COLLABORATIVE INSTANTIATION OF TOPIC MAPS AND OWL ONTOLOGIES

Lutz Maicher[1], Martin Böttcher[2]

Abstract

The problem of collaborative knowledge management is the distributed documentation of facts according to defined schemas or ontologies. But the precise instantiation of ontologies is left to the user or the application layer. To achieve a better collaborative approach the instantiation process should be disclosed together with the ontology. The necessity of disclosure is due to the fact, that an ontology inherently implies a set of different model types. We call that fact the semantic gap in ontologies. This paper proposes a solution for a disclosure based on a generic, workflow-based description of the instantiation method: the Modelling Workflow Patterns (MWP). Based on Petri nets as information model, MWPs can be processed by generic interpreters to create valid instances of the specified model type. This paper presents an implemented architecture consuming MWPs for Topic Maps and OWL ontologies.

1. The Challenge of Collaborative Knowledge Management

Due to an ever evolving information technology, better logistics of information and goods and increasing interdependencies between markets, the *collaboration* of companies and organisations is a promising constellation for meeting new challenges. Knowledge Management is a classical logistic problem: providing the appropriate information to support persons in decision processes, to help these persons to assess and recombine the delivered information. For our work we define *collaborative knowledge management* as the creation, distribution, assembly and usage of information among distributed and independent people, organisations and companies. This leads to the following requirements:

- ability to *semantically* integrate distributed information,
- which is documented in consistent terms or vocabularies and
- is stored in system independent formats.

Semantic approaches try to tackle the mentioned problems and they are appropriate for certain aspects of collaborative knowledge management. Ontologies or schemas are given, to support users

[1] University of Leipzig, Augustusplatz 10-11, 04109 Leipzig, Germany. maicher@informatik.uni-leipzig.de
[2] University of Leipzig, Augustusplatz 10-11, 04109 Leipzig, Germany boettcher@informatik.uni-leipzig.de

to document relevant information. Information documented according a given schema or ontology can be distributed, assembled and used in a convenient way, especially in distributed and heterogeneous environments.

As we discuss in the following section in full detail, one of the main arsing problems are the Semantic Gaps in ontologies. Simply spoken, semantic gaps are the interpretation spaces each language provides. Given one ontology, a variety of derived model types is imaginable. To close these semantic gaps, the modelling methods to apply will be delivered (as workflows) together with the ontology. Using this solution, distributed and heterogeneous people will document identical observations with identical statements. This allows a semantic integration of information collected in the collaborative environment to support the information logistics in a better way.

Our approach allows a very flexible integration of ontology based information collection tools in desktop production systems. Furthermore, updating and distributing these modelling methods is very simple.

We want to outline, that this paper does describe the necessity, the benefit and some core aspects of disclosing the modelling methods together with the ontology. The interested reader is invited to read the full technical specification in [1], which are not the scope of this paper.

The remainder of the paper is organised as follows. The next chapter describes the semantic gaps in ontologies followed by a detailed description of the benefits of disclosing the modelling methods. In chapter 4 the design decisions of MWPs are described. The architecture of an implemented MWP infrastructure is given in chapter 5. The subsequent chapter gives some details about the basic Petri net data model representing the workflow. In chapter 7 the state of the art is described, followed by a short summary and outlook in the last chapter.

2. Semantic Gaps in Ontologies

The main purpose of the approaches usually developed under the label Semantic Web is to gain better results for interoperability and integration tasks [2]. Thereby the usage of ontologies is usually seen as the silver bullet to solve the upcoming challenges [3]. But there are blind spots, which are often ignored by using ontologies in real world applications. This paper analysis these blind spots which are called semantic gaps, and proposes a solution with benefits for a wide range of applications, especially in collaborative knowledge management.

A computer ontology is said to be an agreement about shared, formal, explicit and partial account of conceptualisations of specific parts of the world. Ontologies define a shared vocabulary and domain rules for the documentation of observations about subjects[3] [4][5]. Although these conceptualisations can be expressed by any data modelling language, within the Semantic Web OWL [6] or Topic Maps [7] are usually used for these purposes.

The rationale for using ontologies seems to be simple: based on the assumption, that ontologies provide the *shared* vocabulary to make statements about subjects, it is usually concluded that

[3] "A subject is anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever." [7] Thereby, a subject can be a concept of a domain or an instance of such domain concepts.

provided this shared vocabulary authors will document identical observations about the world with identical expressions of the provided language.

A short side glance at natural languages shows that provided a shared vocabulary the same fact can be described in a big variety of syntactically and semantically correct sentences.

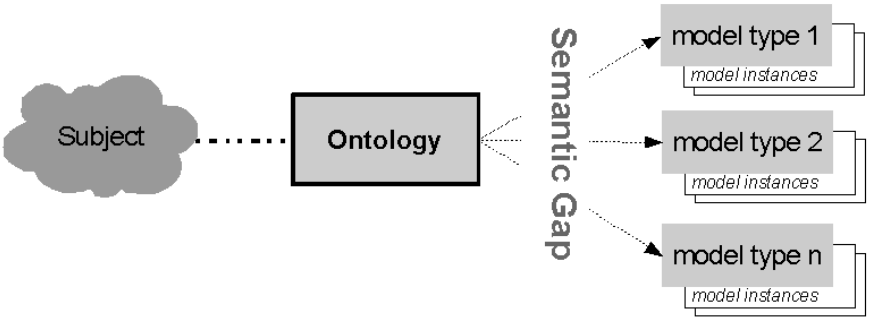

Figure 1: The Semantic Gap in Ontologies

Figure 1 describes the common problem which we call the semantic gap in ontologies: the usage of one ontology allows the creation of various independent, still valid model types, due to the interpretation spaces left by providing only the vocabulary. Closing the semantic gap means to define the *intensions* of each model type by the description of the modelling method to apply.

A modelling method is the manner how the observation of subjects should be documented with the given ontology. For example, there might be two different kinds to observe all employees of an enterprise: the first is requesting the human resources database; the second is running through the offices and noting all people found. Both kinds of observations can be described by using the same ontology, but statements represent different subjects. The two resulting employee lists usually do not contain any knowledge about the applied modelling method. In fact, two syntactically equivalent model instances of two unequal model types are obtained.[4]

Another point concerning the importance of the modelling method is that for a certain task a certain part of the ontology needs to be used and finally instantiated. Unfortunately this is not documented within the ontology. For instance it might be necessary for a collaboration in the automotive industry to add information to the ontology concerning the supply chain. It is not clear whether it is sufficient to add the name of direct predecessor or whether location, dependencies to other suppliers and the amount of employees is necessary too. That leads to the problem, that collaborating enterprises might instantiate an ontology due to a certain task in different ways. Therefore, a modelling method represents a specific *work sequence* (respectively workflow) that needs to be applied to fulfil a certain *task* that usually addresses a certain section of the concepts an ontology provides. As discussed below, a use case based view on ontologies is very valuable at the creation time of model instances.

[4] As discussed in conjunction with the Observation Principle [5][8] only observations of *subject stages* are documented. It depends on the *perspective* taken at integration time, whether subject stages observed at different occasions belong to the same subject [2]. Only in the case subject equality holds, it can be assumed, that the assertions about these subject stages belong to the same subject. There might be a perspective where the documentation of an observed person in an office and an observed data set in the human resources database are supposed to represent the same subject. Only in this case, all assertions about these subject stages are about the same person.

Use cases imply specific *perspectives*. “Each interpretation of any subject is made within a certain perspective. A perspective is a bias in which the universe is seen that determines the expressions that are uttered to describe it.” [2] In the example above two different perspectives on employment and supply chains are applied. As will be discussed below, at consumption time the knowledge about the perspective taken at creation time is very valuable for properly fulfilling integration and interoperability tasks.

Summarised, a modelling method intentionally defines a model type. If the modelling method is disclosed, the semantic gap is closed. But today, the modelling method is usually hidden in the application layer. In that case, the model type is only defined by its extension: the model instances. All information about *use case* and *perspective* is not publicly available at creation time and at consumption time.

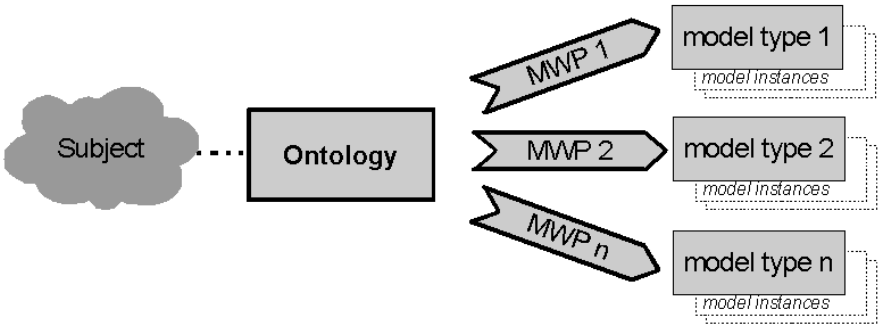

Figure 2: Using MWPs for the intentional definition of model types

As shown in *Figure 2* this paper proposes a method for the generic, workflow-based disclosure of the modelling method to apply: modelling workflow patterns (MWP). Implementations for Topic Maps and OWL ontologies are discussed.

3. Benefits of Disclosing the Modelling Method

This chapter discusses the benefits of disclosing modelling methods. The formally disclosed modelling methods can be used at *creation* and *consumption time* of model instances. In the following the benefits at creation time are described:

Separation of Data and Application Layer. In contrast to the original intension of ontologies, most applications processing ontologies encapsulate and hide the modelling methods for the creation of model instances in the application layer. For example, users or other data sources will be requested for some information about an employee by the applications, which afterwards create instances of the model types governed by the inherent application logic. Disclosing the modelling method opens a path towards a proper separation of data and application layer.

Complexity reduction. As Ontologies are meant to suffice several tasks and user roles they are getting more complex and therefore it is more difficult for a user to understand the whole ontology. It would be more convenient for a user to be guided through the ontology doe to the task that needs to be fulfilled. That can be done by an interpreter applying the disclosed modelling method. The interpreter will support the users to create valid instances of a model types without having knowledge of the whole ontology and the specific model type. *Figure 3* shows a comprehensive conceptual net of an arbitrary domain. For certain *tasks* only parts of this net are of interest.

Therefore it will be of help for the user if only this part is displayed as a view and a description how this part has to be used is given.

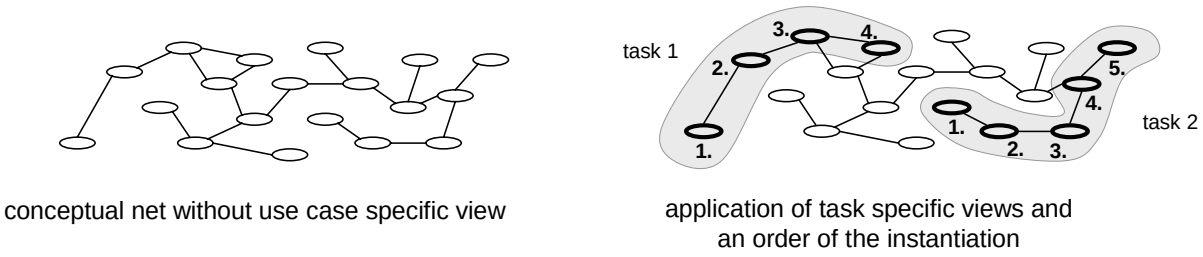

Figure 3 Ontology with visualised use case driven data modelling approach

Autonomous Content. By disclosing the modelling method, the according content becomes autonomous. In a distributed knowledge management scenario, it enables an interpreter to collect information in knowledge networks [9] and describe it autonomously according to the *task* intended by the model type creator. Generic interpreters can provide this functionality to a variety of users in a variety of applications environments. A path towards Smart Content is opened [10].

Summarised, the benefits at creation time are due to a proper separation of data and applications layer and the provision of task specific views on ontologies. At *consumption* time the benefits of disclosing the modelling method are mainly due to the fact, that the perspective taken at consumption time is conserved:

Integration and Interoperability. The profit for interoperability and integration tasks is clearly due to the conservation of the perspective done at creation time. The disclosure allows making traceable decisions about the identity of subjects at consumption time, which is indispensable to solve the challenges of the semantic web.

Maintenance and Reusability. The disclosure of the modelling methods allows a better maintenance and reusability of model types. For instance, changing the model type does not imply changing the application layer. Fewer interdependencies imply risk reduction and cost savings.

Asset Management and Archiving. Because of the autonomy of the model instances, asset management and archiving can be implemented. For instance, if the implementation of an operator which is needed for a specific modelling method should be sorted out of the portfolio, mining the existing model instance portfolio quickly brings conflicts to the light. Model instances become manageable assets.

4. Disclosing Modelling Methods with Modelling Workflow Patterns

There might be at least two general solutions for closing the semantic gap. The first solution is the attempt of more fine-grained specifications of ontologies. As already remarked by Kent [11] in the 1970s, the granularity of the model types, i.e. the decision about to model something as a class or an attribute, is arbitrary and depends on the intended use of the conceptualisation. We assume, that using a more common vocabulary and disclosing the applied modelling method better supports the *usage* of the conceptualisation as the specification of fine-grained, use case specific ontologies.

As discussed in [4], the second solution is the acceptance of interpretation spaces in the vocabulary by coevally disclosing the modelling methods. Our paper proposes a generic, workflow-based

disclosure of the modelling method: the Modelling Workflow Patterns (MWP). It will be demonstrated how MWPs can be used for Topic Maps and OWL ontologies. In the following the *design rationales* of MWPs are described in short detail.

Workflow-based. Disclosing a modelling method implies disclosing a process an author of a model instance has to apply to get a valid and initially intended instance of a model type. This process should be modelled as a workflow. A workflow-based description allows generic interpreters to execute these workflows to guide authors through creation processes. The results of these processes are valid instances of a specific model type. (To our best knowledge there exists no alternative method for modelling a method than describing the workflow to apply.)

Petri Nets. As the rationale above demands a grounding on workflows it will be necessary to use a sound standard for the information model of the workflows. A widely accepted, carefully elaborated and multilaterally used and developed formalism for representing, executing, optimising, and validating workflows are Petri nets [12]. Therefore, MWPs are based on Petri nets.

Self-Containedness. As argued above, the fusion of data and application layer should be avoided. Therefore the description of the modelling method should be part of the data layer. This leads to the need of MWP exchange formats for all used data modelling languages. We decided to create a solution for OWL and Topic Maps, the major players in Semantic Web applications.

Generic Representation. Separating data and application layers implies that the workflow contained in the data layer only indicates and customises the operators which constitute the modelling method. All applications specific issues should be handed over to the application layer. Whether a modelling operator is processed via a web end, a desktop GUI or a sequence of SQL statements is not in interest of the data layer, if the indicated operator is correctly executed according the articulated requirements.

5. Architecture of the MWP infrastructure

Taking the design objectives described above, the architecture depicted in *Figure 4* was designed and implemented to achieve systems consuming Topic Maps and OWL Ontologies containing MWPs.

A model type is defined by an ontology and a MWP. Both are documented using the same syntax, in our example XTM Topic Maps or OWL. The implemented parsers deserialise the MWPs into an instance of the Petri net data model (which is described in detail in 5). The implemented interpreter further processes Petri net data model instances of MWPs independently from the syntax they were originally represented by. According to the tasks demanded in the MWP, operators are executed via a GUI, console, web end or by requesting data bases or other data storages (e.g. a user can be asked to enter some information or a data base can be queried). Depending on the disclosed modelling method and the input gained by executing operators, model instances (in OWL or as XTM Topic Maps) are created by the interpreter.

Using the basic technique the concept of MWPs is applicable for any data model (other than OWL and Topic Maps) by implementing a parser which deserialises such workflow representation onto the given Petri net data model and implementing further plug-ins for coping with the specific type of data representation.

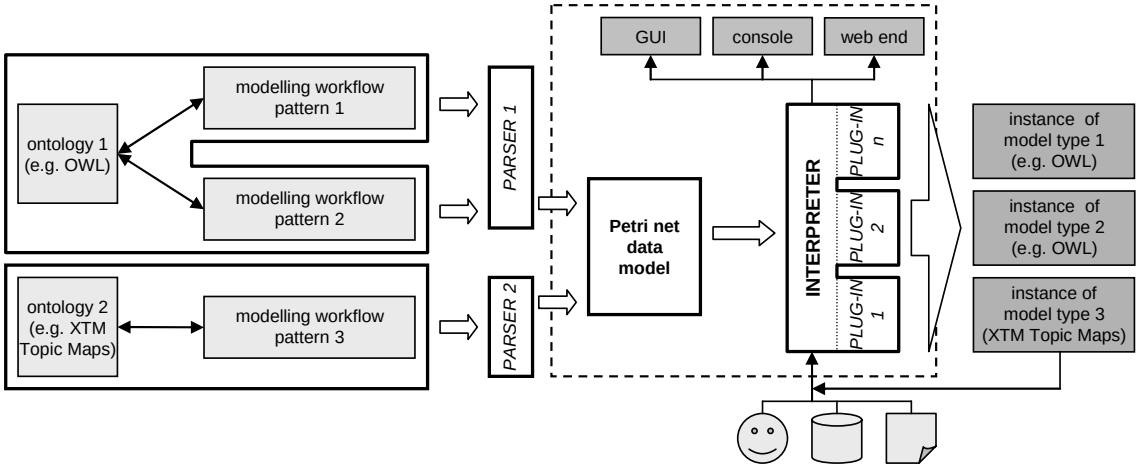

Figure 4: A MWP consuming architecture

To evaluate our approach we implemented all the necessary components to represent and run MWPs within Topic Maps and OWL ontologies using Java.

6. The Petri Net Data Model and Workflow Implementation

To model workflows Petri nets can be used [12]. Basically, a Petri net consists of places and transitions, connected by arcs. The successor of a place is a set of transitions, and the successor of a transition is a set of places. Places host tokens, which flow through the net. Transitions host operators. In the case all incoming places of a transition host at least one token, it is enabled and the hosted operator has to be executed. After the operator's execution the transition fires. By firing, the transition consumes all tokens from the preceding places and supplies one token to all succeeding places.

A *workflow* is a view on a Petri net, defined by a start and an end place. An instantiated workflow is called *case*. There exist several extensions to the basic Petri net (so called high-level Petri nets). Coloured Petri nets (as one possible extension) define that tokens can have specific values so that a transition is only enabled if the values of the tokens of the preceding places and the preconditions of the transition match.

For our approach we developed a Petri net data model based on coloured Petri nets that allows allocating values to the tokens. On top of the data model the processing model defines how an interpreter has to process a data model instance. Furthermore, we developed a Topic Maps and an OWL syntax for representing workflows and we defined the deserialisation of these syntaxes to data model instances. All technical specifications are not in the scope of this paper. These can be found in full detail in [1]. In the following, the basic idea of the Petri net data and processing model is described in short (and simplified) detail.

As shown in *Figure 5*, in the Petri net data model a transition item has the following properties: *[id]*, *[operator]* and *[operand]*. To implement a generic representation as demanded in chapter 3, the value of the property *[operator]* indicates, using an IRI, the operator which has to be applied by the interpreter. The IRI is a Published Subject Identifier (PSI) as introduced by Pepper and Schwab [13]. Operator PSIs refer to published documents which define the operators to apply. An interpreter implementing these operators has to fulfil all requirements defined by these documents

appropriate for its usage context. The values of the properties *[operand]* define the input of the operator. The result of a transition is stored in the fired token.

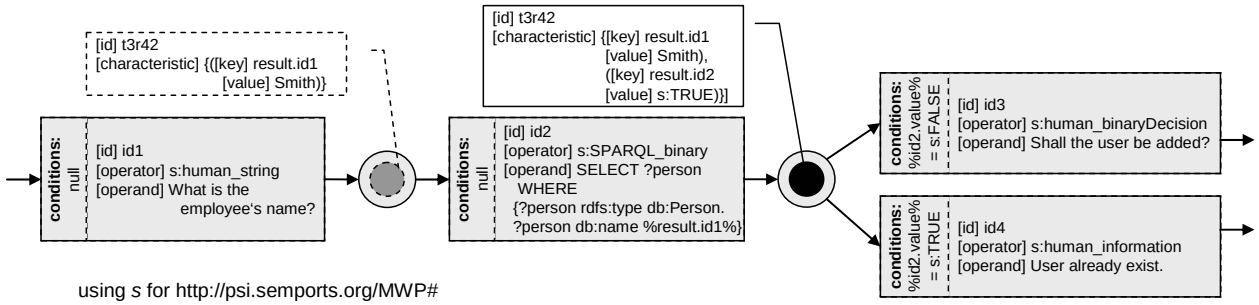

Figure 5 Detail of a Petri net data model instance

In the example depicted in *Figure 5*, the transition with the value id1 of its property *[id]* was enabled. An interpreter had to execute an operator which is indicated by *s:human_string* as operator PSI. As it is specified by the referred information resource, a human had to insert a string due to a given question. The property *[operand]* specified that the question was "What is the employee's name?". It is completely left to the implementation and its context how the question is shown to the user and how the user can express the answer. After the transition *id1* fired the following place contained a token with the value of the result contained in transition *id1*. Using that value transition *id2* starts an OWL-request (due to the operator *s:SPARQL_binary*) and checks whether the name already exists within the available data sources. As a result it fires and the token contains additionally the result of the transition. In our example the name exists and therefore only transition *id3* is enabled (due to the precondition). This transition again executes a task depending on the given *[operator]* and *[operand]*.

The example depicts generic (e.g. *s:human_string*) and data model dependent operators (e.g. *s:SPARQL_binary*). For the Topic Maps implementation, all querying tasks have to be expressed using tolog [14], a predecessor of the upcoming Topic Maps Query Languages. Because tolog does not implement any updating features, new assertions have to be added using parameterised LTM [15] expressions.

7. State of the Art

The first time the concept of MWPs was sketched by the authors in [16] as *TMEP Disclosure* (Disclosing the Process of Topic Maps Engineering).

A schema is a set of constraints a model instance has to fulfil to be valid against this schema. In fact, a schema defines a vocabulary and rules. This is similar to the notion of ontology we use in this paper. We elaborated in detail the shortcomings of ontologies. We assume, that schema definitions are complementary to MWPs. (As will be discussed below, it is part of the future research to assure that a MWP does only create valid instances of a given Schema.)

A slightly equal approach to MWPs is done by the XML Processing Model Working Group. The premise of their work is, that specifications such as XSLT, XML schema, XInclude and XML Canonicalization define transformations that operate on and produce XML documents. But the order in which these transformations are to be applied is not specified anywhere and will in general

yield different result documents [17]. Their solution is the disclosure of the sequence how different operators have to be applied. These disclosures can be used at creation and consumption time.

As discussed above, MWP creates model instances from a use case driven perspective in general by using only a part of the given ontology. This part is comparable with the creation of a view on the ontology. In example, XSLT provides means to create views on any XML data. However, even if complexity reduction might be achieved (see 2), these views do not disclose the modelling method to apply to create valid model instances.

XForms [18], [19], the new generation of web forms, separate the XForms model from the interfaces implementing these models. The XForms model allows the validation of the input data according to a schema definition (at the client side). XForms is an interesting enabler technology to implement input operators of a MWP (bound to some schema definitions) for different usages contexts.

As depicted in *Figure 3* at chapter 3, reducing complexity at creation time is one of our efforts. An approach contrary to MWP is abstraction to yield a higher level conceptualisation based on the given concepts. This provides the user a top down approach, understanding higher level concepts and just decent the necessary branches. However, this approach does not disclose the modelling method to apply and therefore suffers from the well known shortcomings already discussed.

There exist data models and syntaxes for Petri nets [20]. In [1] we defined a Petri net data model as application of the Topic Maps Reference Model (TMRM) [21]. The TMRM is the information model of the semantic integration standard Topic Maps. The advantage of defining the Petri net data model as application of this information model is that the according workflow definitions might easily plunge into the application domain of information integration, if necessary. Well defined merging of process disclosures is another application of a Petri net data model based on the TMRM.

8. Conclusions and Outlook

Our approach of disclosing modelling workflow patterns together with the ontologies helps to reduce the outlined semantic gap. At creation time the disclosed modelling method is used to create valid model instances from the intended perspective. At consumption time this knowledge about use case and taken perspective can be used to support interoperability and integrations tasks. The application of MWPs within Topic Maps and OWL ontologies as well as the described prototypical implementation demonstrates the basic idea. Thus by properly separating the application and the data layer specific aims like complexity reduction, autonomous content, better maintenance and reusability can additionally be achieved.

At least for Topic Maps, there exists no syntax for describing workflows based on a sound workflow data model. The syntax and data model introduced in [1] can be used for the Topic Maps definition of any kind of workflows. The approach is not bound to MWPs. Looking at OWL ontologies several approaches towards workflow description are done (especially within the area of OWL-S [22]) but none of these seems to be mature enough to become a standard nor do they satisfy the requirements of basing on a sound standard like Petri nets. Therefore we developed a syntax for OWL together with a deserialisation to the Petri net data model.

Nevertheless there are a few points still being unsolved. As already mentioned schemas and MWPs are complementary. At the moment we cannot imagine a generic validation mechanism which checks whether a MWP only creates model instances which are valid against the given schema (ontology).

Another point concerns operators to be implemented by interpreters. To enable interpreters to implement a variety of operators, it is necessary to define and publish a comprehensive catalogue of standardised operator descriptions (e.g. “human binary request”). A starting point is given by [1].

At the moment MWP focuses on the creation of model instances. Updating model instances have to be investigated. The same holds for handling changes of the ontologies the model instances base on.

The last point concerns the creation of MWPs. By now the arrangement of places and transitions is quite an uncomfortable task and might generate inconsistencies and failure. It might be comfortable to create a generic MWP for the definition of specific MWPs or to implement plug-ins for ontology editor environments.

Looking back at the problem outlined at the beginning of this paper, our approach is a contribution to make collaborative knowledge management more applicable and therefore brings it forward to mainstream through integration the functionality in desktop production system. The closure of the described semantic gaps satisfies the outlined requirements: ability to integrate distributed information, consistent terms or vocabularies and system independent format. Furthermore we argue that using MWPs for updating and delivering of modelling methods which are integrated in desktop production systems is straightforward.

References

[1] MAICHER, L.; BÖTTCHER, M.: Modelling Workflow Patterns. Data model, processing model and syntaxes for XTM Topic Maps and OWL. Latest version available at: http://www.informatik.uni-leipzig.de/~maicher/mwp/mwp.htm

[2] BIEZUNSKI, M.: A Matter of Perspectives: Talking About Talking About Topic Maps. In: Proceedings of Extreme Markup Languages 2005, Montreal, (2005).

[3] GRIGORIS, A.; VAN HARMELEN, F.: A semantic Web primer. MIT-Press, Cambridge et al, 2004.

[4] SPYNS, P.; MEERSMANN, R.; JARRAR, M.: Data modelling versus Ontology engineering. In: SIGMOD Record, Vol. 31, No. 4, (2002).

[5] MAICHER, L.: Topic Maps and the Absence of Shared Vocabularies. In: Proceedings of TMRA 2005, Leipzig; Springer LNAI 3873, (2006).

[6] W3C: OWL Web Ontology Language. W3C Recommendation. Latest version available at: http://www.w3.org/TR/owl-features/.

[7] ISO/IEC: Topic Maps – Part 2: Data Model. Latest version available at: http://www.isotopicmaps.org/sam

[8] BÖHM, K.; MAICHER, L.: Real-time Generation of Topic Maps from Speech Streams. In: Proceedings of TMRA 2005, Leipzig; Springer LNAI 3873, (2006).

[9] CUEL, R.: A New Methodology for Distributed Knowledge Management Analysis. In: Proceedings of I-KNOW '03, Graz, pp. 531-537, (2003).

[10] BEHRENDT, W.; GESER, G.; MULRENIN, A.; REICH, S.: Dossier on Smart Content as proposed vision for future RTD. Available at: htt://ep2010.salzburgresearch.at

[11] KENT, W.: Data and reality. North-Holland, Amsterdam, New York, (1978).

[12] VAN DER AALST, W. M. P.: The Application of Petri Nets to Workflow Management. In: The Journal of Circuits, Systems and Computers, 8(1):21-66, (1998).

[13] PEPPER, S.; SCHWAB, S.: Curing the Web's Identity Crisis. Subject Indicators for RDF. In Proceedings of: XML Europe 2003.

[14] GARSHOL, L. M.: tolog - a topic maps query language. In: Proceedings of TMRA 2005, Leipzig; Springer LNAI 3873, (2006).

[15] GARSHOL, L. M.: The Linear Topic Map Notation. Available at: http://www.ontopia.net/download/ltm.html (01.03.2006)

[16] SIGEL, A.: Report on the Open Space Sessions. In: Proceedings of TMRA 2005, Leipzig; Springer LNAI 3873, (2006).

[17] WALSH, N.: Charter of the XMP Processing Model Working Group. http://www.w3.org/2005/10/xml-processing-model-wg-charter.htm; Download: 2006-02-28.

[18] BOYER, J.; LANDWEHR, D. et al. (eds.): XForms 1.0 (Second Edition). Latest version available at: http://www.w3.org/TR/xforms/

[19] HUDITSCH, R.: Implementierung eines auf der XForms-Technologie basierenden Editors zur manuellen Erfassung von XTM 1.0 Topic Maps. Diploma thesis at FH Eisenstadt.

[20] KINDER, E (ed.): Proceedings of the Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets, 2004. Available at: http://www.upb.de/cs/kindler/events/XML4PN/xml4pn.pdf

[21] DURUSAU, P.; NEWCOMB, S. R.: Topic Maps—Reference Model, 13250-5. Latest version available at: http://www.isotopicmaps.org/tmrm/

[22] BECO, S.; CANTALUPO, B.; GIAMMARINO, L.; MATSKANIS, N.; SURRIDGE, M.: OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. In: Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05), (2005).