Chapter 12
The Semantic Web

Judi McCuaig

12.1 ICDE and the Semantic Web

Exchanging and sharing data is a cornerstone challenge for application integration. The ICDE platform provides a notification facility to allow third party tools to exchange data. Suppose that an ICDE user is working with a third party tool to analyze financial transaction records from several organizations. The tool generates a list of finance-related keywords to describe the set of transaction records after some complex analysis and stores this list in the ICDE data store. Suppose further that this ICDE user has set up other third party tools to utilize their ICDE data as input to those tools processes. One of these tools uses the stored keyword list to perform a search for new, previously unseen information related to the ongoing financial transaction analysis.

This scenario is possible only when the cooperating tools have the capacity to share data. The sharing must include a consensus understanding of the semantics of the data items being shared. Most often, this consensus is achieved by creating a data structure that is coupled to every application using the shared data. The data structure defines the format (e.g., list, table) and the semantics (e.g., document name, document title, document location, document topic, etc.) of the shared data.

Within the ICDE framework, that shared understanding could be reached by publishing a table structure and requiring all collaborating applications to use that structure to share data. However, the ICDE development team could never anticipate suitable data structures for every third party tool and every application domain in which ICDE would operate. New tables could be added of course, but each third party tool vendor would have to negotiate with the ICDE team to get a suitable data structure defined, making agile tool integration impossible.

A more flexible approach would allow third party tools to publish data via the ICDE data store using any suitable structure. Subsequently, any other authorized tool should be able to dynamically discover the structure of the published data and understand the semantics of the content. No prior, hard-coded knowledge of data structures should be needed.

The obvious requirement for such a flexible solution is to use self-describing data structures for published data. Extensible Markup Language (XML) documents

I. Gorton, *Essential Software Architecture*,
DOI 10.1007/978-3-642-19176-3_12,

would suffice, as any program can dynamically parse an XML document and navigate the data structure. However, raw XML doesn't support semantic discovery making understanding ad hoc data problematic. For example, one third party tool might use the XML tag <location> to indicate the location of some information, whereas another may use <URI>, and another <pathname>. The semantics of these tag names tell a human reader that each tag contains the same information, but there is no way to make that conclusion programmatically using only XML. Forcing all tools to use the same strict tag vocabulary is not any more flexible than forcing them all to use the same data structure.

What's required is a mechanism to share the semantics of the chosen vocabulary, allowing programmatic discovery of terms that describe similar concepts. Using such a mechanism, a tool can determine that <URI> and <location> are actually the same concept, even when the relationship is not explicitly defined in the software or the published data.

The solution to this problem lies in the set of technologies associated with the Semantic Web. The Semantic Web makes it possible to describe data in ways that make its semantics explicit and hence discoverable automatically in software. One of the key innovations lies in the use of ontologies, which describe the relevant concepts in a domain, and the collection of relationships between those concepts.

This chapter introduces the basic technologies of the Semantic Web. It then shows how domain ontologies could be used in the ICDE platform to support ease of integration for third party tool vendors.

12.2 Automated, Distributed Integration and Collaboration

The difficulties associated with software integration have plagued software engineers since the early days of the computing industry. Initial efforts at integration (ignoring the problems of hardware and storage interoperability) centered on making data accessible to multiple applications, typically through some sort of database management system.

More recently efforts have been made to create interoperable processes using components using technologies like CORBA or JEE. As explained in previous chapters, services-oriented architectures and Web services are the latest technologies to give software designers the opportunity to create software systems by gluing together services, potentially from a variety of providers, to create a specialized software system designed for a particular business problem.

There are difficulties associated with locating, integrating, and maintaining a system composed of autonomous services and components. The major challenges include creation, management, and utilization of appropriate metadata to facilitate dynamic interaction with the available information, services, and components. It is precisely these problems that the technologies making up the Semantic Web tackle. They provide tools and approaches to metadata management that are generically useful for dynamically integrating software applications.

12.3 The Semantic Web

The purpose of the Semantic Web initiative is to create machine understandable information where the semantics are explicit and usable by algorithms and computer programs. This original goal has expanded to include the goal of creating services, or processes, that are machine understandable and useable by other processes. This shared understanding, whether it be of data or services, is made possible by a rich collection of metadata description languages and protocols. For the most part, the Semantic Web exists because of these languages.

The interoperability promised by Semantic Web technologies is made possible through:

- The formalization of metadata representation
- Continued development in knowledge representation
- Logic and reasoning techniques that can exploit both the metadata and the represented knowledge

The key capabilities offered are flexible representation of metadata and relationships, encoded as ontologies. These allow translation between metadata vocabularies and reasoning about the represented metadata entities.

Figure 12.1 illustrates the relationships between some of the technologies associated with the Semantic Web. XML, Unicode, and Uniform Resource Identifiers (URI) form the backbone and allow the storage and retrieval of information. The Resource Description Framework (RDF) is the basis for describing the structure of information within Semantic Web applications. Ontologies, frequently encoded using the Web Ontology Language (OWL) and Taxonomies described using the Resource Description Framework Schema (RDFS), provide the layer at

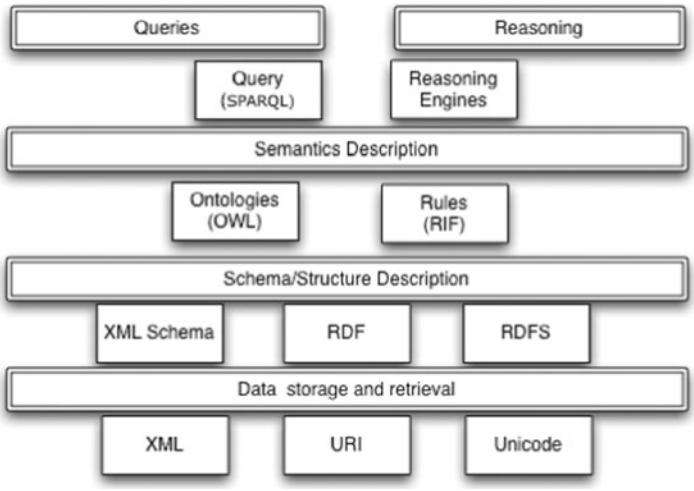

Fig. 12.1 Semantic Web technologies

which the semantics of the information can be described and made available to applications. An additional layer of computation provides facilities for queries and reasoning about the available information. Semantic Web applications are typically written on top of this query and reasoning layer.

12.4 Creating and Using Metadata for the Semantic Web

The advanced capabilities associated with the Semantic Web come almost entirely on the back of extensive efforts in creating and maintaining metadata. The introduction of the XML and the technologies related to it provided a structured, flexible mechanism for describing data that is easily understood by machines (and a subset of humans who like angled brackets). XML provides the means to label entities and their parts, but it provides only weak capabilities for describing the relationships between two entities. For example, consider the XML fragment in Fig. 12.2. It describes a *Person* in terms of *Name*, *Email_Address,* and *Phone_Number*, and a *Transaction* in terms of *Type*, *Client*, and *AccountNumber*. The example also shows the use of attributes to create unique identifiers (*id*) for each entity.

XML is however not adequate for easy identification of relationships between pieces of information. For example, using only the XML tag metadata in the figure, the identification of the email address of the person who conducted a specific transaction is somewhat complex. It relies on the ability to determine that the *Client* field of the transaction represents the name of a person and that if the *Client* field data matches the *Name* field of a person a relationship can be identified and the person's email address used.

A human can quickly make that determination because a human understands that the tags *Client* and *Name* both signify information about people. A software process unfortunately has no such capability because it does not have any way of representing those semantics.

```
<example>
    <Person id="123">
          <Name>J Doe</Name>
          <Email_Address>doe@myplace</Email>
          <Phone_Number>123 456 7899</Phone_Number>
    </Person>
    <Transaction transID="567">
          <Downtick>500</Downtick>
          <Client>Josef Doe</Client>
          <AccountNumber>333222111</AccountNumber>
    </Transaction>
 </example>
```

Fig. 12.2 XML example

To address this problem, the RDF was developed as a machine understandable representation of relationships between entities. It is assumed that each entity and relationship can be identified with a URI. These URIs are used to form an RDF statement of the form {Subject, Predicate, Object}, commonly called a "triple."

To continue the above example discussion, the addition of an RDF relationship *conducted_by* (see RDF example below) between the transaction and the person (using the *id* attributes as the unique identifier) allows a machine to extract the email address of the transaction owner, without requiring replication of information. The RDF statement below indicates that the person referenced by id # 123 conducted the transaction referenced by id # 567.

```
<http://example.net/transaction/id567><http://example.net/conduc
ted_by><http://different.example.net/person/id123>
```

The relationship is explicit and easily exploited using computer programs once a human identifies and records existence of the relationship. RDF doesn't solve the whole problem however, because there is still no mechanism to automatically identify the relationships or to detail any restrictions on the participants in those relationships. For instance, a human quickly understands that a transaction may be conducted by a person, but that a person cannot be conducted by a transaction! The RDF in the example has no such restrictions, so the algorithms processing the RDF have no way of verifying the types or expected attributes of the entities in the relationships.

A partial solution to the relationship identification problem is found in the schema languages for XML and RDF. The schema languages allow a priori definition of entities and relationships that includes domains and ranges for attributes and entities. Entities (or relationships) that reference the schema for their definition can then be checked for consistency with the schema. Programs can then enforce range and data type restrictions during data processing without human intervention.

Together RDF, XML, and their schema languages provide a robust, usable method for encoding metadata and exploiting it to automatically identify relationships between entities. However, our kitbag of essential technologies for automated metadata understanding also needs the ability to make deductions and inferences about metadata.

Consider again the transaction and client example. The completion of a transaction is usually the result of collaboration between several individuals, including a client, financial consultant, and clerk for instance. It would be trivial to modify the XML metadata example given earlier to represent both the consultant and clerk as part of the transaction's metadata, thus explicitly representing the relationship between the transaction and the collaborating individuals.

However, the collaboration between any particular pair of those three entities (consultant, client, clerk) is not explicitly represented in the metadata. A program that needs to identify both the client and the consultant for a transaction has no mechanism for determining whether specific clients and consultants are known to one another using our current set of metadata. One way to remedy this problem is to

simply add more metadata and explicitly identify the client–consultant relationship, but even for this small example it is apparent that metadata would rapidly exceed data in quantity. A more general solution is to define logical rules that delineate the possible deductions with the different types of metadata. Those logical rules define the semantics associated with the metadata and are frequently described in conjunction with the definition of a formal ontology. Ontologies are explained in the next section.

Well-defined and ordered metadata is the backbone of the Semantic Web. Metadata is used to dynamically assemble data from a variety of sources, for making informed decisions, and to provide data for planning such things as, for example, vacations and the shipping of goods. While metadata technologies are most frequently used with Web-based information at the moment, they can be used with equal power to identify connections between software services for the purposes of creating any software system.

12.5 Putting Semantics in the Web

The one feature that distinguishes the Semantic Web from the World Wide Web is the representation and utilization of meaning, or semantics. A common representation for semantics is an ontology. An ontology consists of a set of ideas or concepts and the collection of relationships between those concepts.

An ontology can be used to identify ideas that are related to one another and to provide the structure and rules for a reasoning engine to make inferences about those ideas. An ontology models both abstraction and aggregation relationships. More complex ontologies model domain-specific relationships about individuals and classes in the ontology as well. Ontologies can also provide information about concepts that are equivalent to other concepts. When suitably complex, an ontology can provide the mapping between different metadata vocabularies, making integration of software processes much simpler.

For example, consider the ontology fragments represented in Fig. 12.3. The ontology shows that *Humans* and *Persons* have *Occupations* and that certain kinds of *Occupations* have relationships with other concepts in the ontology. Both *Students* and *Instructors* are concerned with *Courses* and both *Authors* and *Publishers* are concerned with *Publications*. This ontology could be used by an automated system to identify related entities or identify the use of equivalent concepts (such as *Human* and *Person* in this example). The ontology provides logical axioms to a reasoning system, which can then make inferences about the information.

Within the Semantic Web, the OWL is a common representation of the axioms and domain concepts.

Consider once more the example of the financial transaction. An ontology could provide the logic to automatically identify a relationship between a client and a financial consultant, even when the relationship is not explicitly stated in the available metadata or in the schema. A reasoning system could deduce, given the

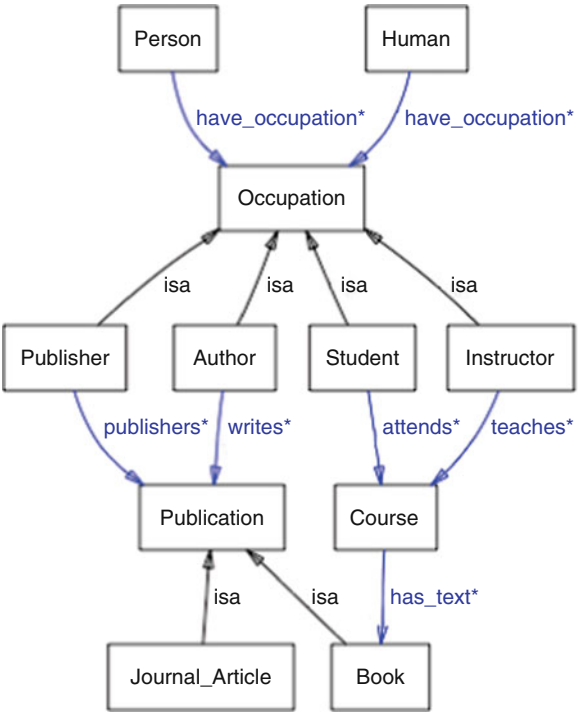

Fig. 12.3 Ontology example

correct rules or training, that a client and consultant are known to one another if they have collaborated on some specified number of transactions. An additional rule could state that if they have collaborated on more than one type of transaction, they are well known to each other.

Together, the financial transaction data, the metadata, and the ontology make up a knowledge base that not only provides information about financial transactions and clients, but can also be used to identify relationships between specific humans. Information about client–consultant relationships could be useful to someone analyzing financial transactions for the purpose of identifying sets or groups of people conducting specific classes of transactions (i.e., transactions occurring in a particular time period), or perhaps for organizations needing to determine the outreach of particular financial consultants.

An ontology can also contain rules that constrain relationships. Suppose that the example ontology contained a rule that precludes the same individual from being both client and clerk for a transaction. The ontology could then be used, in conjunction with a reasoning engine, to detect errors in information or to prevent errors in data entry. Ontologies provide meaning for the metadata that is the backbone of the Semantic Web.

XML, RDF, and OWL are the basic technologies supporting the Semantic Web, which is now beginning to show up in the mainstream web and in industrial

applications. The Semantic Hacker[1] is an example of a stand-alone demonstration of the possibilities of the Semantic Web for information discovery. Ontoprise[2] uses Semantic Web technologies to develop troubleshooting and design validation systems that function much like expert systems with more flexibility in the definition and maintenance of data and rules. Their clients include auto manufacturers, makers of industrial robots, and investment firms.

One of the difficulties for early adoption of Semantic Web technologies was the difficulty in authoring and developing materials. The mastery of XML, RDF, and OWL requires a high level of technical expertise and a significant time commitment. This barrier to use has slowed the adoption of the technologies and also masked much of the progress on Semantic Web development behind prototype sites and proof-of-concept applications. Fortunately, in the last few years that has changed.

In the past year or so, the focus has shifted from individual organizations that provide specific semantically enabled websites to vendors who wrap the technologies associated with the Semantic Web into turnkey systems for publishing particular types of information. For example, Allegrograph[3] provides a database system and query language for managing RDF data, queries using SPARQL and reasoning services on the data, which frees potential developers from the need to build a deep understanding of those technologies. Thetus[4] provides a system to do enterprise-wide knowledge modeling using Semantic Web technology. With the increase in providers of publishing and authoring tools, the incidence of Semantic Web-enabled applications and websites will continue to increase.

12.6 Semantics for ICDE

The ICDE system would benefit from using ontologies to support information exchange and integration tasks for third party tools. As hinted in the chapter introduction, one task within financial transaction analysis that would benefit greatly from a solid description of semantics is the identification of consistent vocabularies. Shown in Fig. 12.4 is a portion of a financial ontology originally created by Teknowledge[5] as part of the SUMO ontology. The ontology fragment shows several different kinds of financial transactions arranged in an abstract hierarchy.

Suppose that this ontology is available to the ICDE system, and an ICDE user was analyzing the example presented in Fig. 12.2. In that data, *Downtick* is the

[1] http://www.semantichacker.com/

[2] http://www.ontoprise.de/de/en/home/products/semanticguide.html

[3] http://www.agraph.franz.com/allegrograph/

[4] http://www.thetus.com/

[5] http://www.teknowledge.com/

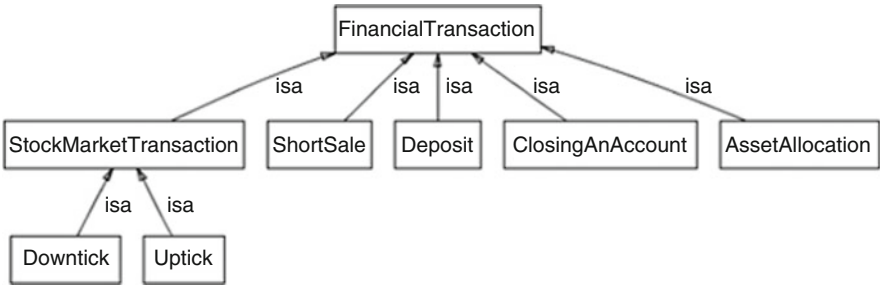

Fig. 12.4 A simple financial transaction ontology

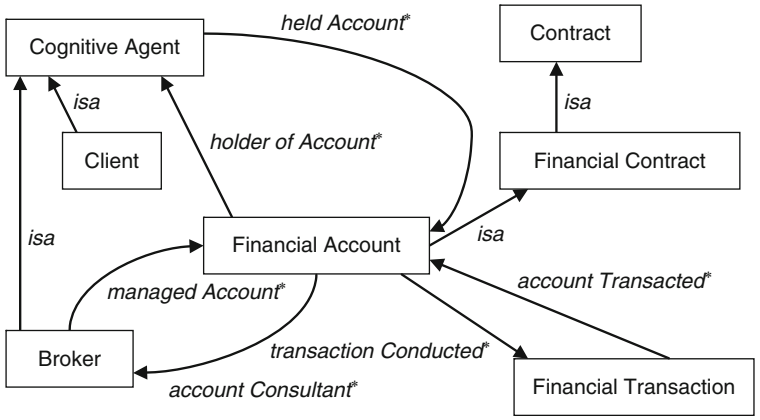

Fig. 12.5 Rules in an ontology

XML tag for the transaction ID, a choice which might prevent other third party ICDE tools from making use of the data because the XML tag is not standard. However, using the ontology and a reasoning engine, it is straightforward to determine that *Downtick* is a type of *Financial Transaction* and that the information should be shared with any tools that are interested in data about financial transactions.

Ontologies could provide much more than just thesaurus services for ICDE tools. An OWL ontology can encode complex rules about the relationships between individuals of particular conceptual type, which would allow reasoning engines to make deductions about individual data elements.

Consider the ontology fragment shown in Fig. 12.5. It shows that the ontology contains rules describing the relationships between accounts, account holders, transactions, and brokers. A reasoning engine can use these descriptions to deduce relationships between a particular client and a broker or to deduce that a particular broker had a likely involvement with an individual transaction, even when the data being analyzed contained no specific linkage between the two entities.

This kind of shared ontology could enable collaborating third party ICDE tools to help the user notice previously unseen connections within data. For instance, suppose that one tool helped a user select and analyze particular types of financial transactions. Another tool assisted the user to identify social networks of individuals based on shared interest in accounts. Individually, neither of these two tools would uncover relationships between an advisor and a particular type of transaction, but the individual results from the two tools could be combined (possibly by a third tool) to uncover the implicit relationships.

12.7 Semantic Web Services

Web services and service-oriented architectures were presented in the previous chapter as a significant step toward a simple solution for the interoperability problems that typically plague enterprise applications. Web services also play a part in the Semantic Web. As Semantic Web applications increase in complexity, and as information consumers become more discerning, the focus is turning from semantically addressable information to semantically addressable services that allow automated creation of customized software system, or Semantic Web services.

Current tools provide the capability to describe Web services but do not have adequate means for categorizing and utilizing those descriptions. The categorizations available, such as WSIndex[6] and Ping the Semantic Web,[7] are designed primarily for human use rather than machine. Automated system composition is the subject of proof-of-concept prototypes at the moment, but few operating systems.

However, web services are typically constructed with generous metadata descriptions, which is the key component of the Semantic Web. As with all metadata, the difficulty in using it for dynamic composition lies in understanding the semantics. Predictably, a substantial research community is focused on applying ontologies and Semantic Web technologies to define a domain called Semantic Web services. Semantic Web services provide a mechanism for creating, locating, and utilizing semantically rich descriptions of services. One of the foremost tasks for this community is to standardize the description of the semantics associated with Web service descriptions. Once the semantics are clear, Web service descriptions can be used to create specifications for composite services, to represent business logic at a more abstract level, and to supply knowledge for reasoning systems which can then intelligently assemble software from service descriptions.

One of the underlying languages for the Semantic Annotation of Web services is SAWSDL (Semantic Annotations for Web Services Description Language).[8]

[6] http://www.wsindex.org

[7] http://www.pingthesemanticweb.com/

[8] http://www.w3.org/2002/ws/sawsdl/

SAWSDL does not specify the ontology but provides the language for identifying the ontological concepts associated with a Web service within the service description. SAWSDL outlines the definition of annotations for a small subset of the possible components of a WSDL description. SAWSDL is ontology agnostic, in that the specification makes no reference to a preferred language or encoding for ontologies.

Languages for describing ontologies about Web services include OWL-S, a service-specific variant of OWL, and the Web Services Modeling Ontology (WSMO). These languages permit the integration of semantic annotation with the Web Services Description Language (WSDL). Integration with WSDL is important since most existing Web services use WSDL as the basis for service description. The creation of Semantic Web Services for the general public however seems quite a long way off at the moment. Good prototypes exist and the Semantic Web community is slowly coming to an agreement about the languages and definitions required to realize Semantic Web Services.

The technologies bear watching though, since successes in constructing and using Semantic Web Services will change the way software is created. The current state of Semantic Web Services shows promise for enterprise integration, but they currently lack the capacity for automated discovery and composition of services. Nonetheless, it seems inevitable that Semantic Web Services will soon define an automated mechanism for finding and composing services and change the way we think about software systems.

12.8 Continued Optimism

The Semantic Web has enjoyed immense publicity in the past few years. Many research project descriptions have been quickly adjusted to reflect even the smallest connection to the Semantic Web in an effort to take advantage of that popularity. Of course, this results in an increase in the scope of research claiming to be Semantic Web research, reducing the concentration of work addressing the important goals of semantically rich, machine understandable metadata for data and processes.

While many believe in the technologies, general wariness seems to prevail. On the surface, the Semantic Web looks like a refactoring of the artificial intelligence projects that went out of vogue several years ago. However, the need for semantic representations in software and information systems is now widely recognized, and the demand for real solutions is growing. This time, the research goals are more aligned with the needs of the public, and the technology might gain acceptance.

The Semantic Web has all of the data management issues associated with any large information system. Who will take the time to provide all the detailed metadata about existing services and information? Who monitors information and services for integrity, authenticity, and accuracy? How are privacy laws and concerns addressed when computing is composed from distributed services? Web services providers will spring up as a new category of business, but how will they

be monitored and regulated? As systems are built that rely on quality metadata, its maintenance and upkeep will become vital operational issues.

Despite the prototypical nature of most of the operational systems so far, the Semantic Web places new techniques, new applications, and important experiences in the toolbox of software architects. The Semantic Web is simply a conglomeration of cooperating tools and technologies, but precisely because of the loose coupling between technologies, the Semantic Web provides a flexible sandbox for developing new frameworks and architectures.

And, if one looks past the hype, the goals of the Semantic Web community are the same as the goals for distributed software architecture: to create loosely coupled, reliable, efficient software that addresses the needs of users. Through the formally defined mechanisms for reasoning with metadata, the Semantic Web provides the basis for creating software that is truly responsive to the needs of users, their tasks, and their physical context.

Software developers and researchers are responding quickly to the needs of semantic computing. The 2008 Semantic Web Conference hosted a research track, a Semantic Web "in use" track, and workshops and tutorials on everything from security to reasoning systems. The topic is active both in industry and in academics. The Semantic Web services architecture identifies message mediation, security, process composition, negotiation and contracting, and message formulation as important aspects of the Semantic Web, and each of these is being explored and prototyped. Developments such as SAWSDL and Service Ontologies (OWL-S and WSMO) show promise as process description and composition languages. The Semantic Web and software architecture are on paths that are rapidly converging on a new, semantically driven, way of building software.

12.9 Further Reading

Three general books on the Semantic Web are:

Liyang Yu Introduction to the Semantic Web and Semantic Web Services Chapman & Hall/CRC. 2007.

Pascal Hitzler, Sebastian Rudolph, Markus Kroetzsch, Foundations of Semantic Web Technologies, Chapman & Hall/CRC. 2009.

Michael C. Daconta, Leo J. Obrst, Kevin T. Smith, The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management, Wiley 2010.

Nigel Shadbolt, Wendy Hall, and Tim Berners-Lee's 2006 revisitation of the original Scientific American article *The Semantic Web* sheds light on the vision of people in the front lines and what they believe is required to realize the promise of the Semantic Web.

Shadbolt, N., Berners-Lee, T., and Hall, W. 2006. The Semantic Web Revisited. IEEE Intelligent Systems 21, 3 (May. 2006), 96–101.

David Provost has recently reviewed a number of organizations in the Semantic Web industry and published his report under Creative Commons License. It is titled *On The Cusp, A Global Review of the Semantic Web Industry* and is available from: http://www.davidprovost.com/

The W3C's Web site is a source of great information on the Semantic Web:

http://www.w3.org/2001/sw/

Specific details about some of the technologies can be found at the following online locations:

OWL	http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
SAWSDL	http://www.w3.org/2002/ws/sawsdl/
RDF	http://www.w3.org/RDF/
WSMO	http://www.cms-wg.sti2.org/home/
OWL-S	http://www.daml.org/services/owl-s/

A tool for building ontologies can be freely downloaded from http://www.protege.stanford.edu/. It's a good tool for exploring how ontologies can be built and used: