Facing Document-Provider Heterogeneity in Knowledge Portals*

Jon Iturrioz, Oscar Díaz, and Sergio F. Anzuola

The EKIN group
Department of Languages and Computer Systems
University of the Basque Country
{jipitsaj,jipdiago}@si.ehu.es, jibfeans@sc.ehu.es

Abstract. Knowledge portals aim at facilitating the location, sharing and dissemination of information by sitting ontologies at the core of the system. For heterogeneous environments where content-providers are free to deliver the content in any format, mechanisms are required that extract and lift these content sources onto a common ontology model. This paper focuses on document providers where diversity stems from either the metadata vocabulary or the metadata location mechanism used. The ontology repository should be isolated from this heterogeneity. To this end, a rule-based approach is presented where rules encapsulate the specificities of each provider. The paper presents a working system where *JENA, WebDAV,* and *QuickRules* realise the knowledge portal, the resource repository and the rule component, respectively. Rules are given for *PDF*, *WORD* and *OpenOffice* resources.

1 Introduction

Knowledge portals *"exploit an ontology for achieving a conceptual foundation for all their functionalities, i.e. information integration as well as information selection and presentation, are glued together by a conceptual model"* [9]. In the same way that a database is characterised by its schema, a knowledge portal is distinguished by its ontology. While traditional databases take data structures out of the applications, a knowledge portal extracts knowledge out of the applications that hereafter, rely on the portal for knowledge obtention (i.e. inference capabilities are externalised to the portal).

An ontology (i.e. *"an agreement about a shared, formal, explicit and partial account of a conceptualisation"* [21]) is "populated" by metadata that describe the "portal resources" (e.g. HTML pages, documents, images, etc) that stand for instances of the ontology. The resource providers can be numerous, heterogeneous and evolve as time goes by. Document processors are a case in point. The new crop of document processors (e.g. editors, workflow, content managers) are metadata-aware. Each proc-

* This work was partially supported by the Spanish Science and Technology Ministry (MCYT) and European Social Funds (FEDER) under contract TIC2002-01442. It also benefits of funding from la "Consejería de Ciencia y Tecnología" de la Junta de la Comunidad de Castilla La Mancha (PCB-02-001).

A. Persson and J. Stirna (Eds.): CAiSE 2004, LNCS 3084, pp. 384–397, 2004.

essor takes the document as input, processes it, and potentially, attaches, either manually or automatically, some metadata.

The issue is how to keep consistent the document collection with its ontology model counterpart. A mechanism is needed that punctually reflects operations on the document to its ontology counterpart so that portal users can obtain an ontologically valid view of this resource. And this extractive operation is processor-dependant since, for instance, a Word document and a PDF document have different mechanisms to support metadata.

Specifically, document-processors difer on both the metadata vocabulary and the metadata location. As for the vocabularies, the PDF world is promoting the *Extensible Metadata Platform* (XMP) [5]. XMP metadata defines a core set of metadata properties that are relevant for a wide range of applications including all of Adobe's authoring and publishing products. On the other hand, *OpenOffice* [3] has a different set of metadata which also overlaps (but does not totally coincides) with *Dublin Core* [2]. Hence, the first issue is to harmonize the vocabularies coming from different document processors.

It should be note that most previous works focus on how to characterize Web resources (e.g. an HTML page) along a given ontology, i.e. how to obtain the ontology description after the content/layout of the HTML page [17,8,16,11]. By contrast, this work assumes that such a characterisation is somehow attached to the resource, and the issue is how to cope with the diversity of the formats used for document storage.

As for metadata location, it refers to how metadata is attached to the subject document (i.e. the resource). Currently, there are two alternatives. One option is to keep the metadata embedded in the document itself, and hence, metadata appears as part of the resource data stream (e.g. this is the PDF option using XMP [5]). A special API is then used to extract this metadata. This has the advantage that the metadata stays with the application data file, even if the file is moved. The downside is that the API is dependent on the editor you use to generate the data file. By contrast, other option is to store the metadata in a separate file which is associated by convention with the source document. *OpenOffice* is a case in point. This works universally, but has the disadvantage that the metadata can be lost in a processing step if the metadata file is not kept together with the data file.

Both metadata vocabularies and metadata location, are two source of variations whose variants are far from being settle down, i.e. ontology standards, processors and resource formats are all evolving rapidly.

To isolate the ontology layer for the heterogeneity and evolvable nature of content providers, this paper presents a rule-based propagation mechanism. The rule paradigm has proved to facilitate the building of systems that are easy to adapt to evolving requirements. Their self-contained, isolated and decoupled nature, allows rules to be easily enlarged, removed or updated. The use of this reactive architecture accounts for the separate evolution of each layer (i.e. the ontology, and the content providers that populate the ontology) as well as the smooth inclusion of new document processors as time goes by.

As a proof of concepts, this paper presents an implementation. The knowledge portal[1], the resource repository, and the rule manager are three independent components.

[1] For the aim of this paper, the knowledge portal is restricted to a repository for RDF expressions.

Which have been realised through *JENA* [12], *WebDAV* [1], and *QuickRules* [20] were used to support the knowledge portal, the resource repository and the rule component, respectively.

The rest of the paper is organized as follows. Section 2 outlines the architecture. Section 3, 4 and 5 presents the *ontology* component, the *resourceCollection* component and the *ruleManager* component, respectively. Rules are presented in section 6. Finally, conclusions are given.

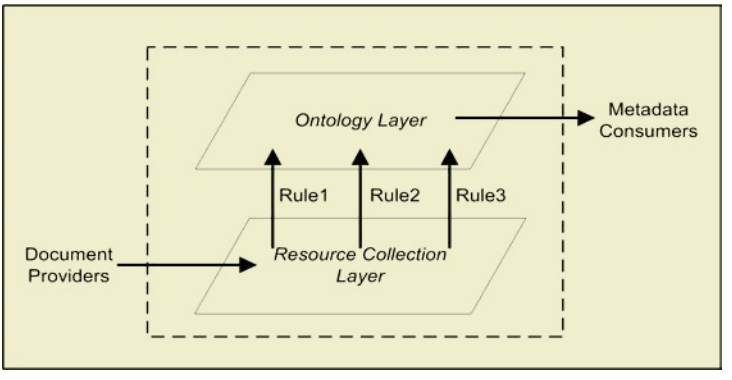

Fig. 1. A two-layers architecture.

2 The Architecture

The DOGMA initiative [18] is promoting a formal ontology engineering framework that generally consists of a three-layer architecture comprising (i) the heterogeneous data sources, (ii) the wrappers that lift these data sources onto a common ontology model, (iii) the integration modules (mediators in the dynamic case) that reconcile the varying semantics of the different data sources, and (iv) the consumers of the ontology.

This architecture is also basically followed in this work where data sources are restricted to document providers, the ontology model is RDF, and where wrapping and mediator concerns are realised as rules (see figure 1).

The idea is to sit an ontology layer on top of the collection of documents. The structure of the ontology (and, in the future, the inference power of the ontology formalism) facilitates a more dynamic and powerful location and navigation of the underlying document collection. This enables automated processing of the resources regardless of both its origin (i.e. the editor used to create it) or its use (i.e. whether they are used for presentation matters by a content manager, or for defining the document flow in a Lotus-like application). The main endeavor of this work is then to keep the *RDF* model consistent with the underlying document repository.

To this end, an architecture is introduced with three components namely, the *ontology* component, which is realised using the JENA API [12], the *resourceCollection* component, which is supported by *WebDAV* [1], and the *ruleManager* component, which is implemented using *QuickRules* [20], respectively. The dependencies between these components is depicted in figure 2. Next sections present the concrete realisation.

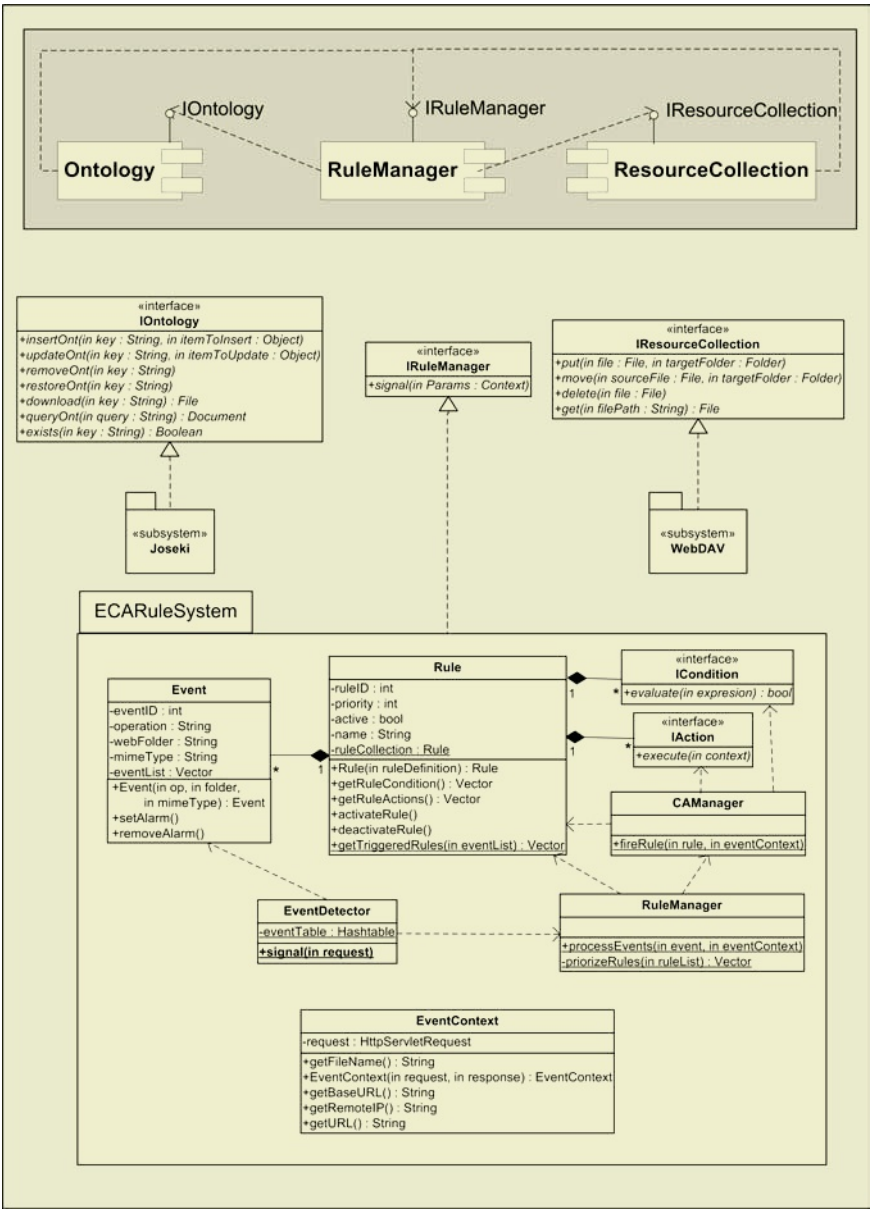

Fig. 2. The components.

3 The *Ontology* Component

The World Wide Web has introduced the Resource Description Framework (RDF) [7], in an attempt to produce a standard language for *machine-understandable* descriptions of resources on the Web.

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dce="http://www.atarix.org/dcExtension#">
    <rdf:Description rdf:about="http://www.vldb.org/03mdde/03mdde.sxw">
        <dc:title>Facing content-provider heterogeneity in Knowledge Portals</dc:title>
        <dc:description>
            Ontology languages are being proposed to provide machine-understandable
            descriptions of resources that permit easy location of these resource.
        </dc:description>
        <dce:creator>Sergio Fernández</dce:creator>
        <dc:language>en-US</dc:language>
        <dc:date>2003-05-21T17:41:00</dc:date>
        <dc:subject>Ontology</dc:subject>
        <dc:subject>Document Engineering</dc:subject>
        <dc:creator>Jon Iturrioz</dc:creator>
        <dc:creator>Oscar Díaz</dc:creator>
        <dc:creator>Sergio Fernández</dc:creator>
        <dce:table-count>0</dce:table-count>
        <dce:image-count>3</dce:image-count>
        <dce:object-count>0</dce:object-count>
        <dce:page-count>10</dce:page-count>
        <dce:paragraph-count>304</dce:paragraph-count>
        <dce:word-count>1905</dce:word-count>
        <dce:character-count>14613</dce:character-count>
        <dc:format>texto/sxw</dc:format>
        <dce:submissionDate>2003-06-10T00:00:00</dce:submissionDate>
        <dce:workshop>CAISE'04</dce:workshop>
        <dce:retrievedBy>158.227.111.111</dce:retrievedBy>
        <dce:consultedBy>158.227.111.112</dce:consultedBy>
    </rdf:Description>
</rdf:RDF>
```

Fig. 3. *RDF* representation of the resource *http://www.atarix.org/04Caise.sxw*.

To describe a resource in terms of RDF, three things are required: a data model, a schema, and the statement itself using XML syntax. The RDF data model is expressed using directed labeled graphs which identify the properties and property values that qualify the resource. Next, a schema needs to be defined or existing schemas must be identified. Schemas provide the RDF type system. The properties could be expressed using appropriate existing vocabularies (e.g. Dublin Core) that can be complemented with properties specific to the application at hand. Finally, a specific resource can be described along the lines of the schemas. Here, *rdf/xml* is used for this purpose.

This paper considers a very simple ontology with a single class. This class describe resources along distinct properties (see figure 3). Properties can be:

- attributive, if they kept basic properties of the resource. To this end, the *Dublin Core* vocabulary is used. In figure 3, these attributes come from 'dc' namespace.
- relational, if they provide links to other resources. Here, relational properties are not extracted from the document but obtained through inference rules. For instance, the property <*relatedTo*> can be inferred if two documents share at least an author. This problem is not addressed here.
- operational, if they are related with operations the document has undergone, and therefore, no related with the content of the document. For instance, *OpenOffice* provides the *"editing-cycles"* metadata that specifies the number of editing cycles the document has been through while in the realm of *OpenOffice*. As you move away from editing to other kind of processing, the corresponding software can en-

rich the document with additional operational metadata. For instance, if the *"submission-date"* needs to be kept, this data can not be recorded by the document editor but registered by the repository manager when the document enters the repository. In figure 3, the "dce" namespace is used for this extended vocabulary.

RDF descriptions are kept in JOSEKI [13]. Broadly speaking, JOSEKI is a DBMS for RDF data models that support a SQL-like language to query the data model. The Data Manipulation Language follows the JENA API [12], a *de facto* standard for insertion, deletion and updates of data models. This API is abstracted into the *Iontology* interface (see figure 2) with the aim of decoupling the system for the specific DBMS used to store the RDF descriptions. This permits to change either the ontology standard (e.g. OWL [22]) or the ontology repository without affecting the rest of the system.

4 The resourceCollection Component

The *Iresource* interface offers the signatures for inserting (PUT), renaming (MOVE), deleting (DELETE) and retrieving (GET) documents (see figure 2). This interface is realized using a *WebDAV* implementation.

WebDAV is an extension of *HTTP* that aims to provide a standard infrastructure for asynchronous, collaborative, authoring across the Internet. The *WebDAV* extensions support the use of *HTTP* for interoperable publishing of variety of content, providing a common interface to many types of repositories, and making the Web analogous to a large-grain, network-accessible file system [23].

WebDAV is also metadata aware. That is, it permits to collect (operational) metadata about the document being placed in the repository. These metadata, called "properties" in *WebDAV*, roughly correspond to the parameters of the *httpRequest* object which is generated when the document is transported to the repository. This information includes the IP of the user agent which generates the petition, or the date when the petition was issued. This is known as the *eventContext*.

5 The *ruleManager* Component

The heterogeneity of metadata as well as the impossibility to foreseen in advance the range of processors to be addressed, make us choose a rule-based approach. Event-condition-action rules (ECA rules) or triggers have long been available in Database Management Systems (DBMSs) to perform a wide range of tasks [19], including change propagations between tables.

In a DBMS setting, rules are tightly coupled with the "resource repository" (i.e. the database). This is not the case in this setting. Here, the RDF resource repository (i.e. JENA) and the document repository (i.e. WebDAV) are two different components. In this scenario, the rule manager needs to capture events coming from either component, and acts appropriately on the other component. This accounts for a loose coupling between both component.

Moreover, the self-contained nature of rules allows to encapsulate within a rule the specificities of a document editor. In this way, we cope with the heterogeneity of both metadata vocabulary and metadata location. A rule addresses the idiosyncrasies of a single document editor. Accordingly, introducing/deleting a format (e.g. *ppt*) is traced back to a single rule. If an existing editor changes the format of its metadata vocabulary/location in a new version (e.g. WORD), the impact of the change is isolated in the corresponding rule. This facilitates the evolution and maintenance of the system.

The *ruleManager* component provides the *Irule* interface with a single operation, *SIGNAL*, (see figure 2(c)), and utilizes both the *Iontology* interface and the *Iresource* interface. Basically, the component captures signals coming from the *resourceCollection* component, and reacts by invoking operations on the *ontology* components.

Implementation wise, the *Irule* interface is realized through the class diagram depicted in figure 2. The following classes are included:

- *EventDetector,* whose duty is to capture the signals. It implements the *SIGNAL* operation.
- *RuleManager*, which is the class responsible for obtaining the rules triggered by the event being signaled, for priorizing them, and for dispatching them in the priority order.
- *CAManager*, which is responsible for evaluating the rule's condition. This class is implemented using the *QuickRules* system [20] which provides an expressive condition language.
- *Rule*. The elements of an ECA rule are, not surprisingly, an event, a condition and an action. The event part of a rule describes a happening to which the rule may be able to respond. The condition part examines the context in which the event has taken place. Finally, the action describes the task to be carried out by the rule if the relevant event has taken place and the condition has evaluated to true. Besides, each rule has a name, a priority and a Boolean tag to indicate whether the rule is enabled or not.

Some rule examples are presented in the next section.

6 Some Rule Examples

This section illustrates how the *ontology* component can be isolated from the heterogeneity of the content providers through the use of rules. These rules are triggered by events produced by the *resourceCollection* component, and act on the *ontology* component. More specifically:

- *the event* is signaled by the *resourceCollection* component. Hence, the event corresponds to an insertion (PUT), renaming (MOVE), deletion (DELETE) or retrieving (GET) of a document. These events abstract away from the concrete classes that realise the component. In this case, these operations have a *WebDAV* counterpart, though *WebDAV* can be substituted for other repository without affecting the rules. The event is parameterized by an *eventContext* object. In this case, this object holds the *activeFolder*, i.e. the *WebDAV* folder where the document is placed, or the *mimeType*, which holds a regular expression on the files names (e.g. *"*.ps"* re-

fers to those files with *Postscript* extension). The *activeFolder* can provide properties about the resource. For instance, by placing a document on the folder VLDB, the value of a *conference* property can be obtained. Broadly speaking, the *eventContext* object mirrors the *HttpServletRequest* object.
- *the condition* is a predicate on the state of the context where the rule is fired. This context includes both the resource repository and the RDF descriptions.
- *the action* processes the resource, and propagates relevant metadata to the ontology layer. Hence, the action duties include retrieving the metadata, solving metadata mismatches between the terms used by the document editor and the terms of the ontology, and finally, adding the new RDF description. Most of these aspects are processor-dependent.

Next subsections provide examples for *OpenOffice, WORD* and *PDF*. The heterogeneity in the location and metadata model vindicates the use of the rule approach.

6.1 Rules for *OpenOffice*

Location Model. *OpenOffice* [3][2] uses XML as the storage format. Specifically, when storing an *OpenOffice* document on disk, an *OpenOffice*-compliant software creates a ZIP file which comprises four subdocuments, namely: *meta.xml*, which keeps the document meta information, *styles.xml*, which records the styles used in the document, *content.xml*, the content as such, and *setting.xml* which corresponds to application-specific settings, such as the window size or printer information. These four documents are delivered as a single ZIP file.

Metadata Model. Figure 4 shows an example of the metadata generated by *OpenOffice* when editing a manuscript similar to this one. Some tag examples follow: <*generator*> which corresponds to the document processor being used to generate the content, <*title*> which corresponds to the title of the document, <*editing-cycles*> which indicates the number of times the document has been edited, <*creator*> that specifies the name of the person who last modified the document, etc. Besides, these automatically-generated tags, the user can introduce up to four specific metadata. In this example, the authors of the manuscript have been added as <*user-defined*> metadata.

Extractive Process. When the user moves an *OpenOffice* document to the *WebDAV* repository, a rule is triggered that lifts this document onto the common ontology model. Specifically, the components of the rule are defined as follows:

[2] OpenOffice is an OASIS initiative to decouple content from how this content is processed. Today most of documents are stored within binary formats, which means they are worthless without the applications that created them. Worse, there is not guarantee, given today's undocumented, proprietary formats, that these documents will be readable even five years from now. Besides, that binary format prevents applications others than the creators from re-using these documents for their own means, and jeopardizes document portability and interoperability.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE office:document-meta PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN" "office.dtd">
<office:document-meta xmlns:office="http://openoffice.org/2000/office"
                      xmlns:xlink="http://www.w3.org/1999/xlink"
                      xmlns:dc="http://purl.org/dc/elements/1.1/"
                      xmlns:meta="http://openoffice.org/2000/meta"
                      office:version="1.0">
    <office:meta>
        <meta:generator>OpenOffice.org 1.0.3 (Win32)</meta:generator>
        <dc:title>Facing content-provider heterogeneity in Knowledge Portals</dc:title>
        <dc:description>
            Ontology languages are being proposed to provide machine-understandable
            descriptions of resources that permit easy location of these resource.
        </dc:description>
        <meta:creation-date>2003-05-21T17:41:00</meta:creation-date>
        <dc:creator>Ekin Member</dc:creator>
        <dc:date>2003-07-28T18:52:54</dc:date>
        <meta:keywords>
            <meta:keyword>Webdav</meta:keyword>
            <meta:keyword>RDF</meta:keyword>
            <meta:keyword>Document Engineering</meta:keyword>
        </meta:keywords>
        <dc:language>en-US</dc:language>
        <meta:editing-cycles>15</meta:editing-cycles>
        <meta:editing-duration>PT2H20M45S</meta:editing-duration>
        <meta:user-defined meta:name="Authors">
                Jon Iturrioz, Oscar Díaz, Sergio Fernández
        </meta:user-defined>
        <meta:user-defined meta:name="Info 2"/>
        <meta:user-defined meta:name="Info 3"/>
        <meta:user-defined meta:name="Info 4"/>
        <meta:document-statistic meta:table-count="0" meta:image-count="3" meta:object-count="0"
                                 meta:page-count="10" meta:paragraph-count="304"
                                 meta:word-count="1905" meta:character-count="14613"/>
    </office:meta>
</office:document-meta>
```

Fig. 4. *OpenOffice*. The content and the metadata of the document are placed in two different files. The figure shows the one holding metadata.

- an event, that can be described by the triplet <*PUT, *, "*.sxw"*>. This triplet indicates that the event happens as a result of introducing a new document in *WebDAV* regardless of the folder. However, the extension of the document must be *"sxw"* (i.e. that is, an *OpenOffice* document) for the rule to be fired.
- a condition, that verifies that there exists no document with the same title in the repository -since the title is used as key. The JOSEKI's API is used for this purpose. Specifically, the *exists* operation takes a *key* as an input, and returns a boolean which indicates whether a resource with this key exists or not.
- an action, that updates the RDF model to reflect this insertion. This implies to:

  1. extract the metadata from the document to an XML intermediate document,
  2. map the *OpenOffice* metadata model to the canonical ontology model defined in RDF,
  3. append additional data obtained from *WebDAV* (e.g. the <*submitted_date*> property),
  4. introduce the new RDF description.

The concrete code follows:

```
//Extract and normalize metadata from document
zipFilePath = eventContext.getFileName( )
xmlDoc = OpenOfficeExtraction.extract (zipFilePath)
//Transform to the RDF description
rdfDocument = RDFTransform.processXML(xmlDoc, zipFilePath)
//Introduce the new resource description into the RDF model
joseki.insertOnt ( zipFilePath, rdfDocument)
```

6.2 Rules for WORD 2003

Location Mode. *Microsoft Office Word 2003* widely embrace XML. Specifically, this new release introduces the *"Save as XML"* command that produces an XML-compliant counterpart of the traditional, proprietary Word-format files [4]. These XML files are produced according with the *WordML* schema, a *XML Schema* that is designed to mirror the information found in traditional *.doc* files. The root element of a *WordML* document is always *w:wordDocument*. This element contains several other elements that represent the complete *Word* document structure, including properties, fonts, lists, styles, and the actual document body that contains sections and paragraphs. Figure 5 gives an example.

Metadata Model. For the purpose of this work, we focus on the element *<o:DocumentProperties>* (see figure 5). This element contains the document properties available for all *Microsoft Office* documents such as title, author, last author, creation date, last saved date and so on. Notice that this element is from a different namespace that *WordML* since it applies to all *Microsoft Office* documents, and then, it will be shared across *Office* XML vocabularies (e.g. *ExcelML*). This greatly facilitates our job as the homogenization of the metadata is at least already achieved for *Microsoft Office* resources.

Moreover, a WordML document also holds a <docPr> element that contains Word-specific properties for the given document. So far, these properties are mainly related with style-like issues (e.g. view and zoom settings, existence of a template). However, nothing prevents the user from using this element to keep content-related metadata. Indeed, annotation mechanisms such as those described in [14,10] can be used to obtain metadata from the document content. When the document is saved, "the annotator" attaches the corresponding metadata to the *<docPr>* element.

Extractive Process. When the user moves a *"Word.xml"* document to the *WebDAV* repository, a rule is triggered that lifts this document onto the common ontology model. Specifically, the components of the rule are defined as follows:

- an event, that can be described by the triplet *<PUT, *, "*.xml">*. This triplet indicates that the event happens as a result of introducing a new document in *WebDAV* regardless of the folder. The extension of the document must be *"xml"*. However, this is not enough to ensure that this document comes from a *Word* processor. An additional verification needs to be done in the rule's condition.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?mso-application progid="Word.Document"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
    xmlns:o="urn:schemas-microsoft-com:office:office">
    <o:DocumentProperties>
        <o:Title>Facing content-provider heterogeneity in Knowledge Portals</o:Title>
        <o:Author>Sergio F. Anzuola</o:Author>
        <o:LastAuthor>Jon Iturrioz</o:LastAuthor>
        <o:Revision>2</o:Revision>
        <o:TotalTime>1</o:TotalTime>
        <o:Created>2003-08-14T12:44:00Z</o:Created>
        <o:LastSaved>2003-08-14T12:44:00Z</o:LastSaved>
        <o:Pages>1</o:Pages>
        <o:Words>10</o:Words>
        <o:Characters>59</o:Characters>
        <o:Company>Ekin Group</o:Company>
        <o:Lines>1</o:Lines>
        <o:Paragraphs>1</o:Paragraphs>
        <o:CharactersWithSpaces>68</o:CharactersWithSpaces>
        <o:Version>11.5329</o:Version>
    </o:DocumentProperties>
    <w:fonts>
        <!-- fonts used in document -->
    </w:fonts>
    <w:styles>
        <!-- named styles used in document -->
    </w:styles>
    <w:docPr>
        <!-- Word-specific document properties -->
    </w:docPr>
    <w:body>
        <!-- document content -->
    </w:body>
</w:wordDocument>
```

Fig. 5. *WordML.* Both the content and metadata of the document are located in the same XML file.

- a condition, that first checks that the *".xml"* document has been produced with a Word editor and hence, follows the WordML schema. To this end, the condition checks the existence of a processing instruction in the *".xml"* document: *<? mso-application progid= "Word.Document" ? >*. Next, and similar to the previous rule, the condition verifies that there exists no document with the same title in the repository.
- an action, that updates the RDF model to reflect this insertion. This implies to:

1. extract the metadata from the *WordML* document (i.e. the *DocumentProperties* element),
2. map the Word-proprietary metadata model to the canonical ontology model defined in RDF,
3. append additional data obtained from *WebDAV* (e.g. the <*submitted_date*> property),
4. introduce the new RDF description in the ontology model.

The code is similar to rule 6.2.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:iX="http://ns.adobe.com/iX/1.0/">
    <rdf:Description about="" xmlns="http://ns.adobe.com/pdf/1.3/" xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
        <pdf:CreationDate>2003-11-28T09:43:08Z</pdf:CreationDate>
        <pdf:ModDate>2003-11-28T09:43:08Z</pdf:ModDate>
        <pdf:Producer>Acrobat Distiller 5.0 (Windows)</pdf:Producer>
        <pdf:Author>Sergio F. Anzuola</pdf:Author>
        <pdf:Creator>PScript5.dll Version 5.2</pdf:Creator>
        <pdf:Title>
            Facing content-provider heterogeneity in Knowledge Portals
        </pdf:Title>
        <pdf:Subject>
            Ontology languages are being proposed to provide machine-understandable
            descriptions of resources that permit easy location of these resource.
        </pdf:Subject>
        <pdf:Keywords>Ontology, Document Engineering</pdf:Keywords>
    </rdf:Description>
    <rdf:Description about="" xmlns="http://ns.adobe.com/xap/1.0/" xmlns:xap="http://ns.adobe.com/xap/1.0/">
        <xap:CreateDate>2003-11-28T09:43:08Z</xap:CreateDate>
        <xap:ModifyDate>2003-11-28T09:43:08Z</xap:ModifyDate>
        <xap:Author>Sergio F. Anzuola</xap:Author>
        <xap:MetadataDate>2003-11-28T09:43:08Z</xap:MetadataDate>
        <xap:Title>
            <rdf:Alt>
                <rdf:li xml:lang="x-default">
                    Facing content-provider heterogeneity in Knowledge Portals
                </rdf:li>
            </rdf:Alt>
        </xap:Title>
        <xap:Description>
            <rdf:Alt>
                <rdf:li xml:lang="x-default">
                    Ontology languages are being proposed to provide machine-understandable
                    descriptions of resources that permit easy location of these resource.
                </rdf:li>
            </rdf:Alt>
        </xap:Description>
    </rdf:Description>
    <rdf:Description about="" xmlns="http://purl.org/dc/elements/1.1/" xmlns:dc="http://purl.org/dc/elements/1.1/">
        <dc:creator>Sergio F. Anzuola</dc:creator>
        <dc:title>
            Facing content-provider heterogeneity in Knowledge Portals
        </dc:title>
        <dc:description>
            Ontology languages are being proposed to provide machine-understandable
            descriptions of resources that permit easy location of these resource.
        </dc:description>
    </rdf:Description>
</rdf:RDF>
```

Fig. 6. *PDF*. Both the content and metadata of the document are placed in the same BINARY file.

6.3 Rules for PDF

Location Model. PDF stores the document's content in a binary file using a native format, and uses *Adobe Extensible Metadata Platform* (XMP) [5] to embed the document's metadata into the very same file. Metadata extractors retrieve the document's metadata through a XMP's API (for a concrete realisation, see [15]). It is important to note that XMP can be applied to any binary file other than PDF (e.g. JPEG).

Medatada Model. The metadata stored in PDF documents can be split into three sections (see figure 6). First, a section dedicated to PDF standard properties. A second section that holds *Adobe XAP* digital asset metatadata. This is an initiative to find a

common ground to describe digital resources [6]. Finally, a last section that captures *DublinCore*-compliant properties. As shown in figure 6, these three views overlap for some properties that are replicated in the three sections though using different annotation vocabularies.

Extractive Process. In this case, the rule's parts include:

- an event, described by <PUT,*,"*.pdf">. This event happens when introducing a *PDF* file in the *WebDAV* repository,
- a condition, that checks out the document's title is unique,
- an action, that introduce a new RDF instance that correspond to this new resource. This is similar to the previous cases. The difference stems from the metadata being obtained through the XMP's API. Specifically, this work uses the *Java XMP Parser* found at *http://www.kegel-mediendesign.de/XmpUtil/* .

7 Conclusions

As new tools require/provide metadata about the resources they process, it becomes paramount to facilitate the sharing of metadata among heterogeneous tools. This paper introduced an architecture which is characterized by (1), an ontology layer that provides a canonical model for resource metadata, and (2), a rule component for achieving a metadata "synchronization" between the ontology layer and the resource layer. The use of this reactive architecture accounts for the separate evolution of each layer as well as the inclusion of new document processors as time goes by.

As a proof of concept, this work outlines an implementation using *WebDAV*, *JOSEKI* and *QuickRules* as the supporting technologies, and *OpenOffice*, *WORD* and *PDF* as the content types.

Our next work addresses how to embed annotation tools into document processors so that metadata can be automatically obtained at the time the document is saved.

References

1. IETF WEBDAV Working Group Home Page, 2002. http: //www.ics.uci.edu/ ejw/authoring
2. Dublin Core Metadata Initiative, 2003. http://dublincore.org
3. OpenOffice.org 1.0: The Open Source Office Suite, 2003. http: //www.openoffice.org
4. Aaron Skonnard. The XML Files: XML in Microsoft Office Word 2003 – MSDN Magazine, November 2003. http: //msdn.microsoft.com/msdnmag/issues/03/11/XMLFiles/default.aspx.
5. Adobe.XMP Extensible Metadata Platform, 2002. http://partners.adobe.com/asn/ developer/xmp/pdf/ MetadataFramework.pdf
6. Andrew Salop. XAP, a Digital Asset Metadata Architecture Utilizing XML and RDF, 2001. http: //www.gca.org/papers/xmleurope2001/papers/html/sid-03-9b.html
7. W3C: World Wide Web Consortium. Resource Description Framework (RDF)/W3C Semantic Web Activity, 2003. www.w3.org/RDF
8. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.

9. Daniel Oberle and Peter Spyns. The knowledge portal "ontoweb". *STAR Lab Technical Report*, 2003.
10. S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *The Eleventh International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA*, pages 462–473, 2002.
11. Alberto Pan, Juan Raposo, Manuel Álvarez, Justo Hidalgo and Ángel Viña. Semi-automatic wrapper generation for commercial web sources. In Colette Rolland, Sjaak Brinkkemper, and Motoshi Saeki, editors, *Engineering Information Systems in the Internet Context (EISIC)*, volume 231 of *IFIP Conference Proceedings*, pages 265–283. Kluwer, September 2002.
12. HP. jena, 2003. http: //www.hpl.hp.com/semweb/jena.htm
13. HP. Joseki, 2003. http: //www.joseki.org
14. Jose Kahan and Marja-Ritta Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In *World Wide Web*, pages 623–632, 2001.
15. Knud Kegel. Java XMP parserl, 2003. http: //www.kegel-mediendesign.de/XmpUtil/
16. Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Internation Conference on Data Engineering (ICDE)*, pages 611–621, 2000.
17. Ling Liu, Calton Pu, and Wei Han. An XML-enabled data extraction toolkit for web sources. *Information Systems*, 26(8): 563–583, 2001.
18. Mustafa Jarrar and Robert Meersman. Formal Ontology Engineering in the DOGMA Approach. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1238–1254. Springer-Verlag, 2002.
19. N.W. Paton and O. Diaz. Active Database Systems. *ACM Computing Surveys*, 1(31): 63–103, 1999.
20. Yasu Technologies. Java rules engine, 2003. http://www.yasutech.com/products/quickrules/ index_SE.htm
21. Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2): 93–155, 1996.
22. World Wide Web Consortium (W3C). Owl web ontology language overview, 2003. http://www.w3.org/TR/owl-features/
23. E. James Whitehead and Meredith Wiggins. WEBDAV: IETF Standard for Collaborative Authoring on the Web. *IEEE Internet Computing*, 2(5): 34–40, 1998.