What is This Book About?

The current World Wide Web is syntactic and the content itself is only readable by humans. The Semantic Web proposes the mark-up of content on the Web using formal ontologies that structure underlying data for the purpose of comprehensive machine understanding. Currently most Web resources can only be found and queried by syntactical search engines. One of the goals of the Semantic Web is to enable reasoning about data entities on different Web pages or Web resources. The Semantic Web is an extension of the current Web in which information is given well-defined meaning, enabling computers and people to work in co-operation.

Along with the Semantic Web, systems and infrastructures are currently being developed to support Web services. The main idea is to encapsulate an organization’s functionality within an appropriate interface and advertise it as Web services. While in some cases Web services may be utilized in an isolated form, it is normal to expect Web services to be integrated as part of Web processes. There is a growing consensus that Web services alone will not be sufficient to develop valuable Web processes due the degree of heterogeneity, autonomy, and distribution of the Web. Several researchers agree that it is essential for Web services to be machine understandable in order to support all the phases of the lifecycle of Web processes.

It is therefore indispensable to interrelate and associate two of the hottest R&D and technology areas currently associated with the Web—Web services and the Semantic Web. The study of the application of semantics to each of the steps in the Semantic Web process lifecycle can help address critical issues in reuse, integration and scalability.

Why Did I Put a Lot of Effort in Creating This Book?

I started using Semantic Web technologies in 2001 right after Tim Berners-Lee, James Hendler, and Ora Lassila published their article entitled “The Semantic Web” in the May issue of Scientific American. This seminal article described some of the future potential of what was called the Semantic Web, the impact of computers understanding and interpreting semantic information, and how searches could be dramatically improved when using semantic metadata. In 2004, I started planning to teach a course on Semantic Web at the University of Madeira (Portugal). When looking for material and textbooks on the topic for my students, I realized that there was only a hand full of good books discussing the concepts associated with the Semantic Web. But none aggregated in one place the theory, the tools, and the applications of the Semantic Web. So, I decided to write this comprehensive and handy book for students, teachers, and researchers.

The major goal of this book is to bring contributions from researchers, scientists from both industry and academics, and representatives from different communities together to study, understand, and explore the theory, tools and applications of the Semantic Web. It brings together computing that deal with the
design and integration, bio-informatics, education, and so forth. Ontological engineering is defined as the set of activities that concern the ontology development process, the ontology life cycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them. In Chapter III we provide an overview of all these activities, describing the current trends, issues and problems. More specifically, we cover the following aspects of ontological engineering: (a) Methods and methodologies for ontology development. We cover both comprehensive methodologies that give support to a large number of tasks of the ontology development process and methods and techniques that focus on specific activities of this process, focusing on: ontology learning, ontology alignment and merge, ontology evolution and versioning, and ontology evaluation; (b) Tools for ontology development. We describe the most relevant ontology development tools, which give support to most of the ontology development tasks (especially formalization and implementation) and tools that have been created for specific tasks, such as the ones identified before: learning, alignment and merge, evolution and versioning and evaluation, and (c) finally, we describe the languages that can be used in the context of the Semantic Web. This includes W3C recommendations, such as RDF, RDF schema and OWL, and emerging languages, such as WSML.

Chapter IV gives an overview of editing tools for building ontologies. The construction of an ontology demands the use of specialized software tools. Therefore, we give a synopsis of the tools that we consider more relevant. The tools we have selected were Protégé, OntoEdit, DOE, IsaViz, Ontolingua, Altova Semantic Works, OilEd, WebODE, pOWL and SWOOP. We started by describing each tool and identifying which tools supported a methodology or other important features for ontology construction. It is possible to identify some general distinctive features for each software tool. Protégé is used for domain modeling and for building knowledge-base systems and promotes interoperability. DOE allows users to build ontologies according to the methodology proposed by Bruno Bachimont. Ontolingua was built to ease the development of ontologies with a form-based Web interface. Altova SemanticWorks is a commercial visual editor that has an intuitive visual interface and drag-and-drop functionalities. OilEd’s interface was strongly influenced by Stanford’s Protégé toolkit. This editor does not provide a full ontology development environment. However, it allows users to build ontologies and to check ontologies for consistency by using the FaCT reasoner. WebODE is a Web application. This editor supports ontology edition, navigation, documentation, merge, reasoning and other activities involved in the ontology development process. pOWL is capable of supporting parsing, storing, querying, manipulation, versioning and serialization of RDFS and OWL knowledge bases in a collaborative Web enabled environment. SWOOP is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment.

The aim of Chapter V is to give a general introduction to some of the ontology languages that play a prominent role on the Semantic Web. In particular, it will explain the role of ontologies on the Web, review the current standards of RDFS and OWL, and discuss open issues for further developments. In the context of the Web, ontologies can be used to formulate a shared understanding of a domain in order to deal with differences in terminology of users, communities, disciplines and languages as it appears in texts. One of the goals of the Semantic Web initiative is to advance the state of the current Web through the use of semantics. More specifically, it proposes to use semantic annotations to describe the meaning of certain parts of Web information and, increasingly, the meaning of message elements employed by Web services. For example, the Web site of a hotel could be suitably annotated to distinguish between the hotel name, location, category, number of rooms, available services and so forth. Such meta-data could facilitate the automated processing of the information on the Web site, thus making it accessible to machines and not primarily to human users, as it is the case today. The current and most prominent Web standard for semantic annotations is RDF and RDF schema, and its extension OWL.
Semantic Web, ontologies, knowledge management and engineering, Web services, and Web processes. It serves as the platform for exchange of both practical technologies and far reaching research.

Organization of the Book

This book is divided into 13 chapters and it is organized in a manner that allows a gradual progression of the main subject toward more advanced topics. The first five chapters cover the logic and engineering approaches needed to develop ontologies and bring into play semantics. Chapters VII and VIII introduce two technological areas, Web services and Web processes, which have received a considerable amount of attention and focus from the Semantic Web community. The remaining chapters, Chapters IX, X, XI, XII, and XIII, describe in detail how semantics are being used to annotate Web services, discover Web services, and deploy semantic search engines.

Chapter I introduces the concepts of syntactic and Semantic Web. The World Wide Web composed of HTML documents can be characterized as a syntactic or visual Web since documents are meant only to be displayed by Web browsers. In the visual Web, machines cannot understand the meaning of the information present in HTML pages, since they are mainly made up of ASCII codes and images. The visual Web prevents computers from automating information processing, integration, and interoperability. Currently the Web is undergoing an evolution and different approaches are being sought for adding semantics to Web pages and resources in general. Due to the widespread importance of integration and interoperability for intra- and inter-business processes, the research community has already developed several semantic standards such as the resource description framework (RDF), RDF schema (RDFS) and the Web Ontology Language (OWL). RDF, RDFS and OWL standards enable the Web to be a global infrastructure for sharing both documents and data, which make searching and reusing information easier and more reliable as well. RDF is a standard for creating descriptions of information, especially information available on the World Wide Web. What XML is for syntax, RDF is for semantics. The latter provides a clear set of rules for providing simple descriptive information. OWL provides a language for defining structured Web-based ontologies which allows a richer integration and interoperability of data among communities and domains. Even though the Semantic Web is still in its infancy, there are already applications and tools that use this conceptual approach to build Semantic Web-based systems. Therefore, in this chapter, we present the state of the art of the applications that use semantics and ontologies. We describe various applications ranging from the use of Semantic Web services, semantic integration of tourism information sources, and semantic digital libraries to the development of bioinformatics ontologies.

Chapter II introduces a number of formal logical languages which form the backbone of the Semantic Web. They are used for the representation of both ontologies and rules. The basis for all languages presented in this chapter is the classical first-order logic. Description logics is a family of languages which represent subsets of first-order logic. Expressive description logic languages form the basis for popular ontology languages on the Semantic Web. Logic programming is based on a subset of first-order logic, namely Horn logic, but uses a slightly different semantics and can be extended with non-monotonic negation. Many Semantic Web reasoners are based on logic programming principles and rule languages for the Semantic Web based on logic programming are an ongoing discussion. Frame logic allows object-oriented style (frame-based) modeling in a logical language. RuleML is an XML-based syntax consisting of different sub-languages for the exchange of specifications in different logical languages over the Web.

In computer science, ontologies are defined as formal, explicit specifications of shared conceptualizations. Their origin in this discipline can be referred back to 1991, in the context of the DARPA knowledge sharing effort. Since then, considerable progress has been made and ontologies are now considered as a commodity that can be used for the development of a large number of applications in different fields, such as knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database
In Chapter VI we describe and explain how reasoning can be carried out in on the Semantic Web. Reasoning is the process needed for using logic. Efficiently performing this process is a prerequisite for using logic to present information in a declarative way and to construct models of reality. In this chapter we describe both what the reasoning over the formal semantics of description logic amounts to and, to illustrate how formal reasoning can (and cannot!) be used for understanding real world semantics given a good formal model of the situation. We first describe how the formal semantics of description logic can be understood in terms of completing oriented labeled graphs. In other words we interpret the formal semantics of description logic as rules for inferring implied arrows in a dots and arrows diagram. We give an essentially complete “graphical” overview of OWL that may be used as an introduction to the semantics of this language. We then touch on the algorithmic complexity of this graph completion problem giving a simple version of the tableau algorithm, and give pointers to existing implementations of OWL reasoners. The second part deals with semantics as the relation between a formal model and reality. We give an extended example building up a small toy ontology of concepts useful for describing buildings, their physical layout and physical objects such as wireless routers and printers in the turtle notation for OWL. We then describe a (imaginary) building with routers in these terms. We explain how such a model can help in determining the location of resources given an idealized wireless device that is in or out of range of a router. We emphasize how different assumptions on the way routers and buildings work are formalized and made explicit in the formal semantics of the logical model. In particular we explain the sharp distinction between knowing some facts and knowing all facts (open, versus closed world assumption). The example also illustrates the fact that reasoning is no magical substitute for insufficient data. This section should be helpful when using ontologies and incomplete real world knowledge in applications.

Chapter VII gives an introduction to Web service technology. Web services are emerging technologies that allow programmatic access to resources on the Internet. Web services provide a means to create distributed systems which are loosely couple, meaning that the interaction between the client and service is not dependent on one having any knowledge of the other. This type of interaction between components is defined formally by the service-oriented architecture (SOA). The backbone of Web services is XML. Extensible Markup Language (XML) is a platform independent data representation which allows the flexibility that Web services need to fulfill their promise. Simple object access protocol, or SOAP, is the XML-based protocol that governs the communication between a service and the client. It provides a platform and programming language independent way for Web services to exchange messages. Web Service Description Language (WSDL) is an XML-based language for describing a service. It describes all the information needed to advertise and invoke a Web service. UDDI is a standard for storing WSDL files as a registry so that they can be discovered by clients. There are other standards for describing policy, security, reliability, and transactions of Web services that are described in the chapter. With all this power and flexibility, Web services are fairly easy to build. Standard software engineering practices are still valid with this new technology though tool support is making some of the steps trivial. Initially, we design the service as a UML class diagram. This diagram can then be translated (either by hand or by tools like Posiden) to a Java interface. This class can become a Web service by adding some annotations to the Java code that will be used to create the WSDL file for the service. At this point, we need only to implement the business logic of the service to have a system that is capable of performing the needed tasks. Next, the service is deployed on an application server, tested for access and logic correctness, and published to a registry so that it can be discovered by clients.

In Chapter VIII we introduce and provide an overview of the Business Process Execution Language for Web services (known as BPEL4WS or BPEL for short), an emerging standard for specifying the behavior of Web services at different levels of details using business process modeling constructs. BPEL
represents a convergence between Web services and business process technology. It defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. Being supported by vendors such as IBM and Microsoft, BPEL is positioned as the “process language of the Internet.” The chapter firstly introduces BPEL by illustrating its key concepts and the usage of its constructs to define service-oriented processes and to model business protocols between interacting Web services. A BPEL process is composed of activities that can be combined through structured operators and related through control links. In addition to the main process flow, BPEL provides event handling, fault handling and compensation capabilities. In the long-running business processes, BPEL applies correlation mechanism to route messages to the correct process instance. On the other hand, BPEL is layered on top of several XML specifications such as WSDL, XML schema and XPath. WSDL message types and XML schema type definitions provide the data model used in BPEL processes, and XPath provides support for data manipulation. All external resources and partners are represented as WSDL services. Next, to further illustrate the BPEL constructs introduced above, a comprehensive working example of a BPEL process is given, which covers the process definition, XML schema definition, WSDL document definition, and the process execution over a popular BPEL-compliant engine. Since the BPEL specification defines only the kernel of BPEL, extensions are allowed to be made in separate documentations. The chapter reviews some perceived limitations of BPEL and extensions that have been proposed by industry vendors to address these limitations. Finally, for an advanced discussion, the chapter considers the possibility of applying formal methods and Semantic Web technology to support the rigorous development of service-oriented processes using BPEL.

Web services show promise to address the needs of application integration by providing a standards-based framework for exchanging information dynamically between applications. Industry efforts to standardize Web service description, discovery and invocation have led to standards such as WSDL, UDDI, and SOAP respectively. These industry standards, in their current form, are designed to represent information about the interfaces of services, how they are deployed, and how to invoke them, but are limited in their ability to express the capabilities and requirements of services. This lack of semantic representation capabilities leaves the promise of automatic integration of applications written to Web services standards unfulfilled. To address this, the Semantic Web community has introduced Semantic Web services. Semantic Web services are the main topic of Chapter IX. By encoding the requirements and capabilities of Web services in an unambiguous and machine-interpretable form semantics make the automatic discovery, composition and integration of software components possible. This chapter introduces Semantic Web services as a means to achieve this vision. It presents an overview of Semantic Web services, their representation mechanisms, related work and use cases. Specifically, the chapter contrasts various Semantic Web service representation mechanisms such as OWL-S, WSMO and WSDL-S and presents an overview of the research work in the area of Web service discovery, and composition that use these representation mechanisms.

Web services are software components that are accessible as Web resources in order to be reused by other Web services or software. Hence, they function as middleware connecting different parties such as companies or organizations distributed over the Web. In Chapter X, we consider the process of provisioning data about a Web service to constitute a specification of the Web service. At this point, the question arises how a machine may attribute machine-understandable meaning to this metadata. Therefore, we argue for the use of ontologies for giving a formal semantics to Web service annotations, that is, we argue in favor of Semantic Web service annotations. A Web service ontology defines general concepts such as service or operation as well as relations that exist between such concepts. The metadata describing a Web service can instantiate concepts of the ontology. This connection supports Web service developers to understand and compare the metadata of different services described by the same or a similar ontology. Consequently, ontology-based Web service annotation leverages the use, reuse and verification of Web services. The process of Semantic Web service annotation in general requires input from multiple sources, that is legacy descriptions, as well as a labor-intensive modeling
effort. Information about a Web service can be gathered for example from the source code of a service (if annotation is done by a service provider), from the API documentation and description, from the overall textual documentation of a Web service or from descriptions in WS* standards. Depending on the structuredness of these sources, semantic annotations may (have to) be provided manually (e.g., if full text is the input), semi-automatically (e.g. for some WS* descriptions) or fully automatically (e.g., if Java interfaces constitute the input). Hence, a semantic description of the signature of a Web service may be provided by automatic means, while the functionality of Web service operations or pre- and post-conditions of a Web service operation may only be modeled manually. Benefits of semantic specifications of Web services include a common framework that integrates semantic descriptions of many relevant Web service properties. It is the purpose of this chapter to explain the conceptual gap between legacy descriptions and semantic specifications and to indicate how this gap is to be bridged.

Chapter XI deals with methods, algorithms and tools for Semantic Web service discovery. Semantic Web has revolutionized, among other things, the implementation of Web services lifecycle. The core phases of this lifecycle, such as service discovery and composition can be performed more effectively through the exploitation of the semantics that annotate the service descriptions. This chapter focuses on the phase of discovery due to its central role in every, service-oriented architecture. Hence, it surveys existing approaches to Semantic Web service (SWS) discovery. Such discovery process is expected to substitute existing keyword-based solutions (e.g., UDDI) in the near future, in order to overcome their limitations. First, the architectural components of a SWS discovery ecosystem, along with potential deployment scenarios, are discussed. Subsequently, a wide range of algorithms and tools that have been proposed for the realization of SWS discovery are presented. The presentation of the various approaches aims at outlining the key characteristics of each proposed solution, without delving into technology-dependent details (e.g., service description languages). The descriptions of the tools included in this chapter provide a starting point for further experimentation by the reader. In this respect, a brief tutorial for a certain tool is provided as an appendix. Finally, key challenges and open issues, not addressed by current systems, are identified (e.g., evaluation of service retrieval, mediation and interoperability issues). The ultimate purpose of this chapter is to update the reader on the recent developments in this area of the distributed systems domain and provide the required background knowledge and stimuli for further research and experimentation in semantics-based service discovery.

Taking an abstract perspective, Web services can be considered as complex resources on the Web, that is, resources that might have more complex structure and properties than conventional data that is shared on the Web. Recently, the Web service modeling ontology (WSMO) has been developed to provide a conceptual framework for semantically describing Web services and their specific properties in detail. WSMO represents a promising and rather general framework for Semantic Web service description and is currently applied in various European projects in the area of Semantic Web services and Grid computing. In Chapter XII, we discuss how Web service discovery can be achieved within the WSMO Framework. First, we motivate Semantic Web services and the idea of applying semantics to Web services. We give a brief high-level overview of the Web service modeling ontology and present the main underlying principles. We discuss the distinction between two notions that are often intermixed when talking about Semantic Web services and thus provide a proper conceptual grounding for our framework, namely we strictly distinguish between services and Web services. Consequently, we distinguish between service discovery and web service discovery, whereas only the latter is then considered in detail in the chapter. Since in open environments like the Web, the assumption of homogeneous vocabularies and descriptions breaks, we briefly consider mediation and discuss its role in service and Web service Discovery. Hereby, we try to identify requirements on the discovery process and respective semantic descriptions which allow facing heterogeneity and scalability at the same time. We then present a layered model of successively
more detailed and precise perspectives on Web services and consider Web service descriptions on each of them. For the two most fine-grained levels, we then discuss how to detect semantic matches between requested and provided functionalities. Based on our model, we are able to integrate and extend matching notions that have been known in the area already. First, we consider Web services essentially as concepts in an ontology, where required inputs and the condition under which a requested service actually can be delivered is neglected. Then, we move forward to a more detailed level of description, where inputs and respective preconditions for service delivery are no longer ignored. We show how to adapt and extend the simpler model and matching notions from before to adequately address richer semantic descriptions on this level. The various levels of descriptions are meant to support a wide range of scenarios that can appear in practical applications, requiring different levels of details in the description of Web services and client requests, as well as different precision and performance.

Chapter XIII focuses on semantic search engines and data integration systems. As the use of the World Wide Web has become increasingly widespread, the business of commercial search engines has become a vital and lucrative part of the Web. Search engines are common place tools for virtually every user of the Internet; and companies, such as Google and Yahoo!, have become household names. Semantic search engines try to augment and improve traditional Web search engines by using not just words, but concepts and logical relationships. We believe that data integration systems, domain ontologies and schema based peer-to-peer architectures are good ingredients for developing semantic search engines with good performance. Data integration is the problem of combining data residing at different autonomous sources, and providing the user with a unified view of these data; the problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. Schema-based peer-to-peer networks are a new class of peer-to-peer networks, combining approaches from peer-to-peer as well as from the data integration and Semantic Web research areas. Such networks build upon peers that use metadata (ontologies) to describe their contents and semantic mappings among concepts of different peers’ ontologies. In this chapter, we will provide empirical evidence for our hypothesis. More precisely, we will describe two projects, SEWASIE and WISDOM, which rely on these architectural features and developed key semantic search functionalities; they both exploit the MOMIS (www.dbgroup.unimo.it/Momis/) data integration system. The first, SEWASIE (www.sewasie.org), rely on a two-level ontology architecture: the low level, called the peer level contains a data integration system; the second one, called super-peer level integrates peers with semantically related content (i.e., related to the same domain). The second, WISDOM (www.dbgroup.unimo.it/wisdom/), is based on an overlay network of semantic peers: each peer contains a data integration system. The cardinal idea of the project is to develop a framework that supports a flexible yet efficient integration of the semantic content.