UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science
Intelligence, Agents, Multimedia Group and
Dependable Systems and Software Engineering Group

Nine month progress report submitted for continuation towards a PhD

Supervisor: Hugh Glaser
Supervisor: Dr Les Carr
Examiner: Dr Harith Alani

Towards a Canonical Method to Solve Patterns of Ontology Modeling Issues

by Benedicto Rodriguez

December 7th, 2006

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science
Intelligence, Agents, Multimedia Group and
Dependable Systems and Software Engineering Group

Nine month progress report submitted for continuation towards a PhD

by Benedicto Rodriguez

This report presents a brief description of the different activities carried out in the field of ontology engineering. It identifies a lack of guidelines on how to address modeling issues during the ontology conceptualization phase, in the current methodologies to build ontologies from scratch. It describes an example scenario of an ontology modeling task and it proposes a possible solution inspired by folksonomy based systems and faceted classification. This is followed by a study of the difficulties found to adapt the IEEE Learning Object Metadata (LOM) standard to model an ontology fit for purpose in a specific university curricula domain. It also gives an example of a prototype for a potential next generation semantic web application and a brief summary of the main viewpoints that will characterize such applications. Finally it outlines possible paths of further research to address ontology modeling issues and it suggests looking at various sources for possible solutions (schemes of folksonomy and faceted classification, design principles of object-oriented and relational database applications, and ontology evaluation).

Table of Contents

1. Introduction

The original idea at the beginning of this PhD program was to evaluate and understand the current state of knowledge technologies, specifically ontology engineering, and assess if these technologies could be applied to the field of software engineering to assist development teams in improving the quality of the artifacts delivered along the software development process.

The evaluation of the current state of ontology engineering, uncovered new areas of research interest linked to some of the difficulties encountered when following any of the methodologies available to create ontologies.

One of such problems was laying out the initial ontology model once the glossary of terms pertaining to the domain knowledge was already available. Ontology creation methodologies provide some guidelines on how to approach this design step however they do not seem to provide enough level of detail on how to address certain modeling issues. This deficiency could lead ontology designers into making incorrect modeling choices having to rely on a subjective interpretation of the problem.

The aim of this research is to explore this aspect of the ontology creation process in depth and try to propose better guidelines that could assist ontologists in solving specific design issues in a more deterministic, reproducible and objective manner.

The content of this report is organized as follows: Section 2 presents a quick overview of the main concepts in ontology engineering in general. Section 3 describes the work undertaken from the beginning of this program relevant to the area of ontology modeling and finally, Section 4 outlines the conclusions gathered from this work and the possible lines of further investigation.

2. The Problem

Ontologies have emerged as one of the key components needed for the realization of the Semantic Web vision (Berners-Lee et al., 2001) and they bring with them a broad range of development activities that can be grouped into what is called Ontology Engineering.

Ontology Engineering practices present many similarities to those in the Software Engineering field and there have been different adaptations of software engineering principles to the ontology engineering domain (Fernandez-Lopez et al., 1997).

Below is a list of the most common ontology engineering practices and a brief description of the work that each one of them entails (Gòmez-Pérez et al., 2004)(Fernandez-Lopez, 2002)(Fernandez-Lopez et al., 1997). This list is not intended to be exhaustive given that new ontology engineering activities continue to appear as ontologies and the applications they are used for, keep on evolving.

- Requirements specification. Similarly to its Software Engineering counterpart, the main deliverable of this activity is an ontology requirements document.

- Conceptualization. This activity produces a conceptual model of the ontology, starting from a glossary of terms that contains the relevant domain knowledge for the ontology.

- Implementation. It constitutes the actual coding of the ontology into a formal ontology language that is machine-readable, such as the Web Ontology Language (OWL), (Dean and Schreiber 2004).

- Evaluation. This activity could be seen as the Verification and Validation tasks performed in the Software Engineering discipline. The idea is to corroborate that the delivered ontology meets the requirements it was built for.

- Documentation. It is an important task that takes place throughout the ontological engineering process in order to understand the built ontology and enable potential future re-use. However, lack of guidelines on how to generate this documentation has been a challenge for ontologists when undertaking this activity. (Skuce 1995)

- Evolution and maintenance. This practice deals with the repercussions of modifications made to a deployed ontology in the applications and systems that the ontology operates. Management of change

- Extension. In situations when an ontology is re-used, it may be necessary to add new classes, properties, or other functionality to adapt it to new requirements. The process of adding or expanding the capabilities of an ontology is also referred to as ontology extension.

- Specialization or refinement. It could be viewed as the contrary process to ontology extension. In this case, the ontology is subtracted of some of its functionality that is not relevant to meet its requirements.

- Pruning or winnowing. It is characterized by tailoring, simplifying, or shrinking an ontology with respect to the needs of the application that is using it (Ehrig et al., 2004)(Alani et al., 2006).

- Integration. It deals with the question of how and whether to use all or part of ontologies that already exist (Uschold et al., 1996).

- Merging. It examines similarities and differences between source ontologies and it aims to produce a single ontology resulting from the combination of all the sources (Noy and Musen, 2000).

- Mapping or alignment. Like in the case of ontology merging, ontology mapping also involves looking at links between existing ontologies to make them consistent with one another, although here, the sources involved will be kept separately (Noy and Musen, 2000).

- Reasoning. This activity deals with the study of the inferring capabilities of the produced ontology.

- Modularization. (Alan Rector)

Out of all these aspects of ontology engineering, this report is primarily focussed in the ontology conceptualization task described above, and on the potential opportunities for improvement in the current state of the art. The idea is to study ontology modelling problems at a specific point in the conceptualization process.

The first part of the conceptualization phase is to develop a glossary of terms representative of the domain knowledge obtained during the preceding knowledge acquisition phase. At this point, the construction of the model for the ontology starts and it is at this point that ontologists will have to solve different modelling issues to convert the glossary of terms into an ontology model. For example, what terms in the glossary should be modelled as classes? What terms should become properties, property values, or instances? This is the specific step in the conceptualization phase that this research is intended to focus on.

The methodologies that address the creation of ontologies from scratch do not provide enough information at the right level of detail about this specific step of the ontology conceptualization phase (Gòmez-Pérez et al., 2004)(Fernandez-Lopez et al., 2002)(Uschold and King, 1995)(Gruninger and Fox, 1995). They look at this activity in broader terms, from a higher level perspective, or from the point of view of what role in the overall ontology engineering lifecycle it plays and what dependencies it has with other engineering activities. Different methodologies provide different levels of detail on how ontology conceptualization should be performed, but none of them discuss in depth the possible modelling problems stated earlier (or their solutions), that ontologists may be faced with, at that specific point of the ontology creation process.

Two examples of previous work that examine ontology modelling issues in the context demanded by this research, can be found in (Noy, 2004) and in (Noy and McGuinness, 2001), which in turn, bases part of its rationale on the principles of object-oriented modelling (Rambaugh et al., 1991).

Lastly, in the next section it is introduced a sample case scenario of some ontology modelling issues to illustrate the type of problem described here.

3. Work Completed

This section presents different artifacts developed during the timeframe covered by this report. Section 3.1 contains certain ontology modeling issues in connection to the ReSIST project (ReSIST, 2006) and Section 3.2 provides a summary of the main work items developed while attending the Fourth European Summer School on Ontological Engineering.

3.1. The ReSIST Project

A good amount of work completed during this time, took place in the context of the ReSIST project. ReSIST stands for Resilience and Survivability in Information Society Technology (IST) and it is a Network of Excellence (NoE) project funded under the Sixth Framework Programme of the European Union (ReSIST, 2006).

One of the objectives of the ReSIST project is to create a Knowledge Base (KB) application in the domain of resilient computing, partly inspired by the features demonstrated by the semantic web application CS AKTive Space (Glaser et al., 2004)(Shadbolt et al., 2004) and with many of the same requirements.

The aim of the ReSIST Knowledge Base (RKB) is to provide an application to the end-user that could serve as a portal to browse and search all type of information in the field of resilient computing: projects, people, institutions, publications, communities of practice, courses, etc.

All the information that constitutes the ReSIST KB is stored as Resource Description Framework (RDF) data that is generated in accordance to different ontologies required to equip the KB of semantic capabilities (Manola and Miller, 2004). The mentioned ontologies are provided in the Web Ontology Language (OWL) format (Dean and Schreiber 2004).

Further information of the main components and technologies being used to develop the ReSIST KB application can be found in (Millard et al., 2006).

3.1.1. The Ontologies in the ReSIST Knowledge Base

A specific aspect that is highly relevant for this report is the different ontologies being used in the ReSIST KB application and the reasons behind them. This section describes such ontologies.

The first ontology is the AKT Reference Ontology (AKT, 2002), initially developed for the CS AKTive Space application (Glaser et al., 2004)(Shadbolt et al., 2004). This ontology provides a model to represent people, institutions, projects, publications, research interests, etc. The ontology fits very well the requirements of the ReSIST KB application and is being re-used in its entirety.

The second ontology is the ReSIST Ontology which is still being developed at present. This ontology models the domain of resilient computing.

The third ontology is the ReSIST University Curricula Ontology. It models the concept of a university course related to the topic of resilient computing.

The following three sections describe the last two ontologies mentioned and a series of problems encountered during the modelling phase that contributed in situating ontology modelling as the main research focus of this report.

3.1.2. An Ontology for Resilience and Survivability in IST

An exploration of existing ontologies was conducted with the goal of identifying a candidate that could be re-used for our target domain. This task was performed by searching in Swoogle[1] (an indexing and retrieval system of ontologies for the Semantic Web) for key concepts from our target domain such as: resilience, dependable, dependability, fault, fault-tolerant, fault-tolerance, etc. The same searches were issued in Google instructing to check only for ontology files, (by using the engine's filetype:owl web search feature). Both tools retrieved several OWL files, (although in the case of Swoogle the number of results was larger and more relevant), but none of them was a suitable option mainly because they applied to different domains. The closest candidate was an ontology for the ACM Computing Classification System (ACM 2002). However, this classification scheme is intended for the whole field of Computing, and does not provide the sufficient level of detail for our needs.

An observation drawn after going through the experience of searching existing ontologies as described is that it is not very intuitive to understand from the OWL files found, the purpose or domain that they belong to. It's not clear if this is due to the lack of relevant comments in the files themselves or if something could be done on how these search engines present their results to the user.

At the view of the results, it was concluded that the ontology for ReSIST will have to be built from scratch.

The first step when building an ontology from scratch is to acquire the knowledge of the domain to model (Noy and McGuinness, 2001)(Fernandez-Lopez et al., 1997)(Uschold and King, 1995)(Gruninger and Fox, 1995). Usually this process requires contacting experts in the domain that can facilitate this information and can be very time-consuming. For our purpose it was agreed to use the terms defined in a relevant key publication in the field as the initial glossary of terms baseline to model in the ReSIST ontology (Avizienis et al., 2005). This would alleviate dramatically the knowledge acquisition task.

1 http://swoogle.umbc.edu/

Very soon, important design challenges started to emerge when attempting to model some of the key concepts outlined in the stated publication. One of the concepts that generated the most modelling issues was, and still is, the concept of “fault”.

Figures 1, 2 and 3 illustrate the taxonomy for the concept of “fault” that should ideally be captured by the ReSIST ontology. The background rationale of the figures in the context of dependable and secure computing can be further studied in (Avizienis et al., 2005).

As Figure 1 shows, the first level of the tree diagram is referred to as the eight basic viewpoints that lead to the elementary fault classes presented in the second level of the tree.

And as the paper also notes, from Figure 3 it could be inferred that if all combinations of these 8 elementary classes were possible, the total number of combined fault classes would be 256. However not all combination occur in reality and Figure 2 and 3 illustrate the 31 most likely combined fault classes as a tree and a matrix representation respectively.

As can be seen the amount and complexity of overlapping information being conveyed by these figures goes beyond that found in simple taxonomies and here is where the ontology modelling challenges begin. Several design questions arose:

- *What concepts should be classes versus properties?*
- *How many classes would be too many?*
- *Should all the information conveyed by the charts be captured in the ontology model?*

According to (Noy and McGuinness, 2001) what determines if certain concepts should be modelled as classes or properties is how important those concepts are in the domain being modelled.

To illustrate this guideline in terms of Figure 3, the Fault class could be modelled having 31 sub-classes, one for each type of combination of fault, or having a property “type-of-combination-fault” that could be set to a value in the range {1, ..., 31}.

In the case of ReSIST, from a domain point of view, all different types of faults are relevant. This fact would favour having every type of fault as a class. However, from an application point of view, it is not very likely that a person, a project, or a publication may describe its research interests in terms of Figure 3 as “Faults 5, 6 and 25” for example. Therefore, including all 31 types of faults in the model does not seem practical.

Regarding how many classes would be too many, (Noy and McGuinness, 2001) concludes that the model should not include a separate class for each distinction of the concept being considered (faults in our case), but the goal should be striking a balance between creating new classes for the purpose of class organization and creating too many classes.

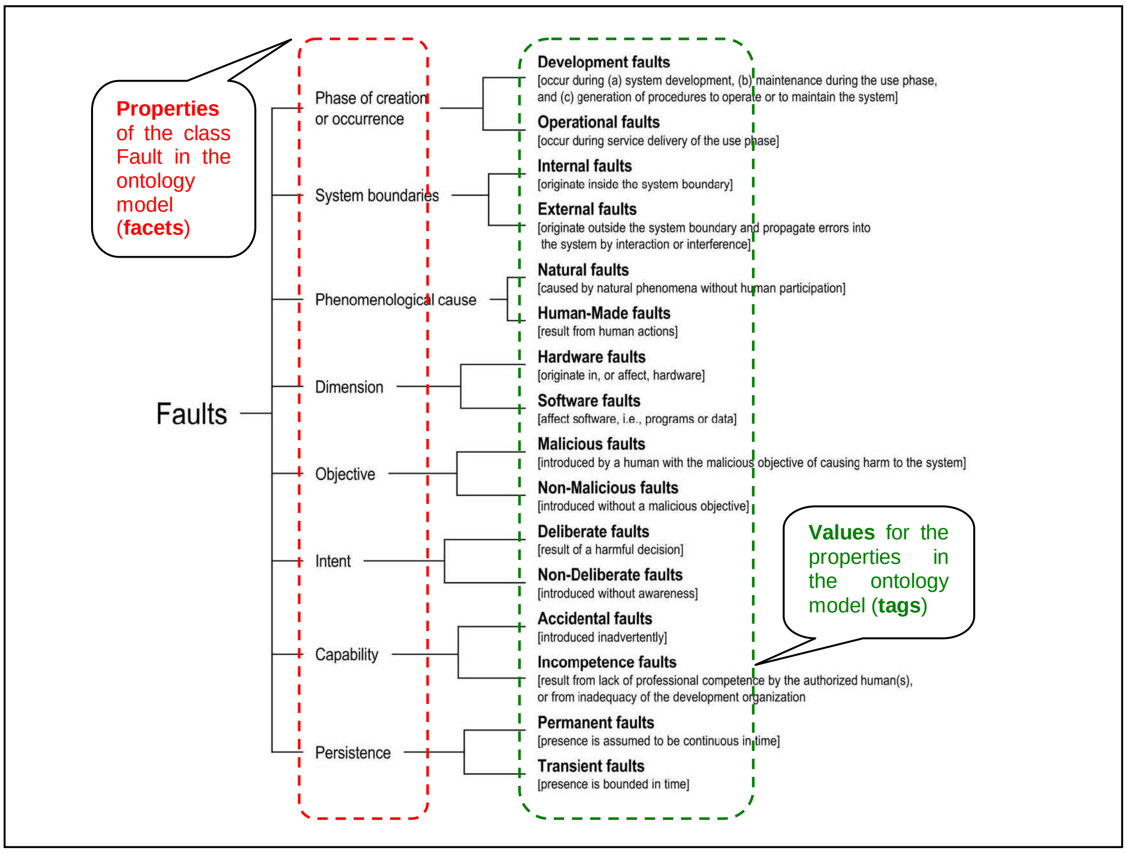

Properties of the class Fault in the ontology model (facets)
Faults
Phase of creation or occurrence
System boundaries
Phenomenological cause
Dimension
Objective
Intent
Capability
Persistence
Development faults
[occur during (a) system development, (b) maintenance during the use phase, and (c) generation of procedures to operate or to maintain the system]
Operational faults
[occur during service delivery of the use phase]
Internal faults
[originate inside the system boundary]
External faults
[originate outside the system boundary and propagate errors into the system by interaction or interference]
Natural faults
[caused by natural phenomena without human participation]
Human-Made faults
[result from human actions]
Hardware faults
[originate in, or affect, hardware]
Software faults
[affect software, i.e., programs or data]
Malicious faults
[introduced by a human with the malicious objective of causing harm to the system]
Non-Malicious faults
[introduced without a malicious objective]
Deliberate faults
[result of a harmful decision]
Non-Deliberate faults
[introduced without awareness]
Accidental faults
[introduced inadvertently]
Incompetence faults
[result from lack of professional competence by the authorized human(s), or from inadequacy of the development organization
Permanent faults
[presence is assumed to be continuous in time]
Transient faults
[presence is bounded in time]
Values for the properties in the ontology model (tags)

Figure 1 - The elementary fault classes.

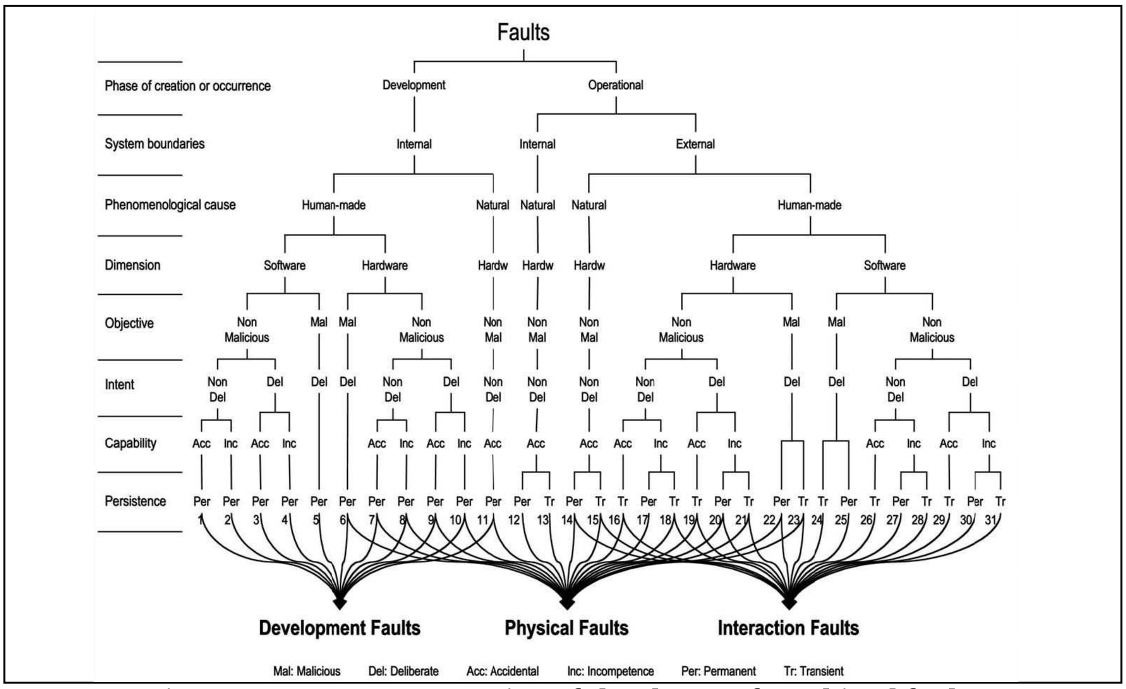


Figure 2 - Tree representation of the classes of combined faults.

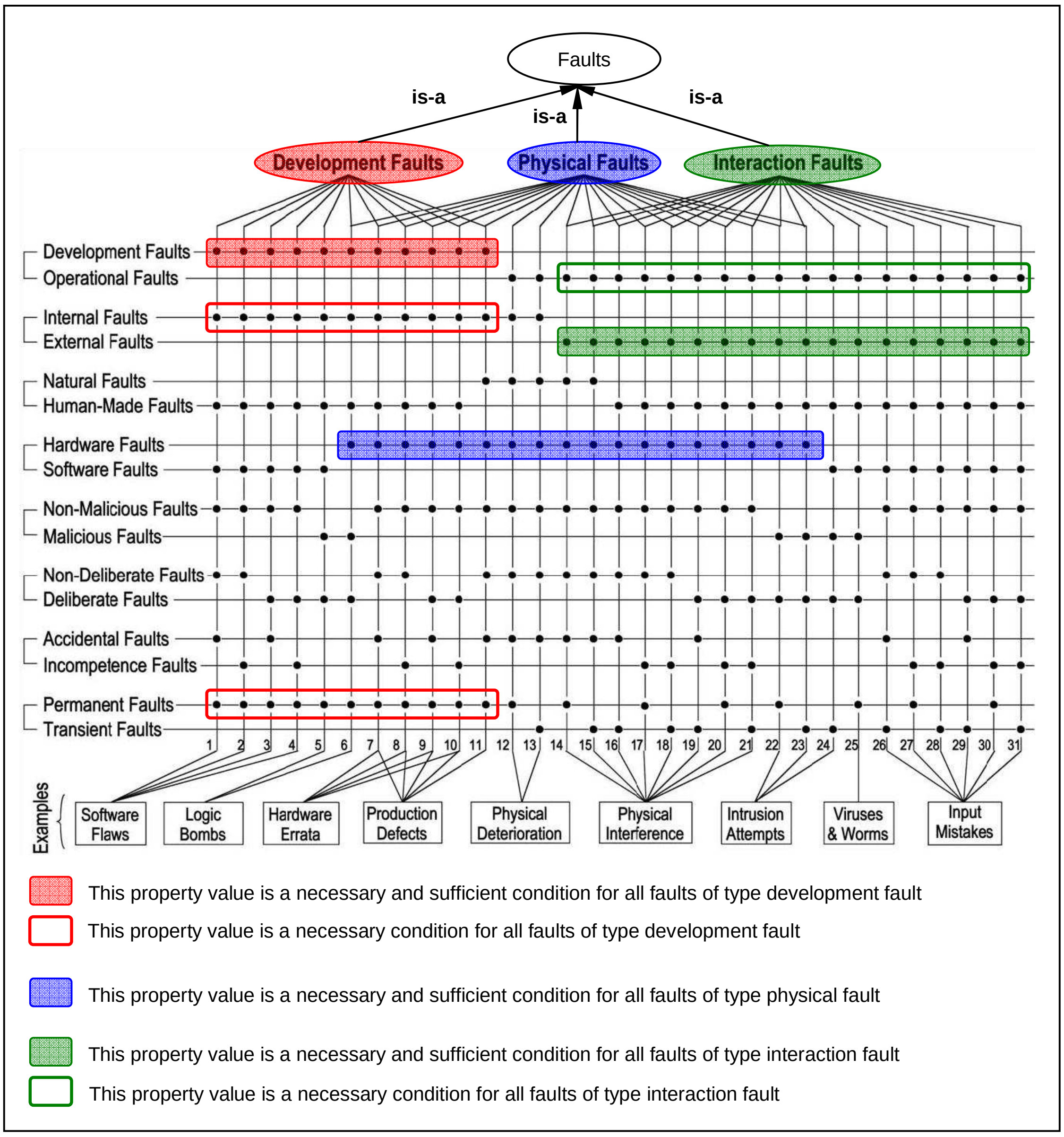


Figure 3 - Matrix representation of the classes of combined faults.

For ReSIST, and again with application use in mind, it would seem unnecessary to create classes for all types of faults, provided that users of the knowledge base may never populate many of those classes with any instances, simply because they may not describe their research interest in those terms.

For this reason, ontology use, together with the notion of building an ontology fit for purpose, it would not deem appropriate attempting to model all the information conveyed by Figures 1 to 3, in terms of one class for every type of fault, with the additional need to provide axioms, domain and range information for each class. This addresses the question of how much information to model. Selecting an adequate level of detail to model would prevent the need to prune or winnow the ontology later after deployment (Alani et al., 2006).

An additional disadvantage of modelling all the information portrayed in Figures 1 to 3 is that if new types of faults appear over time, the ontology would not be up-to-date.

This is true for any domain that is modelled by an ontology. If the reality of the target domain changes the ontology would have to change accordingly. This limitation in the validity of ontologies over time is one of the main arguments in (Shirky, 2005), who question the value of predetermined taxonomies in favour of more flexible and maintainable structures, based on user-created tags as the main mechanism for metadata annotation. Such annotation paradigm has also been coined with the neologism "**folksonomy**". A growing number of internet sites are adopting this increasingly popular tag or keyword based approach to provide certain level of semantic information to their contents. Two prominent examples of such sites are "del.icio.us"[2] and "Flikr"[3].

3.1.3. An Ontology Model for the Concept of Fault

With all this information at hand, what follows is a proposed model for the concept of Fault in the context of the ReSIST KB that attempts to find a compromise between all the factors discussed.

The proposal presents an ontology that initially is fairly simple, yet, it provides sufficient semantic expressivity to represent instances of any type of fault from Figures 1, 2, and 3.

It is inspired by two different notions of viewing concepts in a particular domain. One of them is the previously cited "folksonomy" and the reasons behind it are founded in some of the conclusions of (Shirky, 2005). The second one is based on a *facetted approach* to building ontologies (Pietro-Diaz, 2003) which seemed very appropriate for the challenges in our design.

The initial model is quite basic. It consists of the class "Fault" and it subsumes three other subclasses: "Development-Fault", "Physical-Fault", and "Interaction-Fault". As can be seen from Figure 3, all 31 combined fault classes fall into one or two of these three subclasses. Therefore, so far it is a simple way of representing all faults that are being modelled.

A closer look at the matrix representation in Figure 3 shows eight rows corresponding to the eight viewpoints (or "**facets**") in the "Fault" taxonomy (Figure 1), in terms of a pair of mutually exclusive attribute values for each viewpoint, and 31 columns corresponding to the most likely combined fault classes.

The definition of each type of fault (column) can be given by the 8-tuple of attribute values that apply to that particular fault.

[2] http://del.icio.us/

[3] http://www.flickr.com/

```
Fault-Type-i = TUPLE[ viewpoint1: {value1 | value2},
                      viewpoint2: {value1 | value2},
                      ...,
                      viewpoint8: {value1 | value2} ]
```

These eight pairs of mutually exclusive attribute values could be seen as a closed vocabulary of allowed "**tags**" to define a specific type of fault, where each value pair belongs to one of the eight fault viewpoints. These eight viewpoints could be seen as the eight facets of the concept "Fault". For example, the fault type corresponding to column labelled "1" could be defined as:

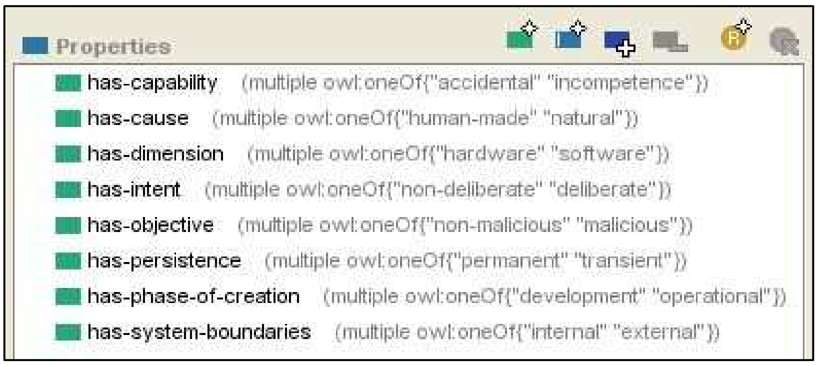
```
Fault-Type-1 = TUPLE[ phase-of-creation:        {development},
                      system-boundary:          {internal},
                      phenomenological-cause:   {human-made},
                      dimension:                {software},
                      objective:                {non-malicious},
                      intent:                   {non-deliberate},
                      capability:               {accidental},
                      persistence:              {permanent}      ]
```

The rest of faults represented in Figure 3 from column "2" to "31" could be represented similarly setting each property to the corresponding value in the 8-tuple.

Furthermore, this definition of a fault class as a function of its 8-tuple of atomic values, allows with the help of basic algebraic constructs, to define any combination or clustering of faults (multiple views of faults could be generated). For example, Figure 3 defines "Logic Bombs" as the faults that belong to either fault type column "5" or fault type column "6". In other words, "Logic Bombs" fault instances could be identified as:

```
Logic-Bombs = Fault-Type-5(8-tuple) ∪ Fault-Type-6(8-tuple)
```

And by the same principle, the rest of named fault classes in Figure 3 could also be characterized: "Software Flaws", "Hardware Errata", "Production Defects", etc.



has-capability (multiple owl:oneOf{"accidental" "incompetence"})
has-cause (multiple owl:oneOf{"human-made" "natural"})
has-dimension (multiple owl:oneOf{"hardware" "software"})
has-intent (multiple owl:oneOf{"non-deliberate" "deliberate"})
has-objective (multiple owl:oneOf{"non-malicious" "malicious"})
has-persistence (multiple owl:oneOf{"permanent" "transient"})
has-phase-of-creation (multiple owl:oneOf{"development" "operational"})
has-system-boundaries (multiple owl:oneOf{"internal" "external"})

Figure 4 - OWL properties of the Fault class.

Figure 4 illustrates how this design for the concept of "Fault" can be brought into the ReSIST KB by giving a graphical view of a partial OWL implementation using the Protégé OWL ontology editor (Horridge et al. 2004). The eight different fault viewpoints (facets) are modelled as OWL Datatype properties and each of the properties can be set to a single enumerated value (tag) from a mutually exclusive pair via "`owl:oneOf`".

To continue grounding the example of faults represented by column "1" in Figure 3 in the context of the ReSIST KB, below is an example of the SPARQL (Prudhommeaux and Seaborne, 2005) syntax that would be required to retrieve all instances of such fault type.

```
PREFIX resist: <http://www.resist-noe.org/ontology/resist#>
SELECT ?fault
WHERE
{
   ?fault resist:has-phase-of-creation “development” .
   ?fault resist:has-system-boundaries “internal” .
   ?fault resist:has-cause             “human-made” .
   ?fault resist:has-dimension         “software” .
   ?fault resist:has-objective         “non-malicious” .
   ?fault resist:has-intent            “non-deliberate” .
   ?fault resist:has-capability        “accidental” .
   ?fault resist:has-persistence       “permanent” .
}
```

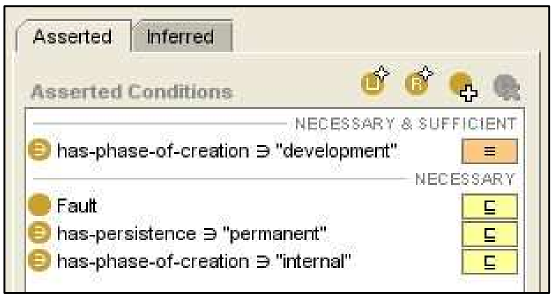
The SPARQL language also allows joining multiple result-sets via the UNION operator. This means that the earlier example regarding "Logic Bombs" could also be rewritten in terms of a SPARQL query to retrieve the union of all instances of the fault represented by column "5" and column "6" from Figure 3 as follows:

```
PREFIX resist: <http://www.resist-noe.org/ontology/resist#>
SELECT ?fault
WHERE
{
      {
         ?fault resist:has-phase-of-creation “development” .
         ?fault resist:has-system-boundaries “internal” .
         ?fault resist:has-cause             “human-made” .
         ?fault resist:has-dimension         “software” .
         ?fault resist:has-objective         “malicious” .
         ?fault resist:has-intent            “deliberate” .
      ###
      ### no value for property ?fault resist:has-capability
      ###
         ?fault resist:has-persistence       “permanent” .
      }
      UNION
      {
         ?fault resist:has-phase-of-creation “development” .
         ?fault resist:has-system-boundaries “internal” .
         ?fault resist:has-cause             “human-made” .
         ?fault resist:has-dimension         “hardware” .
         ?fault resist:has-objective         “malicious” .
         ?fault resist:has-intent            “deliberate” .
      ###
      ### no value for property ?fault resist:has-capability
      ###
         ?fault resist:has-persistence       “permanent” .
      }
}
```

Using the same logic, a corresponding query could be constructed to retrieve any combination of fault types from Figure 3. However this approach introduces one caveat. The performance of the UNION operator in the current draft of the SPARQL definition is far from optimal when applied over large data repositories of RDF triples. In practical terms, an alternative to avoid this limitation would be performing the necessary queries separately, and use additional software logic to combine the individual results. The processing time overhead of the software might still prove more efficient than using the UNION operator.

Nonetheless, from an ontology modelling perspective, the underlying principle that is being put forward, is that the proposed approach to model the multidimensional concept of "Fault" is capable of representing and retrieving any individual fault type as well as any combination or clustering of them, allowing looking at the concept of "Fault" and its instances from any of its overlapping viewpoints or facets.

There is another important characteristic found in the matrix representation of faults in Figure 3 that might be worth noting because it illustrates the ontological concepts of "*necessary*" and "*necessary and sufficient*" conditions and it ties together the selection of classes and properties described for the proposed ontology here.

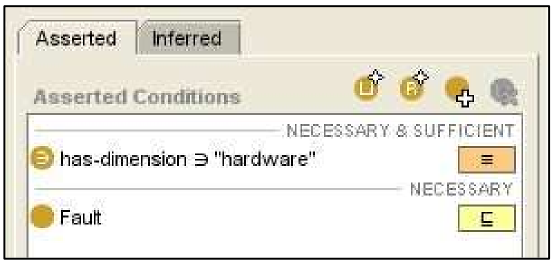

Asserted | Inferred

Asserted Conditions

NECESSARY & SUFFICIENT
has-phase-of-creation ∋ "development" ≡

NECESSARY
Fault ⊑
has-persistence ∋ "permanent" ⊑
has-phase-of-creation ∋ "internal" ⊑

(a) Development Faults

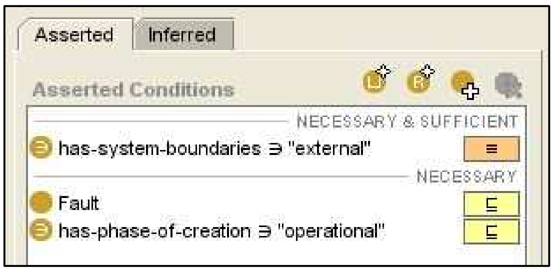

Asserted | Inferred

Asserted Conditions

NECESSARY & SUFFICIENT
has-dimension ∋ "hardware" ≡

NECESSARY
Fault ⊑

(b) Physical Faults


Asserted | Inferred

Asserted Conditions

NECESSARY & SUFFICIENT
has-system-boundaries ∋ "external" ≡

NECESSARY
Fault ⊑
has-phase-of-creation ∋ "operational" ⊑

(c) Interaction Faults

Figure 5 - Conditions in the ontology model for: development faults (a), physical faults (b) and interaction faults (c).

Looking at the concept of "Physical Fault" in Figure 3 for example, it can be seen that all faults that belong to this category has in common that the value for the "Dimension" viewpoint is set to "Hardware Fault" and vice versa. If a fault is of type "Hardware Fault" for its "Dimension" facet then it belongs in the category "Physical Fault". (Note in Figure 3 the solid blue round box along the row labelled "Hardware Faults"). This implication both ways represents a "*necessary and sufficient*" condition for all instances of the class "Physical-Fault" in our ontology with respect to the value of the property "has-dimension", and its graphical representation in Protégé is shown in Figure 5(b).

The same rationale applies to the concept of "Development Faults" and "Interaction Faults" in Figure 3. The solid boxes indicate "*necessary and sufficient*" conditions, while the hollow boxes indicate just "*necessary*" conditions for instances of those classes. Figure 5(a) and 5(b) respectively, shows the graphical representation of these conditions in the Protégé ontology editor. An overview of these two ontological concepts can be found in (Horridge et al. 2004).

In conclusion, the technique described of using OWL Datatype property values as "tags", avoids imposing a fault classification overly complicated and specialized onto the ontology application, *shifting the focus from* having to decide where in the highly coupled taxonomy a fault instance should be classified, *into* selecting the suitable property values for that instance instead.

In general terms this could be seen as an attempt to create an ontology that is more agnostic of a pre-established taxonomy and more oriented to a *user-developed classification* ("folksonomy") without compromising the semantic expressivity of the target data, given that all fault categories can still be represented by virtue of selecting the applicable property values.

Although strictly speaking, the approach conforms more to a facet classification scheme (Pietro-Diaz, 2003) given that the property values conforms to a closed vocabulary of selected terms rather than being entirely user-defined as it is actually the case in a "folksonomy".

3.1.4. An Ontology for ReSIST University Curricula

Another objective of the ReSIST project, to be carried out by its Training and Dissemination working group, is to promote and propagate a resilience culture in university curricula and in engineering best practices (ReSIST, 2006). To support this objective the ReSIST Knowledge Base would devote a section of its application to the management of information regarding university courses in the subject of resilience. This section describes much of the work completed regarding the development of different aspects of this application that is relevant to this report.

The initial source of requirements for the ReSIST University Curricula knowledge base application was a draft template document describing the information to capture about a university course produced by the Training and Dissemination committee in ReSIST.

To make this template part of the overall ReSIST knowledge base, it was necessary to expand the main ontology with the glossary terms and properties captured in the template.

Once again, as described in section 3.1.2., before attempting to build this new "ReSIST Course" ontology from scratch, a quick survey was conducted to re-use an existing one that could deem suitable.

None of the ontologies found, implemented as OWL files, seemed fit for purpose because despite containing some references to the terms searched (course, syllabus, curriculum, etc.), overall, they applied to different domains. However, in this case a promising resource was identified. It consisted of an RDF binding for the Learning Object Metadata (LOM) standard produced within the Learning Technology Standard Committee of the IEEE (IEEE 2002).

The LOM standard is a conceptual model to describe a metadata instance of a learning object, which in turn the standard defines as any entity (digital or non-digital) that may be used for learning, education or training. Our ReSIST courses seemed to fit well this definition. The LOM standard does not impose any technical implementation of its conceptual scheme (binding). It only defines the metadata elements that describe a learning object with the idea of enabling semantic interoperability for the different applications that may decide to support it. Figure 6 provides a schematic view of the more than 70 metadata attributes that integrate LOM. Further information about each element can be found in (IEEE 2000).

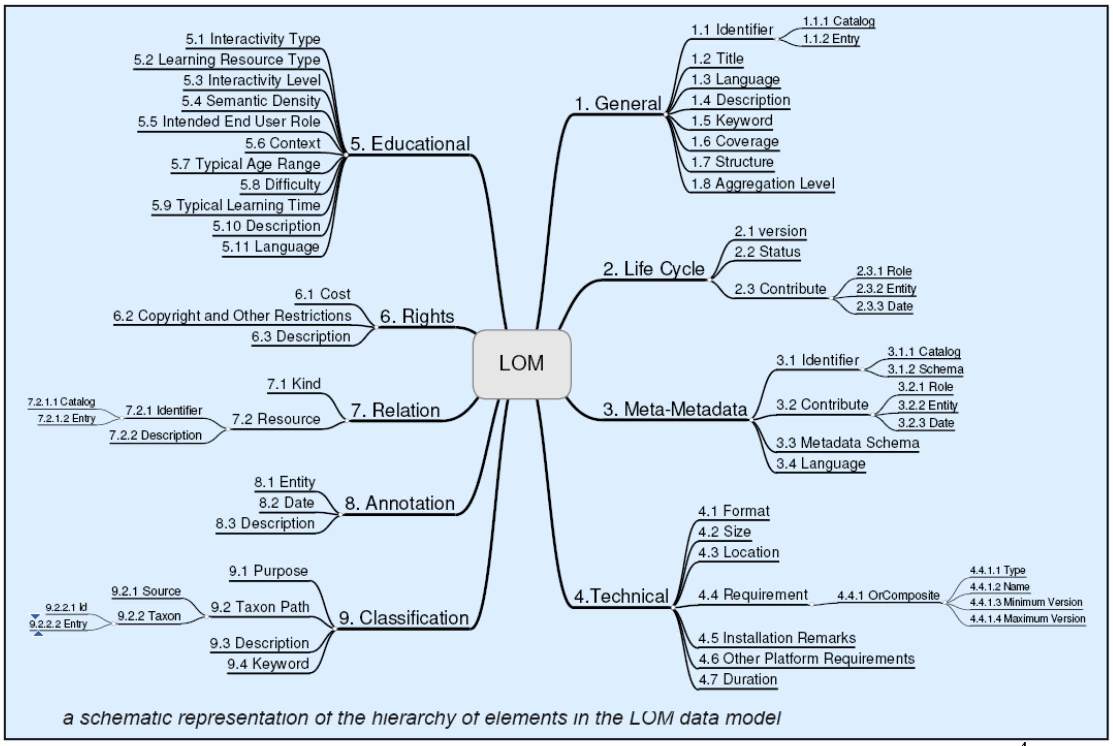


Figure 6 – A schematic representation of the LOM element hierarchy[4]

A quick look at the history of the RDF binding of the LOM standard shows that the implementation was initially developed by the Knowledge Management Research Group of the Royal Institute of Technology in Stockholm, Sweden[5] (Nilsson et al., 2003), leading to its current state where it is being used by a joint taskforce between the IEEE

[4] http://en.wikipedia.org/wiki/Learning_object_metadata/
[5] http://kmr.nada.kth.se/el/ims/metadata.html/

LTSC group and the Dublin Core Metadata Initiative (DCMI) to produce a recommended representation of the LOM elements in the Dublin Core metadata element set[6].

Despite having an RDF binding of the LOM standard available that seemed applicable for the ReSIST Course ontology there were still important obstacles to overcome in order to reuse this resource.

A semantic mapping between the fields presented in the template document for a ReSIST course and the LOM standard was required. Conducting this mapping implied a thorough understanding of the LOM element set and its intended use. This proved a very time-consuming task provided that the LOM standard contains more that 70 metadata elements while the template document to be modelled by our ontology included in the order of 20 fields.

Many of the template fields needed had suitable matches in the LOM model but some others did not seem to have an apparent equivalent element or there were a significant semantic distant with respect to potential candidates. Some of such fields from our requirements template included:

- Number of credits for the course based on the current EU Credit Transfer (ECTS) system guidelines.

- Student interaction type (group or individual homework, projects, lectures, etc.)

- Assessment methods (written examination, laboratory work, attendance, etc.)

- Course pre-requisites in terms of previous knowledge required from a student to successfully follow the course.

- Objectives of the course in terms of learning outcome.

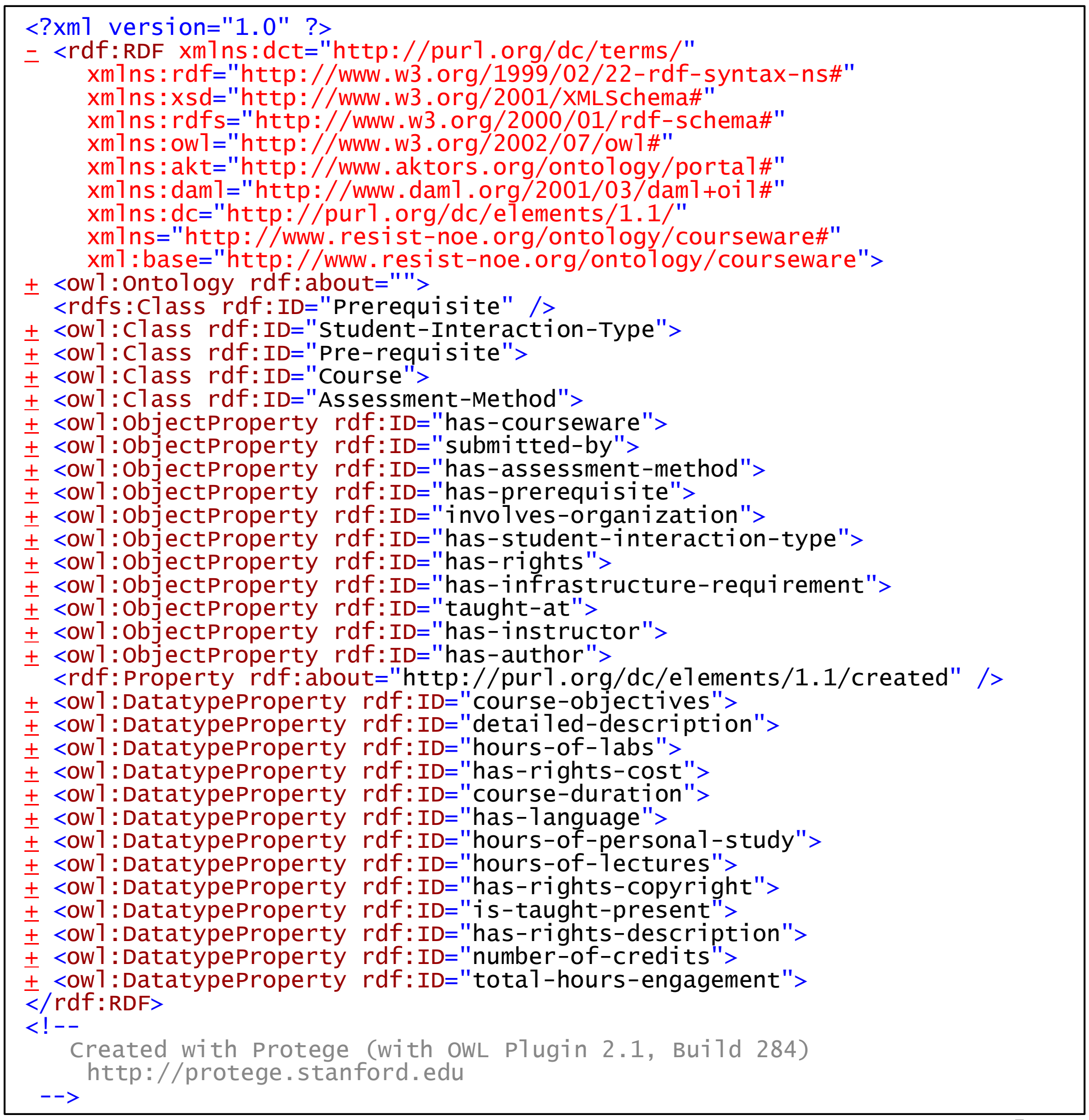
Therefore, reusing the LOM standard for ReSIST would involve, firstly specializing or refining from the more than seventy metadata elements to the subset applicable to ReSIST, and secondly, extending the model again to include those fields in the ReSIST course template that did not present LOM equivalents.

Based on this analysis, it was estimated that the amount of effort needed to re-engineer the LOM RDF binding to be reused for our purpose, would be greater than developing a custom ontology from scratch to model the approximately twenty fields from our ReSIST course template document.

Considering that the amount of time available to produce our ontology deliverable was also fairly limited, it was concluded to create a custom ontology from the ground up that could use the LOM standard as a point of reference.

6 http://dublincore.org/educationwiki/DCMIIEEELTSCTaskforce/

Figure 7 shows the delivered ontology to model the courses of the ReSIST University Curricula application. The root class of this ontology was named “Course”, and it is subsumed by the class “Abstract-Information” in the AKT Portal Ontology.



```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:dct="http://purl.org/dc/terms/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:akt="http://www.aktors.org/ontology/portal#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://www.resist-noe.org/ontology/courseware#"
    xml:base="http://www.resist-noe.org/ontology/courseware">
+ <owl:Ontology rdf:about="">
  <rdfs:Class rdf:ID="Prerequisite" />
+ <owl:Class rdf:ID="Student-Interaction-Type">
+ <owl:Class rdf:ID="Pre-requisite">
+ <owl:Class rdf:ID="Course">
+ <owl:Class rdf:ID="Assessment-Method">
+ <owl:ObjectProperty rdf:ID="has-courseware">
+ <owl:ObjectProperty rdf:ID="submitted-by">
+ <owl:ObjectProperty rdf:ID="has-assessment-method">
+ <owl:ObjectProperty rdf:ID="has-prerequisite">
+ <owl:ObjectProperty rdf:ID="involves-organization">
+ <owl:ObjectProperty rdf:ID="has-student-interaction-type">
+ <owl:ObjectProperty rdf:ID="has-rights">
+ <owl:ObjectProperty rdf:ID="has-infrastructure-requirement">
+ <owl:ObjectProperty rdf:ID="taught-at">
+ <owl:ObjectProperty rdf:ID="has-instructor">
+ <owl:ObjectProperty rdf:ID="has-author">
  <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/created" />
+ <owl:DatatypeProperty rdf:ID="course-objectives">
+ <owl:DatatypeProperty rdf:ID="detailed-description">
+ <owl:DatatypeProperty rdf:ID="hours-of-labs">
+ <owl:DatatypeProperty rdf:ID="has-rights-cost">
+ <owl:DatatypeProperty rdf:ID="course-duration">
+ <owl:DatatypeProperty rdf:ID="has-language">
+ <owl:DatatypeProperty rdf:ID="hours-of-personal-study">
+ <owl:DatatypeProperty rdf:ID="hours-of-lectures">
+ <owl:DatatypeProperty rdf:ID="has-rights-copyright">
+ <owl:DatatypeProperty rdf:ID="is-taught-present">
+ <owl:DatatypeProperty rdf:ID="has-rights-description">
+ <owl:DatatypeProperty rdf:ID="number-of-credits">
+ <owl:DatatypeProperty rdf:ID="total-hours-engagement">
</rdf:RDF>
<!--
   Created with Protege (with OWL Plugin 2.1, Build 284)
    http://protege.stanford.edu
 -->
```

Figure 7 – Main elements of the ReSIST University Curricula Ontology[7]

The actual end-user ReSIST University Curricula application integrated into the overall ReSIST Knowledge Base, that uses this ontology, was developed by Ian Millard (Millard et al., 2006). In my role as a postgraduate student, I contributed to the development of the application carrying out different activities such as:

- The creation of the mentioned ontology as it was detailed here.

[7] http://resist.ecs.soton.ac.uk/ontologies/courseware.owl/

- Generating controlled vocabularies to pre-populate some of the classes in the ontology (“Pre-requisite”, “Student-Interaction-Type”, “Assessment-Method”).

- Modifying different scripts to enable support for rights management for courses; distinguishing between author, lecturer and submitter of a course; addition of university instances outside of the ReSIST network of partners; and various other miscellaneous coding tasks.

- Informing the ReSIST Training and Dissemination Committee of application updates and releases.

Additional information on the functionality of the ReSIST University Curricula application and how it is being used as a knowledge acquisition tool for users that are not familiar with knowledge technologies can be found in (Millard et al., 2006).

Appendix A provides a visual representation of the main PHP files that constitute the ReSIST University Curricula application and their dependencies at the time of this writing. This information proved very useful when changes in the code were needed.

Appendix B presents the view of an instance of a ReSIST course and the information and fields that are recorded for each one of them, similarly to how an end-user would see it on an internet browser.

3.2. Summer School on Ontological Engineering

During the week of the 9th to the 15th of July 2006 I attended the Fourth European Summer School on Ontological Engineering and the Semantic Web in Cercedilla, Madrid, Spain. This was an excellent opportunity to get a broad overview of the different research fronts currently open in the Ontology Engineering field and by extent in the Semantic Web and to meet with some of the principal investigators and fellow students of these subjects.

Some of the activities involved a poster presentation to be held at the beginning of the School and the presentation of a project proposal for a semantic web application as the event closing. This project would be developed in groups of at least four students and the application should make use of the concepts presented throughout the School.

Appendix C shows my poster participation, which attempts to capture how the different phases of the Knowledge Management lifecycle (acquisition, modelling, retrieval, publishing, and maintenance) are represented by the different components of the ReSIST Knowledge Base application, while giving a high-level view of its architecture (ReSIST, 2006).

On the other hand, the group project, titled “Perfect Lifestyle Seeker”, presented a semantic web application that aimed to find suitable lifestyle-related services (places to live, to work, to travel), by understanding lifestyle information about the end-user. A user lifestyle profile would be automatically constructed acquiring data from various sources such as emails, pictures, travel information, online purchases, etc. This profile would become the point of reference for the application to present relevant and semantically compatible lifestyle-related services when requested by the user. Figure 8 shows a high-level diagram of the proposed architecture as it was given during the group project presentation session of the School.

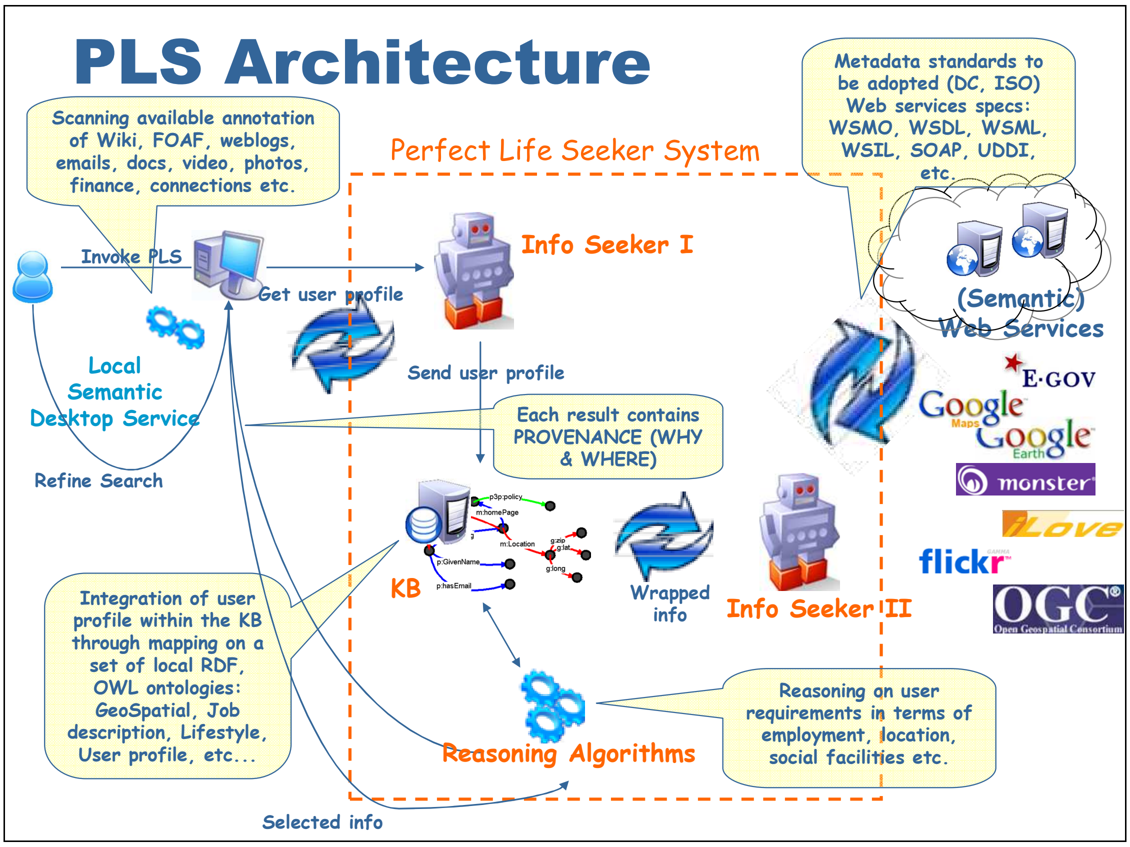

PLS Architecture
Scanning available annotation of Wiki, FOAF, weblogs, emails, docs, video, photos, finance, connections etc.
Perfect Life Seeker System
Metadata standards to be adopted (DC, ISO) Web services specs: WSMO, WSDL, WSML, WSIL, SOAP, UDDI, etc.
Invoke PLS
Get user profile
Info Seeker I
(Semantic) Web Services
Local Semantic Desktop Service
Send user profile
Each result contains PROVENANCE (WHY & WHERE)
Refine Search
E·GOV
Google Maps
Google Earth
monster
iLove
flickr
OGC Open Geospatial Consortium
KB
Wrapped info
Info Seeker II
Integration of user profile within the KB through mapping on a set of local RDF, OWL ontologies: GeoSpatial, Job description, Lifestyle, User profile, etc...
Reasoning on user requirements in terms of employment, location, social facilities etc.
Reasoning Algorithms
Selected info

Figure 8 - Group Project. Semantic Web Application.

The application assumes many technologies that in some cases would still have to be created, or in others would still have to be matured or developed further. But one of the pre-requisites that particularly stands out for its importance among the rest is **trust**. Applications such as the one described here would not be able to succeed if prospective end-users don’t trust them. In order for end-users to exchange and grant a computer program access to sensitive personal information, convincing evidence would have to be exhibited to guarantee that their personal information is safe and secured.

One of the resources that by coincidence became commonplace across all group projects was an evaluation framework to characterize the next generation of semantic web applications (Motta and Sabou, 2006). This framework would evaluate a semantic web application from seven different viewpoints (six at the time of the School) that could be described as:

- Does the application generate its own semantic data or does it reuse existing sources?
- Does it use a single ontology or can it handle multiple ontologies at the same time?
- Is it open to semantic resources? Can it reuse semantic repositories external to the application to handle a request from a user?
- Does the application operate at scale and can it differentiate data quality?
- Is it open to web (non-semantic) resources?
- Does it include features currently attributed to Web 2.0 applications?
- Is it open to web services?

Based on this framework, next generation semantic web applications would be expected to exhibit positively most, if not all, of these stated characteristics.

Finally, there are two key points that I would like to stress, and that in retrospect probably become the "take home" message for me from a research point of view. The first one is having been introduced to the concept of ontology design patterns during Aldo Gangemi's presentation (Gangemi 2005) as a possible source of solutions to some of the problems in ontology design. The second point came from Jim Hendler's[8] presentation closing statement: "Integration, integration, integration [...]"; as a way of highlighting one of the most important keys in his opinion, to unlock the Semantic Web. This is, achieving semantic interoperability across distributed and heterogeneous data repositories.

4. Conclusions and Future Work

The following two sections outline the conclusions gathered as a result of the work presented in this report and identify possible lines of further research.

4.1. Patterns in Ontology Modeling Issues

This report started defining a problem in the methodologies available to build ontologies from scratch. The problem is characterized by a lack of sufficient guidelines in the ontology conceptualization phase to solve certain ontology modeling issues.

Ideally, the ultimate goal of this research would be to define a canonical methodology for the modeling of ontologies starting from a set of well-known ontology requirements and the glossary of terms associated to the target domain. This goal is not possible if we accept the agreed perception that there is no single correct way or method of modeling an ontology as outlined in (Noy and McGuinness, 2001). In that sense, a canonical method to model ontologies might be a goal too broad, because ontology modeling can present an

[8] Jim Hendler is currently a Professor and Director of The Joint Institute for Knowledge Discovery in the University of Maryland, College Park, Maryland, USA.

infinite number of scenarios and use cases. What might be more realistic is to propose a guideline to model a specific **pattern** of ontology modeling issues.

What I intend to explore going forward is the possibility of characterizing the modeling issue described in Section 3.1.3 (with the concept of Fault) as a pattern of ontology modeling issues. The pattern should include the notion of having multiple viewpoints or facets in which the information can be classified by. It is this idea of modeling a concept with multiple facets what I perceive to be the main root-cause for the problems faced with the concept of Fault. To validate the applicability of the pattern it will be necessary to find additional examples from alternative domains with modeling issues that match the proposed blueprint.

The next step would be to recommend a solution to the pattern representing the modeling problem and evaluate it with the examples from other domains. If possible, this solution would be presented as a methodology guideline and would be regarded as the canonical solution to the ontology modeling problem typified by this pattern.

Two activities that could provide assistance in finding potential solutions are the design of relational databases and object-oriented applications. Even though there are fundamental differences between these two disciplines and ontology design, there are certain points of correlation in the abstraction process that all of them experience in order to go from the natural language semantic domain to the computational domain.

Part of the motivation to look outside the ontology modeling practice, is because the nature of some of the issues in ontology modeling seem to originate more from issues in general modeling, than from issues specific to ontologies.

An interesting exercise would be to compare how the design guidelines in the object-oriented community, in the relational database community, and in the ontology engineering community try to solve the same modeling problem.

Additionally, and still in connection with the field of databases, I intend to examine aspects of relational database normalization theory (Connolly and Begg, 1998) that could be extrapolated to ontology models. Database normalization is a formal method to assist database designers identifying issues in relational database schema models that could lead to undesired dependencies in the data to be stored.

Finally, another area that can help guiding the ontology modeling process relates to ontology evaluation tools. Ontology evaluation provides methodologies to check if the concepts in a given ontology are modelled coherently (Guarino and Welty, 2002). They examine properties such as rigidity, identity and unity, misuses of the subsumption relation, and flag potential issues in the ontology classes and its underlining taxonomy.

4.2. Ontology and Folksonomy Interoperability

This report has also shown how some of the characteristics of “folksonomy” classification schemes can bring new elements of decision into the ontology modeling process, especially when the concepts being modeled present a complex taxonomical structure with a high degree of coupling among its taxonomic units.

This notion of looking at ontologies from a “folksonomy” point of view (or vice versa), or finding possible points of correlation between both paradigms, could lead to interesting lines of research.

The need for ontologies having to interact with “folksonomy” based systems is further sustained by the fact that currently in the Web the number of folksonomy based applications clearly outnumbers the available ontology ones. And while the presence of these folksonomy systems continues growing and they become a reality embraced by large user communities, ontology applications are still scarce, marginally used, and often deployed just as research oriented prototypes. This circumstance might encourage the ontology development community to develop new techniques in favor of the interoperability between ontologies and folksonomies.

In that sense, the ontological engineering long-term goal of heterogeneous ontology based system interoperating seamlessly with each other in the open Semantic Web, might have to go through the short-term or midterm goal of having ontology based applications interacting with “folksonomy” based ones as an intermediate step.

Moreover, and pushing the links between ontologies and “folksonomies”, it might be worth studying the feasibility of a user-developed ontology rather than a taxonomy, where users not only specify metadata tags, but can create classes, properties, instances o even more complex ontological elements such as axioms as well. To a limited extent, semantic wikis (i.e. Semantic MediaWiki[9]) could represent a first step towards the notion of user-developed ontologies given that they allow their users to add semantic relations in their Wiki pages.

In summary, the aim of this research going forward would be to propose improvements in the existing ontology modelling methodologies that could shorten the distance between the present state of the art and the unfeasible canonical modelling methodology killer-app. Some areas outside the ontology engineering field that could offer solutions in this direction have been suggested such as: schemes of faceted and folksonomy classification, design principles of object-oriented and relational database models including database normalization analysis, and ontology evaluation tools.

9 http://meta.wikimedia.org/wiki/Semantic_MediaWiki/

References

ACM (1998) The ACM Computing Classification System. Version valid in 2002, http://www.acm.org/class/1998/

AKT (2002) The AKT reference ontology. http://www.aktors.org/publications/ontology/

Alani H, Harris S, O'Neil B (2006) Winnowing Ontologies based on Application Use. In: Proceedings of 3rd European Semantic Web Conference (ESWC), Budva, Montenegro.

Avizienis A, Laprie JC, Randell B, Landwehr C (2005) Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 1(1):11--33.

Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Scientific American.

Berners-Lee T (1998) Semantic Web Roadmap. World Wide Web Consortium (W3C) http://www.w3.org/DesignIssues/Semantic.html/

Brase J and Nejdl W (2003) Ontologies and Metadata for eLearning. Springer Verlag, pp 579-598.

CETIS (2004) UK Learning Object Metadata Core Draft 0.2. Centre for Educational Technology Interoperability Standards. University of Bolton. Bolton, UK. http://www.cetis.ac.uk/profiles/uklomcore/uklomcore_v0p2_may04.doc/

Connolly T, Begg C (1998) Database Systems: A Practical Approach to Design, Implementation, and Management. 2nd Ed. Addison-Wesley, Harlow, England.

Dean M, Schreiber G, (eds) (2004) OWL Web Ontology Language Reference. W3C Recommendation.

Ehrig M, Gabel T, Haase P, Sure Y, Tempich C, Voelker J (2004) Use Cases. SEKT: Semantically Enabled Knowledge Technologies. IST-2003-506826 Project Deliverable 7.1.1.a.

Fernandez-Lopez M (ed) (2002) A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. OntoWeb IST-2000-29243 Project Deliverable 1.4.

Fernandez-Lopez M, Gomez-Perez A, Juristo N (1997) METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33-40.

Gangemi A (2005) Ontology Design Patterns for Semantic Web Content. Proceedings ISWC 2005, LNCS 3729, pp 262-276.

Glaser H, Alani H, Carr L, Chapman S, Ciravegna F, Dingli A, Gibbins N, Harris S, schraefel, mc, Shadbolt N (2004) CS AKTiveSpace: Building a semantic web application. In: Bussler C, Davies J, Fensel D, Studer R, (eds) ESWS. Volume 3053 of Lecture Notes in Computer Science. Springer, pp 417–432.

Gomez-Perez A, Fernandez-Lopez M, Corcho O (2004) Ontological Engineering. Springer Verlag, London.

Gruninger M, Fox MS (1995) Methodology for the design and evaluation of ontologies. In: Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp 6.1-6.10.

Guarino N, Welty C (2002) Evaluating Ontological Decisions with OntoClean. In: Communications of the ACM, 45 (2) pp 61-65.

Horridge M, Knublauch H, Rector A, Stevens R, Wroe C (2004) Practical Guide To Building OWL Ontologies Using the Protégé-OWL Plugin and CO-ODE Tools. Technical Report, Ed. 1.0, The University Of Manchester.

IEEE (2002). Draft Standard for Learning Object Metadata. Sponsored by the IEEE Learning Technology Standards Committee. IEEE 1484.12.1-2002. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

Kingston J (2001) Ontologies, Multi-Perspective Modelling and Knowledge Auditing. In: Ontologies Workshop at the Second German/Austrian Conference on Artificial Intelligence (KI-2001).

Manola F, Miller E (2004) RDF Primer. W3C Recommendation. http://www.w3.org/TR/rdf-primer/

McGuinness DL (2001) Ontologies come of age. In: Fensel D et al (eds) Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential. MIT Press, Cambridge, MA

Millard I, Jaffri A, Glaser H, Rodriguez B (2006) Using a Semantic MediaWiki to Interact with a Knowledge Based Infrustructure (Poster). Submitted to 15th International Conference on Knowledge Engineering and Knowledge Management. Podebrady, Czech Republic.

Motta E, Sabou M (2006) Next Generation Semantic Web Applications. 1st Asian Semantic Web Conference. Beijing, China.

Nilsson M, Palmer M, Brase J (2003) The LOM RDF binding – principles and implementation. The 3rd Annual Ariadne Conference, 20-21 November 2003, Belgium.

Noy N, (2004) Representing Classes As Property Values on the Semantic Web. W3C Working Draft 21 July 2004. http://www.w3.org/TR/2004/WD-swbp-classes-as-values-20040721/

Noy NF, Klein M (2002) Ontology Evolution: Not the Same as Schema Evolution. In: Stanford Medical Informatics Technical Report SMI-2002-0926. Stanford, California.

Noy N, McGuinness DL, (2001) Ontology development 101: A guide to creating your first ontology. In: Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory. Stanford, California.

Noy NF, Musen MA (2000) PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Rosenbloom P, Kautz HA, Porter B, Dechter R, Sutton R, Mittal V (eds) 17th National Conference on Artificial Intelligence (AAAI'00). Austin, Texas, pp 450-455

Powers S (2003) Practical RDF. O'Reilly & Associates. ISBN 0-596-00263-7.

Prieto-Diaz R (2003) A Faceted Approach to Building Ontologies. In: IEEE International Conference on Information Reuse and Integration. IEEE Computer Society Press, pp. 458-465.

Prudhommeaux E, Seaborne (2005) A SPARQL Query Language for RDF. W3C Working Draft. http://www.w3.org/TR/rdf-sparql-query/

The ReSIST Project (2006) Resilience and Survivability in Information Society Technology (IST). IST 4 026764 NOE. http://www.resist-noe.org/

Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W (1991) Object-oriented modeling and design. Englewood Cliffs, New Jersey. Prentice Hall.

Shadbolt NR, Gibbins N, Glaser H, Harris S, schraefel mc (2004) CS AKTive Space or how we stopped worrying and learned to love the Semantic Web. IEEE Intelligent Systems.

Shirky C (2005) Ontology is Overrated: Categories, Links and Tags. In: [online] Clay Shirky's Writings About the Internet. http://shirky.com/writings/ontology_overrated.html

Spaccapientra S, Parent C, Vangenot C, Cullot N (2004) On Using Conceptual Modeling for Ontologies. In: Proceedings of the Web Information Systems Workshops (WISE 2004 Workshops), Lecture Notes in Computer Science 3307, 22-33.

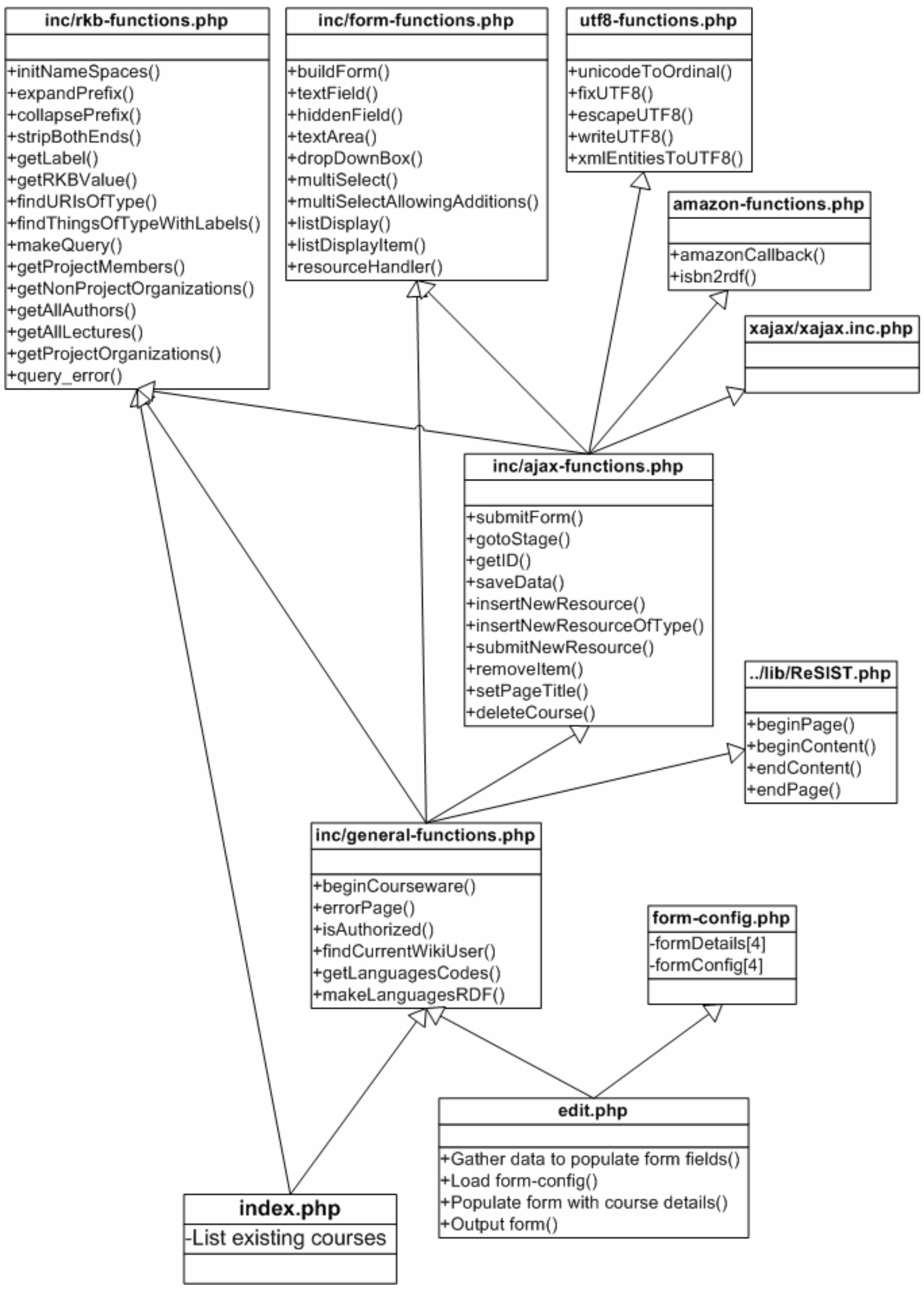
Spyns P, Meersman R, Jarrar M (2002) Data modelling versus ontology engineering. ACM SIGMOD Rec 31(4):12–17.

Uschold M, Gruninger M (1996) Ontologies: Principles, Methods, and Applications. Knowledge Eng. Rev., Vol. 11, No. 2, pp. 93-155.

Uschold M, King M, (1995) Towards a Methodology for Building Ontologies. In: Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp 6.1-6.10.

Appendix A. ReSIST University Curricula Application

Each box represents a PHP file, and it is divided into 3 cells. The top cell displays the name of the file, the middle one displays any significant global variables in the file if any, and the bottom one the most important functions implemented in that file (Loosely analogous to how classes are characterized in object-oriented modelling). The arrows indicate dependency between two files meaning that the file at the origin of the arrow requires or includes the file at the destination of the arrow (Millard et al., 2006).



Appendix B. Example of a ReSIST Course Instance

The example course shown here is also available at: http://resist.ecs.soton.ac.uk/courseware/view/b417613c

RESIST RESILIENCE FOR SURVIVABILITY IN IST
KNOWLEDGE BASE

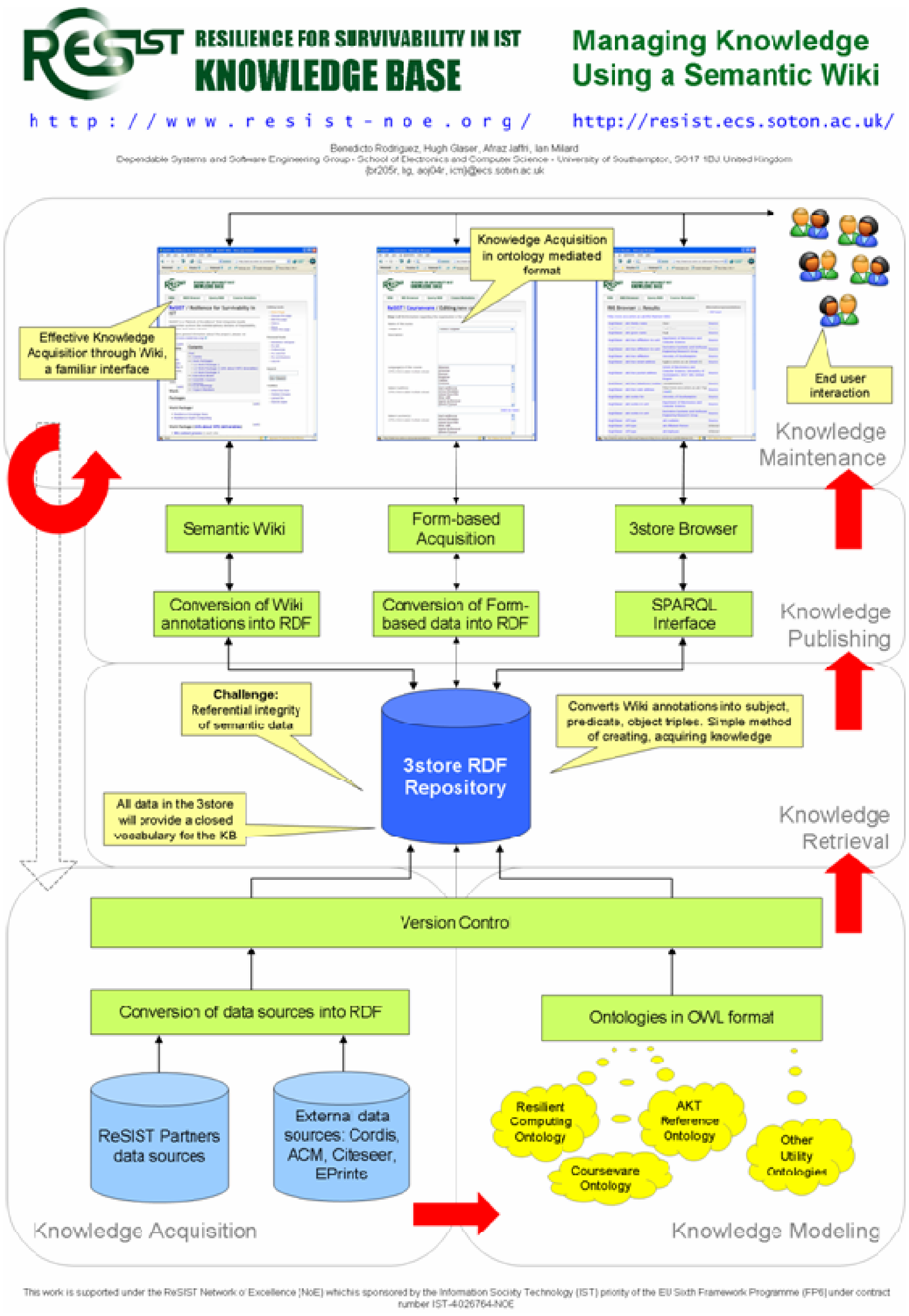
Wiki | RKB Browser | Query RKB | Course Metadata

ReSIST / Courses / {Test} Resist Courseware Material

Name of the course	{Test} Resist Courseware Material
Taught at	The University of Southampton
Currently being taught	Yes
Description	i) Is the course thaught at present or was taught in the past; ii) to change the item "Course Text and resources" in "Courseware used in support to the course (slides, textbooks, personal notes, etc. with links to the relevant files, where feasible), and iii) to add a new item on "Visibility of the courseware material" indicating the types of restrictions to be implemented on accessibility to the material.
Language(s) of the course	http://resist-noe.org/ontology/languages#en
Select Author(s)	Benedicto Rodriguez
Select Lecturer(s)	Benedicto Rodriguez
Submitted by	Benedicto Rodriguez
Number of credits	0
Total hours of lectures	0
Total hours of labs	0
Total hours of personal study	0
Student Interaction Type	http://resist.ecs.soton.ac.uk/courseware/interaction-types#Individual-Project

Assessment methods	http://resist.ecs.soton.ac.uk/courseware/assessment-methods#Individual-Coursework
Pre-requisites	Software Engineering Fundamentals
Infrastructure Required	Semantic Media Wiki
Course Objectives	i) Is the course thaught at present or was taught in the past; ii) to change the item "Course Text and resources" in "Courseware used in support to the course (slides, textbooks, personal notes, etc. with links to the relevant files, where feasible), and iii) to add a new item on "Visibility of the courseware material" indicating the types of restrictions to be implemented on accessibility to the material.
Support courseware used (Textbooks, slides, notes, etc. with their corresponding links when available)	• ReSIST / Resilience for Survivability in IST Freely Available: No Copyright or Restrictions: Yes Description of Cost and Copyright: CORDIS Contract number: 026764 Start date: 1st January 2006 Duration: 3 years • ReSIST Knowledge Architecture: Semantically Enabling Large-Scale Collaborative Projects Freely Available: Yes Copyright or Restrictions: No Description of Cost and Copyright: None

Appendix C. Poster for Ontological Eng. Summer School


RESILIENCE FOR SURVIVABILITY IN IST
KNOWLEDGE BASE
http://www.resist-noe.org/
Managing Knowledge Using a Semantic Wiki
http://resist.ecs.soton.ac.uk/
Benedicto Rodriguez, Hugh Glaser, Afraz Jaffri, Ian Millard
Dependable Systems and Software Engineering Group - School of Electronics and Computer Science - University of Southampton, SO17 1BJ United Kingdom
{br205r, hg, aoj04r, icm}@ecs.soton.ac.uk
Knowledge Acquisition in ontology mediated format
Effective Knowledge Acquisition through Wiki, a familiar interface
End user interaction
Knowledge Maintenance
Semantic Wiki
Form-based Acquisition
3store Browser
Conversion of Wiki annotations into RDF
Conversion of Form-based data into RDF
SPARQL Interface
Knowledge Publishing
Challenge: Referential integrity of semantic data
Converts Wiki annotations into subject, predicate, object triples. Simple method of creating, acquiring knowledge
3store RDF Repository
All data in the 3store will provide a closed vocabulary for the KB
Knowledge Retrieval
Version Control
Conversion of data sources into RDF
Ontologies in OWL format
ReSIST Partners data sources
External data sources: Cordis, ACM, Citeseer, EPrints
Resilient Computing Ontology
AKT Reference Ontology
Courseware Ontology
Other Utility Ontologies
Knowledge Acquisition
Knowledge Modeling
This work is supported under the ReSIST Network of Excellence (NoE) which is sponsored by the Information Society Technology (IST) priority of the EU Sixth Framework Programme (FP6) under contract number IST-4-026764-NOE