Whose ”Fault” Is This? Untangling Domain Concepts in an Ontology of Resilient Computing

B. Rodriguez-Castro and H. Glaser
Dependable Systems and Software Engineering Group
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, U.K.
{br205r, hg}@ecs.soton.ac.uk

Abstract

Certain ontology domain concepts are difficult to model due to the complexity of their definition, the number of roles that they fulfill in the ontology or the different types of relationships they participate in. To assist ontologists in overcoming these challenges, a comparative analysis of two Ontology Design Patterns (ODPs) has been carried out. A terminology is introduced to describe the role and certain reusability scenarios of domain concepts in these ODPs. These findings make explicit certain potentially implicit modeling decisions previously taken in the ontology modeling field. Our contribution is illustrated with a concrete example from an ontology of resilient computing that will benefit from the outcome of this study.

1. Introduction

Ontologies have emerged as one of the key components needed for the realization of the Semantic Web vision [5]. A detailed overview of what an ontology is, including the evolution of its definition in the literature, can be found in Section 1.2 of [9]. Ontologies bring with them a broad range of development activities that can be grouped into what is called ontology engineering [9]. Within ontology engineering, this research primarily focuses on ontology modeling, more specifically on Ontology Design Patterns (ODPs) [8] and on how they can help representing complex domain concepts such us the one presented in the next section. ODPs have evolved from the notion of design pattern (defined in [8] as ”archetypal solutions to design problems in a certain context”) and they are justifiably receiving a significant amount of attention by ontologists due to the preceding success achieved by software design patterns in the context of software engineering [7].

In this study, two specific ODPs are examined [11][13] and a comparative analysis of both allows the introduction of a terminology which characterizes the roles and specific reusability scenarios of the domain concepts that participate in them. These findings make explicit certain potentially implicit modeling decisions previously taken in the development of ontology models.

The rest of this report is organized as follows: Section 2 introduces an example of a use case scenario in the field of resilient computing. Section 3 presents a brief overview of the main work in relation to ontology modeling and ODPs. Section 4 provides the comparative analysis of two ODPs and its contribution in defining role and reusability attributes of ontology domain concepts and finally, Section 5 covers the conclusions gathered from this endeavor and the lines open for further investigation.

2. Motivation

One of the objectives of the ReSIST (Resilience for Survivability in Information Society Technologies) project is to create a knowledge base application in the field of resilient and dependable computing [2]. The ReSIST Knowledge Base (RKB) provides an ontologically mediated web portal that enables the end-user to browse and search different type of information in the area of resilient systems: projects, people, institutions, publications, communities of practice, courses, etc. [3].

To achieve its objectives, the ReSIST KB features an ontology in the domain of resilient systems. This ontology was built using the definitions and taxonomies presented by Avizienis et al. [4]. The domain experts in ReSIST considered this document to be a valid source to cover the concepts that had to be represented in the ontology, saving the need of going through a knowledge acquisition phase during the development process.

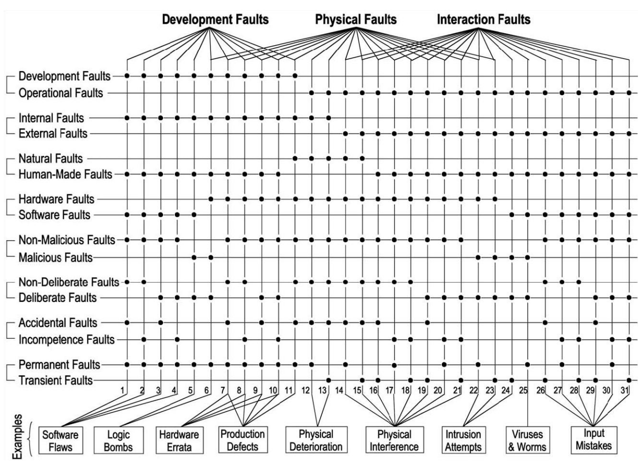

Figure 1. Matrix representation of Fault" from [2] used in the ReSIST KB ontology

Among all the ReSIST concepts, one that particularly stands out from a representational point of view is the concept of "Fault" due to: a) the complexity of its definition (Figure 1), b) the number of roles that it supports in the ontology and c) the number of relationships with other domain concepts in the same ontology. (The concept of "Fault" referred hereto and Figure 1 are explained in detail in [4]).

In the context of ReSIST, the representation of "Fault" has to fulfill the roles of a) classifying occurrences of actual faults in real world systems and b) providing a keyword index for: subjects of publications, research interest areas of projects, institutions or people, and support of resilient mechanisms.

The characteristics of role and reusability of domain concepts identified in the following sections will help to overcome the challenges caused by the multiple usages of the representation of "Fault" in ReSIST.

3. Related Research

There are several methodologies and approaches to build ontologies from scratch that address the topic of ontology conceptualization and more specifically ontology modeling. A comprehensive survey of the most relevant is provided in [6]. These methodologies (in addition of [12]), provide different levels of detail on how ontology conceptualization can be performed although they do not take into account modeling elements specific to OWL given that they are dated prior to the adoption of OWL by the W3C as the preferred ontology modeling language.

Regarding ODPs, related work that has been considered includes [8][11][13][14][1]. Different levels of ontology patterns are also discussed in [14] although in this case in the context of networked ontologies. It distinguishes three: Logical ODPs, Architectural ODPs and Content ODPs. In broad terms, Logical ODPs are equivalent to the modeling elements provided by OWL or to compositions of them. Architectural ODPs are equivalent to Logical ODPs or composition of them and characterize the structure of the ontology determining "how an ontology should look like". A basic example of Architectural ODPs would be a "taxonomy". Lastly, Content ODPs are made of Logical ODPs instances or composition of them and attempt to solve a specific domain modeling problem. The Participation, Role-Task, Design-Artifact ODPs introduced in [8] can be seen as examples of Content ODPs.

Of particular interest are the documents released by the Semantic Web Best Practices and Deployment Working Group of the W3C [1]. They provide an analysis of different approaches to a given ontology modeling problem at the level of detail required for our research and in the context of OWL as the implementation language. We note especially [11] and [13], as they will become a central piece to the outcome of our work discussed in the following section.

All the related work referred hereto; provide guidelines essential to the task of building ontologies and the usage and applicability of ODPs. However, to the best of our knowledge, they do not address approaches to model domain concepts with multiple roles or how these could be reused in the particular case of "Fault" described in ReSIST.

4. Characterizing Role and Reusability

Rector defines in [13] the concept of value partition as a modeling technique where a hierarchy of classes is used to represent features, attributes, or modifiers that describe other concepts in the ontology. According to this definition, a class in the hierarchy is partitioned by a group of subclasses if: a) the subclasses are pairwise disjoint, and b) the subclasses completely exhaust the parent class.

Conversely, Noy presents in [11] different approaches on how to use classes as property values. The situation described occurs when a hierarchy of classes is used as a subject index to annotate other domain concepts in the ontology. However, in this case there is not any restriction on the organization or characteristics that the class hierarchy may possess.

Additionally, both [11] and [13] presents a scenario, Approach 4 and Pattern 2, Variant 2 respectively, where: a) anonymous individuals are used as the value of a property for a domain concept in the ontology and b) the expressivity level of the resulting ontology is within OWL-DL.

4.1. Role of Domain Concepts

Based on this comparative analysis of [13] and [11], we introduce the following terminology to characterize the role that a given hierarchy of classes fulfills in these two ODPs examined:

Generic Class Hierarchy (GCH): refers to a set of classes organized in any hierarchical structure (e.g. a single class or a set of classes organized in a list, a tree or a directed acyclic graph) whether it conforms to the definition of value partition or not.

Domain Class Hierarchy (DCH): refers to any GCH that contains the classes corresponding to the domain concepts that the ontology is intended to represent.

Value Class Hierarchy (VCH): refers to any GCH that is used to provide anonymous individuals as values to properties for other domain concepts in the ontology.

Value Partition Class Hierarchy (VPCH): refers to a GCH that a) is a Value Class Hierarchy and b) conforms to the definition of a value partition.

Furthermore, the ontology could then be divided into two possible spaces:

Domain Concept Space (DCS): identifies the subset of the ontology model that contains all the classes that belong to a Domain Class Hierarchy.

Value Space (VS): identifies the subset of the ontology model that contains all the classes that belong to a Value Class Hierarchy or Value Partition Class Hierarchy.

It is also a good modeling practice, as mentioned in [10] and [13] to make these two spaces, DCS and VS, disjoint.

4.2. Reusability of Domain Concepts

Using the terminology above and based on the roles that class hierarchies fulfill in the ontology, the following reusability scenarios for these can be characterized:

Scenario 1: Let us consider two ontologies O_1 and O_2, with two Domain Class Hierarchies DCH_1 and DCH_2 in their Domain Concept Space respectively. It is possible to apply [11] and [13] to reuse DCH_2 from O_2 to support the role of a Value Class Hierarchy in ontology O_1. In that case:

a) A class (say C_1) of DCH_1 becomes the domain for some property (say P) in O_1.

b) A class (say C_2) of DCH_2 becomes the range for property P in O_1.

c) An anonymous individual from C_2 becomes the value for property P in O_1.

d) DCH_2 becomes part of the Value Space in O_1 and disjoint from DCH_1 in O_1.

Scenario 2: Let us consider a single ontology O_1, with two Domain Class Hierarchies DCH_{11} and DCH_{12}. It is possible to apply [11] and [13] to reuse DCH_{12} support the role of a Value Class Hierarchy for DCH_{11} in O_1. In that case:

a) A class (say C_1) of DCH_{11} becomes the domain for some property (say P) in O_1.

b) A class (say C_2) of DCH_{12} becomes the range for the property P in O_1.

c) An anonymous individual from C_2 becomes the value for property P in O_1.

d) DCH_{12} becomes part of the Value Space in O_1 causing both the DCS and the VS in O_1 to overlap.

4.3. Role and Reuse of "Fault" in ReSIST

The characteristics of role and reusability presented here helped untangling these two aspects in the modeling of the "Fault" concept in the ReSIST KB ontology. This is illustrated as follows:

a) The "Fault" domain concept is represented as a Generic Class Hierarchy resulting in a directed acyclic graph structure to accommodate the different taxonomies that define "Fault" in [4] (Figure 1).

b) The "Fault" Generic Class Hierarchy is used to represent instances of real world faults in the field of resilient and dependable systems. In that sense, it supports the role of a Domain Class Hierarchy in the Domain Concept Space of the ReSIST KB ontology.

c) The "Fault" Generic Class Hierarchy serves as a subject and keyword index for other domain concepts in the ontology. In that sense, it supports the role of a Value Class Hierarchy in the Value Space of the overall ReSIST KB ontology. For example: the "Fault" class hierarchy will supply the value for properties such as "hasResearchSubject" (associated with the concept of "Publication"), "hasResearchInterest" (associated with the concept of "Person", "Institution" or "Project") or "hasSupportFor" (associated with the concept of "Resilient Mechanism").

d) The representation of "Fault" is reused to support the role of both a Domain and Value Class Hierarchy causing the Domain Concept Space and the Value Concept Space of the ReSIST KB ontology to overlap.

This representation of "Fault" coincides with the characteristics outlined in Scenario 2 above for the Domain Class Hierarchy DCH_{12} and helped us explicitly clarify the modeling decisions to be made for this concept in the context of ReSIST.

5. Conclusions and Future Work

Similarities have been discussed between two ontology modeling design patterns that share how they use anonymous individuals to provide the value for properties in the ontology. These similarities allow expanding the notion of value partition to other structures of class hierarchies. A terminology is introduced that describes the roles of domain concepts in the ODPs considered. In the context of this terminology, certain reusability scenarios of domain concepts in ontology models were introduced. These scenarios make explicit certain potentially implicit modeling decisions previously taken in the ontology development field. Our contribution has been illustrated with the representation of the ”Fault” domain concept that is part of the ontology in the knowledge base of the ReSIST project.

Current lines of future work include a study of the conceptual overlap inherent in the class hierarchies that define the ”Fault” domain concept. We are working on a characterization of the different types of conceptual overlap that takes place among the multiple classification criteria that constitute the definition of ”Fault”. We expect that the outcome of this analysis will set the basis for the development of additional ODPs in the context of OWL to address these conceptual overlap scenarios. In addition, the area of ontology evaluation is being explored to identify opportunities for reuse of available evaluation frameworks suitable for these ODPs or the need to develop new ones tailored to our requirements.

Acknowledgements: Many people have contributed directly and indirectly to this work and we thank them all. In particular: Ian Millard, Afraz Jaffri and Madalina Croitoru. The ReSIST Network of Excellence is sponsored by the Information Society Technology (IST) priority in the EU Sixth Framework Programme (FP6) under contract number IST 4 026764 NOE.

References

[1] W3C Semantic Web Best Practices and Deployment Working Group, 2004-5. http://www.w3.org/2001/sw/BestPractices/.

[2] The ReSIST Network of Excellence, 2005. http://www.resist-noe.eu/.

[3] T. Anderson, Z. Andrews, J. Fitzgerald, B. Randell, H. Glaser, and I. Millard. The ReSIST Resilience Knowledge Base. In *DSN 2007 - The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2007.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[6] M. Fernandez-Lopez, A. Gomez-Perez, J. Euzenat, A. Gangemi, Y. Kalfoglou, D. Pisanelli, M. Schorlemmer, G. Steve, L. Stojanovic, G. Stumme, and Y. Sure. A survey on methodologies for developing, maintaining, integration, evaluation and reengineering ontologies. OntoWeb deliverable D1.4, Universidad Politecnia de Madrid, 2002. http://www.aifb.unikarsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-4.pdf.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[8] A. Gangemi. Ontology design patterns for semantic web content. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2005.

[9] A. Gomez-Perez, O. Corcho, and M. Fernandez-Lopez. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)*. Springer, July 2004.

[10] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building owl ontologies using the protege-owl plugin and co-ode tools edition 1.0. Technical report, The University Of Manchester, August 2004.

[11] N. F. Noy. Representing classes as property values on the semantic web. Technical Report Note 5, W3C, Semantic Web Best Practices and Deployment Working Group, 2005. http://www.w3.org/TR/swbp-classes-as-values/.

[12] N. F. Noy and D. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics, Stanford, 2001.

[13] A. Rector. Representing specified values in owl: ”value partitions” and ”value sets”. Technical Report Note 17, W3C, Semantic Web Best Practices and Deployment Working Group, 2005. http://www.w3.org/TR/swbp-specified-values/.

[14] M. C. Suarez-Figueroa, S. Brockmans, A. Gangemi, A. Gomez-Perez, J. Lehmann, H. Lewen, V. Presutti, and M. Sabou. Neon modelling components. NeOn deliverable D5.1.1, Universidad Politecnica de Madrid, 2007.