Semantic Web: Vision and Technologies

Nick Bassiliades
Dept. of Informatics, Aristotle University of Thessaloniki, Greece
nbassili@csd.auth.gr

***Abstract**: This paper introduces the vision behind the Semantic Web by using illustrative examples of how interaction with the future Web will be through the use of intelligent personal agents. Furthermore, the paper overviews the current Semantic Web technologies that will carry out this vision. Finally, the paper briefly presents the research on Semantic Web carried out at the Department of Informatics of the Aristotle University of Thessaloniki.*
***Keywords:** Semantic Web, Intelligent Agents, Metadata, Ontologies, Logic, Inference*

1. INTRODUCTION

Semantic Web is the next step of evolution for the Web [7], where information is given well-defined meaning, enabling computers and people to work in better cooperation. Currently information found on the Web is mainly for human consumption and is not machine-understandable. It is quite difficult to automate things on the Web and, because of the volume of information the Web contains, it is even more difficult to manage it manually.

In this paper, we introduce the vision behind the Semantic Web in section 2, we illustrate with an example how the interaction with personal, intelligent, Semantic Web agents will be in the future in section 3, we describe the Semantic Web architecture in section 4 and we overview the current Semantic Web technologies that will carry out the vision, in section 5. Finally, in section 6 we conclude the paper and we present briefly the research on Semantic Web carried out at the Department of Informatics of the Aristotle University of Thessaloniki.

2. THE SEMANTIC WEB VISION

Tim Berners-Lee, the inventor of the World Wide Web (WWW) [8] back in 1989, had the vision of a web of data, automatically processable by machines, based on the meaning (semantics) of the information and not on its form/appearance, as it is clearly illustrated by his definition of the Semantic Web: *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation"* [7].

Therefore, the main aim of the Semantic Web is to organize the information found in the Web (mainly in Web pages) in a better fashion and interconnect the various pieces of information so that the very same information can be used for discovery, automatization, aggregation, and reuse from various, different and disparate applications, that have not been designed either to work together or to work with every different piece of information found in the Web. Sometimes, this vision of the Semantic Web is described as an effort to turn the Web into a gigantic database.

The Semantic Web will achieve its purpose by extending/enhancing the current web and not by replacing it. This is dictated by the huge success of the current Web and its large penetration in everyday people's work and entertainment. The extension of the

current Web will revolve around defining information in such a way so that it becomes meaningful not only to people but also to web-based applications, which are mainly intelligent agents and web services.

The impact of the Semantic Web, when it is realized, will be enormous both for people and business, because it will achieve interoperability of information between web applications (agents, web services, etc.). This roughly means that various disparate applications will exchange data seamlessly even if these applications have not been designed to work together. This exchange, however, does not only involve data exchange with a predefined meaning, as it is the case of XML today, but also common understanding of the meaning of data on a non-predetermined basis. The Web of the future will also become easier for people to use because they will be able to describe in their own terms to their personal agent-assistant what they are looking for in the Web or what they want to achieve on the Web and the agent will make use of the Semantic Web infrastructure to achieve its user demands.

Today Web largely involves human interactivity. Web pages are mainly built for human consumption. The current technology (HTML language) is concerned with how to present information. Issues like the actual content and structure of the information and/or the meaning of information cannot be handled by current web servers. Even fairly structured information, such as databases, when exported to the Web they lose their structure, which should be re-discovered by intelligent text-processing agents, called *wrappers*. What is missing from today's Web is the definition of semantic information that will make the meaning of the information accessible both to applications and people alike.

An illustrative example of the shortcomings of the current Web is search engines, which base the retrieval of documents relevant to the user's query to keywords found in the text of the document. The main problem associated with this approach is that most of the times too many irrelevant documents are recalled that contain the desired keyword. Furthermore, some times (although much more rarely) no or very few documents are recalled when the search involves an infrequently used keyword. However, if the search was made with a more frequently used synonym of the original word, many more documents would have been recalled. The above problems indicate that the success of keyword-based search is largely sensitive to the vocabulary used both for the search and in documents.

Another problem with today's search engines is that the results returned must be manually inspected by the human for their relevancy. Furthermore, the human must also manually select certain pieces of the information found in such documents in order to combine them into new forms and settings and use them to achieve his current task. The above everyday procedures, though trivial, they are largely time consuming.

In order to solve the above problems and attach or find the hidden meaning of web documents automatically, two solutions have been proposed. The first one involves Natural Language Processing and dictates the textual analysis of the documents in order to find the meaning of the text. However, this solution is very ambitious and far from being implemented with a high degree of accuracy in the next few decades. Automatic language understanding is the Holy Grail of Artificial Intelligence and, despite the huge progress made by scientists the last couple of decades in this area, full-proof language understanding software has not been and probably will not be available in the near future.

The solution discussed in this paper is Semantic Web, which is an engineering issue rather than a deep scientific one. The Semantic Web solution instead of trying to under-

stand text, it attaches additional information to it that semantically marks the information found in the Web. This additional information is called *metadata*. Furthermore, the Semantic Web solution involves fairly low intelligent techniques and applications that exploit such metadata in order to achieve the global information interoperability vision for the Web.

3. SEMANTIC WEB USE CASE

The vision of the Semantic Web concerning the future interaction with the Web through personal intelligent agents can be better illustrated through an example. Assume that a woman, let's call her Jane, is pregnant and visits her gynecologist for her regular monthly examination during pregnancy. The doctor examines her through ultrasound and he gets suspicious that something may be wrong. However, he cannot make an accurate diagnosis because the resolution of his ultrasound equipment is low. Therefore, he suggests that Jane should be further examined by better medical ultrasound equipment with higher resolution. Such equipment exists only at hospitals or diagnostic medical centers.

Jane, when she comes back home, assigns to her personal Semantic Web agent to find such a center and book an appointment for a test. She prefers a medical center affiliated with her insurance organization, so that she does not have to pay for the examination, which costs a lot.

Fig. 1 shows the interaction that takes place between Jane, her agent, and the agents of the rest of the entities involved, namely: the gynecologist, the insurance organization, the medical centers, etc. Initially, Jane’s agent retrieved from the doctor’s agent the formal description of the medical test that Jane should undergo. Then, Jane’s agent asked the agent of the insurance organization for affiliated hospitals and medical centers able to perform such an examination, selected those that are close to Jane’s work or home, e.g. within a radius of 5 Km (Fig. 1a) and checked on their reputation, based on an independent rating agency (Fig. 1b) that is trusted by the Ministry of Health to rate doctors, hospitals, etc.

After the agent came up with some medical centers that satisfy certain quality criteria, the agent contacted them to schedule an appointment, consulting also with the personal work schedule of Jane (Fig. 1b). Finally, the agent came up with two solutions (Fig. 1c), none of which satisfied Jane. In the first case, the earliest available appointment was in two weeks time, which was too late, because the examination should be performed within one week at the latest. In the second case, the earliest available appointment was in three days; however, Jane had to stay in the city center three hours after work, which would be exhausting for her. Jane decided to re-ask her agent with stricter time limits this time.

Her agent came up with a new solution, within the new time limits, which however violated some other constraints. There was an available appointment for the next day with a medical center of an excellent reputation, near Jane’s home, which however was not affiliated with Jane’s insurance organization because it charges more than the maximum insurer’s coverage (Fig. 1e). The agent checked that the insurance organization will reimburse Jane with the maximum coverage; therefore, she had to pay only a few extra euros (Fig. 1d). The medical center was found through an independent list of medical centers listed on a popular medical web directory (Fig. 1d).

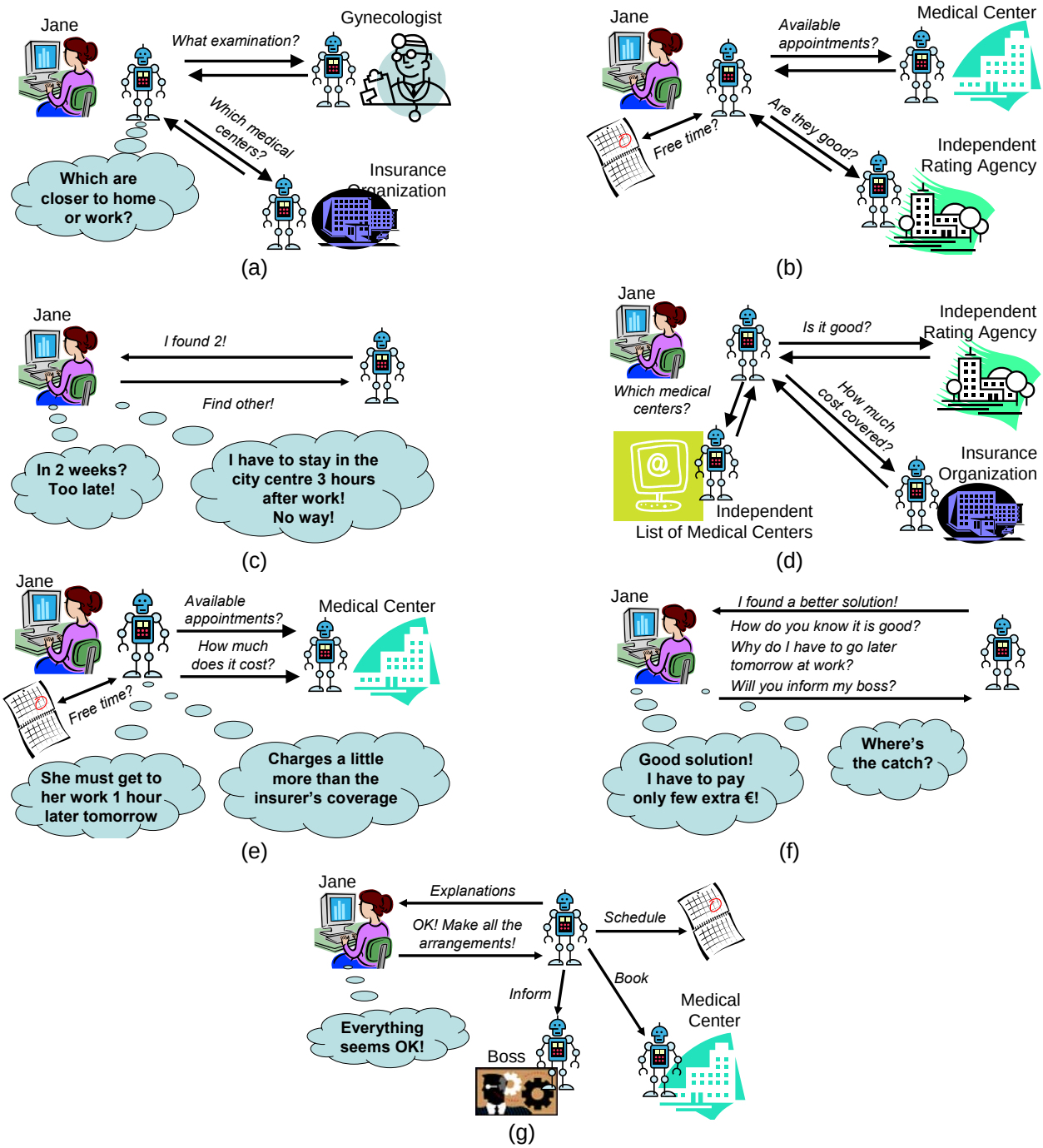

Fig. 1: Interaction between the Semantic Web agent, the user and the rest of the world.

Jane found the solution satisfying, although the appointment would make her late at work the next day (Fig. 1f). The agent offered to inform her boss, so Jane asked her agent to make all the necessary arrangements (Fig. 1g).

The use case described above may seem like science fiction, but its implementation is not very far away within the abilities of today's technology! Its realization is not a matter of new scientific discoveries, but rather a technological issue that requires adoption by the users of the Semantic Web technologies [2], described later in the paper.

4. SEMANTIC WEB ARCHITECTURE

In order to achieve steady adoption of the various technologies involved in the Semantic Web, its development proceeds in layers (Fig. 2). Each layer is built on top of technologies built in the previous layer(s), so that downward compatibility, concerning languages and tools, is preserved. After each layer is standardized and widely accepted by users and organizations, standardization efforts begin for the next layer.

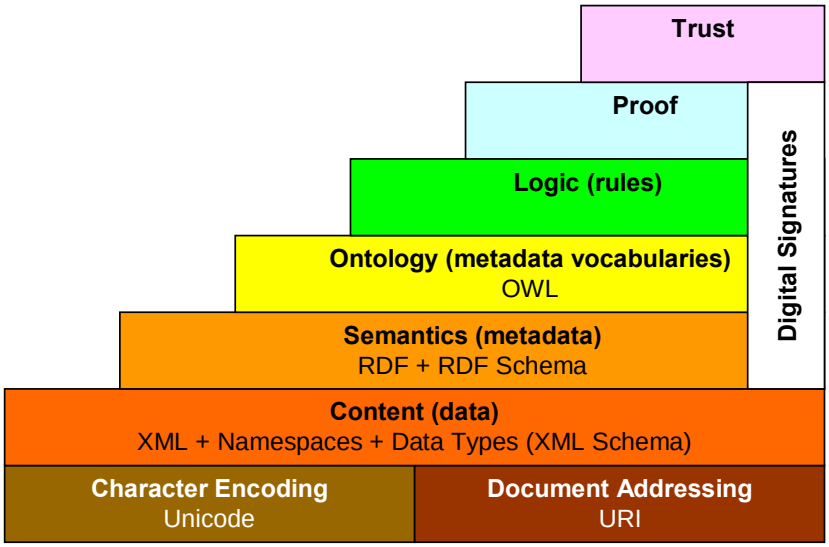

Fig. 2: Semantic Web Architecture.

At the lowest layer is *character encoding*, through Unicode, and *document addressing*, through the URI scheme that also includes URL addresses. On top of it lies the *content layer*, which includes the typing and structuring of web documents through XML (eXtensible Markup Language). XML is the current W3C standard for exchanging documents in the Web through a universally syntactically comprehensible language. Notice, that the widely known HTML language that is used by browsers to visually render a web page is a subset of XML. Furthermore, this layer includes data typing information for XML documents through XML Schema and namespaces, which is a mechanism for setting up and addressing collections of multiple resources within a single document.

The next layer is the *semantics layer*, where metadata expressed in RDF augment the content layer. The allowed metadata vocabularies and their intended meaning are given through the *ontology layer* which includes the OWL Web ontology language [11] and is the highest layer today that has reached a certain level of maturity. Notice that RDF Schema is also a simple ontology language that has been traditionally placed at the semantics layer.

The *logic layer* consists of logic-based rules that (a) can serve as extensions of, or alternatives to, description logic based ontology languages; and (b) can be used to develop declarative systems on top of (using) ontologies. The logic layer is accompanied by the *proof layer* which contains all the necessary inference mechanisms that manipulate the rules of the logic layer and, furthermore, explicitly expose the proofs of inferences in order to generate explanations between proof systems, agents and humans.

Finally, the *trust layer* lies at the top of the pyramid, since trust is very important for the adoption of the Semantic Web by users. The Web, in general, and the Semantic Web with all those task delegations to agents, in particular, will be fully utilized only

when users trust it completely. Operations and tasks performed on the Web should be secured and private. Digital signatures at all Semantic Web layers can partly help to achieve this. Trust derivation will be needed next in order to automate trust management in the Web. Finally, the quality of offered information and web services is also a matter of trust. Independent Rating and Certification Agencies that will qualify web resources will come in need, should we choose to conduct business on a *Web of Trust.*

5. SEMANTIC WEB TECHNOLOGIES

The main Semantic Web technologies that will support its vision are metadata, ontologies, logic and inference, and finally, intelligent agents. In the next subsections we briefly overview each of the above technologies.

5.1. Metadata

Metadata are data that give additional information about other data. For example, the title of a web page or the name of its creator can be metadata about the web page. Other examples of metadata are the date of the latest modification of a web page or the timetable for the availability of each offered service that the web-based information system of the tax office offers to the public.

The current official W3C language (standard) for describing metadata for the Web is RDF (Resource Description Framework), which is rather a data model that allows the declaration of additional information (or properties) about existing web resources, e.g. Web pages, Web services, etc. The main construct of RDF is the *statement*, which asserts a property value for a resource and is composed by a *subject* (the resource for which the property value is asserted for), a *predicate* (which is the name of the property) and an *object* (which is the value of the property for the specific resource subject). An example of an RDF statement is the following:

web page http://ex.com/~jsmith is owned by John Smith

where the subject of the statement is ***http://ex.com/~jsmith***, the predicate is ***is owned by*** and ***John Smith*** is the object.

Fig. 3 shows the representation of an RDF statement as a directed, labeled graph, where the subject and the object are nodes and the predicate is an arc, directed from the subject node to the object node. Many RDF statements form a *semantic net* (Fig. 4).

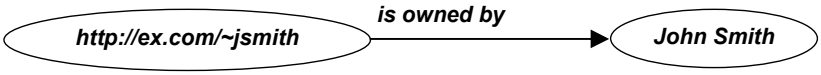

Fig. 3: Graph representation of an RDF statement.

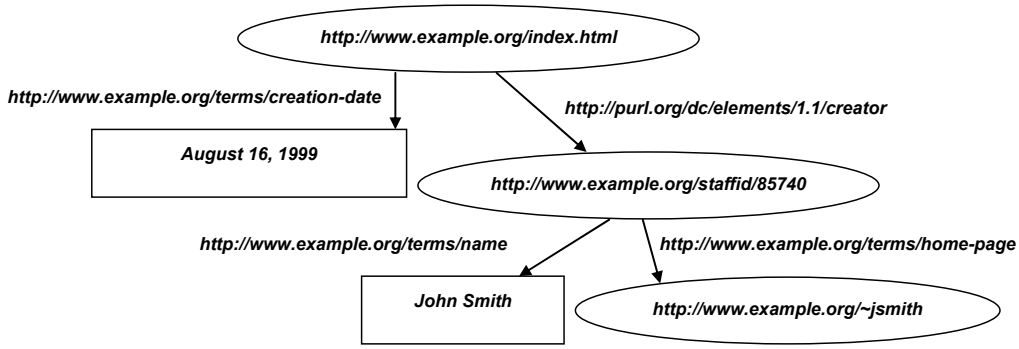

Fig. 4: Multiple RDF statements that form a semantic net.

The subject of a statement is always a resource, whereas the object can either be a literal value (e.g. a date, a number, or a string) or another resource. Predicates are also considered as resources. Resources are uniquely identified in the web with URIs (Uniform Resource Identifiers), even if they do not correspond to accessible web resources. For example, ***John Smith*** can be considered as a web resource and have a URI associated with him. Furthermore, the predicate ***is owned by*** can be considered as a term declared in a specific web document, therefore it can also be represented by a URI. The use of URIs offers clarity of attribution and referencing in the vast space of the web. If we replace the predicate and the object with URIs, the previous statement becomes:

http://ex.com/~jsmith http://terms.org/owned_by http://ex.com/staff.html#id1734

Most of the times, the same web document consists of several smaller components or resources, or it declares several terms. Therefore, the full URI of a resource consists of a common prefix (the address of the containing document) and an internal local name (or anchor) in the form ***address#localName***. Namespaces are aliases for those address prefixes and they are commonly used to collect resources within the same context and can also be used to shorten the URIs of the resources, in the form ***namespace:localName***. For example, if we assign the prefix ***staff*** to the URI ***http://ex.com/staff.html*** and the prefix ***term*** to the URI ***http://terms.org/*** the above statement becomes:

staff:id1734 terms:owner http://ex.com/~jsmith

RDF is located on top of XML in the Semantic Web architecture (Fig. 2), therefore RDF has XML syntax. However, this syntax is used for tool compatibility reasons, since the two languages have completely different semantics. For example, the RDF statement example in RDF/XML syntax is represented as follows:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/...rdf-syntax-ns#"
         xmlns:term="http://terms.org/">
  <rdf:Description rdf:about="http://ex.com/~jsmith/">
    <term:owned_by rdf:resource="http://ex.com/staff.html#id1734"/>
  </rdf:Description>
</rdf:RDF>
```

The main difference between RDF and XML is that the latter does not define the meaning of data, except when there is a commonly agreed meaning between the exchanging parties. On the other hand, RDF has a formally defined way to interpret statements. To illustrate the inability of XML to represent semantics, consider the following XML document example:

```
<faculty>
  <lecturer>
    <name>John Smith</name>
    <teaches>
      <course>
        <name>Algorithms</name>
        <student>aem153</student>
        <student>aem202</student>
        …
      </course>
    </teaches>
  </lecturer>
</faculty>
```

There is no standard interpretation of the tag nesting mechanism of XML. The nesting of tag ***A*** in ***B*** can be interpreted in three different ways:

- ***A*** is a part of ***B***

- ***A*** is a subset of ***B***
- ***A*** is a property of ***B***

In the previous example, the nesting of the `lecturer` tag inside the `faculty` tag is actually a subset relation, the nesting of the `name` tag inside either the `lecturer` or the `course` tags is a property relation, and finally, the nesting of the `student` tags inside the `course` tag implies that students are part of a course. In RDF and RDF Schema there is a standard interpretation for all those relations, as we will see later.

5.2. Ontologies

All resources are not of the same kind. For example, a web page is different in nature than a human URI or a web service. Therefore, resources can be grouped into categories, which determine the number, names and types of properties that can be attached to them. Simple RDF does not provide solutions for the classification of resources; therefore, there is a need for a language for defining which terms, properties, constraints, etc. of resources are allowed, i.e. the allowed metadata *vocabulary*.

In its most complex form this vocabulary is also called *ontology*. *Ontology* is a formal representation of a domain of discourse. An ontology usually includes the following:

- *Terms*, which are classes of objects, also called concepts or prototypes. Examples, of terms/classes are faculty members, staff, students, courses, departments in a University setting.
- *Properties* of terms, e.g. professor ***X*** teaches course ***Y***.
- *Constraints* of properties, e.g. courses are taught only by faculty members.
- *Equivalence* and *disjointness* relations, e.g. a staff member can be nothing else but one of either a faculty member or a member of the technical staff or a member of the administrative staff.
- *Semantic relationships*, such as hierarchical dependencies between terms. For example, all faculty members are also staff members.
- *Logical relationships* between objects, e.g. every department must have at least ten faculty members.

Ontologies can be used for:

- Checking if an object belongs to a class in a hierarchy, by following the hierarchical class relationships. For example, if an object is a postgraduate student, then he/she is a student as well.
- Checking if two classes (or objects) are equivalent, by using the equivalence - disjointness relations of classes or objects.
- Inferring logical conclusions that can be drawn from ontology declarations and constraints. For example, if a person has two mothers, since the mother relationship is unique, it can be inferred that the two different mothers actually represent the same person.
- Checking the consistency of the ontology, when some declarations and constraints are contradicting or mutually exclusive.
- Automatically classifying objects in a hierarchy of classes from their property-value pairs by comparing those pairs to the constraints attached to classes.

One of the most prominent uses of ontologies for the web will be searching for documents and resources, improving today's keyword-based mechanism of search engines. Keyword-based search will be enhanced by connecting words to their intended meaning, improving the accuracy of searches. For example, consider a search with the

word "model" that may recall pages related to fashion, software models (a software engineering issue) and car models. In ontologies, keywords can be hierarchically connected to their intended meaning, so the user may select only one of those meanings, improving the semantic accuracy (precision) of the search. Furthermore, since for each word there are synonyms with the same meaning, semantic-based search can also recall pages that contain words that are not explicitly present in the query, but are semantically related to them. For example, if the word "model" is used as in “car model” or “car type” then a synonym for it can be the word “brand”.

There are two such standard languages proposed by the W3C that will be described next: RDF Schema, which is a very simple vocabulary language, and OWL, which is considered a full-fledged ontology language.

5.2.1. RDF Schema

RDF Schema is an extension of RDF for describing allowed metadata *vocabularies* and is considered as a very simple ontology language. In RDF Schema, similar resources are grouped into *classes*, which are organized in inheritance hierarchies. The resources that are members of a class are called *instances*. When a class is a subclass of another class, then all its instances belong to the superclass as well.

Properties describe certain features of resources, such as title, creator, etc. Properties in RDF are global, i.e. they are defined outside of classes and they are independent. When a property is declared, its domain and value range can be defined. The domain of a property is a class, which means that this property is attached to (characterizes) members of that class. The range of a property defines the allowed values the property can take, which can be either members of another class or literal values (e.g. numbers, strings, etc.). Properties can also be organized in hierarchies. For example, the father property is a subproperty of the parent property, which means that every father is also a parent, but not vice versa.

Below we include an RDF Schema example, expressed in triple statements:

```
staff:id1734 rdf:type uni:lecturer
uni:lecturer rdf:type rdfs:Class
uni:lecturer rdfs:subClassOf uni:faculty
uni:teaches rdf:type rdf:Property
uni:teaches rdfs:domain uni:faculty
uni:teaches rdfs:range uni:course
```

In the example above, a member of class `lecturer` is defined. Furthermore, it is stated that `lecturer` is a subclass of class `faculty`. Finally, a property `teaches` is defined that has the classes `faculty` and `course` as domain and range, respectively.

Here the descriptive semantics of RDF [12] must be clarified against the well-understood prescriptive semantics of OO programming languages and databases. In an OO programming language, property domains and ranges are interpreted as *constraints*. For example, if the property `author` of class `book` has class `person` as range, then the statement "*My cat is an `author` of that book*" will raise an error if such a transaction is attempted. However, in RDF domains and ranges are used for determining the class of a resource instead of prohibiting assertions. For example, the same statement would imply that my cat is rather an instance of class `person`! Although such a treatment would seem ill-designed, the Web is an open, free space, where constraints cannot really be enforced and everyone is allowed to state whatever they like. Therefore, the way the truth of the above statement will actually be treated is a matter of trusting its source. Concerning RDF, its semantics is clear.

5.2.2. OWL

OWL (Web Ontology Language) is the official W3C language for Web ontologies and has been designed as a syntactical extension of RDF Schema. Formally, however, only OWL Full (one of the OWL flavors) is an extension of RDF. OWL, as RDF, includes class descriptions, property definitions, and class instances. However, OWL also gives the ability for logical reasoning, i.e. for producing facts that have not been initially declared, but can be concluded using its semantic rules. OWL is actually a subset of predicate logic, called *description logic*.

The advanced representation capabilities of OWL include:

- Definition of “local” constraints for property domains and ranges. For example, although courses are generally taught by faculty members, first-year courses must be taught by professors.
- Mutually exclusive (or disjoint) classes, e.g. men - women.
- Combination of classes using set (Boolean) operators, such as union, intersection, etc. For example, male persons belong to a class that is an intersection of male and person classes.
- Cardinalities of properties that place restrictions on the number of values a certain property can have, as in "*a PhD student has 3 supervisors*".
- Properties with special features that are very commonly met when building an ontology, such as symmetric, transitive, and inverse properties.

In the following we include an ontology example in OWL. First we define the classes:

```
<owl:Class rdf:ID="course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#is_taught_by"/>
      <owl:cardinality rdf:datatype="&xsd;integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FirstYearCourse">
  <rdfs:subClassOf rdf:resource="#course"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#StudyYear"/>
      <owl:hasValue rdf:datatype="&xsd;integer">1</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The above definitions are interpreted as follows: Class `course` is a class whose instances must be taught by one faculty member. First-year courses are courses whose study year has value *1*.

The properties of the ontology are the following:

```
<owl:ObjectProperty rdf:ID="is_taught_by">
  <rdfs:domain rdf:resource="#course"/>
  <rdfs:range rdf:resource="#faculty"/>
  <owl:inverseOf rdf:resource="#teaches"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="StudyYear">
  <rdfs:domain rdf:resource="#course"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
```

Notice that there are two types of properties: object properties that link two resources together and data type properties that link a resource to a literal value.

Finally, the following is an instance of the `FirstYearCourse` class:

```
<FirstYearCourse rdf:ID="csd102">
  <is_taught_by rdf:resource="#fac101"/>
</FirstYearCourse>
```

5.3. Logic and Inference

Logic can capture semantic relationships between metadata more complex than ontologies. Using logic, in the form of rules, metadata can be processed in various useful ways. Rules usually arise naturally from the business logic of applications. Uses of logic and/or rules include the following:

- Logic (predicate logic) can capture some of the ontological knowledge expressed in description logics (e.g. OWL). For example, the fact that class **lecturer** is a subclass of **faculty** is represented as:

  lecturer(X) → faculty(X)
- Rules can be used to convert data and/or metadata documents from one format or type to another.
- Logic can be used to check the consistency of an ontology or a metadata document using the declarations and constraints of the ontology.
- Rules can be used to formulate questions in order to retrieve resources that belong to specified metadata terms or even to identify-classify unknown terms.
- Finally, intelligent Semantic Web agents can use logical rules for decision support and action selection among a number of possibilities, in a non-deterministic way.

One of the most important advantages of using logic is that it is easy for an agent to provide *explanations* to the user about its conclusions. Simply, the explanations are formed by the sequence of inference steps that the agent's inference mechanism has followed. Explanations are very important for the adoption of Semantic Web because they increase the confidence of users to agents' decisions.

The following example illustrates how explanations are going to be used among agents or between agents and people. Assume that a web banking agent asks from the user's agent permission to debit 658€ to his account in order to pay his monthly credit card bill. The user agent asks for the detailed bill to confirm all transactions. Supposedly the user's agent receives an official e-slip after each credit-card transaction, possibly deposited at a third, trusted, e-agency, in much the same way that we keep our credit card slips. The web banking agent provides the bill to the user's agent and the latter is able to verify all transactions but one that charges 28.75€. The charger for this transaction is the bank itself. So, the user's agent asks the bank agent for explanations about this charge.

The bank agent responds that it is the annual dues for the credit card account, by quoting the following rule from the bank's policy about credit cards, which is published at the web site of the bank:

billDate(CC,D) ∧ laterThan(D,01/09) ∧ annualDues(CC,AD) → charge(CC,AD)

The rule condition indicates facts and predicates that can be easily verified by the user agent, such as the billing date **D** for his account **CC** and the date and the amount for the annual dues, which can be found in the e-contract between the user and the

bank. The contract can be a web document, digitally signed by both parties, and possibly deposited at a notary e-agency.

However, the user's agent discovers that the annual dues are more than 25€, an amount that is mentioned in the contract and it has been paid during the previous years. So, it asks the bank agent to justify the amount for the annual dues, i.e. the **annualDues** fact. The bank agent explains that the contract states that every three years the bank retains the right to increase the annual dues by 15%, by quoting the following rule (included in the contract):

$$\textbf{contractDate(CC,D1)} \wedge \textbf{billDate(CC,D)} \wedge \textbf{yearDiff(D1,D,Y)} \wedge \textbf{Y} \geq \textbf{3} \wedge \textbf{currentAnnualDues(CC,AD)} \rightarrow \textbf{annualDues(CC,AD*1.15)}$$

The rule condition indicates facts and predicates that can be easily verified by the user agent trough the contract.

5.3.1. RuleML

The logic and inference layer of the Semantic Web has not been standardized yet. However there is the RuleML initiative, which tries to combine together all different industry needs for rules and rule types available in the academia. RuleML (Rule Markup Language) is an XML based language for the representation and exchange of rules in the Web [9], which supports different types of rules, such as derivation, transformation, and reaction rules, organized in a type hierarchy. RuleML can be used for querying and inferencing over web documents and ontologies, to provide mappings between ontologies, and to declare and execute dynamic Web behaviors of workflows, services, and agents.

RuleML is a generic extensible and semantically neutral rule markup language, mainly aimed towards the exchange of rules. Accordingly, RuleML programs are not intended to be executed directly, but the business logic of RuleML programs can be implemented via XSLT transformations into the target language of the recipient rule-based systems and then executed.

5.4. Agents

Agents are software entities that sense their environment with the use of sensors, which can be external operating system or internet programs. Being part of that environment, they reason about it and they act on it using actuators, which can also be external or internet programs, in order to achieve some goals, usually specified by their users-owners.

The most prominent features of agents are:

- Autonomy, i.e. agents are capable of acting independently, exhibiting control over their internal state.
- Social ability, i.e. the ability to interact with other agents (and possibly humans) via some kind of agent-communication language, and perhaps cooperate with others.
- Reactiveness, i.e. the ability to respond to changes that occur in the environment, in time for the response to be useful.
- Proactiveness, i.e. the ability to exhibit goal directed behavior by generating and attempting to achieve goals and not simply responding to events.

Some less prominent but still useful features of agents are: mobility, i.e. the ability of an agent to move around an electronic network, and adaptability, i.e. the ability to learn in order to improve their performance over time.

Personal agents on the Semantic Web (of the kind discussed in section 3) will be able to receive some tasks and preferences from their owner, seek information from Web sources, communicate with other agents, compare information about user requirements and preferences, make certain choices, and finally, give answers to the user by presenting alternative courses of actions and letting the user decide which is most preferable.

The technologies required for an agent in the Semantic Web have already been discussed in the paper and include:

- Metadata, in order to identify and extract information from Web sources.
- Ontologies, to increase the accuracy of Web searches, to allow them to interpret the retrieved information, and to be able to communicate with other agents.
- Logic, in order to process the retrieved information and to draw conclusions about the preferable courses of action.

Further technologies that agents will require, which are orthogonal to the Semantic Web technologies above, are [15]:

- Agent communication languages
- Formal representation of beliefs, desires, and intentions of agents
- Creation and maintenance of user models

6. CONCLUSIONS

In this paper we have introduced the vision behind the Semantic Web by using illustrative examples of how interaction with the future Web will be (using intelligent agents). Furthermore, we have overviewed the current Semantic Web technologies, namely metadata, ontologies, logic and agents, that will carry out this vision, and we have briefly presented the current or emerging standards, namely XML, RDF, RDF Schema, OWL and RuleML.

Finally, we briefly present the research on Semantic Web carried out at the Department of Informatics of the Aristotle University of Thessaloniki. Our research mainly revolves around development of languages and rule systems for the Semantic Web. More specifically, we have built the following systems:

- R-DEVICE [6], which is a deductive object-oriented knowledge base system that transforms RDF triples into objects and uses a deductive rule language for querying and reasoning about them. R-DEVICE imports RDF data into the CLIPS production rule system [10] as COOL objects. The main difference between the established RDF triple-based data model and our OO model is that R-DEVICE treats properties both as first-class objects and as normal encapsulated attributes of resource objects. In this way properties of resources are not scattered across several triples as in most other RDF querying/inferencing systems, resulting in increased query performance due to few joins. R-DEVICE tries to reconcile the descriptive semantics of RDF [12] with the prescriptive semantics of an OO programming language, such as COOL.
- DR-DEVICE [4], which is a system for defeasible reasoning on the Web that has been implemented on top of R-DEVICE. The operational semantics of defeasible logic are implemented through compilation into the generic rule language of R-DEVICE.

- O-DEVICE [13], a deductive object-oriented knowledge base system for reasoning over OWL documents. O-DEVICE is an extension of R-DEVICE that imports OWL documents by extending the RDF-to-object mapping scheme.

Rules for all the above systems can be expressed both in a native CLIPS-like language and in an extension of the OO-RuleML syntax [9], aided by an easy-to-use RDF-aware visual editor and development environment [5].

The main aim of building all those reasoning systems for the Semantic Web is to be able to use them for developing intelligent agent ([1], [14]) and web service applications ([3]).

7. REFERENCES

[1] Antoniou, G., Skylogiannis, T., Bikakis, A., Bassiliades, N., (2005) A Semantic Brokering System for the Tourism Domain. *Journal of Information Technology & Tourism* 7(3-4).

[2] Antoniou, G., van Harmelen, F. (2004) *A Semantic Web Primer*. MIT Press.

[3] Bassiliades, N., Anagnostopoulos, D., Vlahavas, I. (2005) Web Service Composition Using a Deductive XML Rule Language. *Distributed and Parallel Databases* 17(2), 135-178.

[4] Bassiliades, N., Antoniou, G., Vlahavas, I. (2005) A Defeasible Logic Reasoner for the Semantic Web. *International Journal on Semantic Web and Information Systems accepted for publication*.

[5] Bassiliades, N., Kontopoulos, E., Antoniou, G. (2005) A Visual Environment for Developing Defeasible Rule Bases for the Semantic Web. *Int. Conf. on Rules and Rule Markup Languages for the Semantic Web (RuleML-2005)*, Galway, Ireland, 10-11 November.

[6] Bassiliades, N., Vlahavas, I. (2004) R-DEVICE: A Deductive RDF Rule Language. *Proc. 3rd Int. Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)* LNCS 3323, 65-80. Springer-Verlag.

[7] Berners-Lee, T., Hendler, J., Lassila, O. (2001) The Semantic Web. *Scientific American* 284, 34-43.

[8] Berners-Lee, T., Fischetti, M., *Weaving the Web*. San Francisco: Harper, 1999.

[9] Boley, H., Tabet, S., Wagner, G. (2001) Design Rationale of RuleML: A Markup Language for Semantic Web Rules. *Proc. Int. Semantic Web Working Symp.*, 381-402.

[10] CLIPS Basic Programming Guide (v. 6.21), http://www.ghg.net/clips/CLIPS.html

[11] Dean, M., Schreiber, G. (eds.) OWL Web Ontology Language Reference. *W3C Recommendation*, http://www.w3.org/TR/owl-ref/

[12] Hayes, P. (ed.) (2004) RDF Semantics. *W3C Recommendation*, http://www.w3.org/TR/rdf-mt/

[13] Meditskos, G., Bassiliades, N. (2005) Towards an Object-Oriented Reasoning System for OWL. *Int. Workshop on OWL: Experiences and Directions*, Galway, Ireland, 11-12 November.

[14] Skylogiannis, T., Antoniou, G., Bassiliades, N., Governatori, G. (2005) DR-NEGOTIATE – A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies. *IEEE International Conference on E-Technology, E-Commerce and E-Service*. IEEE, 44-49.

[15] Wooldridge, M. (2002) *An Introduction to Multiagent Systems*. John Wiley & Sons.