Provenance Metadata for Shared Product Model Databases

E. Petrinja & V. Stankovski & Ž. Turk
Chair of Construction Informatics, Faculty of Civil and Geodetic Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia

ABSTRACT: The process of saving metadata committed to track all changes to some data, is known as "provenance". In the AEC/FM sector provenance data can be exploited for tracking all interactions of different users between each other and with parts of data. For a particular application, we need to consider which metadata are essential for future queries and who is going to use these. The IFC standard already contains some provenance concepts in its entity structure. We have considered these provenance concepts to build a provenance tracking software. The provenance ontology server was developed by using the OWL ontology language, already available IFC concepts and some complementary concepts that we had to include for the sake of generality of our implementation. The developed prototype allows us to upload an IFC file to a web enabled service that parses it and saves instances of retrieved concepts for later queries, according to the ontology we have defined.

1 INTRODUCTION

Managing AEC/FM projects is a challenging task, especially when it comes to the evolving tendencies to outsource different parts of a project to partners World-wide. Information technology gradually makes it possible to control these complex underlying processes. Computer aided engineering modeling standardization is an issue that has been largely taken in consideration in the past. Many standards have been adopted, the most important being the ISO 10303 family of standards (known with the acronym STEP). The Industry Foundation Classes (IFC) - http://www.iai-international.org) standard has been developed specifically for the AEC/FM sector and is conforming to the STEP standard.

For the purpose of managing the product design process a considerable amount of metadata has to be saved along the project data. These data are important for future changes of the project, optimization of particular modeling methods, re-use of already adopted work methodologies and other important managerial issues. In this paper we refer to these metadata as "provenance" data. Few research projects have already considered provenance data from different prospective like myGrid and PASOA projects. We focus our research efforts to understand how it is possible to effectively extract provenance data from already existing IFC files and add some additional provenance information to these IFC data files.

In this paper we present the design and a prototype of a system for managing provenance data about IFC data files. A particular attention in the design phase was given to ensure interoperability of the system with other IFC enabled software products, such as ArchiCAD, Autodesk and others. The working prototype is available through a web browser and it is platform independent. It allows uploading of IFC files, it extracts provenance data from these files and it saves the provenance data in a persistent storage system, which was designed by using the MySQL (www.mysql.com) data management system. The provenance data is formatted and stored according to the conceptual model, which was built by converting specific IFC schema entities and functions like the ifcOwnerHystory entity into the OWL (www.w3.org/TR/owl-features) language, which was found to be well suited for this purpose. The OWL language is standardized by the WWW Consortium and it is used to define complex conceptual models, such as IFC. In the terminology of the OWL standardization work-group these conceptual models are also called "web ontologies". By using the OWL language it is actually possible to define axioms, concepts as well as instances of the concepts. The use of OWL has contributed a flexible and modularized system that offers a high level of interoperability with other already developed IFC compliant programs, which are used by engineers, architects and construction managers.

Construction Informatics Digital Library http://itc.scix.net

2 THE PROVENANCE METADATA CONCEPT

A better classification of data present on the web was the main reason for the distinction between data and its presentation (Barners-Lee, 1997). Along the data we have to consider also metadata that describes it. Provenance data are just part of these metadata, which is concerned with the time of creation, change or deletion of a described data. These kind of metadata is usually referred to as provenance data. We can define provenance data from the end product view as metadata relating data to other data (Myers et al., 2003) or from the process view point as the documentation of the process that leads to some results (Groth et al., 2004).

The web gives us the opportunity to search millions of documents and it has given to researchers another opportunity of gathering data and knowledge in an effective way. However accuracy and trust in knowledge found is not assured as it was with paper based knowledge representation. Data from the web can be changed frequently and by different individuals, but rarely the changes are registered with appropriate metadata that is saved along the data or in other databases. Provenance data provides a traceable path to the origins of other peaces of data. Using an appropriate mechanism to save provenance data, we could solve partly the problem of trustworthy of documents found on the web. At the moment many partial software solutions already exist. Scientists can annotate in-silico experiments on their file system, into some databases or into an electronic notebook, but a standardized prescription of which provenance data is important to be saved and how to share it with others, is still missing.

Knowing what actions lead to a particular value of an element in the product model by saving provenance data of that transactions is important from at least the following viewpoints:

- Legal - Who is responsible?
- Professional - Why does it have such value?
- Managerial - Who is doing the work here?
- Re-use - Can we re-use the process next time we do something similar?

There are two different granularities of provenance metadata (http://www.nesc.ac.uk/esi/events/304) the course-grain and the fine-grain provenance. First refers to movement of data between different databases (Bunemanet al., 2000) in the process of creation of different curated databases. The second traces provenance data generated by workflow management engines and services enacted by a particular workflow. It is argued that the automation of provenance metadata saving mechanism is essential for an efficient tracking of all necessary information to make it possible to rerun workflows and to obtain already computed results by changing only a part of input data (Bose, 2002). Manually entering the information could lead to incomplete provenance documentation, moreover, people are not willing to spend a lot of time tracking and annotating such data.

Automating the process of tracking provenance data and saving it in different databases would allow us to save as much data as we wish or we think we could use in the future. However, there is a cost benefit in using provenance data for certain application. Saving huge amounts of data that could be used one day is not feasible. We have to decide which provenance metadata is really worth saving for future queries. Different provenance data consumers may need completely different parts of data. Besides, everybody has a personal view on the process tracked and registered by a particular part of provenance metadata. We are tempted to save all possible information that a process or a workflow produces. However there are limitations. It is possible to conclude that the amount of provenance metadata is different for every single application whether it is scientific or not. A standardized list of the important topics and parts of the process to be traced is necessary for all possible utilization of provenance metadata saving mechanisms.

3 RE-VISITING THE IFC STANDARD

The STEP standard for product model data is an effort to solve the long lasting and extremely expensive lack of standardization in the segment of product models and product models data exchange between various stakeholders of the manufacturing industry. STEP is now a mature standard implemented in almost all software applications covering single aspects of the manufacturing industry. It is well known in the USA where it is implemented in the AP-203 protocol and in Europe where it is implemented in a similar protocol named AP-214.

A representative group of software developers, information providers, engineers and other stakeholders founded the International Alliance for Interoperability (IAI) organization in 1994. Its main purpose is formulation of AEC standards that would be acceptable World-wide. The first release of the IFC standard was published one year after the foundation of the IAI organization. The main goal of the IFC standard is to achieve interoperability of various software products that cover different stakeholders in the AEC sector. The IFC standard was derived from the already existing ISO 10303 standard (STEP standard) and specified in the Express language (ISO 10303:11). The Express language was developed for conceptual modeling of domains within the field of product data. It was structured with a conscious attempt to avoid including constructs directed towards the ease of implementation or development of physical schemas. Its main constructs are entities that

Construction Informatics Digital Library http://itc.scix.net

have associated attributes and constraints. The second are written using an expressive mix of a declarative and a procedural language.

The IFC standard has changed considerably since the first version. The actual up to date standard is the release IFC 2x2. Since the 2x release the standard core of the standard is fixed and only additional parts change over time.

Founding its initial formation on this already well consolidated and accepted standards, IFC had grew fast and it became a solid and trustworthy standard. It also uses many concepts from research and development projects in the area of information technology.

Its developers describe the four main axes along which the IFC standard extends. The axes are:

- lifecycle that explains how the standard tries to cope with the entire engineering process and every single stage connected with the modeling of a construction product,
- discipline where the objectives of the standard are, to make interact every discipline (or role) that is involved in the process of product modeling,
- the level of detail is also an important topic in the creation of the IFC standard because the vast array of concepts that are present in the AEC/FM information world is virtually impossible to include in a single standard and
- the software axes that is also a very important topic when there is an attempt to create a standard that allows complex interoperability between various applications.

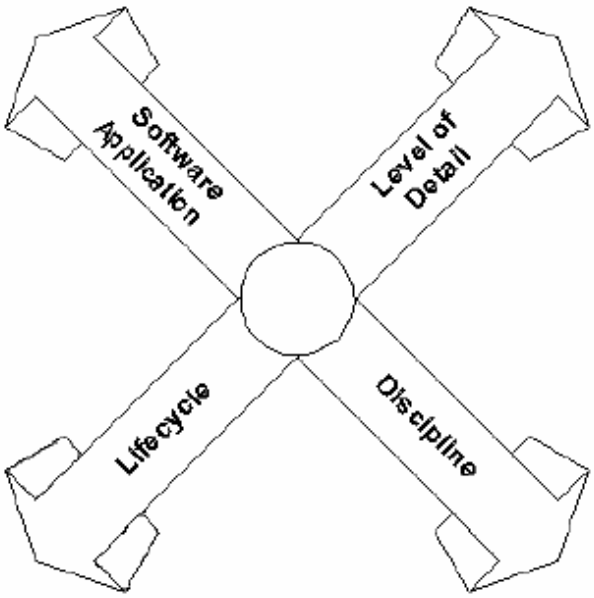

Figure 1. IFC Axes (source: http://www.iai-international.org)

The IFC model architecture is structured into four layers that are connected one with the other by the gravity principle (see Figure 1). Conceptually lower layers cannot reference objects from higher layers but the contrary is possible. The main four layers are (from the lower to the higher layers): Resources layer, Core layer that is subdivided into the Kernel part and the Extension schema part, the Interoperability layer and the higher Domain layer. The IFC kernel consists of the conceptually highest part of the model, all other concepts are specified by a particular concept from the kernel part. The central concept for all others that are specified, is the IFCRoot concept. Attached to this principal concept is also the IFCOwnerHistory. The first level of specialization of the root concept in the IFC standard represents three specific concepts that are:

- ifcObject which describes all physically tangible items,
- ifcPropertyDefinition representing the generalization of all characteristics of objects and
- ifcRelationship that objectifies all relationships that might occur in the model.

A similarity with the OWL ontology definition can be found in higher concepts of the IFC model. A more in depth view is presented in the ontology part of this paper. The ifcObject concept is devided into seven (second level of specification) sub concepts: ifcProduct, ifcProcess, ifcControl, ifcResource, ifcActor, ifcGroup and ifcProject. The ifcRelationship concept specifies into five sub concepts: ifcRelAssigns, ifcRelDefines, ifcRelAssociates, ifcRelConnects and ifcRelDecomposes. IfcPropertyDefinition is specified with two sub concepts that are: ifcTypeObject and ifcPropertySetDefinition.

All simple data types presented in the IFC model shall be specified as defined data types in our ontology.

From the release IFC 1.5.1 all software implementing the standard can undergo a testing process and if certain prescribed quality standards are meet, the software is appropriately certified. The software may be approved for a particular IFC release, different certifications for different applications and similar.

4 PROVENANCE ONTOLOGY CODED WITH THE OWL SYNTAX

Our provenance application relays on ontology technology that became an important research topic in the late nineties. The central role of ontologies is to keep semantics about every concept present in a particular domain. Frequently used definition of ontology is that it is an explicit formal specification of how to represent the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them (http://www.doi.org/handbook_2000/glossary.html).

They were introduced in the so-called “Semantic Web Stack” as the central tool for saving the semantics in the envisioned future web (Berners-Lee, 1997).

Construction Informatics Digital Library http://itc.scix.net

Two persons can communicate because they have a common understanding of the concepts they are talking about. We are thought first the basic and later more and more complex concepts about everything that is around us. We learn rules that govern the world around us every day. However these basic concepts do not change a lot from the principles we already know. There is a huge effort invested by every single person to learn and understand the common concepts. This makes him able to communicate with others that have learnt same or very similar concepts. Computers did not take advantage of this learning process in their past, so they can not understand what is written in simple text files for example.

There is also another problem concerning human to computer interaction. Namely, the natural language used by humans is extremely difficult to be used to define commands for computers (research is done in this sector as well), so a more formalized language has to be used. The amount of formalized computer programming languages that were defined in the last century is huge. XML is one of them. It defines the syntax to be used for expressing almost anything we would like. It is nowadays accepted as a general standard on which the envisioned Semantic Web is based.

In XML we can write a line in this way: <user>SomeName</user>. Using tags we delimit a part of the text that holds a word. We understand that the word SomeName must be a name of somebody is using something, because we have the common concepts understanding in our minds, but to the machine it does not mean anything, unless we have a definition of the tag <user> written in a document that is accessible to the machine. Many different languages for expressing ontology concepts have been used in the past, some of them are: FLogic, OCML, LOOM, OKBC, Ontolingua, KIF, RDF(S), OIL, DAML+OIL and the most recent OWL. We can define the meaning of a concept in an XML file also by using the DTD notation or the XML Schema, but ontologies are much richer in these semantic expressing power. The OWL Web Ontology Language was designed for use by applications that need to process the content of information instead of just presenting information to humans. It has evolved from the former ontology language DAML-OIL and facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. Using it there are almost no concepts we can not describe. However, the OWL Full expressive power is so great that it does not guarantee computability when DL logic rules are applied.. OWL DL on contrary is a bit more restricted and less expressive but it complies with DL logics rules and can be processed in different forms. We had used the OWL DL notation for expressing the provenance ontology used in our application.

Many different tools and toolkits for building and managing ontologies have been developed so far.

We used the Protégé application (http://protege.stanford.edu) that was developed at the Stanford university and it is now available as version 3.0. At the moment it is probably the most elaborate and complete toolkit for managing ontologies. For the purpose of our application simple text OWL form of the ontology was used and Protégé allows exporting the entire ontology project into the desired owl format. Additional editing is possible with a simple text editor application. With one of the many graphical plug-ins that are available for Protégé, the ontology can be represented in a more understandable graphical format.

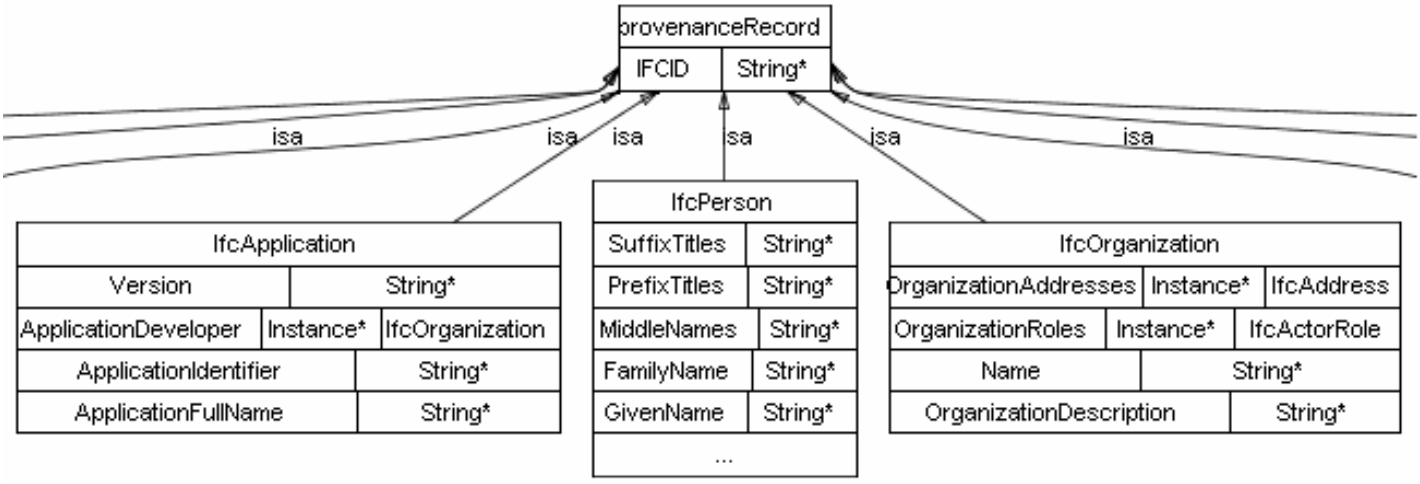

Figure 2. Provenance ontology

In our application we defined a relatively simple ontology that was focused on only some IFC entities concerning provenance concepts. The main goal in the development of this prototype application was to reuse the already available concepts defined in the IFC standard and implemented in various IFC com-

Construction Informatics Digital Library http://itc.scix.net

pliant applications already available to the community. First we decided to limit the scope of this application to the IFC schema version 2.0 mainly because the last ArchiCAD application version is based on this schema as well. It is largely used also by many other IFC compliant applications (as Autodesk ADT, Visio…). A more general development taking in consideration all IFC schemas available (at least the last three versions), will be an issue for future work. For the purpose of our research we restrained ourselves to entities in the IFC schema that are considered useful for some provenance data tracking. The main entity of the IFC standard describing concepts close to provenance data is the ifcOwnerHistory entity (as already mentioned before) and few sub entities. The fact that this entity is attached to the root entity makes it a very largely utilized concept of the schema, because every single instance of an entity defined in the IFC schema can have some data that is connected to the ifcOwnerHistory entity. Root properties are inherited by all sub entities and by their instances. Interesting ifcOwnerHistory sub entities are ifcAuditTrail (7), ifcTransaction (3), ifcApplication (4), ifcOrganization (4), ifcAddress (13), ifcActorRole (2), ifcPersonAndOrganization (4) and ifcPerson (6). The number near each entity name is the number of attributes of every single entity that is defined in the IFC standard and one by one included into our ontology.

In the IFC standard these nine entities form a subtype super-type tree but for a better management of instances we gathered them at the same level under the provenanceRecord concept in our ontology. Our ontology root concept provenanceRecord has only one property that is the IFCID number present at the beginning of every line of an IFC part 21 file. OWL ontology parent properties are inherited by all sub concepts, and the IFCID property is thus present in all provenanceRecord sub concepts. It could be also possible to add to the provenance ontology some comments or annotation concepts that are usually present in the provenance data, however, this was not essential for our research. Our aim was to combine IFC entities with an ontology centered provenance server. However for a more complete provenance server, such concepts would be a useful help to server users. Properties defined in the IFC schema are of different types. They can be simple data types as Strings, Integers and others or relational types that connect to other entities present in the IFC schema. Similarly we can define two different types of properties in OWL ontology: DataType properties and Object properties. The properties used by our prototype have been appropriately mapped between the schema and the ontology.

Part of the ontology is visible in the simple text form and part in a graphical representation visualized by the Protégé plug-in.

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.prov.org/prov_IFC20.owl#"
  xml:base="http://www.prov.org/prov_IFC20.owl">
  <owl:Ontology rdf:about=""/>
<!-- Classes -->
  <owl:Class rdf:ID="provenanceRecord">
  </owl:Class>
  <owl:Class rdf:ID="IfcPersonAndOrganization">
    <rdfs:subClassOf rdf:resource="#provenanceRecord"/>
  </owl:Class>
…
<!-- ObjectProperties -->
  <owl:ObjectProperty rdf:ID="OwningUser">
    <rdfs:domain rdf:resource="#IfcOwnerHistory"/>
    <rdfs:range rdf:resource="#IfcPersonAndOrganization"/>
  </owl:ObjectProperty>
…
 <!-- DatatypeProperties -->
  <owl:DatatypeProperty rdf:ID="IFCID">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#provenanceRecord"/>
  </owl:DatatypeProperty>
…
<!-- DatatypeProperties -->
  <owl:DatatypeProperty rdf:ID="TransactionDate">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#IfcTransaction"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Figure 3.Part of the ontology coded in OWL language

5 THE PROVENANCE DATA SAVING SERVER

Our aim was to develop a prototype provenance server application that would be available through a simple web browser, that it is an essential part of almost any personal computer today, so that our application is accessible to virtually anyone having an internet connection, a browser and can go online. This envisioned goal has led us in the process of deciding which technology to use in the implementation of the prototype. A web application can be utilized by users that use different operating systems and/or computers architectures.

Examination of these requirements led us to the decision to use the Java programming language for the development process. However Java alone is not enough for writing a web enabled application so we wrote the web part of the prototype with JavaServer pages language combined with simple html. We used Jakarta Tomcat for simulating a local server.

Construction Informatics Digital Library http://itc.scix.net

This servlet container can be used as an effective online web server. JSP applications are usually structured with some html/JSP pages and some javaBeans that are pure java classes. JSP is like a bridge between the vastly utilized html representation of data on the web and a powerful programming language as Java is.

The central part of our prototype is relaying on the Jena API (http://jena.sourceforge.net). It is a java framework for building semantic web applications. It provides classes and methods for creating and managing RDF, RDFS and OWL documents. The entire framework includes different semantic web concepts and tools:

- RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- OWL API
- In-memory and persistent storage
- RDQL – a query language for RDF.

We did not use the RDF specifications connected APIs for our prototype, nevertheless because of the integration of different classes inside the Jena framework, the OWL API relies strongly on the RDF API and the main model structures in OWL model schema are inherited or specialized from RDF classes. To achieve our goal we used the persistent storage mechanism supported by Jena (see Figure 4). The rationale is as follows. The envisioned prototype must be able to store enormous quantities of data that must be accessible as quickly as possible. A persistent database storage mechanism is almost obligatory for storing big quantities of data. IFC part 21 files are usually very large. A common single family residential house project like the one we used as a test example had nearly one hundred thousand lines of Express language syntax code. A file based storing mechanism can be good for testing the prototype, but it could have some serious efficiency problems with real life project data. Jena supports three different database systems (MySQL, PostgreSQL and Oracle). We implemented the prototype using the MySQL database management system.

In Jena it is possible to create an ontology model (OntModel) that is an extension of the Jena RDF model. The default model uses OWL Full as the ontology language, so we had to consider the limitations we wanted to take into account for the OWL DL language. By simply using the modelFactory method we create new models. Later we can import already available models from some files or from the persistent storage in a database system (MySQL in our case). If parts of the ontology are not jet created, we can model them according to our needs and append them to the model. The most important objects we can insert into an ontology model are ontology concepts, properties, restrictions and instances. All four objects are derived from the OntResource super class and thus have a common insertion and modification procedure. The OntResource is by itself a Java interface that extends the Jena's RDF Resource interface. Ontology models can be interfaced by the Jena graph to a reasoner that is not predefined. It can be an external application run separately from the central application and can answer user queries. Our application's queries were predefined by ourselves and integrated into the main application class. A separate reasoner was not necessary. However, for future development of the prototype we will have to consider which reasoner would better fit our needs.

In Figure 4 we can see the entire architecture of the prototype. The central part is represented by the main Java class that is physically a compiled .java class saved in the class folder of the Tomcat deployment folder. All necessary java libraries must be placed in the lib directory where we have copied also the Jena API classes. We decided to keep the entire provenance server application coded in a single java file. For a more complex prototype, an appropriate code split mechanism should be adopted.

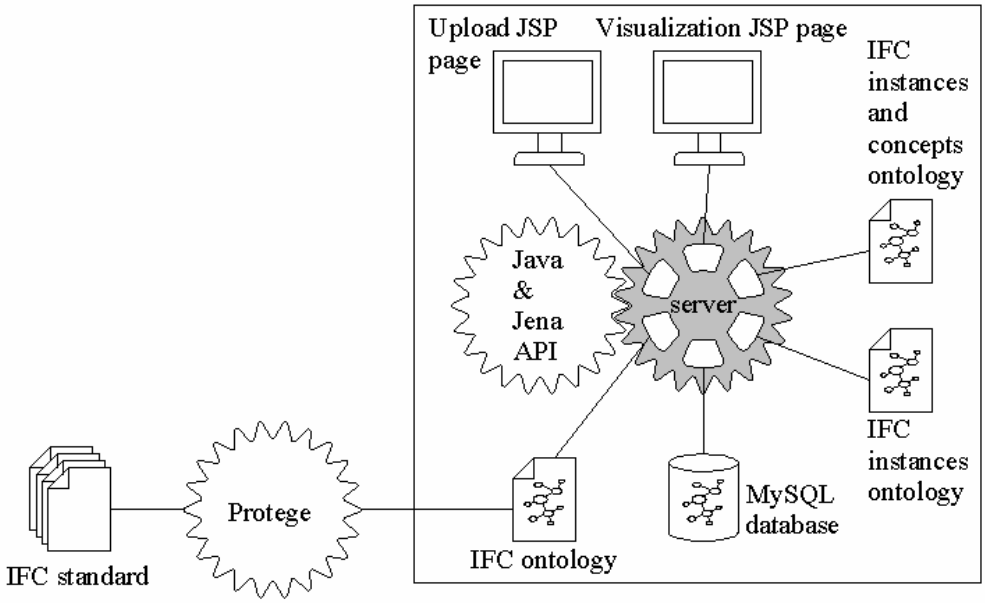

Figure 4. Prototype architecture

Construction Informatics Digital Library http://itc.scix.net

In Figure 4 we can see that there are three ontology files and one file, which is database enabled. The main OWL ontology of the IFC version 2.0 LONGFORM schema is saved separately from the instances. Keeping separately ontology concepts, properties and rules from the instances, we have a clearer picture of the ontology structure and we can reuse the ontology without the instances. Nevertheless the amount of data instances could be considerably greater than the classes and properties part of the ontology. Keeping them separate is thus a smart decision and the Jena toolkit supports this practice. The IFC version 2.0 concepts ontology were prepared as already described and the file saved along the main java class file. Instances files are created during the process of data uploading by the prototype. For the control reasons of the prototype, we allowed the application to create an OWL file with both the concepts and the instances together in the same file. We can import this file easily into the earlier mentioned ontology application Protégé, where we can check the correct instances saving process. This file is created only for the purpose of checking the prototype coding and it will be of minor importance to an end user.

The main repository of instances is saved into the MySQL database. Jena creates automatically tables necessary for its proper functioning and saves instances in a RDF triplet form. It saves separately subjects, objects and predicates. By uploading different IFC files, the server creates separate tables and names them according to the name of the just uploaded file. Instances are all created from a single instance class, so we had to invent a mechanism to keep their identity unique. Thus a mechanism for keeping their identity unique was necessary. We included a unique number in the name of every single instance that is present in each line of a document written in the Express language.

Two JSP pages were created. One for uploading IFC files and the other for queering it and visualizing the provenance data stored in the uploaded files. A more efficient and rich tools are planed for the future development of our provenance prototype. Figures 5 and 6 represent the upload and the visualization screenshots.

The core of the server application is a parser that extracts data from the uploaded IFC files and saves them according to the defined ontology in appropriate instance elements in a database. We have conducted an extended research to find an already implemented parser for IFC files coded in java language, but we have found only some limited versions available on the web. For this reason, a new provenance data extraction mechanism was implemented. We wanted the server to collect only data concerning provenance data saved in the processed files. The application finds first just the lines containing provenance data then it finds out what kind of data is written in each line by matching it with the information retrieved from the uploaded ontology and than it saves each part of the data in separate variables according to the same ontology. The IFC schema defines exactly which part of data can be saved in a particular position of each line. It is thus straightforward to match the extracted data with the correct ontology properties.

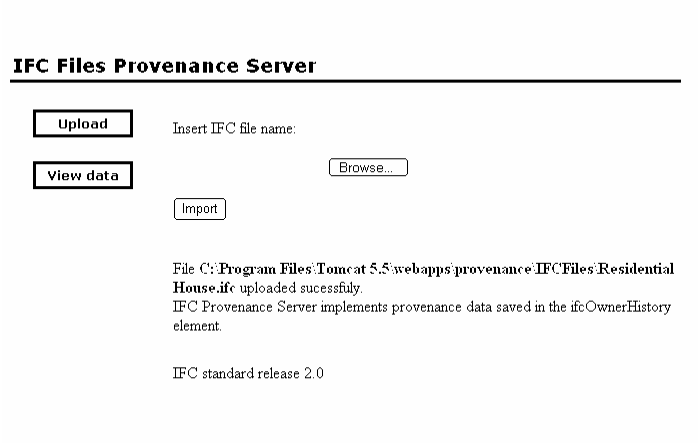

Figure 5. IFC Files Provenance Server Upload screenshot

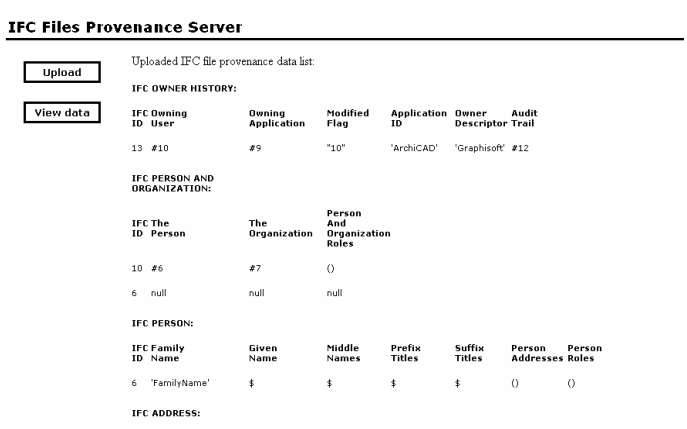

Figure 6. IFC Files Provenance Server Visualize screenshot

6 CONCLUSIONS AND FUTURE WORK

The paper is concerned with the relatively new research area of provenance data management. Collecting metadata about events occurring in the process of modeling, an AEC/FM product can be of great importance to the future reuse of some already used or developed methods and methodologies. It can make possible a more reliable truth conformance checking mechanisms and can help avoid unnecessary mistakes in future similar projects. An important research part of the work presented in this paper is the utilization of ontologies to save and manage provenance metadata. By dividing the conceptual and the instances part we can manage the repository better and can use different ontology concepts or entire ontologies within the same database and the same provenance management server prototype. The central part of the prototype presented, was the web enabled IFC files management system. Provenance

Construction Informatics Digital Library http://itc.scix.net

data is extracted from the uploaded files according to the conceptual layer stored in the ontology and saved into the MySQL database. The saved data can be queried and visualized in a web page as well.

Future work will regard the full implementation of the provenance server with other functions and the support to at least three last versions of the IFC standard schema. We will try to make it possible to query the provenance repository in more sophisticated ways. We would like to produce web centered system for saving additional data as well, along the already implemented functions extracting and saving IFC provenance data.

REFERENCES

Berners-Lee, T. 1997. "Realising the Full Potential of the Web."

Buneman, P., S. Khanna & W. C. Tan 2000. Data Provenance: Some Basic Issues. Proceedings of the *20th Conference on Foundations of Software Technology and Theoretical Computer Science*, Springer-Verlag, pages: 87-93.

Buneman, P., S. Khanna & W.-C. Tan 2001. Why and Where: A Characterization of Data Provenance. *International Conference on Database Theory (ICDT)*.

Greenwood, M., C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau & T. Oinn 2003. Provenance of e-Science Experiments - experience from Bioinformatics. Proceedings of the *UK OST e-Science second All Hands Meeting 2003 (AHM'03)*. Nottingham, UK: 223--226.

Groth, P., L. Moreau & M. Luck 2004. Formalizing a protocol for recording provenance in Grids. Proceedings of the *UK OST e-Science Third All Hands Meeting 2004 (AHM'04)*. Nottingham, UK.

James D. Myers, Carmen Pancerella, Carina Lansing, Karen L. Schuchardt and B. Didier (2003). Multi-Scale Science: Supporting Emerging Practice with Semantically Derived Provenance. 2nd International Semantic Web Conference, Sanibel Island, Florida, USA.

Szomszor, M. & L. Moreau 2003. Recording and Reasoning over Data Provenance in Web and Grid Services. *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'03)*. Catania, Sicily, Italy. 2888: 603-620.

Zhao, J., C. Goble, M. Greenwood & C. W. a. R. Stevens 2003. Annotating, Linking and Browsing Provenance Logs for e-Science. *1st Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*.

Zhao, J., C. Wroe, C. Goble, R. Stevens, D. Quan & M. Greenwood 2004. Using Semantic Web Technologies for Representing e-Science Provenance. *3rd International Semantic Web Conference (ISWC2004)*.

Construction Informatics Digital Library http://itc.scix.net