

# Knowledge Engineering Rediscovered: Towards Reasoning Patterns for the Semantic Web

Frank van Harmelen <sup>a</sup>      Annette ten Teije <sup>a</sup>      Holger Wache <sup>b</sup>

<sup>a</sup> *Vrije Universiteit Amsterdam, {Frank.van.Harmelen,annette}@cs.vu.nl*

<sup>b</sup> *University of Applied Sciences, Northwestern Switzerland*

The full version of this paper appeared in: the 5th International Conference on Knowledge Capture, California, September 1-4, 2009, <http://kcap09.stanford.edu/>.

## 1 Introduction

Research in Knowledge Engineering in the '90s has developed a theory of generic *types of tasks*, which can be implemented by a generic set of *problem solving methods*, decomposable into *primitive inference steps*. This work has led to well-founded methodologies for building knowledge-based systems out of reusable components such as CommonKADS. The important insight was to describe the tasks that Knowledge Based Systems perform at a sufficiently abstract level, the “Knowledge Level”, introduced by Newell in his 1980 AAAI presidential address.

The above raises the question if similar lessons can be applied to Semantic Web engineering. Can we identify reusable patterns and components that can help designers and implementers of Semantic Web applications? Work on the Semantic Web has put great emphasis on the reusability of *knowledge*, in the form of ontologies. *This paper is a first attempt at finding reusable reasoning patterns for SemWeb applications.*

In this work we do the following steps: (1) *Identify typical task types* (and give semi-formal definitions to characterise them in the full paper); (2) *Validate the task types* by showing that a large number Semantic Web applications can be classified into a limited number of such task types; (3) *Define primitive inference steps* (through semi-formal definitions in the full paper); (4) *Validate the primitive inference steps* by showing that the identified task types can be decomposed into the given inference steps.

If the above steps would succeed, this would be of great value to Semantic Web application builders, leading to the possibility of libraries of reusable design patterns and component implementations. It would also constitute an advance in our understanding of the landscape of Semantic Web applications, which has until now mostly grown bottom up, driven by available technical and commercial opportunities, with little or no theory-formation on different types of applications and their relationships.

## 2 Task Types

We will characterise seven different task types. In the full paper we give a more formal description of each task type: the signature and a definition of the functionality.

**Search:** A Semantic Web search engine takes a query in the form of a concept description maps this against an ontology and returns members of the instance-set matching the query-concept.

**Browse:** Browsing is very similar to searching but has as crucial difference that its output can either be a set of instances (as in search), or a set of concepts, that can be used for repeating the same action (i.e. further browsing).

**Data integration:** The goal of data-integration is to take multiple instance sets, each organised in their own ontology, and to construct a single, merged instance set, organised in a single, merged ontology.

**Personalisation and recommending:** Personalisation consists of taking a (typically very large) data set plus a personal profile, and returning a (typically much smaller) data set based on this user profile. The

profile which characterises the interests of the user can be in the form of a set of concepts, or a set of instances (e.g. typical recommender services at on-line shops use previously bought items, which are instances, while news-casting sites typically use general categories of interest, which are concepts).

**Web-service selection:** Rather than only searching for static material such as text and images, the aim of semantic web services is to allow searching for active components, using semantic descriptions of web-services.

**Web-service composition:** The goal of web-service selection is to compose a given number of candidate services into a single composite service with a specific functionality. The input of web-service composition is the same as for the selection of a single web-service above, but the output can now be an arbitrary control flow over a set of web-services.

**Semantic Enrichment** This task type is concerned with annotating objects, such as images or documents, with meta-data. Such added meta-data can be used by task types like search or browse to increase the quality of their answers.

### 3 Validating the task types

In order to measure the completeness and reusability of our list of task types, we have analysed all entries (37) to the Semantic Web Challenge events<sup>1</sup> of the years 2005-2007 to see if they could be properly described with our task types. The analysis of the webchallenges leads us to the following main observations: • All but one of the applications could be classified in terms of our task-types.

- Often, a single application belongs to multiple task types.
- Not a single submission can be described as web-service selection.
- Together search and browse are by far the most commonly occurring task-types.

Taken altogether, we interpret these findings as support for the reasonable completeness and reusability of the task-types that we defined.

### 4 Primitive Inferences

We define a small number of primitive inference steps for Semantic Web applications. A semi-formal definition of these primitive inferences, including their signature can be found in the full paper. We use the following primitive inferences:

**Realisation** determines which concepts a given instance is a member.

**Subsumption** determines whether one concept is a subset of another.

**Mapping** finds a correspondence relation between two concepts. We follow the common approach, where the correspondence relation can be either equivalence, subsumption or disjointness.

**Retrieval** is the inverse of realisation: determining which instances belong the given concept.

**Classification** determines where a given class should be placed in a subsumption hierarchy.

We have chosen the above five as primitive inference steps because they seem to constitute a conceptually coherent (although not formally minimal) set. The full paper shows how each of the task-types can be decomposed into the primitive inferences described. For example search is a combination of *classification* (to locate the query-concept in the ontology in order to find its direct sub- or super-concepts) followed by *retrieval* (to determine the instances of those concepts, which form the answers to the query).

### 5 Concluding remarks

The main contribution of this paper has been to provide a first attempt at providing a *typology of semantic web applications*. We defined a small number of prototypical task-types, and somewhat surprisingly, almost all entries to three years of Semantic Web Challenge competitions can be classified into these task-types. We also showed how each of these prototypical task-types can be decomposed into a small number of primitive inference steps. Analogously to established practice in Knowledge Engineering, these results provide a first step towards a methodology for building semantic web applications out of reusable components. We regard this work indeed as first steps towards this goal. We would expect the typology of task-types to grow beyond the current set of seven to cover a larger corpus of semantic web applications.

---

<sup>1</sup><http://challenge.semanticweb.org/>