RDFT: A Mapping Meta-Ontology for Business Integration

Borys Omelayenko

Division of Mathematics and Computer Science
Vrije Universiteit, De Boelelaan 1081, 1081hv,
Amsterdam, The Netherlands
borys@cs.vu.nl

Abstract. To create new added value the Semantic Web has to provide new means for business integration enabling open world-wide cooperation between various enterprises. Existing business integration techniques assume the enterprises having similar conceptual models of the objects being exchanged, and this limits their application areas. The Semantic Web needs to possess a technique capable of mapping different conceptual models in the business integration context. We propose a mapping meta-ontology built on top of RDF Schema using previous experiences in modelling mappings, specifics of RDF Schema and business integration tasks.

1 Introduction

Historically business integration has been performed within the Electronic Document Exchange[1] (EDI) paradigm via costly Value-Added Networks (VANs) that use private exchange protocols and provide full range of network services for large companies. Each EDI implementation requires a substantial labor effort to program and maintain, and this makes EDI unacceptable for small and medium enterprises searching for cheap and easy solutions. In this respect the World Wide Web and its successor the Semantic Web provide open standard representation means for modeling business information on the Web and allowing development a new generation of integration solutions.

At present time integrating two companies basically means integrating their database inputs and outputs available as EDI or XML documents. The databases are either accessible directly or via the Internet and with formal specification of possible operations on them are now often referred as Web services.

A number of toolkits have been developed to help the user in performing such kind of integration. The MS BizTalk[2] tool supports the integration of different databases accessible directly via SQL (or via EDI documents and appropriate wrappers). The documents pass through several steps. First, source documents are transformed into XML representation by means of wrappers. Second, the source XML schema is mapped to the target XML schema with the BizTalk Mapper tool that provides a user interface for mapping and generates XSLT scripts able to translate instance XML documents according to the maps. Finally, the resulting database record or EDI document is created from the target XML document.

Similar tasks are performed during the integration of different web services described in Web Service Definition Language[3] WSDL. In each message the service is expected to receive or generate an XML document according to the XML Schema specified in the service description. Capestudio[4] contains an XML mapping tool helping the user to map two XML schemas and automatically generate the correspondent XSLT file, similar to the Biztalk's Mapper. In addition, it provides advanced concerning working with online services and WSDL service descriptions (the same functionality is achieved with wrappers and SQL server in the BizTalk's database integration scenario).

The new generation of tools, e.g. the Unicorn[5] toolkit, uses ontologies as structured vocabularies that help the users in developing the mapping rules. In this case the user maps document's elements to ontological concepts and uses the hierarchy of ontological terms to navigate the elements. These terms are used to provide mediating names for database attributes, and do not constitute a mediating conceptual model. For example, it cannot support the case when in the source database a single object is stored in a single record while in the target database the same object is splitted up into several records.

At their present status the integration tools help in specifying the maps between quite similar conceptual models and require substantial programming effort for aligning different conceptual models. We aim at creation of an architecture that allows mapping different conceptual models via a mediating conceptual model.

The paper is organized as follows. The business integration task is outlined in Section 2 with typical conceptual models that need to be mapped, the mapping meta-ontology for aligning conceptual models represented in RDF Schema is discussed in Section 3, and the mapping tool is shown in Section 5, followed by conclusions and future research directions.

2 The Integration Task

Different companies play different roles in business collaborations and hence develop different (and often partial) models of business objects. Each document is specialized for a certain operation and represents an aggregation of properties of several objects specially grouped to support the operation. There does not exist any single exchanged document that represents a complete model of an object.

The mediator needs to aggregate these partial models to construct the mediating model and perform document integration via the aggregated mediating model rather than directly translate the documents.

[1] www.x12.org
[2] www.biztalk.org
[3] http://www.w3.org/TR/wsdl
[4] http://www.capeclear.com/products/capestudio/
[5] http://www.unicorn.com/

The mediating concepts might have the following features:

- All available knowledge about the objects is coming from input and output messages.
- The model must represent all the objects being exchanged and be sufficient for producing all the necessary documents required for interchange.
- The objects tend to change in time, and these changes are often marked with the timepoints of the documents or message validity times indicated in the messages.
- The model must evolve on-line with new customers coming to the mediator and bringing new views of the concepts they are willing to exchange.
- The documents are sometimes assigned to events in a non-trivial way, and a substantial effort may be needed to link the documents to workflow activities [1].

We treat the enterprise integration task as a service integration tasks. We assume that the enterprises, which we are going to integrate, are represented as Web services specified in WSDL. For each of them WSDL specifies the messages that are expected and/or produced and XML Schemas of the documents being transferred with the messages. These messages are produced by the company's ERP system [2], and the logic behind them is not accessible to the integration service, and even not important for the integration.

Accordingly, the integration service interacts with the world via messages (events) described in WSDL that have XML documents attached, which structure is specified in the XML Schemas included into WSDL descriptions.

2.1 Getting Conceptual Models from XML structures

Each message produced or expected by a company has an XML Schema specifying its structure. XML DTDs and Schemas are traditionally regarded as structural information that possesses no formal semantics, however they clearly preserve a large piece of knowledge about the domain objects they represent. A well-defined XML structure captures most of part-of relations between the domain objects and/or literals.

DTDs, which are much less expressive than XML Schemas, can be easily converted to reasonable conceptual models for the objects being described in the documents. A rule-based approach for extracting conceptual models from DTDs [3] provides a list of heuristic rules of two types: lexical rules that generate a taxonomy based on the inclusion relationship between XML tag names and restructuring rules that transfer XML trees into a conceptual model handling object nesting, cardinality constraints, etc.

Let us consider a DTD sample

```
<!ELEMENT A (B,C,(D|E))>
```

where all the elements `B`, `C`, `D`, and `E` are defined as literals (`#PCDATA`) and show how it may be converted to a conceptual model with the a slightly modified version of rules from [3].

Expression `D|E` is processed with rule `C(Choice)` that creates a new element `cD_E`. Elements `D` and `E` are converted to lexical concepts `cD` and `cE` that are connected to `cD_E` with relations `D` and `E` depicted in Figure 1. A composite element `A` is converted to concept `cA` (rule `CE-Composite Element`). Each of the elements `B` and `C` is converted to a literal concept `cB` or `cC` respectively (rule `SCE-Simple Component element`) and relations `B`, `C`, and `D_E` are added. More complicated examples incur other rules concerning repeating elements, cardinality, and complicated composite DTD elements.

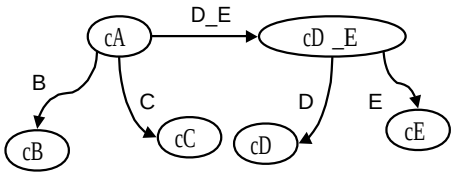

Fig. 1. A fragment of a conceptual model constructed from DTD

A similar algorithm has been proposed for converting XML DTDs to relational schemas and back [4]. Such algorithms take over all the routine tasks of converting DTDs to conceptual models. As a result the user can concentrate at extracting domain objects described in the documents and aligning them to the mediating or upper-level ontologies.

Finally, these conversion techniques ease the step of shifting the focus of the integration task from document interchange to concept exchange discussed in this paper.

2.2 RDF Modelling

There exist a number of means for representing conceptual models: ER and UML (conceptual modeling) F-logic, RDB, etc. (databases), DAML+OIL, description logic, etc. (knowledge engineering). Open business integration techniques on the Semantic Web might use Web standards for representing conceptual models on the Web, and currently RDF and RDF Schema [5] seem to be the best candidates for such a standard. There exist several extensions to RDF Schema, e.g. DAML+OIL[6] proposal that has more expressive power than RDF Schema itself. However, we naturally expect core languages to be adopted wider and faster than the extensions and hence we first concentrate on the use of RDF Schema still keeping in mind these extensions.

RDF (and its schema language RDF Schema) was primarily targeted at annotation of existing documents with conceptual models. The models are targeted at capturing some *partial* knowledge and we face a number of problems while representing *complete* models of events and documents. This leads to the following drawbacks:

- Properties are defined in RDF as first-class objects and they exist independently from classes. Such an interpretation is a bit unhandy for our needs. For example, the property `Address` must be mapped in different ways depending on whether it is attached to class `Invoice`, where it stands for billing address, or to class `DeliveryOrder`, where it stands for delivery address. RDF provides the means to distinguish between different object-property-value triples on the level of instance documents, where each property is assigned to a certain

[6] http://www.w3.org/TR/daml+oil-reference

class. However, property-class assignments are indistinguishable at the level of RDF Schema. DAML+OIL suffers the same problem.
- RDF Schema uses the `rdfs:domain` term which specifies the classes to which the property can be attached. Multiple occurrence of `rdfs:domain` has conjunctive semantics, that is if property `Address` can be used with two classes `Invoice` and `DeliveryOrder` then listing two `rdfs:domain` classes in the definition of `Address` is a wrong way to go. Such statement means that it can be used with a class, a subclass of both `Invoice` and `DeliveryOrder`, and such a class will most like have no sense from the domain point of view. One can model the fact that property `Address` can be used with both `Invoice` and `DeliveryOrder` is to create an artificial superclass for both `Address` and `DeliveryOrder` that has `Address` attached, but is not really informative from the domain point of view. This problem shows up in any attempt to build a mapping ontology where the same property of the ontology, e.g. `SourceClass`, needs to occur many times pointing to different source classes.
- RDF Schema provides an easy way to represent reification: the basic `rdf:Resource` class (the `Thing`) is an instance of and a subclass of itself. This creates certain difficulties in meta-modelling and tool development. In the integration tasks we need to model at three different layers: instance data, conceptual models, and meta-models (e.g. mapping meta-ontologies). These three layers must be tighly integrated and still very well distinguished. Hence, the introduction of three different `Thing`s may be suitable: instance-level resource, schema-level resource, and a meta-resource.
- Two types of things are mixed in common understanding of RDF: universal resource locators URL's that point to a file on the web and should be treated as filenames and universal resource identifiers URI's that look the same as URL's but represent logical names of the things in RDF Schema. Sometimes RDF Schemas stored in separate files need to be treated as logical units and statements about these files need to be created and processed.

We tried to overcome these problems in our model described in Section 3.

2.3 Things to be Mapped: Events, Documents, and Vocabularies

To be able to reason about the inputs and outputs of the companies being integrated we developed a conceptual model of WSDL (the part of WSDL specification that is important for the context of our mapping work skipping protocols and bindings). The model is directly derived from the WSDL specification and provides an RDF Schema for RDF annotations for WSDL documents.

Specifically, WSDL defines the following basic elements of services:

- `Types` that provide links to XML Schemas of the messages exchanged;
- Abstract definitions of `Messages` in accordance with `Types`;
- `Port Types` that specify input and output messages;
- `Bindings` that specify concrete protocols and formats for messages according to `Port Types`.

These elements allow describing the services but do not represent any temporal knowledge about the messages needed for the integration.

The Process Specification Language[7] PSL temporal ontology includes the classes: `activity`, `activity occurrence`, `timepoint`, and `objects`. Activities are performed during certain time intervals marked with timepoints, and each activity occurrence specifies a snapshot of an activity at a moment of time. The objects are defined as things that are not timepoints, not activities and not activity occurrences, and do not possess temporal properties. They may participate in activities at certain timepoints as defined by the `activity-timepoint-object` relation[8]. PSL provides basic temporal semantics of time intervals and timepoints, constraints on objects participating at certain activities, etc.

Accordingly, we extended our WSDL model with PSL ontology. The class diagram of the composite ontology is presented in Figure 2. In contains two root concepts: `mediator:Thing` and `psl:Thing`, subclasses of the former correspond to WSDL concepts and subclasses of the latter correspond to PSL classes. These classes are linked with a number of properties depicted in Figure 3. In both figures classes are represented with circles properties are shown with labeled edges linking the classes.

We must note that the Web Service Flow Language[9] WSFL built on top of WSDL provides the means to specify temporal and workflow information for services, and in some future it will make some of this model redundant. However, WSFL has a longer way to go to become a world-wide standard than WSDL or PSL. And again, in this work we focus at kernel technologies that has big chances of being widely accepted and used rather than at extensions of those.

WSDL annotations made according to our ontology allow performing inference over WSDL descriptions to validate the links established between the enterprises. Also they represent a bit stronger formalization of the services, e.g. by specifying legal ordering of the events. A sample of a realistic sequence of events with their order and timeout values is presented in Figure 4.

3 Mapping Meta-Ontology

The Common Warehouse Model (CWM) Specification [6] by the Object Management Group[10] provides a general architecture for a mapping ontology to be adopted by each specific mapping application. We adopt it to mapping RDF Schemas and specific concepts like events, messages, vocabularies, and XML-specific parts of conceptual models that

[7] http://www.mel.nist.gov/psl/

[8] RDF Schema is bounded to binary relations and provides no means to specify the ternary `activity-timepoint-object` relation. To model it we had to introduce a special class `psl:activity_at_timepoint_relation` attached to `psl:object` and linked to `psl:activity` and `psl:timepoint`.

[9] http://xml.coverpages.org/wsfl.html

[10] http://www.omg.org/

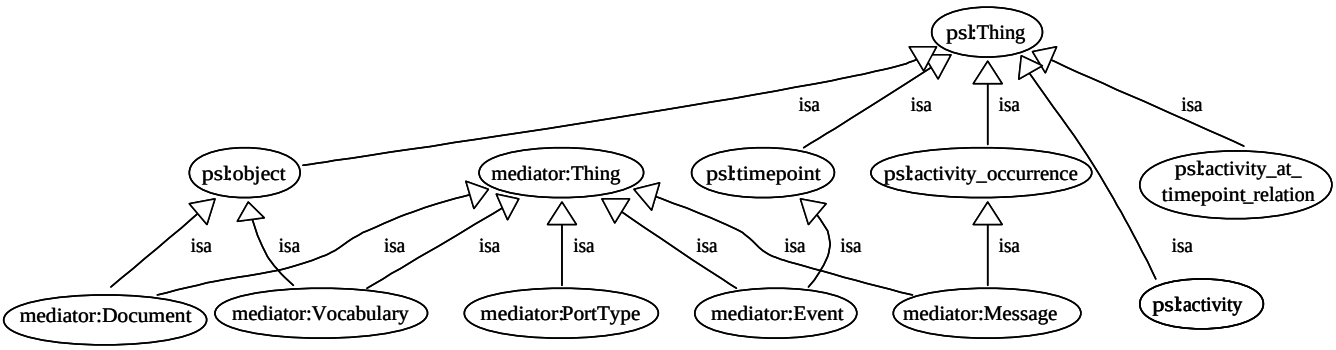

Fig. 2. WSDL extension (class diagram)

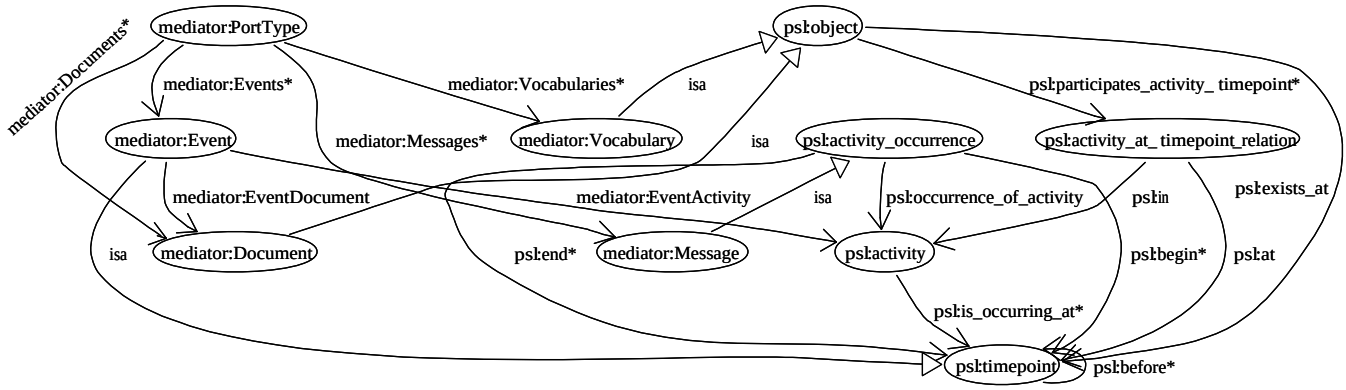

Fig. 3. WSDL extension (descriptions of classes)

```
cls kb:cXMLEvents
  cls kb:cXML_init
  cls kb:cXMLOrderMessage
    pro mediator:TimeOut
      val 60 min
    pro psl:before
      cls kb:cXMLSetupResponse
        pro mediator:TimeOut
          val 5 min
        pro psl:before
          cls kb:cXMLSetupRequest
            pro mediator:TimeOut
              val 25200 min
            pro psl:before
              cls kb:cXMLProductCatalog
  cls kb:cXMLProductCatalog
  cls kb:cXMLSetupRequest
  cls kb:cXMLSetupResponse
```

Fig. 4. Event sequence

occur in the business integration tasks. CWM specifies the following basic primitives for mapping:

- Class `TransformationMap` a subclass of the `Transformation` class. The former is a container for `ClassifierMaps`.
- Each `ClassifierMap` links two groups of classes with two properties source and target with multiple occurrence referring to the groups of source and target classes being mapped with the map.
- Each `ClassifierMap` connects two groups of properties with the `featureMap` property that links it to the `FeatureMap` class connecting `source` and `target` properties (called features). `FeatureMap` also includes a link to another `ClassifierMap` mapping property values.
- Each `ClassifierMap` may connect a group of properties and a group of classes with the `ClassifierFeatureMap` class referenced via the `cfMapproperty`.

The CWM model concerns mapping generic conceptual models and seems to be too expressive for our needs. Mapping classes that normally refer to physical objects to properties that refer to properties of those is a large source of misunderstanding in the business tasks, while conceptually there is nothing wrong with it. In RDF meta-model both properties and classes are treated in a uniform way, while at the level of models those are clearly differentiated. Accordingly, we need to skip it in our mapping framework.

The RDFT (RDF Transformation) mapping meta-ontology[11] specifies a small language for mapping XML DTDs to/and RDF Schemas specially targeted for business integration tasks. The class diagram for the basic top-level classes is presented in Figure 5 using the same notation as our previous drawings.

[11] http://www.cs.vu.nl/ borys/RDFT

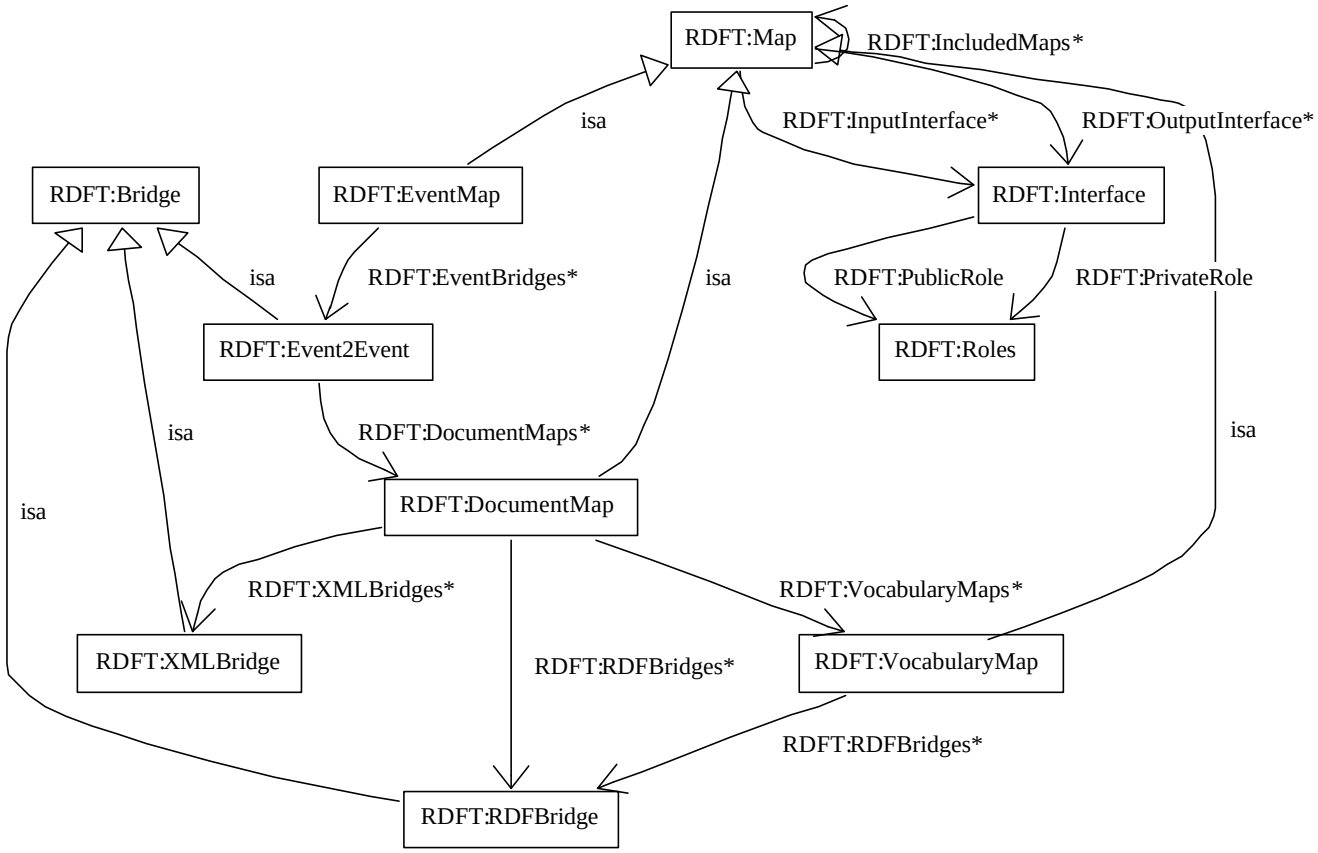

Fig. 5. Main RDFT classes: Bridge, Map, Interface, BridgeRelation, and Roles

The basic RDFT class is `Bridge` that connects two concepts (similar to the CWM's `ClassifierMap`). `Bridge` describes common properties of bridges allowing only one-to-many and many-to-one bridges opposite to CWM allowing many-to-many mappings [12].

The bridges also contain the `Relation` property pointing to one of the relations subclassed from the `BridgeRelation` class: `EquivalenceRelation` or `VersionRelation`.

- `Equivalence` bridges specify that the source element of a one-to-many bridge is equivalent to the target set of elements, and the source set of elements is equivalent to the target element for many-to-one bridges. E.g. a one-to-many bridge connecting a source class to a group of target classes states that the target group is semantically equivalent with respect to the instance data transformation task to the source element.
- `Version` bridges specify that the target set of elements form a (later) version of the source set of elements. Opposite to equivalence bridges, they assume that both source and target concepts belong to the same domain (or document standard), and may refer to two concepts with the same name (but different namespaces indicating versions), and imply that all the relations that held for the original concept must hold for the versioned concept, if the opposite is not stated explicitly.

Several types of `Bridges` are defined in RDFT:

- `Event2Event` bridges link different events, specify temporal event generation conditions, and link the events to the messages transmitted with them. They connect instances of the meta-class `mediator:Event`.
- Two kinds of `RDFbridges`: `Class2Class` and `Property2Property` bridges between RDF Schema classes and properties. `Class2Class` bridges contain `SourceClass` and `TargetClass` properties pointing to `rdfs:Class` instances. `Property2Property` bridges contains `SourceProperty` and `TargetProperty` properties pointing to `rdf:Property` instances. Again, only one-to-many and many-to-one bridges are allowed in RDFT. In RDF Schema properties are defined as first-class objects together with classes, and they capture most of domain knowledge. Classes mostly specify aggregation of properties, and thus we do not include class-to-property and property-to-class bridges in RDFT.
- Four kinds of `XMLBridges`: `Tag2Class` and `Tag2Property` bridges link source XML DTD tags and target RDF Schema classes and properties. `Class2Tag` and `Property2Tag` bridges connect RDF Schema classes and properties, and the elements of the target DTD. These are constructed from `SourceTag` and `TargetTag` properties in addition to

[12] Largely the objective of CWM is to model, i.e. to understand the things, while the objective of RDFT is to be able to transform instance data according to the mappings that imposes certain architectural restrictions.

the `Source/Target-Class/Property` properties mentioned above.

In some cases it is possible to declaratively specify the correspondence of property or class values linked by a bridge (e.g. by specifying the set of `Class2Class` bridges). If it is not feasible then we use the `Expression` property of a `Bridge`. It specifies an XPath [7] expression transforming instance data. XPath defines the means for two tasks: addressing data elements in XML documents and performing element or attribute value transformations (Chapter 4 of the specification). We use only the second part of the XPath functions (e.g. `substring_before`).

We need to create an ontology of a DTD to represent different DTDs in our framework and performing inference involving DTD elements and attributes (e.g. to check whether all the elements are mapped with RDFT bridges). The DTDs themselves are available to the mediating service, and only DTD elements and attributes must be annotated. Accordingly, we introduce two classes to represent them: `XMLElement` and `XMLAttribute`, subclasses of `XMLTag`. Properties `SourceTag` and `TargetTag` take `XMLTag` instances as their values.

Several assignments of a property to a class (e.g. a property with multiple cardinality) are not distinguishable in RDF Schema because they are not that significant from the modeling point of view. However, they are crucially important from the instance data transformation perspective, and we introduce class `Role` to specify each property-class assignment.

The bridges are grouped into maps. Each `Map` is a collection of bridges serving a single purpose. The maps are identified by their names[13] and form minimal reusable modules of RDFT bridges. Each `Map` can include other maps (the `IncludedMaps` property) and serves as a container for `Bridge`s (the `Bridges` property). The maps use some class and property names within the bridges inside the maps and may be reused with other names passed as formal parameters. These parameters are formalized with the `Interface` class depicted in Figure 5. Each `Interface` class contains two properties: `PublicRole` and `PrivateRole` that specify the correspondence between external and internal names, correspondingly.

Connecting two services means connecting their events (instantiating the `EventMap` depicted in Figure 5) consisting of `Event2Event` bridges. Each of the bridges points to a `DocumentMap` aligning the documents attached to events, and, in turn, consisting of `XMLBridges`, `RDFBridges` and `VocabularyMaps`.

We do not impose any restriction on class names of a user ontology conforming the RDFT meta-ontology. Instead, all of them are marked as instances of RDFT meta-classes, as it is illustrated with a `Class2Class` bridge in Figure 8. In the figure user's classes are marked as `S0`, `S1`, and `T0`, and a bridge connecting them is marked with `B_01`.

It is important to differentiate the role of RDFT as a meta-ontology versus a template. Assume a definition of a metaclass `mC` and a definition of property `mP` with domain `mC` and range `rdfs:Literal`. Class `iC`, an RDF instance of `mC` will be a class definition that in addition to RDF Schema standard properties for a class definition (e.g. `rdfs:subClassOf`) possesses instantiated literal property `iC`.

However, we would expect *property definition* of `iC` to be applicable to `mC` rather than the *property itself*. So, we would rather need to call our RDFT ontology as a template ontology, while the term 'template' is not specified within RDF Schema. The instantiation semantics of RDF Schema class definitions, opposite to properties, seems to be acceptable for our needs.

Another important notion related to RDFT is its completeness, i.e. possibility to map arbitrary RDF Schemas. The term 'map' can be defined in different ways and each definition assumes a set of relations that need to be represented with the maps. In our case we try to map different part-of decompositions of object's properties, and one-to-many and many-to-one bridges seem to be sufficient for that. For extracting parts of properties we use quite expressive XPath language.

The main contribution of any architectural solution is to extract several most important tasks in the area and provide the means restricted in expressiveness but explicitly supporting these tasks. The unsupported cases may be handled by programming in expressive languages. We follow this principle in our mapping framework.

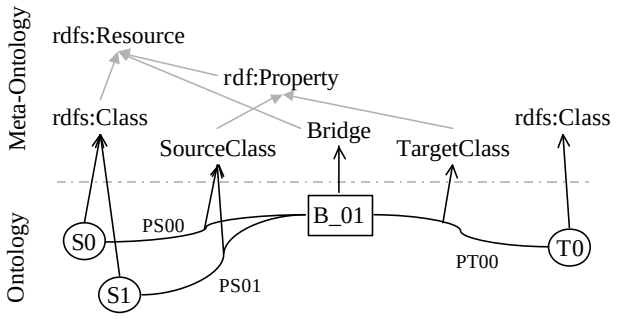

Fig. 8. The use of RDFT meta-ontology

4 Using Expressive Extensions of RDF Schema

It is always possible to express RDFT with expressive extensions of RDF Schema. However, before using them we should always ensure that this does not make things worse. People using DAML+OIL use specific DAML+OIL extensions quite seldom, and mostly they pick some minor but convenient extensions ignoring really expressive language constructs [8].

The business integration scenario is more specific and restricted that a generic modeling scenario. Business objects are quite simple and well-defined. Unlike the general case, most or all the objects might have physical representation, they are explicitly named, and are a part of a very shallow taxonomy.

[13] In this case we have a collision of resource identifiers URI's that's are used in RDF Schema and represent a logical name without any associated physical object and resource locators URL's that point to files somewhere on the Web. The maps are identified by their URI's but should be accessible via their URL's as reusable files with bridges.

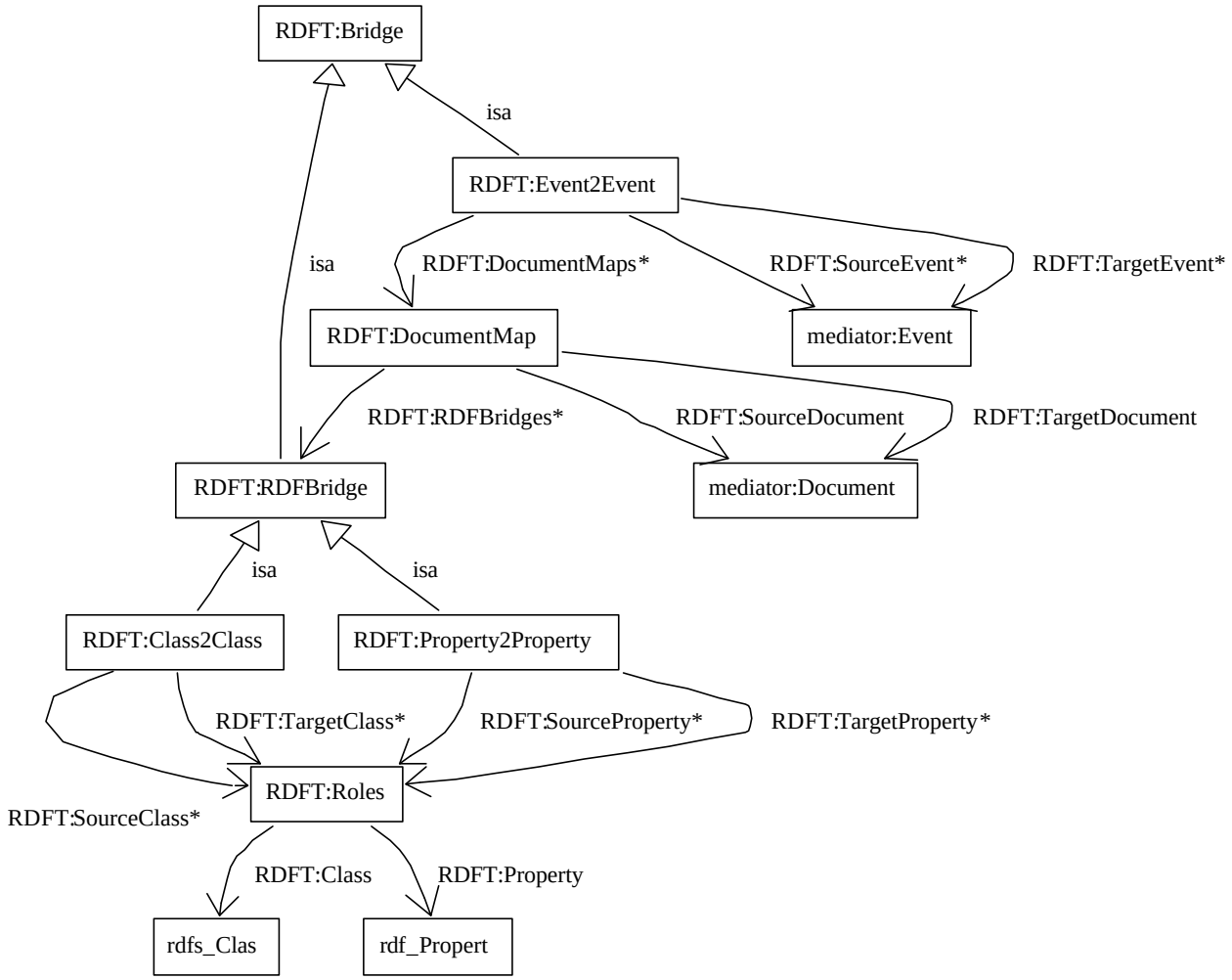

Fig. 6. Bridges and Roles

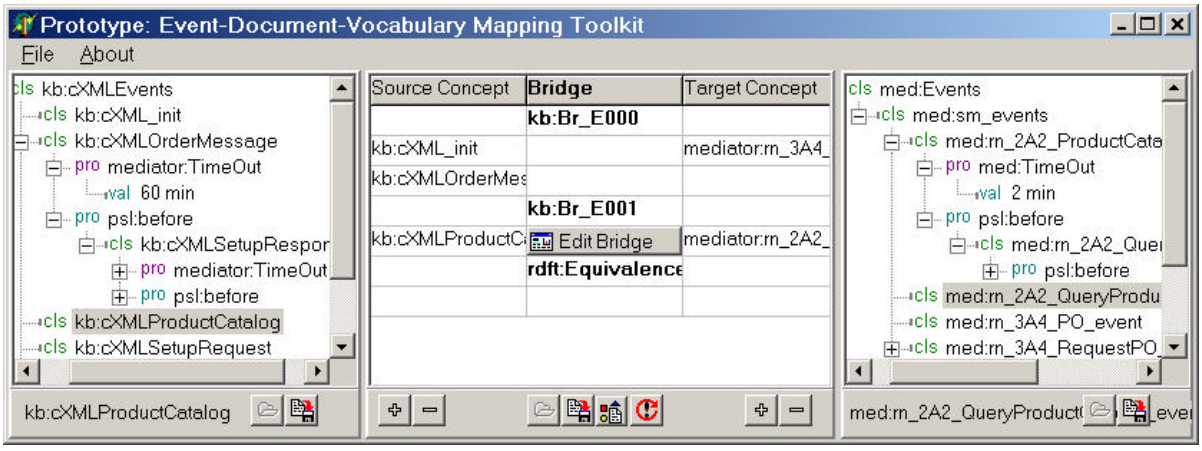

Fig. 7. RDFT tool support: mapping two event sequences

Accordingly, we do not see any immediate needs for using more expressive language than RDF Schema.

The intention of the mapping is to be able parse the maps and generate XML transformation scripts able to translate instance XML documents attached to the messages. That is, for each target DTD element we need to trace which property of which object corresponds to this element, and then trace whether there is a map to a document from the source models. Finally, we need to find out which element from the source DTDs contains the original description of the property.

We need to use search-based inference engines, e.g. CLIPS[14], to find these chains. Then, each chain needs to be translated into an XML transformation language, e.g. XSLT.

5 Tool Support

We are currently developing a prototype tool providing an advanced map browsing and editing facilities and guiding the user through the event, document and vocabulary integration tasks. All these tasks are quite similar from the implementation point of view: all of them can be treated as RDF Schema mapping tasks. In the case of event mapping (see Figure 7 for a screenshot) the prototype tool allows the user to browse two RDF Schemas representing two event sequences and create RDFT maps between the schemas. Each bridge can be edited in details and necessary vocabulary bridges are edited with the same RDFT map editor applied to document schemas instead of event sequences.

Several important features are still under development in the tool: an interface to online web service described in WSDL, an inference engine, and an RDFT execution module. In addition, there are a number of small things that are too annoying to ignore them and too unimportant to talk about them.

6 Conclusions

We proposed a service integration architecture to match the following requirements:

- Allow switching from document transformation and exchange between the services to concept and model exchange.
- Annotate WSDL descriptions with concepts from temporal PSL ontology and some concepts from the business integration domain.
- Contain a mapping meta-ontology specifying mapping information between DTD elements, RDF Schema classes and properties, and events.
- Allow performing inference-based checks for completeness and consistency of the mappings.
- Allow compiling the data transformation chains represented by the mappings to a low-level XML transformation language.

In the paper we present our current progress in satisfying these requirements, namely switching between document-based to concept-based integration, WSDL annotation, mapping meta-ontology and preliminary steps in inference-based validation and compilation.

[14] http://www.ghg.net/clips/CLIPS.html

Acknowledgements. The author would like to thank Dieter Fensel, Christoph Bussler, Michel Klein, and Volodymyr Zykov for their helpful discussions and the reviewers for their comments.

References

1. Bae, H., Kim, Y.: A Document-Process Association Model for Workflow Management. Computers in Industry **47** (2002) 139–154
2. Bussler, C.: Modeling and Executing Semantic B2B Integration. In: Proceedings of the 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce / E-Business Systems (RIDE-2EC'2002) (In conjunction with ICDE-2002), San Jose, USA, IEEE CS Press (2002)
3. Mello, R., Heuser, C.: A Rule-Based Conversion of a DTD to a Conceptual Schema. In Kunii, H., Jojodia, S., Solvberg, A., eds.: Conceptual Modeling - ER'2001. Number 2224 in LNCS, Yokohama, Japan, Springer (2001) 133–148
4. Lee, D., Chu, W.: CPI: Constraint-Preserving Inlining algorithm for mapping XML DTD to relational schema. Data and Knowledge Engineering **39** (2001) 3–25
5. Brickley, D., Guha, R.: Resource Description Framework (RDF) Schema Specification 1.0. Technical report, W3C Candidate Recommendation, March 27 (2000)
6. CWM: Common Warehouse Model Specification. Technical report, Object Management Group (2001)
7. Clark, J.: XSL Transformations (XSLT). Technical report, W3C Recommendation, November 16 (1999)
8. van Harmelen, F.: The Complexity of the Web Ontology Language. IEEE Intelligent Systems **17** (2002) 71–73