# Using Web Standards for Timetabling

P. De Causmaecker, P. Demeester and G. Vanden Berghe

KaHo Sint-Lieven
Information Technology
Gebr. Desmetstraat 1
9000 Gent
Belgium
{patdc, peterdm, greetvb}@kahosl.be

With the advent of the Semantic Web we examine how the technologies that lie at its basis can be applied for timetabling. In this paper, we give some directions for using these recently developed techniques. Many researchers spend a lot of time re-implementing and modifying already existing heuristics in order to make them suitable for their particular problem. We now put forward how the Semantic Web could interpret problem specific knowledge, and thus improve the communication between heuristic approaches and any problem specification.
The ideas behind the Semantic Web lead us to proposing a model for future timetabling development, which is based on loosely coupled components that communicate with each other by sending XML messages. The advantage of such a component model is that only a few components need to be adapted when transferring it from one timetabling application to another.

***Keywords***: timetabling, Semantic Web, web standards

## 1. Introduction

Since the term was first introduced in 1999 by Berners-Lee in 'Weaving the Web' [1], the Semantic Web gained a lot of popularity. Several international conferences are currently devoted to the subject[1] and even DARPA[2] and the European Community sponsor a lot of research in this domain. It is the intention that the Semantic Web will be the successor of the current web. At present the Semantic Web is still in the making.
The terminology used in timetabling approaches is often very confusing (scheduling, rostering, timetabling,…[2]). However, most of the timetabling problems have a similar goal. They all aim at assigning activities to time slots subject to constraints. Timetabling problems are very hard to solve and there exist lots of different approaches for generating good quality solutions.
In this paper we indicate how the technologies involved in the Semantic Web can be used in the domain of timetabling. We introduce the Semantic Web in Section 2 and

[1] See http://www.semanticweb.org/resources.html#events for a list
[2] DARPA: Defense Advanced Research Projects Agency (see http://www.darpa.mil/)

in Section 3 we explain some related acronyms. Section 4 puts forward how timetabling could benefit from the Semantic Web. A few possible applications are discussed in Section 5 and we conclude in Section 6.

## 2. The Semantic Web

There exist numerous definitions of the Semantic Web, all meaning more or less the same. For this paper, we have selected two. The first one is from Berners-Lee and the second one is taken from the W3C Semantic Web Activity Statement.

"*The Semantic Web is a web of data, in some ways like a global database*"[3].

"*The Semantic Web is a vision: the idea of having data on the web defined and linked in a way, that it can be used by machines - not just for display purposes, but for automation, integration and reuse of data across various applications*." [4]

The current web was designed to improve the exchange of information between different people [5]. Unfortunately machines cannot access this information easily since there is too much overhead involved: the content is not cleanly separated from the instructions on how to present content. An HTML[3] document mixes content and layout instructions. Another problem is that a word can have different meanings. Any current

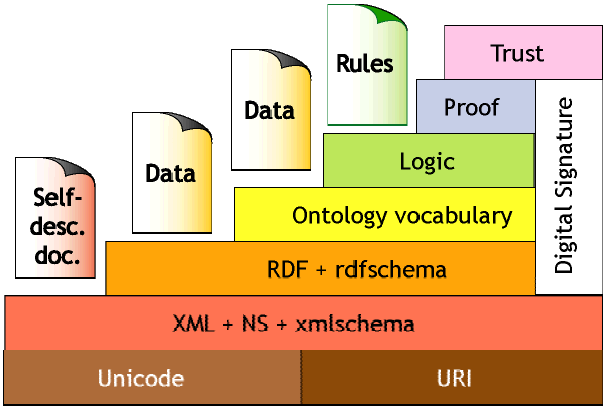


Figure 1: The semantic web architecture as proposed by Tim Berners-Lee

search engine on the web suffers badly from this defect, and a lot of effort is spent trying to remedy that. Using languages that allow automatic interpretation of content can solve this problem at a fundamental level.

It is the goal of the Semantic Web to make information on the web machine-processable. A lot of things which currently need human aid can be automated in this

[3] HyperText Markup Language

way, e.g. making a synthesis of information that is found in different documents, filtering the results of a search engine, etc.
Berners-Lee et al. [6] describe an interesting example of the process of automating the appointment making with a physician when the Semantic Web is ready. To bring this situation to a favourable conclusion, the authors propose the usage of software agents, which roam the Semantic Web looking for information. The Semantic Web will be the natural habitat of software agents [7]. It requires little imagination to apply these ideas to timetabling.

## 3. The different layers of the Semantic Web

Berners-Lee proposed an architecture (Figure 1) for the Semantic Web [8].
Some of the ideas and techniques will be applied in timetabling and therefore we discuss the different layers of the architecture very briefly. More detailed information can be found in [9, 10, 11].

### 3.1 First layer

The lowest layer consists of Uniform Resource Identifiers (abbreviated as URIs) and Unicode[4]. URIs are used to identify objects on the Web. These objects can be anything: a web page, a paragraph from a web page, an MPEG movie, an MP3 sound fragment, a JPEG image, a test bed for timetabling problems, algorithms, rooms, lecturers etc. Once something has a URI it is said to be "on the Web". A well-known form of a URI is the Uniform Resource Locator (URL), this is the string which is entered in the address bar of a browser.

### 3.2 Second layer

XML handles the syntax of documents in the second layer. XML [12] is an acronym for eXtensible Markup Language. XML is more than the successor of HTML. XML offers the possibility for users to create documents using their own structure and

[4] "Unicode is an entirely new idea in setting up binary codes for text or script characters. Officially called the Unicode Worldwide Character Standard, it is a system for "the interchange, processing, and display of the written texts of the diverse languages of the modern world". It also supports many classical and historical texts in a number of languages.
Currently, the Unicode standard contains 34,168 distinct coded characters derived from 24 supported language scripts. These characters cover the principal written languages of the world.
Additional work is underway to add the few modern languages not yet included.
Also see the currently most prevalent script or text codes, ASCII and extended binary-coded decimal interchange code (EBCDIC)." [taken from the whatis.com website, http://whatis.techtarget.com/definition/0,,sid9_gci213250,00.html]

syntax. Unlike in HTML, which only contains a limited number of tags, XML allows to invent tags suitable for a particular problem.

**Example 1** presents an XML document that contains data about the authors of a paper. The tags are chosen by the developer in order to clarify the content of the XML document.

```
<?xml version="1.0"?>
<paper>
     <title>Using web standards for timetabling</title>
     <author>Sep Ducamaercke</author>
     <author>Per Tedesem</author>
     <author>Venghed Banger</author>
     <affiliation>KaHo Sint-Lieven, Gent</affiliation>
</paper>
```

Confusion, which is not unlikely if everybody invents different tags, is avoided by using XML schemas. These schemas determine how the content should be formatted in order to create a syntactically valid XML document. Such document can be exchanged between people, companies or applications. If it satisfies a particular schema constructed by an individual or company, it is guaranteed that it is understandable by this individual or company.

**Example 2** is an XML Schema that corresponds to the XML document of Example 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="paper">
  <xs:complexType>
    <xs:sequence>
       <xs:element name="title" type="xs:string"/>
       <xs:element name="author" type="xs:string"
        maxOccurs="unbounded"/>
       <xs:element name="affiliation" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

The schema describes how the XML document of Example 1 should be formatted. The document consists of a *root* element `paper`, which has `title`, `author` and `affiliation` as *children*, all having `string` as data type. In XML Schema, a large set of simple data types are already included [13]. Any data type that is not built in can be constructed using the `complexType` element. The element `author` can occur several times since its attribute `maxOccurs` has the value `unbounded`.

Let us suppose that computer system of a conference on timetabling uses such a schema to classify the submitted papers. If every submitted paper is accompanied by an XML document satisfying this schema, it is guaranteed that the computer system can read each document and classify it accordingly.

A personal meaning can be attached to documents by user-defined tags.
Since the second layer builds upon the lowest layer, it is possible to uniquely identify tags. URIs are used for that purpose. By identifying tags, an "XML Namespace" [14] will be created. Namespaces created by others can be reused.
As Example 2 demonstrates, XML only describe the content of a problem. In contrast with HTML, content and presentation are separated. XSL [15] (eXtensible Style Language) or CSS [16] (Cascading Style Sheets) are used for presenting XML content.

### 3.3 Third layer

The next layer offers the opportunity to make 'machine-processable' statements. The technology behind it is called Resource Description Framework (RDF) [17]. RDF allows to add formal semantics to the web [18]. That is done by defining a data model, using three object types:

- Resources: objects that can be identified by a URI
- Properties: specific characteristics that can be used to describe a resource
- Statements: triples consisting of a resource, a property and a value of that property

The RDF description model uses triples, which consist of a subject, a predicate and an object. These triples are known as statements. Instances of this description model can be viewed as directed or labelled graphs.

> **Example 3** presents an RDF statement with information about a paper that is written by Sep Ducamaercke: '*PaperX has creator Sep Ducamaercke*.'
> Figure 2 illustrates that 'PaperX' is the resource (or the subject), 'creator' is the predicate (or the property) and 'Sep Ducamaercke' is the object.
> Note that the resource is represented by a URI.

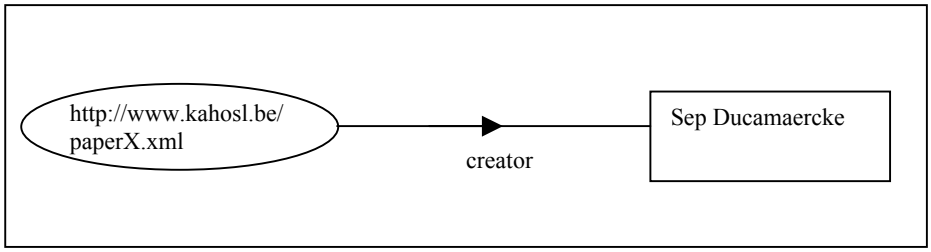


Figure 2: An RDF example

RDF provides only a model for representing metadata[5]. Possible representations are the labelled graph of Figure 2, a triple list, etc. Another obvious candidate for an alternative representation is XML. With XML, instances of the model can be stored in

[5] metadata is data about data.

files and exchanged between users. RDF cannot replace XML. It builds a layer on top of XML, in order to enable interoperable exchange of semantic information.

**Example 4** presents the XML version of the RDF statement of Example 3. Note that we introduce two namespaces of which only one has a prefix (`rdf`). Instead of defining a completely new concept, we re-use the concept `Description` that is already defined in this namespace. The namespace without prefix is called the default namespace.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://sep.ducamaercke.net/rdfexamples">
  <rdf:Description about="http://www.kahosl.be/paperX.xml">
     <creator>Sep Ducamaercke</creator>
  <rdf:Description>
<rdf:RDF>
```

An object-oriented type system is situated on top of the basic RDF model: RDF Schema (sometimes abbreviated as RDFS). RDF Schema is a data-typing model for RDF [19]. Mark however that the functions in RDF Schema and XML Schema have are not the same [18]:
XML schema is used to control the syntax of an XML document, while RDF schema only considers the interpretation of RDF statements.

### 3.4 Fourth layer

The next layer is the ontology layer. It is the layer in which at present most of the research concerning the Semantic Web takes place. Fensel et al. [20] explain why ontologies are nowadays popular: "*ontologies are becoming popular largely because of what they promise: a shared and common understanding that reaches across people and application systems*".
A commonly accepted definition of an ontology is:

"*an ontology is a formal, explicit specification of a shared conceptualisation*"[21].

A conceptualisation is a way of thinking in a particular domain. This is typically expressed as a set of concepts (entities, attributes, processes), their definitions and their inter-relationships.

One of the interesting applications is the ontology-based translation service that maps different data structures onto areas where no standard ontologies exist or where people need to translate their own terminology into the standard. This translation service covers structural, semantic and language differences [20] (Figure 3).
Although ontology languages (LOOM [22], POWERLOOM [23], KIF [24], CYCL [25], Ontolingua [26], etc) were created years before the term Semantic Web was invented, the two most important languages are based on RDF Schema. The creators of both ontology languages quickly realised that it was better to co-operate than to compete. In 2001 they finished a unified ontology language called DAML+OIL, which is

a junction of the two original names. DAML is funded by DARPA and OIL is funded by the EC. DAML+OIL still applies RDF Schema but allows more expressiveness: inverses, unambiguous and unique properties, restrictions on properties, disjoint un-

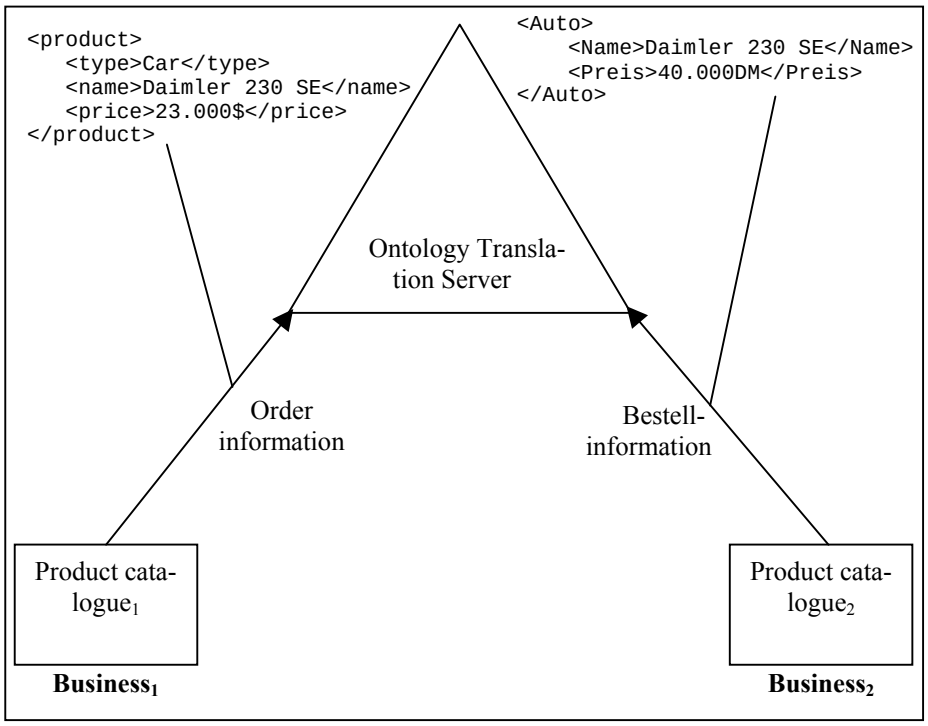


Figure 3: translation of structure, semantics and language. Example taken from [20]

ions,… can be defined. For more details consult [27]. In 2002, the W3C is working on a new standard language for web ontologies (OWL: Ontology Web Language) and they took as a starting point DAML+OIL. They published very recently 3 working drafts [28].

As part of the DAML programme, people from the academic world are working on DAML-S, an ontology of services. Services refer to dynamic web sites which allow actions or changes in the world [29]. In order to use services, software agents need to interpret a description of the web service. They must also know how the service can be accessed. The goal of the DAML-S project is to establish the description based on DAML. Some of the web service tasks, which will be automated once DAML-S is fully functional, are:

- automated web service discovery,
- execution,
- interoperation,
- composition and
- execution monitoring [30].

There is an effort from the industry (especially from IBM, MicroSoft, HP, etc) to develop standards for the description of web-based services: UDDI, WSDL, WSFL, SOAP, .NET, etc. We briefly introduce these services by explaining SOAP as an example. SOAP [31] stands for Simple Object Access Protocol. DevelopMentor, Microsoft, and UserLand Software originally developed it. This protocol specifies how to exchange information in a decentralised, distributed environment. It is operating system and programming language independent. Applications written in Java and running on a SUN-OS environment can easily co-operate with a computer program written in C++ running on a WINTEL environment. SOAP uses HTTP as transport protocol and XML messages to exchange information. A few web sites, already offer "web services". The XMethods website [32], for instance, offers simple applications, such as the conversion of currencies or playing chess against a computer that can be invoked remotely. There will probably be companies in the future, that will offer rather complex applications on payment, evocable from any computer. At present, these services still imply human involvement: users still have to check manually which methods to invoke, which services to use, etc. Once the above-mentioned industry standards are fully functional, all this should happen automatically.
The industry standards focus on short term usage. Since these technologies are not based on languages like DAML, and thus lack expressiveness, they are for humans rather than for software agents. For a further discussion on this subject we refer to [33].

### 3.5 Fifth layer

The fifth and the sixth layer are still under development. The logic layer provides a language for describing sets of deductions, which can be derived from a collection of data. It is the purpose to determine how new facts can be derived from the world described in the ontology layer. The people involved in DAML are now working on DAML-L [34]. It is a logical language with well-defined semantics and the ability to express propositional Horn clauses, which enable a compact representation of constraints and rules for reasoning.

### 3.6 Sixth layer

The proof layer describes the steps taken to reach a conclusion from the facts. These proofs can then be passed around and verified, and used as short cuts to new facts in the system, without repeating the deductions.

### 3.7 Seventh layer

This top layer is not a physical layer. It considers the confidence that will arise once the lower layers are ready. It is actually a metaphor for the reliability of the data received on the Semantic Web. In order to reach this trust, people should attach a digital signature (Figure 1) to their documents.

## 4 The Semantic Web and timetabling

It is our opinion that the above-described technologies can be useful in the domain of timetabling and that timetabling can gain profit from the Semantic Web once it is in operation. In this paragraph we introduce some directions for applications of these technologies, on the basis of the above-mentioned layers. For each of the interesting layers (from the second up to the fourth layer) we will explain how these technologies can be applied in timetabling.

### 4.1 Second layer

At the essential, basic level we have to define a format or language for representing instances of a timetable. Burke et al. [35] mention requirements for a standard data format for timetabling. Such format should be:

- general, expressing all kinds of timetabling problems,
- complete, expressing these problems in full detail,
- accessible, easy to translate to and from [36].

By catching the timetable instances in XML, we satisfy the above requirements:

- through the use of a general ontology we are able to express all kinds of timetabling problems,
- by specialising the general timetabling ontology we can express the timetabling problems in full detail,
- since we have XML parsing tools available, it is easy to translate a format into another format. Almost every database can also generate XML files. This massive support from the software industry is an extra reason to express the elements of the language STTL [36] in XML.

In [35], the authors note that:

"*Programs will be necessary to convert data in a non-standard format into the standard data format. In order to encourage the development of consistent conversion programs we will publish a standard interface. This might then be used to interface with a common conversion application which would be used as the standard front-end to the conversion utility.*"

These requirements are fulfilled by XML through the introduction of the Document Object Model (DOM) [37].

*"The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page."*
This interface allows a programmer to manipulate the content of an XML document: elements can be added, deleted, moved, etc.
Burke et al. [35] conclude with:
"*Many formats used by timetabling applications will have been designed with concerns for minimising data storage requirements or to facilitate the fast processing of data. This means that such formats would not be suitable as a standard format since generality or readability of data (by humans) would be forfeited. Our aim, however, is to provide generality; allowing the maximum number of data instances to be represented as is practicable.*"
It pre-formulates one of the postulates of the Semantic Web. Since XML is simply text (because it is based on Unicode) it is readable by humans. This is often considered a disadvantage because data cannot be saved in a binary format. A text format is not very economical in storage space.

An extra advantage of web technologies for the timetabling domain lies in their intrinsic focus on interactivity. Users should be able to express their goals and their interpretations of soft constraints freely. By providing an ontological description of the domain, the resulting XML files may become very rich in information content. Among the available technologies to realise this, we certainly want to mention fuzzy logic and fuzzy set theory [38, 39, 40, 41].

### 4.2 Third layer

The technologies of this layer enable the creation of instances of timetabling resources. Every resource can be given a unique identifier. We consider using the following resources:

- Students, classes, rooms, teachers,… (see later)
- Documents with timetables, requirement descriptions, goals,…

In **Example 5**, the instances `Sep Ducamaercke, Per Tedesem` and `Venghed Banger` are further described using `rdf:type`. We use `daml:Class` to describe the class to which every instance belongs. This is only because the RDF language does not have such statement.

```
...
<rdf:Description rdf:about="resource.daml#Sep Ducamaercke">
   <rdfs:comment>This is the first instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#fullprofessor"/>
   </rdf:type>
</rdf:Description>
<rdf:Description rdf:about="resource.daml#Per Tedesem">
   <rdfs:comment>This is the second instance</rdfs:comment>
```

```
   <rdf:type>
      <daml:Class rdf:about="resource.daml#researchassistant"/>
   </rdf:type>
</rdf:Description>
<rdf:Description rdf:about="resource.daml#Venghed Banger">
   <rdfs:comment>This is the fourth instance</rdfs:comment>
   <rdf:type>
      <daml:Class rdf:about="resource.daml#associateprofessor"/>
   </rdf:type>
</rdf:Description>
...
```

The instances in RDF can be saved in files, but a better solution is using a database. In 2002, APIs that allow RDF statements in programming languages such as Java, C++,…[42, 43, 44] start to appear.

### 4.3 Fourth layer

As mentioned earlier the fourth layer is the layer where all the action concerning the Semantic Web nowadays is situated. In this paragraph we explain how we want to use ontology as a starting point for designing timetabling system. We also propose an upper level timetabling ontology and we discuss a part of this ontology in more detail. In a last part of this paragraph we investigate how our ontology relates with the STTL ontology as formulated by Kingston [36].

#### 4.3.1 Ontology as a starting point for design

Developing an ontology for timetabling leads to a better understanding of the problem and this offers the opportunity for integrating separate timetabling systems. The developed ontology can be used in two different ways:

- existing timetabling systems can be mapped to this ontology or
- newly developed timetabling systems can take this ontology as a starting point. An ontology can thus be used as the basis for the design of a software system.

Both approaches lead to a better co-operation between timetabling systems. Through the mapping, concepts in the original timetabling application can be ‘translated’ to concepts in another application. In this way it is possible to use components/parts from already existing applications for new timetabling problems. It should be possible to only use those components of different timetabling applications that suit best your needs. If a particular timetabling application has a clearly arranged GUI and another one has a satisfying algorithm then both components can be incorporated in a new application.

Basing newly developed timetabling applications on a commonly accepted ontology will facilitate the co-operation between different applications.

Through the use of domain ontologies, a domain of discourse can be set up that both computers and developers can comprehend. In this way there is a strict distinction between fundamental domain concepts and problem-solving techniques. In modern

knowledge engineering, building intelligent systems is seen as the process of designing and assembling domain ontologies, instantiating knowledge bases (the RDF files that describe the resources) and designing domain-independent problem solving methods.

> *"The goal is to transform system development into a matter of selecting, modifying and assembling previously tested and debugged components, rather than to require programming of each new application from scratch. The hope is that, because these components are at a quite high level of abstraction, the assembly of the components can take place more easily than in other cases of software reuse."*[46]

By incorporating domain ontologies, the domain-dependent information becomes explicit, accessible and editable.

In the design phase a distinction is possible between the computational model and the application model. The application model provides the full specification of the problem and allows for maximal expression of the user's intentions. This information has to be filtered when sending it to specific components. It enables future extension of the application model without disrupting the functionality of the existing components. The computational model can be considered one of the components, working on a subset of the data and concepts visible and accessible for the user.

### 4.3.2 The proposed timetabling ontology

As a starting point for a discussion on a commonly accepted ontology for time-

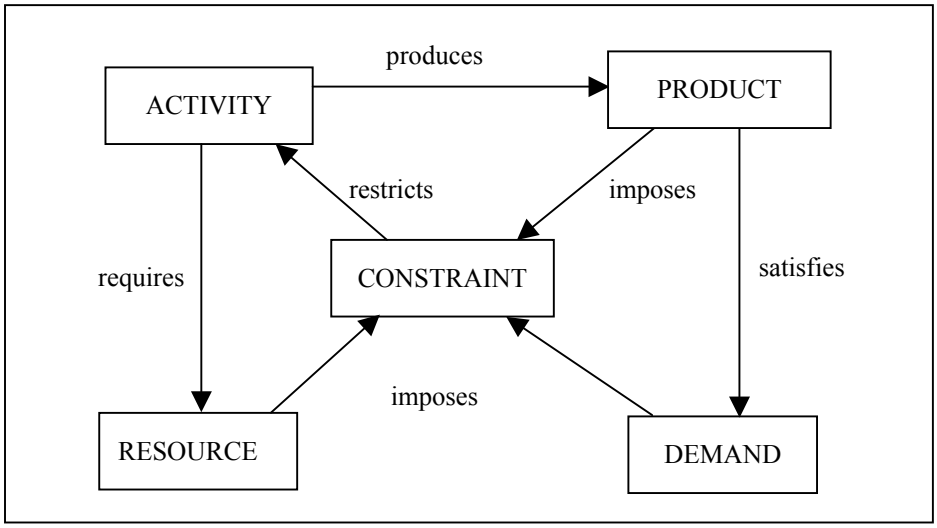


Figure 4: OZONE top-level ontology adapted to timetabling

tabling we propose the upper-level ontology that we developed based on the OZONE [45] scheduling ontology (Figure 4). This upper-ontology has 5 central concepts, namely:

- ACTIVITY: a process that can be executed over a time interval. Resources are needed to execute it. The execution depends on and affects the current state of these resources.
- CONSTRAINT: restriction on the set of values that can be assigned to a variable.
- DEMAND: a request for services (products) that the system provides.
- PRODUCT: is a good or service provided by a system. The product is realised through the execution of a set of activities.
- RESOURCE: an entity that supports or enables the execution of activities. The ultimate timetabling (or scheduling) tool is making efficient use of resources in support of multiple, competing activities.

The benefit of a commonly accepted timetabling ontology is the ease of classifying timetabling problems. According to the kinds of demands, constraints,… timetable problems can be categorised and this will simplify the applications.

### 4.3.3 Example of an ontology: the resource ontology

Starting from these very high-level concepts the next step requires identifying the different types of constraints, resources (re-usable and consumable resources,…), demands, meetings and activities.
As an example we give the resources ontology (constructed in UML) which is based on the habits of our institute (Figure 5). The names reflect the meaning of the resources.

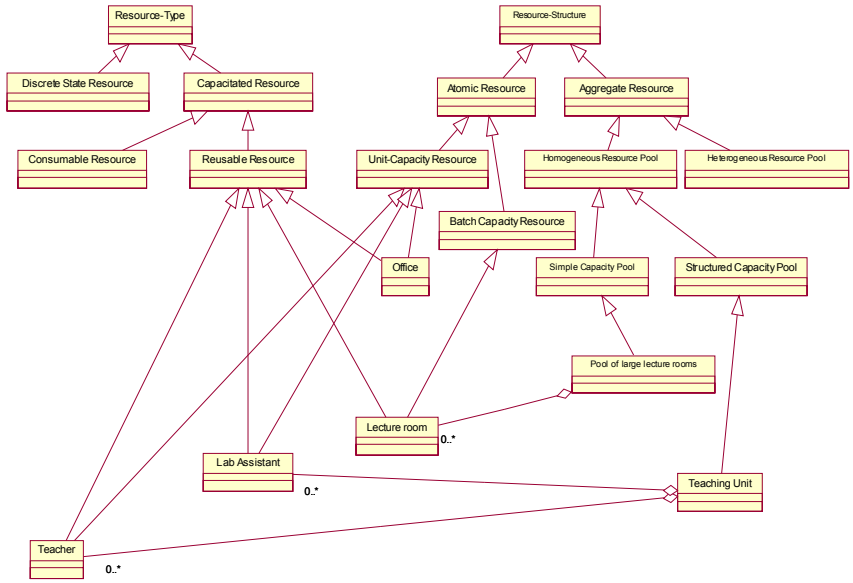


Figure 5: UML example of an OZONE-based ontology [45] for timetabling resources.

We opted not to use time as a (consumable) resource.

### 4.3.4 Relation with STTL

The elements in Kingston's STTL [36] can be reformulated as an ontology. In fact,

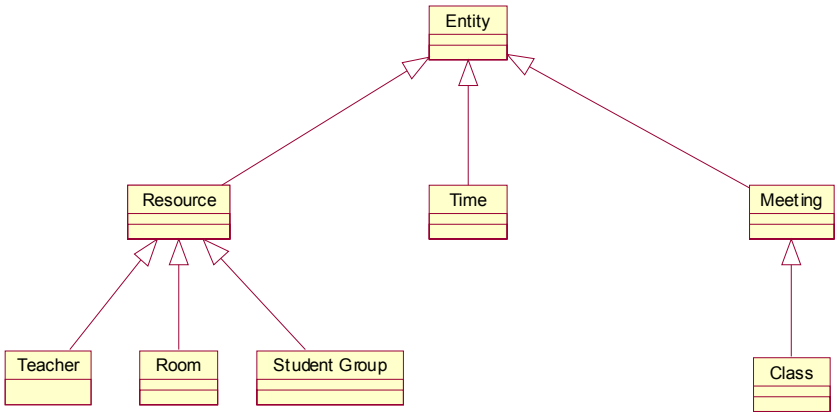


Figure 6: Kingston's implicit timetabling ontology

the timetabling ontology he implicitly uses, can be represented as in Figure 6.

After agreeing on the concepts, the definitions of these concepts and the relations between the concepts, the ontology can be translated to a formal language such as DAML+OIL.

In **Example 6** we translated some of the concepts of Figure 5. There are three different prefixes in this example `daml`, `rdfs` and `oiled`. DAML+OIL took the best of all available technologies: it uses parts of RDF and RDFS. The prefix `oiled` is generated by the OilEd [47] editor we applied to construct the ontology.

```
<daml:Class rdf:about="resource.daml#teachingassistant">
   <rdfs:label>teachingassistant</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
     <oiled:creationDate>
      2002-04-17T09:26:59Z
     </oiled:creationDate>
   <rdfs:subClassOf>
      <daml:Class rdf:about="resource.daml#teacher"/>
   </rdfs:subClassOf>
</daml:Class>
<daml:Class rdf:about="resource.daml#lecturer">
   <rdfs:label>lecturer</rdfs:label>
   <rdfs:comment><![CDATA[]]></rdfs:comment>
```

```
      <oiled:creationDate>
       2002-04-17T09:29:25Z
      </oiled:creationDate>
    <rdfs:subClassOf>
       <daml:Class rdf:about="resource.daml#teacher"/>
    </rdfs:subClassOf>
</daml:Class>
<daml:Class rdf:about="resource.daml#lecture room">
    <rdfs:label>lecture room</rdfs:label>
    <rdfs:comment><![CDATA[]]></rdfs:comment>
      <oiled:creationDate>
       2002-04-16T15:38:25Z
      </oiled:creationDate>
    <rdfs:subClassOf>
       <daml:Class
        rdf:about="resource.daml#reusable resource"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Class
       rdf:about="resource.daml#batch capacity resource"/>
    </rdfs:subClassOf>
</daml:Class>
<daml:Class rdf:about="resource.daml#associateprofessor">
    <rdfs:label>associateprofessor</rdfs:label>
    <rdfs:comment><![CDATA[]]></rdfs:comment>
      <oiled:creationDate>
       2002-04-17T09:28:41Z
      </oiled:creationDate>
    <rdfs:subClassOf>
       <daml:Class rdf:about="resource.daml#teacher"/>
    </rdfs:subClassOf>
</daml:Class>
```

This translated ontology is ready for the Semantic Web. In this way, fully automatic timetabling systems become conceivable. Suppose that a software agent is roaming the Semantic Web to find a solution for a specific timetabling problem. Imagine that this agent finds a web page about timetabling, where the creator added a link to a timetabling ontology, and that this page contains a solution to the problem. On the basis of this ontology the software agent can interpret the page and use it to solve his own problem.

## 5 Discussion

It is our opinion that future timetabling applications should be built with the ideas of the Semantic Web in mind. Our future work concentrates on establishing a timetabling application that consists of separate components working together [48]. These components are linked together through the use of an ontology. Montana [49] introduced a web-based timetabling application, the only similar approach we are aware of.

Di Gaspero et al. [50] report on an Object-Oriented Framework called EasyLocal++ that is designed in 4 abstract layers. The top layers rely on services supplied by the lower levels, analogous to the Semantic Web (Figure 1). In EasyLocal++ the representation of the problem is handled in the lowest layer and has to be defined by the user of the system. The next layer, consisting of the so-called 'helpers', is responsible for performing actions related to specific aspects of the search. The third layer, where the 'runners' reside, contains the algorithmic core. Each runner has a connection with the second layer in order to invoke the helpers for carrying out problem-related tasks. The upper layer contains the solvers that are responsible for the search. They generate initial solutions and decide in which order runners have to be activated.
We consider using EasyLocal++ as a starting point for the component based timetabling system.

Although university and exam timetabling problems are occurring all over the world [51, 52, 53, 54], there exists no generally satisfying solution strategy. That is because the demands, resources and constraints change from university to university. This bold statement is the standing ground for the upper-level ontology for timetabling (Figure 4). We will develop different separate components, such as algorithmic, demand, resource and constraint components. These components can then be published as web services. Applications can call these components through SOAP, which allows running them on a distributed environment. A CPU-intensive algorithmic component could, for instance, run on a fast UNIX server, while the other components can run on a simple WINTEL combination. Schemas can be defined in order to offer data to the components in a readable format. By using a DOM parser, the results of the algorithm can be transferred to XML documents. The XML results can be presented in a browser by applying XSL and CSS.

Each personnel scheduling problem has specific constraints and requires adapted approaches to solve it [55, 56, 57, 41]. In different sectors, we distinguish a different attitude towards cyclical schedules, overtime, locations, qualifications, etc. Just like in the university timetabling problem discussed in the previous paragraph, the existing knowledge and approaches could become available for any personnel scheduling environment by making use of the Semantic Web.

Ontologies have been used to support fast application generation [58]. Instead of starting from scratch when developing a timetabling application that suits the demands, it might be easier to adapt or extend the domain ontology in order to satisfy the preferences. In this way software components can be reused, since domain ontologies and generic problem solving methods provide the right kind of abstraction for building timetabling applications. In contrast with traditional OO techniques in which methods (program code), classes and instance objects (the representation of the domain model) are closely interrelated, these are strictly separated by using domain ontologies and problem-solving methods. This minimises the effort of updating the timetabling applications when requirements change [46]. Another advantage is that any new algorithm with better performance can be plugged in as a problem-solving method. Finally, using ontology in system design makes it function as a breeding ground from

which formal domain descriptions may originate. They can be applied for communication between independently developed systems.

## 6 Conclusion

By applying the ideas of the Semantic Web to the timetabling domain, researchers will have more opportunities to concentrate on new technologies. Instead of repeating and adapting already existing solutions to their problems, they will focus fully on the development and improvement of search algorithms.
The Semantic Web will contribute to simplifying the very time-consuming optimisation problems. By using a common ontology, timetabling problems will easily be recognised, classified and left with the care of software agents that will search appropriate solution techniques on the web. Applying these web technologies will require an effort from the users. They will either have to make their own ontology, refer to, or modify an already existing ontology. In order to make their data available on the web, it has to be machine-readable. However, the overall profit will be high. The available problem ontology will enable widespread co-operation with other systems.

## References

[1] T. Berners-Lee, *Weaving the Web*, Harper Business, 1999, 006251587X.
[2] Wren, A. *Scheduling, timetabling and rostering - a special relationship?* in: E.K. Burke & P. Ross (editors) The Practice and Theory of Automated Timetabling, pp. 45-75 Springer-Verlag. 1996.
[3] T. Berners-Lee, *Semantic Web Roadmap, Internal Note*, World Wide Web Consortium, 1998. (http://www.w3.org/DesignIssues/Semantic.html)
[4] W3C Semantic Web Activity Statement, 2001. (http://www.w3.org/2001/sw/Activity)
[5] T. Berners-Lee, *The World Wide Web: Past, Present and Future*, August 1996. (http://www.w3.org/People/Berners-Lee/1996/ppf.html)
[6] T. Berners-Lee, James Hendler and Ora Lassila, *The Semantic Web*, Scientific American, May 2001. (http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html)
[7] J. Hendler, *Agents and the Semantic Web*, in IEEE Intelligent Systems & their applications, Vol. 16, No. 2, March/April 2001.
[8] T. Berners-Lee, *Semantic Web*, presentation at XML 2000 conference, 2000. (http://www.w3.org/2000/Talks/1206-xml2k-tbl/)
[9] S. B. Palmer, *The Semantic Web, Taking Form.* (http://infomesh.net/2001/06/swform/)
[10] T. Gardner, *An Introduction to Web Services* (http://www.ariadne.ac.uk/issue29/gardner/)
[11] A. Swartz and J. Hendler, *The Semantic Web: A Network of Content for the Digital City* (http://blogspace.com/rdf/SwartzHendler)

[12] T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 2000. (http://www.w3.org/TR/REC-xml)

[13] D. C. Fallside, *XML Schema Part 0: Primer*, W3C Recommendation, 2001. (http://www.w3.org/TR/xmlschema-0/)

[14] T. Bray, D. Hollander and A. Layman, *Namespaces in XML*, World Wide Web Consortium Recommendation, 1999. (http://www.w3.org/TR/REC-xml-names/)

[15] J. Clarck, *XSL Transformations*, W3C Recommendation, 16 November 1999. (http://www.w3.org/TR/xslt)

[16] H. W. Lie, B. Bos, *Cascading Style Sheets, level 1, W3C Recommendation*, 17 December 1996, revised 11 January 1999. (http://www.w3.org/TR/REC-CSS1)

[17] O. Lassila and R. R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification,* W3C Recommendation, 1999. (http://www.w3.org/TR/REC-rdf-syntax/)

[18] J. Broekstra, M. Klein, S. Decker, D. Fensel and I. Horrocks, *Adding formal semantics to the Web: building on top of RDF Schema* In Proc. SemWeb 2000.

[19] D. Brickley and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0,* W3C Candidate Recommendation, 2000. (http://www.w3.org/TR/rdf-schema/)

[20] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, *OIL: An Ontology Infrastructure for the Semantic Web*, in IEEE Intelligent Systems & their applications, Vol. 16, No. 2, March/April 2001.

[21] T.R. Gruber, *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition, Vol. 5, pp.199-220, 1993.

[22] http://www.isi.edu/isd/LOOM/LOOM-HOME.html

[23] A. Valente, T. Russ, R. MacGregor and W. Swartout, *Building and (Re)Using an Ontology of Air Campaign Planning*, In IEEE Intelligent Systems 14:1, pp. 27-36, Jan-Feb, 1999.

[24] R. Patil, R. Fikes, P. Patel-Schneider, D. McKay, T. Finin, T. Gruber and R. Neches, *The DARPA knowledge sharing effort: Progress report*, 1992.

[25] http://www.cyc.com/cycl.html

[26] A. Farquhar, R. Fikes, and J. Rice, *The Ontolingua server: A tool for collaborative ontology construction*, technical report, Stanford KSL, pp. 96-26, 1996.

[27] I. Horrocks, F. van Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, L. Andrea Stein, *DAML+OIL*, March 2001. (http://www.daml.org/2001/03/daml+oil-index.html)

[28] http://www.w3.org/2001/sw/news#x20020731a

[29] DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), *DAML-S: Semantic Markup for Web Services*, in Proceedings of the

International Semantic Web Working Symposium (SWWS). July 30-August 1, 2001.

[30] http://www.daml.org/services/

[31] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer, *Simple Object Access Protocol (SOAP) 1.1,* W3C Note, 2000. (http://www.w3.org/TR/SOAP/)

[32] http://www.xmethods.net/

[33] T. Sollazzo, S. Handschuh, S. Staab, M. Frank, *Semantic Web Service Architecture —Evolving Web Service Standards toward the Semantic Web,* Proc. of the 15th International FLAIRS Conference, Pensacola, Florida, May 16-18, 2002. AAAI Press.

[34] S. A. McIlraith, T.C. Son, H. Zeng, *Mobilizing the Semantic Web with DAML-Enabled Web Services*, in Proceedings of the 2nd International Workshop on the Semantic Web, pp. 82-87, Hongkong, China, May 1, 2001.

[35] E. K. Burke, P. A. Pepper and J. H. Kingston, *A Standard Data Format for Timetabling Instances*, In Practice and Theory of Automated Timetabling II (selected papers from Proceedings of the Second International Conference on Practice and Theory of Automated Timetabling, Toronto 1997) Springer Lecture Notes in Computer Science 1408, pp. 213-222, 1997.

[36] J. H. Kingston, *Modelling Timetabling Problems with STTL*, In Practice and Theory of Automated Timetabling III (selected papers from Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Konstanz 2000) Springer Lecture Notes in Computer Science 2079, pp. 309-321, 2001.

[37] L. Wood, V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, *Document Object Model (DOM) Level 1 Specification*,Version 1.0, W3C Recommendation, 1998. (http://www.w3.org/TR/REC-DOM-Level-1/)

[38] L. A. Zadeh, *Fuzzy Sets*. Information and Control, vol. 8 pp. 338-353, 1965.

[39] L. A. Zadeh, *Fuzzy Logic*. Computer, vol.21, no. 4, pp. 83-93, April 1998.

[40] L. A. Zadeh, *Knowledge Representation in Fuzzy Logic*. IEEE Transactions on Knowledge and Data Engineering 1(1): pp. 89-100, 1989.

[41] H. Meyer auf'm Hofe, *Solving Rostering Tasks as Constraint Optimization*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, pp. 191-212, 2000.

[42] http://www-db.stanford.edu/~melnik/rdf/api.html

[43] http://www.hpl.hp.com/semweb/

[44] http://grcinet.grci.com/maria/www/codipsite/Tools/Components.html

[45] S. F. Smith and M. A. Becker, *An Ontology for Constructing Scheduling Systems*. In Working Notes from 1997 AAAI Spring Symposium on Ontological Engineering, Stanford, CA, March 1997.

[46] M. A. Musen, *Ontology-oriented design and programming*. In J. Cuena et al., Ed., Knowledge Engineering and Agent Technology. IOS Press, Amsterdam, 2000.

[47] http://oiled.man.ac.uk/

[48] http://project.kahosl.be/cofftea

[49] D. J. Montana, *Optimized Scheduling for the Masses*, Genetic and Evolutionary Computation Conference (GECCO-2001), Workshop on The Next Ten Years of Scheduling Research.

[50] L. Di Gaspero and A. Schaerf, *EasyLocal++: An Object-Oriented Framework for the Design of Local Search Algorithms and Metaheuristics*, MIC' 2001, the fourth Metaheuristics International Conference, pp. 287-291, 2001.

[51] E.K. Burke, J.P. Newall, R.F. Weare, *A Memetic Algorithm for University Timetabling*, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference Edinburgh, Springer, 1995.

[52] M.W. Carter, *A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, pp. 64-82, 2000.

[53] B. Paechter, A. Cumming, M.G. Norman, and H. Luchian, *Extensions to a Memetic Timetabling System*, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, pp. 251-265, 1995.

[54] G.M. White, B.S. Xie, *Examination Timetables and Tabu Search with Longer-Term Memory*, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, pp. 85-103, 2000.

[55] U. Aickelin, K. Dowsland, *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem*, Journal of Scheduling, Volume 3 Issue 3, pp. 139-153, 2000.

[56] E.K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe, *A Memetic Approach to the Nurse Rostering Problem*, Applied Intelligence special issue on Simulated Evolution and Learning, Vol. 15, Number 3, Springer, pp. 199-214, 2001.

[57] A. Meisels, E. Gudes, G. Solotorevski, *Combining rules and constraints for employee timetabling*, Journal of Intelligent Systems, Vol. 12, pp. 419-439, 1997.

[58] W. Ceusters, *Terminology and Ontology Management Systems*, Semantic Web and Applications 2002 seminar, Ghent. (http://project.kahosl.be/swa2002/slides/werner%20ceusters/swa2002ceusters.ppt)