Building the Semantic Web with Syntactic and Semantic Building Blocks: Position Paper

Sander Bosman
sanderb@cs.kun.nl

Computing Science Institute, University of Nijmegen
The Netherlands

Abstract. The Semantic Web vision sketches the need for more structured information on the Web, allowing the computer to assist users in creating, finding and viewing structured information in various ways. Creating free text, such as Web pages, is relatively easy for people. But how can users be helped in creating more *structured* information? This position paper describes our view on how natural language building blocks could help users create structured information models.

1 Introduction

From the computer's point of view, the World Wide Web is just one big pile of bytes. Only when a Web page is read by a person, it carries information for that person. For many situations this is not a problem: the World Wide Web is very successful, just because anybody can publish their hypertext documents and anybody can search, browse and use the incredible amount of textual information available.

Still, the relatively unstructured nature of information on the Web is causing problems. Web content can easily be understood by humans. The computer however, has difficulty 'understanding' this relatively unstructured information. This becomes a problem when we need the computer to do something with the information, such as searching. One of the best examples is the search engine, without which finding information on the Web would be almost impossible. A search engine regards a web page as just a list of words; it does not know about various meanings of those words, as it does not know what the page is about. It can not answer the question *"What is John's phone number"*; it only can list the pages containing the words *"John"*,*"phone"* and *"number"*. Web users are getting less patient with the numerous irrelevant 'matches' returned by such a query [2].

The Semantic Web initiative sketches the need for more structured information on the Web [3]. Structured information, combined with automatic processing will likely boost the possibilities in various areas, such as information search, question answering and e-commerce.

Who will create all this structured information? One way is to automatically find structure in existing documents. For example, [6] tries to identify various word meanings based on word co-occurrence. Although such methods can help finding structure, they will not be a replacement for human judgement in the near future. The need for manual creation of structured information remains.

People have more difficulty creating structured information than writing information as free text. Therefore, to build the Semantic Web we will need ways of structured information modeling which are as easy to learn and apply as creating free text Web pages. In this text we describe our view on how users could be helped in modeling structured information.

2 Creating Structured Information

Computer software can be seen as structured information: bits, bytes, commands, data structures, etc. The computer's need for structured data is reflected in the various computer applications. We can regard many applications as information modeling environments: a database, a programming environment, an address book, a CAD drawing program. They all allow the user to model information from various domains and for various reasons.

A modeling environment supplies *modeling concepts* to work with. For example, a relational database provides databases containing a set of tables; a table consists of one or more columns; a table contains tuples which represent concrete instances. The user composes these various building blocks together into a meaningful database. Modeling concepts need a *representation* in order to specify them; often this representation is textual. For relational databases, SQL is the text representation which is used to manipulate the database and to query the information it contains.

Another example of a modeling environment is the Java programming system [1]. Here, the modeling concepts are classes, objects, methods, statements, integers, etc. The representation is again defined as text: the Java syntax.

Of special importance to shaping the Semantic Web are ontologies, another form of model [10]. Although there is no exact definition of an ontology, it minimally allows specification of concepts and concept hierarchies. Various representations exist, most notably RDF [5].

Humans usually find modeling structure difficult. It is often easier to write free text than to model in a structured way. Suppose someone would like to create a list of her CD collection. The easy, unstructured way of doing this would be writing down (or scanning) the text on the cover backs. Now, if she would like to know how many songs played by the group *"Bettie Serveert"* she owns, the only way of doing this is to browse the text. The other approach forces her to think about the structure of the collection. It consists of CD's; each CD is performed by some artist; a CD contains songs, each having a title and a length. Or maybe a song can appear on multiple CD's? What about collection CD's, where not all songs are performed by the same artist?

Clearly there is no such thing as *the* structure, but any structure is just one of the possible views on the world. A structure view depends on context, personal knowledge and personal taste.

Many modeling methods exist, providing modeling concepts work with. Only few of these also provide a 'cookbook' telling how to actually use the concepts to create a model. One example can be found in [8].

Another problem of modeling environments is the limited flexibility. Often the set of modeling concepts offered is fixed, as is the (text) representation of those concepts. For example, in a relational database all information has to be modeled in terms of tables and the existing data types; any maintaining or querying must be done using the SQL syntax. Modeling drawings in such a database becomes awkward: concepts like point, circle and polygon can not be added as data types. In addition, it is not possible to extend the SQL syntax with new representations such as "`(10,20)`" or "`circle of radius 10`".

Limited flexibility often causes awkward modeling constructs. Constructs which a person finds natural do not always have a corresponding modeling concept. For example, specifying three telephone numbers for a person in a relational database can often only be done by listing the person three times.

Also the fixed *interpretation* of text representations can cause confusion. One study shows some people have another interpretation of the Boolean connective 'or' than the interpretation used by SQL [4].

All these problems make creating structured information difficult. We have to simplify structured modeling, in order to get more people create structured information for the Web.

3 Our View

We take the view of the Semantic Web as being the infrastructure for one big, distributed information system. This system contains various types of structured and unstructured information, such as HTML pages, a CD collection or an address book. We also regard the procedures that specify the behaviour of the system as structured information. Typical behaviour is answering a query (*"Where does John live?"*), deriving new information from existing information, or creating an overview of persons and their phone numbers.

A user can browse and query information, as well as modify it. This includes specifying someone's phone number, introducing new concepts to work with, or even adding new behaviour. Of course, all this functionality can be restricted through a security policy.

From a usability point of view, the best information system would probably be a human. A human personal assistent has some characteristics that computers lack, such as:

- interaction with the human assistent can be done using natural language, such as written free text.

– a human assistent can be told new information, and taught new behaviour.
– a human does not need formal, consistent and complete information: he tries to make a coherent picture from small information snippets.
– if the meaning of some sentence, word or task is unclear, a human assistent will give feedback[1] and ask how he should interpret it.

Our goal is to investigate what is needed to create a computer information system having the same properties.

3.1 Machinery of a Possible System

A dialog between a user and our imaginary system starts when the user enters a piece of free text, which we will call a *term*. On entering a term, the system tries to interpret the term in several steps. The first step is to parse the term and find the intended interpretation: a list of actions to perform. If a single, plausible interpretation is reached, these actions are performed in the second step. Actions can update the internal memory of the system, print the result of a query, or they can even change the interpretation procedure itself (expanding the set of possible input terms, for example).

In more detail, the system consists of a model M and an interpreter I. The model is the internal memory of the system. It contains the conceptual structure of the information stored, as well as instances (the data). Various classes of concepts are supported, such as

– object types like 'CD' or 'person'
– relations between objects like "(person) lives in (city)"
– procedures (actions) like "print (text)"
– rules like "if a person is older than 18, he is an adult"

The model can be seen as an ontology in the broader sense. The conceptual structure of the model is not fixed: new concepts can be created when the need arises.

The model should incorporate some form of concept management. For example, concepts can be annotated with the person or organisation who defined it. This makes concept meaning clearer, and allows co-existence of concepts which are 'the same', but defined in different contexts.

The interpreter I parses an entered term, disambiguates it and executes the actions that result. Without going into detail, we will assume the interpretation procedure is part of the model, in the form of interpretation rules. This makes it possible to incrementally extend the set of terms that the system can interpret, by adding interpretation rules.

The system has a basic interface allowing the user to inspect and modify the model. This could be through text terms (e.g., *"create new type: CD"*) or through

[1] or better, '*should* give feedback'

a graphical user interface allowing the manipulation of objects in the model. The basic interface is the simplest (and probably the hardest) way of 'teaching' new information.

3.2 Assisting the User

Having described a piece of machinery which supports (restricted) free text dialogs with the system, we need to focus on how to assist the user with modeling information and teaching new interpretations.

Since users are allowed to enter free text, it will be common that the system can not interpret a term. In that situation, the user is prompted to specify the interpretation. For example, the user enters the following:

```
John lives in Nijmegen
```

The system will ask the user to specify how to interpret this term. The user could say (without going into detail how to say this):

the term represents a fact
John is a person
Nijmegen is a city

Combining these information snippets, the system now can infer enough information to reach an interpretation:

the term is a fact, so I have to store it in the model
the user specified subterms, so the whole term is some sort of relation
there must be types called 'person' and 'city' in the model; if not, create them
there are instances of person and city: 'John' and 'Nijmegen'
there is a binary relation r_1 *between person and city, and an instance* r_1*('John','Nijmegen')*
instances of this relation are specified using a term like "(person) lives in (city)"

3.3 Further Steps

The example above indicates there are good possibilities in assisting the user in structured modeling. Other possible approaches include automatic identifcation of subterms, using general natural language processing techniques to propose sentence structure [6]. Programming by example techniques allow generalization from procedures written for one specific situation [7]. Natural programming is a closely related field which can provide information about how people model information in a 'natural' way [9]. Information modeling techniques such as [8] will provide insight in the structure of the model.

Our aim is to construct a prototype of the system described, using it to develop methods for assisting users with information modeling.

References

1. Ken Arnold and James Gosling. *The Java programming language*. Addison-Wesley, 1996.
2. A.Spink, J.Bateman, and B.Jansen. Users' searching behavior on the EXCITE web search engine. In *Proceedings of the 1998 world conference of the WWW, Internet and Intranet*, 1998.
3. Tim Berners-Lee. *Weaving the Web*. Harper San Francisco, 1999.
4. John Pane Chotirat. Studying the language and structure in non-programmers' solutions to programming problems.
5. W3C Consortium. Resource Description Framework (RDF) Model and Syntax Specification, 1999. URL: http://www.w3.org/TR/REC-rdf-syntax/.
6. F.A. Grootjen. Indexing using a grammerless parser. In *Proceedings of the First International Workshop on Natural Language Processing and Knowledge Engineering (NLPKE 2001)*, Tucson, Arizona, USA, October 7-10 2001.
7. Henry Lieberman. *Your wish is my command: programming by example*. Morgan Kaufmann, 2001.
8. G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989.
9. J.F. Pane, B.A. Myers, and L.B. Miller. Using HCI techniques to design a more usable programming system. .
10. Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.