subclass of `owl:Thing`. In the next step we describe some constructs of the ontology modeled with OWL. If software components of different business partners should interact it must be known what is represented by a place, the meaning of objects contained in places and their relation to other objects.

Figure 2 shows the hierarchy of core elements of our novel Petri net ontology. The Petri net structure comprises the elements place, transition and arc, thus we categorize the Petri net elements in nodes (place and transition) and arcs (fromPlace and toPlace). We express this coherency by adding to the Petri net class the properties *hasNode* and *hasArc*. The main Petri net elements are modeled by corresponding classes. The class transition has as property a place reference (= *placeRef*). The subclass of transition is the class *logicalConcept* with the properties *hasOperation* and *hasAttribute*. In Figure 1 the transition `receive` of Business process I is described by the Operation to sum up the attributes `q` and `q1` to `q2`. In contrast, places are defined by transition reference (*transRef*) and their appropriate marking. Petri net marking depends on the Petri net type. In elementary Petri nets such as place/transition nets places may contain several tokens and a capacity limit representing the maximal capacity of a place. (*place – hasMarking – number*). A place of a condition/event net contains one or zero tokens, thus the marking is *indistinguishable*. The marking of a place in a Pr/T net is regarded as a set of tuples (*place – hasMarking – individualDataItem)*. As demonstrated in Figure 1 the marking of places are sets of individualDataItem with *attributes* and attribute *values*. To represent this structure of elements in our ontology we add to the class *individualDataItem* the property *hasAttribute*. The arcs between places and transitions describe different meaning, thus we distinguish between two types of arcs. The first one is directed from place to transition (fromPlace) and the other from transition to place (toPlace). An arc connecting a transition to a place indicates an *insert* operation, inserting for example attribute values into the transition´s output place. An arc connecting a place to a transition indicates a *delete* operation. In Figure 2 we added to the arcs between two classes cardinality restrictions, this describe quantitative dependencies between of two classes, for example a Petri net consists of 1 to * places.

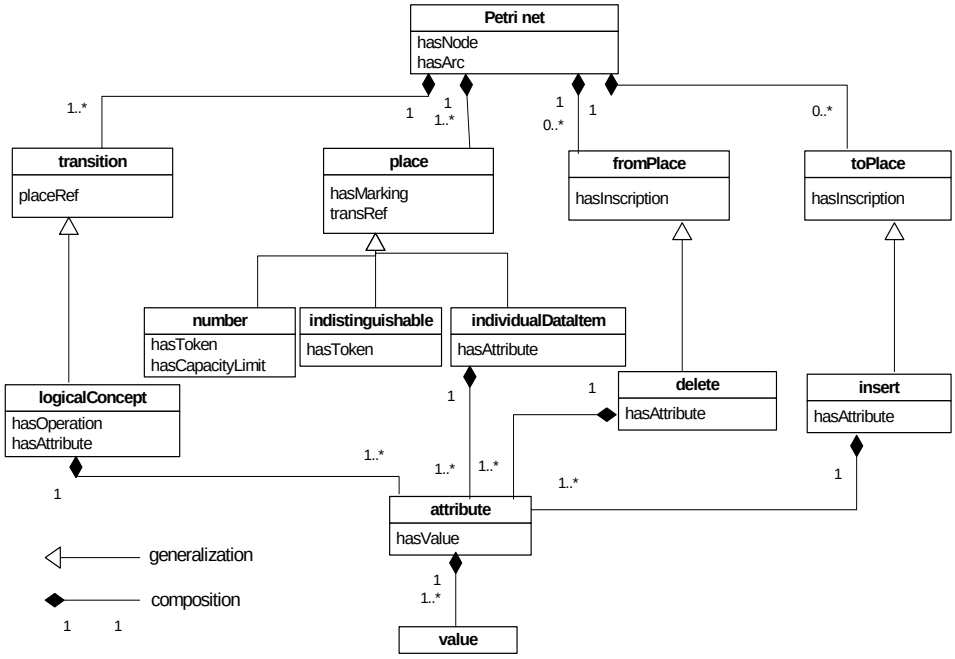


Fig. 2: An ontology for Petri nets – hierarchy of core Petri net concepts

Figure 2 shows the core concepts of the Petri net ontology. Note that an ontology language defines more constructs, e. g. specific properties, as mentioned above. In the next section we will describe the elements of the ontology in detail.

4 Realization

The OWL language provides three increasingly expressive sublanguages (OWL Lite, OWL DL and OWL Full). OWL DL (OWL Description Logics) places a number of constraints on the use of the OWL language constructs. In order to significantly automate the composition of loosely coupled business processes even with non-shared vocabulary we are using OWL DL. The sublanguage OWL Lite only uses some of the OWL language components, e.g. classes can only be defined in terms of named superclasses (superclasses cannot be arbitrary expressions), and only certain kinds of class restrictions can be used. OWL Full is not yet supported by reasoning software. With OWL DL determinable reasoning in ontologies is provided by SHIQ(D) [13].

The Web Ontology Language defines different properties such as Object Properties, (link an individual to an individual), Data Properties (link an individual to an XML Schema data type value or to an rdf literal), Domains and Ranges (properties link individuals from one domain to individuals from another domain), Datatypes and Restriction Types (Quantifier Restrictions, hasValue Restrictions and Cardinality Restrictions) to build an ontology.

The SubClasses *place* and *transition* of the *Petri net* class have to be defined as *Disjoint Classes* such that a single individual cannot be an instance of more than one of these two classes. The disjointness of a places and transitions can be expressed using the `owl:disjointWith` constructor:

```
<owl:Class rdf:about="#transition">
  <owl:disjointWith rdf:resource="#place"/>
  <rdfs:subClassOf rdf:resource="#PetriNet"/>
</owl:Class>
```

Figure 2 shows that the property *hasAttribute* is included in the classes *logicalConcept, individualDataItem, delete and insert.* To express this coherence OWL provides the `owl:unionOf` construct. The OWL-code is as follows:

```
<owl:ObjectProperty rdf:about="#hasAttribute">
   <rdfs:domain>
     <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#individualDataItem"/>
       <owl:Class rdf:about="#delete"/>
       <owl:Class rdf:about="#insert"/>
       <owl:Class rdf:about="#logicalConcept"/>
      </owl:unionOf>
      </owl:Class>
     </rdfs:domain>
   <rdfs:range rdf:resource="#attribute"/>
</owl:ObjectProperty>
```

For modeling inverse properties OWL proposes the OWL Property Characteristics `owl:inverseOf`. For a given individual, there can be at most one individual related to that individual via the property.

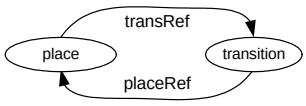


Fig. 3: `owl:inverseOf` Property

The description of the inverse property shown in Figure 3 including the range and domain of the ObjectProperty is as follows:

```
<owl:ObjectProperty rdf:ID="transRef">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="placeRef"/>
    </owl:inverseOf>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
```

```
          <owl:Class rdf:about="#toPlace"/>
          <owl:Class rdf:about="#transition"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#place"/>
          <owl:Class rdf:about="#fromPlace"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
```

Important constructs in OWL are different restriction types, which are used to restrict the individuals that belong to a class. Restrictions in OWL fall into three main categories: Quantifier Restrictions (allValuesFrom, someValuesFrom), Cardinality Restrictions (minCardinality, maxCardinality, cardinality) and hasValue Restrictions. Quantifier Restrictions specify the exact number of relationships that an individual must participate in for a given property. In our Petri net ontology we denote that the class *individualDataItem* has at least one *attribute*:

```
<owl:Class rdf:ID="individualDataItem">
 <rdfs:subClassOf>
  <owl:Restriction>
  <owl:minCardinality rdfdatatype=http://www.w3.org/2001/XMLSchema#int>1
  </owl:minCardinality>
  <owl:onProperty>
   <owl:ObjectProperty rdf:ID="hasAttribute"/>
  </owl:onProperty>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Quantifier restrictions consist of three parts:

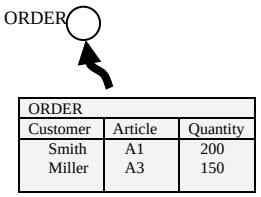
1. A quantifier, which is either the existential quantifier ($\exists$), or the universal quantifier ($\forall$)
2. A property, along which the restriction holds
3. A filler that is a class description

For a given individual, the quantifier effectively puts constraints on the relationships that the individual participates in. This is done by specifying that at least one kind of relationship must exist, or by specifying the only kinds of relationships that can exist. Existential restrictions describe the set of individuals that have at least one specific kind of relationship to individuals that are members of a specific class. In our ontology, a restriction is defined that the arc inscriptions (*fromPlace*) are defined by individuals from the class *attribute*:

```
<owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="hasInscription"/>
   </owl:onProperty>
   <owl:someValuesFrom>
   <owl:Class rdf:ID="attribute"/>
```

```
   </owl:someValuesFrom>
</owl:Restriction>
```

Our starting point was a concise specification of Petri net elements with the OWL element *Class*, the taxonomic constructor *SubClassesOf* and *Property* and their modeling in OWL. In the following we show the modeling of *Individuals* which is the third OWL element besides *Classes* and *Properties*. Individuals or instances are specified by the modeler and depend on the modeling target. As an example, we show for the place `ORDER` of business process II in Figure 1 mapping individuals to the OWL elements.

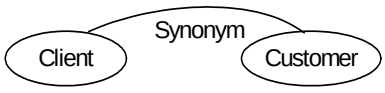


ORDER		
Customer	Article	Quantity
Smith	A1	200
Miller	A3	150

```
<place rdf:ID="ORDER">
   <hasMarking>
    <initial_individualDataItem
rdf:ID="R_ORDER">
    <hasAttribute>
     <attribute rdf:ID="Customer">
       <hasValue rdf:resource="#Smith"/>
       <hasValue rdf:resource="#Miller"/>
     </attribute>
     <attribute rdf:ID="Article">
       <hasValue rdf:resource="#A1"/>
       <hasValue rdf:resource="#A3"/>
     </attribute>
     <attribute rdf:ID="Quantity">
      <hasValue rdf:resource="#200"/>
      <hasValue rdf:resource="#150/>
     </attribute>
    </hasAttribute>
   </initial_individualDataItem>
   </hasMarking>
   ….
   ….
```

Fig. 4: Mapping Individuals to Classes and Properties

In business relationships a commonly agreed vocabulary can usually not be postulated. In Figure 1, e.g., the business partners use different terms having the same meaning. Business partner I utilizes “Client” for customer and business partner II “Customer”. Another example for synonyms is “Article” and “Position”. To express synonyms in OWL the construct `owl:equivalentClass` is utilized. Equivalent classes have the same instances. From this a reasoner can deduce that any individual that is an individual of “Client” is also an individual of “Customer” and vice versa.




```
<owl:equivalentClass>
  <attribute rdf:ID="Customer">
    <owl:equivalentClass
rdf:resource="#Client"/>
  </attribute
</owl:equivalentClass>
```

Fig. 5: Construct owl:equivalentClass

To automatically find synonyms and antonyms OWL mapping techniques for *Individuals* are required. Mapping expressions enable translating data from one source to the other. Thus, transferring source ontology $Individuals^S$ to target ontology $Individuals^T$ according to the semantic relations of both is required [23]. An auto-

matic finding of synonyms and antonyms is needed because usually several business partners, even if they share similar demands, have their own specific vocabularies. In practice an agreement of using a common vocabulary for interpreting *places* and *transitions* cannot be postulated. Mistakes appear in interpreting objects contained in places: the attribute organization (in the sense of planning, administration), e.g., is unequal to organization (in the sense of company). Furthermore, ambiguities are caused by [9]:

- utilizing different items for the same issues (synonyms)
- unequal units (€, $)
- different abstraction levels (name vs. first name and last name)
- diver item structure for complex data types (for example address)

A lot of research is currently being done in the field of mapping techniques for ontologies, for examples see, e.g. [6, 7, 19].

5 Implementation

A Petri net ontology has to be created by using an OWL editor such as Protégé[1] or an editor included in the Semantic Web Development Environment (SWeDE)[2]. SWeDE provides syntax highlighting, autocompletion, and error-detection. Furthermore, in the SWeDE framework the API generation tools Kazuki and Jena SchemaGen are integrated. Kazuki generates Java interfaces for objects contained in an OWL ontology file, based on the structure of the ontology. It is build on Jena2. Jena SchemaGen generates a Java vocabulary class for use with the Jena2 libraries. Jena2 is the most popular ontology management system, an opensource Java framework for writing Semantic Web applications [11]. By creating OWL files with Protégé, the Jena2 API can read the OWL files generated by the editor, and apply changes to the model. For storing the OWL files in Jena2 a database management system is not required. But Jena2 supports relational database management systems such as MySQL, Oracle and PostgreSQL for persistent storage.

The extraction of ontological descriptions of business processes and the mapping to the Petri net ontology is being carried out during the modeling process and is not directly visible to the modeler. The user can model his business processes using a graphical business process editor as shown in Figure 6. After modeling business processes the models can be exported to OWL code and afterwards be sent to the respective business partners. The ontology management system of the business partner is needed for parsing and interpreting the data contained in the Petri net. An ontology management system is not only needed for utilizing mapping techniques, but also for reasoning about the data.

An appropriate tool Ontology Business Processes Modeler (OBPM) is currently being developed. For modeling business processes with Petri net elements (*place*, *transition* and *arc)* a Petri net editor can be used. The relationship to our novel process ontology is provided by the specification of data contained in Petri net elements.

[1] http://protege.stanford.edu/

[2] http://owl-eclipse.projects.semwebcentral.org/

SubClasses of the class *place – name (=individualDataItem*), *attribute* and *value* - are fixed. The user has to insert the Individuals of the classes/subClasses by his own. Describing *arcs* and *transitions* can be applied accordingly like specifying data contained in *places*. By inserting a place a corresponding window will be opened.

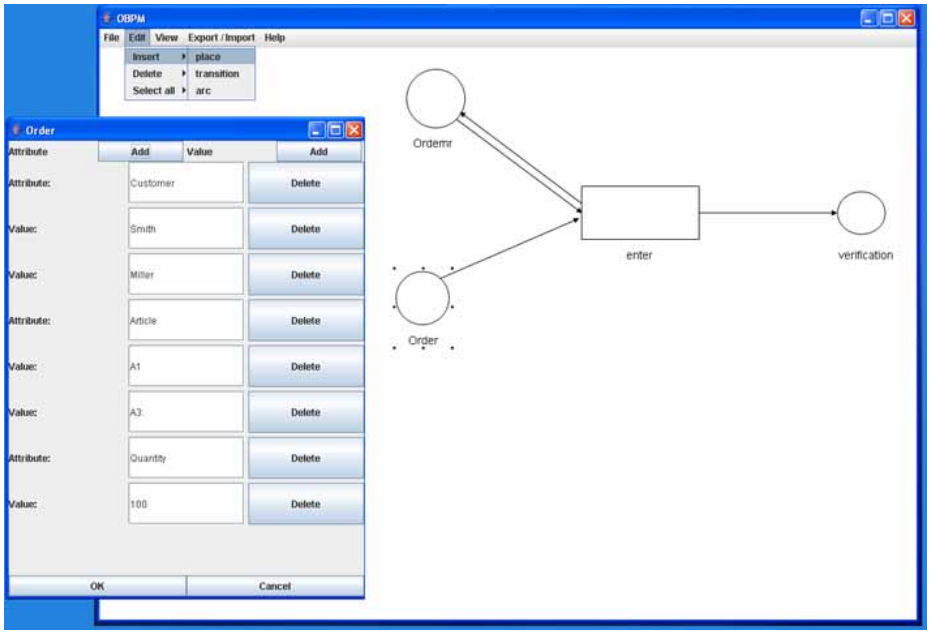


Fig. 6: A graphical tool for modeling ontology based business process descriptions

6 Conclusion and Outlook

The task to make business data computer-interpretable has become ever more important since recent Web Services standards have paved the way for discovery and matching of semantically enriched data and services. Furthermore, the rapid growth of data and communication technologies demand to companies to focus on the content of data. Our approach provides semantic markup of Petri nets and enables to interpret Petri net content by machines. With this semantics one can define restrictions and reason about the process data contained in Petri net components. Beyond the ontological representation of a Petri net we discussed the need of automated mapping techniques that enable structured data to be interpreted unambiguously. Finally, we presented an implementation approach and a tool for modeling ontology based business processes which is currently under development.

The benefits of our approach are flexibility and automation of involved systems in order to facilitate the semantic interconnectivity of business processes and to shorten communication among process-implementing software components.

By defining an ontology based business process description a basis for solving further open problems is provided. In the next step we will apply reasoning techniques such as SWRL [31] and SHIQ(D) [13] to reason about data contained in places. The use of reasoning rules referring to Figure 1 would be to answer questions such as “show all clients that received a confirmation when the stock quantity of article A1 was 25000”.

References

1. Arkin, A.: Business Process Modeling Language. http://www.bpmi.org/bpml.esp
2. Baader, F.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.: The Description Logic Handbook. Cambridge, 2003
3. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks, CACM 13, June 1970, pp. 377-387
4. Cordoso, J.; Sheth, A.: Semantic e-Workflow Composition, Technical Report, LSDIS Lab, Computer Science, University of Georgia, July 2002
5. Curbera, F.; Goland, Y., Klein, J., Leymann, F., Roller, D., S. Thatte, Weerawarana, S. Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/
6. Doan, A.; Halevy, A.Y.; Noy, N.F.: Introduction to the special issue on semantic integration, ACM SIGMOD Record, vol 33, (4), December 2004, pp. 11 - 13
7. Dou, D.; McDermott, D.; Qi, P.: Ontology translation by ontology merging and automated reasoning, in: Proceedings of the EKAW2002, Workshop on Ontologies for Multi-Agent Systems, Spain, 2002, pp. 3-18
8. Falkovych, K: Ontology Extraction from UML Diagrams, Master Thesis, Vrije Universiteit, Amsterdam, 27 August 2002
9. Gahleitner, E.; Wöß, W.: Enabling Distribution and Reuse of Ontology Mapping Information for Semantically Enriched Communication Services, in: 15th International DEXA Workshop, Zaragoza/Spain, August 2004
10. Genrich, H. J. ; Lautenbach, K.: System Modelling with High-Level Petri Nets, Theoretical Computer Science, vol. 13, 1981
11. HP: Jena 2 - A Semantic Web Framework: http://www.hpl.hp.com/semweb/jena.htm
12. Horrocks, I.; Patel-Schneider, P.F.: A Proposal for an OWL Rules Language, in: The 13th International World Wide Web Conference, New York, 2004, pp. 723-731
13. Hustadt, U.; Motik, B.; Sattler, U.: Reducing SHIQ Descrption Logic to Disjunctive Datalog Programs, in: Proceedings of the 9th International Conference on Knowledge Representation and Reasoning, Whistler/Canada, June 2004, pp. 152-162
14. ISO 11179: Information Technology-Specification and Standardization of Data Elements
15. Jensen, K.: A Brief Introduction to Coloured Petri Nets, in: Brinksma, E.: Lecture Notes in Computer Science, Vol. 1217: Tools and Algorithms for the Construction and Analysis of Systems. Proceedings of the TACAS'97 Workshop, Enschede/The Netherlands 1997, pages 201-208. Springer-Verlag, 1997
16. Lenz, K.; Oberweis, A.: Interorganizational Business Process Management with XML Nets, In H. Ehrig, W. Reisig, G. Rozenberg, H. Weber, Petri Net Technology for Communication-Based Systems, Advances in Petri Nets vol. 2472 of LNCS, pp. 243-263. Springer-Verlag, 2003.
17. Lenz, K.; Oberweis, A.: Workflow Services: A Petri Net-Based Approach to Web Services, in: Proceedings of Int. Symposium on Leveraging Applications of Formal Methods, Paphos/Cyprus, November 2004, pp. 35-42

18. Leymann, F.: Web Services Flow Language. http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
19. Maedche, A. et alt: MAFRA - An Ontology Mapping Framework in the Semantic Web, Proceedings of the ECAI Workshop on Knowledge Transformation, Lyon, July 2002
20. Magkanaraki, A. et al.: Benchmarking RDF Schemas for the Semantic Web, First International Semantic Web Conference, Sardinia, June 2002
21. Martin, D. et al.: Bringing Semantics to Web Services: The OWL-S Approach, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, San Diego, July 2004
22. McIlraith, S.; Narayanan, S.: Analysis and Simulation of Web services, in: Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 42 (5), August 2003, pp. 675 – 693
23. Silva, N.; Rocha J.: Semantic Web Complex Ontology Mapping, in: Web Intelligence and Agent Systems Journal; IOS Press; 2003; 1(3-4); pp. 235-248
24. The OWL Services Coalition. OWL-S: semantic Markup for Web-Services. http://www.daml.org/services, 2004
25. Powers, S.: Practical RDF, 1. ed. Beijing; Köln: O´Reilly, 2003
26. Reisig, W.; Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models. Lecture Notes in Computer Science, Vol. 1491, Springer-Verlag, Berlin, 1998
27. Shahmehri, N.; Takkinen, J.; Åberg, C.: Towards Creating Workflows On-the-Fly and Maintaining Them Using the Semantic Web: The sButler Project at Linköpings universitet, Presented as a poster at The 12th International World Wide Web Conference, Budapest, May 2003
28. Weber, M.; Kindler, E.: The Petri Net Markup Language, In: Ehrig, H.: Petri Net Technology for Communication-Based Systems, Advances in Petri Nets Berlin, Springer, 2003, S. 1-21
29. W3C. OWL Web Ontology Language Overview, February 2004, Recommendation, http://www.w3.org/TR/owl-features/
30. W3C. RDF Primer, Februar 2004, Recommendation, http://www.w3.org/TR/rdf-primer/
31. W3C. Semantic Web Rule Language Combining OWL and RuleML, May 2004, W3C Member Submission, http://www.w3.org/Submission/SWRL/#1