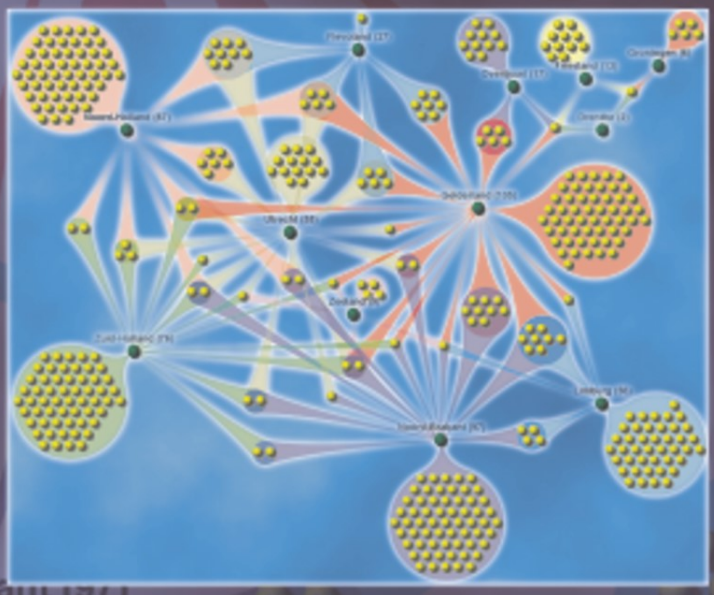
Visualizing the Semantic Web

XML-Based Internet and Information Visualization

Second Edition



Vladimir Geroimenko
and Chaomei Chen (Eds)

Springer

Visualizing the Semantic Web

Vladimir Geroimenko and Chaomei Chen (Eds)

Visualizing the Semantic Web

XML-Based Internet and Information Visualization

Second Edition

With 108 Illustrations

Springer

Vladimir Geroimenko, DSc, PhD, MSc
School of Computing, Communication & Electronics
University of Plymouth
UK

Chaomei Chen, PhD, MSc, BSc
College of Information Science and Technology
Drexel University
USA

British Library Cataloguing Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2005923440

ISBN-10: 1-85233-976-4
ISBN-13: 978-1-85233-976-0

Printed on acid-free paper

1st edition published in 2003 ISBN 1-85233-576-9

Printed in Singapore (TB/KYO)

9 8 7 6 5 4 3 2 1

Springer Science+Business Media
springeronline.com

Preface

The Semantic Web is a vision that has sparked a wide-ranging enthusiasm for a new generation of the Web. The Semantic Web is happening. The central idea of that vision is to make the Web more understandable to computer programs so that people can make more use of this gigantic asset. The use of metadata (data about data) can clearly indicate the meaning of data on the Web so as to provide computers enough information to handle such data.

On the future Web, many additional layers will be required if we want computer programs to handle the semantics (the meaning of data) properly without human intervention. Such layers should deal with the hierarchical relationships between meanings, their similarities and differences, logical rules for making new inferences from the existing data and metadata, and so on. Dozens of new technologies have emerged recently to implement these ideas. XML (eXtensible Markup Language) forms the foundation of the future Web, RDF (Resource Description Framework), OWL (Web Ontology Language) and many other technologies help to erect a "multistory" building of the Semantic Web layer by layer by adding new features and new types of metadata. According to Tim Berners-Lee, the inventor of the current Web and the Semantic Web, it may take up to ten years to complete the building.

The new Web will be much more complex than the current one and will contain enormous amounts of metadata as well as data. How can one manage such information overflow? There are two complementary solutions. The first one is to turn machines into a new, nonhuman, type of Web users, such that they "understand" the meaning of data on the Web and what to do with them, without any involvement of individuals. This is indeed the main purpose of developing a new version of the Web. The second solution is to make the Web more useful for human beings by presenting data and metadata in a comprehensible visual form. XML and related technologies separate data and presentation rules (HTML does not), and that allows us to present data and metadata in any desirable and visually rich form. This is where information visualization comes into the scene. Information visualization in its own right has become one of the hottest topics over the last few years. The sheer size of the Web has provided an ultimate testbed for information visualization technologies. The appeal of information visualization is so strong and far-reaching that one can find a touch of information visualization in almost every field of study concerning accessing and handling large complex information resources.

There are two major approaches to semantic Web visualizations: (1) adopting and applying existing techniques and (2) developing completely new techniques specifically for the new version of the Web. Because the Semantic Web is expected to be a complex multilayered building, its visualizations can vary greatly in their types and nature. In our book, we have tried to explore the most important of them.

The underlying theme of this book is that the Semantic Web and information visualization share many significant issues to such an extent that they demand to be considered together. We call this unifying topic visualization of the Semantic Web. It is an emergent topic. Our book is only one of the initial steps in reflecting the potential of this emerging research field. We hope that this book will stimulate and foster more studies and more books on the visualization of the Second-Generation Web.

The second edition has undergone a number of changes to reflect recent research results, Web standards, developments, and trends: 3 chapters have been removed, 5 new chapters have been added (Chapters 8–11 and 14) as well as 9 remaining chapters have been completely revised and updated.

The new edition of book is arranged as follows.

Chapter 1 introduces the concept and architecture of the Semantic Web and describes the family of XML-related standards that form the technological basis of the new generation of the Web.

Chapter 2 outlines the origin of information visualization and some of the latest advances in relation to the Semantic Web. An illustrative example is included to highlight the challenges that one has to face in seeking for a synergy of information visualization and the Semantic Web.

Chapter 3 presents the Cluster Map, an expressive and interactive visualization mechanism for ontological information. Its use in a number of real-life applications is demonstrated and discussed.

Chapter 4 focuses on the visualization of the semantic structures provided by Topic Maps, RDF graphs and ontologies. The chapter presents and compares several representation and navigation metaphors for the Semantic Web to enhance navigation within complex data sets.

Chapter 5 is a brief introduction to Web Services. It presents the main technologies of Web Services and works through an extended example. Its purpose is a comparison to the Semantic Web, both in terms of intrinsic capabilities (Where do they complement each other? How can they work together?) and in terms of pragmatic context (How fast are they evolving? What tools do they offer to developers?).

Chapter 6 explores recommender systems—particularly those that perform web recommendations using collaborative filtering—and examines how they may be enhanced by the Semantic Web. The chapter discusses a number of interface issues related to filtering and recommending on the Web.

Chapter 7 investigates new technologies—SVG (Scalable Vector Graphics) and X3D (eXtensible 3D)—available for the visualization of 2D and 3D content respectively. The chapter deals with the basics of both technologies and shows that SVG and X3D, based entirely on XML, open essentially new possibilities for the visualization of the Semantic Web.

Chapter 8 introduces GODE (Graphical Ontology Designer Environment), a search paradigm that gives the users the possibility to search both the HTML-based Web and the Semantic Web. This chapter describes how to prepare users for the new flow of information by introducing them to the concept of graphical search step by step. A variety of difficulty levels are described to make sure that everybody can benefit from this approach in different situations. Application areas are discussed for both the simple and the advanced version of GODE.

Chapter 9 shows how a general purpose graph visualization tool can be used for the visualization of large amounts of RDF data. This approach is demonstrated by applying the developed visualization techniques for the RDF data used in a project that investigates the design and development of Web Information Systems on the

Semantic Web. Based on the proposed visualization techniques one can answer complex questions about this data and have an effective insight into its structure.

Chapter10 presents a spring-embedded graph drawing method designed to handle the features of RDF graphs. Mechanisms of the algorithm are presented and then demonstrated and analyzed in case studies of its application to visualizing ontologies and instance data.

Chapter 11 introduces "Semantic Association Networks," a novel means of using semantic web technology to interlink, access, and manage scientific data, services (e.g., algorithms, techniques, or approaches), publications, and expertise (i.e., author and user information) to improve scholarly knowledge, and expertise management.

Chapter 12 investigates the possibility of developing simple but effective interfaces with interactive visually rich content that enable domain experts to access and manipulate XML metadata and underlying ontologies. Native visualizations, that is, those that are integral parts of the process of creating and displaying XML documents, are analyzed in order to utilize their potential in the interface design.

Chapter 13 explores the use of Scalable Vector Graphics (SVG), Geographical Information Systems (GIS) and embedded devices in the medical world, specifically in capturing back pain data. The visualization and analytic capabilities offered by these two technologies are harnessed to provide semantically enhanced solutions for the medical community.

Chapter 14 considers methods for the analysis and visualization of large volumes of Semantic Web data obtained from crawling Friend-of-a-Friend RDF data from LiveJournal, a very large weblog hosting service. Principal Components Analysis and methods from Social Network Analysis are combined to reduce the data to visualizable and socially meaningful units, allowing inferences to be drawn about community formation, social capital, and the relationship between users' interests and their positions within the social network.

Vladimir Geroimenko, DSc, PhD, MSc
Chaomei Chen, PhD, MSc, BSc

Contents

Contributors

Katy Börner, PhD
Assistant Professor
School of Library and Information Science
Indiana University
USA
Website: ella.slis.indiana.edu/∼katy/

Chaomei Chen, PhD, MSc, BSc
College of Information Science and Technology
Drexel University
USA
Website: www.pages.drexel.edu/∼cc345

Christiaan Fluit, MSc
Senior Software Developer
Aduna BV
Amersfoort
The Netherlands
Website: www.aduna.biz

Andrew O. Frank, MBBS, FRCP, Hon DSc
Consultant in Rehabilitation Medicine
Department of Rehabilitation, Medicine and Rheumatology
Northwick Park Hospital and Institute of Medical Research
UK

Flavius Frasincar, BSc, MSc, MTD
PhD Student
Information Systems Group
Department of Computer Science
Eindhoven University of Technology
The Netherlands
Website: www.is.win.tue.nl/∼flaviusf

Larissa Geroimenko, MSc, PhD
Honorary University Fellow
Peninsula Medical School
Universities of Plymouth and Exeter
UK

Vladimir Geroimenko, DSc, PhD, MSc
School of Computing, Communications and Electronics
University of Plymouth
UK
Website: www.tech.plym.ac.uk/soc/staff/vladg

Gheorghita Ghinea, BSc(Hons), MSc, PhD, MBCS
Lecturer in Computing
School of Information Systems, Computing and Mathematics
Brunel University
UK
Website: www.brunel.ac.uk/∼csstggg2

David Gill, BSc, MSc
Researcher
School of Information Systems, Computing and Mathematics
Brunel University
UK

Jennifer Golbeck, PhD
Researcher
Department of Computer Science
University of Maryland
USA
Website: www.cs.umd.edu/∼golbeck/

Benedicte Le Grand, PhD
Associate Professor
LIP6 (Laboratoire d'Informatique de Paris 6)
University Paris 6
France
Website: www.lip6.fr/rp/∼blegrand

Hyoil Han, PhD
Assistant Professor
College of Information Science and Technology
Drexel University
USA
Website: www.cis.drexel.edu/faculty/hhan/

Frank van Harmelen, PhD
Professor
Department of Computer Science
Vrije Universiteit Amsterdam
The Netherlands
Website: www.few.vu.nl/∼frankh

Geert-Jan Houben, PhD
Associate Professor
Information Systems Group
Department of Computer Science
Eindhoven University of Technology
The Netherlands
Website: www.is.win.tue.nl/∼houben

Joseph A. Konstan, AB, MS, PhD
Associate Professor
Department of Computer Science and Engineering
University of Minnesota
USA
Website: www.cs.umn.edu/∼konstan

Paul Mutton, BSc
PhD Student
Computing Department
University of Kent
UK
Website: www.jibble.org

Tom Myers, PhD
Chief Technical Officer
N-Topus Software
Hamilton, NY
USA
Website: www.n-topus.com

Alexander Nakhimovsky, PhD
Associate Professor
Department of Computer Science
Colgate University
USA
Website: cs.colgate.edu/∼sasha

John C. Paolillo, PhD
Associate Professor
School of Library and Information Science
and School of Informatics
Indiana University
Bloomington
USA
Website: ella.slis.indiana.edu/∼paolillo

Lawrence Reeve, BA, ME
Doctoral Student
College of Information Science and Technology
Drexel University
USA
Website: www.pages.drexel.edu/∼lhr24/

John T. Riedl, BS, MS, PhD
Professor
Department of Computer Science and Engineering
University of Minnesota
USA
Website: www.cs.umn.edu/∼riedl

Marta Sabou, MSc
PhD Student
Department of Computer Science
Vrije Universiteit Amsterdam
The Netherlands
Website: www.few.vu.nl/∼marta

J. Ben Schafer, BA, MS, PhD
Assistant Professor
Department of Computer Science
University of Northern Iowa
USA
Website: www.cs.uni.edu/∼schafer

Tacha Serif, BSc, MPhil
Researcher
School of Information Systems, Computing and Mathematics
Brunel University
UK

Michel Soto, PhD
Associate Professor
LIP6 (Laboratoire d'Informatique de Paris 6)
University Paris 5
France
Website: www.lip6.fr/rp/∼soto

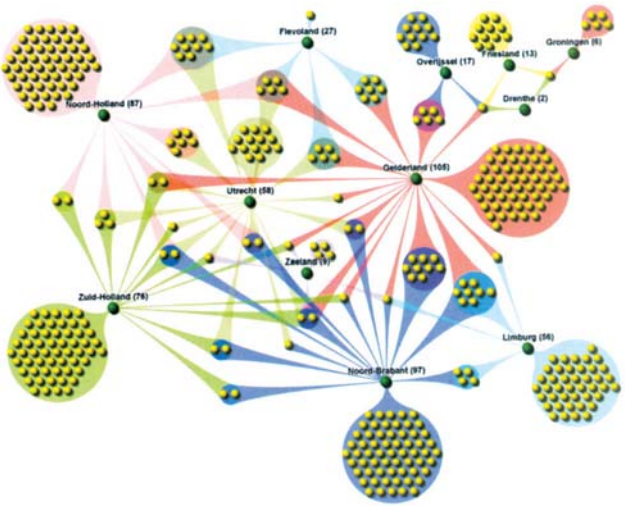
Alexandru Telea, BSc, MSc, PhD
Assistant Professor
Visualization Group
Department of Computer Science
Eindhoven University of Technology
The Netherlands
Website: www.win.tue.nl/∼alext

Leendert W. M. Wienhofen, MSc
Knowledge Engineer
CognIT a.s
Oslo
Norway
Website: www.cognit.no

Elijah Wright, BA, MA
Doctoral Student
School of Library and Information Science
Indiana University
Bloomington
USA
Website: www.geek-guides.com

PART 1

Semantic, Visual, and Technological Facets of the Second-Generation Web



Chapter 1

The Concept and Architecture of the Semantic Web

Vladimir Geroimenko

1.1 From HTML to XML and the Semantic Web

The Internet and especially the World Wide Web belong to the most remarkable achievements in the history of humankind. Without them, it is impossible to imagine current information services, entertainment, and business. Every day more and more ordinary people are getting access to the Web and every of them has possibilities to be an active builder of the Web. For companies and organizations, a presence on the Web has become something equal to their existence in the modern world as such.

The great success of the Web was based on the simple idea of combining hypertext and a global Internet. This idea lead to a revolution, at the heart of which lay HTML (Hypertext Markup Language). Just by following hypertext links, everyone could get desired information from servers around the globe. In a very short period of time, the use of search engines has enhanced these possibilities dramatically. Moreover, information has also become available not only in simple "text + link" format but also in a variety of multimedia forms, such as images, animation, sound, video or virtual reality.

However, the main advantage of HTML—its simplicity—has a reverse side. HTML was created as a means of presenting information on the Web and is about the spatial layout of a presentation, styling fonts and paragraphs, integrating multimedia elements, enabling user interactivity, and the like. Only humans are able to understand the content of such presentations and to deal with them. Because of it, computers have played a passive and an inadequate role in this process—just as the technical means of display, something similar to TV sets or data projectors. They had no real access to the content of a presentation because they were not able to understand the meaning of information on HTML Web pages. On the other hand, the growth of e-commerce created a need for a language that could deal not only with the design (things like "font color" or "image size") but also with the content (things like "item price" or "sale offer") of a presentation. In other words, there was need for a markup language that would go beyond HTML limits in the following aspects: First, it should describe not only the style but also the content of a Web document. Second, it has to mark up this content in such a meaningful way that it would be understandable not only by human beings but also (to a certain extent) by computers. Third, it should be sufficiently flexible to describe specific areas of interest of any of the millions of existing and future businesses, companies, and organizations.

The good news was that such a language had already existed for many years. It was a metalanguage (i.e., a language for defining other languages) called SGML (Standard Generalized Markup Language) that had proved useful in many large publishing applications and was actually used in defining HTML. The bad news was that this language was too complicated, and not very suitable for the Internet.

In early 1998, a new metalanguage was developed by removing the frills from SGML. XML (the eXtensible Markup Language) was intended as a foundation of the next-generation Internet. It very quickly spread through all the major areas of Web-related science, industry, and technology, and the XML revolution began. (For more information about the XML revolution and its semantic approach, see Goldfarb and Prescod, 2003; Hill, 2002; Lassila et al, 2000; Hellman, 1999.)

The XML syntax is easy to read and to write. A real-world example of an XML document is shown in Figure 1.1. Since XML is a plain text format, an XML document can be created using any available text editor and then saved as a file with an extension ".xml". In the above document, the first line called *the XML declaration* is always to be included because it defines the XML version of the document. In this example, the document conforms to the 1.0 specification of XML. The rest of the file is easily understandable by a person, even if he or she has never heard about XML. It is quite obvious that the file describes a book in a book catalog. XML uses *tags* (words or phrases in angle brackets) to mark up the meaning of the data it contains. To show precisely what piece of data they describe, tags usually appear in pairs, *start tag—end tag*. The start tag and the end tag must match each other exactly (since XML is case sensitive) except for a forward slash that has to be included in every end tag after the

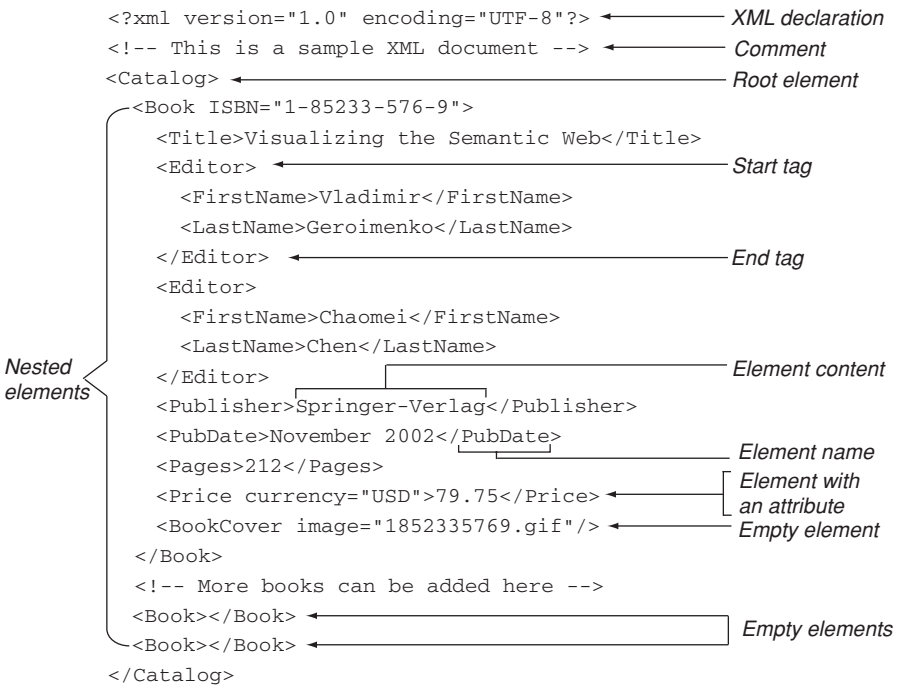


Figure 1.1 The anatomy of an XML document.

opening angle bracket. The combination "the start tag—the content—the end tag," called *an element,* is the main building block of an XML document. Some elements include *attributes* in order to add more information about the content of an element. Attributes are "name-value" pairs enclosed within the start tag of an element. In our example, the element "Book" has an attribute with the attribute name "ISBN" and the attribute value "1-85233-576-9". Some elements can contain no content at all and store data only in their attributes (like the element "BookCover" in our example: `<BookCover image="1852335769.gif"/>`). They are called *empty elements* and combine the start and end tags in one, as shown above. (More information about XML can be found in Geroimenko, 2004; Bates, 2003; Pappamikail, 2002; Dick, 2002; Birbeck et al., 2000; Harold, 1999.)

It is important to emphasize that XML is not a language but a metalanguage, that is, a high-level language specially intended for creating and describing other languages. For a deeper understanding of the nature of XML as a metalanguage, let us point out some contradictory uses of terms. It seems to be quite common and normal to say about specific documents that "they are written in XML." Strictly speaking, however, it is impossible to write even one single document in XML because XML *is not* a language. As a metalanguage, it has no tags at all for describing any specific content and therefore can be used only as a language-definition tool. It means that one has to first develop a specialized XML-based language (something like "MyML") and only after this one has the possibility of creating documents that are written, strictly speaking, not in XML but in MyML (using XML syntax, of course).

Since XML is a metalanguage, it allows a company, organization, or even an individual to create their own domain-specific markup languages giving them considerable flexibility and functionality. At the same time, this most useful feature of the technology can lead to a paradoxical conclusion that the use of XML technology is in principle hardly possible. Indeed, if every company uses its own XML-based language for its specific business, any meaningful communication between them will be out of the question. For example, Company 1 describes its customers and staff using XML tags `<first_name>` and `<last_name>`, Company 2 uses `<given_name>` and `<surname>`, and Company 3 goes for `<Given_Name>` and `<Surname>`. From a human point of view, these metadata tags convey the same meaning. But for computers they are different, even in the case of the languages developed by Company 2 and Company 3 (since XML is case sensitive). To avoid "a Tower of Babel" scenario, significant efforts are required in order to compensate for the unlimited freedom of creating everyone's own markup languages. Basically, there are two possible solutions to this problem. The first is to create special applications that serve as translators between corporate markup languages of interest. The second is to use existing XML vocabularies developed for horizontal or vertical industry as an intermediary language to enable communication and mutual understanding.

Although XML will form the basis of the new generation of the Web, it is not a replacement for HTML. They are designed for different purposes: XML for describing data, HTML for displaying data. XML cannot throw HTML aside because it needs HTML as a means of presenting the data it describes. At the same time, XML forces HTML to change itself. Everything on the future XML-based Web tends to be written or rewritten using the XML syntax. And HTML is not an exception. A new generation of HTML, called XHTML (Extensible HTML), began its life as a reformulation of the latest version of HTML, namely HTML 4.0, in XML. That is, HTML will be replaced by XHTML, not by XML. The latter two languages will complement one another very

Table 1.1 Main features of XML, HTML, and XHTML

XML	HTML	XHTML
Metalanguage	SGML-based language	XML-based language
Intended for describing and structuring data	Intended for formatting and displaying data	Intended for formatting and displaying data
No predefined set of tags	Predefined set of tags	Predefined set of tags
Case sensitive	Case insensitive	Case sensitive. Tag and attribute names must be written in lowercase.
XML documents must be well-formed.	HTML documents do not need to be well-formed.	XHTML documents must be well-formed.
All nonempty elements require end tags.	Some end tags are optional.	All nonempty elements require end tags.
Empty elements must be terminated (e.g., <img />).	Empty elements are not terminated (e.g., <img >).	Empty elements must be terminated (e.g., <img />).
Attribute values must be quoted.	Unquoted attribute values are allowed.	Attribute values must be quoted.
No attribute minimalization is allowed.	The minimal form of an attribute is allowed.	No attribute minimalization is allowed.
Tags must be nested properly, without overlapping.	Tags may be nested with overlapping.	Tags must be nested properly, without overlapping.

well on the future Web. XML will be used to structure and describe the Web data, while XHTML pages will be used to display it. Table 1.1 compares some of the main features of XML, HTML, and XHTML.

Since XML incorporates a revolutionary new approach to the future of the Web, it has numerous advantages and benefits. Here are only some of them:

- XML is an open industry standard defined by the World Wide Web Consortium (W3C). It is a vendor-independent language, endorsed by all the major software producers and market leaders.
- XML is a text format. Since practically all relevant software and devices are able to process text, XML is good for all platforms, devices, programming languages, and software. XML is based on a new multilingual character-encoding system called *Unicode* and because of this enables exchange of information across national and cultural boundaries.
- XML separates the content of an XML document from its presentation rules. As a result, the content or any of its fragments can be presented in many desired forms on a variety of devices, such as computers, mobile phones, personal digital assistants, or printers.
- XML contains self-describing and therefore meaningful information. Metadata tags and attributes allow not only humans but also computers to interpret the meaning of XML data.
- XML is both Web-friendly and data-oriented. It enables us to integrate data from any legacy, current, and future sources such as databases, text documents, and Web pages.

XML forms the technological basis of the Second-Generation Web. Since XML is intended for describing the meaning of Web data (or, in other words, their semantics), the emerging XML-based Web is also called the "Semantic Web." The concept of the Semantic Web is based on a vision of Tim Berners-Lee, the inventor of the World Wide Web and the Director of the World Wide Web Consortium. In particular,

this vision was expressed in his *Semantic Web Road Map* document (Berners-Lee, 1998), his book *Weaving the Web* (Berners-Lee, 1999), and his speech at *XML 2000 Conference* (Berners-Lee, 2000). Recently many online and magazine articles have appeared that provide a more or less clear explanation of what the Semantic Web actually is (for example, Bosak and Bray, 1999; Dumbill, 2000; Decker et al., 2000; Dumbill, 2001; Berners-Lee, Hendler, and Lassila, 2001; Palmer, 2001; Heflin and Hendler, 2001; Cover, 2001; Daconta, Obrst, and Smith, 2003; Davies, 2003; Fensel et al., 2003; Passin, 2003; Geroimenko, 2004; Antoniou, G., and van Harmelen, F., 2004). Some Web sites are specially devoted the Semantic Web and its key technologies (for instance, www.semanticweb.org, www.w3c.org/2001/sw/, www.xml.com and http://www.sigsemis.org).

The main idea of the Semantic Web is to delegate many current human-specific Web activities to computers. They can do them better and quicker that any individuals. But to enable computers to do their new jobs, humans need to express Web data in a machine-readable format suitable for completely automated transactions that do not require human intervention. This can be achieved by (1) identifying all Web and real-world resources in a unique way; (2) adding more and more metadata to Web data using XML, RDF, and other technologies; (3) creating general and domain-specific ontologies using RDF Schemas, OWL, and similar technologies; and (4) enabling computers to use simple logic in order to deal with Web data in a meaningful way (see Figure 1.2). For example, computers should "understand" not only what a bit of data means but also that other pieces of data, located somewhere on the Web, mean the same even if they look different (for instance, `<last_name>` and `<surname>`). The idea of the "sameness" of Web data will provide a new solution to many current problems, such as more meaningful searches on the Web. For example, if you are looking for "Wood" and specifying this word as a person's last name, you will get back only topics related to people, not to timber, firewood, or forest. The use of RDF, OWL, or other high-level metadata technologies can make Web searches even more powerful and therefore more successful. Computers will be able to automatically convert complex expressions from one domain-specific XML language into another in order to process them.

Although the above considerations hopefully provide the reader with some general understanding of the concept, the question "What is the Semantic Web?" is not a simple one. Computer scientists and Web developers from different fields (e.g.,

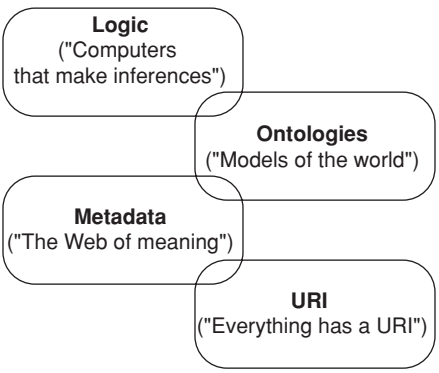


Figure 1.2 Conceptual building blocks of the Semantic Web.

e-commerce, networking, knowledge management, or artificial intelligent) tend to have quite contradictory views. The new generation of the Web will be so complex and multifaceted that it allows people with almost any background to find all that they need or want to see. For one group of researchers and developers, the Semantic Web is a brain for humankind (Fensel and Musen, 2001), for another it is a database for communicating invoices, timetables, and other similar information in XML. The spectrum of such views becomes even more diverse when the matter in question is how to implement the concept of the second-generation Web, what technologies to use, and in what direction to move. Indeed, there are many ways of constructing the Web and many technologies that can be employed.

It is interesting to analyze the conceptual basis of the Semantic Web from a methodological point of view because it helps in developing a deeper understanding of the nature of this new-generation Web. As is generally known, semantics is a branch of linguistics concerned with meaning. The Semantic Web, in contrast to the current HTML-based Web, is all about the meaning of data. For instance, if 100 means $100 it can be described in XML as `<Price currency="GBP">100</Price>`. But if 100 means a speed it might be marked up as `<Speed><mph>100</mph></Speed>`. Obviously, in these cases the same syntax ("`100`") has different semantics. It is not just a number any more, it is something meaningful and therefore much more useful. It is important to keep in mind that we are here talking about data that are meaningful not only for humans but in the first instance for computers. Human beings do not need any markup metadata tags for understanding current Web pages. For them, the existing Web is already the semantic one. But for machines it is meaningless, and therefore nonsemantic (perhaps, the only exception is the `<meta>` tag that can be placed in the head section of a HTML page in order to add nondisplayable information about the author, keywords, and so on). Consequently, a question arises about the point at which the Web becomes meaningful for computers (in other words, becomes "semantic") and to what extent it can be possible.

A fairly common opinion is that the point where the Semantic Web actually starts is not XML (since this language is "not semantic enough") but RDF, OWL, Topic Maps, and other more specialized metadata technologies. The proposed architecture of the second-generation Web will be discussed later in this chapter. Our view is based on a multilevel conceptual model of a machine-processable Web. In our opinion, since XML allows us to add meanings to Web data and these meanings can in principle be understandable by computers, we can talk about XML as the first level of the Semantic Web (see also, for example, Patel-Schneider and Simeon, 2002). This new generation of the Web begins where XML and its companion (XHTML) are replacing HTML. Of course, XML is far from being enough to construct the Semantic Web as such. The human system of meanings, and even the current Web resources, is not so simple that they can be described using XML alone. Therefore, the architectures of the Semantic Web need to add more and more new levels on top of XML. It is impossible to say what kind of technology will be successfully implemented in the future in order to construct a complex hierarchical system of multilayered semantic information that would enable computers to understand a little bit more after adding a new layer. As of today, for example, RDF and OWL seem to be the most suitable technologies for adding more meanings to the Web data and resources. At the same time, XML Topic Maps look like a promising candidate for this job as well.

Thus, the Semantic Web has originated in XML, which provides a minimal (but not zero) semantic level and this version of the Web will be under development for a long

time ahead by adding extra levels of meaning and by using new specialist technologies to do this. As a result, computers will be able to "understand" more and to put this to good use. Although it will work, in reality we can talk about meanings, understandable by computers, only in a metaphorical sense. For machines any XML element is still meaningless. For them, the element `<Price in_GBP="100"/>` makes no more sense than, for example, the element `<Kdg9kj Drdsf="100"/>`. Paradoxically, adding new levels of semantics (using RDF, OWL, or any other future technologies) does not change the situation. As long as computers will not possess a very special feature, similar to human consciousness, they will not be able to understand any meanings at all. However, by using computers the creators of the Semantic Web will be able to simulate some human understanding of meaningful Web data and, what is most important, to force machines to make practical use of this.

1.2 The XML Family of Technologies

As a metalanguage, XML is simple and therefore easy to learn and use. However, there are countless numbers of "big" and "small" languages written in XML—the family of XML-based languages. The members of the XML family can be described and classified in several different ways, such as in Salminen, 2001; Vint, 2001; Bain and Shalloway, 2001; Sall, 2002; Turner, 2002, etc. Our approach is shown in Figure 1.3. The core of the family is formed by numerous custom XML-based languages, such as NewsML or SportXML. Actually, this is the most important part of the XML family since custom languages describe the content of XML documents. In other words, they are intended for marking up the meaning of domain-specific Web data such as "car price." Any organization and even any individual are free to create their own XML-based languages.

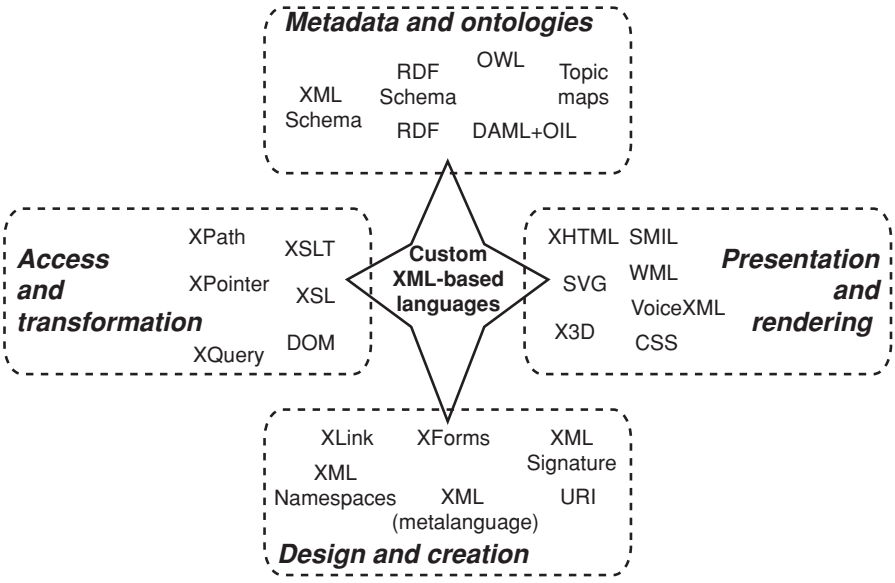


Figure 1.3 The structure and main members of the XML family.

Besides the custom languages, XML has a considerable number of specifications that help to realize its potential. This type of XML-related languages (also known as the XML family of technologies) is mostly being developed by the W3C. Since this area is evolving extremely quickly, the only possibility to find out its state of the art is to visit the Consortium Web site (www.w3.org) in order to comprehend the situation as it develops literally day by day. It is good to know that all results of W3C development activities are presented as *Technical Reports,* each of which can reach one of the following levels of its maturity (from lower to higher): *Working Draft, Candidate Recommendation, Proposed Recommendation, and Recommendation.* A Recommendation represents consensus within W3C and is a *de facto* Web standard.

The XML family of technologies can be divided into four groups, in accordance with their main functionalities: (1) enabling the design and creation of XML-based languages and documents; (2) providing a means of accessing and transforming XML documents; (3) enabling efficient presentation and rendering of XML data; and (4) describing the meaning of XML data using metadata and ontologies. The overview of the XML family provided below is very condensed and just describes their main purpose and meanings of acronyms:

1. XML technologies involved in the *design and creation* of XML-based languages and their documents:
   - *XML*—a metalanguage that allows the creation of markup languages for arbitrary specialized domains and purposes. This is XML as such.
   - *XML Namespaces* prevent name collision in XML documents by using qualified element and attribute names. A qualified name consists of a namespace name and a local part. The namespace name is a prefix identified by a URI (Uniform Resource Identifier) reference.
   - *XLink* provides facilities for creating and describing links between XML documents, including two-way links, links to multiple documents, and other types of linking that are much more sophisticated than in HTML.
   - *XForms* specifies the use of Web form technique on a variety of platforms, such as desktop computers, television sets, or mobile phones.
   - *XML Signature* provides syntax and processing rules for XML digital signatures.
2. XML technologies that are mostly intended for *accessing and transforming* XML documents:
   - *XSL (Extensible Stylesheet Language)* consists of*XSL-T (XSL Transformations)* and *XSL-FO (XSL Formatting Objects)* and is a language for transforming XML documents into other XML documents and for rendering them, for example, into HTML, Braille, audible speech, and many other forms on a variety of platforms and devices, as shown in Figure 1.4.
   - *DOM (Document Object Model)* is an application programming interface that describes an XML document as a tree of nodes, and defines the way the nodes are structured, accessed, and manipulated.
   - *XPath* specifies how to address parts of an XML document.
   - *XPointer* extends XPath by defining fragment identifiers for URI references.
   - *XQuery* is a language for querying XML data that considers an XML file as a database.
3. XML technologies responsible for *presenting and rendering* XML documents:
   - *CSS (Cascading Style Sheets)* is a simple language for specifying style sheets for rendering XML documents.
   - *XHTML (Extensible HTML)* is a reformulation of HTML 4.0 into XML.

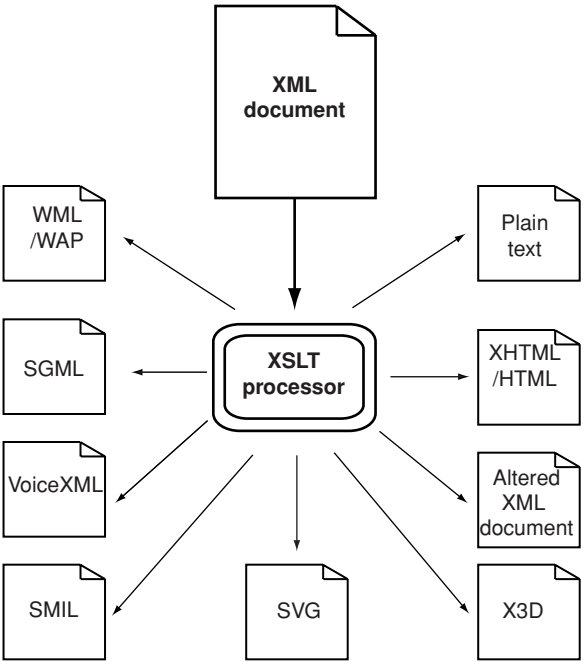
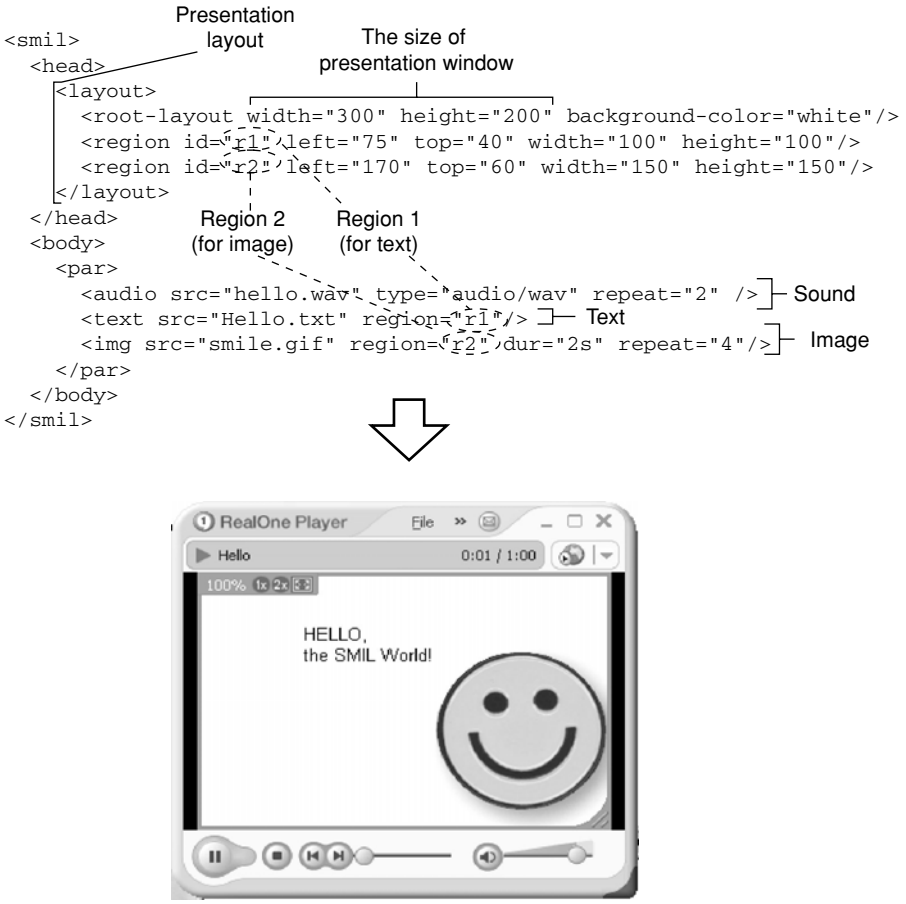


Figure 1.4 Transforming an XML document into other formats using XSLT.

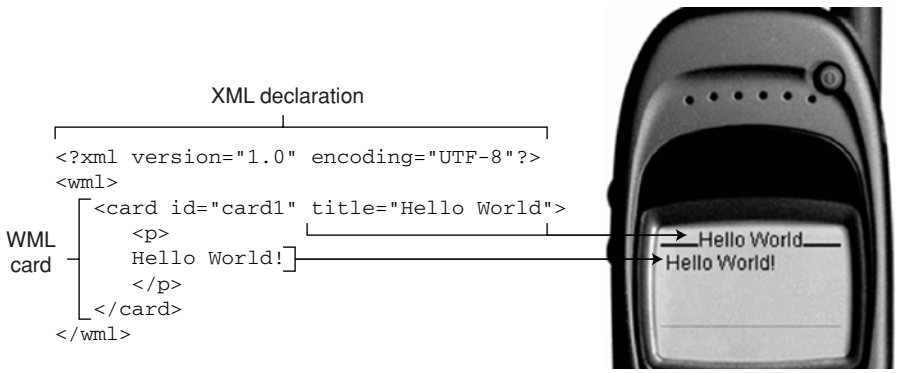
- *SVG (Scalable Vector Graphics)* is a language for describing two-dimensional vector and mixed vector/raster graphics in XML.
- *X3D (Extensible 3D)* is a markup language that allows VRML (Virtual Reality Markup Language) content to be expressed in terms of XML.
- *SMIL (Synchronized Multimedia Integration Language)* is used to create multimedia Web presentations by integrating, synchronizing, and linking independent multimedia elements such as video, sound, and still images. See Figure 1.5 for an example.
- *WML (Wireless Markup Language)* is a language for presenting some content of Web pages on mobile phones and personal digital assistants. Figure 1.6 illustrates the use of WML.
- *MathML (Mathematical Markup Language)* deals with the representation of mathematical formulas.

4. XML technologies that are specially intended for expressing *metadata and ontologies*:
   - *XML Schema* defines types of elements an XML document can contain, their relationships and the data they can include. A schema can be used for both creating and validating a specific class of XML documents. An XML document must be *well-formed,* that is, conform to the syntactic rules of XML, and additionally it can be *valid,* that is, conform to the rules of a schema if it has one.
   - *RDF (Resource Description Framework)* is one of the cornerstones of the Semantic Web. It defines a simple data model using triples (subject, predicate, object), where subject and predicate are URIs and the object is either a URI or a literal.




```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200" background-color="white"/>
      <region id="r1" left="75" top="40" width="100" height="100"/>
      <region id="r2" left="170" top="60" width="150" height="150"/>
    </layout>
  </head>
  <body>
    <par>
      <audio src="hello.wav" type="audio/wav" repeat="2" />
      <text src="Hello.txt" region="r1" />
      <img src="smile.gif" region="r2" dur="2s" repeat="4"/>
    </par>
  </body>
</smil>
```

Figure 1.5 An example SMIL document and its visual rendering.




```
<?xml version="1.0" encoding="UTF-8"?>
<wml>
  <card id="card1" title="Hello World">
    <p>
    Hello World!
    </p>
  </card>
</wml>
```

Figure 1.6 A simple WML document and its view in a mobile phone browser.

Its allows one to describe and to retrieval Web data in a way that is similar to using catalogue cards for describing and finding books in a library.

- *RDF Schema* is a key technology that defines classes, properties, and their interrelation of RDF data model in order to enable computers to make inferences about the data collected from the Web.
- *DAML + OIL (DAPRA Agent Markup Language + Ontology Inference Layer)* are languages for expressing ontologies that extend RDF Schema.
- *OWL (Web Ontology Language)* is the latest language for defining and instantiating Web ontologies to enable machine-processable semantics. It can be used to explicitly represent the meaning of terms in a vocabulary and the relationships of those terms. OWL is a revision of the DAML+OIL Web ontology language. It is based on XML, RDF, and RDF Schema but goes beyond these languages by providing more facilities for expressing the semantics of Web data.
- *Topic Maps* is a technology that allows one to build a structured semantic network above information resources using topics and topic associations. This enables the description and retrieval of Web data in a way that is similar to using the index of a book to find the pages on which a specific topic is covered.

1.3 The Architecture of the Semantic Web

The first attempt to give a high-level plan of the architecture of the Semantic Web was made by Tim Berners-Lee in his *Semantic Web Road Map* (Berners-Lee, 1998) and refined in his following publications and presentations (Berners-Lee, 1999; Berners-Lee, 2000; Berners-Lee, Hendler, and Lassila, 2001). According to him, the Semantic Web will be built by adding more layers on the top of existing ones and may take around 10 years to complete. Figure 1.7 shows Semantic Web architectural relationships. It is based on the famous "layer cake" diagram, presented by Tim Berners-Lee at the *XML 2000 Conference* (Berners-Lee, 2000).

Most of the current Semantic Web technologies belong to the XML family and it is almost certain that all future layers will be XML-based as well. XML is the foundation of the new generation of the Web. XML is powered by the URI, Namespaces, and

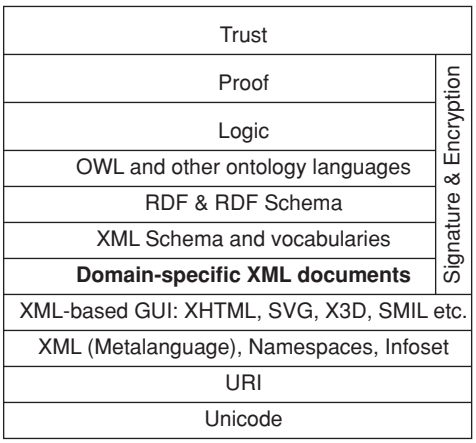


Figure 1.7 The architecture of the Semantic Web.

Unicode technologies. URIs are intended for identifying arbitrary resources in a unique way; they may or may not "point" to resources or serve for their retrieval. Together with XML Namespaces, they allow everyone to uniquely identify elements within an XML document, without the danger of a name collision. Unicode, as a multilingual character-encoding system, provides opportunities for describing Web resources in any natural language and therefore enables exchange of information across national and cultural boundaries.

XML documents form the most substantial layer of the Semantic Web, because they embrace, strictly speaking, not only documents with the domain-specific content (such as product catalogues) but also almost all "technological" documents written in XML (such as XSLT, RDF, or OWL). XML is a universal format for storing and exchanging data and metadata on the new version of the Web.

The XML document layer is to interface with the two main types of Semantic Web users: humans and computers. Although an XML document is, as a rule, saying nothing about how to present its content to an individual, this is not a problem because plenty of formatting and rendering technologies (both legacy and XML-based) are available for displaying XML data in human-readable form (for example, Flash, Java, HTML, XHTML, XSLT, SVG, X3D, etc.—see Figure 1.8). Interfacing with computers and especially autonomous software agents is a much more difficult problem. To make XML documents "understandable" and processable by computers, a hierarchy of special layers should be added in order to achieve the only goal—to make meanings of data clear to nonhuman users of the Web. No one knows how many extra layers will be needed in the future and what kind of new technologies should be implemented.

RDF seems to be one of the main building blocks of the today's Semantic Web, giving a domain-neutral mechanism for describing metadata in a machine-processable format (see, for example, Hjelm, 2001). RDF is built around the following three concepts: resources, properties, and statements. Resources can be anything that can be referred to by a URI (from an entire Web site to a singe element of any of its XML or XHTML pages). A property is a specific characteristic or relation that describes a resource. RDF statements are composed of triplets: an object (a resource), an attribute (a property), and a value (a resource or free text). They are the formal implementation of a simple idea expressed in natural-language sentences of the following type: "Someone is the *creator/owner/etc.* of something else." RDF statements describe additional facts about an XML vocabulary in an explicit, machine-readable format, and therefore allow computers to understand meanings in context. In this way, they act for human abilities of implicit common-sense understanding of the underlying real-world concepts.

RDF Schemas provide appropriate data typing for RDF documents by defining domain-specific properties and classes of resources to which those properties can be applied. These classes and properties are organized in a hierarchical way by using the basic modeling primitives of the RDF Schema technology: *class* and *property* definitions, and *subclass-of* and *subproperty-of* statements.

Topic Maps are a standard defined by the International Organization for Standardization (ISO). Like RDF, they are intended to annotate Web resources in order to make them understandable by computers (see, for instance, Lacher and Decker, 2001). Topic Maps technology can be used to build a semantic network above information resources (some sort of "GPS of the information universe") and thus to enhance navigation in very complex data sets. A Topic Map is an XML document that is based on the following fundamental concepts: *topics, associations, and occurrences.*

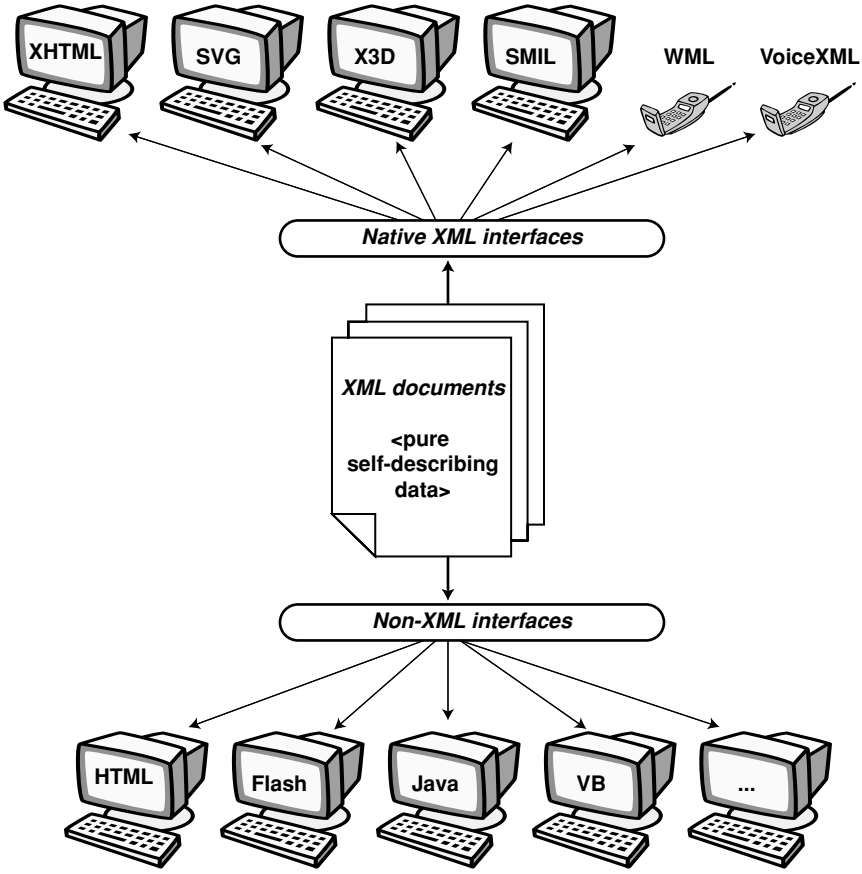


Figure 1.8 Two types of graphical user interfaces for XML documents.

Similar to an entry in an encyclopedia, a topic can represent any subject and therefore almost everything in a Topic Map is a topic. Topics are connected by associations and point to resources through occurrences. An association expresses a relationship between topics. A topic can be linked to one or more occurrences—information resources that are somehow related to this topic. For example, "the Semantic Web" and "the Web" are topics that have an association "is a new version of" and several assurances (places where they are mentioned, including not only text but also images) in this book. The relationship between RDF and Topic Maps technologies is not simple. On the one hand, Topic Maps are in competition with RDF. They provide an effective knowledge-centric approach to metadata in contrast to the resource-centric RFD technique. On the other hand, Topic Maps may be used to model RDF and vice versa.

Ontologies are another fundamental technology for implementing the Semantic Web (Ding, 2001; Fensel, 2001; Fensel et al., 2001; Gomez-Perez and Corcho, 2002; Kim, 2002). They establish a common conceptual description and a joint terminology between members of communities of interest (human or autonomous software agents). An ontology is an explicit specification of a conceptualization (Gruber, 1993).

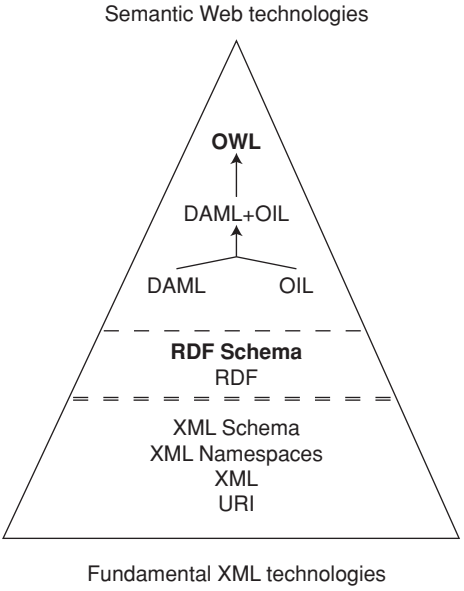


Figure 1.9 The genesis of OWL and its relationships to other XML technologies.

An XML schema can be regarded as a primitive ontology. The construction of the new Web requires ever more expressive languages for describing and matching a variety of "horizontal" and "vertical" ontologies. The latest development in this area includes, on the one hand, generating a variety of ontologies within vertical marketplaces—such as Dublin Core, Common Business Library (CBL), Commerce XML (cXML), Open Catalog Format (OCF), and RosettaNet—and, on the other hand, creating new improved languages that extend RDF Schema by richer sets of modeling primitives for representation of Boolean expressions, property restrictions, axioms, etc., such as OIL (Ontology Inference Layer) and DAML + OIL (DARPA Agent Markup Language + OIL).

At present, the leading semantic Web technology is OWL (Web Ontology Language), which is a revision of the DAML + OIL Web ontology language. It is based on XML, RDF, and RDF Schema but goes beyond these languages by providing more facilities for expressing the semantics of Web data. OWL extends RDF Schema with a richer OWL vocabulary that provides advanced inferencing capabilities. OWL relationships to other XML technologies are shown in Figure 1.9.

The highest layers of the Semantic Web are yet to be fully developed. The logic layer will contain logical rules that allow computers to make inferences and deductions in order to derive new knowledge. The proof layer will implement languages and mechanisms for distinguishing between Web resources with different levels of trustworthiness. The logic and proof, together with the XML Digital Signature, will enable us to construct the "Web of Trust," where autonomous software agents will be able and allowed to undertake important tasks such as finding and buying expensive goods without any human intervention.

Thus, the Semantic Web will make maximal use of both Web resources and computers. Computers will not be just devices for posting and rendering data, but will deal

with resources in a meaningful and competent way. Current technologies for adding multilayered machine-processable semantics and heuristics make this development possible.

1.4 References

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

Bain, S.L., Shalloway, A. (2001). *Introduction to XML and its Family of Technologies.* Net Objectives.

Bates, C. (2003). *XML in Theory & Practice.* John Wiley & Sons.

Berners-Lee, T. (2000). *Semantic Web–XML2000.* Available: http://www.w3.org/2000/Talks/1206-xml2k-tbl/.

Berners-Lee, T. (1999). *Weaving the Web.* Harper, San Francisco.

Berners-Lee, T. (1998). *Semantic Web road map.* Available: http://www.w3.org/DesignIssues/Semantic.html.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American*: May. Available: http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html .

Birbeck, M. et al. (2000). *Professional XML.* Wrox Press Inc.

Bosak, J., Bray, T. (1999). XML and the second-generation Web. *Scientific American*: May. Available: http://www.scientificamerican.com/1999/0599issue/0599bosak.html.

Cover, R. (2001). *XML and "The Semantic Web."* Available: http://www.coverpages.org/xmlAndSemanticWeb.html.

Daconta, M.C., Obrst, L.J., Smith, K.T. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management.* John Wiley & Sons.

Davies, J. (2003). *Towards the Semantic Web: Ontology-driven Knowledge Management.* John Wiley & Sons, Ltd.

Decker, S. et al. (2000). The Semantic Web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–73.

Dick, K. (2002). *XML: A Manager's Guide* (2nd Edition). Addison Wesley Professional.

Ding, Y. (2001). A review of ontologies with the Semantic Web in view. *Journal of Information Science,* 27(6):377–384.

Dumbill, E. (2000). *The Semantic Web: A primer.* Available: http://www.xml.com/lpt/a/2000/11/01/semanticweb/index.html.

Dumbill, E. (2001). *Building the Semantic Web.* Available: http://www.xml.com/pub/a/2001/03/07/buildingsw.html.

Fensel, D. (2001). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer-Verlag, Berlin, Heidelberg, New York.

Fensel, D., Musen, M. (2001). The Semantic Web: A brain for humankind. *IEEE Intelligent Systems,* 15(2):24—25.

Fensel, D. et al. (2001). OIL: An ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems,* 16(2):38–45.

Fensel, D. et al. (2003). *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential.* The MIT Press.

Geroimenko, V. (2004). *Dictionary of XML Technologies and the Semantic Web.* Springer, Berlin, Heidelberg, New York.

Goldfarb, C.F., Prescod, P. (2003). *Charles F. Goldfarb's XML Handbook* (5th Edition). Prentice Hall.

Gomez-Perez, A., Corcho, O. (2002). Ontology languages for the Semantic Web. *IEEE Intelligent Systems,* 17(1):54–60 .

Gruber, T.R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition,* 5(2):199–200.

Harold, E.R. (1999). *XML Bible.* Hungry Minds, Inc.

Heflin, J., Hendler, J. (2001). A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59.

Hellman, R. (1999). A semantic approach adds meaning to the Web. *Computer*, December: 13–16.

Hill, A. (2002). The XML revolution. *Financial Technology*, 1(14):5–6.

Hjelm, J. (2001). *Creating the Semantic Web with RDF.* John Wiley & Sons.

Kim, H. (2002). Predicting how ontologies for the Semantic Web will evolve. *Communications of the ACM*, 45(2):48–54.

Lacher, M.S., Decker, S. (2001). RDF, Topic Maps, and the Semantic Web. *Markup Languages: Theory & Practice*, 3(3):313–331.

Lassila, O. et al. (2000). The Semantic Web and its languages. *IEEE Intelligent Systems*, 15(6):67–73.

Palmer, S. (2001). *The Semantic Web: An introduction.* Available: http://infomesh.net/2001/swintro/.

Pappamikail, P. (2002). *XML by Stealth: A Manager's Guide to Strategic Implementation*. John Wiley & Sons.
Passin, T. (2003). *Semantic Web Field Guide*. Manning Publications.
Patel-Schneider, P.F., Simeon, J. (2002). Building the Semantic Web on XML. *The Semantic Web—ISWC 2002. First International Web Conference, Proceedings*, pp. 147–161.
Sall, K.B. (2002). *XML Family of Specifications: A Practical Guide*. Addison-Wesley Professional.
Salminen, A. (2001). *Summary of the XML family of W3C languages*. Available: http:// www.cs.jyu.fi/ ~airi/xmlfamily-20010806.html.
Turner, R. (2002). *The Essential Guide to XML Technologies*. Prentice Hall PTR.
Vint, D. (2001). XML*Family of Specifications*. Manning Publications, Greenwich, CT.

Chapter 2
Information Visualization and the Semantic Web

Lawrence Reeve, Hyoil Han, and Chaomei Chen

2.1 Introduction

The appeal and potential of information visualization is increasingly recognized in a wide range of information systems. Information visualization aims to produce graphical representations of abstract information structure for human users. The Semantic Web sets out the blueprint of the second generation of the ever-popular World Wide Web, aiming to provide a universal descriptive framework of resources that can be utilized by software agents. Full implementation of the Semantic Web requires widespread availability of semantic annotations for existing and new documents on the Web. Semantic annotations are to tag ontology class instance data and map it into ontology classes. Information visualization and the Semantic Web may complement each other on a number of fundamental issues concerning the organization of and access to large-scale information resources. On the other hand, the two distinct research fields differ in some fundamental ways in terms of how semantics is defined and represented. It is important for designers and users to be able to distinguish the key differences as well as the major similarities between the two. In this chapter, we outline the origin of information visualization and some of the latest advances in relation to the Semantic Web. An illustrative example is included to highlight the challenges that one has to face in seeking a synergy of information visualization and the Semantic Web.

2.2 The Semantic Web

The Semantic Web is the second generation of the 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 (Berners-Lee et al., 2001). It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. The Web can reach its full potential if it becomes a place where data can be shared and processed by automated tools as well as by people. Tim Berners-Lee in 1998 sketched the *Roadmap for the Semantic Web* (Berners-Lee, 1998), which has become the most-cited article about the Semantic Web.

A few topics in the Semantic Web are fundamentally related to information visualization, including Topic Maps and SVG. The Topic Map community

(http://www.egroups.com/group/xtm-wg) has been finding increasing synergy with the RDF data model.

The World Wide Web was originally built for human consumption, and although everything on it is *machine-readable*, these data are not *machine-understandable*. One of the fundamental requirements of Semantic Web is to annotate Web data with ontology to accomplish machine-understandable Web.

It is very hard to automate anything on the Web. The proposed solution is to use *metadata* and *ontology* to describe the data on the Web. Metadata are "data about data," or "data describing Web resources." The distinction between "data" and "metadata" depends on particular applications; the same resource could be interpreted in both ways simultaneously. Ontology represents a conceptual structure of a specific application domain.

Resource Description Framework (RDF) enables automated processing of Web resources. Research in RDF has identified a variety of application areas. In *resource discovery*, RDF may lead to better and more efficient search engines. RDF may facilitate the description of the content and content relationships available at a particular Web site or digital library, facilitate knowledge sharing and exchange, as well as many other appealing possibilities (Berners-Lee, 1998; Berners-Lee et al., 2001).

2.2.1 Visualization Issues

As far as the Semantic Web is concerned, visualizing the structure of a formally defined information resource domain is important not only for designers and developers of the Semantic Web, but also for the business and individual users of the first generation of the Web.

One of the most active areas of the Semantic Web is the development of a variety of tools for authoring, extraction, visualization, and inference RDF. For example, RDFSViz(http://www.dfki.uni-kl.de/frodo/RDFSViz/) provides a visualization service for ontologies represented in RDF Schema. The RDFSViz tool was implemented in the FRODO project (A Framework for Distributed Organizational Memories) at DFKI Kaiserslautern (German Research Center for Artificial Intelligence). It uses the open-source graph drawing program Graphviz from AT&T and Lucent Bell Labs (see Figure 2.1, for example). W3C also provides RDF validation service (http://www.w3.org/RDF/Validator/), which can visualize general RDF models. The OntoViz plugin (http://protege.stanford.edu/plugins/ontoviz/ontoviz.html) is a similar tool for the knowledge acquisition tool Protégé-2000. OntoViz is much more configurable than RDFSViz.

The OntoViz Tab (http://protege.stanford.edu/plugins/ontoviz/ontoviz.html) can visualize Protégé-2000 ontologies with the help of highly sophisticated graph visualization software, namely GraphViz from AT&T.

Once the metadata are available, it is relatively straightforward to use them, to visualize them. It is, however, much harder and more time- and resource-consuming to produce metadata in the first place. Even the quality of metadata generated by domain experts is subject to changes in domain knowledge. In practice, there are situations in which metadata are impossible to generate without an overall understanding of a large-scale complex body of data. Information visualization over the last decade has been devoted to searching for insightful images from such data. A more in-depth analysis of the nature of the problems may help us clarify the scope of the Semantic Web

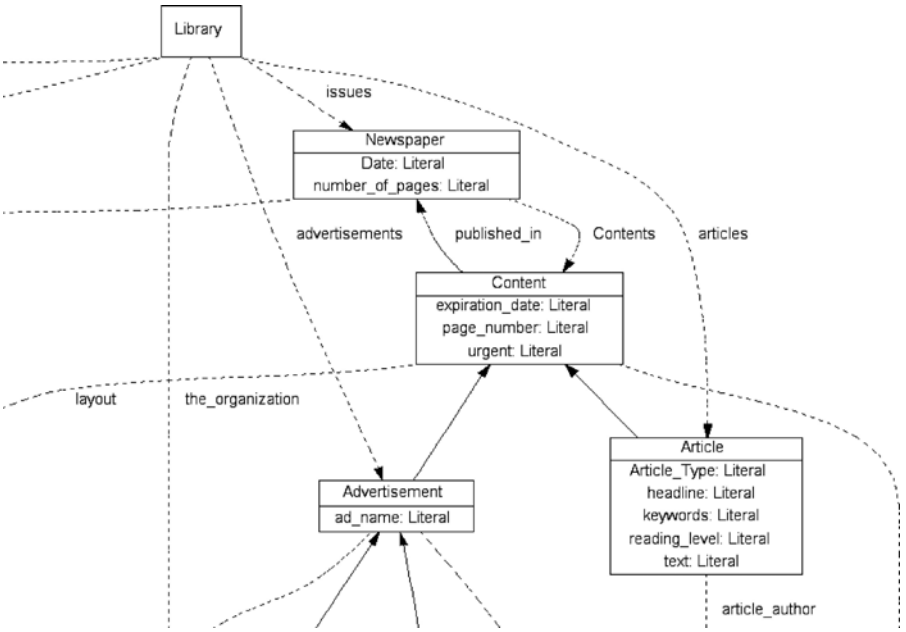


Figure 2.1 A visualization of an RDF model.

approaches as well as how information visualization can benefit from the Semantic Web.

2.2.2 Semantic Annotation

Full implementation of the Semantic Web requires widespread availability of semantic annotations for existing and new documents on the Web. Semantic annotations are to tag ontology class instance data and map them into ontology classes. Manual annotation is more easily accomplished today, using authoring tools such as Semantic Word (Tallis, 2003), which provide an integrated environment for simultaneously authoring and annotating text. However, the use of human annotators is often fraught with errors due to factors such as annotator familiarity with the domain, amount of training, personal motivation, and complex schemas (Bayerl et al., 2003). Manual annotation is also an expensive process, and often does not consider that multiple perspectives of a data source, requiring multiple ontologies, can be beneficial to support the needs of different users. For example, vision-impaired users can use annotations to provide faster navigation through a Web site (Yesilada et al., 2003), while sighted users can use annotations of the same document to provide a detailed view of a domain. A further problem with manual annotation is the volume of existing documents on the Web that must be annotated to become a useful part of the Semantic Web. Manual semantic annotation has led to a knowledge acquisition bottleneck (Maedche and Staab, 2001).

To overcome the annotation acquisition bottleneck, semiautomatic annotation of documents has been proposed. Semiautomatic means, as opposed to completely automatic, are required because it is not yet possible to automatically identify and classify

all entities in source documents with complete accuracy. All existing semantic annotation systems rely on human intervention at some point in the annotation process, using the paradigm of balanced cooperative modeling (Maedche and Staab, 2001). Automated annotation provides the scalability needed to annotate existing documents on the Web, and reduces the burden of annotating new documents. Other potential benefits are consistently applying ontologies, and using multiple ontologies to annotate a single document. *Semantic annotation platforms* (SAPs) provide support for information extraction (IE) implementations, ontology and knowledgebase management, access APIs, storage (e.g., RDF (W3C, 2004) repositories), and user-interfaces for ontology and knowledgebase editors (Popov et al., 2003).

SAPs can be classified based on the type of annotation method used. There are two primary categories, pattern-based and machine-learning-based. In addition, platforms can use methods from both types of categories, called *multistrategy,* in order to take advantage of the strengths, and compensate for the weaknesses, of the methods in each category. The most common techniques SAPs use are manually created rules (Maynard, 2003), pattern matching (Popov et al., 2003), automatic discovery of patterns (Dingli et al., 2003), and wrapper induction, either linguistic (Handschuh et al., 2002) or structural based (Mukherjee et al., 2003). While the machine-learning methods, such as those used by Amilcare (Handschuh et al., 2002), usually perform better (Vargas-Vera et al., 2002), the rule-based MUSE system using conditional processing has shown that rule-based systems can equal the performance of machine-learning-based systems (Maynard, 2003).

All SAPs require some type of lexicons and resources. Rule-based systems require rules, pattern discovery systems require an initial set of seeds, machine-learning systems require a training corpus (usually annotated), while others require the construction of dictionaries for named-entity recognition. Ontologies must also be supplied to SAPs, since the semantic annotations are to tag ontology class instance data and map it into ontology classes. Some ontologies are simple taxonomies and structures such as address books, while others are complex ontology implementations with relationships defined. These ontologies bootstrap the process of annotation by providing enough information to begin annotating. Some systems contain a feedback cycle, where the ontology and a supporting knowledgebase learn more information each time the annotation process is run against a document set. This feedback cycle results in more accurate annotations over time (Popov et al., 2003). Pattern-based systems often require the manual generation of rules. The notable difference is the recent work done with PANKOW (Cimiano et al., 2004) to automatically discover an initial set of seed patterns. This approach can be contrasted with the MUSE system (Maynard, 2003), where rules must be completely defined before the annotation process is started.

SAP architectures can be categorized as extensible or not. Non-extensible architectures usually focus on a single domain, method, or toolkit. For example, AeroDAML relies on Aerotext (Kogut and Holmes, 2001), and SemTag (Dill et al., 2003) relies on its own taxonomy labeling matching. Extensible SAP architectures allow various system components to be replaced or extended with other components. Examples are MUSE (Maynard, 2003) and Ont-O-Mat (Handschuh et al., 2002). Extensible SAPs allow newer annotation methods to be tested and integrated while reusing all other platform features. Most SAPs rely on an external information extraction (IE) system, most of which have been previously developed from the natural language processing community. For example, GATE (Cunningham et al., 2002) has been developed and refined for over eight years, and Amilcare (Handschuh et al., 2002) also has several years of development. IE components typically perform language tasks such as

tokenization, part-of-speech tagging, sentence splitting, and dictionary lookup. Some IE systems provide additional services such as named entity recognition, IE rule induction using machine learning (Handschuh et al., 2002), and finding identity relations between entities in text (coreferencing) (Cunningham et al., 2002).

In this section, we include our review of several SAPs such as AeroDAML and Armadillo. SAPs vary in their architecture, information extraction tools and methods, initial ontology, amount of manual work required to perform annotation, performance, and other supporting features, such as storage management of ontologies, knowledgebases, and annotations. The SAPs were chosen from a literature review of recent semantic annotation journal articles. The idea is to survey the supporting features, annotation methods used, and effectiveness of a set of SAPs that represent each classification from the SAP pattern-based and machine-learning-based classifications.

2.2.2.1 AeroDAML

AeroDAML (Kogut and Holmes, 2001) uses pattern-based approach and is designed to map proper nouns and common relationships to corresponding classes and properties in DARPA Agent Markup Language (DAML) (Greaves, 2004) ontologies. AeroDAML uses AeroText for its information extraction (IE) component. The AeroText Java API is used to access IE parts and map them into RDF triples using an ontology as a guide. The default ontology consists of two parts. The upper level uses the WordNet (Princeton University, 2004) noun synset hierarchy. The lower level uses the knowledgebase provided by AeroText (Kogut and Holmes, 2001). The integrated AeroText system consists of four main components: (1) Knowledge Base (KB) compiler for transforming linguistic data into a run-time knowledge base; (2) Knowledge Base Engine for applying the KB to source documents; (3) an IDE for building and testing KBs, and (4) Common Knowledge Base containing domain independent rules for extracting proper nouns and relations (Kogut and Holmes, 2001). AeroDAML operates in two modes depending on the version used. The Web-based system allows users to enter a URI and have the DAML-based annotation of the page returned using the default ontology. The client-server version allows users to enter a filename and returns the DAML-based annotation of the text returned using a custom ontology.

2.2.2.2 Armadillo

Armadillo (Dingli et al., 2003) uses the Amilcare IE system to perform wrapper induction on Web pages to mine Web sites that have a highly regular structure. Armadillo uses a pattern-based approach to find entities, finding its own initial set of seed patterns, rather than requiring an initial set of seeds, as described in (Brin, 1998). Manual patterns are used for the named-entity recognizer. No manual annotation of corpus documents is required. Once the seeds are found, pattern expansion is then used to discover additional entities. Information redundancy, via queries to Web services such as Google and CiteSeer, is used to verify discovered entities by analyzing query results to confirm or deny the existence of an entity, similar to the way the PANKOW algorithm (Cimiano et al., 2004) operates. The use case implemented in Armadillo is extracting worker details from a university computer science department Web site in order to find personal data, such as name, position, home page, e-mail address, and other contact information. The seed-discovery and expansion finds worker names in

the Web pages. Since many names may be discovered, the Web services are queried to confirm a person actually works in the department. The names are then used to discover home pages, where detailed information about a person can often be found and extracted. Armadillo is also interesting in that it attempts to discover bibliographic citations for each person discovered. The information redundancy approach was also applied to bibliographic entries, but with a lower success rate than discovering and extracting information about people from home pages (Dingli et al., 2003).

2.2.2.3 KIM

The Knowledge and Information Management (KIM) platform (Popov et al., 2003) contains an ontology, knowledgebase, a semantic annotation, indexing and retrieval server, as well as front-ends for interfacing with the server. For ontology and knowledgebase storage it uses the SESAME RDF repository (Broekstra et al., 2002), and for search it uses a modified version of the Lucene (Cutting, 2004) keyword-based search engine. The semantic annotation process relies on a prebuilt lightweight ontology called KIMO as well as an interdomain knowledgebase. KIMO defines a base set of entity classes, relationships, and attribute restrictions. The knowledgebase is populated with 80,000 entities consisting of locations and organizations, gathered from a general news corpus. Named-entities found during the annotation process are matched to their type in the ontology and also to a reference in the knowledgebase. The dual mapping allows the information extraction process to be improved by providing disambiguation clues based on attributes and relations (Popov et al., 2003).

The information extraction component of semantic annotation is performed using components of the GATE (Cunningham et al., 2002) toolkit. GATE provides IE implementations of tokenizers, part-of-speech taggers, gazetters, pattern-matching grammars (JAPE), and coreference resolution (Popov et al., 2003). Some components of GATE have been modified to support the KIM server. For example, pattern-matching grammar rules are based on ontology classes rather than simple types (Popov et al., 2003). Other components of semantic annotation have been custom developed. The gazetteer, for example, performs entity alias lookups using the knowledgebase rather than from an external source.

2.2.2.4 MnM

MnM (Vargas-Vera et al., 2002) provides an environment to manually annotate a training corpus, and then feed the corpus into a wrapper induction system based on the Lazy-NLP (natural language processing) algorithm. The resulting output is a library of induced rules that can be used to extract information from corpus texts (Vargas-Vera et al., 2002). Lazy-NLP systems are based on linguistic information, and rules are generated using sets of conjunctive conditions on adjacent words (Vargas-Vera et al., 2002). The rule induction process generates two types of rules: tagging and correction. A rule is composed of a pattern of conditions on a connected sequence of words, followed by an action performed when the pattern is found to match a part of text (Vargas-Vera et al., 2002). For tagging rules, the action performed is the insertion of a semantic tag into the text. For correction rules, the action performed is the insertion of information that causes semantic tags to shift location, based on training information. The corrective tags are inserted during the training period, when

the training corpus is reannotated using the induced rules. If an induced tagging rule is found to be incorrect in its location, it is corrected using a correction rule rather than replaced.

2.2.2.5 MUSE

MUSE (Maynard, 2003) was designed to perform named entity recognition and coreferencing. It is implemented using the GATE (Cunningham et al., 2002) framework. The IE components, called *processing resources* (PRs), form a processing pipeline used to discover named entities. MUSE executes PRs conditionally based on text attributes. Conditional processing is handled using a Switching Controller, which calls the appropriate PR in the specified order. The use of conditional processing allows MUSE to obtain accuracies similar to machine learning systems (Maynard, 2003). Semantic tagging is accomplished using the Java Annotations Pattern Engine (JAPE) (Cunningham et al., 2000). Rules using the JAPE grammar are constructed to generate annotations. The Semantic Tagger can use tags generated by processing resources run earlier in the pipeline. For example, if the gazetteer recognizes a first name and the part-of-speech tagger recognizes a proper noun, a JAPE rule can use both tags to annotate an entity of type Person (Maynard, 2003). The MUSE system is more sophisticated than a gazetteer because a gazetteer cannot provide an exhaustive list of all potential named-entities, and cannot resolve entity ambiguities (e.g., Washington can be a city or a person) (Maynard, 2003).

2.2.2.6 Ont-O-Mat Using Amilcare

Ont-O-Mat (Handschuh et al., 2002) is an implementation of the S-CREAM (Semiautomatic CREAtion of Metadata) semantic annotation framework. The IE component is based on Amilcare. Amilcare is machine-learning-based and requires a training corpus of manually annotated documents. Amilcare uses the ANNIE ("A Nearly-New IE system") part of the GATE toolkit to perform IE (Handschuh et al., 2002). The result of ANNIE processing is passed to Amilcare, which then induces rules for IE using a variant of the LP^2 algorithm. The wrapper induction process uses linguistic information, and is the same Amilcare wrapper induction process as MnM (Vargas-Vera et al., 2002), generating tagging and correction rules.

2.2.2.7 Ont-O-Mat Using PANKOW

Ont-O-Mat (Handschuh et al., 2002) provides an extensible architecture to replace selected components. The original annotation component is replaced using an implementation of the PANKOW (Pattern-based Annotation through Knowledge On the Web) algorithm (Cimiano et al., 2004). The PANKOW process takes proper nouns from the IE phase and generates hypothesis phrases based on linguistic patterns and the specified ontology. For example, a sports ontology may generate hypothesis phrases from the proper noun "Pete Rose" using patterns such as "Pete Rose *is a* Player" and "Pete Rose *is a* Team," where "Player" and "Team" are ontology concepts. The hypothesis phrases are then presented to the Google Web service. The phrase with the highest query result count is then used to annotate the text with the appropriate concept.

The core principle is called "disambiguation by maximal evidence" (Cimiano et al., 2004). This principle is similar to the approach used by Armadillo (Dingli et al., 2003), which used multiple Web services to find evidence. The World Wide Web is used as the corpus because of its size and the likelihood of finding significant numbers of hypothesis phrase matches. The number of Web pages containing the hypothesis phrase are counted and the count used to indicate the strength of a particular hypothesis phrase. The content of the individual Web pages is ignored and plays no role in determining the validity of a hypothesis phrase. The linguistic patterns used to generate hypothesis phrases have been previously identified in the literature.

2.2.2.8 SemTag

SemTag (Dill et al., 2003) is the semantic annotation component of a comprehensive platform, called Seeker, for performing large-scale annotation of Web pages. SemTag performs annotation in three passes: Spotting, Learning, and Tagging. The Spotting pass examines tokenized words from source documents and finds label matches from the taxonomy. If a label match is found, a window of 10 words to either side of the source document match is kept. In the Learning pass, a sample of the corpus is examined to find the corpus-wide distribution of terms at each node of the taxonomy. The Tagging pass is then executed, scanning all of the windows from the Spotting pass and disambiguating the matches. Once a match is confirmed, the URL, text reference, and other metadata are stored. SemTag/Seeker is an extensible system, so new annotation implementations can replace the existing Taxonomy-based Disambiguation algorithm (TBD). The taxonomy used by SemTag is TAP. TAP is shallow and covers a range of lexical and taxonomic information about popular items such as music, movies, authors, sports, health, and so forth (Dill et al., 2003). The annotations generated by SemTag are stored separate from the source document. The intent of the SemTag/Seeker design is to provide a public repository with an API that will allow agents to retrieve the Web page from its source and then request the annotations separately from a Semantic Label Bureau (Dill et al., 2003).

2.2.2.9 Semantic Annotation Platform Summary

In summary, SAPs can be distinguished primarily by their annotation method, since that component has the largest impact on the effectiveness of semantic annotation. The two primary annotation approaches are pattern-based and machine-learning-based (Reeve and Han, 2005). SAPs have additional characteristics, such as extensible architectures for adapting to evolving technology. For example, the Ont-O-Mat system, initially developed using Amilcare for wrapper induction (Handschuh et al., 2002), had later research where a maximal-evidence, pattern-discovery algorithm replaced the semantic annotation component (Cimiano et al., 2004). In terms of required manual effort, rule-based systems generally require manual rule generation, whereas machine-learning methods require preannotated corpora for training. SAPs also provide features for ontology and knowledgebase management, access APIs, storage management, and user-interfaces for editing ontologies, knowledgebases, and definitions for annotating sources. Each surveyed platform provides different levels of support for each of these features. The SAPs presented are representative samples of the pattern-based and machine-learning-based classifications. The continuing

evolution of SAPs to provide improved annotation accuracy, reduced manual effort, and new features is vital to the realization of the Semantic Web.

2.3 Information Visualization

Information visualization is a rapidly advancing field of study. The number of review and survey articles on information visualization is steadily increasing (Hollan et al., 1997; Card, 1996; Herman et al., 2000; Mukherjea, 1999; Hearst, 1999). There are currently several books on information visualization (notably Card et al., 1999; Chen, 1999; Ware, 2000; and Spence, 2000). A related book is on algorithms for graph visualization (Battista et al., 1999). Palgrave, Macmillan's global academic publishing, is launching a new, peer-reviewed international journal, *Information Visualization,* in 2002.

The goal of information visualization is to reveal patterns, trends, and other new insights into a phenomenon. Information visualization focuses on abstract information. A major challenge in information visualization is to transform nonspatial and nonnumerical information into effective visual form. This distinct orientation is captured by the following definition (Card et al., 1999): *Information visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition.*

Information visualization faces some major fundamental challenges: one is to come up with a design metaphor that will accommodate such transformations from nonspatial, nonnumerical to something visible and meaningful; the other is to find ways to ensure that information visualization functions designed based on a particular metaphor indeed work.

In general, information visualization is at a crossroad, waiting for the "killer applications." Creating, or even selecting an appropriate visual metaphor is not simple. A taxonomy that can help one to match a problem at hand with appropriate design metaphors remains on many people's wish list. Information retrieval has brought countless inspirations and challenges to the field of information visualization. It has played considerable roles in shaping the field. Our quest aims to go beyond information retrieval. Our focus is on the growth of scientific knowledge and what are the key problems to solve and what are the central tasks to support. Instead of focusing on locating specific items in scientific literature, we turn to higher levels of granularity—scientific paradigms and their movements in scientific frontiers.

Information visualization is particularly in need of a generic theory that can help designers and analysts to assess information visualization designs. *Semiotic morphism* is a computational theory on how to preserve the meaning of signs in translating symbol systems. Recently, Joseph Goguen of the University of California at San Diego demonstrated the potential of semiotic morphisms in identifying defects of information visualization (Goguen, 2000).

According to Goguen, the fundamental issues in information visualization can be understood in terms of *representation*: a visualization is a representation of some aspects of the underlying information; and the main questions are *what* to represent, and *how* to represent it. Information visualization needs a theory of representation that can take account not just of the capabilities of current display technology, but also of the structure of complex information, such as scientific data, the capabilities

and limitations of human perception and cognition, and the social context of work.

However, classical semiotics, which studies the meaningful use of signs, is not good enough. Because it has unfortunately not developed in a sufficiently rigorous way for our needs, it has not explicitly addressed representation; also, its approach to meaning has been naive in some crucial respects, especially in neglecting (though not entirely ignoring) the social basis and context of meaning. This is why that semiotics has mainly been used in the humanities, where scholars can compensate for these weaknesses, rather than in engineering design, where descriptions need to be much more explicit. Another deficiency of classical semiotics is its inability to address dynamic signs and their representations, as is necessary for displays that involve change, instead of presenting a fixed static structure (e.g., for standard interactive features like buttons and fill-in forms, as well as for more complex situations like animations and virtual worlds).

Goguen's initial applications of semiotic morphisms on information visualization have led to several principles that may be useful in assessing a range of information visualization design (Goguen, 2000). He suggested three rules of thumb:

1. Measure quality by what is preserved and how it is preserved.
2. It is more important to preserve structure than content when a trade-off is forced.
3. There is a need to take account of social aspects in user interface design.

The semiotics morphisms methodology is not just algebraic but also social. More in-depth studies are needed to verify the power of this approach.

2.3.1 Tracking Knowledge and Technology Trends

Tracking knowledge and technology transfer has been a challenging but potentially rewarding task if it can be accomplished on the new generation of the World Wide Web—the Semantic Web. The aim is to transform billions of machine-readable and human-friendly documents on the first generation of the Web to machine-understandable forms in the second generation of the Web.

Knowledge visualization is not merely knowledge representation gone graphical. With knowledge visualization, the goal is to help us to track down a cognitive and intellectual movement from the available resources on the Web.

2.3.2 Citation Analysis

In information science, the term *specialty* refers to the perceived grouping of scientists who are specialized in the same or closely related topics of research. Theories of how specialties evolve and change started to emerge in the 1970s (Small and Griffith, 1974). Researchers began to focus on the structure of scientific literatures in order to identify and visualize specialties, although they did not use the term "visualization" at that time.

Today's most widely used citation index databases such as SCI and SSCI were conceived in the 1950s, especially in Garfield's pioneering paper published in *Science* (Garfield, 1955). In the 1960s, several pioneering science mapping studies began to emerge. For example, Garfield, Sher, and Torpie created the historical map of research in DNA (Garfield et al., 1964). Sher and Garfield demonstrated the power of citation

analysis in their study of Nobel Prize winners' citation profiles (Sher and Garfield, 1966).

The 1980s saw the beginning of what turned out to be a second fruitful line of development in the use of citation to map science—author co-citation analysis (ACA). Howard White and Belver Griffith introduced ACA in 1981 as a way to map intellectual structures (White and Griffith, 1981). The unit of analysis in ACA is authors and their intellectual relationships as reflected through scientific literatures. The author-centered perspective of ACA led to a new approach to the discovery of knowledge structures in parallel to approaches used by document-centered co-citation analysis (DCA).

The author co-citation map produced by White and Griffith (1981) depicted information science over a 5-year span (1972–1979). In 1998, 17 years later, White and McCain (1998) generated a new map of information science based on a considerably expanded 23-year span (1972–1995).

An author co-citation network offers a useful alternative starting point for co-citation analysis, especially when we encounter a complex document co-citation network, and vice versa. Katherine McCain (1990) gave a comprehensive technical review of mapping authors in intellectual spaces. ACA reached a significant turning point in 1998 when White and McCain (White and McCain, 1998) applied ACA to information science in their thorough study of the field. Since then ACA has flourished and has been adopted by researchers across a number of disciplines beyond the field of citation analysis itself. Their paper won the best JASIS paper award. With both ACA and DCA at our hands, we begin to find ourselves in a position to compare and contrast messages conveyed through different co-citation networks of the same topic. Robert Braam, Henk Moed, and Anthony van Raan investigated whether co-citation analysis indeed provided a useful tool for mapping subject-matter specialties of scientific research (Braam et al., 1991a; 1991b).

2.3.3 Patent Citation Analysis

Patent analysis has a long history in information science, but recently there is a surge of interest from the commercial sector. Numerous newly formed companies are specifically aiming at the patent analysis market. Apart from historical driving forces such as monitoring knowledge and technology transfer and staying in competition, the rising commercial interest in patent analysis is partly due to the publicly accessible patent databases, notably the huge amount of patent applications and grants from the United States Patent and Trademark Office (USPTO). The public can search patents and trademarks at USPTO's Web site http://www.uspto.gov/ and download bibliographic data from ftp://ftp.uspto.gov/pub/patdata/.

The availability of the abundant patent data, the increasingly widespread awareness of information visualization, and the maturity of search engines on the Web are among the most influential factors behind the emerging trend of patent analysis. Many patent search interfaces allow users to search by specific sections in patent databases, for example by claims. Statistical analysis and intuitive visualization functions are by far the most commonly seen selling points from a salesperson's patent analysis portfolio. The term visualization becomes so fashionable now in the patent analysis industry that from time to time we come across visualization software tools that turn out to be little more than standard displays of statistics. Table 2.1 lists some of the prominent vendors and service providers in patent analysis.

Table 2.1 A recent surge of patent analysis

Software	Functionality	Applications	Homepage
Aurigin	Patent searching; analysis; visualization	Patent analysis; knowledge management	http://www.aurigin.com/
Delphion	Patent searching	Intellectual property management	http://www.delphion.com/
ImageSpace	Analysis and visualization	Confocal imaging	http://www.mdyn.com/
Mapit	Patent mining		http://www.mnis.net/
OmniViz Pro	Information visualization; data mining	Life sciences; chemical sciences	http://64.77.30.212/default.htm
PatentLab II	Analysis	Extracting intelligence from patent data	http://www.wisdomain.com/
SemioMap	Text analysis	Multilayered concept maps	http://www1.semio.com/

Source: Chen (2003), *Mapping Scientific Frontiers,* Table 5.3.

A particularly interesting example is from Sandia National Laboratory. Kevin Boyack and his colleagues (2000) used their landscape-like visualization tool VxInsight to analyze the patent bibliographic files from USPTO in order to answer a number of questions. For example, where are competitors placing their efforts? Who is citing our patents, and what types of things have they developed? Are there emerging competitors or collaborators working in related areas? The analysis was based on 15,782 patents retrieved from a specific primary classification class from the U.S. Patent database. The primary classification class is class 360 on *Dynamic Magnetic Information Storage or Retrieval.* A similarity measure was calculated using the direct and co-citation link types of Small (1997). Direct citations were given a weighting five times that of each co-citation link.

2.4 A Harmonious Relationship?

2.4.1 Beyond Information Retrieval

Science mapping reveals structures hidden in scientific literature. The definition of association determines the nature of the structure to be extracted, to be visualized, and to be eventually interpreted. Co-word analysis (Callon et al., 1986) and co-citation analysis (Small, 1973) are among the most fundamental techniques for science mapping. Small (Small, 1988) described the two as follows: "If co-word links are viewed as translations between problems, co-citation links have been viewed as statements relating concepts." They are the technical foundations of the contemporary quantitative studies of science. Each offers a unique perspective on the structure of scientific frontiers. Researchers have found that a combination of co-word and co-citation analysis could lead to a clearer picture of the cognitive content of publications (Braam et al., 1991a; 1991b).

Among the most seminal works in information science, the contribution of Derek de Solla Price (1922–1983) is particularly worth noting, namely his *Networks of Scientific Papers* (Price, 1965), *Little Science, Big Science* (Price, 1963), and *Science*

Since Babylon. In *Little Science, Big Science*, Price raised the profound questions: Why should we not turn the tools of science on science itself? Why not measure and generalize, make hypotheses, and derive conclusions? He used the metaphor of studying the behavior of gas in thermodynamics to analogue the science of science. Thermodynamics studies the behavior of gas under various conditions of temperature and pressure, but the focus is not on the trajectory of a specific molecule. Instead, one considers the phenomenon as a whole. Price suggested that we should study science in a similar way: the volume of science, the trajectory of "molecules" in science, the way in which these "molecules" interact with each other, and the political and social properties of this "gas."

Scientists in general and information scientists in particular have been influenced by Thomas Kuhn's structure of scientific revolutions (Kuhn, 1962), Paul Targard's conceptual revolutions (Thagard, 1992), and Diana Crane's invisible colleges (Crane, 1972). The notion of tracking scientific paradigms is originated in this influence. Two fruitful strands of efforts are particularly worth noting here. One is the work of Eugene Garfield and Henry Small at the Institute for Scientific Information (ISI) in mapping science through citation analysis. The other is the work of Michel Callon and his colleagues in tracking changes in scientific literature using the famous co-word analysis. In fact, their co-word analysis is designated for a much wider scope, *scientific inscriptions*, which includes technical reports, lecture notes, grant proposals, and many others as well as publications in scholarly journals and conference proceedings. More detailed analysis of these examples can be found in Chen (2002). The new trend today focuses on the dynamics of scientific frontiers more specifically. What are the central issues in a prolonged scientific debate? What constitutes a context in which a prevailing theory evolves? How can we visualize the process of a paradigm shift? Where are the rises and falls of competing paradigms in the context of scientific frontiers? What are the most appropriate ways to visualize scientific frontiers? In the context of the Semantic Web, what can the Semantic Web do to help us address these questions?

One of the most influential works in the twentieth century is the theory of the structure of scientific revolutions by Thomas Kuhn (1922–1996) (Kuhn, 1962). Before Kuhn's structure, philosophy of science had been dominated by what is known as the logical empirical approach. The logical empiricism uses modern formal logic to investigate how scientific knowledge could be connected to sense experience. It emphasizes the logical structure of science rather than its psychological and historical development.

Kuhn criticized that the logical empiricism cannot adequately explain the history of science. He claimed that the growth of scientific knowledge is characterized by revolutionary changes in scientific theories. According to Kuhn, most of the time scientists are engaged in what is known as normal science. A period of normal science is typically marked by the dominance of an established framework. The majority of scientists would work on specific hypotheses within such frameworks, or *paradigms.* The foundations of such paradigms largely remain unchallenged until new discoveries begin to cast doubts over fundamental issues. As anomalies build up, attention is suddenly turned to an examination of previously taken-for-granted basic assumptions—science falls into a period of crises. To resolve such crises, radically new theories with greater explanatory power are introduced. New theories replace the ones in trouble in a revolutionary manner. Science regains another period of normal science. Scientific revolutions, as Kuhn claimed, are an integral part of science and science progresses through such revolutionary changes.

Diana Crane is concerned with scientific ideas and how they grow into a body of knowledge in her *Invisible Colleges: Diffusion of Knowledge in Scientific Communities* (Crane, 1972). She suggests that it is the "invisible college" that is responsible for the growth of scientific knowledge. An invisible college constitutes of a small group of highly productive scientists. They share the same field of study, communicate with one another, and thus monitor the rapidly changing structure of knowledge in their field. Howard White and Katherine McCain, of Drexel University, identified the domain structure of information science by using author co-citation analysis (White and McCain, 1998).

Steve Steinberg addressed several questions regarding the use of a quantitative approach to identify paradigm shifts in the real world (Steinberg, 1994). He examined the history of Reduced Instruction Set Computing (RISC). The idea behind RISC was that a processor with only a minimal set of simple instructions could outperform a processor that included instructions for complex high-level tasks. In part, RISC marked a clear shift in computer architecture and had reached some degree of consensus. Steinberg searched for quantitative techniques that could help his investigation. He adapted the co-word analysis technique to produce a map of the field, a visualization of the mechanisms, and a battle chart of the debate. He collected all abstracts with the keyword RISC for the years 1980–1993 from the INSPEC database, filtered out the 200 most common English words, and ranked the remaining words by frequency. The top 300 most frequently occurring words were given to three RISC experts to choose those words central to the field. Finally, words chosen by the experts were aggregated by synonyms into 45 keyword clusters. The inclusion index was used to construct a similarity matrix. This matrix was mapped by MDS with ALSCAL. The font size of a keyword was proportional to the word's frequency. Straight lines in the co-word map denoted strongly linked keywords.

The first papers to explicitly examine and define RISC appeared within the period of 1980–1985. The design philosophy of RISC was so opposed to the traditional computing architecture paradigm that every paper in this period was written to defend and justify RISC. The map shows two main clusters. One is on the left, surrounding keywords such as register, memory, simple, and pipeline. These are the architectural terms that uniquely define RISC. The other cluster is on the right, centered on keywords such as language and CISC. These are the words that identify the debate between the RISC and CISC camps. Language is the most frequent keyword on the map. According to Steinberg, the term *language* most clearly captures the key to the debate between RISC and CISC. While CISC proponents believed that a processor's instruction set should closely correspond to high-level languages such as FORTRAN and COBOL, RISC proponents argue that simple instructions were better than high-level instructions. This debate is shown in the co-word map with the connections between language, CISC, compiler, and programming. For a detailed description and a reproduction of the co-word maps, readers are referred to Chen (2002).

Chen, Kuljis, and Paul (Chen et al., 2001) explored the way to overcome some of the difficulties in visualizing underrepresented latent domain knowledge through two case studies: one on knowledge representation and the other on a controversial theory on the causes of mad cow disease. Chen, Cribbin, Macredie, and Morar (Chen et al., 2002) reported two more case studies of tracking competing paradigms using information visualization. One case study was about the active galactic nuclei paradigm in astronomy and astrophysics and the other was about mad cow disease in medical science.

2.4.2 Yin and Yang

Hjorland has been promoting domain analysis in information science (Hjorland and Albrechtsen, 1995; Hjorland, 1997). The unit of domain analysis is a specialty, a discipline, or a subject matter. In contrast to existing approaches to domain analysis, Hjorland emphasized the essential role of a social perspective instead of the more conventional psychological perspective.

Hjorland called his approach an *activity-theoretical* approach. The traditional approaches focus on individuals as a single user of information in terms of their cognitive structures and strategies. The activity-theoretical approach, on the other hand, emphasizes a holistic view of information retrieval issues in a much broader context so that the needs of a user should be always interpreted in the context of the discipline (see Table 2.2). The relevance of a retrieved item is linked directly to the substance

Table 2.2 Differences between cognitivism and the domain-specific viewpoint (Hjorland and Albrechtsen, 1995)

Cognitivism	The domain-specific view
Priority is given to the understanding of isolated user needs and intrapsychological analysis. Intermediating between producers and users emphasizes psychological understanding.	Priority is given to the understanding of user needs from a social perspective and the functions of information systems in trades or disciplines.
Focus on the single user. Typically looks at the disciplinary context as a part of the cognitive structure of an individual if at all.	Focus on either one knowledge domain or the comparative study of different knowledge domains. Looks at the single user in the context of the discipline.
Mainly inspired by artificial intelligence and cognitive psychology.	Mainly inspired by knowledge about the information structures in domains, by the sociology of knowledge and the theory of knowledge.
The psychological theory emphasizes the role of cognitive strategies in performance.	The psychological theory emphasizes the interaction among aptitudes, strategies, and knowledge in cognitive performance.
Central concepts are individual knowledge structures, individual information processing, short- and long-term memory, categorical versus situational classification.	Central concepts are scientific and professional communication, documents (including bibliographies), disciplines, subjects, information structures, paradigms, etc.
Methodology characterized by an individualistic approach.	Methodology characterized by a collectivistic approach.
Methodological individualism has some connection to a general individualistic view, but the difference between cognitive and the domain-specific view is not a different political perception of the role of information systems, but a different theoretical and methodological approach to the study and optimization of information systems.	Methodological collectivism has some connection to a general collectivistic view, but the difference between cognitivism and the domain-specific view is not a different political perception of the role of information systems, but a different theoretical and methodological approach to the study and optimization of information systems.
Best examples of applications: user interfaces (the outer side of information systems).	Best examples of applications: subject-representation/classification (the inner side of information systems).
Implicit theory of knowledge: mainly rationalistic/positivistic, tendencies toward hermeneutics.	Theory of knowledge: scientific realism/forms of social constructivism with tendencies toward hermeneutics.
Implicit ontological position: subjective idealism.	Ontological position: realism.

of a subject matter. This view is in line with the goal of domain visualization, that is, to provide a meaningful context in which scientists can explore the body of knowledge as a whole, as opposed to dealing with fragmented pieces of knowledge. Domain visualization underlines the development of a research theme in scientific literature.

The major differences between the Semantic Web and information visualization are similar. The Semantic Web emphasizes formal, machine-understandable, and sometimes cognitivism-like approaches. It focuses on the form and even the meaning is achieved through rigorously defined forms. In contrast, information visualization emphasizes the semantics and the meaning that can be conveyed by visual-spatial models to the users. The form for information visualization is secondary. The following is an illustrative example. The question is: To what extent is the Semantic Web suitable to handle semantics associated with high-level, volatile information structures?

2.4.3 An Illustrative Example

In the following example, we illustrate the challenges one has to face in order to benefit from the best of the both worlds. This is a typical citation analysis enhanced by information visualization techniques. The goal of this study is to identify the overall structure and the central issues of patent citation analysis. A description of the procedure for general audiences can be found in Chen and Paul (2001).

In order to generate a brief snapshot of the field of patent citation analysis, we searched the Web of Science on January 15, 2002 across the period of 1990 and 2001. The Web of Science allows the user to specify the appearance of keywords in a number of fields. We would like to find all the articles that not only contain the keyword *patent* and *citation*, but they must appear within the same sentence. The Web of Science allows us to specify our requirement with a search query "patent same citation" in topic, which covers the fields of title, abstract, and keyword (Figure 2.2).

A total of 36 records were found by this query (Figure 2.3). The hit rate is relatively small, considering some queries may result in hundreds and thousands of hits. The lower hit rate like this reflects the size of the patent citation analysis literature. Nevertheless, these 36 records allow us to glimpse the history and the state of the art of patent citation analysis. Each of these 32 records contains a list of references. It is this set of references that form the basis of a snapshot of patent citation analysis (Figure 2.4).

The 36 citing documents cited 97 unique articles at least once. A snapshot of the intellectual structure was generated based on the 97-by-97 document co-citation matrix. Figure 2.5 shows the overview of the snapshot and four closeup views of individual components. Articles that have been cited more than three times are labeled in the scene. The three-dimensional landscape scene was rendered in Virtual Reality Modeling Language (VRML). The base structure is defined by Pathfinder network scaling (Schvaneveldt, 1990; Chen, 1998a; 1998b). The node-and-link structure denotes the connectivity between cited documents on patent citation analysis. The color of a node indicates the specialty membership of the corresponding document cited. The color is determined by the results of Principle Component Analysis (PCA); in particular, the factor loading coefficients of the three most predominant factors are mapped to the red, green, and blue strengths of the final color. A cluster or a subgraph in red essentially corresponds to the largest predominant factor, which normally represents the strongest specialty in the field. The second strongest specialty would be in green and the third in blue. A well-balanced specialty would have a color close to white, which is the perfect balanced color of equal amount of red, green, and blue. In Figure 2.4,

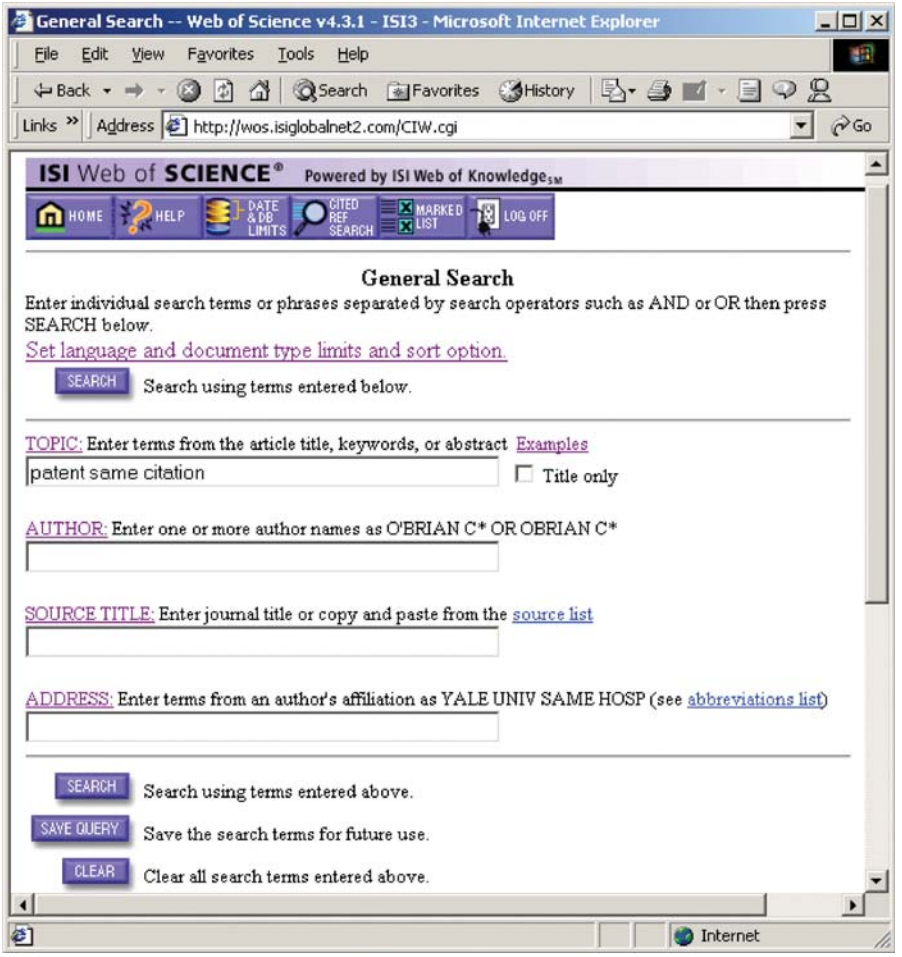


Figure 2.2 The initial search query to the Web of Science to find articles in which both patent and citation appear within the same sentence in title, abstract, or keyword list. Date of search: January 15, 2002 Web of Science (1990–2001).

there is a purple cluster located toward the lower-left corner of the view (cluster 1); a green cluster pointing to the lower-right corner (cluster 2); a small blue cluster in the remote distance (cluster 3); and a long-shaped cluster connecting cluster 3 to the center of the network. Each cluster is also shown in insets at the corners of the figure.

The height of a vertical bar stemmed from an article is proportional to the number of times the article has been cited by the 36 citing articles as a whole. We have demonstrated the power of animated visualizations of citation landscapes in a number of case studies based on ISI's citation index databases (Chen et al., 2001; 2002). We are extending this streamlined procedure from co-citation analysis of scientific literature to patent citation analysis based on patent databases (Chen and Hicks, 2004). On the one hand, we will distinguish citation networks reflected through patent applications and those from granted patents. On the other hand, we are more interested in the

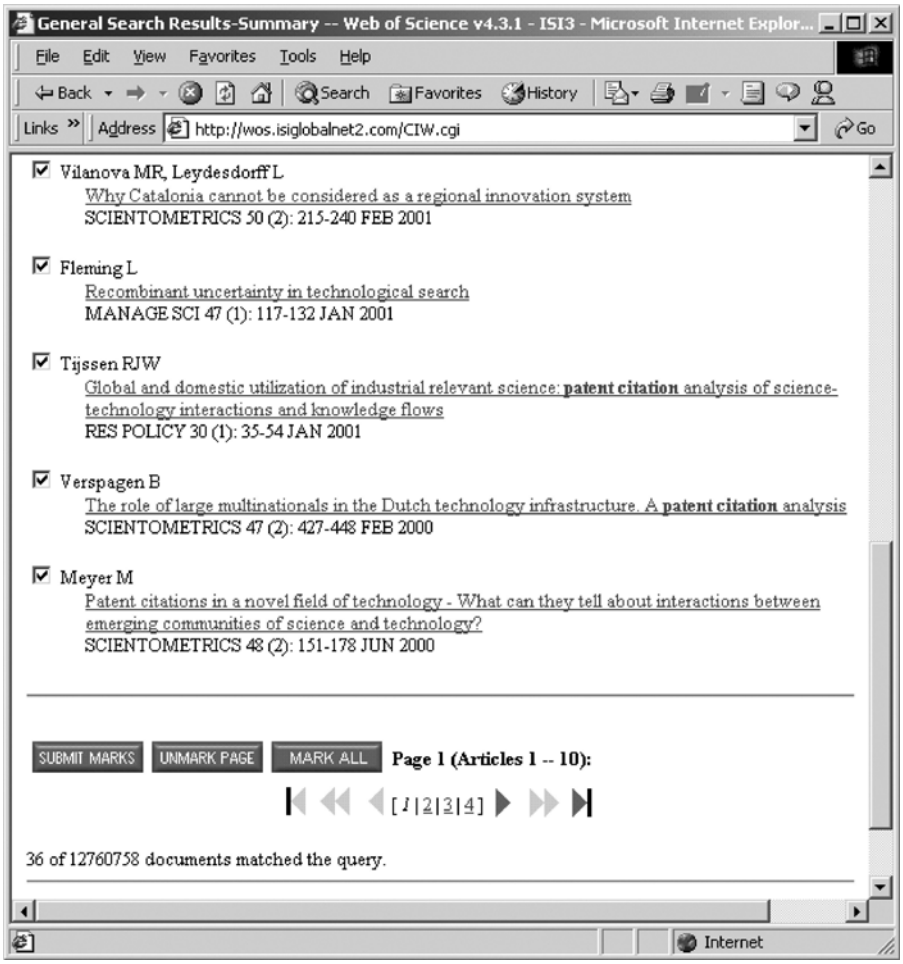


Figure 2.3 A total of 36 hits were returned by Web of Science on patent citation analysis.

interrelationship and time-series relationships involving both scientific literature and patent databases.

Table 2.3 lists the six factors extracted by Principle Component Analysis. Each factor corresponds to a nonexclusive specialty, which means a document could belong to more than one specialties. These six factors explained a total of 94% of variance.

To date, the nature of a specialty cannot be algorithmically identified. This is in part due to the incompleteness of the source data. For example, it contains no further information of a cited reference, no title, and no abstract. This is where the Semantic Web could play a substantial role in piecing together a comprehensive description of a given reference from a number of different resources. Currently, NEC's ResearchIndex has done a great job by scanning scientific publications on the Web and built an increasingly comprehensive, online accessible bibliographic database. Table 2.4 shows a ranked list of documents in the first specialty. For example, Narin's 1997 article published in *Research Policy* has the highest factor loading coefficient 0.932, which

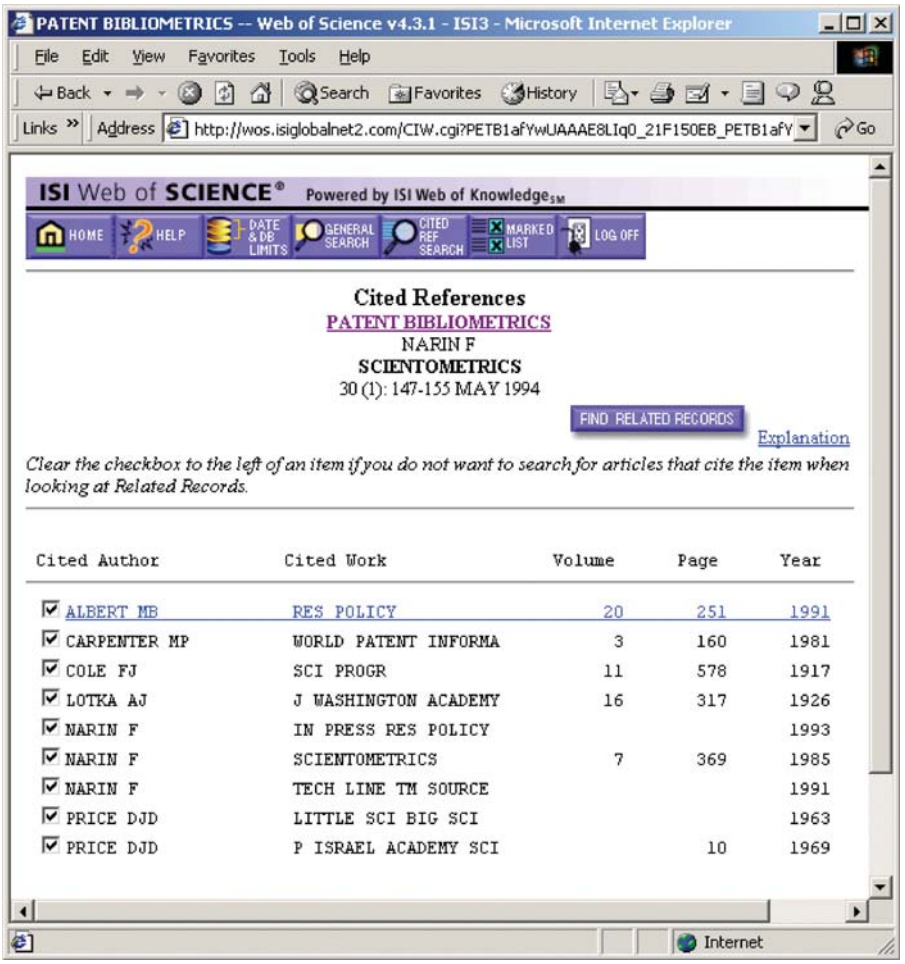


Figure 2.4 Each of the 36 records cites a list of references. The creation of a snapshot is based on interrelationships among all the references cited collectively by the 36 publications.

Table 2.3 Factors extracted by Principle Component Analysis. Each factor corresponds to a nonexclusive specialty

	Extraction sums of squared loadings			Rotation sums of squared loadings		
Component	Eigenvalue	% of variance	Cumulative %	Eigenvalue	% of variance	Cumulative %
1	47.139	48.597	48.597	46.435	47.872	47.872
2	24.338	25.091	73.688	23.171	23.887	71.759
3	8.403	8.663	82.351	7.589	7.824	79.582
4	6.584	6.787	89.138	7.550	7.784	87.366
5	3.120	3.216	92.354	3.569	3.680	91.046
6	1.451	1.496	93.850	2.720	2.804	93.850

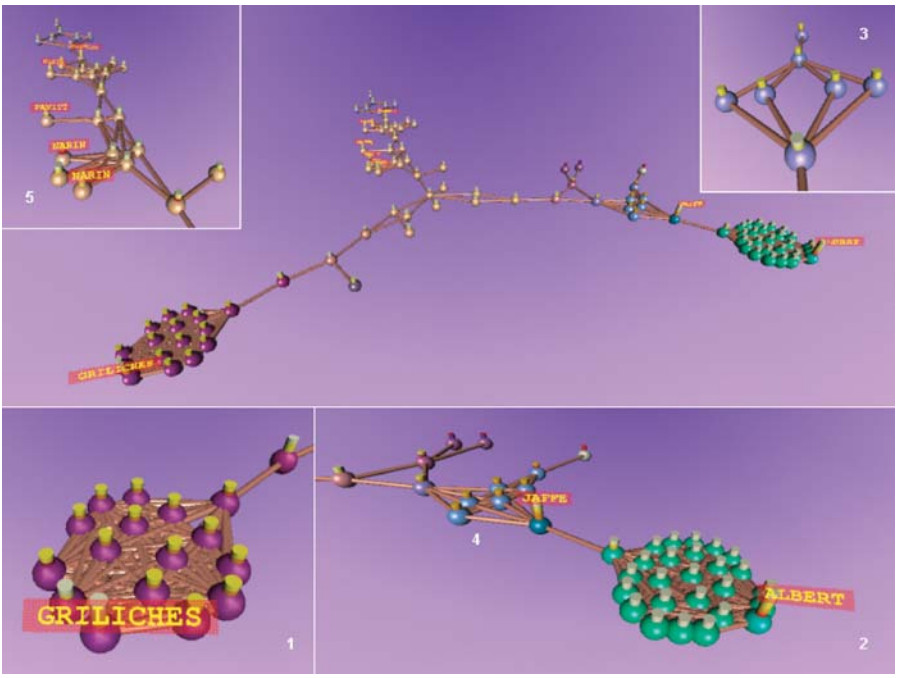


Figure 2.5 A snapshot of the patent citation analysis field based on the co-citation visualization procedure explained in Chen and Paul (2001).

implies Narin's article is the most representative of this specialty. In the Semantic Web, a bibliographic agent could go out and fetch further details of this article, such as its title and abstract, behind the scene. Furthermore, an automated summarization agent could produce a short list of keywords that can be used to characterize the nature of the specialty.

Table 2.4 A ranked list of documents in the first specialty. Documents in this specialty should be colored in red

Factor loading	Cited references
0.932	NARIN F, 1997, RES POLICY, V26, P317
0.906	NARIN F, 1995, RES EVALUAT, V5, P183
0.905	MARTIN B, 1996, RELATIONSHIP PUBLICL
0.905	NOYONS ECM, 1994, RES POLICY, V23, P443
0.905	VANVIANEN BG, 1990, RES POLICY, V19, P61
0.904	NARIN F, 1985, SCIENTOMETRICS, V7, P369
0.899	CARPENTER MP, 1983, WORLD PATENT INFORMA, V5, P180
0.899	SCHMOCH U, 1993, SCIENTOMETRICS, V26, P193
0.889	GIBBONS M, 1994, NEW PRODUCTION KNOWL
0.884	*IITRI, 1968, TECHN RETR CRIT EV S
0.879	PAVITT K, 1984, RES POLICY, V13, P343
0.878	NARIN F, 1994, SCIENTOMETRICS, V30, P147
0.873	BRAUN T, 1997, SCIENTOMETRICS, V38, P321

Table 2.5 A ranked list of documents in the second largest specialty. Documents in this specialty should be essentially colored in green

Factor loading	Cited references
0.507	PRICE DJD, 1963, LITTLE SCI BIG SCI
0.408	ABERNATHY WJ, 1978, TECHNOLOGY REV JUN, P40
0.408	ALLEN T, 1977, MANAGING FLOW TECHNO
0.408	BASALLA G, 1988, EVOLUTION TECHNOLOGY
0.408	CAMERON AC, 1986, J APPLIED ECONOMETRI, V1, P29
0.408	AVETTI G, 2000, ADMIN SCI QUART, V45, P113
0.408	GILFILLAN S, 1935, INVENTING SHIP
0.408	HALL B, 2000, NBER WORKING PAPER, V7741
0.408	HARGADON A, 1997, ADMIN SCI QUART, V42, P716
0.408	HAUSMAN J, 1984, ECONOMETRICA, V52, P909
0.408	HENDERSON RM, 1990, ADMIN SCI QUART, V35, P9
0.408	KING G, 1989, INT STUD QUART, V33, P123
0.408	LEVIN RC, 1987, BROOKINGS PAPERS EC, V3, P783
0.408	MARCH J, 1958, ORGANIZATIONS
0.408	MARCH JG, 1991, ORG SCI, V2, P71
0.408	MCCLUSKEY E, 1986, LOGIC DESIGN PRINCIP
0.408	NELSON R, 1982, EVOLUTIONARY THEORY
0.408	RIVKIN JW, 2000, MANAGE SCI, V46, P824
0.408	SCHUMPETER J, 1939, BUSINESS CYCLES
0.408	SIMON H, 1996, SCI ARTIFICIAL
0.408	TUSHMAN ML, 1986, ADMIN SCI QUART, V31, P439
0.408	ULRICH K, 1995, RES POLICY, V24, P419
0.408	WEITZMAN M, 1996, P AM ECONOM ASS MAY, P207
0.399	ROSENBERG N, 1982, INSIDE BLACK BOX TEC
0.390	TRAJTENBERG M, 1990, RAND J ECON, V21, P172
0.380	ALBERT MB, 1991, RES POLICY, V20, P251
0.377	NARIN F, 1985, SCIENTOMETRICS, V7, P369
0.366	NARIN F, 1994, SCIENTOMETRICS, V30, P147
0.365	CAMPBELL B, 1989, EXPERT EVIDENCE INTE, P210
0.365	COLLINS P, 1988, RES POLICY, V17, P65
0.365	MALSCH I, 1997, NANOTECHNOLOGY EUROP
0.365	MEYERKRAHMER F, 1997, CHEM INFORMATION TEC
0.365	RIP A, 1986, MAPPING DYNAMICS SCI, P84
0.365	VANDENBELT H, 1989, EXPERT EVIDENCE INTE
0.363	MEYER M, 1998, SCIENTOMETRICS, V42, P195
0.360	*IITRI, 1968, TECHN RETR CRIT EV S

Table 2.5 lists the predominant documents in the second specialty. Price's *Little Science, Big Science* tops the list with a factor loading of 0.507. Sometimes the significance of a document like *Little Science, Big Science* can never be fully appreciated without substantial knowledge of a domain. Documents in the second specialty should be found in green in the scene.

Table 2.6 shows a ranked list of documents from the third largest specialty. Documents in this specialty should be in blue. Henry Small's three articles are included in the top 10 of this specialty. Small is well known for his work in citation analysis using ISI's data; thus his three articles strongly indicate the nature of this specialty.

Tables 2.7 through 2.9 show documents in the 4th, 5th, and 6th specialty, respectively.

A number of documents appeared in more than one specialty. The more generic a document, the more likely it will appear in different specialties. Table 2.10 lists documents that appeared in three or more specialties along with the three articles

Table 2.6 A ranked list of documents in the third largest specialty (in blue)

Factor loading	Cited references
0.824	NARIN F, 1988, HDB QUANTITATIVE STU
0.772	TRAJTENBERG M, 1990, RAND J EC, V21
0.761	FRANKLIN JJ, 1988, HDB QUANTITATIVE STU, P325
0.761	MOGEE ME, 1998, EXPERT OPIN THER PAT, V8, P213
0.761	SMALL H, 1985, SCIENTOMETRICS, V7, P391
0.761	SMALL H, 1974, SCI STUD, V4, P17
0.752	SMALL H, 1973, J AM SOC INFORM SCI, V24, P265
0.614	CARPENTER MP, 1981, WORLD PATENT INFORMA, V3, P160
0.602	*US DEP COMM, 1992, TECHN PROF REP SEM
0.602	BRAUN E, 1982, REVOLUTION MINIATURE
0.602	NARIN F, 1987, RES POLICY, V16, P143
0.602	ROGERS EM, 1984, SILICON VALLEY FEVER
0.602	SAXENIAN AL, 1991, RES POLICY, V20, P423
0.572	KRUGMAN P, 1991, GEOGRAPHY TRADE
0.333	CAMPBELL RS, 1979, TECHNOLOGY INDICATOR
0.258	PRICE DJD, 1963, LITTLE SCI BIG SCI
0.124	ALBERT MB, 1991, RES POLICY, V20, P251

of Henry Small. In particular, Derek Price's *Little Science, Big Science* appeared in specialties 2, 3, 4, and 6.

This example illustrates that we can generate a snapshot of a knowledge domain based on co-citation patterns. We are currently undertaking research in order to extend and unify citation analysis of scientific literature and patent citation analysis. There are at least two potentially fruitful routes. First, by visualizing the interconnections between scientific literature and patent offices' databases, one can provide a more detailed understanding of the dynamics of knowledge and technology transfer. Science and technology policy decision makers and researchers would be likely to benefit from such knowledge. Researchers at CHI Research Inc. have been using citations to scientific literature in patent front-page references as an indicator of the extent to which a patent is influenced by basic research. Trends in scientific literature are therefore

Table 2.7 A ranked list of documents in the 4th specialty. The color of a document in this specialty is determined by its factor loadings in the 1st, 2nd, and 3rd specialties

Factor loading	Cited references
0.714	KRUGMAN P, 1991, GEOGRAPHY TRADE
0.672	GROSSMAN GM, 1991, INNOVATION GROWTH GL
0.631	*US DEP COMM, 1992, TECHN PROF REP SEM
0.631	BRAUN E, 1982, REVOLUTION MINIATURE
0.631	NARIN F, 1987, RES POLICY, V16, P143
0.631	ROGERS EM, 1984, SILICON VALLEY FEVER
0.631	SAXENIAN AL, 1991, RES POLICY, V20, P423
0.622	CARPENTER MP, 1981, WORLD PATENT INFORMA, V3, P160
0.436	PRICE DJD, 1963, LITTLE SCI BIG SCI
0.407	NARIN F, 1988, HDB QUANTITATIVE STU, P465
0.379	ARTHUR WB, 1989, ECON J, V99, P116
0.310	JAFFE AB, 1986, AM ECON REV, V76, P984
0.276	PAVITT K, 1982, EMERGING TECHNOLOGIE
0.251	JAFFE AB, 1993, Q J ECON, V108, P577

Table 2.8 A ranked list of documents in the 5th specialty

Factor loading	Cited references
0.391	PRICE DJD, 1965, TECHNOL CULT, V6, P553
0.293	CAMPBELL B, 1989, EXPERT EVIDENCE INTE, P210
0.293	MALSCH I, 1997, NANOTECHNOLOGY EUROP
0.293	MEYERKRAHMER F, 1997, CHEM INFORMATION TEC
0.293	RIP A, 1986, MAPPING DYNAMICS SCI, P84
0.293	VANDENBELT H, 1989, EXPERT EVIDENCE INTE
0.287	MEYER M, 1998, SCIENTOMETRICS, V42, P195
0.264	CARPENTER MP, 1981, WORLD PATENT INFORMA, V3, P160
0.258	BRAUN T, 1997, SCIENTOMETRICS, V38, P321
0.254	*US DEP COMM, 1992, TECHN PROF REP SEM
0.293	BRAUN E, 1982, REVOLUTION MINIATURE
0.293	NARIN F, 1987, RES POLICY, V16, P143
0.293	ROGERS EM, 1984, SILICON VALLEY FEVER
0.293	SAXENIAN AL, 1991, RES POLICY, V20, P423
0.253	COLLINS P, 1988, RES POLICY, V17, P65
0.235	PAVITT K, 1984, RES POLICY, V13, P343
0.197	NARIN F, 1994, SCIENTOMETRICS, V30, P147
0.162	CAMPBELL RS, 1979, TECHNOLOGY INDICATOR

Table 2.9 A ranked list of documents in the 6th specialty. Small's three articles appear again in this specialty, suggesting a considerable overlap between the 3rd and the 6th specialties

Factor loading	Cited references
0.273	GROSSMAN GM, 1991, INNOVATION GROWTH GL
0.211	PRICE DJD, 1963, LITTLE SCI BIG SCI
0.199	ARTHUR WB, 1989, ECON J, V99, P116
0.177	NARIN F, 1988, HDB QUANTITATIVE STU, P465
0.140	SMALL H, 1973, J AM SOC INFORM SCI, V24, P265
0.127	FRANKLIN JJ, 1988, HDB QUANTITATIVE STU, P325
0.127	MOGEE ME, 1998, EXPERT OPIN THER PAT, V8, P213
0.127	SMALL H, 1985, SCIENTOMETRICS, V7, P391
0.127	SMALL H, 1974, SCI STUD, V4, P17
0.108	NARIN F, 1988, HDB QUANTITATIVE STU
0.106	NELSON RR, 1982, EVOLUTIONARY THEORY
0.104	*IITRI, 1968, TECHN RETR CRIT EV S
0.103	MANSFIELD E, 1991, RES POLICY, V20, P1

Table 2.10 Documents appearing in three or more specialties

Number of specialties	Occurring specialties	Documents
4	2,3,4,6	PRICE DJD, 1963, LITTLE SCI BIG SCI
3	1,2,5	NARIN F, 1994, SCIENTOMETRICS, V30, P147
3	1,2,6	*IITRI, 1968, TECHN RETR CRIT EV S
3	3,4,5	*US DEP COMM, 1992, TECHN PROF REP SEM
3	3,4,5	BRAUN E, 1982, REVOLUTION MINIATURE
3	3,4,5	CARPENTER MP, 1981, WORLD PATENT INFORMA, V3, P160
3	3,4,5	NARIN F, 1987, RES POLICY, V16, P143
3	3,4,5	ROGERS EM, 1984, SILICON VALLEY FEVER
3	3,4,5	SAXENIAN AL, 1991, RES POLICY, V20, P423
3	3,4,6	NARIN F, 1988, HDB QUANTITATIVE STU
2	3,6	SMALL H, 1973, J AM SOC INFORM SCI, V24, P265
2	3,6	SMALL H, 1974, SCI STUD, V4, P17
2	3,6	SMALL H, 1985, SCIENTOMETRICS, V7, P391

potential predictors for the emergent trend in technological innovations. Second, a clearer understanding of trends in technological innovations marked by patents in the broader context of scientific literature and patent application information is likely to provide strategic signposts for scientists and engineers to orient their research and development. For small businesses, which normally do not have the resources to track the trends of intellectual properties, patent analysis enhanced by information visualization technology can cut through a thick layer of obscured trends of innovations so as to steer their business in the direction of a more competitive market position.

2.5 Conclusion

The Semantic Web emphasizes that data should be machine-understandable, whereas information visualization aims to maximize our perceptional and cognitive abilities to make sense of visual-spatial representations of abstract information structures. One of the fundamental requirements of the Semantic Web is to annotate Web data with ontology to accomplish machine-understandable Web. Will they fit to work along with one another? Our illustrative example is intended to demonstrate that on the one hand, the Semantic Web can largely simplify some information visualization tasks today, and semantic annotation can be utilized for semantic visualization; on the other hand, the two fields differ from their philosophical groundings to tactical approaches to individual problems such as knowledge modeling and representation. A lot of theoretical and practical work remains to be done to find the right track for the two fields to work together harmoniously.

2.6 References

Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall.

Bayerl, P.S., Lungen, H., Gut, U., Paul, K.I. (2003). Methodology for reliable schema development and evaluation of manual annotations. *Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (KCAP).*

Berners-Lee, T. (1998). *Semantic Web Roadmap* [Web]. W3. Retrieved December 10, 2001, 2002, from the World Wide Web: http://www.w3.org/DesignIssues/Semantic.html.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American,* 5(1).

Boyack, K.W., Wylie, B.N., Davidson, G.S., Johnson, D.K. (2000). *Analysis of Patent Databases Using Vxinsight (SAND2000-2266C).* Albuquerque, NM: Sandia National Laboratories.

Braam, R.R., Moed, H.F., Raan, AFJv. (1991a). Mapping of science by combined co-citation and word analysis. I: Structural aspects. *Journal of the American Society for Information Science,* 42(4):233–251.

Braam, R.R., Moed, H.F., Raan, AFJv. (1991b). Mapping of science by combined co-citation and word analysis. II: Dynamical aspects. *Journal of the American Society for Information Science,* 42(4):252–266.

Brin, S. (1998). Extracting patterns and relations from the World Wide Web. *WebDB Workshop at 6th International Conference on Extending Database Technology.*

Broekstra, J., Kampman, A., Harmelen, Fv. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. *International Semantic Web Conference 2002.* Sardinia, Italy.

Callon, M., Law, J., Rip, A. (Eds.). (1986). *Mapping the Dynamics of Science and Technology: Sociology of Science in the Real World.* London: Macmillan Press.

Card, S., Mackinlay, J., Shneiderman, B. (Eds.). (1999). *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann.

Card, S.K. (1996). Visualizing retrieved information: A survey. *IEEE Computer Graphics and Applications,* 16(2):63–67.

Chen, C. (1998a). Bridging the gap: The use of Pathfinder networks in visual navigation. *Journal of Visual Languages and Computing,* 9(3):267–286.
Chen, C. (1998b). Generalised similarity analysis and Pathfinder network scaling. *Interacting with Computers,* 10(2):107–128.
Chen, C. (1999). *Information Visualisation and Virtual Environments.* Springer-Verlag London.
Chen, C. (2003). *Mapping Scientific Frontiers: The Quest for Knowledge Visualization.* Springer-Verlag London.
Chen, C., Cribbin, T., Macredie, R., Morar, S. (2002). Visualizing and tracking the growth of competing paradigms: Two case studies. *Journal of the American Society for Information Science and Technology,* 53(8):678–689.
Chen, C., Kuljis, J., Paul, R.J. (2001). Visualizing latent knowledge. *IEEE Transactions on Systems, Man, and Cybernetics,* Part C, 31(4):518–529.
Chen, C., Paul, R.J. (2001). Visualizing a knowledge domain's intellectual structure. *Computer,* 34(3):65–71.
Cimiano, P., Handschuh, S., Staab, S. (2004). Towards the self-annotating Web. *Thirteenth International Conference on World Wide Web,* pp. 462–471.
Crane, D. (1972). *Invisible Colleges: Diffusion of Knowledge in Scientific Communities.* Chicago: University of Chicago Press.
Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. *40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02).*
Cunningham, H., Maynard, D., Tablan, V. (2000). *JAPE: A Java Annotation Patterns Engine.* Department of Computer Science, University of Sheffield.
Cutting, D. (2004). *Apache Jakarta Lucene.* Retrieved August 31, 2004, from the World Wide Web: http://jakarta.apache.org/lucene/docs/index.html.
Dill, S., Gibson, N., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y. (2003). SemTag and Seeker: Bootstrapping the Semantic Web via automated semantic annotation. *Twelfth International World Wide Web Conference,* May 20–24, 2003, Budapest, Hungary, pp. 178–186.
Dingli, A., Ciravegna, F., Wilks, Y. (2003). Automatic semantic annotation using unsupervised information extraction and integration. *K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation.*
Garfield, E. (1955). Citation indexes for science: A new dimension in documentation through association of ideas. *Science,* 122:108–111.
Garfield, E., H. S.I., Torpie, R.J. (1964). *The Use of Citation Data in Writing the History of Science.* Philadelphia: Institute for Scientific Information.
Goguen, J. (2000). *Information Visualization and Semiotic Morphisms.* University of California at San Diego. Retrieved August 8, 2001, from the World Wide Web: http://www-cse.ucsd.edu/users/goguen/papers/sm/vzln.html.
Greaves, M. (2004). *DAML-DARPA Agent Markup Language.* Defense Advanced Research Projects Agency. Retrieved August 31, 2004, from the World Wide Web: http://www.daml.org/.
Handschuh, S., Staab, S., Ciravogna, F. (2002). S-CREAM: Semi-automatic CREAtion of Metadata. *SAAKM 2002 Semantic Authoring, Annotation & Knowledge Markup: Preliminary Workshop Programme,* Tuesday, July 23, 2002.
He, D.C., Wang, L. (1990). Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing,* 28(4).
Hearst, M.A. (1999). User interfaces and visualization. In: Baeza-Yates, R., Ribeiro-Neto, B. (Eds.), *Modern Information Retrieval.* Addison-Wesley (pp. 257–224).
Herman, I., Melançon, G., Marshall, M.S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics,* 1:24–44.
Hjorland, B. (1997). *Information Seeking and Subject Representation: An Activity-Theoretical Approach to Information Science.* Westport: Greenwood Press.
Hjorland, B., Albrechtsen, H. (1995). Toward a new horizon in information science: Domain analysis. *Journal of the American Society for Information Science,* 46(6):400–425.
Hollan, J.D., Bederson, B.B., Helfman, J. (1997). Information visualization. In: Helenader, M.G., Landauer, T.K., Prabhu, P. (Eds.), *The Handbook of Human Computer Interaction.* The Netherlands: Elsevier Science (pp. 33–48).
Kogut, P., Holmes, W. (2001). AeroDAML: Applying information extraction to generate DAML annotations from Web pages. *First International Conference on Knowledge Capture.*
Kuhn, T.S. (1962). *The Structure of Scientific Revolutions.* Chicago: University of Chicago Press.
Maedche, A., Staab, S. (2001). Ontology learning for the Semantic Web. *IEEE Intelligent Systems,* 16(2): 72–79.
Maynard, D. (2003). *Multi-Source and Multilingual Information Extraction.* Expert Update.

Mukherjea, S. (1999). Information visualization for hypermedia systems. *ACM Computing Surveys.*
Mukherjee, S., Yang, G., Ramakrishnan, I.V. (2003). Automatic annotation of content-rich HTML documents: Structural and semantic analysis. *Second International Semantic Web Conference.*
Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., Goranov, M. (2003). KIM: Semantic annotation platform. *2nd International Semantic Web Conference (ISWC2003),* October 20–23, 2003, Florida, USA, pp. 834–849.
Price, D.D. (1963). *Little Science, Big Science.* New York: Columbia University Press.
Price, D.D. (1965). Networks of scientific papers. *Science,* 149:510–515.
Princeton University CSL (2004). *WordNet: A Lexical Database for the English Language.* Retrieved August 31, 2004, from the World Wide Web: http://www.cogsci.princeton.edu/∼wn/.
Reeve, L., Han, H. (2005). Survey of semantic annotation platforms. *20th Annual ACM Symposium on Applied Computing,* March 13–17, 2005, Santa Fe, New Mexico.
Schvaneveldt, R.W. (Ed.). (1990). *Pathfinder Associative Networks: Studies in Knowledge Organization.* Norwood, NJ: Ablex Publishing.
Sher, I., Garfield, E. (1966). New tools for improving and evaluating the effectiveness of research. *Research Program Effectiveness,* July 27–29, 1965, Washington, DC, pp. 135–146.
Small, H. (1973). Co-citation in scientific literature: A new measure of the relationship between publications. *Journal of the American Society for Information Science,* 24:265–269.
Small, H.G., Griffith, B.C. (1974). The structure of scientific literatures. I: Identifying and graphing specialties. *Science Studies,* 4:17–40.
Small, H.S. (1988). Book review of Callon et al., *Scientometrics,* 14(1–2):165–168.
Spence, B. (2000). *Information Visualization.* Addison-Wesley.
Steinberg, S.G. (1994). The ontogeny of RISC. *Intertek,* 3(5):1–10.
Tallis, M. (2003). Semantic word processing for content authors. *Second International Conference on Knowledge Capture,* October 26, 2003, Sanibel, Florida.
Thagard, P. (1992). *Conceptual Revolutions.* Princeton, NJ: Princeton University Press.
Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F. (2002). MnM: Ontology driven semi-automatic and automatic support for semantic markup. *13th International Conference on Knowledge Engineering and Management (EKAW 2002),* Spain, pp. 379–391.
W3C. (2004). *Resource Description Framework (RDF).* Retrieved August 31, 2004, from the World Wide Web: http://www.w3.org/RDF/.
Ware, C. (2000). Information Visualization.
White, H.D., Griffith, B.C. (1981). Author co-citation: A literature measure of intellectual structure. *Journal of the American Society for Information Science,* 32:163–172.
White, H.D., McCain, K.W. (1998). Visualizing a discipline: An author co-citation analysis of information science, 1972–1995. *Journal of the American Society for Information Science,* 49(4):327–356.
Yesilada, Y., Harper, S., Goble, C., Stevens, R. (2003). Ontology-based semantic annotation for enhancing mobility support for visually impaired Web users. *K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation.*

Chapter 3

Ontology-Based Information Visualization: Toward Semantic Web Applications

Christiaan Fluit, Marta Sabou, and Frank van Harmelen

3.1 Introduction

The Semantic Web is an extension of the current World Wide Web, based on the idea of exchanging information with explicit, formal, and machine-accessible descriptions of meaning. Providing information with such semantics will enable the construction of applications that have an increased awareness of what is contained in the information they process and that can therefore operate more accurately. This has the potential of improving the way we deal with information in the broadest sense possible, for example, better search engines, mobile agents for various tasks, or even applications yet unheard of. Rather than being merely a vision, the Semantic Web has significant backing from various institutes such as DARPA, the European Union, and the W3C, all of which have performed a variety of Semantic Web activities.

In order to be able to exchange the semantics of information, one first needs to agree on how to explicitly model it. Ontologies are a mechanism for representing such formal and shared domain descriptions. They can be used to annotate data with labels (metadata) indicating their meaning, thereby making their semantics explicit and machine-accessible.

Many Semantic Web initiatives emphasize the capability of *machines* to exchange the meaning of information. Although their efforts will lead to an increased quality of the application's results, their user interfaces often take little or no advantage of the increased semantics. For example, an ontology-based search engine could use its ontology to enrich the presentation of the resulting list to the end user, for example, by replacing the endless list of hits with a navigation structure based on the semantics of the hits.

Visualization is becoming increasingly important in Semantic Web tools. In particular, visualization is used in tools that support the development of ontologies, such as ontology extraction tools (OntoLift, Text-to-Onto) or ontology editors (Protégé, OntoLift). The intended users of these tools are ontology engineers that need to gain an insight into the complexity of the ontology. Therefore, these tools employ schema visualization techniques that primarily focus on the structure of the ontology (i.e., its concepts and their relationships). We presented a detailed overview of these tools in Fluit et al. (2003).

The Cluster Map visualization technique, developed by the Dutch software vendor Aduna (http://aduna.biz), bridges the gap between complex semantic structures and

their simple, intuitive user-oriented presentation. It presents semantic data to end users who want to leverage the benefits of Semantic Web technology without being burdened with the complexity of the underlying metadata. For end users, instance information is often more important than the structure of the ontology that is used to describe these instances. Accordingly, the Cluster Map technique focuses on visualizing instances and their classifications according to the concepts of the ontology.

We have reported in previous work (Fluit et al., 2002; 2003) on case studies that exploit the expressive power of this technique. Since then, the growth of the Semantic Web has made it possible to take this technology a step further and integrate it in three different applications. Two of them are employed within Semantic Web research projects. The third is a commercial information retrieval application. These applications exhibit the characteristics of a typical Semantic Web tool: they provide easy (visual) access to a set of heterogeneous, distributed data sources and rely on Semantic Web encoding languages and storage facilities for the manipulation of the visualized data.

This chapter is centered on the description of these three applications. First, we will explain the contents of the Cluster Map visualization and the kind of ontologies it visualizes in Section 3.2. Section 3.3 presents the three real-life applications that incorporate the visualization. These two sections lead to a discussion in Section 3.4 on how the visualization can support several user tasks, such as analysis, search, and exploration. Some considerations for future work and a summary conclude this chapter.

3.2 Cluster Map Basics

The Cluster Map is used to visualize lightweight ontologies that describe a domain through a set of classes (concepts) and their hierarchical relationships. Also known as taxonomies, such ontologies are frequently used in several domains (biology, chemistry, libraries) as classification systems. Information architects consider taxonomies as basic building blocks, representing the backbone of most Web sites. Nonformal taxonomies are already widely used in Web applications for product classification (e.g., Amazon) or Web directories (e.g., Yahoo, Open Directory Project—ODP). Taxonomies are also part of Semantic Web standards such as RDF and Topic Maps.

Due to the specialization relationship that is encoded in the hierarchy, the set of objects in a subclass is a subset of its superclass. The set of subclasses of a class is *incomplete* when their union does not contain all the objects of the superclass. Classes that share instances are *overlapping* if no specialization relationship holds between them. These characteristics are very common for taxonomies. However, they are difficult to show satisfactorily in textual representations. Our visualization offers an alternative in this matter.

The Cluster Map visualizes the instances of a number of selected classes from a hierarchy, organized by their classifications. The Cluster Map in Figure 3.1 shows a collection of job offers organized according to a very simple ontology. Each small yellow sphere represents an instance (a job offer). The classes are represented as rounded rectangles, stating their names and cardinalities. Directed edges connect classes and point from specific to generic (e.g., *IT* is a subclass of *Job Vacancies*). Balloon-shaped edges connect instances to their most specific class(es). Instances with the same class membership are grouped in *clusters*. Our example contains six clusters; two of them represent overlaps between classes.

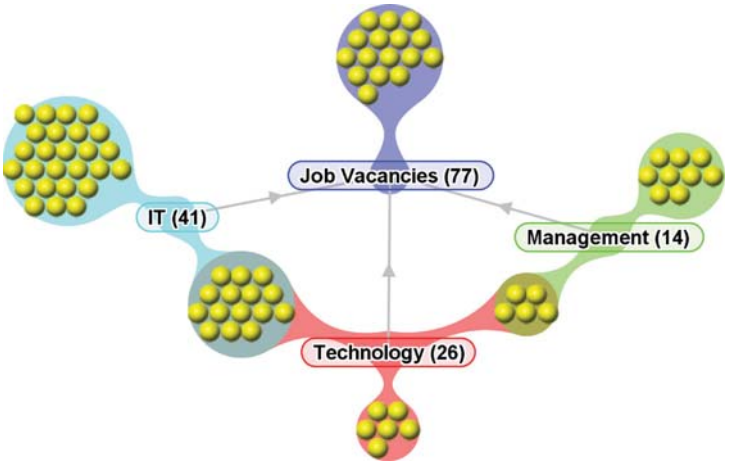


Figure 3.1 An example Cluster Map.

The organization of the graph is computed using a variant of the well-known spring-embedder algorithm (Eades, 1984). On the one hand, the class and cluster nodes repel each other. On the other hand, the edges, connecting two classes or clusters to their classes, produce an attractive force between the connected nodes (i.e., they work as "springs"). The layout algorithm has been optimized for this particular kind of graph, both *qualitatively* (using the semantics of the graph to obtain a more insightful visualization) and *quantitatively* (obtaining a layout fast enough to permit interactive usage).

The added value of our visualization lies in its expressivity. The classes and their relationships (the vocabulary of the domain) are easy to detect. Also, it is immediately apparent which items belong to one or multiple classes, which classes overlap (e.g., *Technology* and *Management*) and which do not (*IT* and *Management*). The cardinality of classes and clusters is visible: the top class has the most objects; *Technology* shares more objects with *IT* than with *Management*. The subclasses of the root class are incomplete as their union does not cover the superclass: some members of *Job Vacancies* were not further classified.

Another interesting aspect of the visualization is that geometric closeness in the map is related to semantic closeness. This is a consequence of the graph layout algorithm. Classes are semantically close if they share many objects. Indeed, the more objects two classes share, the closer they are represented in the graph. Objects are semantically close if they belong to the same class(es). Indeed, objects with the same class membership are clustered.

The Cluster Map visualization is highly configurable. Several graph color schemes are available, each tuned toward a different task, for example, by displaying each class using a unique color (see Figure 3.1) or by indicating cluster relevance using color brightness (see Figure 3.5). Filters can reduce the complexity of the visualization, for example, by removing all clusters representing the overlap of less or more than n classes, or all clusters with cardinality above or below a certain threshold. The scalability of the visualization can be improved by displaying a cluster as a single graphical entity with its cardinality indicated; see Fluit et al. (2003) for an example.

3.3 Applications

In this section, we describe three applications in which the Cluster Map is prominently used. The first two are results from research projects; the last one is a commercial off-the-shelf application.

3.3.1 The DOPE Browser

The exploration of large information spaces is a difficult task, especially if the user is not familiar with the terminology used to describe information. Conceptual models of a domain in terms of thesauri or ontologies can remedy this problem to some extent. In order to be useful, there is a need for interactive tools for exploring large information sets based on conceptual knowledge. The DOPE project (a collaboration between Elsevier, Collexis, Aduna, and the Vrije Universiteit Amsterdam) has developed a thesaurus-based browser that supports a mixed-initiative exploration of large online resources in the domain of drugs and diseases. The DOPE Browser provides support for thesaurus-based search and topic-based exploration of query results, and makes extensive use of Cluster Maps for both visualizing and exploring query results.

The DOPE Browser supports the user in navigating the information space by providing the following functions:

- *Query Formulation:* Query interfaces that contain a graphical presentation of the available knowledge (ontology) make the query formulation task much easier as the user is already acquainted with the used terminology. The ideal situation would be that queries can be formulated visually.
- *Query Result Presentation:* Inadequate presentation of query answers shadows the performance of many search engines as the user is overwhelmed with results without being able to pick the part that is of interest for him. Graphical presentations can explain the relation of the answer to the original terms of the query.
- *Query Reformulation:* Very often the user is not satisfied by the returned answer: there are either too many items or no items at all. Graphical presentation of results allows a user to perform query relaxation, query broadening, and query narrowing.

Query Formulation: The DOPE Browser uses Elsevier's EMTREE thesaurus to suggest disambiguations of query terms. The EMTREE thesaurus contains about 48,000 preferred terms and 200,000 synonyms in the domain of drugs and diseases, organized in a multilevel hierarchy. The user can browse the hierarchy and select a term or type in a free term that is matched against EMTREE and a corresponding term is proposed. Figure 3.2 shows the state of the DOPE Browser after the following dialogues:

1. The user has first entered an initial search term (the term "aspirin" in the top-level entry-field).
2. This search term was subsequently disambiguated to the EMTREE term "acetylsalicylic acid" by offering to the user different senses of the term that occur in the EMTREE thesaurus.

Query Result Presentation and Exploration: The DOPE Browser presents a visual representation of terms and documents in a network where documents are connected

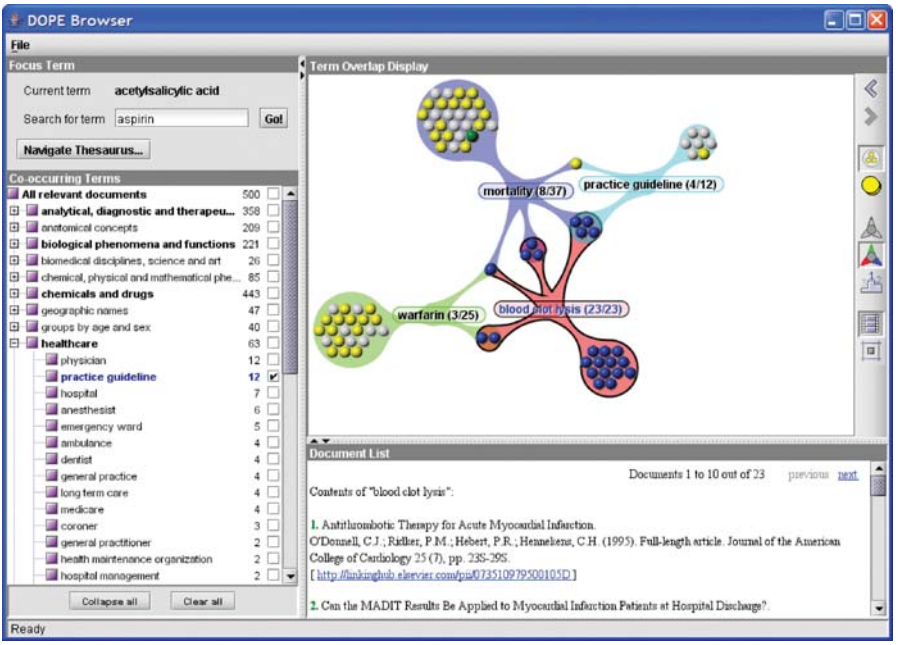


Figure 3.2 The DOPE Browser's main screen.

to the terms they contain (compare Hemmje et al., 1994 and Korfhage, 1991). The advantage of this representation is that it provides an overview of the result set rather than individual information for each document.

Again in terms of the example shown in Figure 3.2:

3. The total number of hits for this term (500, in this example) is presented to the user in terms of which retrieved documents belong to which EMTREE classes (the hierarchical pane on the left). Note that this only shows the subset of EMTREE classes occurring in this document set.
4. The user can then explore the structure of this result-set by selecting the EMTREE classes he or she is interested in.
5. The relations between the selected document sets are displayed as a Cluster Map in the right-hand pane.

Steps 4 and 5 above enable an interactive and semantically rich exploration of the result set of a query. This presentation should be compared with the typical flat list of 500 answers returned by traditional retrieval engines. For example, in Figure 3.2, the user has checked the terms *mortality*, *practice guideline*, *blood clot lysis* and *warfarin*. The visualization shows that within the set of documents about aspirin there is some significant overlap between the terms *blood clot lysis* and *mortality*, and that four of the *practice guidelines* documents relate to these two topics as well.

Various ways exist to further explore this graph. The user can click on a term or a cluster of articles to highlight their spheres and list the document metadata in the Document List panel at the lower right. Moving the mouse over the spheres reveals

the same metadata in a tool tip. The visualizations can also be exported to a clickable image map that can be opened in a Web browser.

A small-scale evaluation of this user interface was conducted at a Drug Discovery conference held in the United States in 2003. Results are reported in Stuckenschmidt et al. (2004).

3.3.2 Xarop/SWAP: Peer-to-Peer Knowledge Management

The tremendous excitement generated by Napster, Gnutella, Kazaa, and the like show the large potential of the peer-to-peer (P2P) paradigm. A case study of the EU-funded SWAP consortium (http://swap.semanticweb.org) has exploited the P2P approach to solve a distributed knowledge management problem.

Knowledge management solutions relying on central repositories sometimes have not met expectations, since users often create knowledge ad-hoc using their individual vocabulary and using their own individual IT infrastructure (e.g., their laptop). Bonifacio et al. (2002) provide an explanation for the failed cases. They argue that traditional knowledge management systems take on a traditional managerial control paradigm, while subjectivity and sociality are essential features of knowledge creation and sharing. From that point they overhaul the traditional architecture of knowledge management systems toward a more decentralized architecture.

The case study that was considered in the SWAP project is based in the tourism sector of the Balearic Islands. On the Balearic Islands, many players (governmental, commercial, and even academic) are working together to further the goals of the tourism industry, which is the major source of income for the Islands. The parties involved are very heterogeneous (government departments, tour operators, hotel chains, economics researchers), and geographically distributed (among the different Balearic Islands). Due to the different working areas and objectives of the collaborating organizations, it proved impossible to set up a centralized knowledge management system or even a completely centralized ontology. The case study partners asked explicitly for a system without a central server, where knowledge sharing is integrated into the normal work, but where very different kinds of information could be easily shared with others.

From a technical point of view, the different organizations can be seen as independently operating nodes within a knowledge network. Nodes can join or disconnect from the network at any moment and can live or act independently of the behavior of other nodes in the system. A node may perform several tasks. The most important one is that it acts as a peer in the network, so it can communicate with other nodes to achieve its goals. But apart from that it may act as an interface to interact with the human user of the network, or it may access knowledge sources to accomplish its tasks. One node may have one or more knowledge sources associated with it. These sources contain the information that a peer can make available to other peers. Examples are a user's file system and mail folders or a locally installed database.

In the Xarop system that was built according to the view above, nodes wrap knowledge from their local sources (files, e-mails, etc.). Nodes ask for and retrieve knowledge from their peers. For communicating knowledge, Xarop transmits RDF structures, which are used to convey conceptual structures (e.g., the definition of what an indicator for air travel is) as well as corresponding data (e.g., data about the number of arrivals by plane). For structured queries as well as for keyword queries, Xarop uses SeRQL, an SQL-like query language that allows for queries combining the conceptual structure of the information with keyword occurrences.

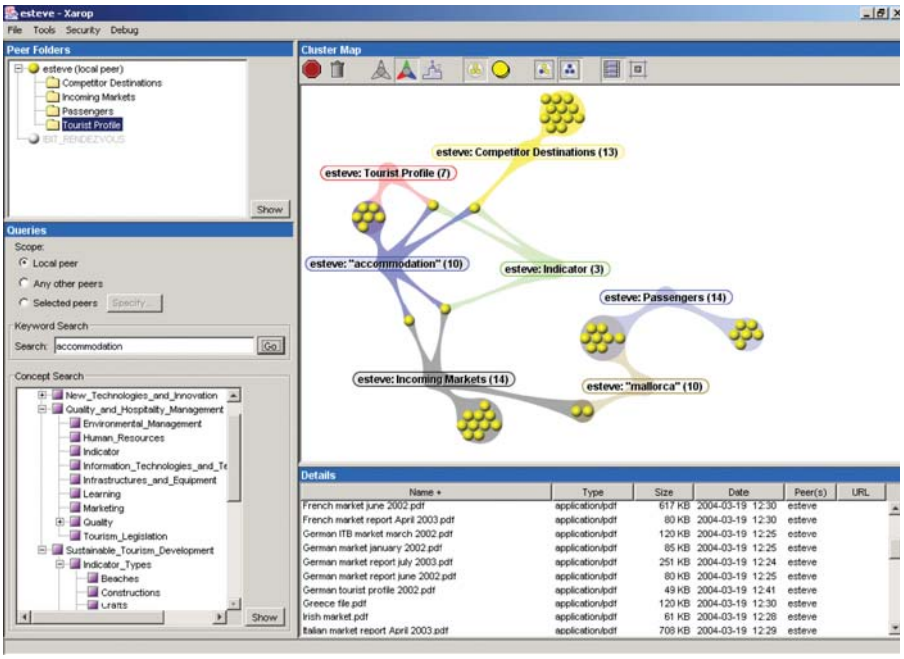


Figure 3.3 Xarop's main screen.

There are three modes of using the system:

1. The basic use of the system is to search for information. One way of doing this is by manually browsing other peer folders. Xarop works in this scenario in a way similar to file systems, with the exception of using P2P technology and unifying views from different sources (like different peers, and from folders/files and bookmarks from Web browser).
2. More advanced information search is possible using full-text search. This way users could easily find all documents containing particular words (e.g., typing "tourism" would result in finding all documents containing the word "tourism").
3. The most powerful mechanism is topic search. In this case, a user is interested in some topic and asks for all documents related to that topic (e.g., tourist arrivals). The system allows this, relying on mechanisms for automatic assignment of documents into topics. This mechanism is multilingual, so the query would take into consideration documents in any of the supported languages—namely, English, Spanish, or Catalan. As automatic mechanisms may not be perfectly accurate, the user is able to manually change document-topic alignments, and these alignments are considered when performing the search.

Figure 3.3 shows the main interface of Xarop, and it covers the following functionalities: sharing folders, full indexing of documents, automatic classification of documents under concepts of the local ontology, manual classification of documents under concepts of the local ontology using drag and drop, browsing folders, keyword-based search, conceptual search, presenting and visualizing results, configuring security access, and downloading files.

The Cluster Map is embedded in a highly interactive GUI, which is designed for browsing-oriented exploration of the populated ontology. Users can browse shared

folders of other peers at the upper left. Alternatively, they can enter keyword queries or select concept terms at the lower left and indicate whether these should be sent to the local node, or to the entire network, or whether specific peers should be addressed. Note that, unlike some of the popular P2P file-sharing applications, the specific peers available on the network are prominently used in the interaction, thereby stimulating community awareness.

The results of these queries are shown as populated classes in the Cluster Map on the right. The software can animate the transition from one visualization to the next. Through interaction a user can also retrieve information about the specific documents that are contained in a class or cluster. Furthermore the visualization can be fine-tuned in several ways, in order to support certain tasks or improve scalability. In the context of Xarop it was important to account for the particularities of P2P systems. Hence, the results are marked with the peer name they are coming from. Results are added to the Cluster Map incrementally, since not all peers answer at the same time. New results are highlighted. The search can be stopped when the user is satisfied.

3.3.3 Aduna AutoFocus

The last system we discuss is AufoFocus, a commercial application developed at Aduna that brings semantic, multidimensional information visualization and navigation to everyone's desktop. It lets users oversee and access the overwhelming amount of information that they may have in their personal information sources.

AutoFocus discloses a user's personal information sources, such as files stored on a hard drive, e-mails, frequently used Web sites, and intranets. Its interface lets the user navigate through the various types of metadata extracted from these sources and visualize their document sets.

More precisely, users explicitly define their sources in AutoFocus, for example, by specifying one or more folders on the hard drive or in a mailbox. The system will subsequently scan the files in these folders and determine attributes such as file type, size, and date. Depending on the file type, the document text and any available metadata such as titles and authors may be extracted as well. All extracted information is indexed for fast retrieval in a local Sesame (Broekstra et al., 2002) RDF repository.

Figure 3.4 shows the main screen, where the user has defined six different information sources. At the left all the available metadata facets are shown. Each window allows the user to express a query using a facet-specific interface. The uppermost window is an exception since it restricts query evaluation to the selected sources. The query results are shown as populated classes in the Cluster Map at the right. In this example we see the results of three queries: a keyword query for "visualization," a phrase search for "semantic web," and an author query for "C. F. M. Fluit." The latter query is matched against the author metadata such as can be found in MS Office, PDF, and HTML files.

The development of AutoFocus has been influenced in numerous ways by the systems developed within the DOPE and SWAP projects described above. For example, the AutoFocus architecture mimics the DOPE architecture in that it uses virtual Sesame repositories to store and provide access to the indexed text and metadata of each information source, while below the surface a number of different storage formats is used, each optimized for a specific type of data.

Using Sesame as the single point of access in the back-end has also led to the development of another product, Aduna Metadata Server, which indexes information

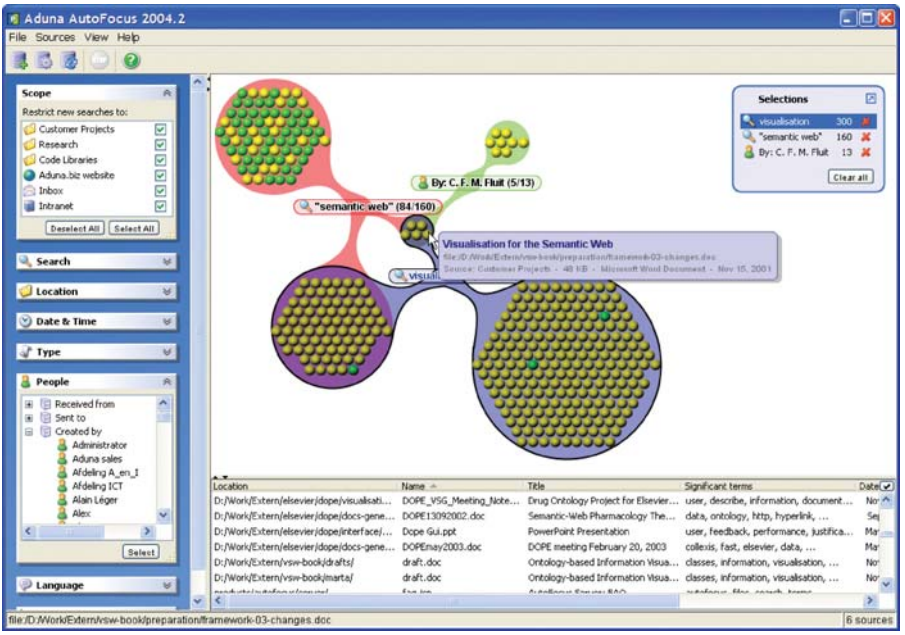


Figure 3.4 AutoFocus's main screen.

sources and provides programmatic access to its metadata using standard Sesame access mechanisms (typically XML over HTTP, but others are possible as well). This way the result of an indexation process, which can be expensive to establish and complex to maintain, can be shared in enterprise environments, where shared network drives and intranet servers are in common use.

The use of the Sesame API provides great flexibility in creating customer solutions. For example, an Aduna Metadata Server can easily be adapted or even replaced by a tailor-made Sesame server that discloses the documents in a document management system. AutoFocus clients can easily query these servers without adaptations. On the other hand, AutoFocus can just as easily be replaced by a custom-made client, as long as it uses Sesame's APIs to query the servers. The only assumption (for now) is that the information in the repositories conform to the AutoFocus RDF schema.

The SWAP project provided an environment for us to experiment with handling asynchronous query responses in a highly interactive user interface. This is a real issue in P2P applications, where you never know when and if a response will arrive. The same is true to a lesser extent for AutoFocus when the user makes use of Metadata Server repositories. Typically a trade-off needs to be made between updating the interface with new results as fast as possible and buffering incoming results to prevent overwhelming the user with changes.

Finally, both the DOPE and the SWAP projects resulted in extensive experiences with designing and implementing a user interface for browsing faceted metadata with information visualization support.

Recently, we have seen an increase in the number of applications entering the desktop search market. It is being targeted by major technology suppliers such as Google and Apple as well as a number of smaller and lesser-known companies. Relatively

few of these applications use information visualization techniques, with Groxis being a well-known exception. The use of Semantic Web standards in this area is still limited to research applications. With AutoFocus we believe that we have made an application that stands out of the competition by the use of unique information visualization techniques and Semantic Web standards that provide a basis for sharing and interoperability in enterprise environments.

An evaluation copy of AutoFocus can be downloaded from http://aduna.biz.

3.4 Uses of Ontology-Based Visualization

So far we have demonstrated the use of our visualization in real-life applications. In what follows we will discuss how our maps support three generic information-seeking tasks: *data analysis*, *querying*, and *exploration*.

3.4.1 Data Analysis

A user may want to analyze a data set in order to gain a better insight into its characteristics, to obtain a global understanding of the collection, and to discover unexpected patterns.

The *static version* of the Cluster Map (i.e., the image itself) already contains information that can be used for such purposes. The classes and their hierarchy provide an understanding of the domain of the data set. The way instances are classified results in class characteristics such as incompleteness and overlaps, showing class relations at the instance level. The cardinality of classes and clusters supports a quantitative analysis. Interpreting this information, one can already derive some domain-specific observations.

The *strategy* used to obtain a static Cluster Map plays an important role. Different strategies support different analysis scenarios, as shown in Fluit et al. (2002):

- *Analysis within a single domain:* In this case a data set is visualized from one or more perspectives, giving insight into the collection. One of our case studies in the abovementioned paper investigated characteristics of a set of job offers by visualizing it according to perspectives such as region and relevant economic sector.
- *Comparison of different data sets:* Data sets can be compared by visualizing them using the same ontology. In Fluit et al. (2002) we have compared two banks by imposing the same ontology on their Web sites and analyzing the two visualizations.
- *Monitoring:* Analyzing a data set at different points in time provides insight into the way it evolves. For example, one could monitor the site of a bank over time and see how its activities evolve (expand/interconnect/. . .).

Viewers that incorporate the Cluster Map (e.g., the DOPE browser or Xarop) turn analysis into an interactive and explorative process. The overview, zoom, and filter facilities offered by the user interface qualitatively enhance the analysis process.

3.4.2 Querying

The goal of a query task is to find a narrow set of items in a large collection that satisfy a well-understood information need (Marchionini, 1995). We will show the benefits

of using the Cluster Map viewer in the four stages of the search task (Shneiderman, 1998):

1. *Query formulation:* The step of expressing an information need through a query is a difficult task. Users encounter difficulties when having to provide terms that best describe their information needs (vocabulary problem). Furthermore, combining these terms in simple logical expressions using "AND" and "OR" is even more complicated. See Shneiderman (1996) for a demonstration of this problem.

   In both the DOPE browser and the Xarop client the classes that describe the domain of a data set are explicitly shown, making the vocabulary choice much easier. There is no need to specify Boolean expressions as they are already visible on the map.
2. *Initiation of action:* Classes selected in the left panel become terms of a query. The search is launched at a mouse-click.
3. *Review of results:* The results of the query are graphically presented to the user. For *n* selected classes the following is shown:
   - The union of the classes (disjunction of *all* query terms)
   - All intersections of the selected classes (conjunction of *some* query terms)
   - As a particular case of 2, the intersection of all classes (conjunction of *all* query terms), if it exists

   Note that the results of simple Boolean expressions are intuitively shown in the map. If the user wants a disjunction of the terms he will analyze all the presented objects. As an added value he will see how the corresponding classes overlap. A more interesting (and probably more frequent) case is when users want the conjunction of the terms. In that scenario, two extreme situations can happen:
   1. The result set is too large (underspecification).
   2. The result set is empty (overspecification).

   If the result set is empty, the user can at least find objects that partially satisfy the query. The left side of Figure 3.5 illustrates a case when the result set to a four-term query is empty: there are no holiday destinations in the analyzed data set that are located in *Loire*, have *2 rooms* that accommodate *4 persons,* and provide a *3 star* quality. The degree of relevance of certain clusters is suggested by their color: the more relevant the darker the shade of the color. It is evident from the visualization that there are two destinations that would be potentially interesting if the customer

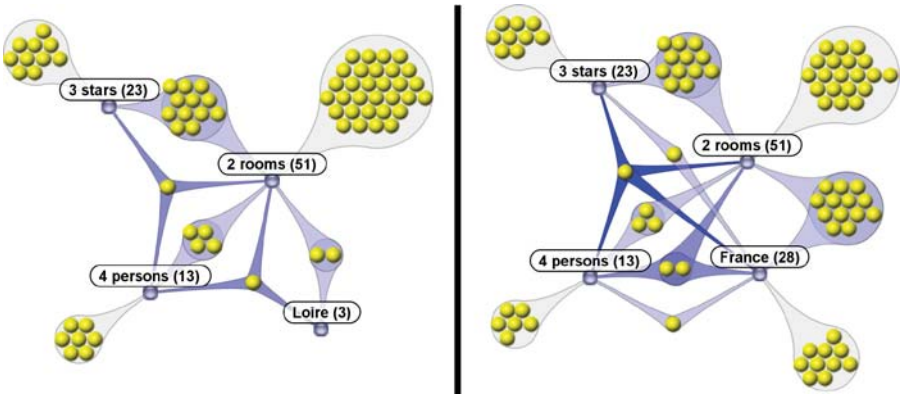


Figure 3.5 Query refinement.

dropped one requirement (quality of accommodation or location). This is a form of *query relaxation.*

4. *Refinement:* According to the conclusions of the result interpretation, the user can narrow or broaden the scope of his search by refining the query.

   If the result set is too large, the user can replace some classes with more specific subclasses. Imagine that in the travel agent scenario the customer would like to go to *France* and that the system returns a huge cluster that satisfies all the criteria. Instead of looking at each returned object, the customer can *refine* his query to use a more specific class, for example *Loire.* This would narrow the scope of the query and return a smaller set of options.

   At the other extreme, in case of an empty set, some classes can be replaced by their superclass. For example, the search for a vacation in Loire was empty for the given settings; however, one destination was found in France, as shown at the right side of Figure 3.5. Note that both narrowing and broadening the scope of the query are possible due to the ontological nature of the domain description. The viewer facilitates choosing more specific or more general classes.

3.4.3 Exploration

We define the exploration task as the process of finding information that is loosely relevant to the user's current activities.

It differs from the querying task in that no specific question needs to be answered. Instead, the user wants to know about relevant information at a more global level (e.g., to see what information is available). Exploration differs from general analysis in that the issue is not to oversee the entire data set in a holistic way but only inspect those parts relevant to the user's current activities.

User studies have shown that exploration support is one of the biggest achievements of the Cluster Map visualization. For example, the informal DOPE usability testing (Stuckenschmidt et al., 2004) suggested a strong applicability for scientific literature surveys for research and educational purposes.

Exploration is strongly related to serendipity, usually defined as the making of fortunate discoveries by accident. Whereas serendipity is a welcome quality of every interface and during any task, we see exploration as a task where the user purposefully browses an information space to find such information.

Serendipitous and explorative interfaces can be realized in many ways. Of course any system that shows documents based on a user query is to some extent serendipitous as it may show a useful document that the user did not know about or forgot about. However, our applications go further than this. When displaying several populated classes in a Cluster Map, users often encounter unsuspected clusterings of the document. For example, overlaps may exist between classes that the user expected to be disjoint, or there may be "missing" overlaps (e.g., when a user expected a certain class to completely include all instances of another class and there are several instances that fall out of this subsumption).

Our AutoFocus application goes even further and suggests new search terms when the user clicks on a class or cluster. Using a statistical algorithm the application tries to find informative terms that occur in the selected set of documents and presents them to the user. The benefit of this is twofold:

1. The user quickly learns about the rough contents of the document set without having to browse through a long result list and inspect every single document. This helps her in building an abstract view of the information.
2. The user is guided in her exploration by being offered search terms that help her to drill down in this document set.

Explorative interfaces turn out to be an effective instrument in battling the information overload that knowledge workers have to face every day, because they diminish the chance of missing valuable information. With such interfaces users are better informed about the information that is at their disposal.

3.5 Future Work

Future developments mainly concern the visualization itself and AutoFocus, as currently these are still continuously being refined. We will mainly focus on those improvements that are relevant to information visualization and Semantic Web research.

First, we want to investigate the use of more expressive ontological structures. RDF Schema and related standards like OWL allow for much more expressive models than the hierarchical, populated classification trees visualized by the Cluster Map. High on our priority list are arbitrary binary relations between documents, since these occur frequently in the data sets we encounter. Examples are hypertext links between Web pages and aggregated information objects such as e-mails containing attachments and zip files. Such improvements impact the query formulation user interface as well as the display of results in the Cluster Map.

In several cases we have observed that the inclusion of a time dimension in the display of information may be useful. An example is the publication date of a document. Visualizing how the population of a set of classes changes over time can greatly improve insight into how the entire document set evolves over time.

Finally, we envision several AutoFocus extensions that improve its general usefulness. One example is the ability to let the user define his own taxonomy, with each class in the taxonomy being defined by a filter on the document characteristics or, when sufficient, by simply listing all the relevant documents. Similar functionality can already be found in certain mail readers, where they define virtual mail folders, and MP3 collection managers, where they define dynamic playlists. In our context, this concept can be extended to arbitrary document sets. Additionally, since AutoFocus and Metadata Server share the same Sesame-based back-end, this can provide a basis toward a P2P version of AutoFocus. Only the typical P2P communication layer has to be provided; a query-answering mechanism is already in place. These two extensions may reinforce each other, as the possibility to share (populated) taxonomies may make their creation and maintenance well worth the effort.

3.6 Summary

This chapter has demonstrated an elegant way to visually represent ontological data. We have described how the Cluster Map visualization can use ontologies to create expressive information visualizations, with the attractive property that classes and objects that are semantically related are also spatially close in the visualization.

Another key aspect of the visualization is that it focuses on visualizing instances rather than ontological models, thereby making it very useful for information retrieval purposes.

A number of applications developed in the past few years have been described that prominently incorporate the Cluster Map visualization. Based on these descriptions, we could distinguish a number of generic information retrieval tasks that are well supported by the visualization.

These applications prove the usability and usefulness of the Cluster Map in real-life scenarios. Furthermore, these applications show the applicability of the visualization in Semantic Web–based environments, where lightweight ontologies are playing a crucial role in organizing and accessing heterogeneous and decentralized information sources.

3.7 References

Bonifacio, M., Bouquet, P., Traverso, P. (2002). Enabling distributed knowledge management: Managerial and technological implications. *Novatica and Informatik/Informatique III.*

Broekstra, J., Van Harmelen, F., Kampman, A. (2002). Sesame: A generic architecture for storing and querying RDF and RDF Schema. *Proceedings of the First International Semantic Web Conference (ISWC'02).*

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium,* 42:149–160.

Fluit, C., Sabou, M., Van Harmelen, F. (2002). Ontology-based information visualisation. In: Geroimenko, V., Chen, C. (Eds.), *Visualizing the Semantic Web.* Springer-Verlag.

Fluit, C., Sabou, M., Van Harmelen, F. (2003). Supporting user tasks through visualisation of lightweight ontologies. *Ontology Handbook.* Springer-Verlag.

Hemmje, M., Kunkel, C., Willett, A. (1994). Lyberworld: A visualization user interface supporting full text retrieval. *Proceedings of the 17th Annual International ACM/SIGIR Conference,* pp. 249–259.

Korfhage, R. (1991). To see or not to see—is that the query? *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pp. 134–141.

Marchionini, G. (1995). *Information Seeking in Electronic Environments.* Cambridge, UK: Cambridge University Press.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. *Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96)*, pp. 336–343.

Shneiderman, B. (1998). *Designing the User Interface.* Menlo Park: Addison-Wesley, pp. 509–551.

Stuckenschmidt, H., De Waard, A., Bhogal, R., Fluit, C., Kampman, A., Van Buel, J., Van Mulligen, E., Broekstra, J., Crowlesmith, I., Van Harmelen, F., Scerri, T. (2004). A topic-based browser for large online resources. *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW'04).*

Chapter 4
Topic Maps, RDF Graphs, and Ontologies Visualization

Bénédicte Le Grand and Michel Soto

4.1 Introduction

Information retrieval in current information systems has become very difficult. These systems' complexity is due to the large volume of data, their lack of structure, and their multidimensionality. The Semantic Web Initiative, proposed by Tim Berners-Lee (2001), aims at making computers process data more efficiently by adding semantics—such as definitions or relationships—between resources on the Web. The Semantic Web is not intended to replace the current Web, but to extend it.

Several levels compose the Semantic Web, thus providing several degrees of expressivity:

- XML is a first level of semantics, which allows users to structure data with regard to their content rather than their presentation (Yergeau et al., 2004).
- However, more semantics can be added with the Resource Description Framework (RDF) or Topic Maps standards. RDF was developed by the World Wide Web Consortium (1999) whereas Topic Maps were defined by the International Organization for Standardization (1997). The Topic Map paradigm was adapted to the Web by the TopicMaps.Org Consortium (2001). Both RDF and Topic Maps aim at representing knowledge about information resources by annotating them.
- Ontologies generally consist of a taxonomy—or vocabulary—and of inference rules such as transitivity and symmetry. Ontologies may be used in conjunction with RDF or Topic Maps (e.g., to allow consistency checking or to infer new information).
- The Semantic Web architecture also defines more expressive layers that provide further reasoning capabilities; they will not be developed in this chapter.

With such standards, any data on the Web may now be annotated semantically. Does this mean that the problem of Web mining is solved? Unfortunately not, but we are definitely on the right way toward a really more efficient information retrieval on the (Semantic) Web.

The first remaining issue is that not all Web resources are semantically annotated. However, more and more people and organizations are aware of the need for metadata on the Web, so this situation will evolve. Moreover, intensive research is conducted in the domain of automatic concepts and relationships extraction so as to automate a part of the semantic annotation. A minimum human involvement will always be needed, at least for checking purposes, but it is really worth it.

The second issue is the volume of metadata itself. As there may be several layers of metadata (metadata about metadata, and so on), we may wonder whether there could be *too much* metadata! The answer is *no*; relevant metadata are always a good thing. Nevertheless, we will have to deal with large quantities of semantic annotations, and thus perform a Semantic Web mining. Web mining and Semantic Web mining are two different things, as the latter has a structure and semantics attached to it. Information retrieval in the Semantic Web is more efficient, because powerful query languages may be used. Several query languages for RDF and Topic Maps have been proposed, such as TMQL (Barta and Garshol, 2003) and RQL (Karvounarakis et al., 2002).

Query languages are the best way to find an answer to a precise question. In this chapter, we focus on another kind of information retrieval, where a user needs to have a global understanding of data or metadata. Visualization is very interesting for such purposes as users may navigate and discover new information. Therefore, we will study visualization mechanisms for Semantic Web formalisms.

This chapter is organized as follows: we first present Topic Maps, RDF, and ontologies; then, we discuss the requirements for the visualization of these semantic structures and we study how existing visualization techniques may be applied to their representation. We conclude by giving a few directions that could lead to the "ultimate" Semantic Web visualization tool.

4.2 Topic Maps, RDF, and Ontologies Basic Concepts

4.2.1 Topic Maps

Topic Maps are an ISO standard that allows us to describe knowledge and to link it to existing information resources. Topic Maps are qualified as "GPS of the information universe," as they are intended to enhance navigation in complex data sets. Although Topic Maps allow organizing and representing very complex structures, the basic concepts of this model—*topics, occurrences,* and *associations*—are simple.

A *topic* is a syntactic construct that corresponds to the expression of a real-world concept in a computer system. Figure 4.1 represents a very small Topic Map that contains four topics: *XML Europe 2000*, *Paris, Ile-de-France,* and *France.* These topics are instances of other topics*: XML Europe* is a *conference*, *Paris* is a *city, Ile-de-France* is a *region,* and *France* is a *country*. A topic type is a topic itself, which means that *conference, city, region,* and *country* are also topics.

A topic may be linked to several information resources (e.g., Web pages) that are considered to be somehow related to this topic. These resources are called *occurrences* of a topic. These occurrences are typed; in the Topic Map represented in Figure 4.1, occurrences of the topic *Paris* may be URLs pointing to some pictures or maps. Occurrences provide means of linking real resources to abstract concepts, which helps organize data and understand their context. It is important to notice that topics and information resources belong to two different layers. Users may navigate at an abstract level—the topic level—instead of navigating directly within data.

The concepts presented so far—topic, topic type, name, occurrence, and occurrence type—allow us to organize information resources with regard to a concept. However, it is interesting to describe relationships between these concepts. This is possible in Topic Maps through topic *associations.*

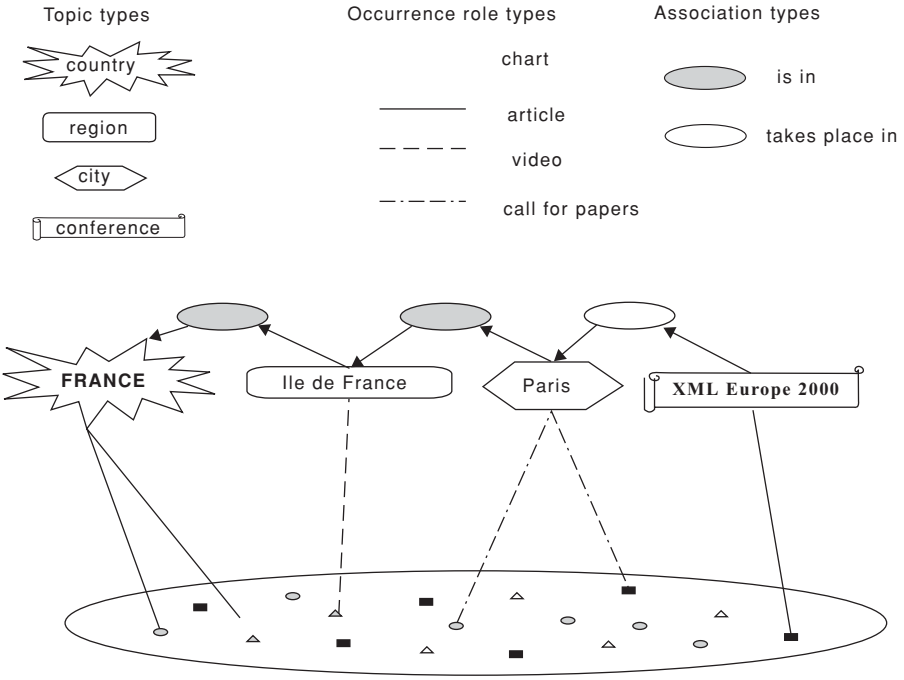


Figure 4.1 Example of Topic Map.

An association adds semantics to data by expressing a relationship between several topics, such as *XML Europe 2000 takes place in Paris, Paris is located in Ile-de-France*, etc. Every topic involved in an association plays a specific role in this association; for example, *Ile-de-France* plays the role of *container* and *Paris* plays the role of *containee*. Associations' roles are also topics. Actually, almost everything in a Topic Map is a topic.

One advantage of Topic Maps is that they add semantics to existing data—by organizing and describing them—without modifying them. Moreover, one Topic Map may describe several information pools and several Topic Maps may apply to one single information pool.

In this section, we described Topic Maps basic constructs; Topic Maps contain *topics* that are connected by *associations* and that point to information resources through *occurrences*. Topics Maps can enhance navigation and information retrieval in complex data sets by adding semantics to these resources. However, a topic may have a high number of dimensions, as it is characterized by its name(s), its type(s), its occurrence(s)—the resources that are related to it—and the role(s) that it plays in associations. Moreover, Topic Maps may also be complex because of their size: they may contain millions of topics and associations.

4.2.2 RDF

The Resource Description Framework (RDF) is a language designed to represent information about resources in the World Wide Web. Examples of such metadata

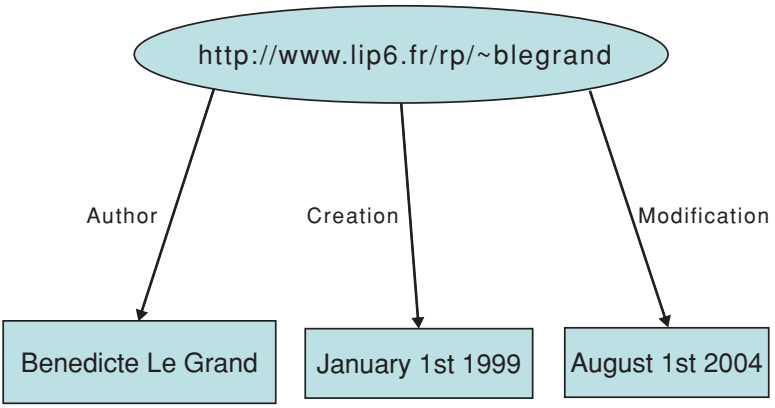


Figure 4.2 Example of RDF graph.

are the *author*, the *creation,* and *modification dates* of a Web page. RDF provides a common framework for expressing semantic information about data so it can be exchanged between applications without loss of meaning. RDF and Topic Maps thus have similar goals. RDF identifies things with Web identifiers (called Uniform Resource Identifiers, or URIs), and describes *resources* in terms of *properties* and property *values*.

Figure 4.2 shows the graphical RDF description of a Web page. This semantic annotation specifies that this page belongs to Benedicte Le Grand, and that it was created on January 1, 1999 and modified on August 1, 2004. This corresponds to three RDF *statements*, giving information respectively on the *author*, the *creation,* and the *modification dates* of this page. Each statement consists of a (*Resource, Property, Value*) triplet. In our example:

- `http://www.lip6.fr/rp/~blegrand` is a **resource.**
- The element <*author*> is a **property.**
- The string "*Benedicte Le Grand*" is a **value.**

As shown in Figure 4.2, statements about resources can be represented as a *graph* of nodes and arcs corresponding to the resources, and their properties and values. Therefore, visualizing an RDF structure comes to visualizing the corresponding RDF graph.

RDF provides an XML syntax (called serialization syntax) for these graphs. The following code is the XML translation of the graph in Figure 4.2:

```
<?xml version="1.0"?>
<RDF>
      <Description about="http://www.lip6.fr/rp/~blegrand">
       <author>Benedicte Le Grand</author>
      <created>January 1, 1999</created>
      <modified>August 1, 2004</modified>
</Description>
</RDF>
```

RDF and Topic Maps are two compatible formalisms: Moore (2001) stated that RDF could be used to model Topic Maps and vice versa. There are slight differences; the notion of scope (context) exists in Topic Maps and not in RDF. RDF is more synthetic and better adapted to queries whereas Topic Maps are better for navigation purposes.

4.2.3 Ontologies

RDF and Topic Maps both describe data and relationships between data. They may also refer to external knowledge, formalized as *ontologies*. Ontologies have a more expressive power; they allow the specification of constraints on classes. They also make reasoning possible, as new knowledge may be inferred (e.g., by transitivity). Let us try to give a definition of ontologies.

According to Tom Gruber (1993), "An ontology is an explicit specification of a conceptualization."

Jeff Heflin, editor of the *OWL Use Cases and Requirements* (2004), considers that "an ontology defines the terms used to describe and represent an area of knowledge. . . .Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. . . .Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations."

Tim Berners-Lee, James Hendler, and Ora Lassila (2001) say that "artificial-intelligence and Web researchers have co-opted the term for their own jargon, and for them an ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules."

Topic Maps, RDF, and ontologies all aim at making the Web more semantic. However, the volume of semantic metadata makes its interpretation challenging. Therefore, we study visualization techniques for these semantic structures. We saw that ontologies were more expressive than Topic Maps or RDF, but all these structures define concepts and relationships between concepts. Therefore, the visualization of Topic Maps, RDF graphs, and ontologies can be called "semantic graphs visualization."

As we have seen in this section, semantic graphs such as Topic Maps, RDF graphs, and ontologies build a kind of semantic network above information resources, which allows users to navigate at a higher level of abstraction (i.e., at the level of metadata rather than data). However, these semantic structures are high-dimensional knowledge bases and they may be very large. Users may still have problems finding relevant information within a Topic Map or an RDF graph, or within an ontology. Therefore, the issue of semantic graphs visualization and navigation is essential. In this chapter, the examples generally deal with one specific category of semantic graph. However, all techniques presented here can be easily applied to all of them.

4.3 Semantic Graphs Visualization

In this section, we address Topic Maps, RDF graphs, and ontologies visualization. Topic Maps and RDF have very similar structures basically consisting of concepts and

relationships between concepts. Ontologies also contain this kind of information, together with reasoning capabilities that will not be described here. Our goal is to ease the visualization and navigation of the semantic graph corresponding to a Topic Map, an RDF graph, or an ontology.

Topic Maps, RDF graphs, and ontologies are very powerful but they may be complex. Intuitive visual user interfaces may significantly reduce the cognitive load of users when working with these complex structures. Visualization is a promising technique for both enhancing users' perception of structure in large information spaces and providing navigation facilities. According to Gershon at al. (1995), it also enables people to use a natural tool of observation and processing—their eyes as well as their brain—to extract knowledge more efficiently and find insights.

First, we present the goals of Topic Maps, RDF graphs, and ontologies visualization. Then, we review different visualization techniques that have been or may be applied to these semantic graphs.

4.3.1 Visualization Goals

The goal of semantic graphs visualization is to help users locate relevant information quickly and explore the structure easily. Thus, there are two kinds of requirements for semantic graphs visualization: representation and navigation. A good representation helps users identify interesting spots whereas an efficient navigation is essential to access information rapidly. We need to understand the structure of metadata and to locate relevant information easily.

According to Schneiderman (1996), "The visual information-seeking mantra is: overview first, zoom and filter, then details on-demand."

4.3.1.1 Representation Requirements

First, we need to provide the user with an overview of the semantic graph. This overview must show the main features of the structure in order to deduce its main characteristics at a glance. Visual representations are particularly fitted to these needs, as they exploit human abilities to detect patterns.

Let us consider the case of a Topic Map (the conclusions may be applied to RDF graphs and to ontologies as stated before). The first thing we need to know about a Topic Map is what it deals with, that is, what its main concepts are. Once they are identified, we need more structural information, such as the generality or specificity of the Topic Map. This kind of information should appear clearly on the representation so as to help users compare different Topic Maps quickly and explore only the most relevant ones in detail. The position of topics on the visual display should reflect their semantic proximity. These properties can be deduced from the computation of Topic Maps metrics, as shown by Le Grand and Soto (2001).

Moreover, Topic Maps are multidimensional knowledge bases. Topics, associations, and occurrences are characterized by many parameters, which should appear somehow in the visualization. The same issue appears in ontologies, where each term can be described by many attributes (slots).

The requirements we stated before are not compatible, as it is neither possible nor relevant to display simultaneously general information and details. We can compare

this with a geographic map; a map of the world cannot and should not be precise. If a user is interested in details, she must be precise in her interest, for example, by choosing a specific country. As in geographical maps, we need to provide different scales in Topic Maps representations.

Moreover, visualizations should be dynamic to adapt to users' needs in real time; combinations of time and space can help ground visual images in one's experience of the real world and so tap into the users' knowledge base and inherent structures.

4.3.1.2 Navigation Requirements

A good navigation allows users to explore the semantic graph and to access information quickly. Navigation should be intuitive so that it is easy to get from one place to another. Several metaphors are possible: users may travel by car, by plane, by metro, or may simply be "teleported"—as on the Web—to their destination. The differences lie in what they see during their journey. From a car, they see details, from a plane they have an overview, etc. Navigation is essential because it helps users build their own cognitive map—a map-like cognitive representation of an environment—and increases the rate at which they can assimilate and understand information.

Free navigation should be kept for small structures or expert users as the probability of getting lost is very high. For beginners who do not know where to start their exploration, predefined navigation paths are preferable until topics of interest are identified.

To sum up the visualization goals, we need to represent the semantic graph as a whole in order to help users understand it globally. This overview should reflect the main properties of the structure. However, users should be able to *focus* on any part of the semantic structure and see all the dimensions they need. Providing these several scales requires the use of different *levels of detail.* Finally, users should be able to navigate easily and intuitively at these different levels of detail.

In the following section, we study which visualization techniques meet our requirements and may be used to represent Topic Maps, RDF graphs, and ontologies.

4.3.2 Visualization Techniques

Many visualization techniques and metaphors are currently available to represent complex data, in particular textual interfaces, graphs, trees, and maps.

4.3.2.1 Textual Interface

Protégé is a textual editor and browser for knowledge models, as described by Noy et al. (2001). It may be used for Topic Maps, RDF graphs, and ontologies. An example of ontology visualization with *Protégé* is given in Figure 4.3. This tool presents a hierarchy of terms (multiple inheritance is possible; in this case terms appear several times in the hierarchy). This textual interface is simple but it is not adapted to large semantic graphs. Visualization capabilities must be added to deal with this problem. Several visualization components for *Protégé* are described in the next section.

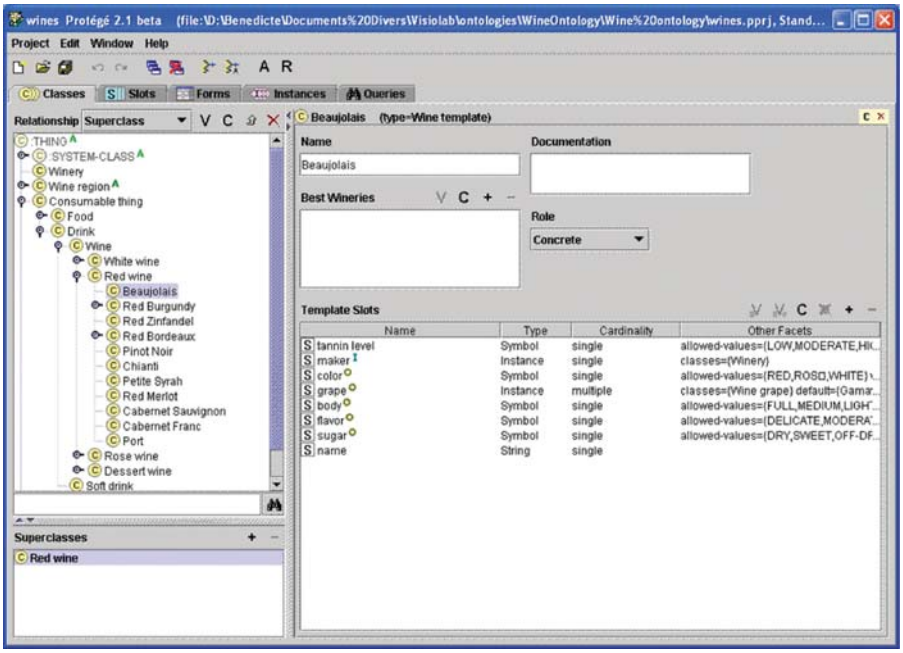


Figure 4.3 *Protégé* ontology editor and browser.

4.3.2.2 Graphs and Trees

Representing Topic Maps, RDF graphs, or ontologies as graphs seems natural as they can be seen as a network, or graph, of topics and associations. The Topic Map shown in Figure 4.1 is displayed as a graph consisting of nodes and arcs. However, this simple representation may become cluttered and difficult to interpret when the size of the structure increases. As a Topic Map may contain millions of topics and associations, it becomes necessary to use sophisticated graph visualization techniques. This is also true for RDF graphs and ontologies.

Graphs and trees are suited for representing the global structure of semantic graphs. However, trees are better understood by human beings since they are hierarchical. Trees are easier to interpret than graphs. Topic Maps, RDF graphs, and ontologies are not simple hierarchies (other types of relationships may exist between concepts) and thus may not be *directly* represented as trees. However, it may be interesting to transform small parts of the semantic structure into trees. By doing so, we may benefit from the advantages of trees.

The challenge of graph visualization is to provide graphs that display many nodes but that remain readable. A first solution, proposed by Munzner (1997), is to use hyperbolic geometry instead of Euclidian geometry, which allows displaying a very large number of nodes on the screen. Figure 4.4 is an example of Topic Maps representation as hyperbolic trees, obtained with the *K42* Topic Map engine developed by *Empolis* (2001).

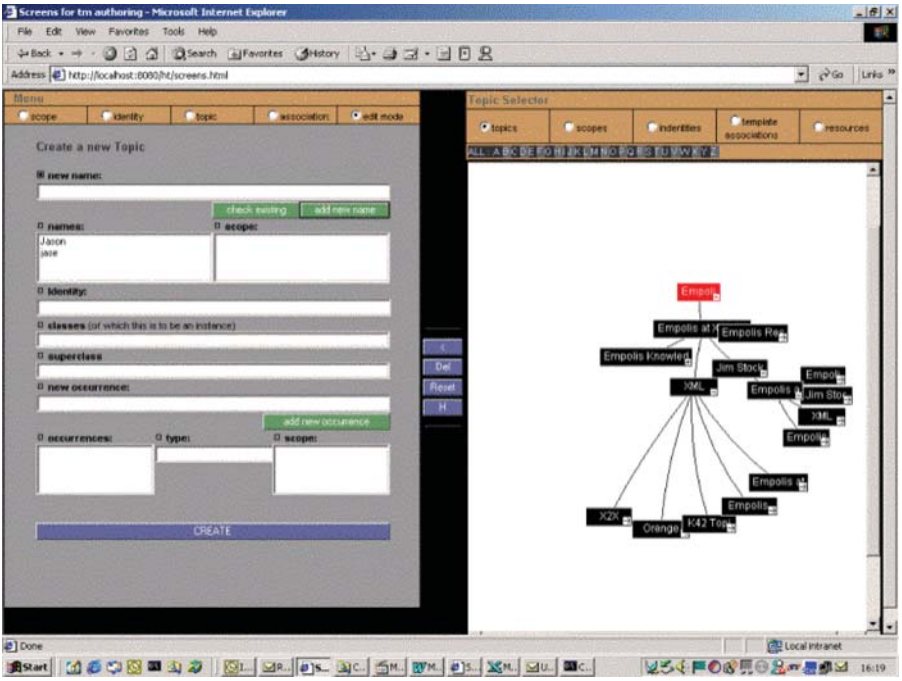


Figure 4.4 *Empolis K42* application.

Another solution to the lack of space on the screen is to represent the semantic graph in three dimensions. A 3D interactive Topic Map visualization tool—*UNIVIT* (Universal Interactive Visualization Tool)—was implemented by Le Grand and Soto (2000). The example shown in Figure 4.5 was generated with *UNIVIT*, which uses virtual reality techniques such as 3D, interaction, and different levels of detail.

Moreover, the quality of the visualization can be increased by an efficient node positioning, which makes it possible to intuitively derive information from the distance between nodes. For instance:

- Related concepts may be represented close to each other in the graph.
- Concepts of the same type or pointing to the same type of resources may be clustered.

Graphs and trees meet our first requirement since they may represent the whole semantic structure. However, users also need to see detailed information about the concepts they are interested in. This second requirement, which consists in representing all the different parameters of a semantic graph (name, type, etc.), may be really challenging. Figure 4.6 is a graph obtained with the *NV3D* software (2000). Different shapes and colors are used to symbolize various dimensions of nodes and arcs of the graph. This kind of graph could be used to visualize a semantic graph; topics would be nodes and associations would be arcs. However, the number of different shapes, colors, icons, and textures is limited. This representation is not suited for a

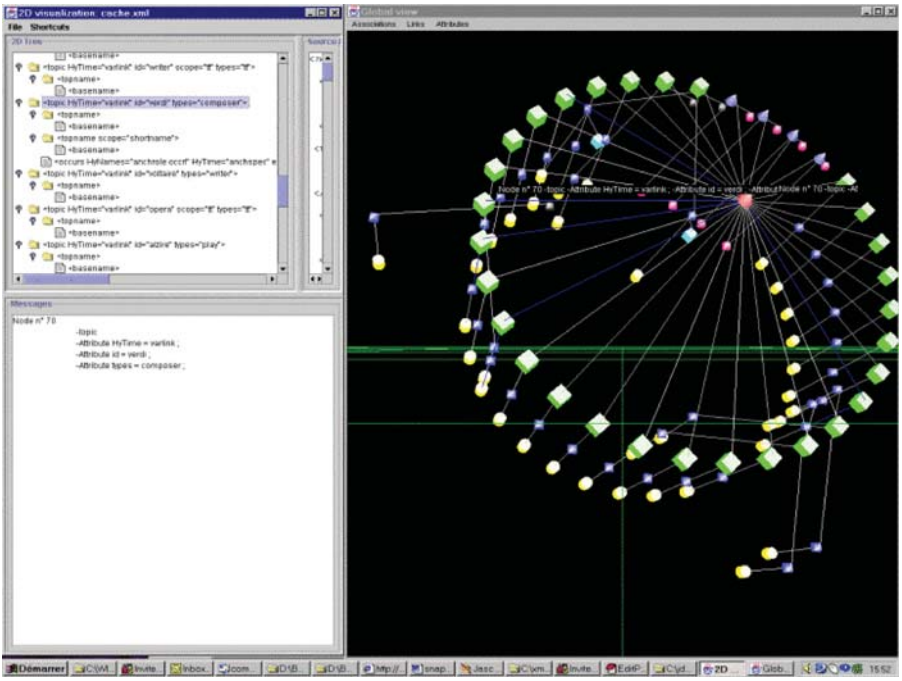


Figure 4.5 3D interactive Topic Maps visualization with *UNIVIT*.

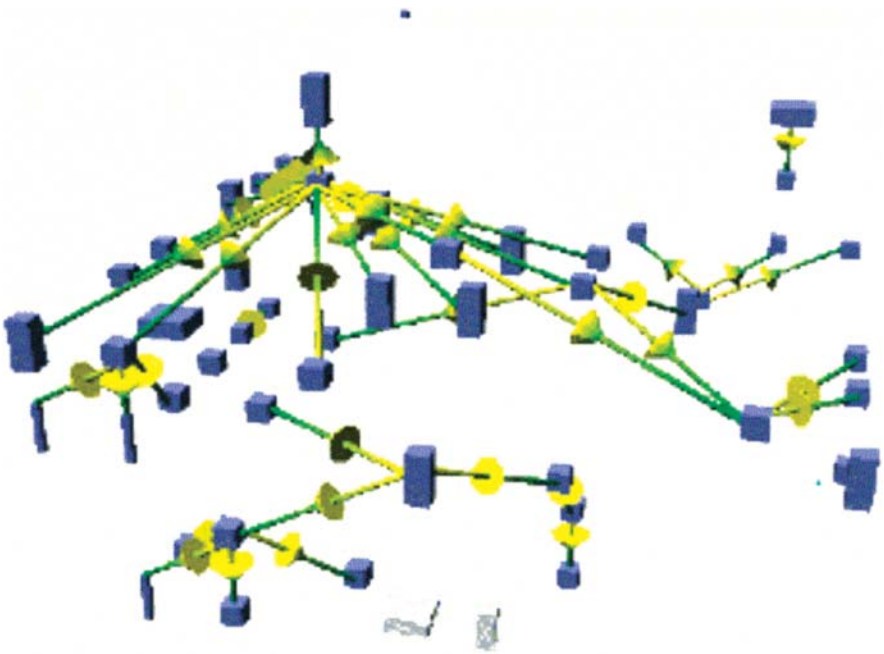


Figure 4.6 *NV3D* graph representation.

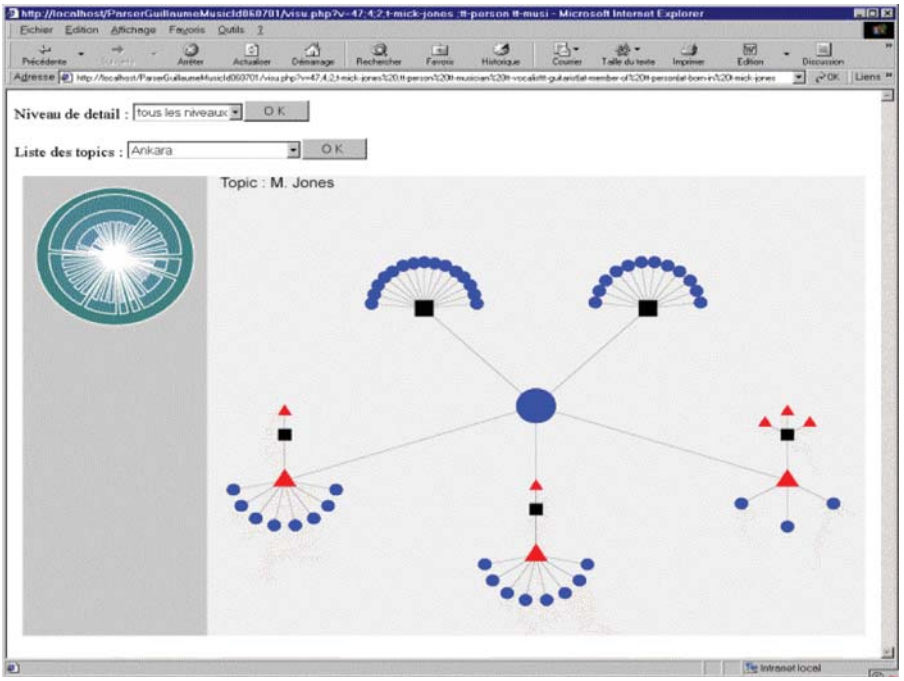


Figure 4.7 Graphical representation of a topic.

Topic Map, an RDF graph, or an ontology that contains millions of elements and relationships.

Let us take the example of a Topic Map. In order to display detailed information, it is thus necessary to focus on a part of the Topic Map. The graph-like overview of the Topic Map helps users understand the structure globally and select specific topics they may be interested in. Once a topic is selected, it is easy to display very precise information about it. Figure 4.7 is a very simple representation of a topic, proposed by Le Grand (2001). The topic itself is a disc, located at the center of the visualization. This topic may be an instance of other topics, which are above him in the graph. The associations in which this topic is involved are symbolized by triangles.

Once users' needs are clearly identified, another useful way to represent a Topic Map is to display a list (or index) from which it is possible to select a topic and see related information. The navigation is usually the same as on Web sites: users click on a link to open a new topic or association. An example of such visualization is provided by the *Ontopia Navigator* (2001), as shown in Figure 4.8.

We showed that the global view and precise parameters could not be displayed at the same level of detail. It is essential that users can navigate easily from one level of detail to another. Several tools already provide interactive graphical visualizations. *Mondeca*'s *Topic Navigator* (2001) builds graph representations in real-time, according to what users are allowed to or need to see (Figure 4.9). Figure 4.10 is an example of Techquila's (2001) *TM4J*'s Topic Map dynamic visualization using *TheBrain* software (2001).

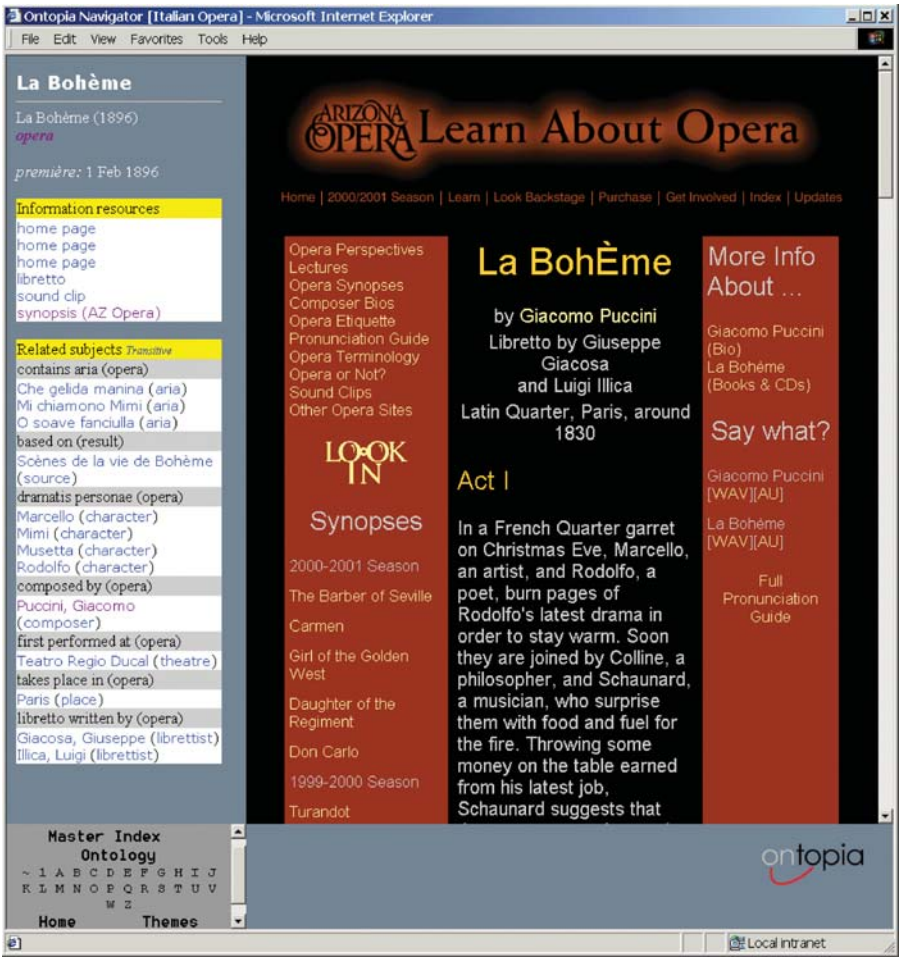


Figure 4.8 *Ontopia Navigator.*

So far, we have given visualization examples applied to Topic Maps representation and navigation. The following solutions have been more specifically used for RDF graphs and ontologies visualization.

IsAViz (Pietriga, 2002) is a visual browsing tool for RDF models, based on AT&T's *GraphViz* package (Ganzner and North, 1999). Because of the layout of *IsAViz* graphs, illustrated in Figure 4.11, it is difficult to see the overall structure of a set of instances, as stated by Mutton and Golbeck (2003).

The following three tools, *Ontoviz*, *Jambalaya,* and *TGVizTab*, are *Protégé* plug-ins, described by Ernst and Allen (to appear).

1. *Ontoviz* is based on *GraphViz*, like *IsAViz* (Figure 4.12). One limit of *Ontoviz* is that the only layout supported is based on the inheritance relationships, whereas other types of links may exist between concepts and need to be stressed by the visual interface.

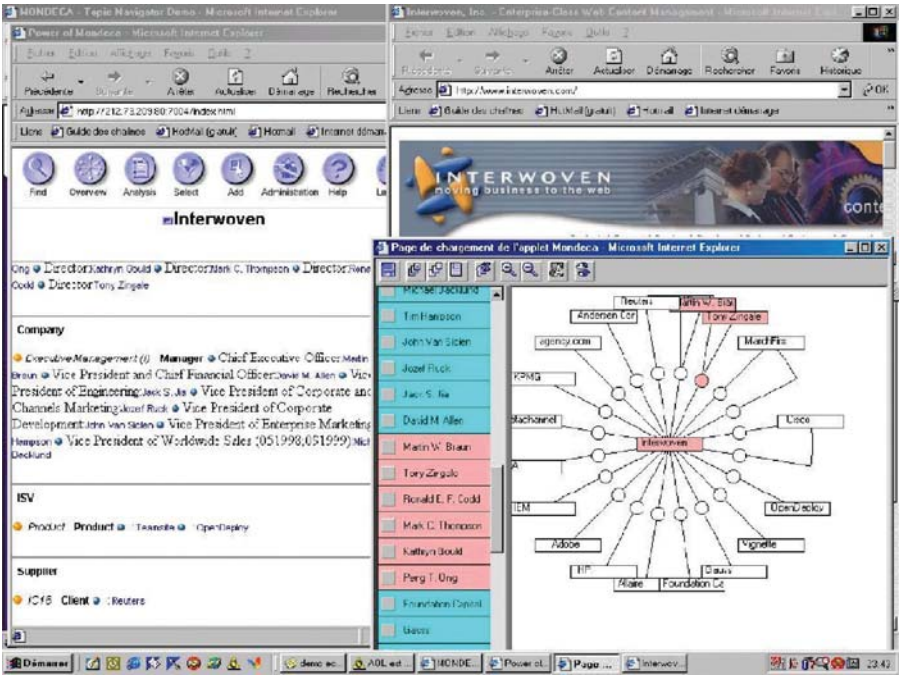


Figure 4.9 *Mondeca's Topic Navigator.*

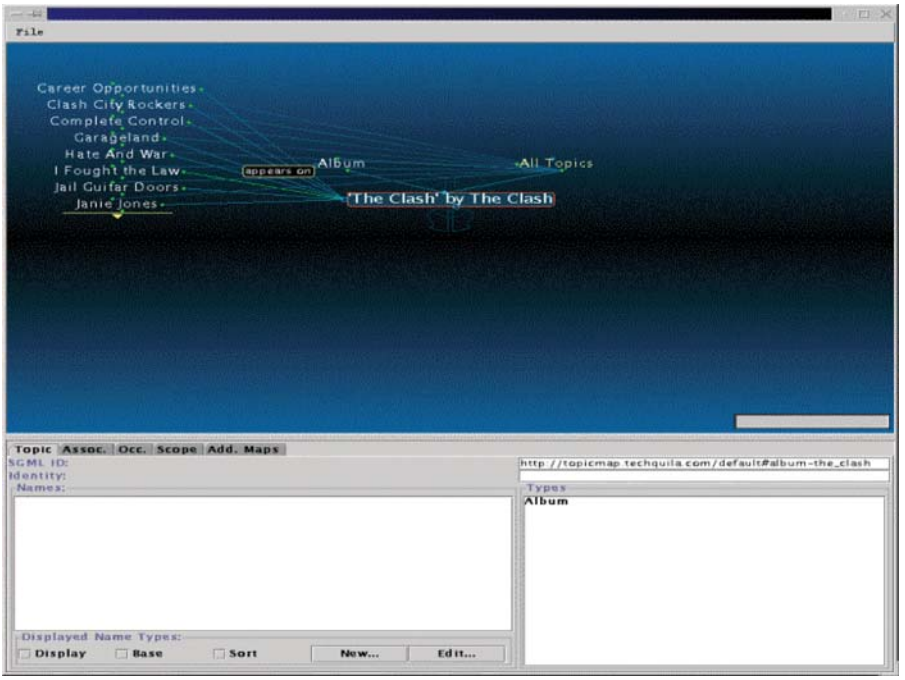


Figure 4.10 *TM4J*'s Topic Map visualization with *TheBrain* software.

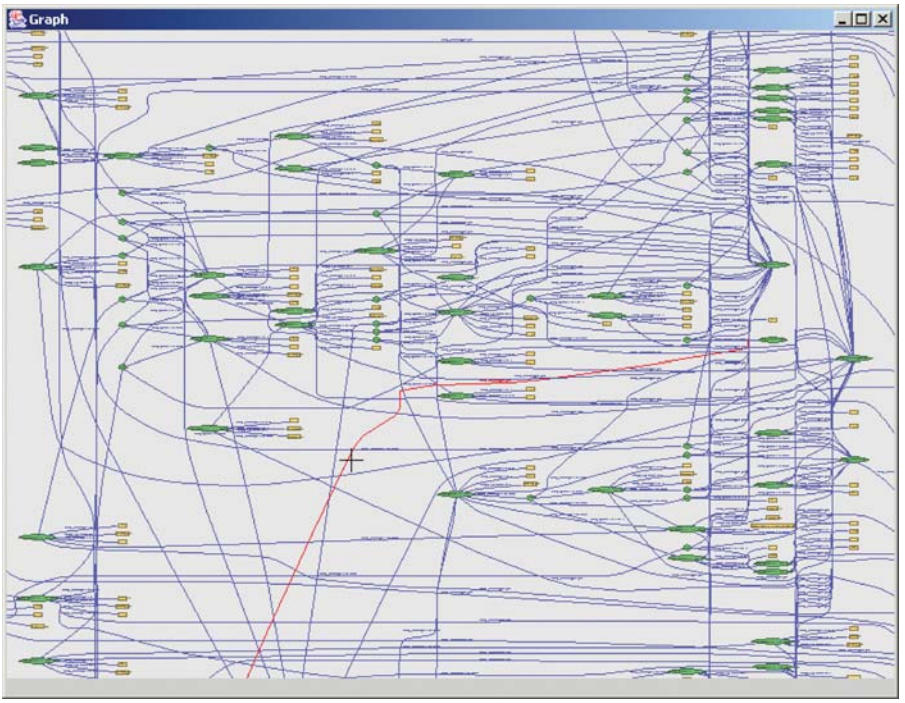


Figure 4.11 *IsAViz* graph.

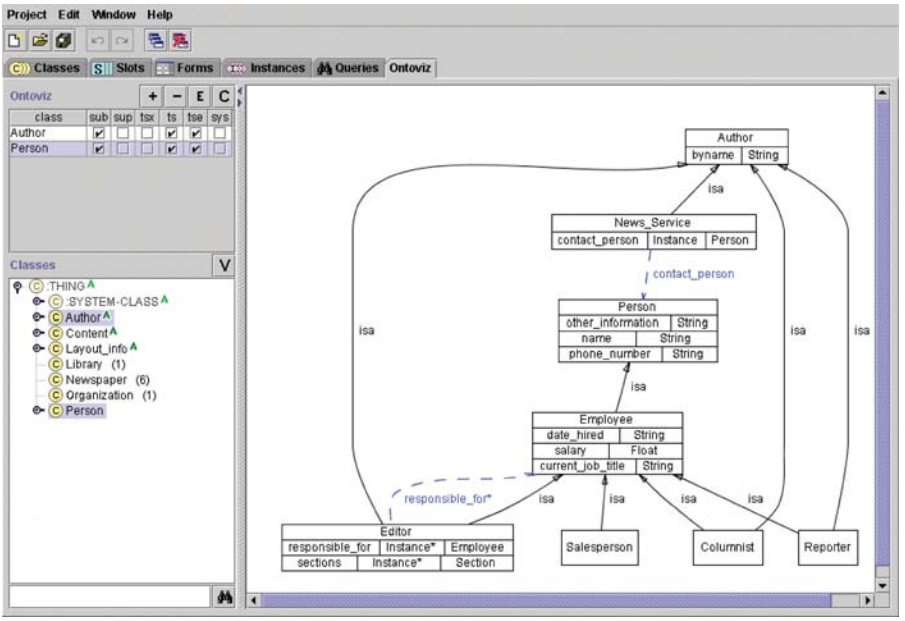


Figure 4.12 *Ontoviz* visualisation component.

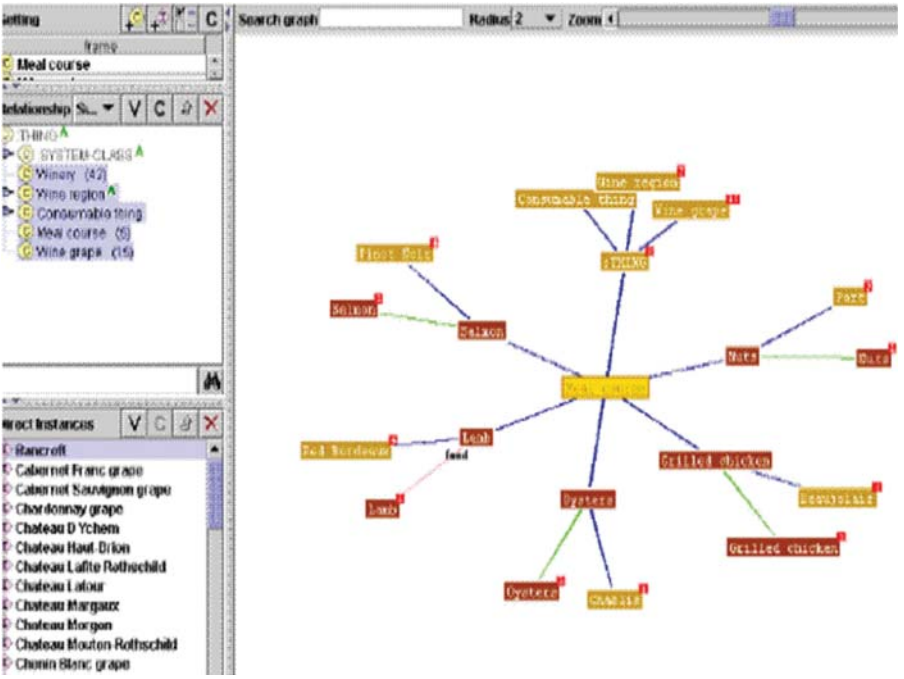


Figure 4.13 *TGVizTab* hyperbolic graph.

2. *Jambalaya* provides zooming capabilities. However, with this extension, the relationships between concepts may be hidden by information about concepts themselves.
3. *TGVizTab*, presented by Alani (2003), uses a spring-embedding algorithm to implement a customizable hyperbolic layout for concepts and relationships, as shown in Figure 4.13. This tool is adapted to an incremental navigation, but it becomes cluttered rapidly and is therefore not fitted to a top-down exploration.

Spectacle, described by Fluit et al. (2002), groups instances in clusters according to their class. This makes navigation easier, but this is not ideal for understanding the structure of data, which is one of our visualization goals.

4.3.2.3 Maps

In this section, we consider the metaphor of a *map* to visualize Topic Maps. This metaphor can be used for RDF graphs or for ontologies, but it has been used for Topic Maps because they were especially designed for navigation, which explains the choice of the name.

Topic Maps aim at enhancing navigation within complex data sets. In the real world, maps are used to achieve this goal. It is thus natural to consider representing Topic Maps as maps.

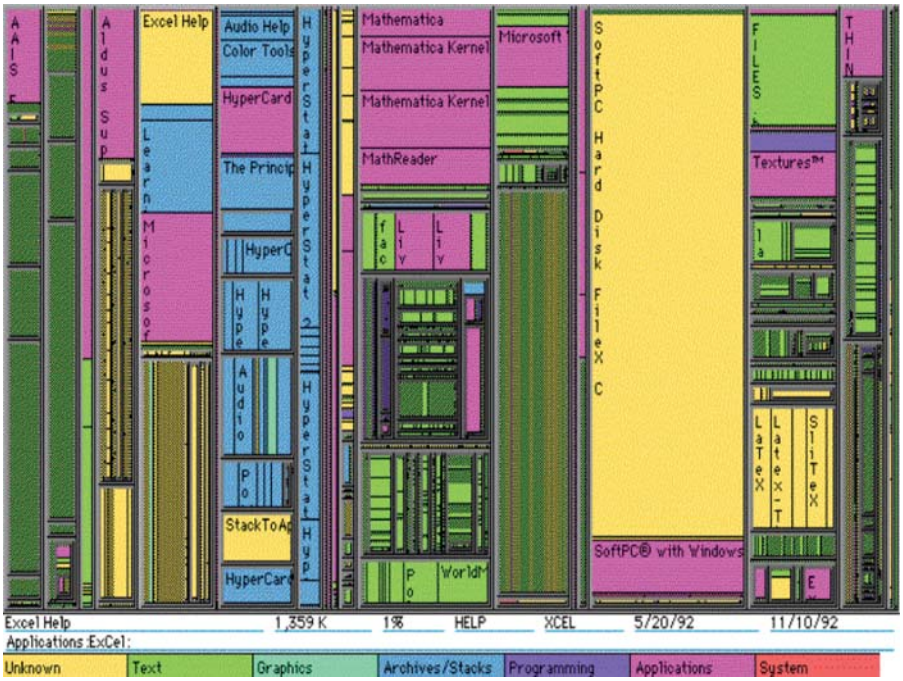


Figure 4.14 Example of *Tree-Map*.

A map should display the most significant elements of the structure. The location and the size of these elements on the map are essential. In the *Tree-Maps* proposed by Bruls et al. (2000), data are reorganized so as to reflect the relative importance of topics, as shown in Figure 4.14. These *Tree-Maps* may be used to represent Topic Maps.

One of the challenges of Topic Maps visualization as maps is to find optimal coordinates for the topics. The *Self-Organizing Maps* (SOM) algorithm, proposed by Kaski, Kohonen et al. (1998) can be used to achieve this by organizing the topics onto a two-dimensional grid so that related topics appear close to each other, as shown in Figure 4.15.

Factor analysis can also be used to compute topics coordinates. Davison (1992) explains how the multidimensional scaling (MDS) algorithm uses similarity measures between topics to provide a 2D map of the structure. Figure 4.16 is an example of map that represents a small Topic Map about rock music.

The *ThemeScape* software by Cartia Inc. (1992) provides different types of maps. They look like topographical maps with mountains and valleys, as shown in Figure 4.17. The concept of the layout is simple: documents with similar content are placed close to each other and peaks appear where there is a concentration of closely related documents; higher numbers of documents create higher peaks. The valleys between peaks can be interesting because they contain fewer documents and more unique content. The labels reflect the major two or three topics represented in a given area of the map, providing a quick indication of what the documents are about. Additional labels often appear when we zoom into the map for greater detail. We can zoom to

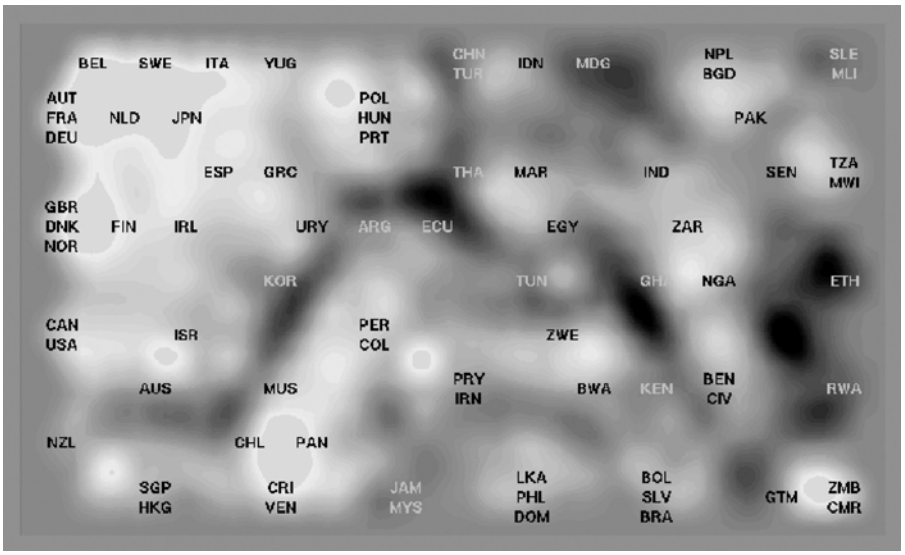


Figure 4.15 Self-organizing map.

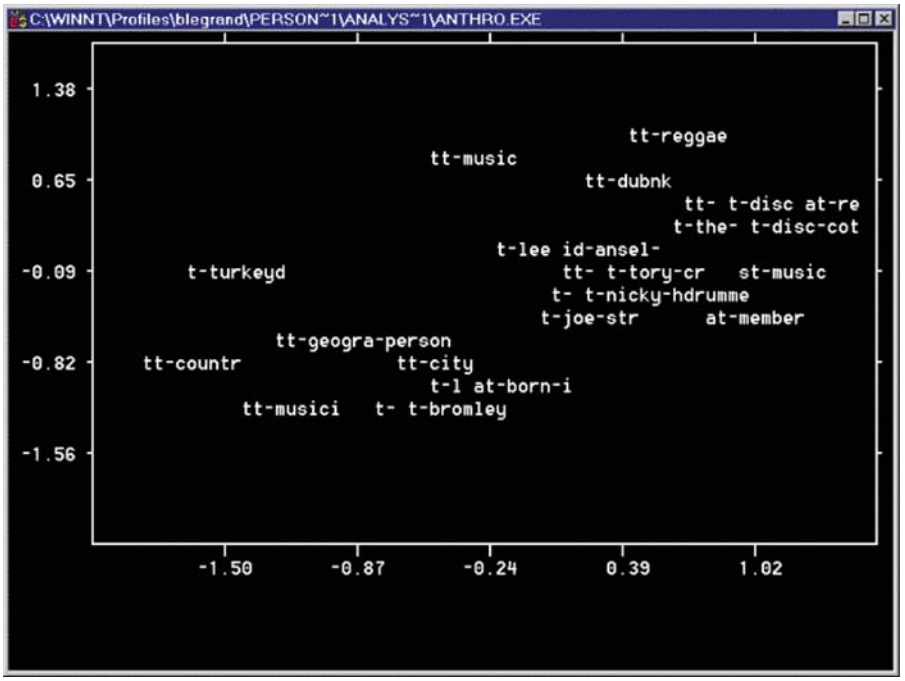


Figure 4.16 MDS map.

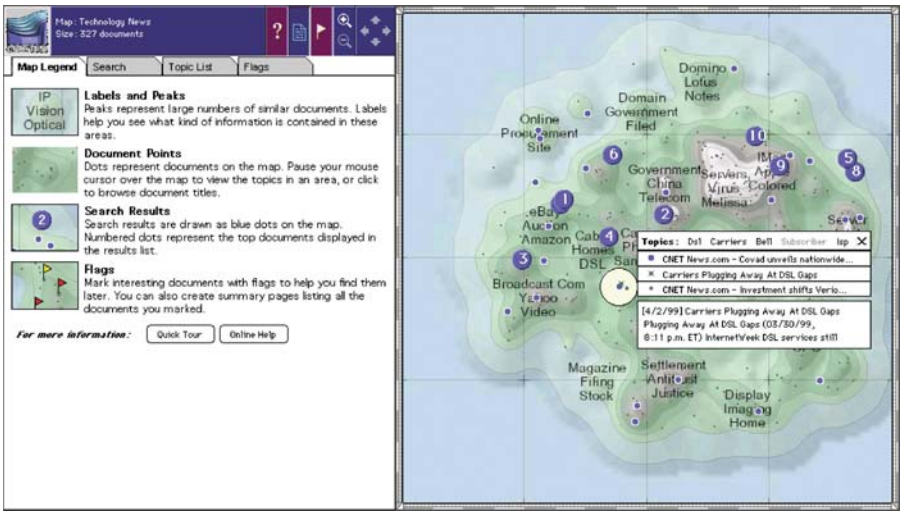


Figure 4.17 *ThemeScape* map.

different levels of magnification to declutter the map and reveal additional documents and labels.

This visualization is very interesting since it combines different representations in several windows. Users may choose one of them according to the selected type of information.

Visual data mining tools depict original data or resulting models using 3D visualizations, enabling users to interactively explore data and quickly discover meaningful new patterns, trends, and relationships.

Visual tools may utilize animated 3D landscapes that take advantage of human beings' ability to navigate in three-dimensional spaces, recognize patterns, track movement, and compare objects of different sizes and colors. Users may have complete control over the appearance of data. Virtual reality techniques include interactivity and the use of different levels of detail (LOD). Immersion in virtual worlds makes users feel more involved in the visualization.

A representation of Topic Maps as virtual cities, developed by Le Grand (2001), is shown in Figure 4.18. Topics are represented as buildings whose coordinates are computed from a matrix of similarities between topics. Users may navigate freely or follow a guided tour through the city; they may also choose to walk or fly. The properties of topics are symbolized by the characteristics of the corresponding buildings, such as name, color, height, width, depth, etc. Occurrences and associated topics are displayed in two windows at the bottom of the screen. As human beings are used to 2D, a traditional 2D map is also provided and the two views—the map and the virtual city—are always consistent.

Users may explore the world and interact with data. However, they may get lost in the virtual world. In order to avoid these problems, predefined navigation paths are also proposed. The different levels of detail make it possible to display many scales: details appear only when the user is close to the subject of interest.

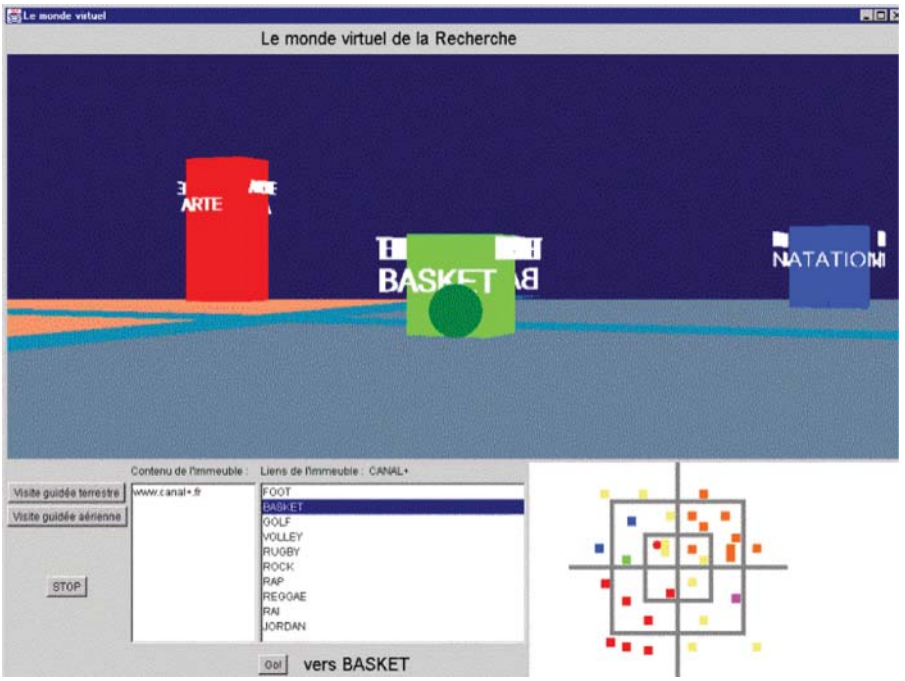


Figure 4.18 Virtual city and 2D map.

4.4 Conclusion and Perspectives

In this chapter, we presented Topic Maps, RDF, and ontologies basic concepts. These semantic structures may be complex, thus efficient visualization techniques are essential. We reviewed different types of visualization metaphors, especially textual interfaces, graphs, trees, maps, and virtual worlds. Some of them may represent efficiently the global structure while others are better at displaying details or providing interaction with data. In fact, each technique is well suited for a specific level of detail. The best way to benefit from the advantages of each method is thus to provide several levels of details for the representation of semantic graphs (Le Grand, 2001). This can be done by displaying several windows or by selecting the most appropriate representation at a given level of detail. Sometimes, the user needs a very low level of abstraction to capture detailed or accurate information. On the other hand, she will need a high level of abstraction to get general information from a semantic graph. Nevertheless, the appropriate levels of detail needed by the user may not always be provided from the structure of a given semantic graph, simply because they do not exist. In conclusion, the missing levels of details have to be provided by using external information, such as ontology, and by using classification techniques.

This chapter showed how one type of semantic graph—Topic Map, RDF graph, or ontology—could be visualized individually. We have seen that this is a difficult issue when the size of the semantic structure is important. At LIP6, we have started working on a visualization of combinations of such semantic graphs (e.g., the visualization of a

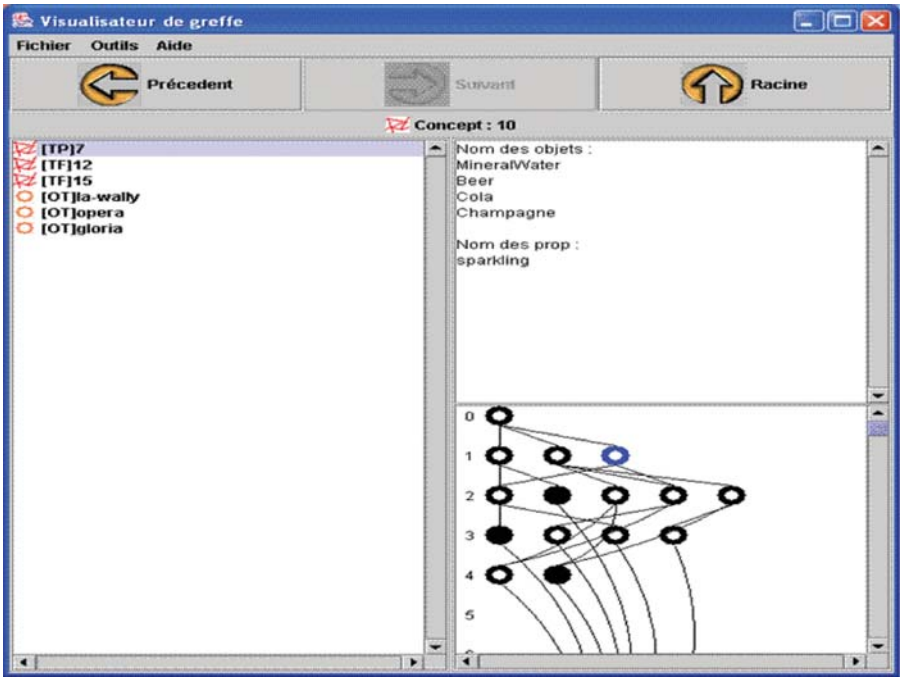


Figure 4.19 Combined Topic Map/ontology visualization.

Topic Map that has links to an ontology). This is a very likely situation, as well as RDF nodes linked to ontological concepts. Figure 4.19 is a screenshot of the first prototype. The Topic Map is mathematically analyzed with a conceptual classification technique and transformed into a Galois lattice. Each node of the lattice corresponds to a cluster of topics that are somewhat similar; the lower in the lattice, the more similar. When we click on a node of the lattice, the content of the cluster appears on the top-right window. Related nodes and related classes of the ontology are listed at the left side of the screen. It is possible to navigate transparently within the lattice or the ontology. On the lattice, the white circles indicate clusters that contain concepts that refer to an ontology. Black circles are clusters that contain standalone concepts. This interface is an early prototype; it will be enhanced so that the lattice structure remains implicit for the user. However, it already has interesting features such as the ability to see all concepts related to the same node of an ontology.

4.5 References

Alani, H. (2003). TGVizTab: An ontology visualisation extension for Protégé. In: *Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*, Sanibel Island, Florida, USA.

Barta, R., Garshol, L.M. (2003). Topic Map Query Language: Use cases, JTC1/SC 34. *ISO Project 18048.*

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American.*

Bruls, M., Huizing, K., van Wijk, J.J. (2000). Squarified treemaps. *Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization*, IEEE Press, pp. 33–42.

Cartia Inc. (1992). *ThemeScape Product Suite*, http://www.cartia.com/products/index.html.

Davison, M.L. (1992). *Multidimensional Scaling.* Malabar, Fl, CA: Krieger Publishing.
Empolis (2001). *K42, Intelligent Retrieval with Topic Maps*, http://www.empolis.com/englisch/pdf/-k42_eng.pdf.
Ernst, N.A., Storey, M.-A., Allen, P. (to appear). Cognitive support for ontology modeling. *International Journal on Human-Computer Studies*, IJHCS, http://www.neilernst.net/docs/pubs/ijhcs-protege.pdf.
Fluit, C., Sabou, M., van Harmelen, F. (2002). Ontology-based information visualization. *Proceedings of Information Visualization.*
Gansner, E.R., North, S.C. (1999). An open graph visualization system and its applications to software engineering. *Software: Practise and Experience*, 00(S1):1–5.
Gershon, N., Eick, S.G. (1995). Visualisation's new tack: Making sense of information. *IEEE Spectrum*, pp. 38–56.
Gruber, T.G. (1993). Toward principles for the design of ontologies used for knowledge sharing. In: Guarino, N., Poli, R. (Eds.), *International Workshop on Formal Ontology*, Padova, Italy.
Heflin, J. (2004). OWL Web ontology language use cases and requirements. *W3C Recommendation.*
International Organisation for Standardization (ISO), International Electrotechnical Commission (IEC), *Topic Maps, International Standard ISO/IEC 13250:1999*, April 19, 1999.
Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M. (2002). RQL: A declarative query language for RDF. *WWW2002*, May 7–11, Honolulu, Hawaii, USA. ACM 1-58113-449-5/02/0005.
Kaski, S., Honkela, T., Lagus, K., Kohonen, T. (1998). WEBSOM: Self-organising maps of document collections. *Neurocomputing*, 21:101–117.
Le Grand, B., Soto, M. (2000). Information management: Topic Maps visualization. *XML Europe 2000*, Paris, France.
Le Grand, B., Soto, M. (2001). XML Topic Maps and Semantic Web mining. *Semantic Web Mining Workshop, ECML/PKDD 2001 Conference*, Friburg, Germany.
Le Grand, B. (2001). Extraction d'information et visualisation de systèmes complexes sémantiquement structurés, thesis (PhD), *Laboratoire d'Informatique de Paris 6*, Paris, France.
Mondeca (2001). *Topic Navigator*, http://www.mondeca.com/site/products/products.html.
Moore, G. (2001). RDF and Topic Maps: An exercise in convergence. *XML Europe 2001*, Berlin, Germany.
Munzner, T. (1997). H3: Laying out large directed graphs in 3D hyperbolic space. *IEEE Symposium on Information Visualization.*
Mutton, P., Golback, J. (2003). Visualisation of semantic metadata and ontologies. *Proceedings of Information Visualization*, July 16–18, 2003, London, UK.
Noy, N.F., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A. (2001). Creating Semantic Web contents with Protégé 2000. *IEEE Intelligent Systems*, 16(2):60–71.
Nvision Software Systems Inc. (2000). *NV3D Technical Capabilities Overview*, http://www.nv3d.com/html/tco.pdf
Ontopia (2001). *Ontopia Topic Map Navigator*, http://www.ontopia.net/solutions/navigator.html.
Pietriga, E. (2002). IsAViz: A visual environment for browsing and authoring RDF models. *Eleventh International World-Wide-Web Conference Developers Day.*
Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. *Proceedings of IEEE Visual Languages*, Boulder, CO, pp. 336–343.
TheBrain Technologies Corp. (2001). Web, http://www.thebrain.com.
Techquila (2001). *TM4J: A Topic Map Engine for Java*, http://www.techquila.com.
TopicMaps.Org XTM Authoring Group (2001). XTM: XML Topic Maps (XTM) 1.0. *TopicMaps.Org Specification.*
World Wide Web Consortium (1999). *Resource Description Framework (RDF) Model and Syntax Specification.* W3C.
Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, S., Maler, E. (2004). Extensible Markup Language (XML) 1.0 (3rd Edition). *W3C Recommendation.*

Chapter 5

Web Services: Description, Interfaces, and Ontology

Alexander Nakhimovsky and Tom Myers

5.1 Introduction

In this chapter, we would like to draw a comparison between Semantic Web and Web Services, in the hope that such a comparison will shed light on each and help us understand the directions of their evolution. Both Semantic Web and Web Services are young, rapidly changing technologies that overlap in significant ways. A better understanding on how they are similar and where they differ may add clarity to their development and mutual accommodation. If, as we believe is the case, they have complementary strengths and weaknesses, their mutual understanding can be of help to both.

Since the publication of the first edition, the underpinnings of Web Services technology have changed slightly, but the available Semantic Web technology has changed a great deal. This chapter will therefore change mostly in those sections that indicate the areas of Web Services that dovetail with Semantic Web languages and tools.

We start with a high-level juxtaposition drawn in broad strokes, unavoidably inaccurate in small details. Small details will follow soon thereafter.

5.2 Semantic Web and Web Services: A Comparison

Semantic Web and Web Services share a vision. Both would like to see the Web as one giant information store operated upon by loosely coupled cooperating applications. Both see XML as the main tool for making multiple pieces of the Web interoperable. A major difference between them is that Semantic Web thinks of the Web as a repository of knowledge, while Web Services think of it as a marketplace. Hence more emphasis in Semantic Web on knowledge representation and management, and more emphasis in Web Services on an actual action (frequently, but not necessarily a commercial transaction) that brings about a change in the world. While Semantic Web's metaphor for the Web is an encyclopedia, the Web Services' metaphor is a telephone book where you or your agent can find a number to call and get service.

A useful analogy from earlier programming languages and systems would be Prolog versus C. Semantic Web would like to see the Web as a Prolog-style database and an inference engine, while Web Services would like to see it as a more traditional information store and a distributed C/Java/C# application. Semantic Web's roots are in AI: declarative knowledge representation and inference. Web Services' roots are in distributed programming: an introduction to Web Services frequently starts with a

brief review of RPC, DCOM, CORBA, and RMI. Sociologically, too, the worlds of Semantic Web and Web Services are quite different: people working on Semantic Web frequently come from the "symbolic AI" tradition and bring to Semantic Web the ideas of knowledge representation, planning, and logic programming, while people working on Web Services are mostly professional programmers working for software companies. The Semantic Web worker tries to implement a vision. The Web Services worker has to deliver a product that fills a real or perceived need.

Any comparison between Semantic Web and Web Services must recognize that Web Services are much closer to "the metal and wire" of the Web than Semantic Web, with such items as ports, protocol bindings, and patterns of message exchange explicitly represented in software. This may point to a possible synergy: just as C is a level or two closer to the iron than Prolog, and Prolog interpreters can be and have been implemented in C, so it might be useful to think of Semantic Web as being a level or two above Web Services, composed of meaning units that are built out of Web Services.

Another thing to recognize is that in practical terms, Web Services are way ahead of Semantic Web and moving ahead faster. Web Services are backed up by huge investments and hundreds if not thousands of developers while Semantic Web is backed up by a grant from DARPA, a committee at W3C, and a relatively small group of mostly academic researchers. Web Services are designed so that, with relatively little additional training, currently existing and widespread programming skills can be redeployed for Web Services programming. By contrast, to create a Semantic Web application, you need relatively exotic programming skills and a background in AI and logic. It does not help that the RDF/XML syntax of the initial RDF Model and Syntax Recommendation is arguably the worst formal notation ever invented.

As a result, Web Services are rapidly creating a set of conditions on the ground that Semantic Web will have to contend with. It seems likely that Semantic Web will come into existence within an infrastructure created by Web Services. The strength of Semantic Web is in its foundations in logic and AI and a rigorous approach to software design. On the other hand, it should be remembered that Prolog never escaped a niche status, and the entire enterprise of symbolic AI has yielded relatively little in terms of working systems and market success.

5.3 Web Services Definition and Description Layers

A Web Service, most generally, is a distributed self-describing Web application that uses an XML communication protocol and can be discovered and invoked on the basis of its self-description. More specifically, a Web Service uses

- Simple Object Access Protocol (SOAP) for communication and elements of self-description
- Web Services Description Language (WSDL) for a low-level description of SOAP connectivity and interfaces
- Universal Description, Discovery and Integration language (UDDI) for a higher-level description of functionality

Although WSDL professes protocol-independence, and defines a SOAP binding as one of several possible bindings, in practice SOAP is firmly entrenched as the best-developed and most important layer. It is also closest to becoming a standard.

5.3.1 Standardization Efforts

One commonly encounters in Web Services literature or presentations assertions to the effect that Web Services are "based on standards," meaning SOAP, WSDL, and UDDI. This is somewhat misleading. Of the three specifications, only SOAP and WSDL have been published as Recommendations by W3C (see SOAP 2003-1,-2,-3 and WSDL 2004-1,-2,-3). W3C is not a standards body, so its Recommendations are not, strictly speaking, standards, but for all intents and purposes they are, or at least have been until now. In addition, SOAP activity is wrapped into larger XML Protocol (XMLP) activity that is developing a context for, and extensions of SOAP, such as Security and Quality of Service. (See XMLP 2004.) As for UDDI (UDDI 2003), it does not have any connection to W3C or any standards body at all. Its only claim to standardhood is that it is supported by a very powerful industry consortium.

5.3.2 The Significance of SOAP

Technically, perhaps the strongest difference between Semantic Web and Web Services is that Web Services are built on top of their own communication protocol. The significance of a separate, XML-based communications protocol is that it provides a platform-independent medium for describing APIs, specifying meta-level constraints (such as: this service must "understand" such-and-such a procedure call), and delivering level-specific error codes, as opposed to generic lower-level error codes. On the other hand, a separate protocol with its own rules of engagement between clients and servers creates a danger that Web Services will develop into an alternative Web, a Web of SOAP servers and clients that has little or no connection to the HTML/HTTP Web; even the human front end to it does not have to be a Web browser at all, and may, in fact, be platform-specific. We will return to this topic after we learn more about SOAP.

5.4 SOAP in Greater Detail

The current version of SOAP is WD 1.2. Compared to version 1.1, the specification is broken in two parts: the essential SOAP (Part 1) and SOAP Adjuncts (Part 2); there is also a Primer in Part 0. We concentrate on Part 1, the Messaging Framework.

5.4.1 SOAP Message

Appropriately for a communication protocol, the main concept of SOAP is a *message*. A SOAP message is a one-way transmission from a SOAP sender to a SOAP receiver. However, SOAP messages can be combined to implement various Message Exchange Patterns (MEPs) such as request/response or multicast. A SOAP message can also travel from the message originator to its final destination via intermediaries that can simply pass the message on or process it and emit a modified message or a fault condition. *SOAP node* is a general name for the initial SOAP sender, the ultimate SOAP receiver, or a SOAP intermediary (which is both a SOAP sender and a SOAP receiver). Ultimately, a Web Service is a collection of SOAP nodes.

Within this picture, the main questions addressed by the specification are:

- What is the message structure as an XML document?
- How do the elements of the message correspond to programming objects that are manipulated by SOAP Nodes? (SOAP encoding)
- What is the underlying transport that actually delivers the message?
- What happens to the message along the way and at destination? (Message Exchange Model)
- How does SOAP report its own error conditions (called Faults)?

Our main interest is in message structure, but we briefly comment on the other parts of the specification here and in the context of SOAP examples.

As regards **encoding**, the biggest change between SOAP versions 1.1 and 1.2 is that SOAP encoding has been taken out of the main body of the specification and placed into Part 2, the Adjuncts. The intent is to indicate that it is ultimately up to individual applications and messages to decide on the details of XML encoding and data binding. The encoding used in our examples is the one specified in SOAP 1.1 and SOAP 1.2 Part 2.

As regards the **underlying transport**, all our examples will use HTTP. This means that the SOAP message will travel as the body of an HTTP message, using the HTTP POST method. In principle, SOAP can be used with a variety of protocols; in practice, HTTP is by far the most common, with SMPT a distant second.

The **Message Exchange Model** is mostly concerned with how individual elements of the message are processed at intermediary SOAP nodes along the path from origin to final destination. The processing is controlled by meta-information attached to such elements. The meta-information is expressed as the element's attributes, *actor* and *mustUnderstand*.

Fault elements are for specifying SOAP-level error or status conditions. A fault element must contain a fault code and a message string. Fault codes are qualified names as specified in XML Namespaces: they consist of a namespace URI and a local name. SOAP 1.2 defines a small set of SOAP fault codes covering basic SOAP faults. We will show examples of fault elements in our code.

5.4.2 The Structure of a SOAP Message

A SOAP message is an XML document and as such has an Infoset. (Infoset is a W3C specification that defines the object model of XML documents.) The SOAP specification defines the structure of SOAP messages in Infoset terms rather than in the more traditional way of defining the syntactic structure of the serialized XML document. Since Infoset is relatively recent and less well-known that the syntactic notions, we will recast the definition in syntactic terms.

The root element of a SOAP message is *Envelope*, in the SOAP-Envelope namespace (http://www.w3.org/2003/05/soap-envelope). Three other elements are declared in that namespace: *Header*, *Body,* and *Fault.* Several attributes are also defined in that namespace, including *encodingStyle,* which specifies the SOAP encoding to be used in processing the message.

The Header and Body elements are children of the root, Header optional, Body required. Both Header and Body consist of *blocks*, that is, children elements in their own namespace(s), with no constraints on their internal structure. The content of the Body, sometimes called the payload, is intended to be processed by the message's

final destination. The content of the optional Header may be intended for intermediate nodes, with each Header block targeted individually, as specified in the Message Exchange Model. In general, Header blocks serve for application-specific extensions. We do not use headers in our examples.

5.4.3 Examples of SOAP Messages

Suppose we want to create a very simple Web Service that counts the number of times a given pattern occurs within a given string. For instance, if the pattern is *an* and the string is *ananas*, the service will return the integer 2. The action of such a service can be expressed in Java as follows:

Listing 5.1. Java class for counting patterns in strings

```
public class CountPat{
 public int count(String string,String pat){
  int cnt=0,pos=string.indexOf(pat);
  while(pos >= 0){
   cnt++;
   pos=string.indexOf(pat,pos+pat.length());
  }
  return cnt;
 }
}
```

As defined in Listing 5.1, the service will count only nonoverlapping occurrences of the pattern: if the pattern is *aa* and the string is *aaa*, the service will return 1, not 2. To count all occurrences, we would replace the line in boldface with the one below:

```
  pos=string.indexOf(pat,pos+1);
```

This is not the most efficient way to search strings for patterns, but we are not interested in algorithmic efficiency: we are interested in how and where the difference in the semantics of the service can be expressed. We will return to this question in the end of the chapter. In the meantime, we note that in either case the service expects two inputs, both of them strings, and returns an integer. A SOAP message addressed to such a service would look like this:

Listing 5.2. SOAP request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle =
  "http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENV =
  "http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance"
>
 <SOAP-ENV:Body>
```

```
  <count>
   <arg0 xsi:type="xsd:string">
   alphabetaalphagammaalpha
   </arg0>
   <arg1 xsi:type="xsd:string">
   alph
   </arg1>
  </count>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The root element uses the global *encodingStyle* attribute to specify which encoding to use. The value of the attribute is a URI that serves as an identifier of the intended encoding. The encoding specified in our example is the encoding that is defined in SOAP 1.1 and SOAP 1.2 Part 2. The *encodingStyle* attribute is in the same namespace as the Envelope and Body elements. In addition to that namespace (mapped to *SOAP-ENV*), two others are declared, mapped to *xsi* and *xsd*. Both come from the XML Schema Recommendation and have to do with assigning data types to element content of SOAP messages. Our message Body consists of a single block whose name is the name of the remote procedure to invoke. The block consists of two elements that are eventually converted to arguments of the remote procedure call.

As we mentioned, this message will travel from origin to destination over HTTP, as the body of an HTTP request. The header of the HTTP request looks like this:

Listing 5.3. HTTP request whose body is SOAP request of Listing 5.2

```
POST/axis/CountPat.jws HTTP/1.0
Content-Length: 467
Host: localhost
Content-Type: text/xml; charset=utf-8
SOAPAction: "/count"
```

As you can see, this is standard HTTP, with one new header, SOAPAction, added for SOAP purposes. It specifies the programmatic action to be invoked on the SOAP server in response to the message. The generated HTTP response, containing SOAP response in its body, is as follows (this time we show the entire HTTP message all together):

Listing 5.4. HTTP response containing SOAP response as its body

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 427
Date: Sun, 06 Jan 2002 16:52:50 GMT
Server: Apache Tomcat/4.0.1-b1 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle=
  "http://schemas.xmlsoap.org/soap/encoding/"
```

```
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <countResponse>
  <countResult xsi:type="xsd:int">3</countResult>
  </countResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Now we need a SOAP client that is capable of sending out a message as in Listings 5.3 and 5.2 and interpreting the response as in Listing 5.4. Such a client can be constructed within the same framework that is used for deploying the SOAP server, or in a totally different one. In any case, there is a great deal of support for client construction built into the framework (just as there is a great deal of support for server construction). The framework support can be framework-dependent, or it can be based on the framework-independent WSDL description of the service. It is time to look at how SOAP servers and clients are programmed.

5.5 What Is It Like for a Programmer?

It is essential to keep in mind the pragmatic or sociological aspect of Web Services: they are distributed applications consisting of modules that are familiar programming modules: Java/C++/C# classes, COM objects, and so on. All the major players in the "Web Services space" have invested a lot of thought and effort into making Web Services programming easy for a programmer. In particular, a lot of attention has been paid to the question of how familiar software modules within a client-server framework can become a SOAP server and how one generates a SOAP client for it. The goal is to automate as much of this process as possible. We present the tools of the open-source Apache Axis project (Axis 2004). Comparable facilities have been for some time available within IBM's Web Services Development Toolkit, which has been extended and renamed as the IBM Emerging Technologies Toolkit (WSTK/EETK 2003) and Microsoft's .NET framework (see MS .NET 2004). Our first task is to describe the directory structure of an Axis installation, which (in the Java version, see Axis 2004) has just moved from beta to RC-1 but with little change in the introductory material we cover.

5.5.1 Axis SOAP Server and Tomcat Servlet Engine

A SOAP server can be implemented as a J2EE Web Application. In other words, a Web Service can be implemented as a Java servlet or a JavaServer Page (JSP). To run servlets and JSPs, one needs a servlet/JSP engine attached to a Web server. If the incoming HTTP request is addressed to a servlet or JSP, the server will redirect it to the servlet/JSP engine, which will generate a response to be sent to the client.

Apache Tomcat is a Web server and a servlet/JSP engine rolled into one piece of software. (It can also be attached as a servlet/JSP engine to a different server.) By default, the Tomcat server runs on port 8080. The directory in which Tomcat is installed (call

it TOMCAT_HOME) has a subdirectory called webapps. This directory is the server root of the Tomcat Web server. It is in this directory that we install Axis, in a directory called axis. Every Web Service (or SOAP server) run by Axis will be a subdirectory of TOMCAT_HOME/webapps/axis. If you look back in Listing 5.3, the first line specifies `/axis/CountPat.jws` as the destination of the SOAP message. The extension `.jws` stands for "Java Web Service." So how does the Java code of Listing 5.1 in file CountPat.java become a SOAP server in file CountPat.jws?

5.5.2 From Java Class to SOAP Server

There are three ways to convert a Java class into a SOAP server within the Axis framework. The simplest but least flexible one goes like this:

Step 1. Make a copy of CountPat.java and rename it as CountPat.jws.
Step 2. Place CountPat.jws into TOMCAT_HOME/webapps/axis/ directory.
Step 3. There is no step 3; you are done.

A second, more flexible and powerful way of configuring a Web Service within Axis is by using an XML file in the Axis Web Services Deployment Descriptor (WSDD) format. The details of WSDD are outside the scope of this chapter but can be found in the Axis User Guide.

The third way of constructing a Web Service from a Java class is via the intermediate stage of a WSDL description of the service. We will discuss WSDL shortly.

5.5.3 Constructing a SOAP Client

To invoke the service deployed as described in the preceding section, we need to send an HTTP request whose header is Listing 5.3 and body is the SOAP message in Listing 5.2. Within Axis, we construct a client using classes within the `org.apache.axis.client` package. Here is a client for the CountPat service, implemented as a JSP. The punchline is the call of the invoke() method of the ServiceClient class that invokes the service. The method takes three arguments, the namespace of the method element (the empty string in our case), the local name of the method and its arguments as an array of Objects:

Listing 5.5. JSP SOAP client for the CountPat service

```
<%
String method = request.getParameter("method");
String arg1 = request.getParameter("arg1");
String arg2 = request.getParameter("arg2");
String endpoint = "http://localhost:8080/axis/CountPat.jws";

org.apache.axis.client.ServiceClient client =
 new org.apache.axis.client.ServiceClient(endpoint);

try {
  Object[] args = new Object [] { arg1, arg2 };
  Object obResult = client.invoke("",method,args);
%>
```

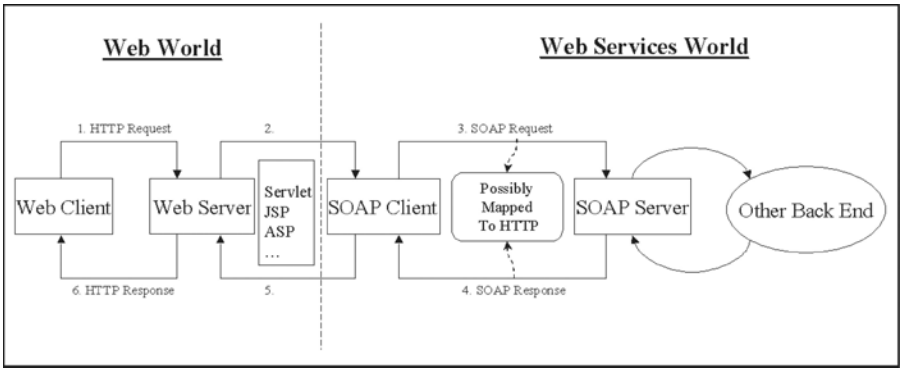


Figure 5.1 Software configuration for running a Web Service.

```
<html><body><h3>
 The result is: <%= obResult.toString() %>
</h3></body></html>
<%
  }catch(Exception ex){/* appropriate message */}
%>
```

The client is itself a Web application invoked from an HTML form with three input fields for the method name and its two parameters. The entire configuration of software is shown in Figure 5.1.

In the configuration of Figure 5.1, the GUI and the SOAP client do not have to be on separate machines. A SOAP client is a fairly lightweight piece of software that can easily be outfitted with a GUI of its own, such as a WindowsForm of the .NET framework. When this happens, the web of distributed SOAP applications may become rather tenuously connected to the HTTP/HTML Web of HTML documents.

While the code of our SOAP client is quite transparent and easy to compose, its creation, just as the creation of the SOAP server, can be further automated by first obtaining the WSDL description of the service. This is the subject of the next section.

5.6 WSDL

WSDL is an XML language for describing Web Services. It is, in effect, an interface definition language for SOAP. Its design goals, from the beginning, included automation tasks, such as:

- Generate a WSDL description of a service from its implementation in a programming language.
- Generate a SOAP service from its WSDL description.
- In the case of request-response services, generate a client from the WSDL description of the server.

The Web Services toolkits and frameworks from Microsoft, IBM, and Apache all implement these capabilities. We will continue using Axis for our examples. As we illustrate the structure of a WSDL document with samples of generated code, keep in mind that the URI of our CountPat service is http://localhost:8080/axis/CountPat.jws, the name of the class is CountPat, and the name of the operation supported by the service is count. While the document structure is, of course, toolkit-independent, some of the naming conventions are those of Axis.

5.6.1 WSDL Document Structure and Examples

WSDL is primarily designed to describe exchange of typed messages between two endpoints, in effect, remote procedure calls. A partial list of elements defined in WSDL is shown below, in the order of appearance, with examples from generated code.

The root element is `definitions`. It declares several namespaces, including a generated namespace for service-specific elements, which is also the target namespace:

Listing 5.6. The root element and namespaces

```
<definitions
  targetNamespace="http://localhost:8080/axis/CountPat.jws"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:serviceNS="http://localhost:8080/axis/CountPat.jws"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Elements of type `message` describe requests and responses. In our example, we have:

Listing 5.7. Message elements

```
<message name="countResponse">
 <part name="countResult" type="xsd:int"/>
</message>
<message name="countRequest">
 <part name="arg0" type="xsd:string"/>
 <part name="arg1" type="xsd:string"/>
</message>
```

As you can see, message elements have children `parts` that describe the data types of arguments and returned values. The data types are those of XML Schema. If application-specific data types are defined, a `types` element would precede message elements.

Elements of type `operation` describe operations supported by the message. They appear grouped within a port type element. Our example supports a single operation:

Listing 5.8. Port type and operations

```
<portType name="CountPatPortType">
 <operation name="count">
  <input message="serviceNS:countRequest"/>
  <output message="serviceNS:countResponse"/>
 </operation>
</portType>
```

Next, a `binding` element specifies the concrete protocol (SOAP) and provides protocol-specific details about the operation(s) supported by the service:

Listing 5.9. Binding to SOAP

```
<binding
  name="CountPatSoapBinding"
  type="serviceNS:CountPatPortType">
 <soap:binding
  style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="count">
  <soap:operation soapAction="" style="rpc"/>
  <input>
   <soap:body
   encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/"
   namespace="" use="encoded"/>
  </input>
  <output>
   <!-- identical to input -->
  </output>
 </operation>
</binding>
```

Finally, `port` is a single endpoint defined as a combination of a binding and a network address, and `service` is a collection of related endpoints.

Listing 5.10. Service and port(s)

```
<service name="CountPat">
  <port
   binding="serviceNS:CountPatSoapBinding"
   name="CountPatPort">
  <soap:address
   location=
    "http://localhost:8080/axis/CountPat.jws"/>
  </port>
 </service>
</definitions>
```

To obtain its automatically generated WSDL description we add the query string ?wsdl to the service URI: http://localhost:8080/axis/CountPat.jws?wsdl. The output, in outline and with references to Listings 5.6 through 5.10, is in Listing 5.11:

Listing 5.11. WSDL description of CountPat service

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
 namespace declarations see Listing 5.6
>
 <message name="countResponse">...</message>
 <message name="countRequest">...</message>
see Listing 5.7

 <portType name="CountPatPortType">...</portType>
 see Listing 5.8

 <binding name="..." type="...">
 see Listing 5.9
 </binding>

 <service name="CountPat">
 see Listing 5.10
  </service>
 </definitions>
```

5.6.2 Options and Alternatives

The generated WSDL of Listings 5.6 to 5.11 shows a WSDL description of a service that implements a remote procedure call using SOAP as the protocol and XML Schema data types for the call signature. These are not the only options, as the WSDL specification makes clear:

- WSDL can describe message exchange patterns other than remote procedure call.
- In addition to SOAP, two other protocols are provided for, plain HTTP and MIME.
- Data type libraries other than XML Schema Part 2 can be used.

However, the configuration of our example is by far the most common and fits the specification most naturally.

5.6.3 What Can One Do with WSDL?

Axis and other toolkits provide tools for automatic code generation from WSDL descriptions. In Axis, the tools are in the Wsdl2java class within the org.apache.axis.wsdl package. Here is a command line to build a stub for the service client:

```
java -cp wsdl4j.jar;axis.jar;clutil.jar;C:\cp\xerces.jar
   org.apache.axis.wsdl.Wsdl2java
   --verbose
  http://localhost:8080/axis/CountPat.jws?wsdl
```

This instructs the Java Virtual Machine to run Wsdl2java on our WSDL, in verbose mode, using several libraries including Xerces XML parser. The output (in verbose mode) is:

```
Parsing XML File: http://localhost:8080/axis/CountPat.jws?wsdl
Generating portType interface: CountPatPortType.java
Generating client-side stub: CountPatSoapBindingStub.java
Generating service class: CountPat.java
```

The three Java files are indeed generated and the "client-side stub" contains a definition of the public count() method that takes two strings and returns an integer. The method, in turn, calls the invoke() method to invoke the service. Of course, the generated stub does not provide any mechanism for passing parameters and it has no idea what you would like to do with the returned result, but it can serve as a useful starting point. The WSDL description of a Web Service implemented in Java (and perhaps automatically generated from a Java class) can also be used to generate a SOAP client on a .NET platform. Finally, WSDL can be used to generate a starting point for SOAP servers, also in any framework, on either Java or .NET platform. In Axis, you would simply insert the—skeleton flag next to—verbose.

We can conclude that WSDL is a very useful XML language for describing SOAP-based Web Services and automating several aspects of Web Services programming. We can also conclude that it contains no facilities for describing the intended semantics of the service.

5.7 UDDI

UDDI, like WSDL, is a layer of description built on top of SOAP. While WSDL is an automation tool that provides a low-level XML description of SOAP-based services, UDDI is intended for more high-level tasks: service discovery and integration. At the center of UDDI is the notion of a registry that would provide another XML description of a service, based on business semantics. The UDDI specification does not mention WSDL at all, but within the UDDI description of a service it provides an element, tModel, for a technical specification of the service interfaces and bindings. This is where WSDL fits into the UDDI view of the world; Colgrave and Januszewski (2004) provides extensive material on how to use WSDL for tModel descriptions.

5.7.1 Components of a UDDI Entry

A UDDI entry describes a *business entity*, such as a firm or a nonprofit organization. Each business entity can support a number of *business services*. Each service can be offered in a variety of binding templates that describe where to find a particular implementation of the service and how to connect to it. Each binding template contains a reference to a *tModel*, the technical description of the service's interface.

The XML Schema that defines UDDI entry (UDDI v3 Schema 2003) specifies the following containments:

- A *businessEntity* element contains a *businessServices* element that in turn contains any number of *businessService* elements.
- A *businessService* element contains a *bindingTemplates* element that contains any number of *bindingTemplate* elements.
- Each *bindingTemplate* contains a *tModelInstanceDetails* that contains *tModel* elements, each of which contains an *overviewDoc* element that contains an *overviewURL*. This is where a reference to a WSDL file would be inserted.

In addition to these other substantive elements, most UDDI elements have a key for easy retrieval and a name for easy reference.

5.7.2 UDDI and WSDL

The way WSDL descriptions find their way into UDDI entries can be summarized as follows (see Colgrave and Januszewski, 2003 for more detail):

- Industry groups define a set of service types, and describe them with one or more WSDL documents. The WSDL service interface definitions are then registered as UDDI tModels; the *overviewDoc* field in each new tModel points to the corresponding WSDL document.
- Programmers build services that conform to the industry standard service definitions. In essence, they retrieve the WSDL definitions using the URL in the overviewDoc element and construct services on the basis of WSDL as described in the preceding section of this chapter.
- The new service is entered into the UDDI registry as a businessService element within a businessEntity. Typically, some kind of a deployment descriptor can be generated at the same time. Much of this work can and will be automated by UDDI tools.

Our next question is: "Where in this process is the semantics of a Web Service defined?" As we have seen, WSDL has nothing to say about semantics. UDDI provides, on top of WSDL, three service taxonomies:

1. Services classified by industry, using the North-American Industry Classification System (NAICS; see http://www.ntis.gov/product/naics.htm)
2. Services classified by product or service, using the Universal Standard Products and Services Classification (UNSPSC; see http://eccma.org/unspsc/)
3. Services classified by location, using ISO 3166 (see http://www.din.de/gremien/nas/nabd/iso3166ma/)

Additional taxonomies, such as those by Yahoo or Google, can be registered with UDDI and used to classify services. These taxonomic hierarchies constitute a simple semantical characterization of UDDI-registered services. However, as of UDDI 3.01, there are no inference rules associated with them; nor is there an agreed-upon syntax for writing down such rules for use by automatic agents. This is where Semantic Web initiatives may find an important role to play. The languages of the Semantic Web, RDF (Resource Description Framework) and OWL (a scrambled acronym for Web Ontology Language), have been designed precisely for the purpose of providing descriptions of Web resources, organized into hierarchies within an agreed-upon shared ontology, for use by cooperating automatic agents. Specifications for these languages have been released as W3C recommendations. OWL, or at least its OWL Lite subset, is supported by open-source inference engines including Jena (JENA 2004). OWL-S, based on OWL, is specifically an ontology for Web Services, defined within the Semantic Web community: "OWL-S is a OWL-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form." (See http://www.daml.org/services/owl-s/1.1.)

5.7.3 Semantical and Ontological Needs

What does it mean to "provide an ontology" for describing Web Services? The word *ontology* has been much used lately, with different intended meanings. In this case, the authors mean providing a language such that we can logically prove statements about its lexical items (which may include names of functions and relations). Such languages usually can describe both individual facts and general rules, and the language processor (also known as an *inference engine*) would know how to combine facts and rules to infer new facts. For instance, consider this formal notation:

Listing 5.12. A logical notation

```
father(abraham,isaac); // Abraham is the father of Isaac
father(isaac,esau); // Isaac is the father of Esau
father(X,Y),father(Y,Z):-grandfather(X,Z)
  // X is the father of Y and Y is the father of Z
  // implies X is grandfather of Z
```

On the basis of this information, an inference engine would be able to infer (and add to the database of facts) the fact that Abraham is grandfather of Esau. In a similar fashion, if two Web Services, `s1` and `s2`, offer the same widget `w` for prices `p1` and p2 respectively, and p1 < p2, an inference engine would infer that, on some scale of preference, service s1 is preferable to s2.

This information can, of course, be expressed by a piece of code in a program somewhere that, for instance, subtracts the price, multiplied by a weight factor, from the preference metric so that lower price gets higher preference:

```
s1.preference -= s1.price(w)*PRICE_FACTOR;
```

Voluminous discussions of declarative versus procedural semantics, both within AI and within the theory of programming languages, have documented why it is in many respects better to express a rule as a declarative statement in a specially designed language than as a line of code in a program. (See, e.g., Date, 2000.) Many years of practice have also demonstrated that it is very difficult to design ontologies that are expressive enough to describe nontrivial programming tasks. Even the simple distinction between counting overlapping versus nonoverlapping patterns in our CountPat service would take a substantial amount of effort to express in a rigorous way, using some algebraic notation, even though CountPat is a very simple Web Service that involves a single request-response in which only integers and strings are exchanged.

Can we indeed deliver a language for describing Web Services such that it will be possible for automatic agents to reason about them in reliably rigorous ways? Yes, certainly. Can we make it practical to do so? That is not clear at all. The amount of work that would be required is staggering and the track record of similar endeavors is not very encouraging. However, as technical committees sit down to formulate industry-specific languages for describing UDDI Web Service entries, they must be aware of the semantic distinctions that those languages have to express. To the extent that those languages are capable of expressing those distinctions, the world of Web Services will be capable of moving closer to the vision of the Semantic Web. To the extent that those languages can conveniently be used to express useful distinctions (not necessarily all of the required distinctions, just a useful subset), the world of Web Services will actually move closer to that vision. In the meantime, the core of

Web Services description is now WSDL and UDDI. We will consider UDDI, and in particular its tModels.

5.7.3.1 TModel Summary

The beauty of UDDI tModel usage is that any given item (business, Web Service, whatever) is located within multiple hierarchies, usually fairly shallow ones that may or may not relate to one another logically. A particular organic farm's Web site services might be found within a hierarchy of agricultural entities, a hierarchy of environmental-interest sites, a hierarchy of health-related information, and simultaneously a hierarchy of geographical areas. Even our simple example of string-matching can similarly be found in a collection of hierarchies. We can classify it according to

a. The kind of pattern language: strings, regular expressions, context-free grammars, or programming languages such as perl
b. The kind of input: plain ASCII text, a variety of text encodings, text found within markup (HTML, XML, Postscript, etc.)
c. The kind of output: match-counts, match-locations, strings matched with or without context, structure
d. System requirements (perl implementation, Windows-only, less than a gigabyte . . .)

This gives us a collection of loosely related hierarchies. Almost everything can be usefully described with such descriptors. Unfortunately, they will probably not be standardized hierarchies applicable to a wide range of systems and situations, and we are not likely to see an international standard for hierarchies describing our Web Service. It is possible that an actual UDDI library will (or already does) describe some pattern-matching services within tModels, in a way that could be used by creators of a new service, but we don't expect it. In our experience UDDI is more of a heavyweight system, too much work for idiosyncratic usages like this. We want to be able to create these casual hierarchies on the fly, compose them with other hierarchies already known, and reason about them. We want the Green Rabbit organic farm, if it ever offers Web Services, to be able to classify those services in novel ways and to combine them with related services. The Semantic Web toolkit has the potential to make this possible. In the next subsection, we will very briefly summarize that toolkit: RDF, RDFS (RDF Schema), and OWL.

5.7.3.2 Semantic Web: Simple Assertions

The W3C description of RDF, http://www.w3.org/RDF/, is quite complex. We share the distaste voiced by many for the RDF/XML syntax and we will use the earlier N3 syntax from http://www.w3.org/DesignIssues/Notation3.html for this chapter, but RDF/XML serves the purpose of getting RDF data into the XML processors, which are now found all over, including every modern browser. Mozilla has long supported RDF directly, as "a common data model and API" (http://www.mozilla.org/rdf/doc/). The increasingly popular Mozilla spinoffs, Firefox and Thunderbird, depend upon it for defining extensions. The critical idea, from our point of view, is that we write data in the form of collections of assertions, each of which has three parts:

```
John loves Mary
John isTallerThan Mary
Mary loves Joe
Mary isTallerThan Joe
```

From the programmer's viewpoint, this is just a 3-column table, or perhaps a graph where the first and last columns are labels for nodes and the middle column labels an arc. We can now look up everyone who is loved: `?X WHERE ?Y loves ?X`. Similarly we can look up everyone who is logically connected to Mary: `?X WHERE ?Y ?R Mary OR Mary ?R ?Z`. This is the core of Semantic Web processing. Naturally, the details are tremendously involved, even when we constrain ourselves to literal values such as `"1"^^xsd:nonNegativeInteger`, which is the Semantic Web way of describing the integer 1 by reference to XML Schema. For our purposes, however, we can get most of the value of the system without those types; we just want to retrieve the assertions we have put in, along with some of their logical consequences.

Namespaces: We will want to combine assertions, to write programs that effectively "know" inference rules, such as the assertion that

```
IF ?X isTallerThan ?Y AND ?Y isTallerThan ?Z
THEN ?X isTallerThan ?Z
```

To perform reliable inferences on the names in our assertions, we want to make sure the names are unique, with no accidental duplications. For them to be unique, we give them namespaces: `nt:John`, for example, where we make sure there's only one "`John`" in the "`nt`" namespace, where we define that namespace as a URI that we control. If somebody else has already defined the "`loves`" relation in some namespace, we borrow it; perhaps `emot:loves` will become a standard way of describing that sort of data. In order to describe our Web Service hierarchies, we invent or borrow names for entities and their relationships. RDF itself provides a basic vocabulary for concepts such as "`rdf:type`"; RDF Schema adds a simple `rdfs:Class` system with subclasses and related vocabulary items. Let's assert that our particular string-matching Web Service is a `stringService` and that such a service could be extended into a `regExpService`, perhaps one that takes a URI and a regular expression, downloads the URI contents and runs the Unix `grep` utility, then packages the resulting lines back into XML. Similarly a regular-expression service can be extended into a `grammarService`—perhaps one that takes a URI and a context-free grammar expressed in a form suitable for the Unix `yacc` utility, then returns an XML-encoded parse tree. Describing these services in full is hard, but the notion of extension is easy:

```
nt:stringService     rdf:type    rdfs:Class
nt:regExpService     rdf:type    rdfs:Class
nt:grammarService    rdf:type    rdfs:Class
nt:regExpService     rdfs:subClassOf nt:stringService
nt:grammarService    rdfs:subClassOf
nt:stringService
```

RDF toolkits should be able to "understand" this; any toolkit that provides a graphical way to visualize an RDF class hierarchy will automatically be able to display this, in an outline form, or as nested boxes, or a flash interactive animation, or simply as a path expression, such as

```
nt:stringService/nt:regExpService/nt:grepServiceA
   nt:stringService/nt:regExpService/nt:grammarService/
nt:yaccServiceB
```

We don't have to decide on visual expression at this level. We can read the assertions as text, make sure they say what we want them to say, and leave it to the interface designers working on the rest of this book.

Composability: If Joe's Web site provides a useful set of assertions about Web Services in general, we can simply import that set of assertions—if he uses the prefix `nt:` then of course we will substitute something else as the prefix representing his namespace. We can then connect his assertions with ours by something as simple as

```
nt:stringService   rdfs:subClassOf   joe:WebService
```

Inferences can now proceed; in particular, it should be now be clear that a grammarService is a WebService and that a query returning all WebServices had better return every grammarService. We can easily imagine applications that compose on-the-fly hierarchies of Web Services from disparate sources and present them to human users, who will then be able to pick the services they want and plug them into their own programs. If somebody develops appropriate hierarchies of predefined vocabulary for Web Service classes, then we can even expect to present these choices to programs.

Multiplicity of Hierarchies: After constructing a hierarchy, we can then consider other attributes, any other attributes. For example, we can split services into those that return integers and those that return text; further split those that return integers into those that return match-counts and those that return match-locations. Any attribute might be thought of as a trivially shallow hierarchy, one that splits up all groups into subgroups corresponding to the values of that attribute. This doesn't seem to be a very helpful way to think with UDDI's TModels, but RDF and RDFS make it easy.

Multipart Relations: When we think of data as lists of triples, it's easy enough to handle a mere pair: we might say that `isTallerThan` is a transitive relation by saying `transitive(isTallerThan)`, but we can fake it acceptably with a dummy argument for "`transitive`", or by asserting something of the form

isTallerThan isA transitiveRelation

(We will get to an actual standard vocabulary for that in a moment.) That's not a problem, but what if we want to assert something that takes four slots, such as "`inLoveTriangle(John,Mary,Joe)`"? No three-slot assertion can express this, nor can one express the notion that "`John installed perl` on `myLinuxBox` on `9/15/2004",` which takes five slots. This does not restrict the expressive power of RDF, however, because we allow new names to be introduced; RDF has a special notation for a new name to be treated as an existential variable. We can get such ideas across with assertions of the form

```
_:x isA LoveTriangle
 John isIn _:x
 Mary isIn _:x
 Joe isIn _:x
_:y isA Installation
_:y wasInstallOf perl
_:y wasInstallBy John
```

The system naturally doesn't care whether the data are business or personal; they are there, to be looked up, and we can look up people who've installed perl, or people who are in some group with Joe, by asking for

```
?person WHERE ?i isA Installation AND ?i wasInstallOf
 perl AND ?i wasInstallBy
?person
?x WHERE ?x isIn ?g AND Joe isIn ?g
```

The level of matching we're looking for is somewhere in between what we're used to in SQL, on the one hand, and what we're used to in artificial-intelligence languages like Prolog, on the other.

Almost any RDF system will have to deal with this kind of inference. Now we can add rules:

```
IF (?X isTallerThan ?Y)  AND (?Y isTallerThan ?Z)
THEN (?X isTallerThan ?Z)
```

From this we should be able to look up those who are taller than Joe, and find John as well as Mary. Much RDF processing will involve such rules, as was part of the plan from the beginning: see http://www.w3.org/2000/10/swap/doc/Rules.

Within RDF, the triples may be comprised of literals or URIs (QNames, with namespaces), and we gain some traction by sharing these with the world. For example, rather than saying `thisBook hasPublisher Springer`, we would use a convention for referring to "this" document, which is "<>"; we would use the Dublin Core standard vocabulary within which `dc:publisher` is well-defined, and make sure that we're using the same publisher name that is used elsewhere, saying

```
<> dc:publisher Springer-Verlag
```

RDF software can then look for `dc:publisher` anywhere in an RDF structure, and produce lists of publications. RDF itself has a fairly extensive standardized vocabulary, including not only general notions of `rdf:type` but structural types such as `rdf:Seq` and `rdf:Bag`; the entanglement of John, Mary, and Joe would be described as an `rdf:Bag` because it doesn't matter what order you write them in.

Facets: We have not gone far in covering the basics of RDF, but we have enough to look at a real example illustrating what we mean by ad hoc hierarchies. The best-developed of these is probably Siderean's Seamark system, described in Chapter 15 of Practical RDF. We will explore the first of Siderean's Featured Sites, found on http://www.siderean.com/sites.html as we write (November 2004). This is http://www.environmentalhealthnews.org, one of whose feeds is syndicated daily by the United Nations Environmental Programme's Our Planet site at http://www.ourplanet.com/, and is set up to for flexible syndication by other environmental and health sites, so the data are seen by millions every month. (Full disclosure: the biologist who runs the site, J. Peterson Myers, is the brother of one of us.) The purpose of the site is to help the user keep up with news in its particular area, and the simplest usage is to visit the page and see what's new—but it is also straightforward to subscribe to an RSS newsfeed, and this newsfeed need not contain everything that's new. As the site's "About EHN" page explains, "Software driving the EHN website has been developed from the ground up using Semantic Web principles, in a partnership of Siderean Software, the Edgerton Foundation, and EHS. All data on the site are stored using the Resource Description Framework (RDF), which lends itself nicely to the site's following functions: . . ."

Every incoming item is tagged as being of Type "News Stories," "Scientific Studies," etc.; that Type is only a one-level hierarchy. Each item is separately tagged according to the "Current Issues" it applies to; one of these is "Children's Health." Select that, and see several subcategories, including "Birth Defects." Select that, and see several subcategories, including "Down syndrome." Select that, and we're down to 29 stories, which for the most part were mechanically tagged, brought into the ontology, because somewhere in the text was a match for a regular expression representing that syndrome. This is then checked manually, but it does not have to be entered manually because the position of "Down syndrome" in the hierarchy is predetermined. Anyone who has subscribed to categories including these, for example, "Birth Defects" (without even selecting "News Stories"), will automatically see this.

Let's try again, for a specific multifaceted query, one that identifies a couple of news stories about clearing contaminated dirt in Louisiana:

```
Current query:
Type: News Stories
Current issues: Hazardous products > Paints, sealers,
 and varnishes
Human health condition: Cognitive/behavioral >
 Learning disabilities > ADD/ADHD
Contamination agent: Fuels > Petroleum products/byproducts
Exposure pathway: Accidental exposure
Coverage: Louisiana
```

This query picks out "News Stories" as the value of the Type facet. The "Current issues" facet is a two-level hierarchical facet: "Paints, sealers, and varnishes" within "Hazardous products." The "Human health conditiong facet has three levels; the Contamination agent has two. It is perfectly straightforward to subscribe to the RSS feed containing all stories matching this, and to be notified by bloglines.com if any new ones appear. The actual query is exposed to the world as one long URL as follows (wrapped for readability):

```
http://www1.environmentalhealthnews.org/archives.jsp?
 sm=fr4%3Btype6%3B5Story12%3BNews+Storiesfr13%3B
 currentissues31%3B5Paints%2C_sealers%2C_and_varnishes30%3B
 Paints%2C+sealers%2C+and+varnishesfr13%3B
 humhealthcond9%3B5ADD_ADHD8%3BADD%2FADHDfr18%3B
 contaminationagent30%3B
 5Petroleum_products_byproducts29%3BPetroleum+products%2F
 byproductsfr15%3B
 exposurepathway20%3B
 5Accidental_exposure19%3BAccidental+exposurefl8%3B
 coverage9%3BLouisiana
```

It's a curious way to visualize a Semantic Web, a very limited way, but it works quite well—except that those two stories linked by EHN are more than a year old, and have vanished from the Web, leaving only their brief descriptions at EHN behind them. RDF works beautifully at flexible descriptions of a loosely coupled collection of hierarchies, but it doesn't prevent link rot.

We expect a slowly increasing range of services to be offered as Web Services; we expect RDF descriptions of them to become increasingly common; we expect human-oriented interfaces, like EHN's, to use facets to organize them so that you can plug

your service client to the right service just as EHN lets you plug your RSS client to the right feed. Note the role of the human here: the human role is being facilitated, but not replaced. We have an automatic transmission for the information superhighway, but we don't have an autopilot. Can we go beyond that? There's some hope, but not yet a trend.

Web Ontology: For improved matching and inference within RDF, the W3C has developed the OWL system described at http://www.w3.org/TR/owl-features/. Just as RDF itself standardizes structure, and RDFS standardizes hierarchy, OWL standardizes inference—not that any of them is really doing much of anything, but they provide a vocabulary that can be used in a consistent way. For example, we often want to use transitive relations; in the form suggested by http://www.w3.org/TR/owl-test/byFunction#function-TransitiveProperty we can say

```
nt:isTallerThan rdf:type owl:TransitiveProperty
```

With that input, a system that understands OWL will now be able to retrieve both of the values for `?x WHERE ?x isTallerThan Joe`. That's not much to ask, and it's quite useful. Similarly, we can talk about a romantic triangle such as the "`_:x`" above, involving John, Mary, and Joe. OWL has a predefined vocabulary for cardinality, so we can assert that "`_:x`" is of cardinality three:

```
_:x    owl:cardinality    "3"^^xsd:nonNegativeInteger
```

Some of these are harder to handle than others; the minimal OWL is called OWL Lite, and handles the TransitiveProperty but the only cardinalities it recognizes are 0 and 1. To make inferences involving specific numbers in general would require OWL Full, where inferences are not guaranteed to take a reasonable or even finite amount of time. That's all right; OWL Lite is quite enough to describe most of what we'd want in setting out an ontology of Web Services, whereas even OWL Full doesn't go far toward providing useful axiomatic descriptions of those services' input and output. OWL Lite reasoning systems, such as the Jena framework (in Java) at http://www.hpl.hp.com/semweb/jena.htm, are freely available.

Where Are We Going? It is fundamentally straightforward to provide RDF/OWL ontology services via SOAP, simply by XML messaging. We are more interested in the possibility of ontologies of Web Services, extending the possibilities of interconnection for which UDDI was designed and implemented. For the moment, we can't quite visualize how this is going to work, but we are closer than we were.

5.8 References

Ankolenkar et al. (2001). DAML-S: A semantic markup language for Web Services. In: *Proceedings of SWWS' 01: The First Semantic Web Working Symposium.* Available: http://www.semanticweb.org/SWWS/program/full/SWWSProceedings.pdf.

Axis (2004). *Axis User Guide.* Available: http://xml.apache.org/axis/index.html.

Colgrave, J., Januszewski, K. (2004). *Using WSDL in a UDDI Registry, Version 2.0.2.* Available: http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v202-20040631.htm.

Date, C.J. (2000). *What, Not How: The Business Rules Approach to Application Development.* Reading, MA: Addison-Wesley.

JENA (2004). *Jena2 Reasoner Subsystem.* Available: http://www.hpl.hp.com/semweb/jena.htm.

MS .NET (2004). *Microsoft .NET Development Center.* Available: http://msdn.microsoft.com/netframework/.

OWL (2004-0). *OWL Web Ontology Language Guide.* Available: http://www.w3.org/TR/2004/REC-owl-guide-20040210.
OWL (2004-1). *OWL Web Ontology Language Test Cases.* Available: http://www.w3.org/TR/owl-test/.
Powers, S. (2003). *Practical RDF.* Sebastopol, CA: O'Reilly.
RDF (1997). *RDF Architecture: W3C Note 29.* Available: http://www.w3.org/TR/NOTE-rdfarch.
RDF (2004-0). *RDF Primer.* Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.
RDF (2004-1). *RDF Concepts and Abstract Syntax.* Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
RDF (2004-2). *RDF Vocabulary Description Language 1.0.* Available: http://www.w3.org/TR/rdf-schema/.
SOAP (2003-0). *SOAP Version 1.2, Part 0: Primer.* Available: http://www.w3.org/TR/soap12-part0/.
SOAP (2003-1). *SOAP Version 1.2, Part 1: Messaging Framework.* Available: http://www.w3.org/TR/soap12-part1/.
SOAP (2003-2). *SOAP Version 1.2, Part 2: Adjuncts.* Available: http://www.w3.org/TR/soap12-part2/.
UDDI (2003). *v3.01 Specification.* Available: http://uddi.org/pubs/uddi_v3.htm.
UDDI (2003). *v3 Schema.* Available: http://www.uddi.org/schema/uddi_v3.xsd.
WSDK/ETTK (2003). *IBM Emerging Technologies Toolkit.* Available: http://www.alphaworks.ibm.com/tech/ettk.
WSDL (2004-1). *WSDL 2.0, Part 1: Core Language.* Available: http://www.w3.org/TR/2004/WD-wsdl20-20040803.
WSDL (2004-2). *WSDL 2.0, Part 2: Predefined Extensions.* Available: http://www.w3.org/TR/2004/WD-wsdl20-extensions-20040803/.
WSDL (2004-3). *WSDL 2.0, Part 3: Bindings.* Available: http://www.w3.org/TR/2004/WD-wsdl20-bindings-20040803/.
XMLP (2004). *XML Protocol Working Group.* Available: http://www.w3.org/2000/xp/Group/.

Chapter 6

Recommender Systems for the Web

J. Ben Schafer, Joseph A. Konstan, and John T. Riedl

6.1 Introduction

As the Web rapidly evolved into an immense repository of content, human users discovered that they could no longer effectively identify the content of most interest to them. Several approaches developed for improving our ability to find content. Syntactic search engines helped index and rapidly scan millions of pages for keywords, but we quickly learned that the amount of content with matching keywords was still too high. Semantic annotation helps assist automated (or computer-assisted) processing of content to better identify the real contents of pages. A Semantic Web would help people differentiate between articles on "china" plates and articles about "China" the country. Recommender systems tried a different approach—relying on the collected efforts of a community of users to assess the quality and importance of contents. For example, a Web page recommender may identify that a particular page is popular overall, or better yet, popular among people with tastes like yours.

Recommender systems are generally independent of syntactic and semantic approaches. Hence, recommenders are often combined with other technologies. For example, the search engine Google (www.google.com) combines syntactic indexing with a nonpersonalized recommender (the PageRank algorithm) that assigns higher value to pages that are linked to by high-value pages. Similarly, most recommender system interfaces allow users to first use search techniques to narrow down the set of candidates being evaluated.

The term "recommender systems" was first coined at a workshop in 1996, and has been used imprecisely and inconsistently in published work. For this chapter, we limit ourselves to discussing systems that have two key properties: (1) they incorporate the experiences or opinions of a community, and (2) they produce for the user a mapping of scores to items, a set of recommended items, a ranking of items, or a combination of these three. We further focus on personalized recommenders—those that produce different recommendations for different users—and automated recommenders—those that do not require explicit actions to send or receive recommendations. These are the recommender systems that have been most prominent and successful in industrial applications (Schafer et al., 2001).

Such recommenders have not only been successful commercially, but they have been shown to be valuable in experimental research. Early studies of recommenders in Usenet news showed that users valued personal recommendations for articles and that they were nearly twice as likely to read an article with a high recommendation

score compared with one with a low score (Miller et al., 1997). More recent studies have shown that the displayed recommendation score also influences the user's subsequent rating of items that are recommended, and that user satisfaction with recommender systems suffers when recommendations are randomly perturbed to create a less accurate recommender (Cosley et al., 2003).

This chapter provides a survey of research on recommender systems, focusing on recommenders that can be applied to the Web. While reflecting on research that was largely independent of the Semantic Web, it looks at how recommenders and the Semantic Web can be integrated to provide better information sources for users. Finally, it reviews a number of user interface issues related to recommenders on the Web.

6.2 The Beginning of Collaborative Filtering

As content bases grew from mostly "official" content, such as libraries and corporate document sets, to "informal" content such as discussion lists and e-mail archives, users began to experience a new type of overload. Pure content-based techniques such as information retrieval and information filtering were no longer adequate to help users find the documents they wanted. Keyword-based representations could do an adequate job of describing the topic of documents, but could do little to help users understand the nature or quality of those documents. Hence, a scholarly report on economic conditions in Singapore could be confused with a shallow message by a recent visitor who speculates that "the economy is strong—after all, I paid over $12 for a hotel breakfast!" In the early 1990s there seemed to be two possible solutions to this new challenge: (1) wait for improvements in artificial intelligence that would allow better automated classification of documents, or (2) bring human judgment into the loop. Sadly, the AI challenges involved are still formidable, but fortunately human judgment has proved valuable and relatively easy to incorporate into semiautomated systems.

The *Tapestry* system, developed at Xerox PARC, took the first step in this direction by incorporating user actions and opinions into a message database and search system (Goldberg et al., 1992). Tapestry stored the contents of messages, along with metadata about authors, readers, and responders. It also allowed any user to store annotations about messages, such as "useful survey" or "Phil should see this!" Tapestry users could form queries that combined basic textual information (e.g., contains the phrase "recommender systems") with semantic metadata queries (e.g., written by John OR replied to by Joe) and annotation queries (e.g., marked as "excellent" by Chris). This model has become known as *pull-active collaborative filtering*, because it is the responsibility of the user who desires recommendations to actively pull the recommendations out of the database.

Tapestry predates the Web, let alone the Semantic Web, but it carries several important lessons. First, there is a great deal of information to be gained from even simple semantic markup. Being able to select pages based on authors, modification dates, and other commonly available fields makes it easier for users to find the pages they want. Second, usage patterns from other users can be valuable, especially if those users are trusted. Pull-active collaborative filtering requires a community of people who know each other well enough to decide which opinions matter; I need to know that Chris's "excellent" means something, while Jean's may not. Finally, Tapestry reminds us that after a certain point, it becomes more effective to add deep (in this

case human) semantic annotations, rather than wait for the technology to improve automated syntactic and surface semantic processing.

Soon after the emergence of Tapestry, other researchers began to recognize the potential for exploiting the human "information hubs" that seem to naturally occur within organizations. Maltz and Ehrlich (1995) developed a *push-active collaborative filtering* recommender system that made it easy for a person reading a document to push that document on to others in the organization who should see it. This push-recommender role has become popular, with many people today serving as "joke hubs." They receive jokes from all over and forward them to those they believe would appreciate them (though often with far less discriminating thought than was envisioned).

Tacit Corporation's ActiveNet (www.tacit.com) extends these ideas into a broader knowledge-management system. The system combines traditional information filtering with a push-distribution interface and query system to find experts. The system builds private profiles of each user's interests based on keyword analysis of the content the user generates during daily work. Depending on the source of these keywords (e.g., public documents vs. private emails) differing privacy levels are assigned to keywords, and users can override these initial settings using the Profile Management Interface (Figure 6.1). The push-distribution interface allows the sender of a message to ask the

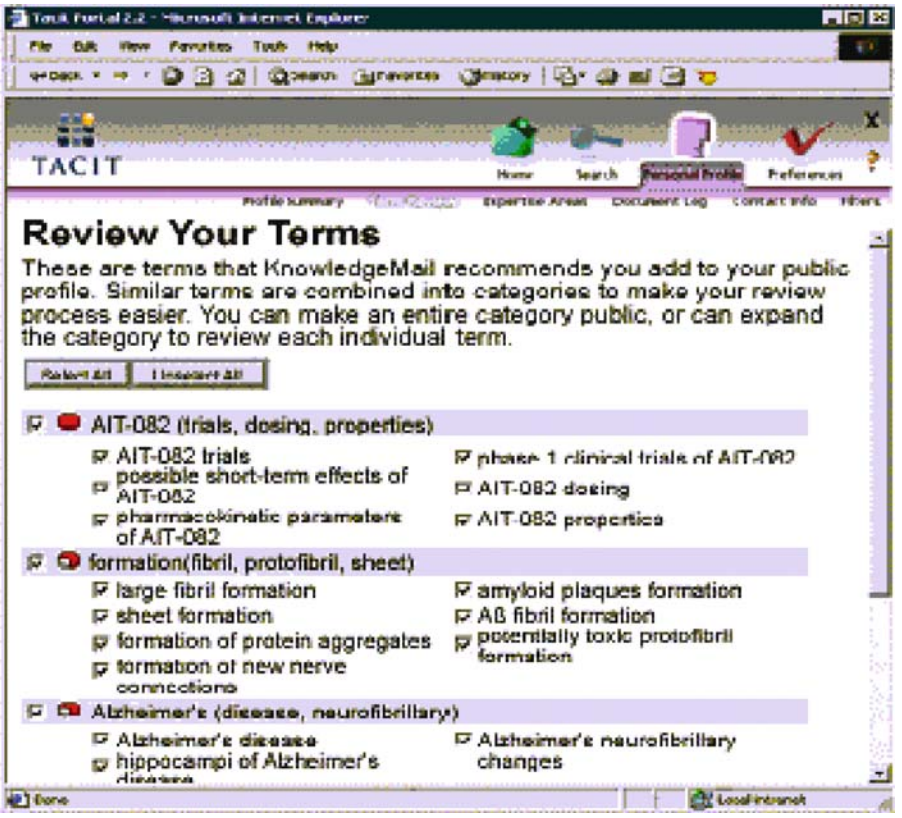


Figure 6.1 Tacit's KnowledgeMail provides a Profile Management Interface through which users can modify the privacy levels of their profile. Other users can push content to these users based on the public portions of a profile.

system to notify appropriate recipients based on interest. In addition, users can seek expertise by asking the system to contact experts on the topic of a particular message. This hybrid of content and manual collaborative filtering can help an organization be more efficient in distributing information to those who need it without overwhelming everyone with everything.

So far, push-active recommenders lack the rich semantic annotation promised in the Semantic Web. There is great potential for improvement of these systems if semantic tagging could be included—but only if it is automated. For example, if automated tagging could identify business-related semantic entities such as competitors, markets, products, etc., messages could almost certainly be routed more automatically, and profiles could be structured more sensibly and compactly.

6.3 Automated Collaborative Filtering

A limitation of active collaborative filtering systems is that they require a community of people who know each other. Pull-active systems require that the user know whose opinions to trust; push-active systems require that the user know to whom particular content may be interesting. Automated collaborative filtering (ACF) systems relieve users of this burden by using a database of historical user opinions to automatically match each individual to others with similar opinions. Another advantage of ACF systems is the ability for users to participate using a pseudonym, and therefore without revealing their true identity.

Intuitively, ACF systems follow a process of gathering ratings from users, computing the correlations between pairs of users to identify a user's "neighbors" in taste space, and combining the ratings of those neighbors to make recommendations. More formally, the early ACF systems—GroupLens (Resnick et al., 1994), Ringo (Shardanand and Maes, 1995), and Video Recommender (Hill et al., 1995)—all used variants of a weighted *k*-nearest neighbour (NN) prediction algorithm.

Since the construction of these early ACF systems, a variety of algorithmic research has been conducted that attempts to refine or extend the basic NN algorithm. Herlocker et al. (1999) studied the effect on accuracy of the various refinements that could be made to the tradition NN algorithm including various similarity correlations, weighting techniques, methods for selecting neighborhoods, and the use of normalization of ratings to reflect different user rating schemes. Sarwar et al. (2001) extend the NN algorithm to correlate items-to-items, which can be helpful when there are many more users than items.

Deshpande and Karypis (2004) have confirmed that such systems are as accurate as user-based techniques while producing recommendations up to two orders of magnitude faster. McLaughlin et al. (2004) have modified the algorithm to include belief distributions, which can be used to better differentiate between highly rated items by weakly correlated users and "average" rated items by strongly correlated users.

Other algorithmic research has focused on significantly different approaches. Breese et al. (1998) studied the generation of Bayesian networks with decision trees at each node of the structure. Wolf et al. (1999) generated recommendations through a similarity graph-based approach known as horting. Sarwar et al. (2000) generated more efficient recommenders by considering the application of dimensionality reduction through singular value decomposition. Jin et al. (2003) build a system that employs two complementary models, one of a user's preferences, and the other of a user's rating

scheme, and use a Bayesian algorithm to combine the predictions produced across varying models.

With the variety of algorithms that can be applied to the generation of recommendations in ACF systems, it is important to consider mechanisms for evaluating different systems. Herlocker et al. (2004) studied the different metrics different practitioners have used to determine if one system is "better" than another. In doing so, they have discovered large differences in techniques and rationale. Evaluation techniques vary based on the types of task being evaluated and the types of data used in generating predictions. Furthermore, metrics vary from those evaluating the pure quality of the predictions to those that attempt to analyze additional attributes such as user satisfaction and the ability to make decisions with confidence. They have concluded that metrics can be collapsed into three equivalence classes where metrics within each class correlate strongly with each other, but weakly with those from other classes.

Algorithmic research for pure ACF has slowed in recent years as the change in accuracy from these new algorithms has decreased significantly. Instead, research has focused on algorithms for systems with very different objectives, such as algorithms for hybrid systems, or those designed to support specific interfaces or unusual data sets. This type of research is presented in more detail in future sections.

Interfaces for recommender systems attempt to visually present recommendations in a manner useful to users.

The GroupLens recommender system (Resnick et al., 1994; Konstan et al., 1997) used a very explicit interface where ratings were entered manually by keystroke or button, and ratings were displayed numerically or graphically (Figure 6.2). Improvements to this type of interface have taken two forms: (1) less visible interfaces such as the implicit ratings discussed below and the use of subtle suggestions (e.g., product placement) in place of explicit predictions and identified recommendations, and (2) more information-intensive interfaces, which have generally been used to display additional data to help users make decisions, such as Ringo's community interaction interfaces and the explanations interfaces presented below.

Commercial users of recommender systems may classify interfaces according to the types of recommendations made. *Predictions* are forecasts of how much a user will like an item, and can be used in response to queries, as annotations to be used while browsing, or in combination with other techniques. *Suggestions* are lists of items, or individual items, presented to a user as recommended, often these are associated with the top levels of a site or a personalized "my site" page. *Item-associated* recommendations are displayed in conjunction with specific items, usually based on evidence that those who liked or purchased the item being viewed also liked the recommended ones. Finally, *organic* interfaces often remove the evidence of the recommendation, making the site simply appear natural, yet personalized. For example, a custom Web-based newspaper may rearrange stories to place ones with high predicted interest at the front (Bharat et al., 1998).

The basic approach of automated collaborative filtering has been developed quite apart from the semantic tagging and analysis that is emerging in the Web. Indeed, one of the goals of early collaborative filtering was to be "content-neutral," and indeed the early ACF software was completely ignorant of all content attributes. Later work has shown that this approach was too pure; humans care about content, and both their ratings and their usage of recommender systems can be content-specific. Accordingly, later systems, including commercial systems, support the definition of content attributes that can be used both to partition the space of items (to define subspaces with better correlations among users) and to support interfaces that allow

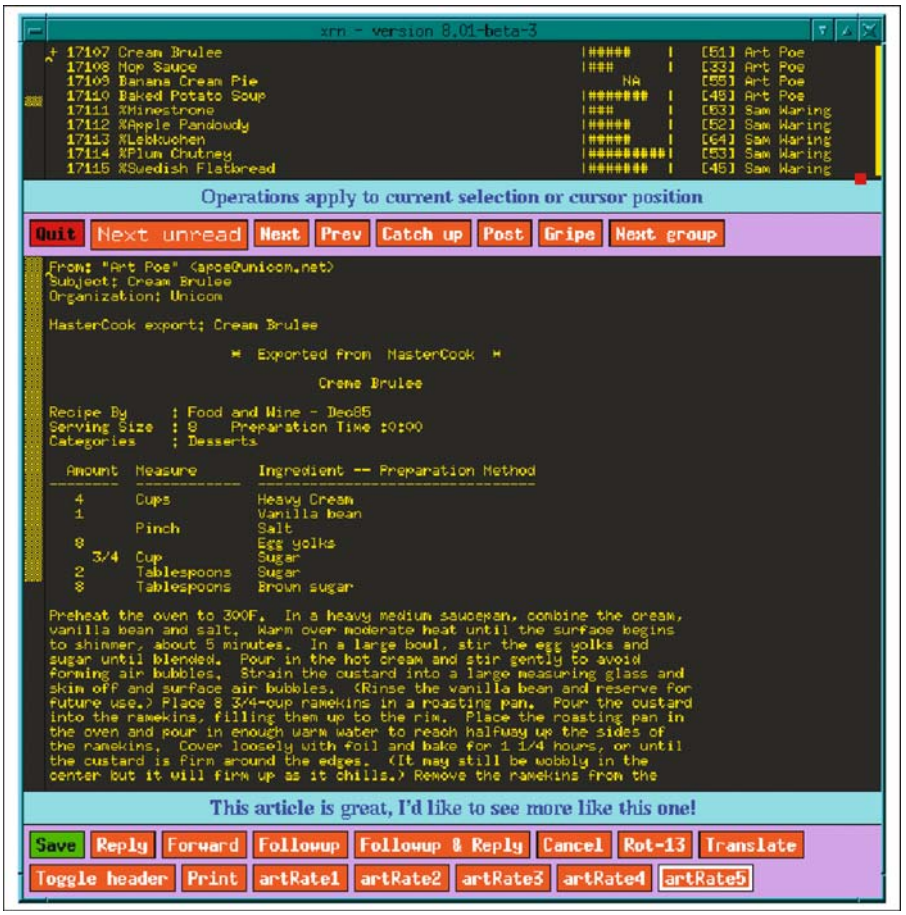


Figure 6.2 A modified Xrn news reader. The GroupLens project added article predictions (lines of ### on the top right) and article rating buttons (bottom).

users to overlay content and quality questions. A simple example of this approach is found in MovieLens, which allows users to receive recommendations for movies by genre and date.

Looking forward, however, there are areas of synergy between automated collaborative filtering and the Semantic Web. First, semantically tagged content can be better classified, and ratings of it can be better analyzed to build models of user interest. Second, user opinions *are* semantic evaluations of content; they can be processed to supplement other techniques for semantic analysis. For example, if a semantic tagger cannot tell whether a particular Web page refers to the old New York Giants baseball team, or the New York Giants football team, perhaps because the content is sufficiently well in the overlap, human ratings could well disambiguate the reference. Finally, automated collaborative filtering is still useful in a Semantic Web, and may add more value to users when a topic is already well-established.

Interesting examples of such overlap occur in the Platform for Internet Content Selection (PICS) and the Platform for Privacy Preferences (P3P). PICS was designed to provide a standard interface through which Web browsers could access rating bureaus,

and through which rating bureaus could provide page ratings for their subscribers (Resnick and Miller, 1996). The idea was to unify the wide range of content rating and filtering services that were emerging with a standard interface, so that those who wanted to filter out pornography or immorality could do so, and so that others who might want to rate pages on other criteria (including automated collaborative filtering) could set up such rating bureaus. Though the rapid explosion and pace of the Web dissuaded many would-be ratings services, today we can think of PICS annotations as yet another source of semantic annotations for Web pages and links. Indeed, a Semantic Web browser today may choose to consult the filtering services that identify content inappropriate for minors so that they can alter the display of a page (by omitting such links or obscuring inappropriate images, for example).

P3P grew out of PICS and was designed to provide a standard interface through which Web sites can communicate about their privacy policies (Cranor, 2002). "Traditional" policy statements give consumers access to information about a site's privacy policies. These policies are written in a human-readable language that can be confusing due to human error in editing or purposefully written to contain room for interpretation. As such, users must decipher the language of the policy to determine if the practices are acceptable to them and whether they choose to do business with a site. P3P provides a way for Web sites to present their privacy policies in a standardized and machine-readable format by answering a series of multiple-choice questions that clearly place the site within one of several "buckets." By building P3P tools into browsers and other Web-based applications, users can configure a set of user preferences for the type of data they are willing to share and the circumstances under which these data may be shared. Web tools can compare the published privacy policies of visited sites with a users preference profile and offer advice or alerts when users encounter sites whose privacy practices may not meet their comfort level.

6.4 Enhancing Collaborative Filtering with Semantics

Content, and in particular semantic content, can be used to address two of the major problems facing collaborative filtering recommender systems: the startup problem and the problem of changing moods. This section explores three different approaches to using content within ACF recommender systems.

6.4.1 New Users and New Items

One opportunity to improve collaborative filtering systems is during the startup process for new users or new items. Collaborative filtering systems cannot recommend items that have never been rated by any user because they have no data on which to base a recommendation. Similarly, collaborative filtering systems offer little value to new users who have not yet rated items. Designing systems that can effectively obtain data from new users or about new items should improve the overall accuracy and helpfulness of recommenders.

Several projects from the GroupLens Research group have considered methods for more efficiently gaining data from new users. Rashid et al. (2002) studied six techniques that CF systems can use to select which items to ask a user about to gain information about a new user. These include entropy, selecting the items that will tell the system the most about the user but may be less likely to have actually been

viewed by the user; popularity, which are the items most likely to have been viewed by a user but may produce less value as a predictive tool; a combination of these two; and item-to-item prediction, a technique that uses the already-given ratings to calculate the items most likely to have been seen by the user based on item-item correlations. Results suggest that the technique to choose depends on the domain as well as the type of user experience the system wants to provide. In a study in the MovieLens movie recommender system, item-to-item prediction identified items that the user could rate, but that were of low marginal information value. Entropy yielded almost entirely obscure films that the user could not rate. The best performance came from a mixture of entropy and popularity.

McNee et al. (2003a) considered interfaces that allow the user to help select the selection of items that are used to develop the initial user model. They conclude that such interfaces have little effect on the accuracy of the system, but do affect user perception of the time and effort spent during the initial rating period and also can affect user loyalty to the system.

Another approach is to use the Semantic Web to help these users find items of interest immediately, according to their past interest profile. For instance, a user navigating to a new site might be recommended items that are labeled the same as items she has liked on other sites she has visited. Over time, the recommendations could be tuned based on the user's specific interactions with the new site in addition to the Semantic Web data. Incorporating both types of recommendations in the same interface would enable users to smoothly switch from startup situations to long-term personalization relationships with Web sites.

Semantic Web information can be used in other ways to improve the performance of collaborative filtering on Web sites. One idea is to use the Semantic Web categories to provide a partitioning of the information on a Web site into categories that should be separately recommended. Experience with collaborative filtering has shown that recommendations made based on data from closely related items are more accurate than recommendations made across broad categories. For instance, if a Web site has medical data on both strength training and cardiovascular fitness, recommendations of content may be more valuable if they are within those categories rather than across categories.

Alternatively, collaborative filtering can be used to identify potential areas for improvement within the Semantic Web labeling for a Web site. For instance, the collaborative filtering engine might point out areas of the site that are rated very similarly though they are far apart in the Semantic Web hierarchy. These differences may point out areas in which the Semantic Web labeling differs from the way users look at data on the site—and may provide an opportunity to improve the usability of the site.

Another possible type of convergence between collaborative filtering and the Semantic Web is the use of collaborative filtering to choose among several possible labelings for a Web site. As the Semantic Web spreads, there are likely to be competing alternatives for labeling popular sites. A new visitor to one of those sites may use collaborative filtering to choose the most useful of the labelings for her.

6.4.2 Integrated Content/Collaborative Filtering Solutions

Combining the Semantic Web and collaborative filtering to solve the startup problem is related to a general approach to collaborative filtering problems in which classical content filtering is combined with collaborative filtering to produce better

recommendations. Pure collaborative filtering is impractical in many domains because of the rate of new items (that arrive unrated) and because of the high sparsity (and high desired sparsity) of the domain. Domains where items are created frequently but have a short useful lifespan are particularly problematic, limiting the effectiveness of recommender systems for news, live discussions, and live or scheduled entertainment. At the same time, there is great interest in recommendations for such content—particularly in the area of news and television programs. Several researchers are examining approaches to better integrate content filtering with collaborative filtering to take better advantage of the strengths of each. Burke (2002) describes several of these approaches including several discussed below, in his survey of hybrid recommender systems.

The GroupLens Research group completed two different studies of automated filtering agents; dubbed Filterbots. Sarwar et al. (1998) considered basic nonpersonalized, text-analysis agents (the length of the articles, the percentage of new content, and the quality of spelling, etc.) in the domain of Usenet news articles. Results showed that different newsgroups and users benefited from different agents, but that overall the agents could be used to improve the recommender system for most users. Good et al. (1999) extended this work by adding a variety of personalized agents for the domain of movies. The agents were trained using rule-induction systems, TFIDF information filtering systems, and basic linear regression on data including genre, cast lists, and descriptive keywords. Results showed that for a small user community (50 users), the best agents outperformed collaborative filtering using only users, but the combination of users and agents in a collaborative filtering system outperformed all other combinations. Later results suggest that as communities get larger, they reach a cut-off point at which agents stop adding much value.

Claypool et al.'s (1999) online newspaper project, P-Tango, creates a personalized newspaper for each user by combining collaborative filtering with user-selected categories and keywords. P-Tango is able to provide better recommendations for users by combining the two techniques than with either technique separately. Figure 6.3 shows the P-Tango front page for an established user.

Fab takes a very different approach to the same goal of combining content and collaborative filtering (Balabonovic and Shoham, 1997). In Fab, user profiles are created using classical content filtering techniques. Distances between neighbors are computed directly on the content-based profiles, rather than on user ratings vectors. These distances are then used by a collaborative filtering algorithm to select neighbors and make predictions in more or less the usual way.

The TechLens Project (Torres et al., 2004) recommends research papers to users by both mapping a Web of citations between papers into the collaborative filtering user-item ratings matrix and analyzing the content of a paper through a TF-IDF approach. Research has considered five differing methods for combining the results of these recommendations. Results of both online and offline studies indicate that different combination algorithms are appropriate in different contexts (is the user looking for papers that are a reasonable introduction to a topic, are the authoritative paper on a topic, present novel innovations on a topic, etc.) but that the combination of data produces results that can aid digital libraries in recommending helpful and meaningful research papers.

Quickstep and Foxtrot (Middleton et al., 2004) also recommend research papers, but do so based on the application of ontological user profiling. Quickstep maintains a personal interest profile for each user based on the topics browsed by users and explicit relevance feedback. More actively, it monitors the URLs visited by users and performs a nightly "offline" ontological classification of these using a computer science research

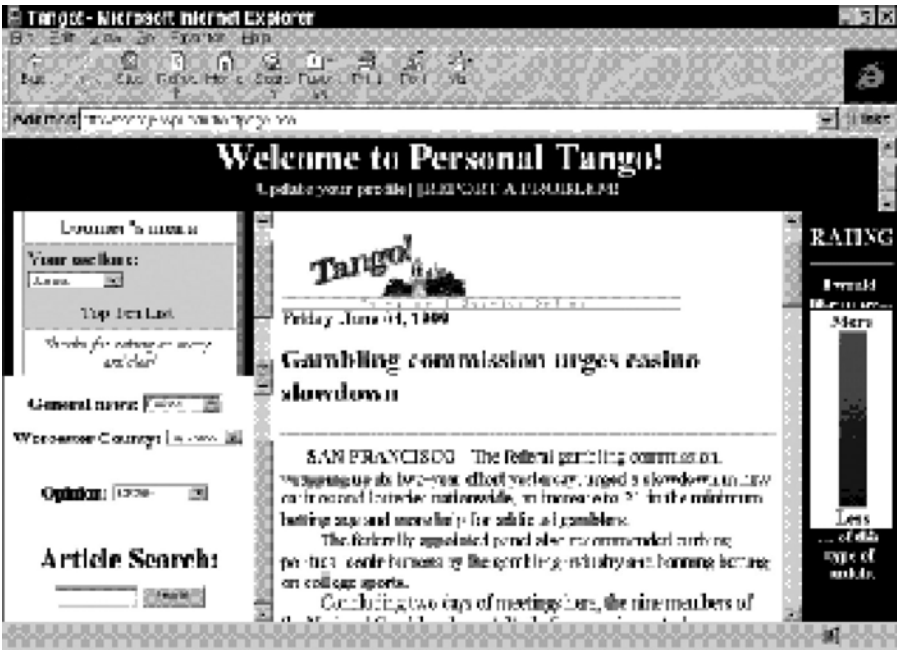


Figure 6.3 The Tango personalized newspaper combines collaborative and content-based filtering. It provides an appealing rating scale on the right.

paper topic ontology. From this information, Quickstep produces a set of offline recommendations available via the Web browser. Foxtrot added to the functionality of Quickstep by introducing the ability for users to access their profile, the inclusion of a search interface, and e-mail notification of recommendations.

Hoffman (2004) proposes an alternative approach to these techniques by considering the construction of Latent Semantic models for application in collaborative filtering settings. Among other techniques, statistical modeling algorithms in this class of recommenders can be modified to handle the appropriate mixture of explicit user ratings with latent class variables to discover user communities for the generation of recommendations.

All of these techniques can also be applied to the Semantic Web. Filterbots have been proposed as automatic semantic labeling agents for Web sites. The information in the Semantic Web on a site can be used to provide content hierarchies that can be used by systems like P-Tango and TechLens that combine collaborative and content filtering. The Fab approach suggests that neighborhoods can be selected based on other types of data than ratings, such as Semantic Web labels a user tends to prefer. Quickstep and Foxtrot add to the user-defined labels from the Semantic Web to introduce machine-defined ontologies. The Semantic Web may prove a platform for bringing together content and collaborative techniques, improving value to users.

6.4.3 Situational and Task-Focused Recommenders

Collaborative filtering recommenders usually work to find the best recommendations according to a user's complete history of interactions with a Web site. This is, after all,

one of their benefits; they learn what you want based on what you've done in the past. In many cases, though, a recommendation that is based on a less complete history, but that is more appropriate to a particular task, is preferred. For instance, a visitor to a medical Web site might be a long-standing visitor to the site who usually reads expert information about a chronic disease her daughter has been suffering from for years. However, on this particular visit she might be looking for information about a disease her mother has just been diagnosed with. In this case, she does not want the site to steer her toward the chronic disease she usually reads about, and she also does not want to see the expert information she usually reads. Instead, she wants to focus on a new disease, and wants to be treated as a novice. A situational recommender might give her the tools she needs to specify her short-term interest in contrast to her long-term profile.

MetaLens is an example of one approach to a situational recommender (Schafer et al., 2002, Schafer et al., 2004). MetaLens is an extension of the MovieLens framework in which users can specify attributes of the show they want to go to, including its cost, distance to the theatre, show times, appropriateness for children, critic recommendations, and MovieLens prediction. Users have control over which attributes they would like the recommended movies to match, and to which degree matches should be rewarded (or nonmatches penalized). This allows users to indicate that while they are interested in seeing a movie they will enjoy, it may be more important that the movie be appropriate for a 10-year-old and be completed by 9:00 P.M. Figure 6.4 shows a sample MetaLens screen shot of an experiment we did in which users were given specific instructions about a particular situation in which they might need recommendations.

Another situational recommender we have studied in the context of MovieLens enables users to create a set of profiles based on groups of movies (Herlocker and Konstan, 2001). Each profile is a group of movies that have something in common to the user. For instance, one such profile might be "Intelligent Spy Flicks." When the user asks for recommendations, the situational recommender looks for movies that

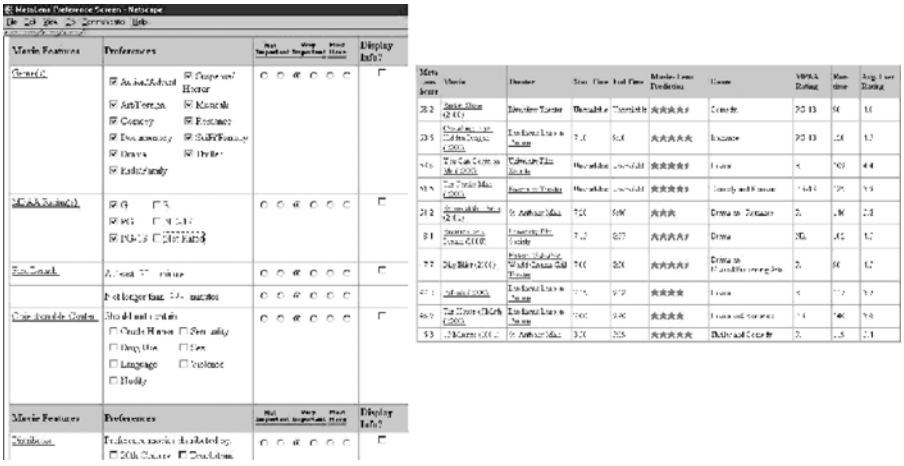


Figure 6.4 Query formation and combined recommendation screens from MetaLens. Users form query profiles by specifying preferences for specific features, the importance of each feature in the overall query, and which features to display with recommendations. The recommendation list is sorted by a single, composite score.

are closely related to those in the group, and that are appropriate according to his long-term interest profile.

One example of a situational recommender outside the context of movies is early work on TechLens. McNee et al. (2002) considered ways to recommend additional citations for a target research paper based on the current citation list for the paper. Offline research considered the formation of such recommendations, while online research studied the user perceived effectiveness of such recommendations.

Situational recommenders like these would benefit from the Semantic Web. The semantic hierarchy on a Web site could serve as information that would be used by a situational recommender to create recommendations appropriate to a particular user at a particular time. For instance, the labels on content at the medical Web site might be used to select content appropriate to the user's newly discovered interest. Collaborative filtering could select among the novice information on the new disease that the user would be most likely to find interesting.

6.5 Explanation and Inference

Recommender systems have two principal interfaces—the recommendation interface and the rating interface. This section discusses one of the more important examples of research to improve each interface by better matching it to user tasks and needs.

6.5.1 Explaining Recommendations

Recommender systems do not exist for their own sake; rather they exist to help users engaged in a task that would otherwise have too many alternatives to adequately consider. In some low-risk domains, users may simply be grateful for the suggestion, and may follow it; in other domains, however, they need to be convinced. In particular, they need to understand enough about a recommendation to decide whether and when to trust it.

One simple mechanism for helping users determine if they can trust a recommendation is through the addition of a confidence metric. While mathematical indicators of confidence exist and could be displayed to users, McNee et al. (2003b) chose to study the effectiveness of displaying simple graphics alongside recommendations for which the system was less confident. Figure 6.5 illustrates how such graphics were used. The single die icon near the movie title *The Horse's Mouth* indicates that it is risky, while the double dice icon alongside *Akahige* indicates that

Figure 6.5 Dice icons indicate recommendations that MovieLens considers risky due to limited ratings contained in the system. The single die icon indicates that *The Horse's Mouth* is risky while the double dice icon suggests that *Akahige* is very risky (McNee et al., 2003b).

it is very risky due to very few ratings in the database. Results indicated that such graphics were effective at conveying a system's confidence in its recommendations, but that minimal training was appropriate to teach users how to interpret these graphics.

More detailed methods of explaining how recommendations were generated can provide more information to a user to aid in her decision-making process. Four common models of explaining collaborative filtering recommendations are process, recommendation data, track record, and external data. Process explanations address the manner in which recommendations were produced, helping users determine whether that manner is appropriate for the decision being considered. Recommendation data can be explained generally, through summarization, or they can be presented or visualized in greater detail. The recommender system's track record for the user or in similar cases may help users gauge how well the system is likely to perform for them. Finally, external data, while not used in the recommendation process, can provide additional input to help users determine whether to follow a particular recommendation. These explanation models can be combined to create more elaborate recommendations. Additional explanation models can be used to address social navigations systems with data other than ratings.

Herlocker et al. (2000) studied 21 different explanations for Movie recommendations. Six of these explanations had a statistically significant positive effect on decision-making: three simple displays of neighbor ratings, one measure of historical correctness, and two directly content-focused explanations—the similarity of the recommended movie to other movies rated and the assertion that one of the user's favorite actors is in the cast. It should not be surprising that explanations that tie to content and semantics would be appealing to users—these explanations speak directly to user interests. A key conclusion is that Web recommender system effectiveness, overall, can be greatly enhanced by providing explanations that exploit any available semantic data. Similarly, automated markup systems may help users better cope with mistakes by offering users explanations for the markups made.

6.5.2 Focusing Implicit Ratings

Recommender system researchers have been studying ways in which ratings can be collected from users with the least effort and the most value. One promising approach is *implicit ratings*. Implicit ratings are measures of interest that are derived from observing the user's interaction with a site without asking the user to modify his behavior in any way. For instance, Tapestry stored annotations with documents identifying whether a user had printed or replied to a message (Goldberg et al., 1992). Other users could base their decisions on these annotations if they chose. The GroupLens project experimented with time-spent-reading as an implicit rating for Usenet news articles (Miller et al., 1997). Building on the work by Morita and Shinoda (1994) that showed that users spent more time reading articles they preferred, the GroupLens project was able to show that collaborative filtering was able to predict ratings for substantially more articles by using time-spent-reading and that the ratings predicted were of similar quality to explicit ratings-based predictions.

Another implicit rating is available in the link structure of the Web. Web links, in addition to providing direct access from one Web page to another, represent the Web page author's endorsement of the target page as useful and relevant, at least

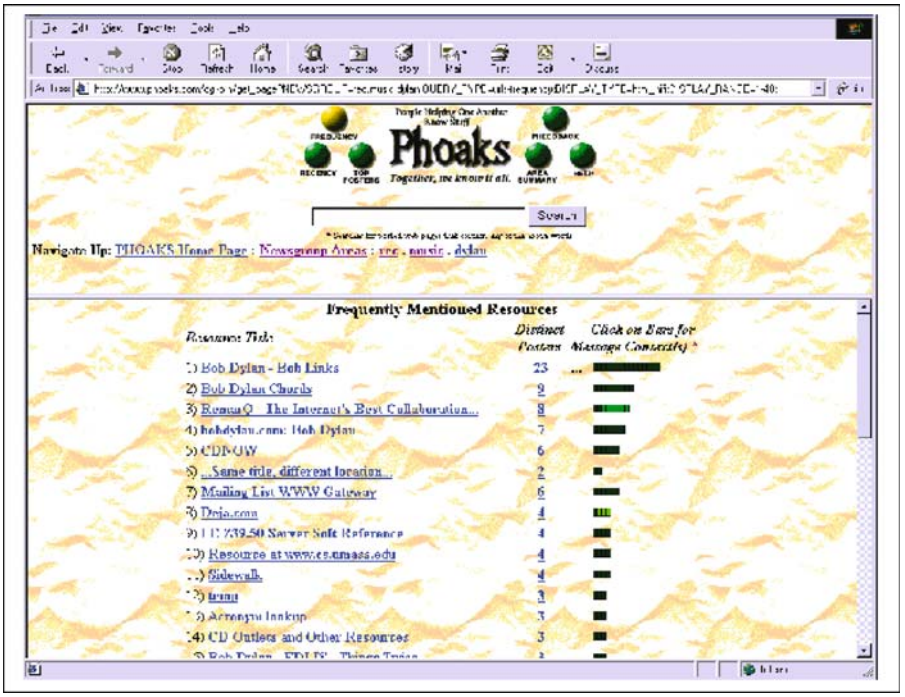


Figure 6.6 A set of recommendations from PHOAKS. Each recommendation shows the number of distinct posters who mentioned the page, and provides links to their messages, a form of explanation (Terveen et al., 1997).

in context. Terveen and Hill (1998) found these data to be useful in classifying sites, specifically for identifying authoritative sites and information hubs. The Google search engine employs a similar technique to rank results returned from a search in an attempt to present the most important sites first; sites that are linked to from many other authoritative sites are considered more important than sites that are sparsely linked to. Hill and Terveen (1996) even found that cross-medium recommendations were effective, mining endorsements of Web pages from Usenet news articles in the PHOAKS system shown in Figure 6.6 (Terveen et al., 1997). Both Terveen and Hill's systems and Google use only information already created by authors for another purpose, benefiting from both the availability and the relative objectivity of implicit ratings.

These implicit ratings could be improved if the meaning of the pages or links was understood better. Did the author link to the page because it was an example of a bad page on a given topic? The Semantic Web might be able to record this information so recommenders could use it. Further, if a user frequently visits Web pages on many sites that have particular Semantic Web tags, that user may like Web pages on other sites that have those same tags. Typical implicit ratings systems would not detect the similarity between these pages, since the URLs are different. Combining the technology for extracting implicit ratings with the Semantic Web's rich data about each page might make possible richer implicit ratings systems that would be even more effective at extracting meaning for users.

6.6 Socially Aware Recommenders

Recommender systems started as systems that used group information to help each individual, but too often these systems ignore the importance of the groups themselves. This section discusses social navigation systems, which make community behavior visible in aggregate, and recommenders for smaller subgroups of a larger community.

6.6.1 Social Navigation

Social navigation systems have emerged as a broad array of techniques that enable people to work together to help each other find their way through crowded spaces. In a social navigation system, each user who visits a Web site does a small amount of work to untangle which of the paths from that Web site are most valuable. Early users leave information signposts that help later users make sense of the wealth of alternatives available to them. Later users benefit from the signposts, because they are able to direct their attention to the parts of the site that are most valuable to them. Some forms of social navigation are very closely related to collaborative filtering. For instance, Höök et al. (2003) discuss ways in which the passage of users through an information space can leave footprints that help other users find their way more readily through that same space (see Figure 6.7). As information spaces become more crowded with users, we may find it important to have automated systems that show us only those footprints that are most useful to us.

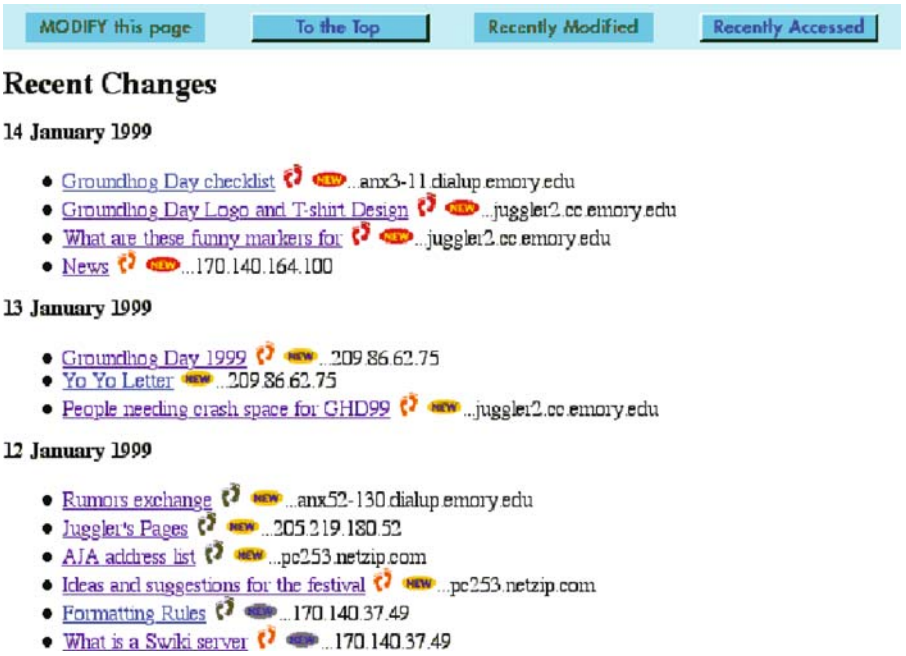


Figure 6.7 Footprints show Web visitors which links the community has followed.

Social navigation may fit into the Semantic Web in many different ways. Most directly, social navigation may extend to systems in which visitors to Web pages explicitly label those pages according to their semantic content. After all, labeling only by the authors of the pages is limiting because the authors are generally at a very different level in their understanding of the content than most visitors. As different visitors label pages, collaborative filtering can be used to select the view of the Semantic Web that is most valuable for each visitor, according to his viewing history and relationships with the previous visitors.

Social navigation can also fit into the Semantic Web by making visible which of the labels on a page seem most appropriate with respect to what visitors to that page actually do. For instance, one page might be labeled as having information on both sailing and windsurfing; however, if all visitors to that page end up following links to windsurfing information, social navigation could adjust the relative strengths of the labels to indicate that windsurfing is a more important label for that page.

Social navigation and the Semantic Web have similar goals. Both provide facilities that make navigating the Web more efficient and successful for users. The two techniques are complementary, and over time we anticipate that they will be used synergistically.

6.6.2 Recommending for Groups

Most automated collaborative filtering systems have focused exclusively on recommending items to individuals. In some domains, such as Usenet News (Resnick et al., 1994; Konstan et al., 1997; Miller et al., 1997), this limitation is understandable. Few users read articles collectively. In other domains such as books or music (Shardanand and Maes, 1995), it is common to enjoy the media both alone and in groups. Moreover certain items, among them board games and movies (Hill et al., 1995), are more commonly enjoyed in groups. Recommender systems that identify items such as movies for individuals do not address the user's key question, which is not "What movie should I see?" but rather "What movie should we see?" Recommendations can be provided to a group either by extending the automatic recommender to automatically recommend for the group, or by creating a new interface that displays individual recommendations for each member of the group, so the users can form their own group recommendation. In either case, the problem requires extending existing recommendation algorithms.

We built such a group recommendation interface for our MovieLens research Web site. We call the group interface PolyLens. PolyLens encourages users to form groups of other MovieLens users, using either the e-mail address or MovieLens pseudonym of the other users. Once a group is formed, each user in the group can get lists of recommended movies with the PolyLens prediction of how much that user will like the movie, and a separate prediction of how much the whole group will like the movie (see Figure 6.8). Overall, our experience has been that users are enthusiastic about group recommenders, and that group recommenders would help users make choices (O'Connor, 2001).

MusicFX is another system that supports group formation, though without using collaborative filtering (McCarthy and Anagnost, 1998). MusicFX was designed to address the challenge of selecting music for the often-large groups of people using a corporate gym. Each person filled out a taste profile describing his or her music preferences. The computer that tracked who was in the gym also kept track of what type of music would be most appropriate for the current group of people who were

Figure 6.8 PolyLens group recommendation display. Iterative user testing showed that users preferred a single list with separate recommendations for the group and for each individual group member.

working out. Over time, some people changed their work habits to arrange to be at the gym when other people, often strangers, with similar musical tastes were there. Figure 6.9 shows the system's view of five workout participants and their taste in music.

Currently most Web navigation is solo. Group navigation is likely to become increasingly important over time, much as multiplayer games are becoming prevalent, and multi-user support is now expected in document authoring systems. For instance, MultECommerce supports a group of shoppers visiting a virtual store together (Puglia et al., 2000). A recommender system in such a store should be integrated into the group experience, perhaps providing recommendations of items for individuals in the group, but basing the recommendations in part on the taste of the entire group. For instance, the recommendations of clothing shown to a group of teenage girls shopping alone might be very different to the clothing recommendations shown to one of the girls shopping with her mother.

Similarly, Maglio et al. (2003) combine the social navigation from the prior section with group formation as a way to make the Web a more social space. They propose a means of connecting individuals as they traverse through the Web. In essence, they

i	*Genre* *Person*	A	B	C	D	E	GP_i	Pr_i
1	Alternative Rock	2	2	0	2	2	68	0.48
2	Hottest Hits	1	1	2	0	-2	38	0.27
3	New Music	1	1	1	0	0	35	0.25
4	Hot Country	2	0	0	0	-2	28	0.00
5	Dance	2	-1	1	-1	-1	28	0.00
6	World Beat	0	1	-1	1	-2	23	0.00
7	Traditional Country	1	0	0	-2	-2	17	0.00
8	50's Oldies	0	0	0	-1	-1	14	0.00
9	Heavy Metal	-1	-1	-1	-1	-2	4	0.00
10	Polka	-1	-1	-2	-2	-2	2	0.00

Figure 6.9 Example preferences from the MusicFX system. Users A–E are the five users currently in the gym. Each of the users A–E has specified her preference for each radio station that could be chosen to be played in the gym. MusicFX has computed an aggregate preference score for each of the radio stations for the five users. The stations are sorted by aggregate preferences. In this case, the top three stations dominate for these five users, so those stations will be played most of the time. Note that MusicFX is not a winner-takes-all system: stations with lower aggregate preference will still be played, though less frequently. (From McCarthy and Anagnost, 1998.)

create a recommender whose recommendations are other individuals who share the same "place" on the Web as the user receiving the recommendations. They discuss a variety of ways to define what it means to share the same place, from the trivial (people sharing the same host organization) to the more complex (people who are browsing the Web within several links of each other). Their belief is that knowledge of such individuals may make the Web a more social place to be.

Ludford et al. (2004) also consider group formation but extend this to include a mechanism to provide for continued group involvement. Historically, and as a whole, online groups have a tendency to go dormant and over time die out. Ludford et al. propose that one reason for this lack of activity is the failure for online groups to generate a proper social structure. Results of their controlled studies in the domain of movies indicate that groups formed from individuals with a dissimilar set of interests actually generated more activity than those formed from similar individuals. Furthermore, they discovered that individuals who understood what made them unique to the community (or at the very least, to the current discussion) would often exploit their uniqueness to contribute to the community or spark additional discussion. Perhaps these results can be used by designers of recommenders to create sites and communities that encourage contributions to the system, and thus, the betterment of the community.

Similar to the changes in recommender interfaces that have been necessary to support groups, the interface to the Semantic Web will have to change to support groups. One such interface might be a recommender supporting a group of people navigating together, and using both collaborative filtering and Semantic Web information to make its recommendations. Such an interface might provide a view of the Semantic Web that would be appropriate to the group of individuals navigating together. Each member of the group might see some pages that are selected to be appropriate for just him, a group view that is appropriate for the group together, and see or hear comments made by group members about the pages. So little work has been done in this area that it is hard to know which interface will be most successful, but experience suggests that making navigating the Web a social experience will be important over the long term.

6.7 Portable Recommenders

A key advantage of recommenders is their helpfulness in suggesting a select subset of available information to be displayed in a limited amount of screen real estate. This advantage of recommenders is even more valuable in small devices, on which the amount of screen real estate is minuscule, and through which users want immediately valuable information without having to browse through a large number of alternatives.

Miller et al. (2003) examined four interfaces to recommendations that could be used on mobile devices: (1) WAP, a Web technology for cell phone screens; (2) Avant-Go, a technology for synchronizing selected Web pages to a PDA; (3) live HTML on a PDA; and (4) a VoiceXML interface for speaking to the recommender over a telephone. The overall results were that users very much liked the portable recommenders, because they could get recommendations right when and where they needed them. On the bad side, each of the four interfaces required significant design particular to the device and interface technology used. A generalized framework for developing small-device information browsing interfaces would be invaluable, though we seem many years of research away from such a framework. Users also found significant advantages to the

Internet-connected recommenders, because those interfaces had all of the information available, and had the most up-to-date information. However, users of these interfaces found connectivity frustrating, since wide-area wireless Internet networks are far from ubiquitous and reliable.

One solution to the problem of connectivity is to create a recommender algorithm that can run directly on the small device. Miller et al. (2004) explored recommender algorithms that could compute recommendations on a PDA-sized computer in real-time. They were able to show that such algorithms can produce accuracy nearly as good as the best workstation-based recommenders, with a small memory footprint, and fast interactive performance. The key insight is to specialize the model stored on the PDA for the single user of that PDA. The authors postulate that such a recommender frees a user from the influence of the operator of a centralized recommender, who may have goals (such as selling overstock products) that are not shared by the user.

Access to information on small devices seems likely to be increasingly important. Recommenders can be a valuable part of small-device information browsers, because they can dramatically reduce the amount of information the user must sift through to find the items he is interested in. Developing such recommender algorithms on large computers is relatively straightforward with modern technology, and the resulting interfaces can be displayed on the portable devices. If the device is not always connected to the Internet when information is desired, it may be possible to develop new algorithms that can perform the recommender tasks on the small device.

6.8 Cheating with Recommenders

One important question in recommender systems is the extent to which users can manipulate the system by putting in ratings that are not representative of their real preferences. For instance, a record label might pay a thousand users to falsely claim to love a newly released single. There are many examples of this sort of manipulation in real-world recommenders (e.g., Sony using fake quotes from a made-up film critic to promote its films (BBC News, 2001)), so it seems likely to happen with online recommenders, too. Several empirical studies have explored the effectiveness of putative attacks on recommender systems. O'Mahoney et al. (2003) showed that attacks on kNN users-user algorithms could both *push* items by raising predicted ratings, and *nuke* items by lowering predicted ratings. Lam et al. [2] demonstrated that the kNN item-item algorithm seems somewhat less susceptible to these attacks—though this paper also showed that the methods recommender systems researchers use to look for these attacks are not sensitive even to very harsh attacks on the system. Much more research is needed to understand how to identify attacks when they occur, and—crucially—how to protect recommender systems from such attacks. Similar attacks are likely on semantic Web sites that use information from the community to label the sites.

6.9 Conclusion

Recommender systems already provide substantial user value by personalizing a number of sites on the Web. The Semantic Web brings forward rich opportunities for improving these interfaces, and for striking a better balance between content and collaborative personalization methods.

6.10 Acknowledgments

We gratefully acknowledge the contributions of the students who are current and alumni members of the GroupLens Research Group at the University of Minnesota, without whom this work could not have been done. This work was supported by the National Science Foundation under grants IIS 9613960, IIS 9734442, IIS 9978717, and DGE 9554517. Support was also provided by Net Perceptions Inc., a company that we co-founded that sells an automatic collaborative filtering system.

6.11 References

Balabanovic, M., Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3).

BBC News (2001). *Sony Admits to Using Fake Reviewer.* Retrieved from http://news.bbc.co.uk/1/hi/entertainment/film/1368666.stm.

Bharat, K., Kamba, T., Albers, M. (1998). Personalized, interactive news on the Web. *Multimedia Systems*, 6(5):349–358.

Breese, J., Heckerman, D., Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, Morgan Kaufmann, pp. 43–52.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. In: *User Modeling and User-Adapted Interaction*, 12:331–370, Kluwer Academic Publishers.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. *Proceedings of the SIGIR 1999 Workshop on Recommender Systems: Algorithms and Evaluation.* Available: http://www.cs.umbc.edu/~ian/sigir99-rec/.

Cosley, D., Lam, S.K., Albert, I., Konstan, J.A., Riedl, J. (2003). Is seeing believing? How recommender systems influence users' opinions. *Proceedings of CHI 2003 Conference on Human Factors in Computing Systems*, Fort Lauderdale, FL, pp. 585–592.

Cranor, L. (2002). *Web Privacy with P3P.* O'Reilly & Associates.

Deshpande, M., Karypis, G. (2004). Item-based Top-N recommendation algorithms. *ACM Transactions on Information Systems,* 22(1):143–177.

Goldberg, D., Nichols, D., Oki, B.M., Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. *Proceedings of AAAI-99*, AAAI Press, pp. 439–446.

Herlocker, J., Konstan, J.A. (2001). Content-independent task-focused recommendation. *IEEE Internet Computing*, 5(6).

Herlocker, J., Konstan, J.A., Borchers, A., Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of SIGIR'99*, ACM, pp. 230–237.

Herlocker, J., Konstan, J.A., Riedl, J. (2000). Explaining collaborative filtering recommendations. *Proceedings of the ACM 2000 Conference on Computer-Supported Cooperative Work*, ACM, pp. 241–250.

Herlocker, J., Konstan, J.A., Terveen, L., Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53.

Hill, W., Terveen, L. (1996). Using frequency-of-mention in public conversations for social filtering. *Proceedings of the ACM 1996 Conference on Computer-Supported Cooperative Work*, ACM, pp. 106–112.

Hill, W., Stead, L., Rosenstein, M., Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, ACM, pp. 194–201.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems,* 22(1):89–115.

Höök, K., Benyon, D., Munro, A. (2003). Footprints in the snow. In: Höök, K., Benyon, D., Munro, A. (Eds.), *Social Navigation of Information Space.* Springer-Verlag, London.

Jin, R., Si, L., Zhai, C., Callan, J. (2003). Collaborative filtering with decoupled models for preferences and ratings. *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, ACM, pp. 309–316.

Konstan, J.A., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J. (1997). GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87.

Lam, S.K., Riedl, J. (2004). Shilling Recommender systems for fun and profit. *Proceedings of WWW 2001*, WWW10 Limited, pp. 393–394.

Ludford, P., Cosley, D., Frankowski, D., Terveen, L. (2004). Think different: Increasing online community participation using uniqueness and group dissimilarity. *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, ACM, pp. 631–638.

Maglio, P., Barrett, R., Farrell, S. (2003). WebPlaces: Using intermediaries to add people to the Web. In: Höök, K., Benyon, D., Munro, A. (Eds.), *Social Navigation of Information Space*s. Springer-Verlag, London.

O'Mahony, M.P., Hurley, N., Kushmerick, N., Silvestre, G. (2003). Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology, 2003.* Special Issue on Machine Learning for the Internet, 4(3).

Maltz, D., Ehrlich, E. (1995). Pointing the way: Active collaborative filtering. *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, ACM, pp. 202–209.

McCarthy, J., Anagnost, T. (1998). MusicFX: An arbiter of group preferences for computer supported collaborative workouts. *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, ACM, pp. 363–372.

McLaughlin, M., Herlocker, J. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *Proceedings of the ACM SIGIR'04 Conference*, ACM, pp. 329–336.

McNee, S., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S.K., Rashid, A.M., Konstan, J.A., Riedl, J. (2002). On the recommending of citations for research papers. *Proceedings of 2002 Conference on Computer Supported Cooperative Work*, ACM, pp. 116–125.

McNee, S., Lam, S.K., Konstan. J.A., Riedl, J. (2003a). Interfaces for eliciting new user preferences in recommender systems. *Proceedings of the 9th International Conference on User Modeling (UM'2003)*, LNAI 2702, pp. 178–188.

McNee, S., Lam, S.K., Guetzlaff, C., Konstan, J.A., Riedl, J. (2003b). Confidence displays and training in recommender systems. *Proceedings of INTERACT '03 IFIP TC13 International Conference on Human-Computer Interaction*, pp. 176–183.

Middleton, S., Shadbolt, N., De Roure, D. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88.

Miller, B., Riedl, J., Konstan, J.A. (1997). Experiences with GroupLens: Making Usenet useful again. *Proceedings of the 1997 Usenix Winter Technical Conference*, USENIX, January 1997.

Miller, B., Albert, I., Lam, S.K., Konstan, J.A., Riedl, J. (2003). MovieLens unplugged: Experiences with a recommender system on four mobile devices. *Proceedings of the 17th Annual Human-Computer Interaction Conference (HCI 2003)*, British HCI Group, September 2003.

Miller, B., Konstan, J.A., Terveen, L., Riedl, J. (2004). PocketLens: Towards a personal recommender system. *ACM Transactions on Information Systems*, 22(3):437–476, July 2004.

Morita, M., Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. *Proceedings of the 17th Annual International SIGIR Conference on Research and Development*, ACM, pp. 272–281.

O'Connor, M., Cosley, D., Konstan, J.A., Riedl, J. (2001). PolyLens: A recommender system for groups of users. *Proceedings of ECSCW 2001*, Kluwer Academic Publishers, Bonn, Germany.

Puglia, S., Carter, R., Jain, R. (2000). MultECommerce: A distributed architecture for collaborative shopping on the WWW. *Proceedings of the 2nd ACM Conference on Electronic Commerce*, ACM, Minneapolis, MN, pp. 215–224.

Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S., Konstan, J.A., Riedl, J. (2002). Getting to know you: Learning new user preferences in recommender systems. *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, ACM, pp. 127–134.

Resnick, P., Miller, J. (1996). PICS: Internet access controls without censorship. *Communications of the ACM*, 39(10):87–93.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, ACM, pp. 175–186.

Sarwar, B., Konstan, J.A., Borchers, A., Herlocker, J., Miller, B., Riedl, J. (1998). Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. *Proceedings of 1998 Conference on Computer-Supported Collaborative Work*, ACM.

Sarwar, B., Karypis, G., Konstan, J.A., Riedl, J. (2000). Analysis of recommender algorithms for e-commerce. *Proceedings of the ACM E-Commerce 2000 Conference*, ACM.

Sarwar, B., Karypis, G., Konstan, J.A., Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of WWW 2001*, WWW10 Limited.

Schafer, J.B., Konstan, J.A., Riedl, J. (2001). Electronic commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 5(1/2):115–152.

Schafer, J.B., Konstan, J.A., Riedl, J. (2002). Meta-recommender systems: User-controlled integration of diverse recommendations. *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, ACM, pp. 43–51.

Schafer, J.B., Konstan, J.A., Riedl, J. (2005). The view through MetaLens: Usage patterns for a meta-recommender system. *IEE Proceedings Software*, IEE.

Shardanand, U., Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth." *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, ACM, pp. 210–217.

Terveen, L., Hill, W. (1998). Finding and visualizing inter-site clan graphs. *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems*, ACM, pp. 448–455.

Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J. (1997). PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62.

Torres, R., McNee, S., Abel, M., Konstan, J.A., Riedl, J. (2004). Enhancing digital libraries with TechLens. *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries*, ACM, pp. 228–236.

Wolf, J., Aggarwal, C., Wu, K-L, Yu, P. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Diego, CA.

Chapter 7

SVG and X3D: New XML Technologies for 2D and 3D Visualization

Vladimir Geroimenko and Larissa Geroimenko

7.1 Introduction

On the Semantic Web, everything tends to be written in XML-based plain text format. Web graphics are not an exception. Both two-dimensional and three-dimensional graphics will be presented in XML-based formats, called SVG (Scalable Vector Graphics) and X3D (Extensible 3D Graphics) respectively.

A pioneering research monograph, "Visualizing Information Using SVG and X3D: XML-Based Technologies for the XML-based Web," was published in 2004 by Springer (Geroimenko and Chen, 2004) as a conceptual sequel to the first edition of the book *Visualizing the Semantic Web: XML-Based Internet and Information Visualization* (Geroimenko and Chen, 2002). It explores the use of SVG and X3D in generic Web applications and the applications of these novel technologies to specific problems. All these areas are considered in the broader context of the XML family of technologies and the emerging Semantic Web.

The aim of this chapter is to provide the reader with a basic introduction to new technologies and their role in the Semantic Web. For much more detailed analysis, see Geroimenko and Chen (2004).

SVG and X3D are leading-edge technologies primarily because they are native to the XML-based Web. That means that they can be easily and seamlessly integrated with practically any other XML technologies, including ones dealing with semantics, ontologies, and metadata. Moreover, XML desperately needs these native technologies for representing Web data in a visually rich form since one of the fundamental principles behind XML is the separation of data from presentation rules. Basically, XML documents contain nothing but pure data, and therefore the use of additional technologies is required to go beyond graphically primitive representations of XML data.

7.2 SVG

Scalable Vector Graphics (SVG) is the newest language for describing two-dimensional graphics in XML. Its fist draft was issued by W3C (The World Wide Web Consortium) in February 1999 and its current specification (SVG 1.1) was released as a W3C Recommendation (i.e., a Web standard) in January 2003. The SVG 1.2 specification is

currently being developed and is available in W3C Working Draft form (published on 27 October 2004). The latest, up-to-the-moment information about this technology can be found on the W3C Web site (http://www.w3.org/Graphics/SVG/). Some books and online articles specially devoted to SVG are also available (Teague and Campbell, 2003; Cagle, 2002; Eisenberg, 2001; 2002; Pearlman and House, 2002; Quint, 2001; Watt, 2001; Thomas, 2000).

SVG is completely based on XML and this is its main advantage. It lets authors handle Web graphics the same way as text and data, using XML as the universal format. SVG benefits from its tight integration with other members of the XML family of technologies such as DOM (Document Object Model), CSS (Cascading Style Sheets), XSL (Extensible Stylesheet Language), SMIL (Synchronized Multimedia Integration Language), RDF (Resource Description Framework), OWL (Web Ontology Language), XLink, XPointer, etc.

Since SVG conforms to the DOM, the JavaScript language can be used to access and manipulate any SVG file components at runtime. Web users are able to interact with an SVG image on the basis of the same event model as for HTML pages (i.e., to use functions such as "onClick" or "onMouseOver"). It is important to note that every element (a shape or text) and every attribute of an element is accessible and can be manipulated, for example, animated. XSLT (XSL Transformations) allows the creation of SVG documents on the fly and their transforming into any desired format. The use of RDF and OWL makes it possible to add metadata to SVG files and describes them in the same way as any other text or data resources on the Semantic Web, which therefore can be searched, "comprehended," and processed by computers. SMIL is an XML-based language that allows integrating SVG with other Web resources into an interactive multimedia presentation. XLink and XPointer provide mechanism for linking from within SVG files to other SVG files, SMIL presentations, HTML pages, and other documents on the Web.

SVG supports three types of graphics objects (vector graphic shapes, images, and text) that can be grouped, styled, transformed, and composed with other rendered objects. It contains six predefined objects—basic shapes: rectangle `<rect>`, circle `<circle>`, ellipse `<ellipse>`, polyline `<polyline>`, polygon `<polygon>`, and path `<path>`.

If you are familiar with XML, an SVG file is easy to read. Its editing and hand-coding using any ordinary text editor is much more intuitive than writing HTML and can be fun.

Consider an example. The file "Hello.svg" contains the following code that defines an SVG drawing shown in Figure 7.1:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="17cm" height="14cm">
     <rect style="fill:red;" x="4cm" y="1.5cm"
width="3cm" height="3cm"/>
     <rect style="fill:yellow; stroke:navy; stroke-
width:0.5cm;" x="9cm" y="1cm" width="5.5cm"
height="2.5cm"/>
     <ellipse style="fill:green;" cx="8cm" cy="9cm"
rx="5.5cm" ry="2.5cm"/>
```

```
    <text style="fill:black; font-family:Verdana; font-
size:24pt;" x="3cm" y="6cm">Hello, SVG World!</text>
</svg>
```

Since SVG is an XML-based language, the first line of the SVG file is the declaration that identifies it as an XML file. The next line provides version information and the URL of the Document Type Definition (DTD)—a standard set of rules for writing SVG. The `<svg>` tag tells a browser or standalone viewer that this file is an SVG document and also includes the attributes `width` and `height` that define the canvas of the SVG document. All the SVG content is placed between the `<svg>` and `</svg>` tags. The `<rect/>` tag is the code that defines a rectangle and declares its properties: the `width` and `height`, the `fill` (the fill color of the rectangle), the `stroke` (an outline for the rectangle, its width and color). It is interesting to note that SVG uses ordinary Cascading Style Sheet syntax within the `style` attribute. The `<ellipse/>` tag defines an ellipse by the coordinates of its center and by its radius in a horizontal (`rx`) and vertical (`ry`) direction. The `<text>` tag contains text as character data and defines its fill color, font family, font size, and position on the screen (using the `x` and `y` attributes).

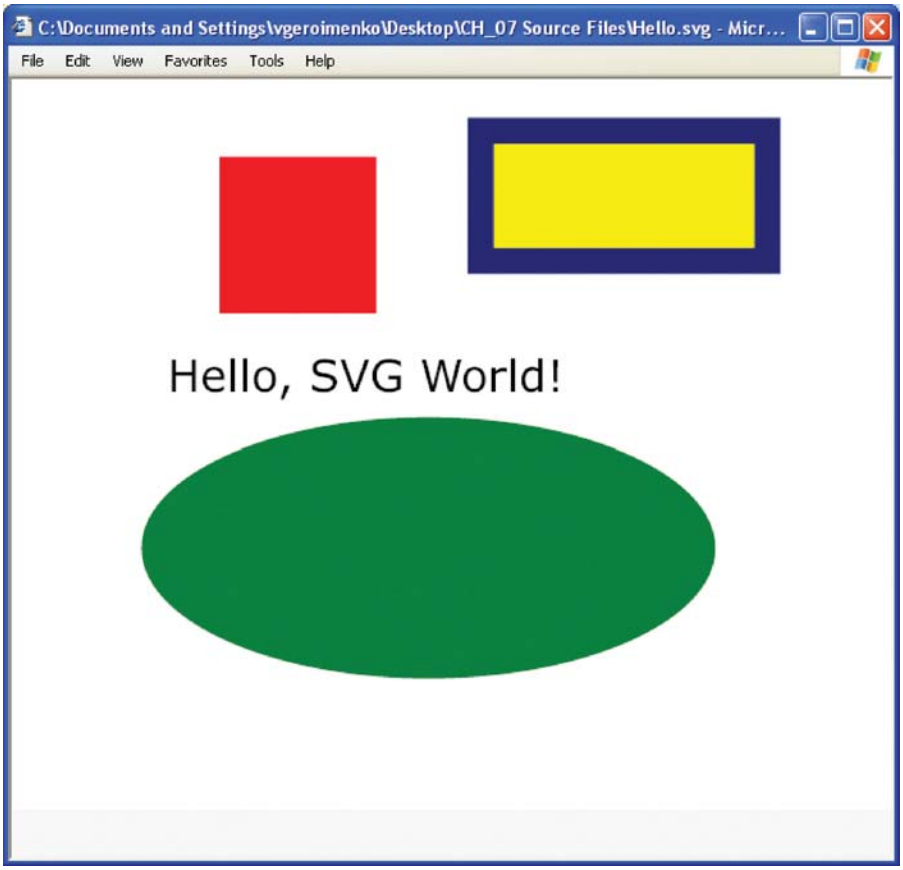


Figure 7.1 A sample SVG drawing.

Thus, SVG is the language of choice for 2D visualizations on the Semantic Web. Both SVG tags and ontology-related metadata tags are readable and understandable not only by humans but also by computers, making SVG drawings suitable for automated search and processing. Any element and attribute of an SVG document can be accessed and changed programmatically using any XML-enabled software and technologies. Tight integration with DOM, XSL, SMIL, RDF, OWL, other current or future XML technologies, and also any XML document makes SVG visualizations into an integral part of the Semantic Web.

7.3 X3D

Many Web developers believe that the Second-Generation Web will implement not only semantics but also 3D. "Web3D" is a term that refers to all technologies that enable interactive 3D graphics on the Web, including open and proprietary formats. The most comprehensive explanation of a wide range of Web3D technologies is provided by Walsh and Bourges-Sévenier (2001) in their 1,088-page book. The latest information about all aspects of Web3D technologies can be found on the Web site of the Web3D Consortium (www.web3d.org). The Consortium was formed to provide a forum for the creation of open standards for Web3D specifications.

The first successful specification developed by the Web3D Consortium was VRML (Virtual Reality Modeling Language) and this is why the Consortium was formerly called the VRML Consortium. For a long time, VRML was the leading technology for the presentation of three-dimensional images on the Web. As a simple language for publishing 3D Web pages, VRML is a 3D analogue to HTML. The first version (VRML 1.0) was released in May 1995; the second version (VRML 2.0) was issued in August 1996 and became an international standard (known informally as VRML97) in December 1997.

In 2004, VRML97 was superseded by X3D, an open standard for 3D content delivery that is the next revision of the VRML97 specification, incorporating the latest advances in commercial graphics hardware features as well as architectural improvements based on years of feedback from the VRML97 development community (X3D FAQ, 2004). It is important to note that the X3D specification includes the following three encoding formats of the 3D scene graph: the XML encoding (X3D as such), the UFT-8 encoding (VRML97 style encoding), and the binary encoding. Specialist tools enable an automated translation between these encoding formats.

As a more advanced successor to VRML, X3D provides Web developers with many new features, including extensibility, modular architecture, improved APIs, and additional data encodings. It allows the use of 2D and 3D graphics, spatialized audio and video, animation, user navigation and interaction, user-defined objects, scripting, and networking. In other words, the new version will go beyond the limits of VRML97 by adding new graphical, behavioral, and interactive objects (Hayes, 2001; Ressler, 2000a; 1999).

Let's consider an example to see the differences between the VRML97 encoding and the X3D encoding of a scene graph that defines a simple VR world shown in Figure 7.2. This navigable 3D world consists of a red sphere and two lines of text and also includes nodes that define the user's initial viewpoint and allowed types of navigation. The `Group` and `children` nodes are used to group other nodes. The `Transform` node places the text in the desired position within the virtual world.

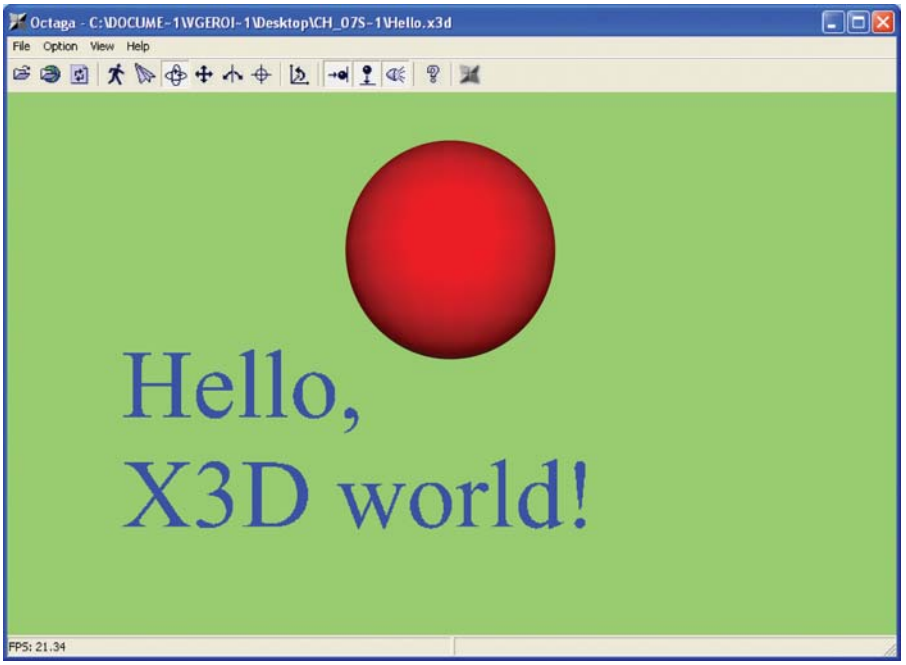


Figure 7.2 A sample X3D world.

Although the VRML code is pretty easy to understand, generally speaking it is foreign to Web developers with the exception of the VRML community.

```
#VRML V2.0 utf8
NavigationInfo {
      type [ "EXAMINE" "ANY" ]
    }
Background {
      skyColor [0.5 1 0.5]
    }
Viewpoint {
      orientation 0 1 0 1.57
      position 6 -1 0
    }
Group {
  children [
    Shape {
      geometry Sphere {
      }
      appearance Appearance {
     material Material {
       diffuseColor 1 0 0
     }
      }
    }
```

```
    Transform {
      rotation 0 1 0 1.57
      translation 0 -1.5 3
      children [
     Shape {
       geometry Text {
         string [ "Hello," "X3D world!" ]
       }
       appearance Appearance {
         material Material {
           diffuseColor 0 0 1
         }
       }
     }
      ]
    }
  ]
}
```

The same virtual world can be encoded using X3D as shown below. In this case, the audience that is able to read and understand the code is much wider because it includes everyone who is familiar with XML (and also XML-enabled software, tools and autonomous agents). The following code illustrates the use of X3D tags and attributes. Similar to an SVG or any other XML file, it starts with the XML declaration and the Document Type Definition. The VRML nodes from the above VRML97 example are presented as X3D tags written in XML and node fields as tag attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D>
   <head>
   </head>
   <Scene>
     <NavigationInfo type="&quot;EXAMINE&quot;
&quot;ANY&quot;"/>
     <Background skyColor='0.5, 1, 0.5'/>
     <Viewpoint orientation="0 1 0 1.57" position="6 -1
0"/>
     <Group>
        <Shape>
            <Sphere/>
            <Appearance>
               <Material diffuseColor="1 0 0"/>
            </Appearance>
        </Shape>
     <Transform rotation="0 1 0 1.57" translation="0 -
1.5 3">
        <Shape>
          <Text string="&quot;Hello,&quot; &quot;X3D
```

```
world!&quot;"/>
          <Appearance>
             <Material diffuseColor="0 0 1"/>
          </Appearance>
        </Shape>
     </Transform>
   </Group>
  </Scene>
</X3D>
```

Hundreds of similar examples are available on the Web site of the Web3D Consortium at http://www.web3d.org/x3d/content/examples/, including the entire set of VRML examples from the VRML 2.0 Sourcebook (Ames, Nadeau, and Moreland, 1997) translated into X3D.

X3D provides tight integration between 3D technology and the XML family of standards such as XML Schema, XML DOM, XSLT, or SVG. Since X3D is written in XML, an X3D document can be validated against an appropriate XML Schema. It can be transformed into other XML documents (for example, SVG) using XSLT and its X3D data can be assessed and manipulated using the DOM. Similar to SVG, XML has many other advantages of being an XML-based language; for example, an X3D world can be viewed on any device with any screen size. In other worlds, "the future of the Web is XML and the future for 3D on the Web is X3D" (Ressler, 2000b). To continue this thought, it is also right to add that the future for 2D on the Web is SVG. X3D and SVG technologies, based entirely on XML, open essentially new possibilities for the visualization of the Semantic Web.

7.4 The Use and Advantages of SVG and X3D

SVG and X3D are the main members of the XML family of technologies intended for authoring two- and three-dimensional Web graphics respectively. They allow developers to create visually rich and functionally smart interfaces to XML documents of different kinds. At the same time, other languages and technologies can also be used for the visualization and rendering of Web data. Some of them are based on XML (such as XHTML or SMIL); some are not (such as Flash or Java). Why are SVG and X3D unique? They are the principal XML-based technologies for 2D and 3D graphics. Moreover, they can be used together with practically any other members of the XML family, providing unprecedented new opportunities for Web graphics.

Since most members of the XML family are still under development, the opportunities provided by their mutual use are yet to be explored. But theoretically any XML document (including SVG and X3D ones) may be embedded or transformed into another document, because all of them are written in the same language. Figure 7.3 shows different ways of presenting domain-specific XML documents in 2D, 3D, and multimedia form.

SVG and X3D play a central technological role in visualizing the Semantic Web (Geroimenko and Chen, 2004; Geroimenko, 2004; Geroimenko and Geroimenko, 2000; Lau et al., 2003; Cleary and O'Donoghue, 2001). On the one hand, metadata and Web ontologies can be embedded into SVG and X3D files, turning them into "smart" machine-understandable graphics. On the other hand, SVG and X3D can effectively be

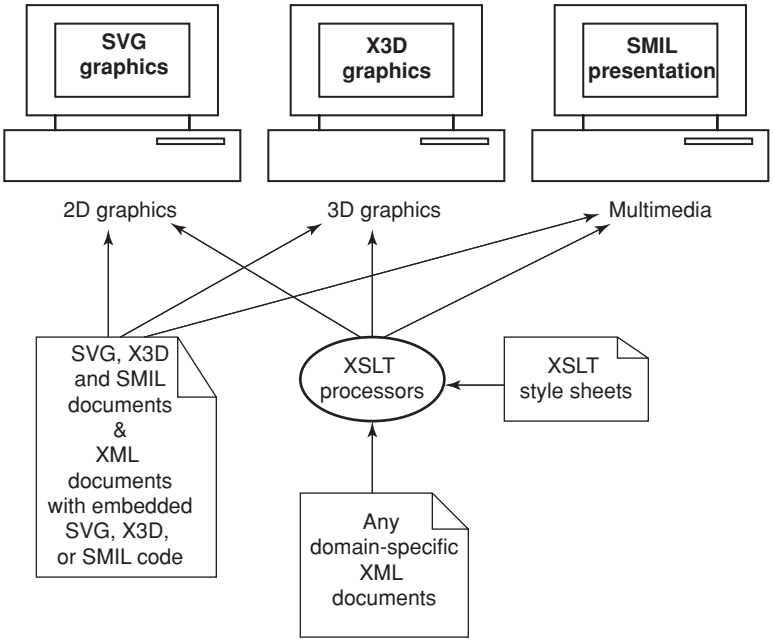


Figure 7.3 Visualizing XML data using SVG, X3D, SMIL, and XSLT.

used for visualizing metadata, ontologies, and other conceptual structures of human meanings. This "two-way" relationship is illustrated in Figure 7.4.

Many advantages of the SVG and X3D technologies are thoroughly explored in Geroimenko and Chen (2004). Here are some of beneficial features of SVG and X3D graphics:

- High-quality Web graphics with precise structural and visual control
- Dynamically generated graphics driven by real-time XML data

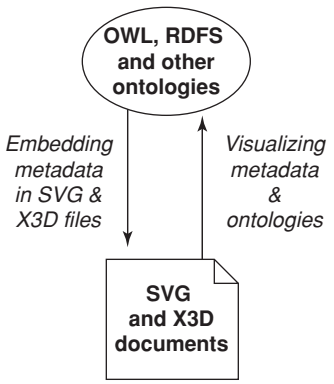


Figure 7.4 Relationships between SVG/X3D and metadata/ontologies.

- Interactive and scriptable graphics utilizing DOM, JavaScript, XSLT, Java, Visual Basic, etc.
- Graphics tightly integrated with other members of the XML family
- Personalized graphics easily customizable to all users
- Graphics automatically optimizable for mobile phones, PDAs, etc.
- Easily updateable graphics (since design is separated from content)
- Custom-zoomable graphics that allow users to drill down to additional levels of details
- Graphics that enable internationalization and localization (the use of many of the world's languages)
- Graphics in text-based human-readable format
- Open, royalty-free standards

Since both SVG and X3D are written in XML, they might be integrated with each other. For example, SVG can be extended into three dimensions. In this case, some surfaces within an X3D world will be able to act as rendering canvases for SVG graphics. Theoretically, the mix of SVG and X3D syntaxes opens new possibilities for developers to create really exciting and useful "all-in-one" visualization applications.

Thus, SVG and X3D are the two current Web graphic standards that will play an increasingly important role in visualizing information and creating graphical interfaces for the next generation of the Web. They are XML-based graphics of choice for the emerging Semantic Web, since they allow dynamic visualization of XML documents, tight and seamless integration with each other and numerous XML-based technologies, and the use of metadata, ontologies, and other Semantic Web technologies.

7.5 References

Ames, A., Nadeau, J., Moreland, J. (1997). *VRML 2.0 SourceBook*. John Wiley & Sons. (X3D versions of VRML 2.0 Sourcebook Files available at http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook/.)

Cagle, K. (2002).*SVG Programming: The Graphical Web*. Apress.

Cleary, D., O'Donoghue, D. (2001). Creating a Semantic Web interface with virtual reality. *Proceedings of the SPIE—The International Society for Optical Engineering*, 4528:138–146

Eisenberg, D. (2001). *An Introduction to Scalable Vector Graphics*. Available: http://www.xml.com/pub/a/2001/03/21/svg.html.

Eisenberg, D. (2002). *SVG Essentials*. O'Reilly & Associates.

Hayes, H. (2001). 3D comes into focus on the Web. *Federal Computer Week*, March 26. Available: http://fcw.com/fcw/articles/2001/0326/tec-3d-03-26-01.asp.

Geroimenko, V., Chen, C. (Eds.) (2004). *Visualizing Information Using SVG and X3D: XML-Based Technologies for the XML-Based Web*. Springer, Berlin-Heidelberg-New York.

Geroimenko, V., Chen, C. (Eds.) (2002). *Visualizing the Semantic Web: XML-Based Internet and the Information Visualization*. Springer, Berlin-Heidelberg-New York.

Geroimenko, V. (2004). *Dictionary of XML Technologies and the Semantic Web*. Springer, Berlin-Heidelberg-New York.

Geroimenko, V., Geroimenko, L. (2000). Visualizing human consciousness content using Java3D/X3D and psychological techniques. *2000 IEEE Conference on Information Visualization: An International Conference on Computer Visualization and Graphics*, pp. 529–532.

Lau, R.W., Li, F., Kunii, T.L., Guo, B., Zhang, B., Magnenat-Thalmann, N., Kshirsagar, S., Thalmann, D., Gutierrez, M. (2003). Emerging Web graphics standards and technologies. *IEEE Computer Graphics and Applications*, 23(1):66–75.

Pearlman, E., House, L. (2002). *Developing SVG-Based Web Applications*. Prentice Hall PTR.

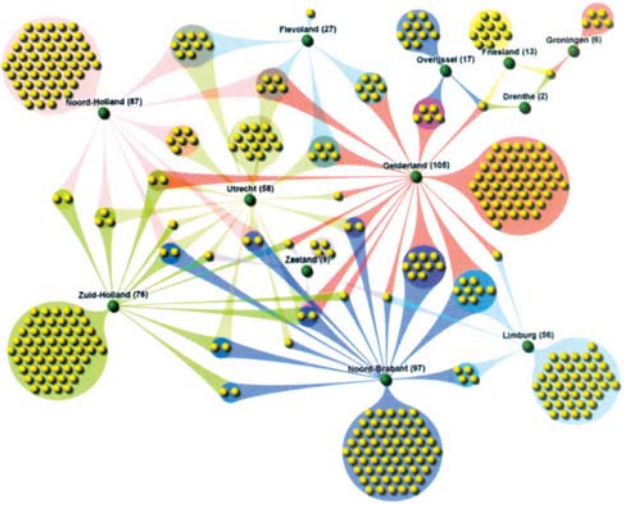
Quint, A. (2001). *SVG: Where Are We Now?* Available: http://www.xml.com/pub/a/2001/11/21/svgtools.html.

Ressler, S. (1999). *X3D, Next Generation VRML*. Available: http://web3d.about.com/library/weekly/aa070799.htm.

Ressler, S. (2000a). *A Simple Introduction to X3D*. Available: http://web3d.about.com/library/weekly/aa080900a.htm.
Ressler, S. (2000b). *Looking at 3D and VRML*. Available: http://web3d.about.com/library/weekly/aa112900a.htm.
Teague, J., Campbell, M. (2003). *SVG for Web Designers*. John Wiley & Sons.
Thomas, K. (2000). *Extreme SVG: A Programmer's Guide to Scalable Vector Graphics*. TBO Books.
Walsh, A., Bourges-Sévenier (2001). *Core Web3D*. Prentice Hall PTR.
Watt, A. (2001). *Designing SVG Web Graphics: Visual Components for Graphics in the Internet Age*. New Riders Publishing.
X3D FAQ (2004).*X3D FAQ*. Available: http://www.web3d.org/x3d/faq/index.html.

PART 2

Visual Techniques and Applications for the Semantic Web



Chapter 8

Using Graphically Represented Ontologies for Searching Content on the Semantic Web

Leendert W.M. Wienhofen

8.1 Introduction

The Semantic Web (Berners-Lee, 1998) is a revolution for machine-understandability of Web pages, yet for the typical kind of user, the nontechnical one, the benefits may not be as obvious as for researchers. In order to enable "naive" users to benefit from the Semantic Web, this chapter proposes a search paradigm using graphical ontologies to retrieve content. Retrieval problems started when the Internet became available for everyone. The ease of publishing led to an abundance of mostly unstructured data, since HTML is meant to display content for humans and not machines. If we wish that all Web pages become Semantic Web enabled, publishing needs to be as easy as it currently is, and retrieval methods need to be as easy as they currently are, but of course the relevance of the retrieved content needs to be much better.

The paradigm presented, called GODE (Graphical Ontology Designer Environment) (Wienhofen, 2003), gives users the possibility to search both the Web and the Semantic Web.

This chapter describes how to prepare users for the new flow of information, by introducing them to the concept of graphical search step by step. A benefit of using graphical search is that it is query language independent. Users have a uniform method of accessing information; a conversion algorithm can be made and used as a plug-in for the search language for each query language available. A variety of difficulty levels are identified to make sure that everybody can benefit from this approach in different situations. Application areas are discussed for both the simple and the advanced version of GODE.

8.2 Visual Query Languages

An experimental proof by Catarci and Santucci (1995) shows that QBD* (Query By Diagram*), a visual query language, is easier to use and gives better results than text based SQL queries. The experiment defined three groups of users: naive, medium, and expert. All three groups got better results faster by using this visual approach.

Other visual query languages, such VISUAL (Balkir et al., 2002) and GLASS (Ni and Ling, 2003), are available. Even though most are designed for use with databases

or XML files, the type of use presented in this chapter is not that much different, as most Semantic Web languages are XML based and define semantic relations. Database queries (SQL) are based on relations, and XML file queries are done on structured data. The available visual query languages, however, are generally not focused toward the naive users, though they are no doubt the largest group of users. This chapter presents a search method that is aimed at the naive user, yet having enough possibilities for it to be useful to expert users as well. In fact, it is built up with the goal that naive users gradually can become medium-level users and eventually expert users (Wienhofen, 2004).

8.3 The Graphical Ontology Designer Environment

Different building blocks and ideas are presented, which are used as a foundation for building the Graphical Ontology Designer Environment (GODE). An ontology is most often defined in these words: "An ontology is an explicit specification of a conceptualisation" (Gruber, 1993). In this chapter, the functionality and capabilities of GODE are described, and some preliminary parts of what the Graphical User Interface (GUI) will look like are included. The problem with existing ontology development tools is that they are mostly aimed at the knowledge engineer, and not the "normal" Web user (who most likely is not even aware of any Semantic Web yet). The same goes for the textual query languages used on the Semantic Web, such as XQL (XML Query Language), RQL (RDF Query Language), TMQL (Topic Map Query Language), OWL-QL (OWL Query Language), etc. For users who are used to typing some search words and then being presented with a list of results, these approaches are far too difficult and unintuitive. GODE will make sure that these users as well may benefit fully from the Semantic Web.

Perhaps not as easy as simply typing the search concepts and pushing a button, is carrying out a search based on a graphically presented ontology. We introduce GODE as a platform for this type of search, along with the enabling technologies, possibilities, and potential dangers.

8.3.1 Enabling Technologies

This section briefly describes how ontologies can be represented both textually and graphically and shows four tools that can serve as the building blocks for GODE. During the OnToKnowledge project (Fensel et al., 2000; 2002), a project in the European Union's Information Society Technologies (IST) Program for Research, Technology Development & Demonstration under the 5th Framework Program, a whole range of different tools have been used and developed. Two of these tools are OntoExtract (Engels and Bremdal, 2001) and the Concept Graph Viewer, which is usually called the CCA (Core Concept Area) viewer. OntoExtract has been developed by CognIT a.s, Norway, and the CCA viewer by AIdministrator, the Netherlands. The CCA viewer has been used in the OTK project only to a very limited degree, and has never left the project boundaries, though it is referred to in Davies et al. (2003). These two tools are the predecessors of the Intelligent Components Kernel and the visualization/editing tool that goes by the codename "Connector."

8.3.1.1 Ontology Representation

Ontologies can be represented using various methods, in XML, RDF (Brickley et al., 2003), OWL, with Topic Maps, etc. However, these are text-based and usually rather voluminous. In addition, ontologies can also be visualized using UML, hyperbolic view, or as a tree, although for representing them in a graphical way the spring embedder algorithm (Eades, 1984) provides a method that is easy for anyone to understand.

The spring embedder algorithm is a heuristic approach to graph drawing based on a mechanical system in which a graph's edges are replaced by springs and vertices are replaced by rings. From the initial configuration of ring positions, the system oscillates until it stabilizes at a minimum-energy configuration (Kumar et al., 1996). Examples of visualizations by means of the spring embedder algorithm can be found in the next section.

8.3.1.2 CORPORUM® OntoExtract

CORPORUM® OntoExtract (with the ontology generator as a GUI), usually referred to as just OntoExtract (Engels and Bremdal, 2001), is based on the second version of CognIT's MÍMÍR core (Engels and Bremdal, 2000) and has the capability to extract ontologies from a piece of natural language text and display them in XML or RDF (depending on settings). MÍMÍR is CognIT's linguistic kernel for intelligent content analysis. See also Section 8.3.1.4 for the third version of MÍMÍR.

The ontology generator program (which uses the OntoExtract technology) has a very straightforward interface (see Figure 8.1). It has an input pane, an output pane, and a CCA view pane (which is not in use in version 0.3). Instead of viewing the graph view in the CCA pane, the graphs are displayed in the original CCA viewer (see Section 8.3.1.3). A file can be loaded, or text that will serve as input can be typed on the input pane. The user can then choose an output option from three different XML outputs and the RDF OIL output. As these output types were designed for use in the OTK project, they are not necessarily compatible with other programs. However, the OTK-XML v0.1 standard is compatible with the author's CCA viewer and will therefore be used to illustrate, although some RDF OIL output will also be shown to illustrate relations in RDF.

When RDF OIL output is selected, background knowledge can be included in the ontology output. Background knowledge means including meronyms or hypernyms for each noun. WordNet, an online lexical reference system of which the design is inspired by current psycholinguistic theories of human lexical memory (Miller et al., 1990), provides these meronyms and hypernyms. By running a noun through WordNet, the ontology is enriched by adding extra meaning around the noun, making it easier to find related resources when these use a slightly different description of the same noun.

Excerpts of the RDFS output (without background knowledge) will be shown as well as the XML output, both accompanied by an explanation.

The RDFS output describes the ontology as extracted by OntoExtract. Before the ontology is described, the RDFS output contains Dublin Core-Based Ontology Metadata, followed by the schema and property definitions. (The Dublin Core-Based Ontology Metadata contains fields for title, creator, subject, description (a summary),

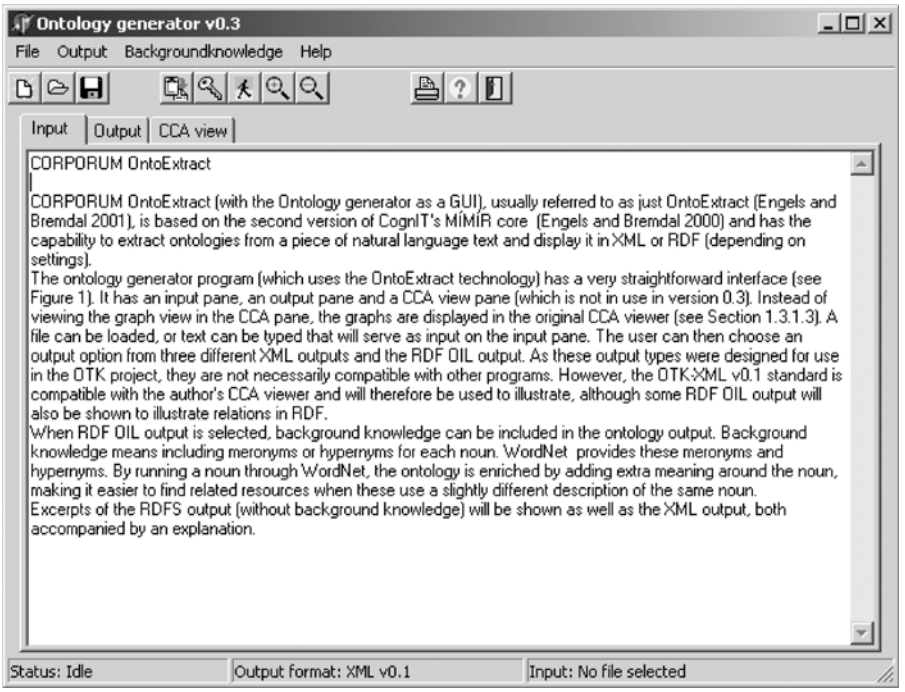


Figure 8.1 Ontology generator, an OntoExtract interface.

publisher, date, type, format, and language. See http://dublincore.org/ for an elaborate explanation.)

The ontology information first describes all single concepts that are present in the ontology and defines them as a subclass of either #TOP or #MISC, as shown in the excerpt below.

```
<oe:Concept rdf:about="#setting">
    <rdfs:label xml:lang="en">setting</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Top"/>
</oe:Concept>
<oe:Concept rdf:about="#CORPORUM_OntoExtract">
    <rdfs:label xml:lang="en">CORPORUM
OntoExtract</rdfs:label>
    <rdfs:subClassOf rdf:resource="#MISC"/>
</oe:Concept>
```

Then specifiers for the concepts are given. The excerpt below states that "#interface_1" has the properties "straightforward" and "very".

```
<rdf:Description rdf:about="">
  <oe:isAbout>
    <ns1:interface rdf:about="#interface_1">
     <oe:hasSomeProperty
xml:lang="en">straightforward</oe:hasSomeProperty>
     <oe:hasSomeProperty
```

```
xml:lang="en">very</oe:hasSomeProperty>
    </ns1:interface>
  </oe:isAbout>
</rdf:Description>
```

In the same part we define subclasses, for example, that "#ontology_generator_program" is a subclass of "#program", as shown in the excerpt below.

```
<oe:Concept rdf:about="#ontology_generator_program">
    <rdfs:label xml:lang="en">ontology generator
program</rdfs:label>
    <rdfs:subClassOf rdf:resource="#program"/>
</oe:Concept>
```

Since the CCA viewer requires XML as input, the structure of this XML output is shown next.

First a list is created of all important concepts found in the input. This is shown in the excerpt below (note that these are just *some* and not *all* of the extracted concepts).

```
<CONCEPTLIST>
  <CONCEPT>CORPORUM_OntoExtract</CONCEPT>
  <CONCEPT>CORPORUM</CONCEPT>
  <CONCEPT>OntoExtract</CONCEPT>
  <CONCEPT>second_version_CognIT_MİMİR_core</CONCEPT>
  <CONCEPT>CognIT_MİMİR</CONCEPT>
  <CONCEPT>CognIT</CONCEPT>
  <CONCEPT>MİMİR</CONCEPT>
  <CONCEPT>engels_Bremdal</CONCEPT>
  <CONCEPT>ontology_generator</CONCEPT>
  <CONCEPT>ontology</CONCEPT>
  <CONCEPT>generator</CONCEPT>
  <CONCEPT>RDF</CONCEPT>
  <CONCEPT>setting</CONCEPT>
  <CONCEPT>GUI</CONCEPT>
</CONCEPTLIST>
```

Then, the relation strength between these concepts is displayed. Relation strength per concept pair is presented as a triple, shown in the excerpt below.

```
<RELATION>
  <CONCEPT>CORPORUM_OntoExtract</CONCEPT>
  <STRENGTH>0.7500</STRENGTH>
  <CONCEPT>CORPORUM</CONCEPT>
</RELATION>
<RELATION>
  <CONCEPT>CORPORUM</CONCEPT>
  <STRENGTH>0.2000</STRENGTH>
  <CONCEPT>CORPORUM_OntoExtract</CONCEPT>
</RELATION>
```

Relation strength is bidirectional, as one concept usually is not related to another concept with the same relation strength. In the excerpt above, it becomes clear that *CORPORUM_OntoExtract* is more strongly related to *CORPORUM* than the other way

around. This is logical since CORPORUM is the name of a group of products, while CORPORUM OntoExtract is the name of a single product. The relation strength is used by the CCA viewer to create a graphical representation of the ontology. However, the CCA viewer is required to use the average relation strength to create the output. Information on how the CCA viewer works is shown in the next section.

8.3.1.3 CCA Viewer

The CCA viewer interprets the XML output from OntoExtract and creates edges between each concept that is related to another concept (see Figure 8.2), without taking into consideration the relation strength. Then, this network of concepts is run through the spring embedder algorithm, and displays the ontology graphically (see Figure 8.3). In general, closely related concepts are placed near each other; less related concepts are placed further away.

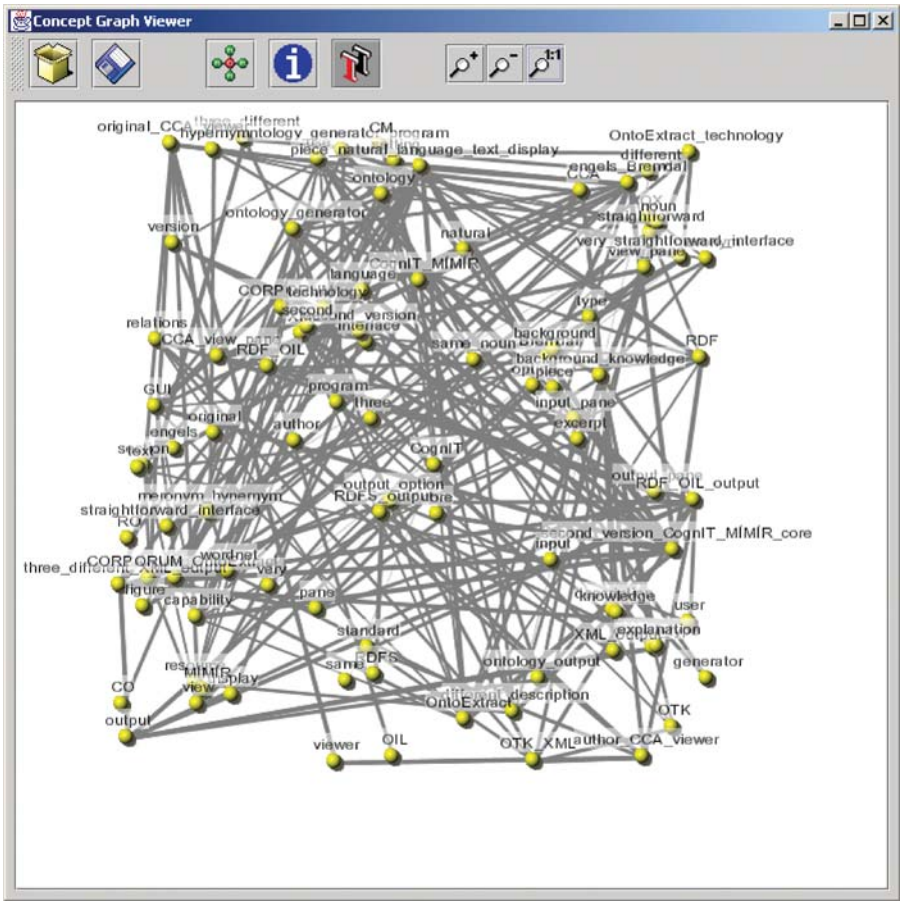


Figure 8.2 Screenshot of CCA viewer before running the Spring Embedder algorithm.

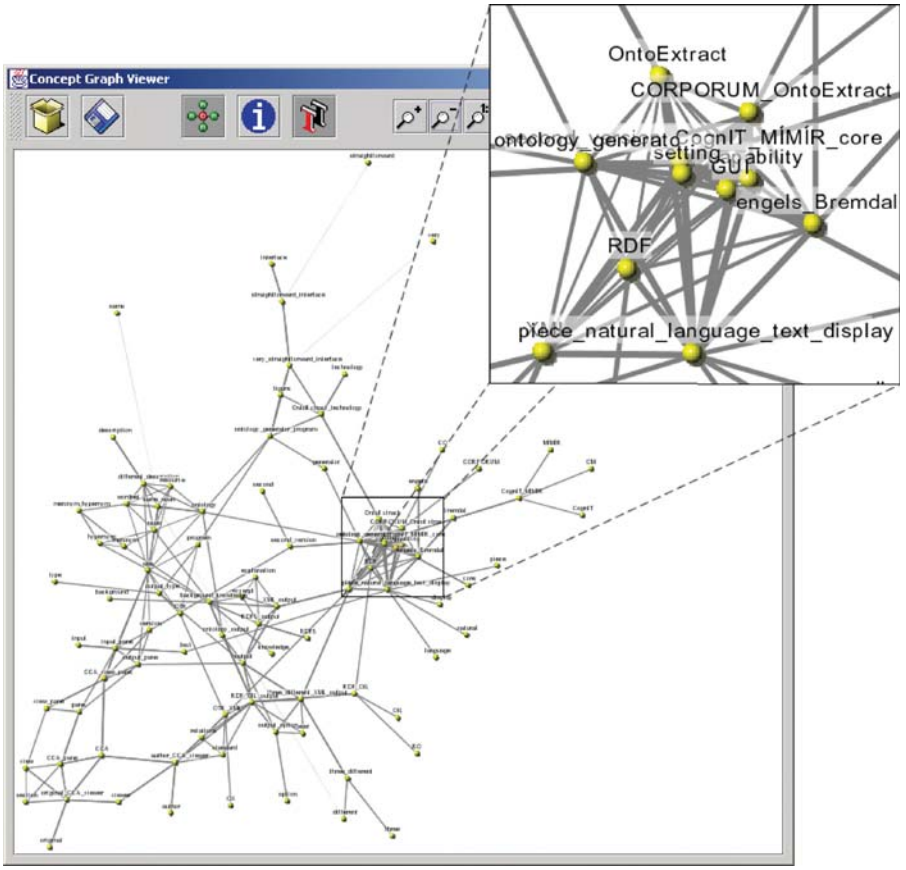


Figure 8.3 Screenshot of CCA viewer after running the Spring Embedder algorithm.

The spring embedder algorithm is based on the functioning of a spring, where nodes are connected to each other by means of strings. The spring strength will attract or repel the nodes until minimum energy strength is reached (Eades, 1984).

When one imagines the link strength between concepts as the strength of the springs, it is possible to graphically display the relation between words. However, words are not always equally related to each other; therefore some information may be lost when average link strength is used.

The parallel with the springs will make it easier to understand how to create your own graphical ontology; the more related the concepts are, the stronger the spring and the closer these concepts are to each other. And when concepts are less related to each other, the spring is weaker and therefore the distance between the concepts will increase.

8.3.1.4 CORPORUM® Intelligent Components Kernel

CognIT's CORPORUM® Intelligent Component Kernel (IC kernel) is the third version of MÍMÍR and provides more accurate linguistic analysis than the previous

versions. Unlike pure statistical solutions that focus on pattern matching of symbols only, the IC kernel combines linguistic analysis and ontological computation to (among other things) generate interest-based summaries and document associations based on relationships between concepts.

The IC kernel component is used by other components in the CORPORUM® Intelligent Components Toolbox to provide the basis for knowledge management activities, though it is also possible to build other applications using the IC kernel's API. It could replace the kernel in OntoExtract in order to enable an even more accurate text analysis.

8.3.1.5 "Connector"

The graphical presentation module, which at the time of writing goes by the codename "Connector," uses the ontology output from the IC kernel and provides the basis for the ideas of GODE as it allows for dragging and editing the visualized nodes. The screenshots in Figure 8.4 are taken from a beta version, which represents the ontologies in 3D (a 2D version is under development). It is possible to search for nodes with textual search in order to select it for navigating with it or changing the label. At the time of writing, functionality for deleting and manually adding nodes is under development, as well as the possibility to export the visualized ontology to a format which can be used by the IC kernel to search for similar documents. Functionality for typing the edges, as well as adding the canvas as described in the next section, is scheduled for development at a later stage.

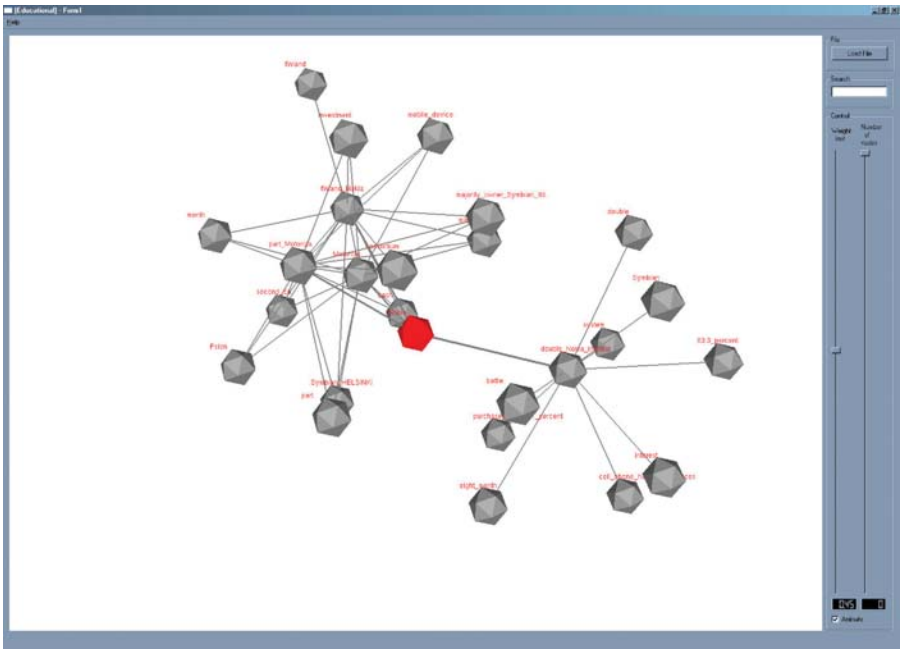


Figure 8.4 Screenshots from the beta version of the "Connector" program, developed by CognIT.

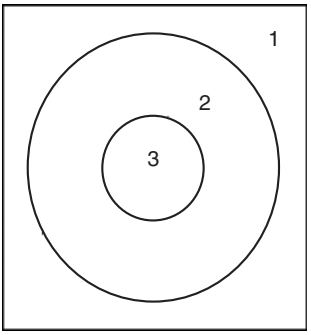


Figure 8.5 Example canvas. Legend: 1–Canvas; 2–Generalization area; 3–Main concepts.

8.3.2 GODE GUI and Functionality

The interface for the Graphical Ontology Designer Environment contains:

- A canvas for drawing the ontologies, using a graph editor that has the possibility to add, edit, move, or remove the nodes and edges on the graphical ontology. This canvas includes two main circles, one for the main focus, and one for background knowledge. Both circles can be adjusted in size. The canvas part not covered by the circles can serve as a "temporary canvas" where concepts can be put "on hold" (meaning they are not part of the search). See Figure 8.5 for an impression. Note that the canvas area can be used to temporarily store concepts that will not be part of the search.
- Result page
- Load and save function
- Tools for
  - Making concepts
  - Editing concepts
  - Removing concepts
  - Creating links between concepts
  - Editing links between concepts (relation strength, type of relation, etc)
  - Removing links between concepts
- Example ontologies
- A semantic knowledge base, or connection to a semantic knowledge base
- Concepts suggestion pane, to show related concepts
- A help function
- Algorithms to translate the graphical queries to:
  - Advanced search queries for the Web
  - Semantic Web queries (RQL, TMQL, etc.)

8.3.3 Simple Search

The simple search version of the graphical ontology search does not take into consideration the type of relations between the search concepts (with the exception of

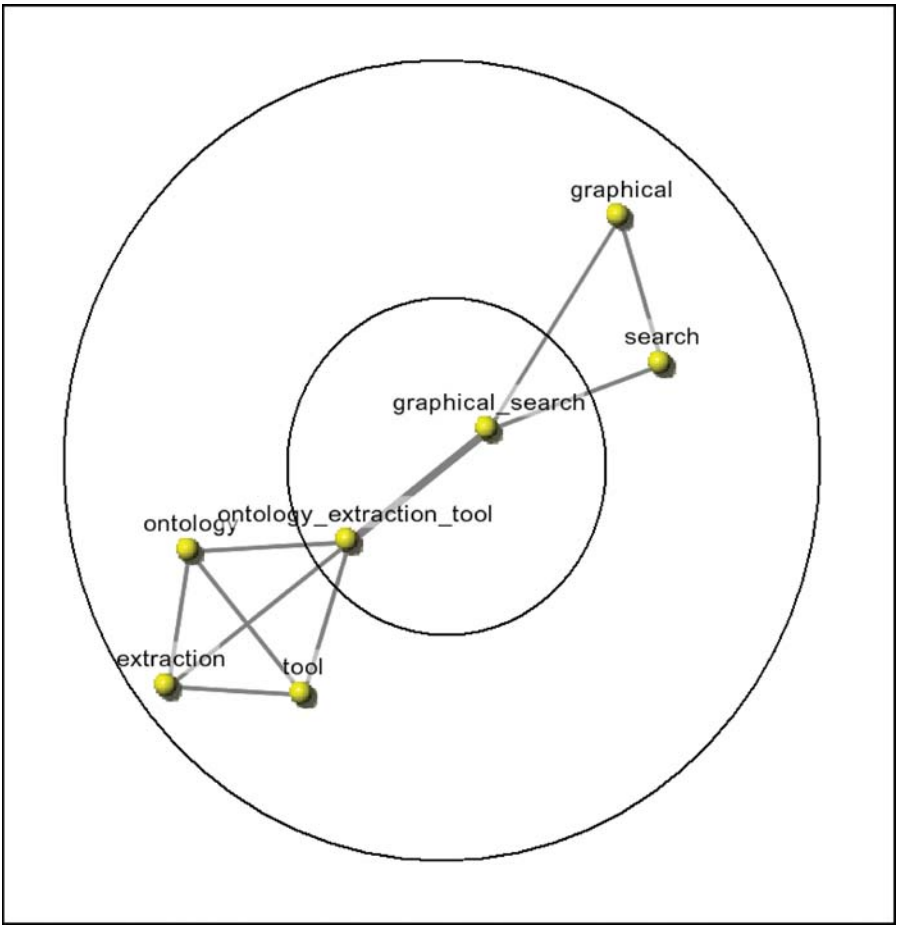


Figure 8.6 Graphical ontology with "unfolded" collocations (with GODE canvas).

numbers), hence the "simple search." Since the simple search does not take these relation types into consideration, this type of search technique can also be carried out on the Web. Section 8.3.4 explains the advanced search scenario, where these relation types are taken into consideration.

Basically, there are two possible scenarios. The most user-friendly one is the guided search, which will be shown first. It is meant to get users accustomed to the concept of graphical searching. Guided search in this case means that the user will be able to type a query in plain natural language, and this will be automatically transformed to a graphical ontology, which in its turn can be edited to better suit the searcher's needs. The second option is building a graphical ontology from scratch, which of course requires some familiarity with the concept before a user can take advantage of this more powerful method. Simple search with guided search is very suitable for naive users. Without the guided search, it is suitable for use by medium-level users.

Both these search methods are improvements on the common type of search on the Web, that is, context. Since words are related to each other and thereby placed in context, the usefulness of the retrieved results increases.

8.3.3.1 Guided Search

As shown in Section 8.3.1, tools for making graphical ontologies already exist. However, they are not perfect; just like the beginning graphical searcher needs a hand to get started, these programs need a human hand in order to perform better (Davies et al., 2003).

The main idea with guided search is that a user simply types a sentence or perhaps a whole section or document to base the graphical query on. GODE then runs this piece of text through OntoExtract or the Intelligent Component Kernel application program (for both, see Section 8.3.1) and presents the graphical output (it will automatically process the XML output from OntoExtract and run it through the spring embedder algorithm).

Now, the user can take this graphical ontology as a starting point and add, remove, edit, and/or move the nodes.

8.3.3.2 Building Your Own Graphical Ontology

Building your own graphical ontology from scratch is not as simple as typing a couple of keywords and getting presented with a list with results.

First, begin by making nodes for the words/concepts that you normally would use in any "normal" search engine. Where applicable, connect these concepts with one other by adding an edge. Then decide where to place the concepts in relation to each other. Bear in mind the spring embedder approach to guide you through the placement. Strong links are strong edges and therefore placed close to each other; weaker related edges are placed further away from each other. When all concepts have equal distance between them (only possible in very small ontologies), then each concept is equally important. A semantic knowledge base, as mentioned in Section 8.3.3.1, can also help to pick concepts faster and more accurately. In case collocations are used, these can either be presented as the two concepts with an underscore delimiter (e.g., "ontology_extraction_tool"), or as separate, though related nodes. The third option is to do both, as shown in Figure 8.6. By placing the single concepts in the outer ring, the relevance of the concepts by themselves is lower than the collocation presented in the center ring as this is the core search area.

8.3.3.3 Ontology Checking

When building a graphical ontology without any guidance, one may be in doubt of the quality of the ontology. Therefore, it would be very nice to get some feedback about which part of the ontology should be adjusted in case the result page is empty or contains irrelevant results. The ontology checking can be either based on a semantic knowledge base, or based on the results after a search. These ideas will reflect only

Semantic Web searches, and not Web searches (see Section 8.3.3.4 for information on the application areas of the graphical search).

Some general advice regarding the graphical ontology can be given before running a query or evaluation based upon a knowledge base. This advice can take into consideration (this is a nonexhaustive list, partly inspired by information found in (Noy et al., 2001)):

Too few concepts. This makes the search too general.
Too many concepts placed too closely to each other. This makes it hard to focus on which concepts should have the highest weight in the search.
More than one numerical range. When wrongly related to concepts this can cause trouble.
Repeated concepts. When a concept is mentioned once, it can always be related to other concepts in some way.
Incorrectly spelled concepts. A simple spelling checker can supply alternatives.
Wrong collocation concepts. A combination of words that can never form a collocation will never be a concept.

When the ontology is checked after running a search on the Semantic Web, information can easily be gathered based upon the results. The result set may, for example, reveal that a concept is never related to a concept that the user chose to relate it to; in this case, the relation may not exist.

Checking based upon a semantic knowledge base includes a potential extra source of error in case the knowledge base is rather small. Relations assumed by the user that are not present in the knowledge base might exist in large numbers on the Semantic Web, yet these relations will be marked as a potential problem, meaning the ontology is wrongly evaluated.

8.3.3.4 Application Area of Simple Graphical Ontologies

As mentioned before, simple search with or without guided search is very suitable for naive and medium-level users. Simple graphical search, with or without guidance, can be used for searching both the Web and the Semantic Web. Simple graphical search will not, unlike advanced graphical search, unleash the full power of the Semantic Web, since valuable relation information is missing.

When using simple graphical search for the Web, probably the best result will be achieved using a two-step method. The first step is to translate relations to advanced queries for Web search engines and execute these queries for a user-defined selection of search engines. The translation consists of creating a query out of the concepts that are contained in the main concept ring (see Figure 8.5). Concepts in the second ring, as well as on the remainder of the canvas are not used at all in this iteration. All concepts in the main concept ring that are connected with an edge will be bound by the AND parameter. Negated edges will get the negation tag valid for the chosen search engine.

The second step is to use a user-defined number of results that are presented by each of the chosen search engines as a basis for further processing by CORPORUM® technology (Bremdal and Johansen, 2000). This is needed since the results from the search engines only give a notion of the fact that the concepts are present on the retrieved page without knowing whether the concepts are related to each other. By running these results through, for example, the IC kernel, the semantic relations of the

results are extracted and compared to the concepts in the original graphical ontology. The results are then rearranged according to the semantic relevancy.

When translating to Semantic Web queries, concept relation strength can be calculated based on the length of the edges and the placement of concepts in relation to each other as the length more or less indicates the linguistic distance. Relation strength will be equal from concept A to B and B to A. This is valid for all concepts except the ones placed on the square part of the canvas (the "on hold" part).

Simple graphical ontologies can also be used as a reader's aid. Instead of reading a summary of a paper, one can simply take a look at the ontology and judge by the core concepts and their relations if an article is worth reading. Of course, these ontologies should be larger than the example provided in this section; a reasonable minimum amount is approximately 15 concepts. The simple graphical ontology can be produced either manually or by the steps described in the beginning of this section.

8.3.3.5 Intended Audience for Simple Graphical Search

Simple graphical search is intended for users who want to become familiar with the strength of the Semantic Web, as well as users who want to get more relevant results on the Web. Little a-priori knowledge is needed to start using simple graphical search, although it is necessary to have some sense of logic and language. In general, users should experience some parallels with the keyword search they are used to, especially when using the guided search.

8.3.4 Advanced Search

It has already been pointed out that the simple search alternative is meant for the general, nonspecialized audience, or naive and medium-level users, as it lacks the possibility to make full use of the semantic information provided on the Semantic Web. Therefore, an advanced search alternative that will unleash the full potential of the Semantic Web is presented in this section. The advanced search includes bidirectional relation strengths and typed edges (e.g., negation and range) to indicate the type of relation between two concepts and much more.

The environment for advanced search will be much like the simple search, yet many more options will be available. Edges can have the following properties or types: bidirectional relation strength, negation, range, synonym, meronym, holonym, hypernym (superclass), or hyponym (subclass). All of these are powerful tools to retrieve valuable information from the Semantic Web more accurately.

The negation and range properties are especially valuable for including in the main concept area. A negation relation between two concepts is much stronger than two concepts that are not connected to each other (since even though they are not connected, they could be related). The range property gives the user the ability to search a range of numerical values or dates. A newspaper article that was published online sometime last week will be a lot easier to retrieve taking into account the date range as well as the semantic relation of the content of the article.

The other relation types (synonym, meronym, holonym, hypernym, hyponym) will typically be used between the main concept area and the generalization area as they serve as a type of background information. Bidirectional relation strength can be used between all concepts, when needed in addition to the other properties. For a

more detailed explanation of these relation types and how they can be used, refer to Wienhofen (2004).

Also, with the advanced search, a type of guidance can be built in. The guided search, as mentioned in Section 8.3.3.1, can also be used to get a kick-start with advanced search. However, only a relation between the concepts is established; the user must manually define the type of relation between these concepts.

It is not always easy to express the correct meaning of a concept; WordNet can assist users by supplying all known meanings of the concepts. The user then simply picks the correct meaning and adds this as a relation to the concept: disambiguation in practice! WordNet can also assist in finding the correct relation type between concepts, or carry out a search on, for example, all meronyms of the concept "car." Note, however, that WordNet is being continuously worked on, and it is aiming at providing lexical references for all known words; yet it will take time before this ultimate goal is achieved.

8.3.5 Application Area of Advanced Graphical Ontologies

It would be pointless to invest time and effort into creating ontologies with many advanced relations and to simply strip all this information in order to be able to apply it to a Web search. That is why advanced graphical search is solely intended for use on the Semantic Web or in corporate situations where resources are semantically annotated.

The first and foremost application method for advanced graphical ontologies is searching on the Semantic Web. With all advanced properties on edges it should be relatively easy to find relevant information on a topic. Example methods for translating graphical ontologies to Semantic Web queries are described in Wienhofen (2003).

As described in Wienhofen (2004), one can use large(r) advanced graphical ontologies for knowledge management. A large corporation can, for example, define an ontology for each business unit, work process, and the skills of each employee. This way, skills management can be put in practice. The human resource department will be able to find employees internally to fill a vacancy, instead of sending out a job description. The whole can also identify skill gaps in the organization, enabling effective hiring of people. These ontologies can also be very useful for finding relevant documentation on the Semantic Web or on the (semantically annotated) corporate intranet. The problem with information is that even though it is often available it might not be visible. By making information visible, time can be saved and employees can work more effectively.

Furthermore, when an ontology is defined for each business unit, there will no longer be ambiguity between concepts that are the same in writing, yet different in meaning. Take, for example, a medium-sized account and advisor firm, large enough to have its own IT department. For accountants, advisors, and/or lawyers (the vocabulary of these three groups itself is diverse enough to define an ontology for every single one, but this is not the point in this case), a whole different vocabulary is used than for the IT department. Take, for example, "to implement something"; for an employee in the IT department it means something different than for a lawyer or accountant. When the background of the employee is known, it can be used as background knowledge for searching, and the concept "implement" is automatically enriched by the knowledge that this employee is interested in the IT point of view on this word. In other words,

by using advanced graphical ontologies to define the context of a person once (much like the interest model in CORPORUM® Summarizer), one can already enable better result sets, as ambiguous words are already put in context.

8.3.6 Intended Audience for Advanced Graphical Search

Advanced graphical search can be used by people who have become familiar with the concept of graphical searching, and who want even better results. Knowledge engineers and medium or large corporations with a lot of procedures and business units are other candidates. As mentioned, the human resource department in a company could benefit greatly from having the possibility to query a semantic database that contains an ontology describing each employee. Knowledge engineers can create this semantic database by creating graphical ontologies and store the semantic relations of all concepts in a semantic database, for example, the Sesame database (Broekstra et al., 2001).

8.3.7 Possible Traps

Simple graphical search still has the problem that it does not support relations like is-a, has-a, etc., so the full power of the Semantic Web cannot be used.

Implicit relation strengths (meaning strength based on placement of concepts, rather than an explicitly mentioned relation between two concepts) are a challenge to calculate. Take, for example, four concepts, each of which is related to each other one with equal strength, implying that the graphical representation looks like a square with an X in it (each node is one corner). In this case, the diagonal edges are longer than the other edges, yet all edges from this example ideally have equal strength. The spring embedder algorithm would place the concepts in the same way, as this is the minimum energy situation. A conversion algorithm could take the wrong decision and give the diagonally connected concepts a lower strength, which in turn could lead to unwanted results. Since this example is quite small, the effect wouldn't have much impact, yet larger ontologies may suffer from such miscalculations.

The advanced option can be a real challenge to master, as many different ways of representing an ontology are possible. It may take quite a bit of training in order to be able to create advanced graphical ontologies, hence this is for expert-level users.

8.4 Conclusion and Further Work

As presented in this chapter, the main building blocks of the GODE are already available, though, at the time of writing, not integrated. By implementing the GODE as a plug-in for existing technology such as CORPORUM® Knowledge Server or CORPORUM® Intelligent Components, users can already get accustomed to the new way of searching to get ready for the next step: the Semantic Web. When the Semantic Web becomes more mature, plug-ins for conversion of visual queries to the various Semantic Web query languages will be made in order to make the Semantic Web as available for the general audience as the Web is today.

The Graphical Ontology Designer Environment is meant to give today's search engine users the possibility to draw on the full use of the Semantic Web, without

having to learn complex query languages. By gently introducing the presented search concept by means of a guided search, where one can type a natural language text of which a graphical ontology is created, the naive users get the opportunity to explore the Semantic Web with a low threshold. In a later phase (medium-level) users can construct their own graphical ontologies from scratch and run it through an ontology-checking algorithm to make sure the ontology is of good quality and does not contain any contradictions or "impossible" situations. Once the users have reached the expert level, they can benefit from the advanced options of the GODE and define relation types between the search concepts to retrieve even more accurate results.

8.5 Acknowledgments

The author would like to thank all his colleagues at CognIT who actively contributed to shaping the ideas on GODE, and who supported the technology development that makes GODE possible (especially CORPORUM® Intelligent Components Kernel and "Connector").

8.6 References

Balkir, N.H., Ozsoyoglu, G., Ozsoyoglu, Z.M. (2002). A graphical query language: VISUAL and its query processing. *IEEE Transactions on Knowledge and Data Engineering,* 14(5):955–978.

Berners-Lee, T. (1998). *Semantic Web Roadmap*. Available: http://www.w3.org/DesignIssues/Semantic.html.

Bremdal, B., Johansen, F. (2000). CORPORUM technology and applications. White paper, CognIT a.s, Norway.

Brickley, D., Guha, R.V. (2003). *RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft,* January 23, 2003. Available http://www.w3.org/TR/rdf-schema/.

Broekstra, J., Kampman, A., Harmelen, F. (2001). Sesame: An architecture for storing and querying RDF data and schema information. *Semantics for the WWW.* MIT Press.

Catarci, T., Santucci, G. (1995). Are visual query languages easier to use than traditional ones? An experimental proof. *International Conference on Human-Computer Interaction (HCI95).*

Davies, J., Fensel, D., Harmelen, F. (2003). *Towards the Semantic Web: Ontology-Driven Knowledge Management.* John Wiley & Sons. ISBN 0470 84867 7.

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium* 42:149–160.

Engels, R., Bremdal, B. (2000). Information extraction: State-of-the-art report. *Deliverable 5 of the EU 5th Framework Project OnToKnowledge (IST-1999-10132)*, CognIT a.s, Norway.

Engels, R., Bremdal, B. (2001). Ontology extraction tool. *Deliverable 6 of the EU 5th Framework Project OnToKnowledge (IST-1999-10132),* CognIT a.s, Norway.

Fensel, D., Harmelen, F., Klein, M., Akkermans, H., Broekstra, J., Fluit, C., Meer, J., Schnurr, H.P., Studer, R., Hughes, J., Krohn, U., Davies, J., Engels, R., Bremdal, B., Ygge, F., Lau, T., Novotny, B., Reimer, U., Horrocks, I. (2000). On-To-Knowledge: Ontology-based tools for knowledge management. *Proceedings of the eBusiness and eWork 2000 (EMMSEC2000) Conferenc*e, Madrid, Spain.

Fensel, D., Harmelen, F., Ding, Y., Klein, M., Akkermans, H., Broekstra, J., Kampman, A., Meer, J., Sure, Y., Studer, R., Krohn, U., Davies, J., Engels, R., Iosif, V., Kiryakov, A., Lau, T., Reimer, U., Horrocks, I. (2002). On-To-Knowledge in a Nutshell. *IEEE Computer.*

Gruber, T.R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220.

Kumar, A., Fowler, R.H. (1996). A spring modeling algorithm to position nodes of an undirected graph in three dimensions. *Technical Report Department of Computer Science University of Texas.* Available: http://www.cs.panam.edu/info_vis/spr_tr.html.

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography (special issue),* 3(4):235–312.

Ni, W., Ling, T.W. (2003). *GLASS: A Graphical Query Language for Semi-Structured Data.* Available: http://www.comp.nus.edu.sg/~lingtw/papers/dasfaa2003.pdf.

Noy, N.F., McGuinness, D.L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.*

Wienhofen, L.W.M. (2003). Using graphical ontologies for searching the (Semantic) Web. MSc Thesis, University College of Østfold, Norway.

Wienhofen, L.W.M. (2004). Using graphically represented ontologies for searching content on the Semantic Web. *Proceedings of 8th International Conference on Information Visualisation, IV 2004, IEEE Computer Society 2004,* ISBN 0-7695-2177-0 IV 2004, pp. 801–806.

Chapter 9

Adapting Graph Visualization Techniques for the Visualization of RDF Data

Flavius Frasincar, Alexandru Telea, and Geert-Jan Houben

9.1 Introduction

The foundation language for the Semantic Web is the Resource Description Framework (RDF). RDF is intended to describe the Web metadata so that the Web content is not only machine readable but also machine understandable. In this way one can better support the interoperability of Web applications. RDF Schema (RDFS) is used to describe different RDF vocabularies (schemas), that is, the classes and properties associated to a particular application domain. An instantiation of these classes and properties form an RDF instance. It is important to note that both an RDF schema and an RDF instance have RDF graph representations.

Realizing the advantages that RDF offers, in the last couple of years, many tools were built in order to support the browsing and editing of RDF data. Among these tools we mention Protégé (Noy et al., 2001), OntoEdit (Sure et al., 2003), and RDF Instance Creator (RIC) (Grove, 2002). Most of the text-based environments are unable to cope with large amounts of data in the sense of presenting them in a way that is easy to understand and navigate (Card et al., 1999). The RDF data we have to deal with describes a large number of Web resources, and can thus easily reach tens of thousands of instances and attributes. We advocate the use of visual tools for browsing RDF data, as visual presentation and navigation enables users to effectively understand the complex structure and interrelationships of such data. Existing visualization tools for RDF data are: IsaViz (Pietriga, 2002), OntoRAMA (Eklund et al., 2002), and the Protégé visualization plug-ins like OntoViz (Sintek, 2004) and Jambalaya (Storey et al., 2001).

The most popular textual RDF browser/editor is Protégé (Noy et al., 2001). The generic modeling primitives of Protégé enable the export of the built model in different data formats, among which is also RDF/XML. Protégé distinguishes between schema and instance information, allowing for an incremental view of the instances based on the selected schema elements. One of the disadvantages of Protégé is that it displays the information in a hierarchical way, that is, using a tree layout (Sugiyama et al., 1981), which makes it difficult to grasp the inherent graph structure of RDF data.

In this chapter, we advocate the use of a highly customizable, interactive visualization system for the understanding of different RDF data structures. We implemented an RDF data format plug-in for GViz (Telea et al., 2002), a general-purpose visual environment for browsing and editing graph data. The largest advantage that GViz provides in comparison with other RDF visualization tools is the fact that it is easily

and fully customizable. GViz was architected with the specific goal in mind of allowing users to define new operations for data processing, visualization, and interaction to support application-specific scenarios. GViz also integrates a number of standard operations for manipulation and visualization of relational data, such as data viewers, graph layout tools, and data format support. This combination of features has enabled us to produce, in a short time, customized visualization scenarios for answering several questions about RDF data. We demonstrate our approach to RDF data visualization by using a real dataset example of considerable size.

In the next section, we describe the real-world dataset we use, and show the results obtained when visualizing it with several existing RDF tools. Our visualization tool, GViz, is presented in Section 9.3. Section 9.4 presents several visualization scenarios we built with GViz for the RDF dataset used, and details of various lessons learned when building and using such visualizations. Finally, Section 9.5 concludes the chapter, proposing future directions for visualizing RDF information.

9.2 Background

Throughout this chapter, we will use an example based on real data made available by the Rijksmuseum in Amsterdam, the largest art and history museum in the Netherlands. In the example there is a museum schema used to classify different artists and their artefacts. The museum instance describes more than 1,000 artists and artefacts. For comparison purposes, we chose to represent the same museum RDFS schema in several browsing tools.

Figure 9.1 depicts the museum schema in Protégé. As can be noticed from this figure such a text-based representation cannot nicely depict the structure of a large amount

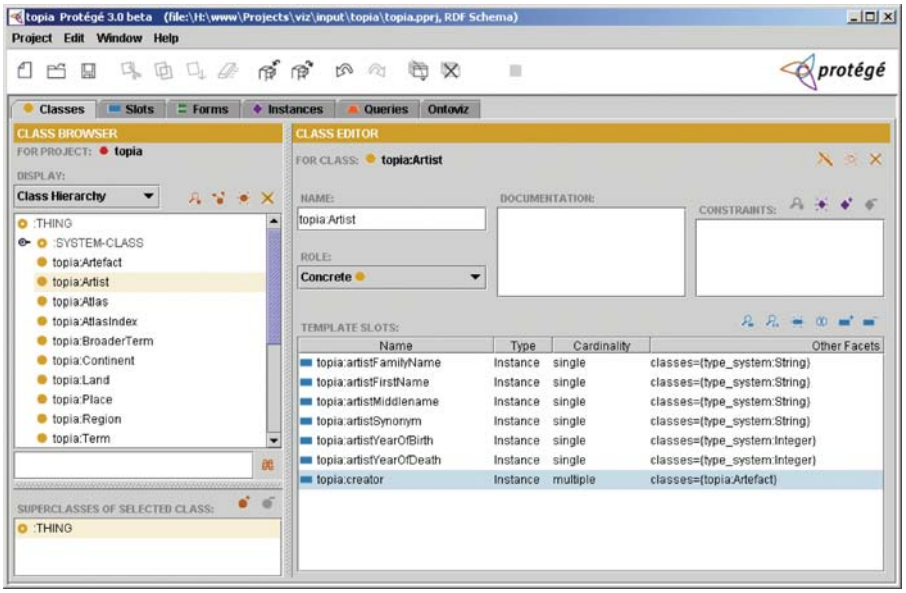


Figure 9.1 Museum schema in Protégé (text-based).

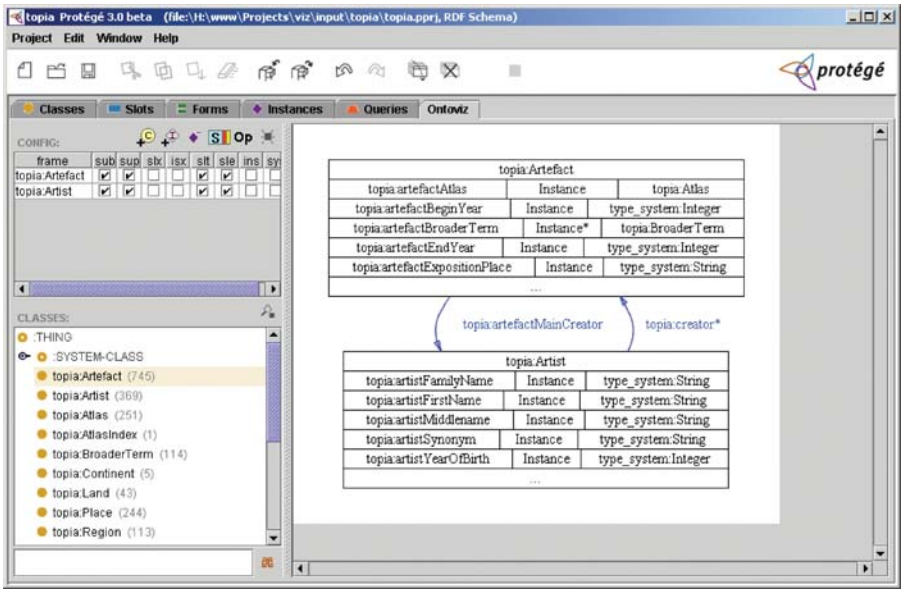


Figure 9.2 Museum schema in Protégé (with OntoViz plug-in).

of data. More exactly, a text-based display is very effective for *data mining*, that is, posing targeted queries on a dataset once one knows what structure one is looking for. However, text-based displays are not effective for *data understanding*, that is, making sense of a given (large) dataset of which the global structure is unknown to the user.

In order to alleviate the above limitation, Protégé offers a number of built-in visualization plug-ins. Figure 9.2 shows the graph representation generated by the OntoViz plug-in for two classes from the museum schema. The weak point of OntoViz is the fact that it is not able to produce good (understandable) layouts for graphs that have more than 10 nodes.

IsaViz (Pietriga, 2002) is a visual tool for browsing/editing RDF models. IsaViz uses AT&T's GraphViz package (North and Koutsofios, 1996) for the graph layout.

Figure 9.3 shows the same museum schema using IsaViz. The layout produced by the tool is much better than the one generated with OntoViz. However, the directed acyclic graph layout used (Sugiyama et al., 1981) becomes ineffective when the dataset at hand has roughly more than a hundred nodes, as can be seen from Figure 9.3. IsaViz has a 2.5D GUI with zooming capabilities and provides numerous operations like text-based search, copy-and-paste, editing of the geometry of nodes and arcs, textual attribute browsing, and graph navigation.

For all these reasons, we believe that IsaViz is a state-of-the-art tool for browsing/editing RDF models. However, its rigid architecture makes it difficult to define application-dependent operations other than the standard ones currently provided by the tool. Experience in several communities interested in visualizing relational data in general, such as software engineering and Web engineering, and our own experience with RDF data in particular, has shown that tool customization is extremely important. Indeed, there is no "silver bullet" or best way to visualize large graph-like datasets. The questions to be answered, the data structure and size, and the user

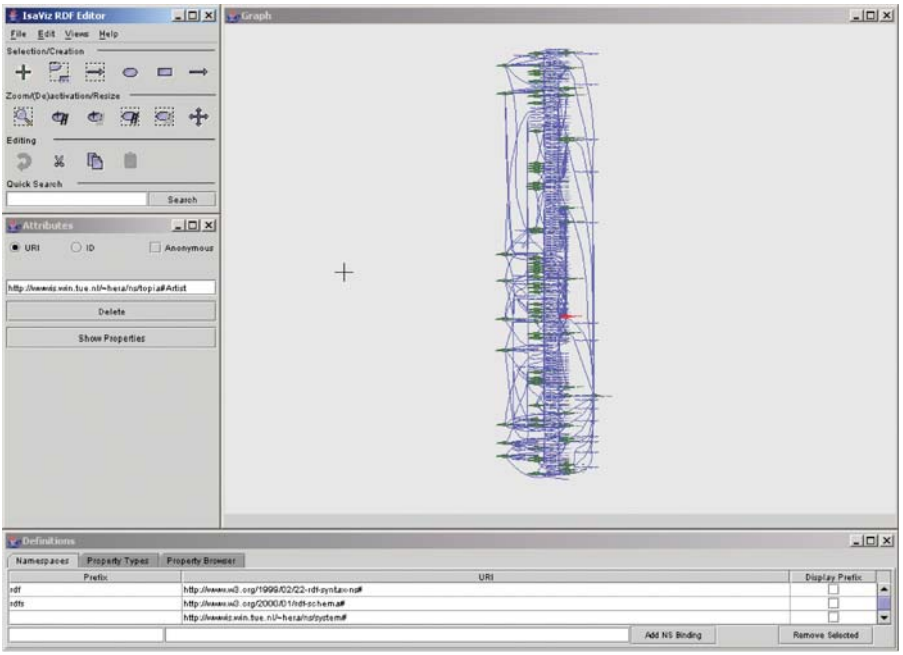


Figure 9.3 Museum schema in IsaViz.

preferences all determine the "visualization scenario," that is, the kind of (interactive) operations the users may want to perform to get insight into the data and answers to their questions. It is not that each separate application domain demands a specific visualization scenario. Users of the same domain and/or even the same dataset within the same domain may require different scenarios. Building such scenarios often is responsible for a large part of the complete time spent in understanding a given dataset (Telea, 2004). This clearly requires the visualization tool in use to be highly (and easily) customizable.

9.3 GViz

In our attempt to understand RDF data through visual representations, an existing tool was used. We implemented an RDF data format plug-in for GViz (Telea et al., 2002), a general-purpose visual environment for browsing and editing graph data. The largest advantage that GViz provides in comparison with other RDF visualization tools is the fact that it is easily and quickly customizable. One can seamlessly define new operations to support application-specific scenarios, making the tool more amenable for the user needs. In the past, GViz was successfully used in the reverse engineering domain, in order to define application-specific visualization scenarios.

Figure 9.4 presents the architecture of GViz based on four components: selection, mapping, editing, and visualization. In the next section we describe the data model used in GViz. Next, we outline the operation model describing the tasks that can be defined on the graph data. We finish the description of the GViz architecture

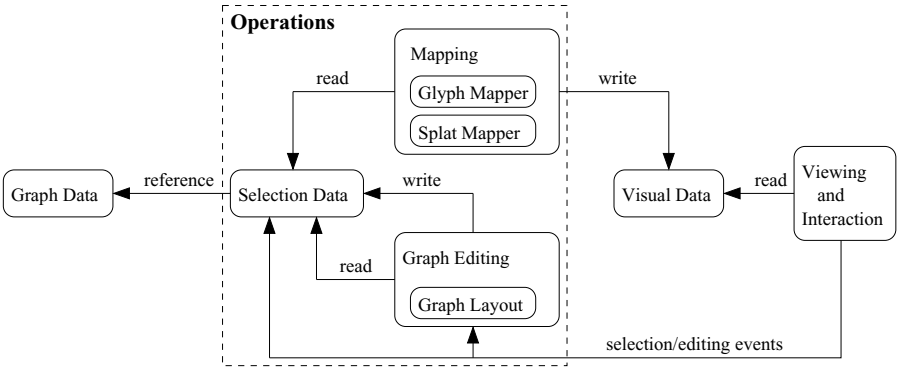


Figure 9.4 GViz architecture.

with the visualization component, which we illustrate using the museum schema dataset.

The GViz core implementation is done in C++ while the user interface and scripting layer were implemented in Tcl (Raines, 1998) to take advantage of the runtime scripting and weak typing flexibility that this language provides. All the GViz customization code developed for the RDF visualization scenarios presented in this chapter was done in Tcl.

9.3.1 Data Model

The data model of GViz consists of three elements:

1. *Graph data.* This is the RDF graph model, that is, a labeled directed multigraph in which no edges between the same two nodes are allowed to share the same label. Nodes stand for RDF resources/literals and edges denote properties. Each node has a `type` attribute that states if the node is a `NResource` (named resource), an `AResource` (anonymous resource), or a `Literal`. The label associated to a node/edge is given by the `value` attribute. The labels for `NResource` nodes and edges are URIs. The label for `Literals` is their associated string. The value of an `AResource` is an internal identifier with no RDF semantics. Note that the `type` and `value` attributes are GViz-specific attributes that should not be confused with their RDF counterparts. Since GViz's standard data model is an arbitrary attributed graph, with any number of (name, value) type of attributes per node and edge, the RDF data model is directly accommodated by the tool.
2. S*election data.* Selections are subsets of nodes and edges in the graph data. Selections are used in GViz to specify the inputs and outputs of its operations; their use is detailed in Section 9.3.2.
3. V*isual data.* This is the information that GViz ultimately displays and allows the user to interact with. Since GViz allows customizing the mapping operation (i.e., the way graph data are used to produce visual data), the latter may assume different look-and-feel appearances. Section 9.4 illustrates this in the context of our application.

9.3.2 Operation Model

As shown in Figure 9.4, the operation model of GViz has three operation types: selection, graph editing, and mapping. Selection operations allow users to specify subsets of interest from the whole input graph. In the RDF visualization scenarios that we built with GViz, we defined different complex selections based on the attributes of the input model. These selections can perform tasks like: "extract the schema from an input set of RDF(S) data (which mixes schema and instance elements)." Custom selections are almost always needed when visualizing relational data, since (1) the user doesn't usually want to look at too many data elements at the same time, and (2) different subsets of the input data may have different semantics, and thus have to be visualized in different ways. A basic example of the latter assertion is the schema extraction selection mentioned earlier.

Graph editing operations enable the modification, creation, and deletion of nodes/edges and/or their attributes. For our RDF visualization scenarios, we did not create or delete nodes or edges. However, we did create new data attributes, as follows. One of the key features of GViz is that it separates the graph layout, that is, computing 2D or 3D geometrical positions that specify *where* to draw nodes and edges, from the graph mapping, that is, specifying *how* to draw nodes and edges. The graph layout is defined as a graph editing operation that computes position attributes. Among the different layouts that GViz supports we mention the spring embedder, the directed tree, the 3D stacked layout, and the nested layout (Telea et al., 2002). Although based on the same GraphViz package as IsaViz, the layouts of GViz are relatively more effective, as the user can customize their behavior in detail via several parameters.

Mapping operations, or briefly mappers, associate nodes/edges (containing also their layout information) to visual data. The latter is implemented using the Open Inventor 3D toolkit, which delivers high-quality, efficient rendering and interaction with large 2D and 3D geometric datasets (Wernecke, 1993). GViz implements two mappers: the glyph mapper and the splat mapper. The glyph mapper associates to every node/edge in the input selection a graphical icon (the glyph) and positions the glyphs based on the corresponding node/edge layout attributes. Essentially, the glyph mapper produces the "classical" kind of graph drawings (e.g., similar to those output by IsaViz). However, in contrast to many graph visualization tools, the glyph mapper in GViz allows full customization of the way the nodes and edges are drawn. The user can specify, for example, shapes, sizes, and colors for every separate node and edge, if desired, by writing a small Tcl script of 10 to 20 lines of code on the average. We used this feature extensively to produce our RDF visualizations described in Section 9.4. The splat mapper produces a continuous two-dimensional splat field for the input selection. For every 2D point, the field value is proportional to the density of nodes per unit area at that point. Essentially, the splat mapper shows high values where the graph layout used has placed many nodes, and low values where there are few nodes. Given that a reasonably good layout will cluster highly interconnected nodes together, the splat mapper offers a quick and easy way to visually find the clusters in the input graph (see Figure 9.9, Section 9.4). For more details on this layout, see Van Liere and De Leeuw (2003).

A final way to customize the visualizations in GViz is to associate custom *interaction* to the mappers. These are provided in the form of Tcl callback scripts that are called by the tool whenever the user interactively selects some node or edge glyph with the mouse, in the respective mapper windows. These scripts can initiate any desired operation using the selected elements as arguments, for example, showing some

attributes of the selected arguments. Examples of this mechanism are discussed in Section 9.4.

As explained above, GViz allows users to easily define new operations. For the incremental view of RDF(S) data, we defined operations as: extract schema, select classes and their corresponding instances, select instances and their attributes. As for the glyph mappers, these operations have been implemented as Tcl scripts of 10 to 25 lines of code. The usage of the custom selection, layout, and mapping operations for visualizing RDF(S) data is detailed in the remainder of this chapter.

9.3.3 Visualization

Figure 9.5 presents the museum data schema in GViz. We use here a radial tree layout, also available in the GraphViz package, instead of the directed tree layout illustrated

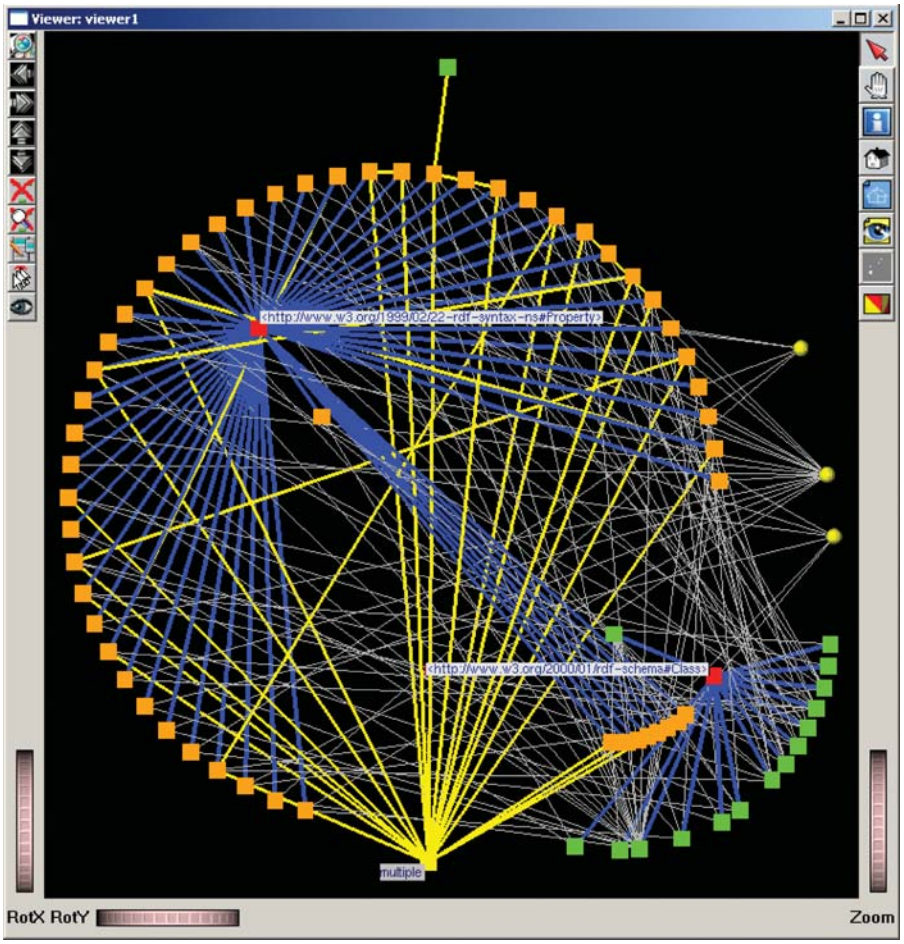


Figure 9.5 Museum schema in GViz (2D).

in Figure 9.3 for IsaViz. As a consequence, the structure of the schema is easier to understand now.

In the picture in Figure 9.5 the edges with the label `rdf:type` are depicted in blue. There are two red nodes to which these blue edges connect, one with the label `rdfs:Class` and the other with the label `rdf:Property`, shown near the nodes as balloon pop-up texts. We chose to depict the property nodes (laid out in a large circular arc around the upper-left red node) in orange and the class nodes (laid out in a smaller circle arc around the lower-right red node) in green. As can be noticed from the picture there are a lot of orange nodes, which is in accordance with the property-centric approach for defining RDFS schemas. In order to express richer domain models we extended the RDFS primitives with the cardinality of properties, the inverse of properties, and a media type system. These extensions are shown in yellow edges (see also below) and yellow spheres (positioned at the right end of the image). The yellow edges that connect to orange nodes represent the inverse of a property. The yellow edges that connect an orange node with the yellow rectangle labeled "multiple" (positioned at the middle of the figure bottom) state that this property has cardinality one-to-many. The default cardinality is one-to-one. Note that there are no many-to-many properties as we had previously decomposed these properties in two one-to-many properties. The three yellow spheres represent the media types: `String, Integer, and Image`. The light-gray thin edges denote the domain and the range of properties. Note that only range edges can have a media node at one of its ends. As these edges are (1) not so important for the user and (2) quite numerous and quite hard to lay out without many overlaps, we chose to represent them in a visually inconspicuous way (i.e., make them thin and use a background-like, light-gray color).

The tailoring of the graph visualization presented above is only one example. One can define some other visualizations depending on one's needs. Figure 9.6 presents a 3D view of the same museum schema example. Here, we used a spring embedder layout, also available from the GraphViz package, to position all schema nodes in a 2D plane. Next, we designed a custom operation that selects the two `rdfs:Class` and `rdf:Property` nodes and offsets them away from the 2D layout plane, in opposite directions. This creates a 3D layout, which allows the user to better distinguish the different kinds of edges. For example, the edges labeled `rdf:type` (colored in blue) are now clearly separated, as they reach out of the 2D plane to the offset nodes.

9.4 Applications

In order to better understand the context in which we developed our visualization applications we now briefly describe the Hera project (Vdovjak et al., 2003). Hera is a methodology for designing and developing Web Information Systems (WISs) on the Semantic Web. All the system specifications are represented in RDFS. For the scope of this chapter it is important to look at two of these specifications: the conceptual model (domain model) and the application model (navigation model).

The conceptual model describes the types of the instances that need to be presented. An example of the conceptual model we already saw in Figure 9.5. A conceptual model is composed of concepts and concept properties. There are two kinds of concept

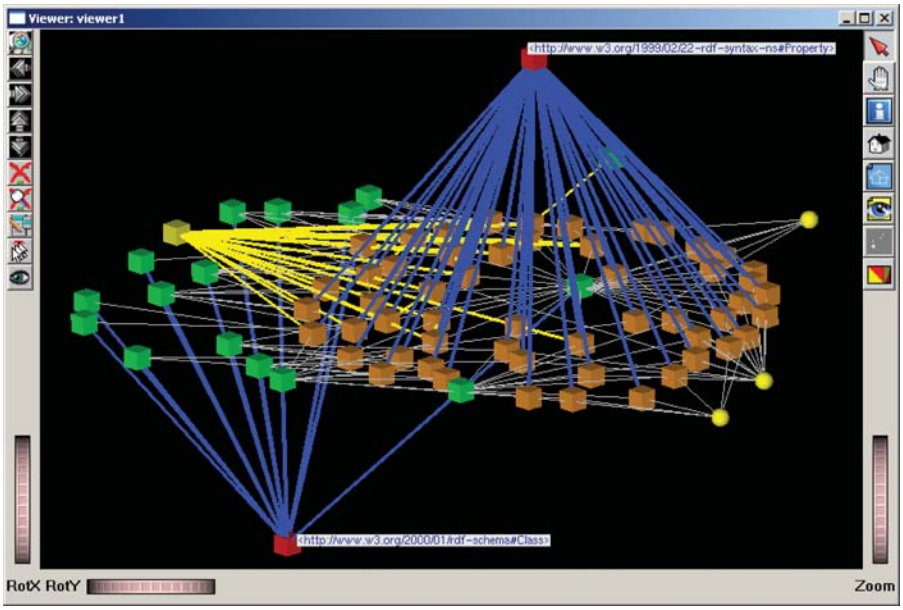


Figure 9.6 Museum schema in GViz (3D).

properties: relationships (properties between concepts) and attributes (properties that refer to media types).

The application model defines the navigation over the data, that is, a view on the conceptual model that contains appropriate navigation links. The application model is an overlay model over the conceptual model, a feature exploited in the definition of the transformations of the conceptual model instances into application model instances. An application model is composed of slices and slice properties. A slice contains concept attributes (not necessarily from the same concept) as well as other slices. There are two kinds of slice properties: compositional properties (aggregations) and navigational properties (links). The owner relationship is used to associate a slice to a concept. Each slice has a title attribute related to it.

A conceptual model instance and an application model instance are represented in RDF (which should be valid according to the corresponding RDFS specifications, i.e., the conceptual model and the application model, respectively). In the WIS application it is only the application model instance that will be visible to the user. We consider four types of RDF(S)-related visualization scenarios that are relevant in the support of the WIS application designer:

1. Conceptual model visualization
2. Conceptual model instance visualization
3. Application model visualization
4. Application model instance visualization

In Section 9.3.3 we already showed how one can visualize conceptual models. A second similar scenario for the conceptual model visualization is described next.

9.4.1 Conceptual Model Visualization

The conceptual model visualization enables one to better understand the structure of the application's domain. It answers questions such as:

What are the concepts?
What are the properties?
What are the relationships between concepts and properties?
What are the most referenced concepts?
What are the most referenced media types?

Figure 9.7 shows the extracted schema from an RDF file that contains both the schema and its associated instance. The extraction is done by a custom selection operation, as described in Section 9.3.2. The picture is very similar to the one from Figure 9.5. However, there are two differences between this picture and the one from Figure 9.5. First, we now use a different layout, that is, a spring embedder instead

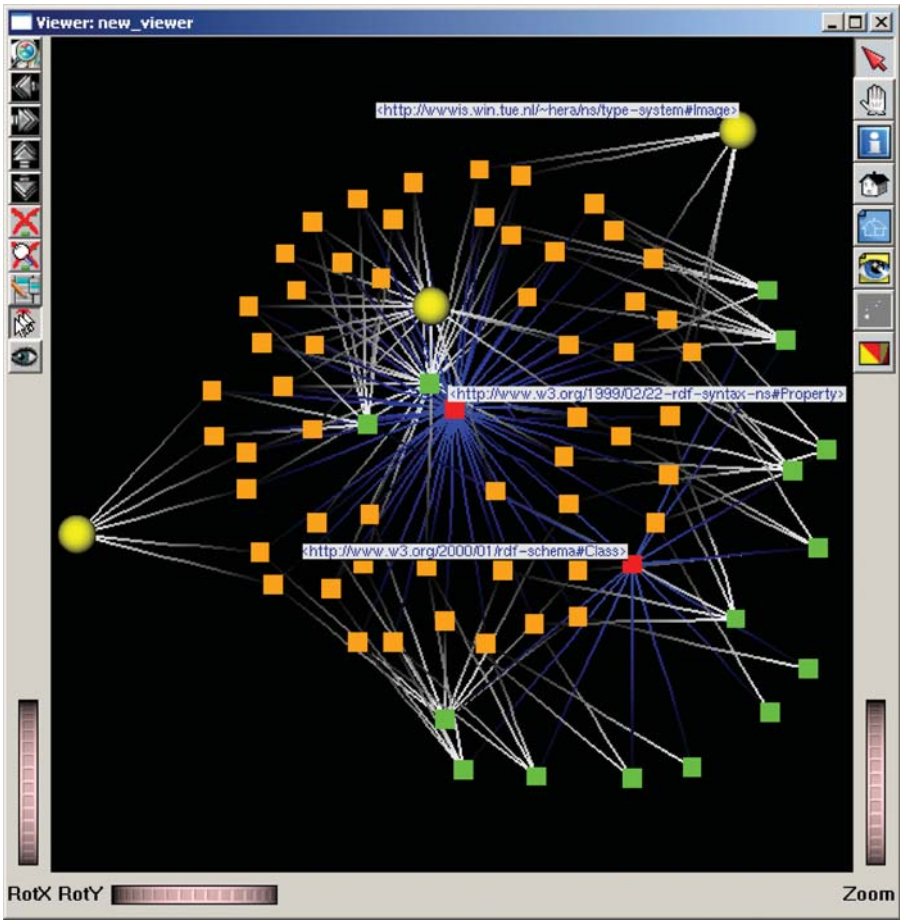


Figure 9.7 Museum extracted conceptual model.

of a radial tree. Second, we now depict also the *direction* of the edges. The edges are fading out toward the start node. A direction effect is created: the edges get brighter as they approach the end node. We found this representation of the edge direction much more effective than the arrow representation when visualizing large graphs, as the drawing of arrows produces too much visual clutter in this case. Moreover, the edge fading glyph is faster to render than an arrow glyph, as it involves a single (shaded) line primitive.

From Figure 9.7 we can deduce that the most used media type is `String` (the text-based descriptions are the most popular for this domain specification) and the most referenced concept is the `Artefact` (it has the most relationships). Each artefact is classified by some museum terms (e.g., `Self Portraits`). There is a hierarchy of museum terms; terms are grouped in broader terms (e.g., `Portraits`), and broader terms are grouped in top terms (e.g., `Paintings`).

9.4.2 Conceptual Model Instance Visualization

The conceptual model instance visualization answers questions such as:

What are the instances of a certain concept?
What are the relations between two selected concept instances?
What are the most referenced instances?
What are the attributes of a selected instance?

In most of the encountered situations, there are (much) more concept instances than concepts. For example, our museum dataset contains tens of thousands of instances. It is easy to imagine other applications where this number goes up to hundreds of thousands, or even more. Drawing *all* these instances simultaneously is neither efficient nor effective. Indeed, no graph layout we were able to test could produce an *understandable* image of an arbitrary, relatively tightly connected graph with tens of thousands of nodes in a *reasonable* amount of time (e.g., tens of seconds). In order to keep the instance visualization manageable, we decided for an incremental view scenario on the RDF(S) data. First, the user selects the subpart of the schema for which he wants the corresponding instances to be visualized. Next, we use a custom interaction script (Section 9.3.2) of about 10 lines of code to separately visualize the instances of the selected items. For example, when the user selects the `Artist` and `Artefact` concepts from Figure 9.7, the GViz tool automatically shows the instances of these concepts and their relations in another window, using a spring embedder layout (Figure 9.8). In Figure 9.8 we used a custom glyph mapper to depict the artefacts with blue rectangles and the artists with green rectangles. The relations between these instances are represented by fading white edges. One can note that there are more artefacts than artists, as expected.

Figure 9.9 shows the same selected data (artists and artefacts) but using a splat mapper instead of the classical glyph mapper. The scalar density function (splat field) is constructed as outlined in Section 9.3.2. We visualize the splat field using a blue-to-red colormap that associates cold hues (blue) to low values and warm hues (yellow, red) to medium and high values. Figure 9.9 (left) shows the splat field as seen from above. Figure 9.9 (right) shows the same splat field, this time visualized using an elevation plot that shows high-density areas also by offsetting these points in the Z (vertical) direction. A red/yellow color (Figure 9.9 left and right) or a high elevation point (Figure 9.9 right) indicates that there are a lot of relations for a particular instance

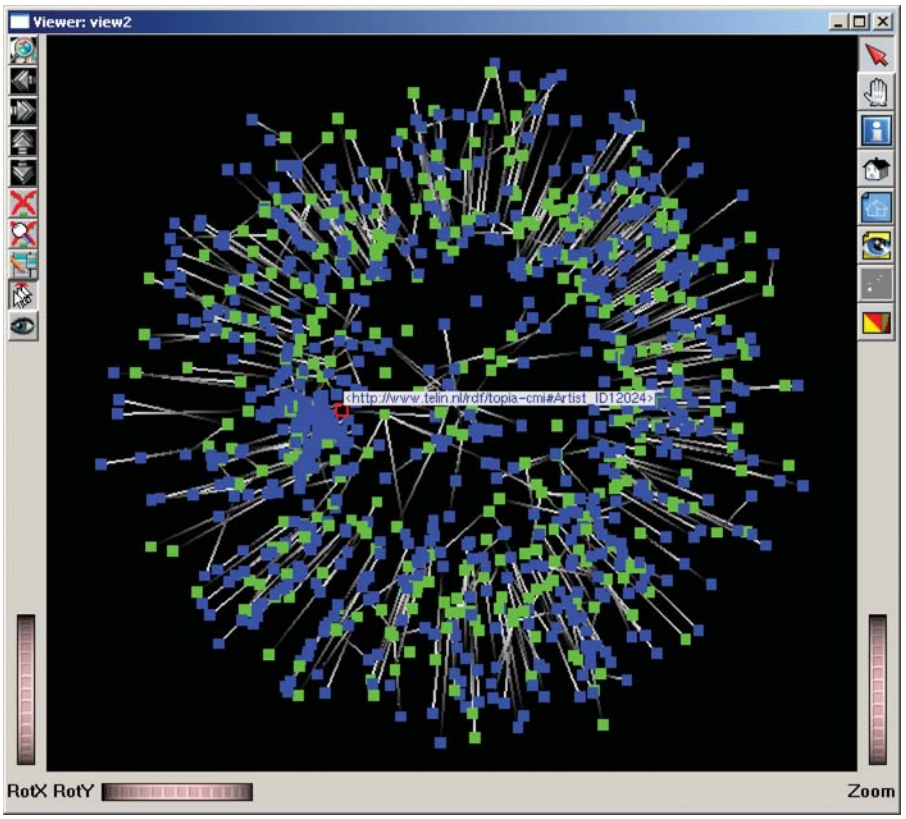


Figure 9.8 Artists/artefacts properties in the conceptual model instance.

or group of instances. In this way one can notice from Figure 9.9 which are the artists with the most artefacts. The artists with the most artefacts are the unknown artists (potter, goldsmith, bronzesmith, etc.) that show up as the singular peak in the left of Figure 9.9 (left). On the average, these artists have several tens (up to 60) of artefacts. They are followed by Rembrandt and the unknown painters, who show up as the other two higher peaks to the right of Figure 9.9 (left). This can be explained by the fact that in the seventeenth century, for which the Rijksmuseum has a special focus, there were a lot of artefacts done by unknown artists.

We have further customized our visualization scenario as follows. When the user selects one instance of Figure 9.8, we use a custom interaction script on the mapper of Figure 9.8 to pop up another window to display the instance attributes.

The selected instance is shown as having the balloon pop-up label in Figure 9.8. Figure 9.10 shows the attributes of the selected instance, in this case Rembrandt: the painter's year of birth, year of death, first name, etc.

9.4.3 Application Model Visualization

The application model visualization enables one to better understand the navigation structure of a hypermedia presentation. It answers questions such as:

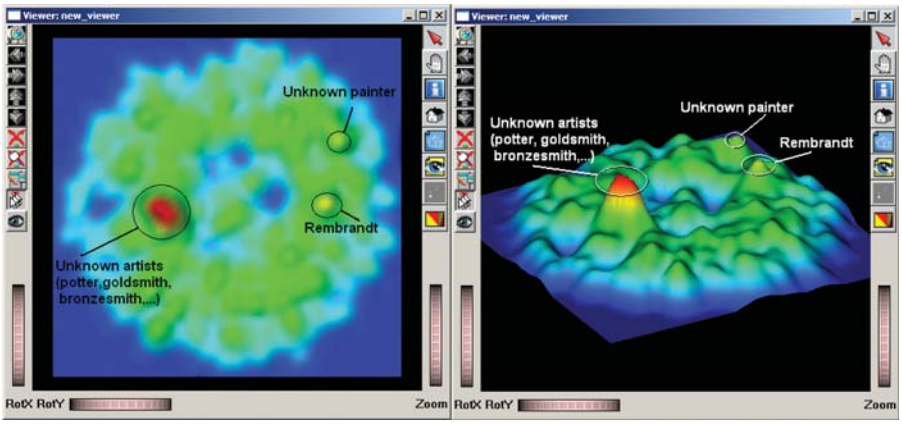


Figure 9.9 Conceptual model instance splatting (left: 2D; right: elevation plot).

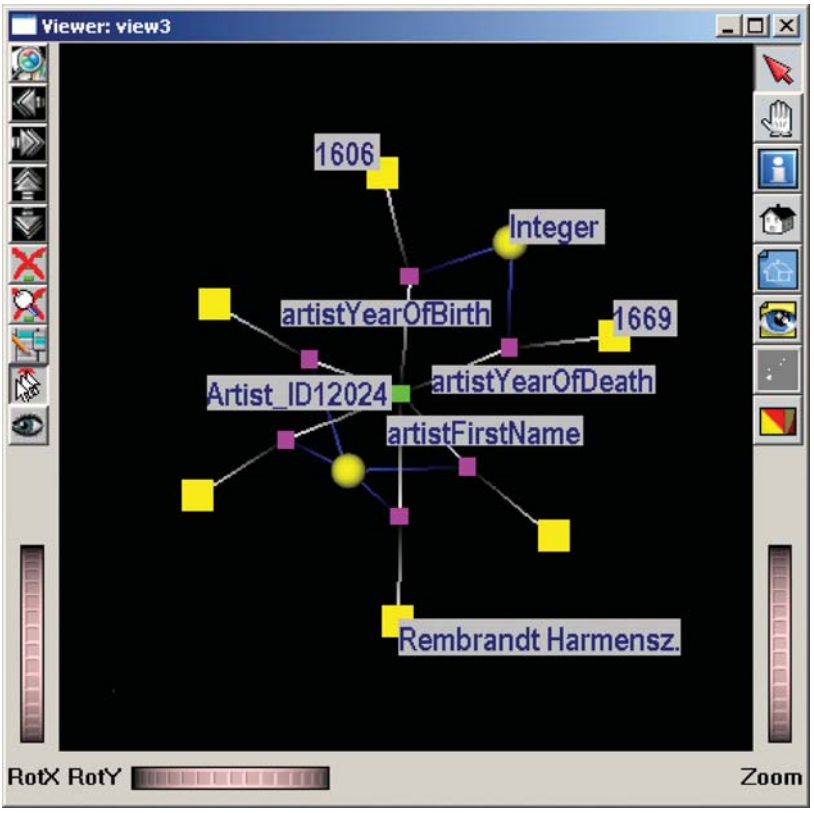


Figure 9.10 Attributes of a selected concept instance.

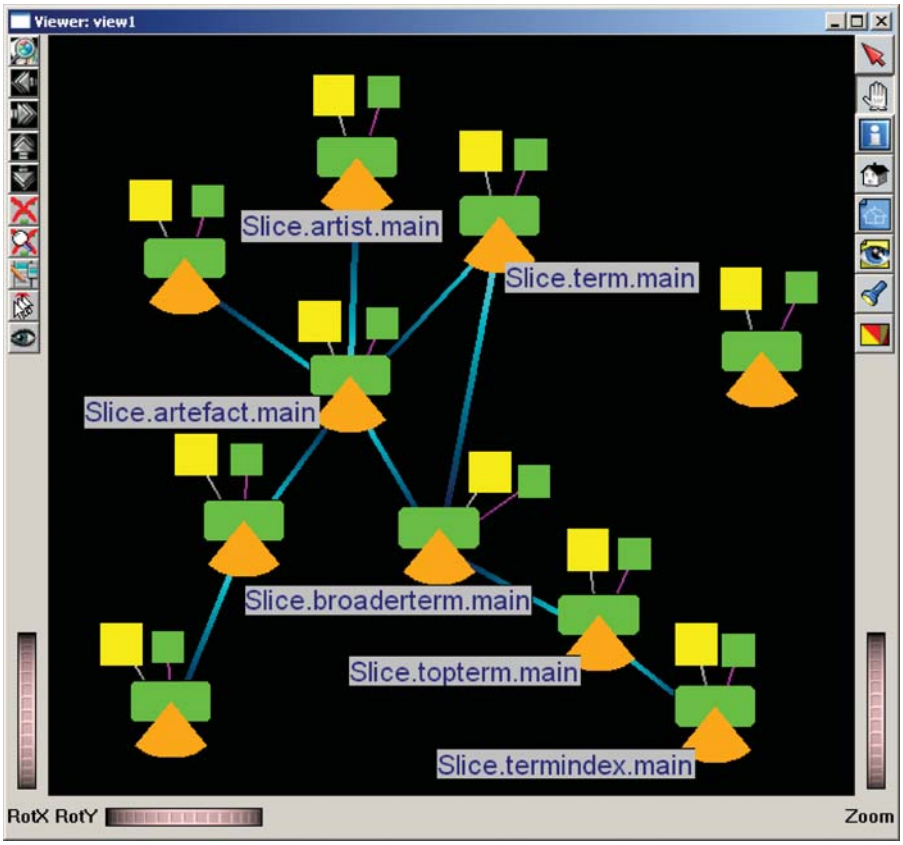


Figure 9.11 Museum application model.

What are the application model slices?
What are their links?
What are the slice owners?
What are the slice titles?
What slices are navigation hubs?
What are possible navigation paths from a certain slice?

Figure 9.11 depicts the application model for the museum example. We chose to present here the top-level slices (slices that correspond to Web pages) and the links between them in order to decrease the complexity of the picture. A new glyph shape was designed in order to represent the pizza-slice shape for slices (as defined in the application model's graphical representation language). The blue thick edges represent links between slices. Each slice has associated with it two attributes. We use a custom layout to place these nodes right above the top of the slice node. The slice nodes themselves are laid out using the spring embedder already discussed before. The two attributes of each slice are visualized by using two custom square glyphs, as follows: the yellow glyph (left) stands for the name of the slice and the green glyph (right) denotes the concept owner of the slice (remember that the concept owner is a concept from

the conceptual model). In the center of the picture is the `Slice.artefact.main` slice, which has the most links associated with it (i.e., it is a navigation hub). The figure also shows the designer's choice to present the museum information based on the terms hierarchy: top terms, broader terms, and terms.

9.4.4 Application Model Instance Visualization

The application model instance visualization answers questions like:

What are the instances of a certain slice?
What are the slice instances reachable from a certain slice instance?
What are the most referenced slice instances?
What are the attributes of a selected slice instance?

As there are more slice instances than slices, in order to keep the visualization manageable we used the same visualization scenario as for conceptual model instances (i.e., to use incremental views). The user can select from the mapper in Figure 9.11 the slices for which she wants the corresponding instances to be visualized. For example, after selecting the `Slice.topterm.main`, `Slice.broaderterm.main`, and `Slice.term.main` slices from Figure 9.11, we use the same mechanism of a custom interaction script (Section 9.3.2) to pop up another window that shows the instances of these slices and their associated links. Figures 9.12 and 9.13 show the corresponding slice instances, as described below.

For the visualizations in Figures 9.12 and 9.13, we use yellow sphere glyphs for nodes labeled `Slice.topterm.main`, green sphere glyphs for nodes labeled `Slice.broaderterm.main`, and blue rectangle glyphs for nodes labeled Slice.term.main. The chosen colors and shapes are motivated by the need to produce an expressive, easy-to-understand picture when presenting a large number of instances coming from three slices linked in a hierarchical way, as follows. We did give up the pizza-slice glyph for these visualizations as we found out that this glyph produces too much visual clutter for large graphs. Next, we chose colors of increasing brightness (blue, green, and yellow) to display items of increasing importance (terms, broader terms, and top terms, respectively). The size of the glyphs used for these items also reflects their importance (the top term glyphs are the largest, whereas the term glyphs are the smallest). A final significant cue is the shape: the more important top and broader terms are drawn as *3D shaded* spheres, whereas the less important terms are drawn as *2D flat* squares. For the edges connecting these glyphs in the visualization, we used a varying color and size scheme that varies both line color and line thickness along the edge between the end nodes' colors and sizes respectively. Summing up, the combination of above choices produces a visualization where the overall structure of top terms and broader terms "pops" into the foreground, whereas the less important terms and their links "fade" into the background. As a comparison, we were unable to get the same clear view of the structure by just varying the layout parameters and using the same glyph for all nodes.

After selecting the slice instance corresponding to the `Paintings` top term, we obtain in Figure 9.12 the broader term slice instances accessible *after one step*, shown in red. By this, we mean the terms that a user of the Web site (whose design our dataset captures) can access after one navigation step. This translates to nodes that are directly connected (via an edge) to the selected slice instance in our RDF dataset. In Figure 9.13 we visualize the term slice instances accessible from the same `Paintings` top

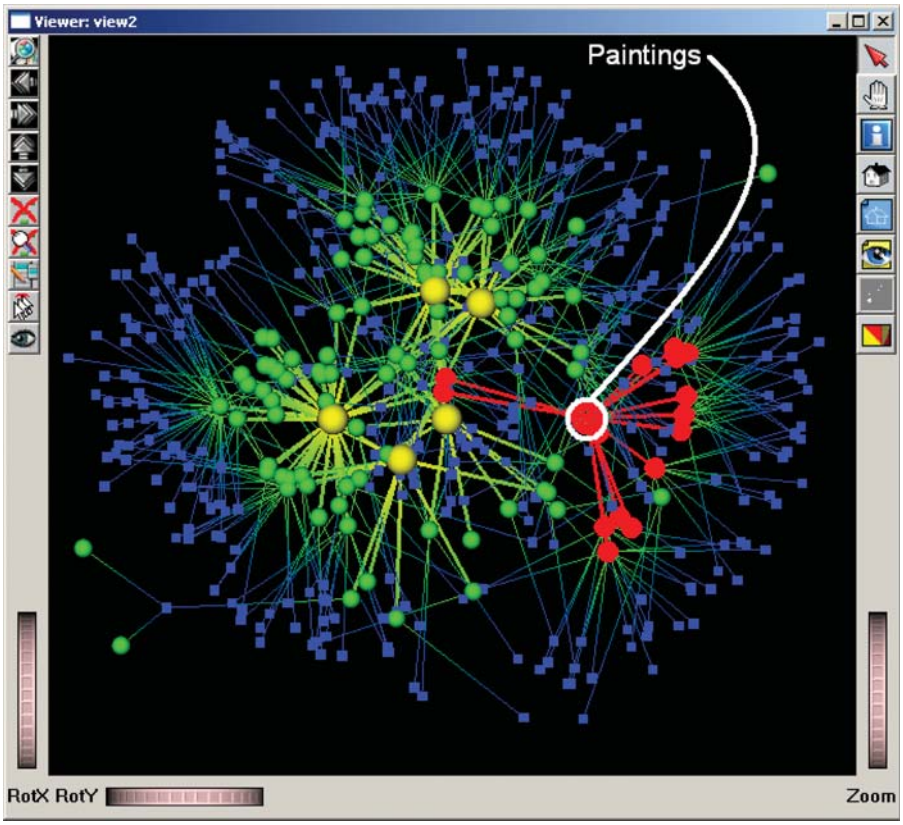


Figure 9.12 Broader term slice instances accessible from the `Paintings` slice instance (one step).

term instance slice *after two steps*, also drawn in red. These correspond to Web pages that the user of the Web site can access after two navigation steps. An example for the second step is the navigation from the broader term `Portraits`. Using such a visualization scenario one can view which are the slice instances reachable from a selected slice instance after a certain number of navigation steps.

9.5 Future Work

In the future, we would like to explore the GViz 3*D* visualization capabilities for RDF data, possibly getting an even better insight into the data structure. Another research direction would be to use GViz in conjunction with a popular RDF query language (like RQL, for example). Our purpose is here twofold: to use the RDF query language as a selection operation implementation for GViz when visualizing RDF data and to support the RDF query language with the visualization of the input and resulting set of RDF data. Finally, as it is planned in the Hera project to use OWL instead of RDF for the future input data/design specifications we would like also to conduct visualization experiments on the more semantically rich OWL data.

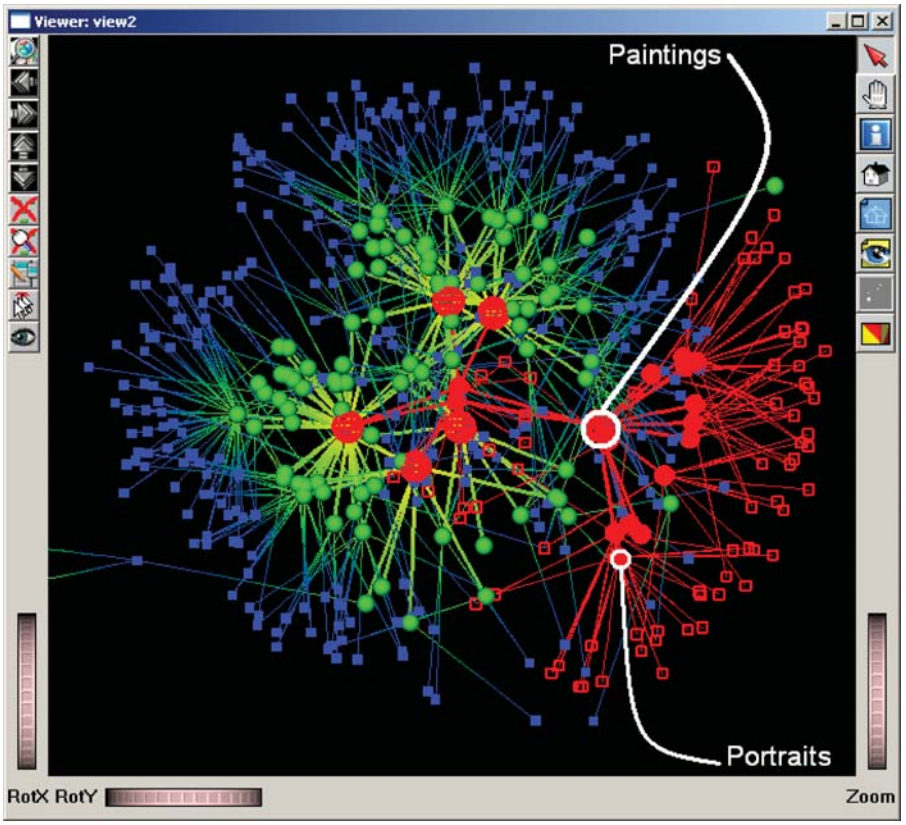


Figure 9.13 Term slice instances accessible from the `Paintings` slice instance (two steps).

9.6 Summary

In this chapter we have shown how a general-purpose graph visualization tool, GViz, can be used for the visualization of large RDF graphs produced from real-world data. All experiments were performed in the context of the Hera project, a project that investigates the designing and developing of Web Information Systems on the Semantic Web. The visualization of large amounts of RDF input data and RDF design specifications enabled us to answer complex questions about these data and to give an effective insight into their structure.

Several ingredients were crucial for obtaining these results. First, the amount of customizability of the GViz tool (layouts, selections, node and edge drawing, choice between glyph and splat mappers, and custom user interaction) was absolutely necessary to produce the desired visualization scenarios. We found all these elements to be necessary to create the desired results. We have actually experimented with customizing just the layout but not the glyphs and/or the interaction. In all cases, the results were not flexible enough to give the users the desired look-and-feel that would make the scenario effective for answering the relevant questions. Second, the script-based customization mechanism of GViz allowed a user experienced with Tcl scripting to

produce the scenarios described here (which were imagined by a different user, inexperienced with Tcl) in a matter of minutes. Third, we found that using several visual cues (shape, color, size, and shading) together to enhance a single attribute, as for example described in Section 9.4.4, is much more effective than using a single cue. Finally, we mention that none of the investigated RDF visualization tools (Section 9.2) showed the high degree of customization of GViz needed for our scenarios.

9.7 Acknowledgments

The authors would like to thank the Rijksmuseum in Amsterdam for their kind permission to use copyrighted data in our running example. We also wish to thank our colleagues Lloyd Rutledge and Lynda Hardman, from CWI, Amsterdam, for helpful discussions.

9.8 References

Card, S., Mackinlay, J., Shneiderman, B. (1999). *Readings in Information Visualization.* Morgan Kaufmann.

Eklund, P.W., Roberts, N., Green, S.P. (2002). OntoRama: Browsing RDF ontologies using a hyperbolic-style browser. *The First International Symposium on CyberWorlds (CW2002), Theory and Practices,* IEEE CS Press, pp. 405–411.

Grove, M. (2002). *RDF Instance Creator (RIC) for the MIND-SWAP Project.* Available: http://www.mindswap.org/∼mhgrove/RIC/RIC.shtml.

North, S.C., Koutsofios, E. (1996). *DOT and NEATO User's Guide.* AT&T Bell Labs Reports.

Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A. (2001). Creating Semantic Web contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71.

Pietriga, E. (2002). IsaViz: A visual environment for browsing and authoring RDF models. *The Eleventh International World Wide Web Conference (WWW2002), Developer's Day.*

Raines, P. (1998). *Tcl/Tk Pocket Reference.* O'Reilly & Associates.

Sintek, M. (2004).*OntoViz Tab: Visualizing Protégé Ontologies.* Available: http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

Storey, M.-A.D., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N. (2001). Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001).*

Sugiyama, C., Tagawa, S., Toda, M. (1981). Methods for visual understanding of hierarchical systems. *IEEE Transactions of System, Man, and Cybernetics*, 11(2):109–125.

Sure, Y., Angele, J., Staab, S. (2003). OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, LNCS 2800, pp. 128–152, Springer.

Telea, A., Maccari, A., Riva, C. (2002). An Open Toolkit for Prototyping Reverse Engineering Visualization. *IEEE EG Symposium on Visualization (VisSym'02)*, pp. 241–250. The Eurographics Association.

Telea, A. (2004). An open architecture for visual reverse engineering. *Managing Corporate Information Systems Evolution and Maintenance*, pp. 211–227, Idea Group Inc.

Van Liere, R., De Leeuw, W. (2003). GraphSplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, IEEE CS Press.

Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P. (2003). Engineering Semantic Web information systems in Hera. *Journal of Web Engineering (JWE)*, 2(1–2):3–26, Rinton Press.

Wernecke, J. (1993). *The Inventor Mentor: Programming Object-Oriented 3D Graphics.* Addison Wesley.

Chapter 10

Spring-Embedded Graphs for Semantic Visualization

Jennifer Golbeck and Paul Mutton

10.1 Introduction

Implicit information embedded in Semantic Web graphs, such as topography, clusters, and disconnected subgraphs, is difficult to extract from text files. Visualizations of the graphs can reveal some of these features, but existing systems for visualizing metadata focus on aspects other than understanding the greater structure. In this chapter, we present a tool for generating visualizations of ontologies and metadata by using a modified spring embedder to achieve an automatic layout. Through a case study using a mid-sized ontology, we show that interesting information about the data relationships can be extracted through our visualization of the physical graph structure.

The Semantic Web is based on the idea of creating "machine understandable" data that can be used and exchanged. Using Web languages, such as RDF, DAML+OIL, and OWL, it is possible to create semantically rich data models. These models are made up of triples (subject-predicate-object), where subjects and objects are entities, and predicates indicate relationships between those entities. Users can define their own properties, as well as their own classes. Classes in Semantic Web languages are categories or types, similar to how classes work in programming languages. Instances of these classes can then be created and described with values for related properties.

Implicit in these models is more information than can usually be found in their text representation. Each triple forms a graph with two nodes connected by an edge. Each instance can have several properties, and that graph expands to have many nodes connected to the central instance. Finally, when two instances are connected via a property, their respective subgraphs become connected. The graphs produced from RDF triples contain more information than just which entities are related to which. Implicit information, such as the underlying structure of a data model or which instances are most closely connected, is all contained in a graph. This information, though, is difficult if not impossible to extract from a text-based reading of the data.

Since the Semantic Web is so new and the languages themselves are still quickly evolving, much of the research in this area has focused on editors, applications, tools, and languages. Tools for viewing the data have been primarily text based. The few graphical visualization tools have focused on other aspects of the data and their use.

Here, we present a tool for generating graphs of ontologies and instance data on the Semantic Web. We apply graph layout algorithms that attempt to place related nodes near to each other while keeping all other nodes evenly distributed. The resulting

graph drawings give the user insight into the structure and relationships in the data model that are hard to see in text.

10.2 A Suitable Graph Drawing Algorithm

Graphs are often used to visualize relationships and patterns between entities. Graph drawing methods are important in such visualizations for providing automatic layout of entities and their relationships. A good layout can ease user exploration and make it easier to detect patterns in the data. We define a graph G = (N,E), where N is the set of node entities and E is the set of directed edge relationships, each between a pair of nodes.

Spring embedding (Eades, 1984; Fruchterman and Reingold, 1991) is one such graph drawing method that is suitable for application to our data. Its effect is to distribute nodes in a two-dimensional plane with some separation, while attempting to keep connected nodes reasonably close together. The spring embedder graph drawing process considers the graph model as a force system that must be simulated. Each node in the graph is modeled as a charged particle, thereby causing a repulsive force between every pair of nodes. Each edge is modeled as a spring that exerts an attractive force between the pair of nodes it connects. The graph is laid out by repeated iterations of a procedure that calculates the repulsive and attractive forces acting on all nodes in the graph. At the end of each iteration, all nodes are moved according to the resultant forces acting on them. (See Figure 10.1.)

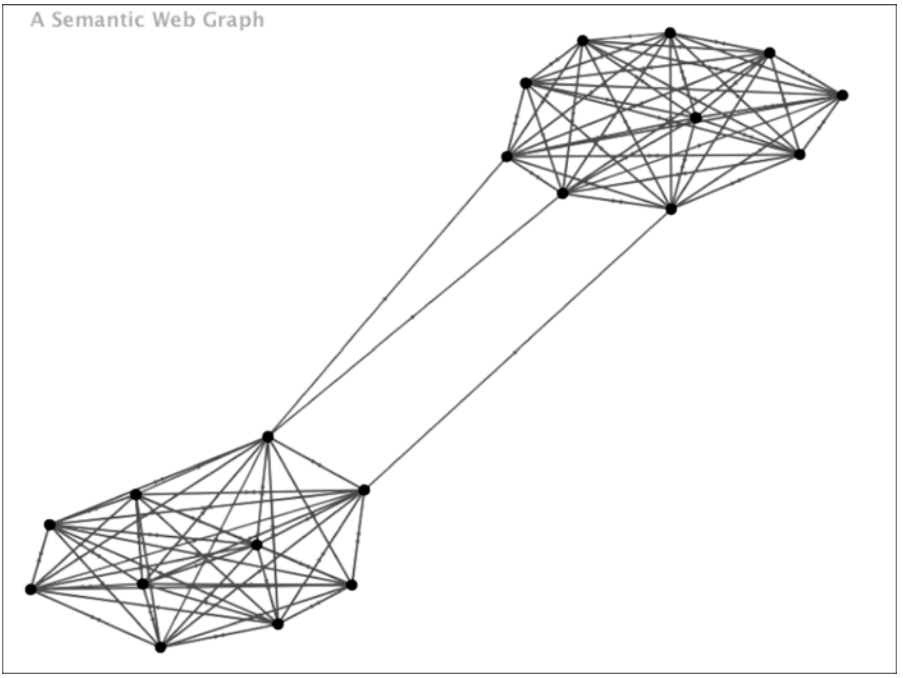


Figure 10.1 Spring embedded ontology.

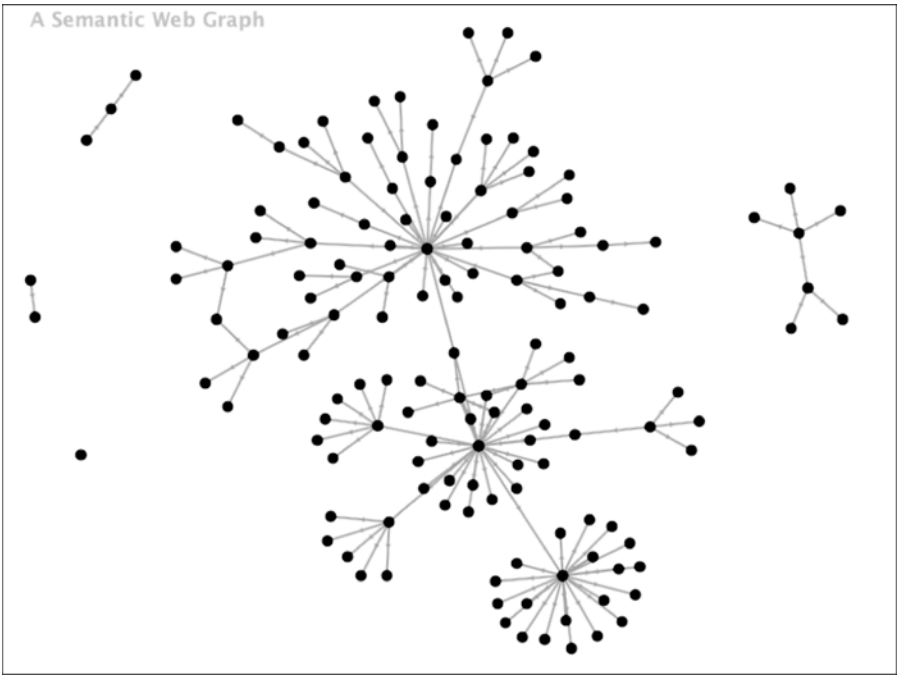


Figure 10.2 A disconnected graph.

The force models that we use for the spring embedder are based on those of Fruchterman and Reingold (1991). This version of the spring embedder is effective and widely used. It is also relatively easy to implement and requires a minimal set of parameter values that can be adjusted to achieve good automatic layouts. In this model, the repulsive force acting between a pair of nodes is $-k^2/d$ and the attractive force due to an edge is d^2/k, where d is the distance between the two nodes and k is a constant. We start our graph drawing process by allocating each node to a random location on a two dimensional plane and then we begin the iterative calculation of these forces and move nodes accordingly. This results in a layout where connected nodes are close together, yet no pair of nodes are too close to each other due to the repulsive forces acting between them.

When visualizing ontologies and instances, not all graphs we encounter will be connected. (See Figure 10.2.) With a simple spring embedder model, this can cause the layout to rapidly expand, as there is nothing to counter the repulsive forces acting between each of the largest connected subgraphs. We solve this problem by limiting the distance over which repulsive forces may act. A pair of nodes with separation greater than m does not exert a repulsive force. This alteration to the force model ensures that we do not end up with an unnecessarily sparse graph drawing, which can be caused when nodes drift apart for every stage of the simulation. We refer to this variation as the *m-limited* force model. (See Figure 10.3.)

A simple implementation of the spring embedder calculates the repulsive force between every pair of nodes and so has a time complexity of $O(N^2)$ per iteration. In practical terms, this limits the maximum size of our drawings to several hundred nodes

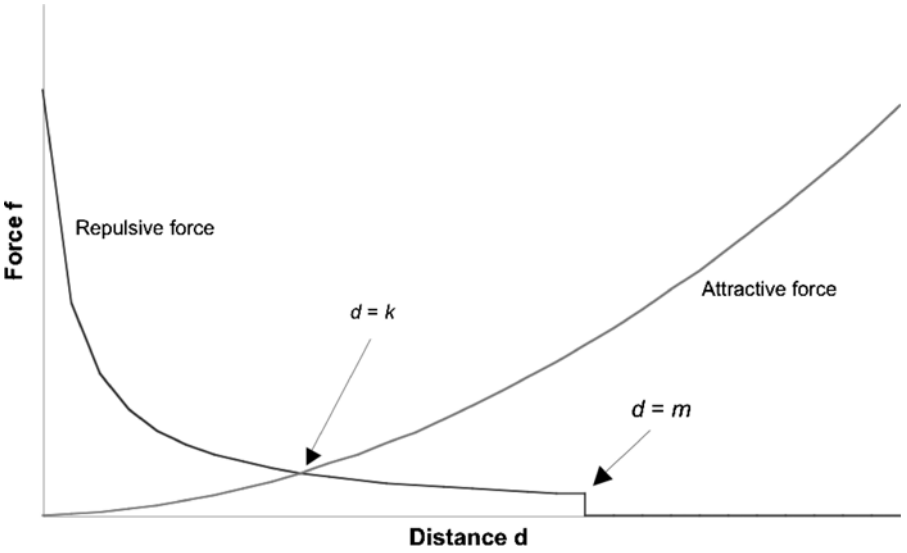


Figure 10.3 The m-limited force model.

if we want them to be drawn in less than one second on affordable hardware. Various optimizations exist to make this process quicker, such as preprocessing the initial random layout with linear time complexity (Mutton and Rodgers, 2002), speeding up the calculation of forces between pairs of nodes (Tunkelang, 1998), or reducing the number of nodes that are paired (Tunkelang, 1998; Eaders, 2001). Multilevel approaches (Harel and Koren, 2001; Walshaw, 2001) provide a heuristic method that clusters a graph and lays out the coarsened graph, reintroducing the other nodes in uncoarsening steps until a final drawing is produced. These can be used to reduce the time complexity of each spring embedder iteration to O(N*log*N) without any significant reduction in its effectiveness, making the method suitable for application to graphs with tens of thousands of nodes in real-time.

In this implementation, the algorithms work well on graphs with up to several hundred nodes. Clustering nodes and producing a hierarchical navigation structure with an improved implementation of the spring embedder would be useful when dealing with larger graphs. A common observation worth noting is that the spring embedding process as a whole requires a greater number of iterations to reach an equilibrium with larger graphs, so the overall time complexity is greater than that stated for a single iteration.

10.3 The Tool

The graph drawing algorithm described above is combined with an RDF and OWL parser to make it capable of reading in graphs of Semantic Web data. Each Semantic Web triple becomes part of the graph, with the subject and object represented as nodes and the predicate as a directed edge connecting them. This representation is blind to the type of each part of the triple, so classes, instances, and literal values all are represented as nodes in the graph. The one modification we made was to ignore the

namespace information and other administrative triples, since these would otherwise form an isolated independent subgraph that may confuse the visualization of the actual data.

10.4 Case Study 1: Visualizing Ontologies—Zoological Information Management System

The Zoological Information Management System (ZIMS) is a project of the International Animal Data Information Systems Committee (IADISC). As they state, "Standardized data collection and animal records are very important to the zoo and aquarium profession. They are critical for basic animal management and for achieving our conservation, research, and education goals. Unfortunately, zoological facilities are currently struggling with outdated software and inconsistent records that hinder our ability to efficiently and scientifically manage the animals in our care. A most notable inadequacy of the current system is that it does not track the history of group animals, such as fish and invertebrates, or environmental conditions to the extent necessary for many aquatic collections. In an effort to solve this problem, an international, coordinated effort, called the Zoological Information Management System (ZIMS) Project, is underway to improve how animal data is managed" (DuBois, 2003).

Over a series of workshops, the ZIMS project has produced an evolving conceptual data model (CDM). The CDM is designed to clearly define rules and concepts associated with a particular area from a data management perspective. This data model contains over two hundred concepts and relations. Concepts, also called entities, are explicitly described in Table 10.1.

Figure 10.4 shows the visualization of the ZIMS CDM in its current format as a UML class diagram. The Unified Modeling Language (UML) is a standard language for modeling systems. UML class diagrams represent classes as boxes and lines that connect two classes describe the relationship. At the endpoints of the edges, symbols indicate the type of relationship. The CDM diagram is a large UML diagram, but typical of the layout used when visualizing a UML mode.

Table 10.1 A sample set of terms and descriptions taken from the ZIMS Conceptual Data Model

Entity type	Description
Animal	Represents all mammals, reptiles, birds, fish, etc. that are managed by a zoo or aquarium and includes actual or estimated birthdate, which may be postdated to capture information prior to an animal's birth. All relationships apply to individual animals or on a percentage basis for group animals.
AnimalBusinessTransactionTransferPhysical	Captures all animals that are part of a physical transfer business transaction.
Animal-Collection	Captures the historical ownership of animals at a zoo or aquarium and may include a localized accession number. Note: Partial ownership of animals may be added later but is not currently considered part of core requirements.
AnimalEnclosure	Captures which animals were in which enclosure at a given point in time and supports multispecies displays and includes datetime.

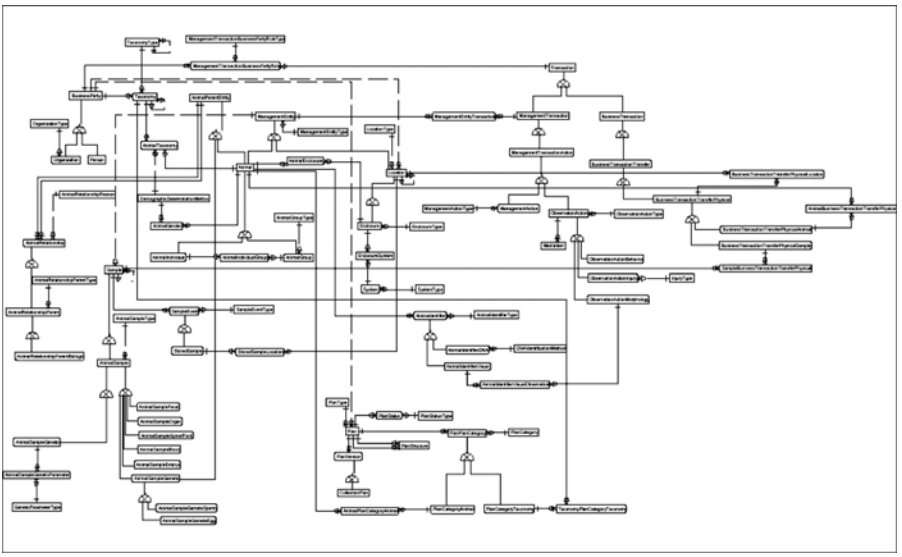


Figure 10.4 The ZIMS Conceptual Data Model showing entities and relations.

While UML diagrams are often very useful for describing the structure of a system, they do not provide much information about that structure from a visualization standpoint. Someone reading the diagram must look closely at each class to see how it relates to others and through what properties. Looking at the CDM diagram in Figure 10.4, it is clear that it does not visually convey much information about the nature relationships between entities. A user could certainly follow the links along to find relationships between entities, but the information is not easily visually accessible. Furthermore, there is no way to easily identify which groups of entities form closely related clusters of information and which are just adjacent by chance. This is not a downfall of UML diagrams, but rather is just outside their purpose. Another picture is necessary to achieve a visual overview.

The ZIMS conceptual data model has been represented as an OWL ontology, describing the entities as classes, and the relationships between them as properties. This conversion to a standardized form, understandable by intelligent Web agents, is a great step toward improving the way zoological information at a wide range of distributed centers is coherently managed. The representation puts the data into a form where they can be visualized by the graph drawing system described here.

Figure 10.5 shows the same CDM from Figure 10.4 as a spring embedded graph. For visual clarity, the labels have been removed from edges and nodes, though in application form they can be accessed as tool tips. The spring embedded graph reveals several details about the general graph structure that are not clear from the CDM diagram.

First, there is a central ring structure, made up of connected clusters. These clusters are groups of closely interrelated concepts. One example is a collection of entities in the CDM describing business transactions. Semantically, it makes sense that these terms would be connected to one another, but the clustered relationship is clear from neither the text of the ontology nor the original UML diagram.

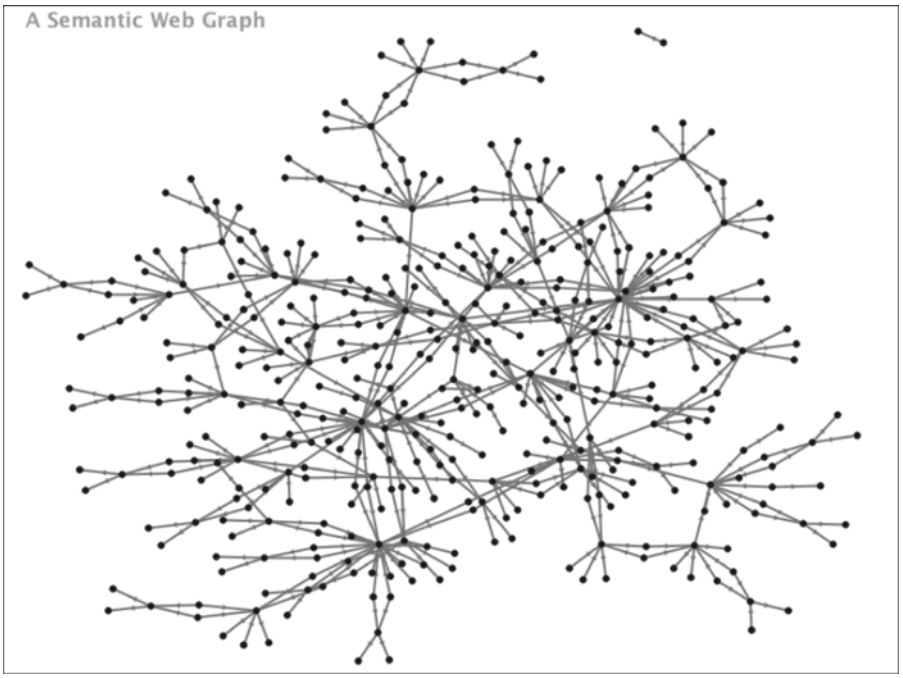


Figure 10.5 A spring embedded graph of the OWL version of the ZIMS CDM. This graph displays the entities and relations as well as other data associated with each entity.

Long chain-like structures also appear extending off around the edges of the graph. These show sequentially linked concepts, and reveal the presence of potentially important indirect relationships between the concepts at the start and end of the chain. For example, one of the longest of these chains begins with "Animal" and ends with "AnimalParentRelationshipBiologic." Again, it is intuitive that these two concepts should have some semantic relation, and through one intermediate concept and a simple string of subclasses, the two are chained together in the graph. However, the five steps separating the two concepts make it nearly impossible to recognize this relationship through text. Though tracing the path through the original CDM diagram is not difficult, there are no visual clues that would indicate it without close inspection.

By adjusting parameters of the layout algorithm, we are able to generate the graph in Figure 10.6 with data points grouped more tightly into clusters. Though some of the nodes obscure one another, the clustering makes the structural features of the ontology all the more obvious.

10.5 Case Study 2: Visualizing Instance Data—Social Network Visualization

Social networks are a particularly clear example of when it is difficult to gain an understanding of the graph structure without a visualization. The Friend-Of-A-Friend

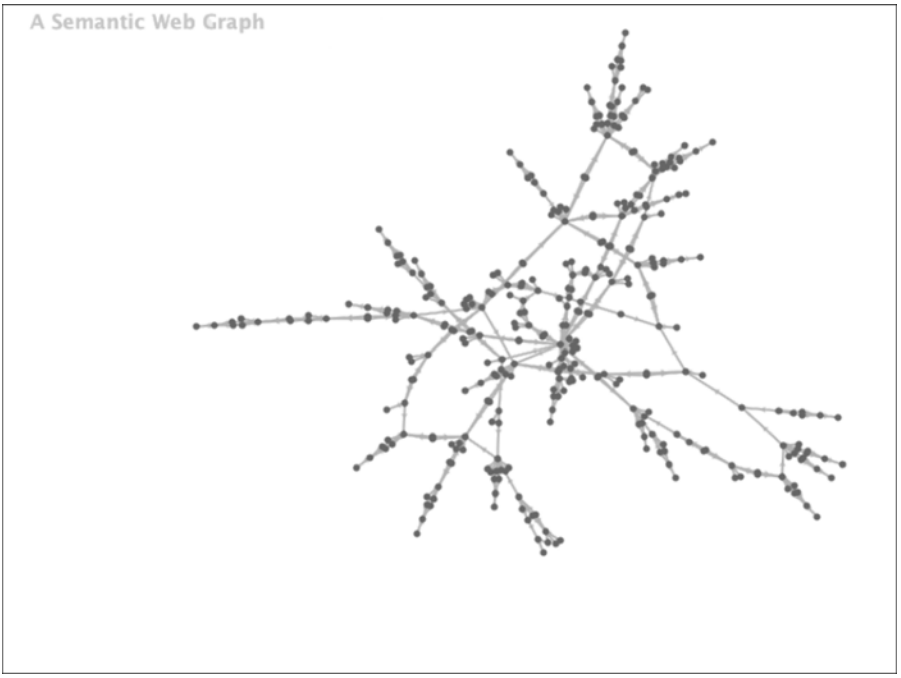


Figure 10.6 Spring embedded graph of the ZIMS CDM with tight clustering.

(FOAF) project is a Semantic Web-based social network vocabulary. Users create files with data describing themselves and their connections to other people. There are millions of FOAF files and users with data spread across the Semantic Web. Because FOAF is one of the most dominant sources of data on the Semantic Web reinforces the need for a mechanism to visualize social networks.

Our spring embedded graph algorithm can be used to produce intuitive social network visualizations. It has been incorporated into the Trust Project, a Semantic Web–based social network augmented with trust relationships that extends the FOAF project (Golbeck, 2004). There are nearly 2,000 individuals in that network, and it is a reasonably sized social network that clearly illustrates the benefits of visualization.

Figure 10.7 shows a graph of the entire trust network. Each node represents a person and each edge is a relationship between two people. The network comprises data aggregated from over 600 files. The most striking feature of this graph is that there is a large cluster of interconnected people in the center, some of whom have rated very large groups of otherwise unconnected people, surrounded by 42 independent subgraphs. These represent small groups of people who have ratings for one another, but have not rated or been rated by any person in the large cluster. Recognizing these independent groups is interesting for what it exposes about how the network evolves, and also for the information it reveals about how other algorithms that utilize the network may perform. Because there are many individuals disconnected from the main cluster, network analysis methods that look for paths may produce unusual results if they encounter these independent graphs while expecting a connected graph.

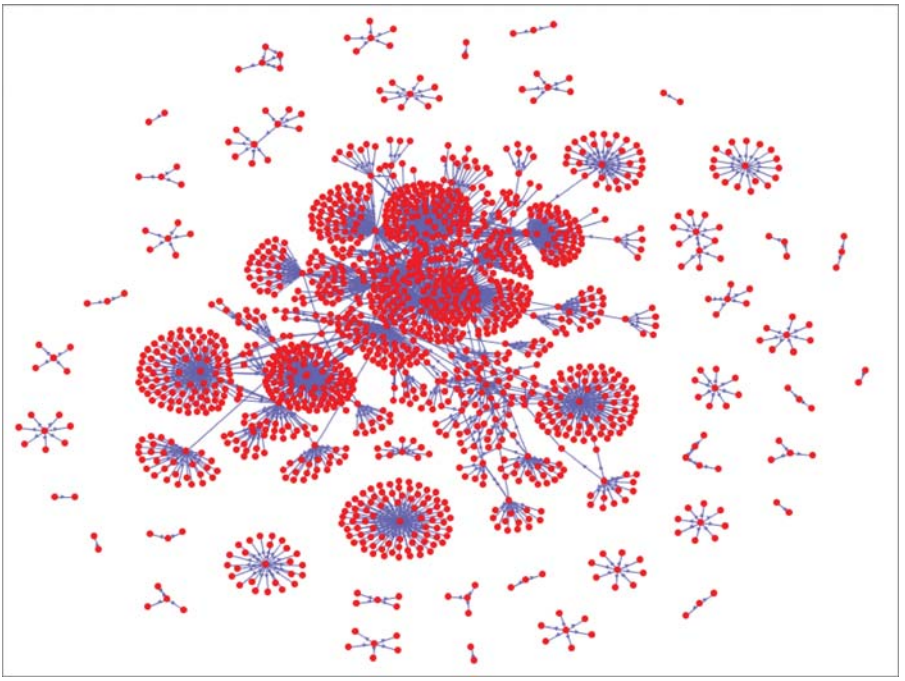


Figure 10.7 A visualization of the Semantic Web–based social network aggregated as part of the Trust Project.

In addition to providing information about the general structure, the spring embedded graphs help identify the position of specific nodes within the network. Because individuals are connected in by the connections in their files *and* connections other individuals may have expressed to them, users cannot always know where they fit into the larger system. Figure 10.8 shows the individual highlighting feature, where two different nodes in different positions are shown.

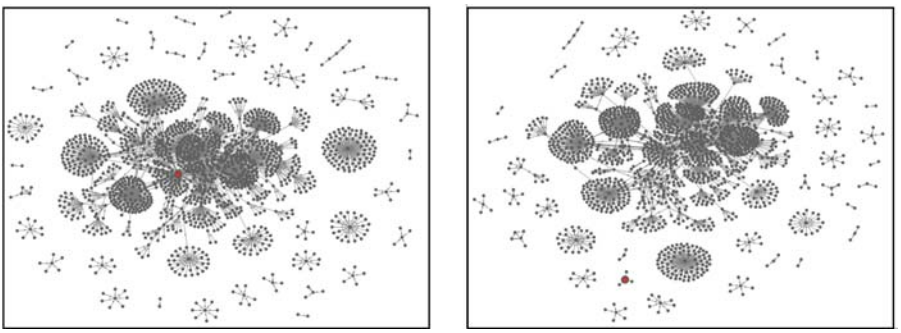


Figure 10.8 Two sample graphs where individual nodes have their position highlighted. The color has been altered here to make the highlighting more prominent on these small images.

10.6 Future Work

The next step in expanding this graph drawing system is to make it an effective tool for ontology browsing. In its current state, users can mouse over a node or edge, and see its URI. We identify two features that, when added to the graph drawing system, will increase its power to illustrate the structure of the ontology and to provide information about the elements.

The first is to add in elements of a more traditional ontology browser. Brownsauce (http://brownsauce.sourceforge.net) is one browser that shows all of the properties and values associated with a given instance, coupled with links to related information. Giving the user an option to click a node and see this information adds a lot of power to the exploratory process. The second improvement is to add a dynamic query interface. Dynamic queries allow the user to rapidly adjust query parameters and see those changes reflected in the visualization in real-time (Ahlberg et al., 1992). When viewing instance data, for example, users could select specific classes and see only instances of those classes reflected in the graph. By adding and removing instances in this fashion and quickly seeing the results in the graph, patterns about which instances are closely grouped, or how instances of specific classes are connected, become clearer. This in itself is powerful and leads the way to integrating a more robust query interface to the visualization if it appears to be useful.

10.7 Conclusions

In this chapter we have presented a graph drawing system for visualizing ontologies and collections of instance data on the Semantic Web. Related entities are drawn close to each other with a directed edge to symbolize the relationship, and the system is also capable of producing sensible automatic layouts of disconnected graphs. Through two case studies involving a real, deployed ontology and an aggregated set of instance data, we show how patterns about the underlying structure are more easily understood through the graph drawing than through text or other types of visual displays.

10.8 References

Ahlberg, C., Williamson, Shneiderman, B. (1992). Dynamic queries for information exploration: An implementation and evaluation. *Proceedings of Conference on Human Factors in Computer Systems 1992,* pp. 619–626.

DuBois, S. (2003). The ZIMS project: Building better animal information systems for zoos and aquariums. *Presented at the 2003 Management and Technology Conference.*

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium,* 42:149–60.

Fluit, C., Sabou, M., van Harmelen, F. (2002). Ontology-based information visualization. *Proceedings of Information Visualization '02.*

Fruchterman, T., Reingold, E. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience,* 21(11):1129–1164.

Golbeck, J., Hendler, J. (2004). Reputation network analysis for email filtering. *Proceedings of the First Conference on Email and Anti-Spam.*

Harel, D., Koren, Y. (2001). A fast multi-scale method for drawing large graphs. *Proceedings of the 8th International Symposium on Graph Drawing (GD 2000),* pp. 183–196.

Mutton, P., Rodgers, P. (2002). Spring embedder preprocessing for WWW visualization. *Proceedings of International Symposium on Web Graphics and Visualization (IV02-WGV).*

Quigley, Eades, P. (2001). FADE: Graph drawing, clustering, and visual abstraction. *Proceedings of the 8th International Symposium on Graph Drawing (GD 2000)*, pp. 197–210.
Tunkelang, D. (1998). JIGGLE: Java interactive graph layout algorithm. *Proceedings of the 6th International Symposium on Graph Drawing (GD 1998)*, pp. 413–422.
Walshaw, C. (2001). A multilevel algorithm for force-directed graph drawing. *Proceedings of the 8th International Symposium on Graph Drawing (GD 2000)*, pp. 171–182.

Chapter 11

Semantic Association Networks: Using Semantic Web Technology to Improve Scholarly Knowledge and Expertise Management

Katy Börner

11.1 Introduction

This chapter introduces *Semantic Association Networks* (SANs), a novel means of using Semantic Web technology to tag and interlink scientific datasets, services (e.g., algorithms, techniques, or approaches), publications (e.g., papers, patents, grants), and expertise (i.e., author and user information) to improve scholarly knowledge and expertise management. Among other ends, the proposed SANs:

- Facilitate new types of searches (e.g., the retrieval of all authors that worked with dataset *x* or all papers that used algorithm *y*).
- Ease the reuse of datasets and services, thus increasing the reproducibility of results.
- Enable dataset/algorithm/result comparisons at the data/code/implication level.
- Exploit data access and data origin logs to indicate the usefulness of resources and the reputation of authors.

Section 11.2 outlines the need for improved scholarly knowledge and expertise management by discussing major trends in the evolution of science and our current means to access scholarly knowledge and expertise. Section 11.3 introduces *Semantic Association Networks*, and their usage for improving storage, correlation, analysis, access, and management of scientific knowledge and expertise. Section 11.4 discusses opportunities and sociotechnical challenges for the implementation of SANs. Section 11.5 contains concluding remarks and presents a vision for the future of scholarly publishing knowledge and expertise management.

The content and style of this chapter were motivated by my research on knowledge domain visualizations (KDVis) (Börner et al., 2003; Shiffrin and Börner, 2004). KDVis apply advanced data mining and information visualization techniques to analyze, correlate, and visualize the semantic space of researchers, publications, funding, etc. The resulting visualizations can be utilized to objectively identify major research areas, experts, institutions, grants, publications, journals, etc., in a research area of interest. In addition, they can assist in the identification of interconnections, of import and export of research between fields, of the dynamics (speed of growth, diversification) of scientific fields, scientific and social networks, and the impact of strategic and applied

research funding programs. Hence, KDVis give researchers and practitioners a global view of the structure and evolution of a research area.

High-quality datasets that report scholarly activity are required to map science on a large scale and in a comprehensive manner. Consequently, I spent a considerable amount of my time gaining access to high-quality datasets including publications, patents, and grants. All datasets are stored in a multiterabyte database to facilitate the cross-correlation and search of the diverse datasets, for example, interlinking author and investigator names from diverse publication and grant datasets, linking papers/patents to cited papers/patents, etc. While creating this database, I was amazed to learn how little care we take of our collective scholarly knowledge. While some communities support excellent efforts—such as the digitization of all *Physical Review* papers going back to 1893—many institutions and organizations do not have the funds or have not prioritized the digitization and preservation of datasets for general or expanded use. Vast quantities of valuable, irreplaceable, scholarly digital datasets have already been lost forever due to the implementation of new data formats or the migration of computer systems.

However, access to high-quality, cross-referenced digital datasets covering decades and centuries of scholarly work, reliably preserved for future generations, is the basis for better means of scholarly data access, management, and ultimately the understanding and fostering of scientific progress. We now have the technological means to digitize, store, and make available scholarly data and to research science using the scientific methods as suggested by Derek J. deSolla Price exactly 40 years ago (1965). It is my hope that this chapter inspires others to work toward the implementation of better means to improve access to scholarly knowledge and expertise and to facilitate the analysis and communication of the structure and evolution of science.

11.2 Scientific Trends and Current Means to Access Knowledge and Expertise

Recently, a number of papers have been published that express a growing dissatisfaction with the existing communication system in terms of what can be published, by what means, for what price, with which latency, and using what copyright (Lynch, 2003; Henry, 2003; Smith et al., 2003; Williams et al., 2003; Van de Sompel et al., 2004). The papers propose to rethink the scholarly communication system in terms of the size and media types of the "published unit," open access, copyright issues, but they also discuss value-added services that improve access and management.

Herbert Van de Sompel and his colleagues discuss a scholarly communication system that scholars deserve (2004). They provide a detailed overview of the changing nature of scholarly research and the resulting new demands on scholarly communication systems. They argue that most of today's research is highly collaborative, network-based and data-intensive and that fundamental changes in scholarly communication are needed. They point out the need for a system that, among other things, offers interoperability across publishing venues and persistent identifiers to publications (and I will later argue also to authors, services, and datasets), and supports navigation across publishing venues (e.g., between journal papers from different publishers but also patents or grants). A scholarly "interoperability substrate" is needed to design value-adding services that are not "vertically" locked; see Figure 11.1.

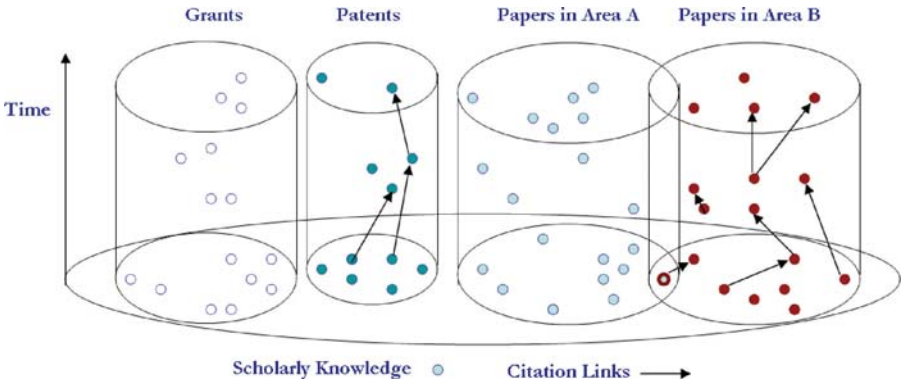


Figure 11.1 The interoperability and cross linkage problem. Many but not all of today's scholarly datasets (e.g., papers, patents, grants) are stored and made available so that "vertical" citation linkages can be traversed. There are very few instances in which datasets of different origin and/or type are "horizontally" interlinked.

Subsequently, I review general trends in the amount and quality of data produced and used by today's scientists, currently available means to store scholarly data, and the knowledge and expertise management that needs to be supported.

Many, if not all, areas of science are experiencing a tidal wave of data. In some fields of science, the volume of data doubles every 18 months (Scholtz, 2000). Data-intensive sciences such as astronomy, geography, biology, and physics generate data at a great rate. Large digital sky surveys are becoming dominant, generating 10 to 100 terabytes—and soon petabytes—per survey (Djorgovski, 2004). Theoretical simulations are also becoming more complex and can generate terabytes of data. The generated datasets are orders of magnitude larger, more complex, and more heterogeneous than in the past.

The advent of the computer, the birth of the WWW, and the widespread availability of digitized information have vastly changed our ways of accessing information (Borgman, 2000). As Richard Shiffrin puts it: "The traditional method involved books, reference works and physical materials on library shelves, most of which had been verified for accuracy by one or another authority. Now, we sit at computers and cast our net into a sea of information, much of which is inaccurate or misleading" (Shiffrin, 2004). Today, a major challenge is to help people navigate the growing, partially interconnected "knowledge webs." People need guidance in the selection of the most correct or relevant data source, support in the examination of search results, and a reliable estimate of the quality of retrieved results. While the (scientific) promise of having a major part of humanity's knowledge at our fingertips is tremendous, the usefulness of this knowledge depends critically on our ability to extract relevant information and to make sense of it.

In this tidal wave of data, scholars act as information sources and sinks; they need to gain access to high-quality information and they are interested in effortlessly diffusing their own scientific results. An infrastructure that aims to support collective scholarly knowledge and expertise management will need to take advantage of the perceptual and cognitive abilities of their human users. It should be technology enabled, but driven by scientific needs. The infrastructure will need to federate existing massive, complex datasets (measures, simulated data, publications), content (data, metadata) services, standards for data representation, and the identification of

analysis/compute services. It should be openly distributed on a large scale so that all scholars can participate, and also dynamically evolving so that future needs can be met.

Over the past decades, diverse systems and approaches have been proposed that support the storage, interlinkage, and linkage-based retrieval of facts. Here I report major historical milestones in chronological order.

The *Shepard's Citations* published for and used by the legal profession since 1873 interlink legal texts, thus making linkage-based search possible. In 1938, Herbert G. Wells described his conception of the future information center in his book *World Brain*, inspiring numerous efforts to create a global repository of interlinked knowledge (1938). Vannevar Bush (1945), in his seminal work, "As We May Think," developed a new concept called "memex" for the automatic storage and efficient access of books, records, and individual communications. Eugene Garfield's work on the *Science Citation Index* was deeply inspired by the *Shepard's Citations.* He envisioned and later implemented the citation indexes for science as an "association-of-ideas index" (Garfield, 1955a). Influenced by V. Bush's ideas, Douglas Engelbart (1963) describes one of the first hypertext systems. Also in 1963, Theodor H. Nelson coined the words *hypertext* and *hypermedia* to describe his vision of worldwide hypertext—a universe of interactive literary and artistic works and personal writings deeply intertwined via hyperlinks. In 1960, Theodor H. Nelson founded *Xanadu* (http://xanadu.com), a project devoted to the design of a system that supports two-way, unbreakable links, deep version management, incremental publishing, document–document intercomparison, copyright simplification and softening, and origin connection (e.g., by retrieving quoted contents from the virtual original of the author or rights holder such that exact royalty payment for each download can be calculated). In 1988, Thinking Machines Inc. developed the *Wide Area Information Server* (WAIS), a distributed system to search index databases on remote computers. In 1989, Tim Berners-Lee architected the *World Wide Web*, an Internet-based hypermedia initiative for global information sharing. Even with its one-way, fragile links and no inherent management of version or contents, the Web—billions of Web pages authored and interlinked by millions of people around our planet—is the largest "knowledge web" in existence today.

For the last decade, our main means of accessing the knowledge stored in digital libraries, repositories, or the Web has been the search engine. Search engine usage resembles "charging a needle" with a search query and "sticking it into a haystack" of unknown size and consistency. Upon "pulling the needle out," one checks the linearly sorted items that got "stuck" on the needle. However, one typically does not get any information on the appropriateness of the haystack probed, the algorithm applied to retrieve the items, or the quality of the data retrieved. Some systems support linkage-based search (e.g., finding all papers that cite paper x or all Web pages that link to page y; see Figure 11.2), but no search engine provides a more global view of the data searched or retrieved.

The dominant usage of search engines in combination with the accelerating pace of information generation and nearly constant human cognitive abilities leads to a general loss of global knowledge on the content of knowledge repositories, and on the general structure and evolution of scientific disciplines. No one today has a global view of humankind's knowledge. In fact, our bird's-eye views are at best one meter above the landscape of science whereas a global view would be possible only from a 100-meter height. This estimate was made based on our work on validating knowledge domain visualizations. Depending on the coverage of the knowledge domain map and

Figure 11.2 Search for concrete facts versus gaining a global overview of the structure and evolution of science.

the expertise of our subjects, 5 to 10 zooms of a 1:3 to 1:10 zooming factor are necessary until subjects recognize the set of papers that they are familiar with. Sadly, our distance above ground is decreasing steadily as the amount of information is growing.

The difference between the search for concrete facts and gaining a global overview of a research domain is illustrated in Figure 11.1. While today's search engines support fact finding and link traversal well, they fail to equip scholars with a bird's-eye view of the global structure and dynamics of scholarly knowledge and expertise. However, a more global view is needed to understand the structure and dynamics of science as it is conducted today, to detect emergent research frontiers (e.g., based on highly cited research articles), and for many other related tasks (e.g., setting research priorities).

Visual interfaces to digital libraries (Börner and Chen, 2002) aim to help users mentally organize, electronically interact with, and manage large, complex information spaces. They apply powerful data analysis and information visualization techniques (Card et al., 1999) to transform data and information that are not inherently spatial into a visual form. The visualizations shift the user's mental load from slow reading to faster perceptual processes such as visual pattern recognition. Frequently, visual interfaces exploit human beings' powerful spatial cognition (Lakoff, 1987) and the method of loci—a mnemonic technique that originated with the ancient Greeks—to associate and attach any digital information, tool, or service to a spatial location or, using an identification tag, to other people. They have been successfully implemented and deployed for the PubMed database (http://pubmed.antarcti.ca/start) by Antarcti.ca System Inc., and also in form of self-organizing maps for astronomy and astrophysics articles (http://simbad.u-strasbg.fr/A+A/map.pl).

Another major shortcoming of today's digital information spaces is the scarcity of social navigation cues (e.g., who is online, what resources are accessed frequently, etc.), making it difficult to find relevant resources and expertise or to collaborate. Research on social visualizations (Freeman, 2000; Donath et al., 1999; Erickson et al., 1999; Börner and Penumarthy, 2003) aims to show data about a person, illuminate relationships among people—even people they do not know—or visualize group activity to facilitate information access, collaboration, and decision-making.

People's "information neighborhood" is not only made up of documents, but also, and perhaps more importantly, by people, including family, friends, neighbors, coworkers, and a shifting network of acquaintances (Haythornthwaite and Wellman, 1998). Having access not only to knowledge, but also to expertise (experts with deep knowledge about a subject matter) seems to be important for human "information foraging" (Sandstrom, 1999; Pirolli and Card, 1999). Research on community knowledge portals aims to help participants capture, access, and manage knowledge and expertise created during their work process, to link community members to each other and to

relevant content, and to offer personalized services tailored to the individuals and communities based on collaborative filtering (Mack et al., 2001).

A system that aims to support scholarly knowledge and expertise management should have an interface that is as easy to use as the Google search interface. It should let users query not only for author names and paper titles, but also for the datasets used and the services (e.g., techniques, approaches, and/or software packages) applied to arrive at the results reported in the papers. In order to enable scholars to track the diffusion of scholarly knowledge via co-authorships and the citation of papers, the tool should also support users in the traversal of co-authorship and citation linkages. Visual interfaces might be employed to give users a global view of a dataset, a co-author network, or a paper-citation network. Usage data should be collected to support social navigation and to estimate the usefulness of data records and author contributions.

11.3 Semantic Association Networks

In *Needed: A National Science Intelligence and Documentation Center,* Eugene Garfield (1955b) wrote: "We may have ivory towers, but they are all connected by cables under the ground or by invisible channels in the air. Scientific progress—the accumulated effect of your individual contributions—depends upon a free flow of information, of thousands of minute facts, of millions of seemingly unrelated observations made and reported by scientists in diverse specialties."

This section describes *Semantic Association Networks* (SANs), a new means to interlink, access, and manage scholarly knowledge and expertise. It explains association-based storage and access using SANs, their potential application for judging data quality and author reputation, and how they may be used to gain a more global view of knowledge and expertise. I start with a review of publication networks and publication-author networks and describe their extension to dataset-service-publication-author networks.

Publication Networks. Today, an increasing number of publishers and some search engines support the traversal of scholarly publications via citation linkages. The citation linkages are either supplied by hand or extracted automatically. CiteSeer (Giles et al., 1998) and the Google scholarly search engine (http://scholar.google.com/) successfully support users in the traversal of automatically extracted citation networks.

Publication-Author Networks. Each publication also reports a set of authors that produced the publication. Similar to the automatic extraction of citation interlinkages, authors can be identified and interconnected via bidirectional "co-authors_with" links; see Figure 11.3. This makes possible the traversal of co-author networks and their associated publication results. Queries such as "Retrieve all authors that collaborated with author x" can be executed.

Publication datasets can also be mined to determine the productivity, reputation, and topical expertise of an author. The number of papers published per time unit can be used as a measure for publication productivity. The quality of publication venues and the number of received citations are an indicator for the author's reputation. Keywords associated with an author's publications or the words appearing in the title, abstract, or full text of these publications can be used to identify topic coverage, topic changes, etc. Queries such as "Retrieve all authors that have many publications in year x" or "Retrieve all authors that have a high reputation in area x" can be executed.

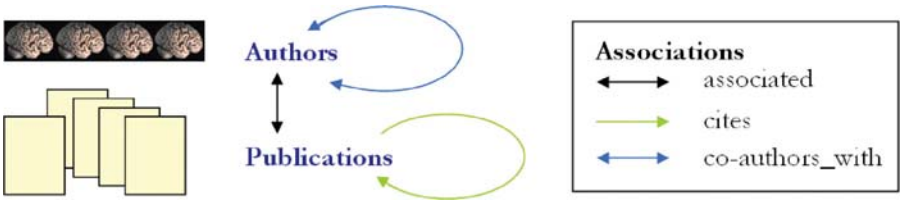


Figure 11.3 Publication-citation and associated co-author networks: "Publications" are interconnected by "cites" links. "Authors" are connected by bidirectional "co-authors_with" links. Each "Publication" has an associated set of "Authors."

Information on co-author relationships can also be used to determine successful co-author teams (Börner et al., 2005) and to answer queries such as "Retrieve the set of authors that is most successful in area y."

Data-Service-Publication-Author Networks. Today, diverse scientific communities create and contribute not only to repositories of scholarly publications, but also to repositories of datasets and services. Although a time- and resource-demanding enterprise, the creation of central databases gives researchers easy access to existing datasets and tools.

For example, the National Library of Medicine provides access to the Online Mendelian Inheritance in Man (OMIM) database, a catalog of human genes and genetic disorders authored and edited by Dr. Victor A. McKusick and his colleagues at Johns Hopkins and elsewhere. OMIM interlinks gene data and publication data based on manually identified correlations. Service repositories facilitate sharing, evaluation, and comparison of algorithms and software, and reduce the time and effort spent on repeatedly reimplementing algorithms. For example, the CERN Library (http://wwwinfo.cern.ch/asd/) provides a large collection of general-purpose programs maintained and offered in both source and object code. Researchers and students are encouraged to use this code repository rather than their own code. Aside from saving users time and effort, the repository code is more likely to be correct after having been tested by many other people.

Plug-in-based software frameworks (Penumarthy et al., submitted) that integrate algorithms written in diverse programming languages, which support a multitude of file formats and offer menu-driven interfaces, appear to serve interdisciplinary user communities well.

If data and services are associated with the publications and authors that report them, then new means of search and association discovery become possible (see Figure 11.4). Queries such as "Retrieve all authors that used service x," "Retrieve all papers that report results from dataset y," and "Retrieve all services that were used with dataset y" can be executed.

The availability of data, service, and publication repositories also eases the replication of results reported in papers, the application of existing services to new data, or the analysis of existing data with new services. Service repositories support the examination of algorithms at the code level, as opposed to the pseudocode level reported in papers.

Users of association networks would contribute publications that also report datasets and services used. If they used new datasets and/or services they would be required to submit those as well (see Figure 11.4). Note that journals such as *Science* and *Nature* already require authors to provide access to datasets and services before

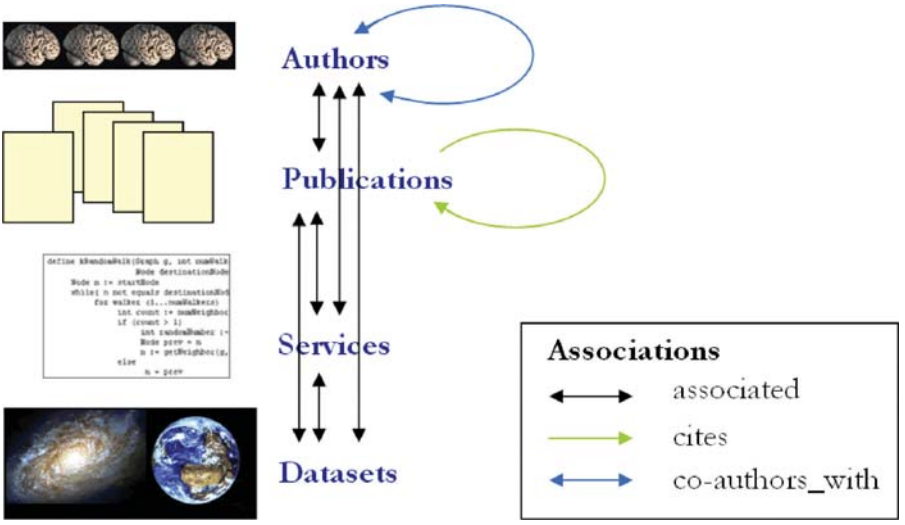


Figure 11.4 "Publications" associated with "Authors," "Services," and "Datasets."

a paper is published. The *National Science Foundation* and the *National Institutes of Health* also promote and financially support the development of datasets and service repositories.

If papers are submitted with links to used datasets and services then information on "Authors" and the "associated" links to "Authors," "Services," and "Datasets" can be derived automatically, as can "cites" and "co-authors_with" relationships.

A major effort will be the identification of identical authors, publications, services, and datasets. Authors may report their names in different ways (e.g., with or without a middle name). They may change their name due to marriage. Publishers might translate and spell foreign names in different ways. Publications might report nearly or completely identical results. Services, for example, clustering algorithms, are developed by diverse communities and most likely are given different names. It is only at the code level and/or input/output level that they can be compared. Identical datasets might exist in different formats and with different names and metainformation.

"Is_identical" associations can be employed to denote if two authors are the same, if two or more publications report the same result, if a set of services has identical input/output behavior, or if a set of data is identical (see Figure 11.5). Identity can be established using automatic means or could be supplied by users of the system.

User activity would be logged and the number of accesses would be used to estimate the value and quality of any of the data records. Resources that are accessed or cited more often or are published in higher-reputation journals might be given higher-quality placement. Higher reputation would be given to authors/donators of high-quality resources.

Visual interfaces should be implemented to provide users of SANs with a global, bird's-eye view of the data that is available at their fingertips. Visualizations would also reveal and help users understand the complex associations among the different data entries. They would help users identify areas of high activity and/or growth, understand what data are accessed most often, understand what data are widely regarded as most correct, etc.

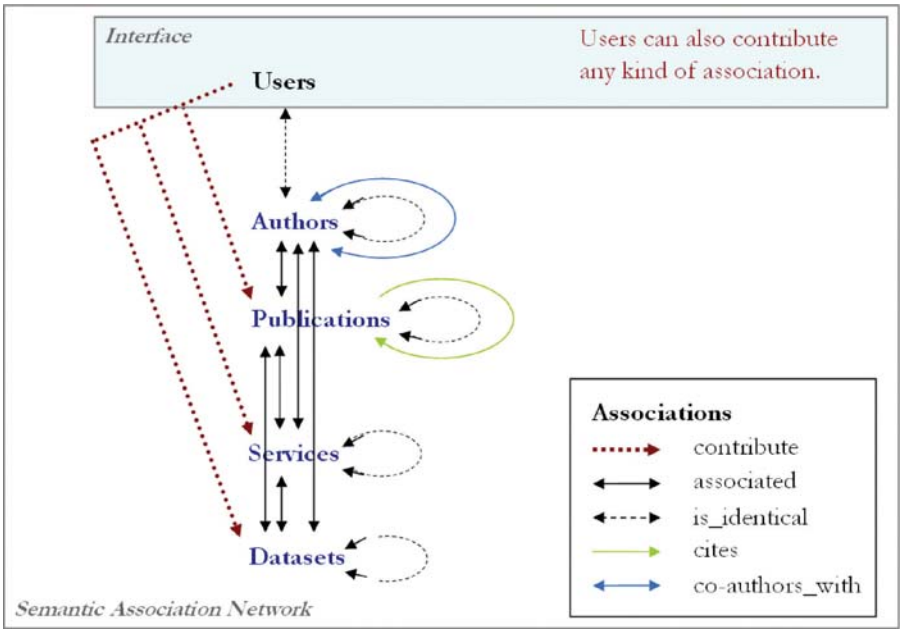


Figure 11.5 *Semantic Association Networks* interconnect "Datasets," "Services," "Publications," "Authors," and "Users" via diverse associations.

Some very promising examples of effective interfaces to publication data are already available. *VisualLink*, developed by Xia Lin, Howard D. White, and Jan Buzydlowski, provides a global view of author association networks by analyzing and visualizing the number of times authors are cited together (Lin et al., 2003). The *HistCite* software developed by Eugene Garfield and colleagues generates global maps of paper-citation networks in support of a local and global examination of highly cited papers and their interlinkages (Garfield, 2004).

Semantic Association Networks in RDF Representation. Semantic Web technology such as the W3C's Resource Description Framework (RDF) (http://www.w3.org/RDF/) can be applied to define and represent *Semantic Association Networks* in a way such that automatic association discovery, retrieval, and management become possible.

RDF is a very simple data model for representing information about online resources. Each resource (any identifiable thing, including things that may not be directly retrievable on the Web) has a *Uniform Resource Identifier* (URI). A URI can have simple properties and property values. To ease adoption, a relatively small subset of RDF will be applied to encode the semantics of SANs. Listing 11.1 shows an exemplary RDF presentation of SANs that is explained subsequently.

The RDF representations of SAN nodes are discussed first. "Author," "Paper," "Service," and "Dataset" nodes in the SAN are defined as subclasses of "Entity" with properties "entity_created_on" and "times_accessed." Hence all their instantiations have an attribute value for "entity_created_on" that indicates their time of creation as well as a value for "times_accessed" denoting how often they have been accessed by a user. Each instantiation of any of the subclasses has a unique ID. Multiple implementations of

one service might have different IDs until their identity is discovered. A class "Association" is defined with properties "association_created_on" and "times_occurences" to specify the time at which SAN links are generated and how often they occur (e.g., how often a pair of authors co-authored). Class "User" has a property "contributed" that provides information on what "User" contributed what "Entity" or "Association."

SAN links such as "associated," "co-author_of," "cites," "is_identical_author," "is_identical_paper," "is_identical_service," "is_identical_dataset" are represented as subclasses of the class "Association." The bidirectional "associated" links exist among subclasses of type "Entity": "Datasets," "Services," "Publications," and "Authors." Only "Papers" can be linked via directed "cites" links and only "Authors" can be linked via bidirectional "co-authors_with" links. Bidirectional "is_identical_*" links exist among subclass instantiations of the same type as well as among "Users" and "Authors." Directed "contributes" relations exist between users and any subclass of "Entity" or "Association."

Note that the proposed RDF metadata would be assigned automatically, not manually. Users would login to a scholarly repository using a unique author identifier. If they access an "Entity" then the "times_accessed" property of this entity would be updated automatically. If they submit an "Entity" or "Association" then a "*_created_on" property would be generated and their user ID would be connected via a "contributes" property to the new entity and/or association. The user never sees any RDF tags, but their availability makes new means of search, interlinkage, and management possible.

Note that the RDF specification given in Listing 11.1 can be easily extended to incorporate other data such as "Comments" or an "evaluation score" "contributed" by a "User" for any "Entity." The former could be implemented as another subclass of "Entity." The latter could be encoded as an additional property of "Entity." It could also be beneficial to have subclasses of class "Publication" denoting that a certain publication is a "Paper," "Patent," or "Grant" or to tag a paper as being a "Journal Paper," "Book Chapter," etc.

Listing 11.1. ***Semantic Association Networks*** **in RDF representation**

```
<!—Define class Entity and all its properties -->
<rdfs:Class rdf:ID="Entity"/>
<rdf:Property rdf:ID="entity_created_on">
     <rdfs:domain rdf:resource="#Entity"/>
</rdf:Property>
<rdf:Property rdf:ID="times_accessed">
     <rdfs:domain rdf:resource="#Entity"/>
</rdf:Property>

<!—Define subclasses of Entity-->
<rdfs:Class rdf:ID="Author">
      <rdfs:subClassOf rdf:resource="#Entity"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Paper">
      <rdfs:subClassOf rdf:resource="#Entity"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Service">
      <rdfs:subClassOf rdf:resource="#Entity"/>
</rdfs:Class>
```

Listing 11.1. Continued

```
<rdfs:Class rdf:ID="Dataset">
      <rdfs:subClassOf rdf:resource="#Entity"/>
</rdfs:Class>

<!—Define class Association and all its properties -->
<rdfs:Class rdf:ID="Association"/>
<rdf:Property rdf:ID="association_created_on">
     <rdfs:domain rdf:resource="#Association"/>
</rdf:Property>
<rdf:Property rdf:ID="times_occurences">
     <rdfs:domain rdf:resource="#Association"/>
</rdf:Property>

<!—Define subclasses of Association-->
<rdf:Property rdf:ID="associated">
<rdfs:subClassOf rdf:resource="#Association"/>
      <rdfs:domain rdf:resource="#Entity"/>
      <rdfs:range rdf:resource="#Entity"/>
</rdf:Property>
<rdf:Property rdf:ID="co-authors_with">
<rdfs:subClassOf rdf:resource="#Association"/>
      <rdfs:domain rdf:resource="#Author"/>
      <rdfs:range rdf:resource="#Author"/>
</rdf:Property>

<rdf:Property rdf:ID="cites">
     <rdfs:subClassOf rdf:resource="#Association"/>
     <rdfs:domain rdf:resource="#Paper"/>
     <rdfs:range rdf:resource="#Paper"/>
</rdf:Property>

<rdf:Property rdf:ID="is_identical_author">
     <rdfs:subClassOf rdf:resource="#Association"/>
     <rdfs:domain rdf:resource="#Author"/>
     <rdfs:range rdf:resource="#Author"/>
<rdfs:range rdf:resource="#User"/>
</rdf:Property>

<!—define rdf:Property "is_identical_paper",
"is_identical_service",
"is_identical_dataset" analogously to rdf:Property
"is_identical_author"-->

<!—Define class User and its property -->
<rdfs:Class rdf:ID="User"/>
<rdf:Property rdf:ID="contributes">
     <rdfs:domain rdf:resource="#User"/>
     <rdfs:range rdf:resource="#Association"/>
     <rdfs:range rdf:resource="#Entity"/>
</rdf:Property>
</rdf:RDF>
```

To implement the proposed *Semantic Association Networks*, scientific communities, institutions, and companies will need to record, make available, and preserve not only scholarly papers, but also datasets and software services. In addition, authors need to be encouraged to report used datasets and applied services in their publications. This will provide the basis for the semiautomatic generation of SANs.

Most likely, a combination of automatic citation indexing (Giles et al., 1998) and association discovery and simple Web forms via which registered users can correct or add links manually will be most successful in acquiring accurate and comprehensive SANs.

Initially, SANs and user registration could be hosted and managed centrally. Over time, SAN registries would need to become decentralized to improve access time and error tolerance. The growth and usage of the semantic association networks should be monitored extensively. Resulting usage data could be employed to optimize the SANs (e.g., to mirror highly accessed resources to improve access time).

11.4 Implementing SANs: Opportunities and Challenges

The implementation of SANs provides a number of potentially very powerful new means to access and correlate scholarly knowledge and expertise. SANs could help to create a scholarly "interoperability substrate" to design value-adding services that are not "vertically" locked. For example, they provide the data needed for the generation of effective visual interfaces to digital libraries or the design of global maps of science that report the structure and evolution of science.

Note that SANs are envisioned to serve all areas of science, interconnecting and hence helping to cross-fertilize interdisciplinary research. Given that everybody with Internet access—including school kids or hobby researchers—would be able to use SANs to gain access to first-rate knowledge and expertise, SANs can be seen as a means of mass education and empowerment.

Widespread availability of SANs would also constitute a great testbed application for diverse scientific communities that aim to develop novel ways to store, preserve, integrate, correlate, access, analyze, map, or interact with data.

Subsequently, I review the diverse challenges that one would face when trying to implement SANs on a global scale.

Social Challenges. As mentioned above, appropriate incentive structures need to be implemented to make authors donate not only papers, but also datasets and services used.

Standards will have to be developed to uniquely, persistently, and globally identify the content of a data record (e.g., papers, authors, datasets, and services). The *Digital Object Identifier* (DOI) system (http://www.doi.org/) might be used and extended to also cover authors, datasets, and services. Another rich source of information on bibliographic, holdings, authority, classification, and community information is the MARC (MAchine-Readable Cataloging) data format (http://www.loc.gov/marc/) that emerged from a Library of Congress–led initiative. MARC records (which became USMARC in the 1980s and MARC 21 in the late 1990s) have been extensively used by most libraries for the last 30 years. The MARC author name authority records might provide a useful nucleus for the creation of a unique author identifier. While these author authorization files identify authors via a fixed name, date of birth, etc., I believe that an author identifier should not contain the name of the author (as it

might change due to marriage, etc.), the date of birth, social security numbers, or passport numbers. A simple running number will suffice.

Other social challenges relate to data protection, privacy concerns, legitimacy via content contribution and evaluation by distributed subject and professional teams, and sustainable resource models. The scale-free topology of scientific networks poses serious challenges with regard to multilingualism, preservation of diverse traditions, views, and approaches as only a minority of sources and experts is highly visible and accessible while the vast majority is too weakly connected to be seen.

Technological Challenges. If scholars can be persuaded to provide access to datasets and services and to report the usage of datasets and services in the papers they publish, then automatic means can be employed to associate data/services, papers, and their authors by means of semantic association networks. Major technical challenges comprise the federation of data from different databases, dealing with all kinds of data quality issues (e.g., multiple formats, misspellings, omissions, etc.), planning for sustainability, robustness, support of heterogeneous hardware, and the preservation of datasets and services as technology and data formats evolve.

I do not foresee a shortage in terms of computing power, data storage, or bandwidth. Computer power is doubling almost every year (Djorgovski, 2004) and theoretically, this trend can continue for 600 more years (Krauss and Starkman). The performance/price of disk-based data storage is improving even faster than the performance/price of computing power, at a factor of about 100 over the last 10 years (Hayes, 2002). Bandwidth is increasing at a much slower pace. Global efforts such as the TeraGrid (http://www.teragrid.org/)—a multiyear effort to build and deploy the world's largest, most comprehensive, distributed infrastructure for open scientific research—will soon interconnect major databases and services at high bandwidth. However, the cost-effective interconnection of customer sites with comparatively low amounts of traffic to major bandwidth hubs, also called the "last mile problem," is unresolved.

Given the computing power, data storage capacity, and bandwidth available today, it is surprising to see how old-fashioned our current means of accessing and managing scholarly knowledge and expertise are.

Funding Acquisition Challenges. The development of standards and the implementation of SANs will require funding. The effort to implement SANs might be best compared with the implementation of the diverse Citation Indexes provided by the Institute of Scientific Information (ISI) or the implementation of Google. However, the dollar amounts spent on those two projects are not available to us.

Instead, I report the effort spent on Douglas Lenat's Cyc project (http://www.cyc.com/), most likely the world's largest common-sense knowledge base in existence today. The Cyc project was funded over 20 years with 25 million artificial intelligence research dollars. It is a 600-person-per-year effort that assembled a knowledge base containing 3 million rules of thumb that the average person knows about the world, plus about 300,000 terms or concepts nodes (a typical person is assumed to have about 100 million concepts).

The proposed SANs differ from Cyc in that they do not require the careful manual compilation and encoding of logical rules that interconnect concept nodes. Instead, a semiautomatic process is proposed to cross-correlate existing data, services, publication, and author databases. The interlinkage of all scholarly knowledge might very well exceed the resources spent to create Cyc but it would benefit all of science.

Where to Start? Physics and astronomy are two scholarly communities that keep particularly good track of their data, services, and publications. International efforts

such as the International Virtual Observatory aim to develop standards for data and interfaces as well as for software packages, source code libraries, and development tools (Williams et al.). The American Physics Society has *Physical Review Letters* publication data available in full text from 1893 until today (OCR'ed full text exists for everything from about 1995 back). Physicists and astronomers use the arxiv.org e-print archive extensively for timely scholarly publication. The physics and astronomy domains would be good candidates for implementing prototype SANs.

11.5 Concluding Remarks

Eugene Garfield, when proposing a *Unified Index to Science* wrote: "Grandiose schemes always meet with excessive resistance, not because they are impossible to achieve, but because there are only a few with sufficient persistence to materialize their dreams and even fewer to carry them out. Ultimately, most large endeavors must fall by the wayside, to be replaced by others. However, their value at a particular stage of history cannot be disputed" (1959, p. 468).

This chapter described *Semantic Association Networks*, a new means to tag, interconnect, and manage scholarly datasets, services, results, and expertise as a means to keep better records of our collective scientific knowledge, expertise, and progress.

I conclude this chapter with a vision for the future of scholarly publishing that might supersede today's writing, peer-review, and publication of papers. This process becomes unmanageable as the flood of information is increasing and there are no automatic means to extract and compile knowledge summaries for paper collections.

Somewhere in the not-too-distant future, reporting a scholarly result might not involve writing a paper. Instead, scholars may add a "knowledge nugget node" or an "association link" to a complex semantic association network of humanity's knowledge; some Bloggers are already practicing this today. The nodes in this network will describe tangible objects (e.g., a pottery piece found at a certain place by an archeologist together with information about its origin and intermediate positions/usages up to today) or intangible objects (e.g., a formula). Links will represent associations of diverse types (e.g., causal ones such as "Ozone is created electrically in nature during active thunderstorms"). Each node and each link would have information on who added, modified, or deleted it. A scholar's reputation would depend on the number of nodes and/or links she or he contributed and their usefulness to humankind.

Scholars would navigate, mine, and add to this vast network of human knowledge using the information associated with nodes and links and data about the usage of nodes and links. If a scholar adds a new "knowledge nugget" or "association link" the network would be recompiled with the new node leading to

- The identification of redundant data/algorithm/results,
- A confirmation of existing facts improving the correctness of human knowledge,
- A confirmation of the novelty of a fact increasing human knowledge, or
- A conflict with existing facts.

In the latter case, either the new nugget or link is wrong, or other nodes/links are wrong, alternative interpretations/views are acceptable, or the entire conceptualization is not working and a scientific revolution in Kuhn's sense (1962) is needed.

11.6 Acknowledgments

I would like to thank Kevin Boyack, Joe Futrelle, Stacy Kowalczyk, Deborah MacPherson, Ketan K. Mane, Shashikant Penumarthy, Bonnie DeVarco, and Elijah Wright for insightful comments on a draft of this paper. Shashikant Penumarthy inspired and collaborated on the representation of SANs in RDF format. Eugene Garfield, Peter A. Hook, and Mark Meiss provided helpful pointers. This work is supported by National Foundation CAREER Grant IIS-0238261 and National Science Foundation Grant DUE-0333623.

11.7 References

Borgman, C. (2000). Scholarly communication and bibliometrics revisited. In: Cronin, B., Atkins, H.B. (Eds.), *The Web of Knowledge*. Medford, NJ: Information Today (pp. 143–162).

Börner, K., Chen, C. (2002). Visual interfaces to digital libraries. In: *LNCS*: Springer Verlag.

Börner, K., Chen, C., Boyack, K. (2003). Visualizing knowledge domains. In: Cronin, B. (Ed.), *Annual Review of Information Science & Technology*. Medford, NJ: Information Today, Inc./American Society for Information Science and Technology (pp. 179–255).

Börner, K., Dall'Asta, L., Ke, W., Vespignani, A. (2005). Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams. *Complexity*, 10(4):58–67.

Börner, K., Penumarthy, S. (2003). Social diffusion patterns in three-dimensional virtual worlds. *Information Visualization*, 2:182–198.

Bush, V. (1945). As we may think. *The Atlantic Monthly*, 176:101–108.

Card, S., Mackinlay, J., Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann.

Djorgovski, S.G. (2004). *Virtual Observatory, Cyber Science, and the Rebirth of Libraries*. Available: http://www.arl.org/forum04/djorgovski.html, Accessed 11/16/2004.

Donath, J.S., Karahalios, K., Viegas, F. (1999). Visualizing conversation. *Journal of Computer Mediated Communication*, 4.

Engelbart, D.C. (1963). A conceptual framework for the augmentation of man's intellect. In: Howerton, P.W., Weeks, D.C. (Eds.), *Vistas in Information Handling*. Washington D.C.: Spartan Books (pp. 1–29).

Erickson, T., Smith, D.N., Kellogg, W.A., Laff, M., Richards, J.T., Bradner, E. (1999). Socially translucent systems: Social proxies, persistent conversation, and the design of "Babble." *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems: The CHI Is the Limit*, ACM Press, pp. 72–79.

Freeman, L.C. (2000). Visualizing social networks. *Journal of Social Structure*, 1.

Garfield, E. (1955a). Citation indexes for science: A new dimension in documentation through association of ideas. *Science*, 122:108–111.

Garfield, E. (1955b). Needed: A National Science Intelligence and Documentation Center. *Symposium on Storage and Retrieval of Scientific Information of the Annual Meeting of the American Association for the Advancement of Science,* Atlanta, Georgia. Available: http://www.garfield.library.upenn.edu/papers/natlsciinteldoccenter.html, accessed 10/18/2004.

Garfield, E. (1959). A unified index to science. *International Conference on Scientific Information*, Washington, D.C., National Academy of Sciences, pp. 461–474. Available: http://books.nap.edu/html/sci_inf_1959/461-474.pdf, accessed 10/18/2004.

Garfield, E. (2004). Historiographic mapping of knowledge domains literature. *Journal of Information Science*, 30:119–145.

Giles, C.L., Bollacker, K., Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In: Witten, I., R.A., Shipman, F.M. III (Eds.), *Digital Libraries 98: The Third ACM Conference on Digital Libraries,* Pittsburgh, ACM Press, pp. 89–98.

Hayes, B. (2002). Terabyte territory. *American Scientist*, 90:212–216.

Haythornthwaite, C., Wellman, B. (1998). Work, friendship, and media use for information exchange in a networked organization. *Journal of the American Society for Information Science*, 49:1101–1114.

Henry, G. (2003). On-line publishing in the 21st Century: Challenges and Opportunities. *D-Lib Magazine*, 9.

Krauss, L.M., Starkman, G.D. Universal limits on computation. *arxiv: astro-ph/0404510*.

Kuhn, T.S. (1962). *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press.

Lin, X., White, H.D., Buzydlowski, J. (2003). Real-time author co-cocitation mapping for online searching. *Information Processing and Management*, 39:689–706.
Lynch, C. (2003). Institutional repositories: Essential infrastructure for scholarship in the digital age. *ARL Bimonthly Report 226.* Available: http://www.arl.org/newsltr/226/ir.html, accessed 10/19/2004.
Mack, R., Ravin, Y., Byrd, R.J. (2001). Knowledge portals and the emerging digital knowledge workplace. *IBM Systems Journal*, 40:925–955.
Penumarthy, S., Börner, K., Herr, B. (Submitted). Information visualization cyberinfrastructure software framework. *Information Visualization.*
Pirolli, P., Card, S. (1999). Information foraging. *Psychological Review*, 106:643–675.
Price, D.J.D. (1965). Networks of scientific papers. *Science*, 149:510–515.
Sandstrom, P.E. (1999). Scholars as subsistence foragers. *Bulletin of the American Society for Information Science*, 25:17–20. Available: http://www.asis.org/Bulletin/Feb-99/sandstrom.html, accessed 10/18/2004.
Scholtz, J. (2000). DARPA/ITO Information Management Program Background.
Shiffrin, R. (2004). *Scientists Seek "Map of Science."* BBC News. Available: http://news.bbc.co.uk/1/hi/sci/tech/3608385.stm, accessed 10/24/2004.
Shiffrin, R., Börner, K. (2004). Mapping knowledge domains. *Proceedings of the National Academy of Sciences (Suppl. 1),* vol. 101.
Smith, M., Bass, M., McClellan, G., Tansley, R., Barton, M., Branschofsky, M., Stuve, D., Walker, J. (2003). DSpace: An open source dynamic digital repository. *D-Lib Magazine*, 9.
Van de Sompel, H., Payette, S., Erickson, J., Lagoze, C., Warner, S. (2004). Rethinking scholarly communication: Building the system that scholars deserve. *D-Lib Magazine*, 10.
Wells, H.G. (1938). *World Brain.* Garden City, NY: Doubleday, Doran.
Williams, R., Moore, R., Hanisch, R.A. (2003). *Virtual Observatory Vision Based on Publishing and Virtual Data.* Available: http://us-vo.org/pubs/files/VO-vision.pdf, accessed 11/10/2004.

Chapter 12

Interactive Interfaces for Mapping E-Commerce Ontologies

Vladimir Geroimenko and Larissa Geroimenko

12.1 XML-Based Communication between Companies: Visualizing a Mutual Understanding

XML is a metalanguage that allows a company of any size to create its own markup languages that describe its business domain in a way that is most suitable and convenient for the company. This is one of the main advantages of using XML: it can provide a company with its own domain-specific language, a highly specialized set of tags (metadata labels) also called a vocabulary. The problem arises when two companies start to communicate with each other using their own XML-based languages. Because the languages use vocabularies that are partly or even completely different, it is impossible for computers to translate automatically from one language into another. One of the aims of the Semantic Web is to enable such translation requiring no human intervention. But before the new generation of the Web reaches that stage of maturity, there is a need for some simple but effective techniques to establish XML-based communication between companies.

To solve the problem, the cooperation of at least two people is required. One should be an expert in the business domain, able to say for sure that, for example, `<sale_price>` for Company 1 means the same as `<our_price>` for Company 2. The other has to be a programmer who can create appropriate XSLT (Extensible Stylesheet Languages Transformations) stylesheets that will translate XML documents of Company 1 into those of Company 2 (and back if needed). This job can be partly automated. Generally speaking, a domain expert does not really need to involve a programmer if he or she is able to manipulate XML metadata labels directly in order to show what metadata have the same meaning. On this basis, a specialist program can write an appropriate stylesheet automatically. It is not even particularly difficult, because an XSLT stylesheet is nothing other than a specialist XML document.

A domain expert cannot (and does not need to) work with the complex codes of the XSLT language. All that he or she needs is clear and comprehensive visualizations of two different vocabularies in order to match them. Usually, a list of words used as metadata labels (i.e., a vocabulary) is not enough to model a business domain. A more advanced conceptual model has to include explicitly all relationships between the entries of a vocabulary. Such a model is called an ontology. In other words, an ontology is an explicit specification of a conceptualization (Gruber, 1993). Ontologies are a fundamental technology for implementing the Semantic Web. They are important because they make it possible to establish a common conceptual description and a joint

terminology between members of communities of interest (human or autonomous software agents).

This chapter explores the possibility of developing interactive visually rich interfaces to enable Web users (especially domain experts) to access and manipulate XML metadata and underlying ontologies. Native visualizations, that is, those that are integral parts of the process of creating and displaying XML documents, are analyzed in order to utilize their potential in our interface design. By adapting and merging various native visualizations, a novel and efficient representation of the XML document ontology has been developed—the Generalized Document Object Model Tree (G-DOM-Tree) Interface. This interface represents XML metadata and their structure in a comprehensive, intelligible and compact way that is suitable for the Web-based manipulation of its elements. After reviewing the main existing technologies for implementing the G-DOM-Tree Interface, a working prototype of an XML Ontology Translator is presented. This application dynamically generates ontological models of two XML documents in the G-DOM-Tree form, and then allows Web users to map them in order to establish XSLT-based communications between XML dialects that are syntactically dissimilar but describe the same domain. This demonstrates that visual interaction with XML ontologies can provide a simple but effective way of dealing with the problem of semantic interoperability of XML documents within real-life Web-based applications.

Due to the novelty of the field, significant results to date are restricted to those described in a few recent publications (Yu, Beyls, and D'Hollander, 2004; Korhonen and Salminen, 2003; Baeza-Yates, Lemus, Ponceleon, and Srinivasan, 2003; Berder, Klein, Disch, and Ebert, 2000; Bonfire, Casadei, and Salomoni, 2000; etc.). At the same time, there is an enormous arsenal of previously developed techniques and metaphors for information visualization (Card, MacKinlay, and Shneiderman, 1999; Chen, 2004; Ware, 2004). Many of them may be adopted creatively for an effective use on the XML-based Web.

12.2 The Process of Creating and Reading XML Documents and Its Native Visualizations

The process of designing and parsing (reading) an XML document necessarily includes several visual representations of the document content and structure. We will call them the native visualizations of XML documents. Although they are currently being used exclusively for design and programming purposes, our hypothesis is that they might also be helpful within visual interfaces for XML documents. Of course, they will probably need to be transformed and adapted in order to do their new job efficiently.

To analyze and compare the main native visualizations, we will take the process of creating a simple Product Catalogue as an example. This process starts with choosing words and phrases that are able to describe all the required aspects of the Product Catalogue content, and which can therefore be used as tag names (i.e., a vocabulary). The next step is to show relationships between vocabulary entries, or in other words, to create an ontology of the Product Catalogue domain. A common way of doing this is by using UML (Unified Markup Language), its diagrams, and notations.

The UML-based ontology shown in Figure 12.1 is the initial native visualization of the content of the XML documents to be created. Such diagrams are much more human-oriented than any other native visualization. They are thorough conceptual

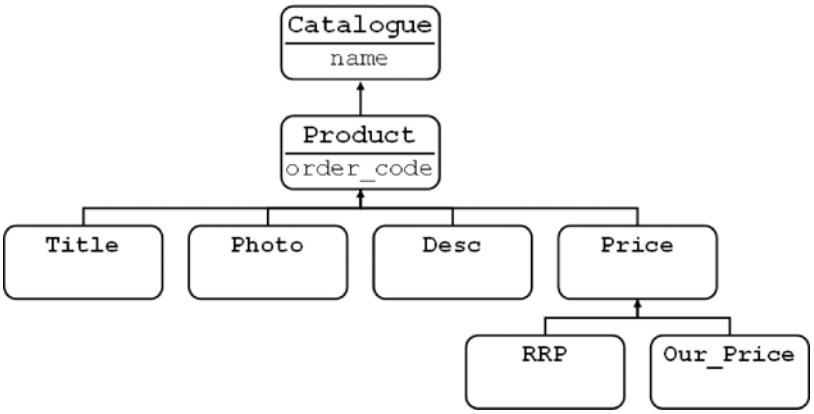


Figure 12.1 The UML-based visualization of the Product Catalogue ontology.

models that capture a human view of the domain, its objects (elements), their properties (attributes), and relationships. Therefore, they have a very high heuristic potential for user interfaces that enable visual interaction with XML ontologies.

Figure 12.2 shows the XML schema of the Product Catalogue, presented in a visual, diagrammatic form. It looks very similar to the above ontological model. In principle, a UML model can be converted into an XML schema automatically, although in the case of complex XML documents it may be difficult to achieve without human intellectual help. In comparison to a UML ontology, an XML schema contains some additional technical components that are necessary for computers but are not so important for human understanding of the domain structure.

Once an XML schema is constructed, it can be used as a template for creating an unlimited number of XML documents. The design process is finished, and the next stage of reading and displaying the created documents lies ahead. The first step in this stage is parsing (reading) an XML file using a special program called the XML parser. It can be a standalone parser or an integral part of a Web browser. A parser reads an XML file and constructs a hierarchical tree of nodes in the computer memory (called the XML Document Object Model or the XML DOM). The DOM of our Product Catalogue created by a Java-based parser is shown in Figure 12.3. Every structural component of an XML document (elements, data, etc.) is represented within its DOM as a node. Basically, a proper XML DOM contains nothing but nodes. The DOM

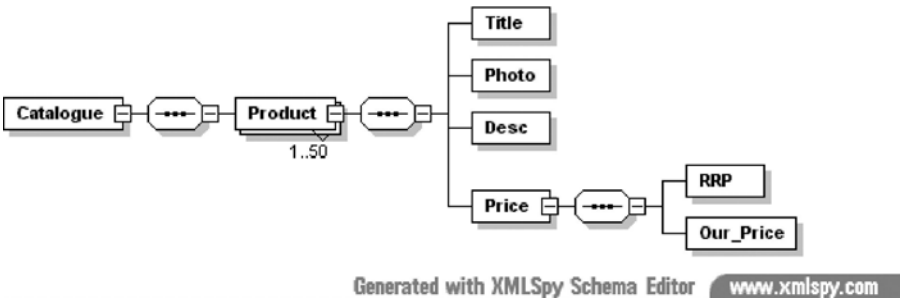


Figure 12.2 The XML Schema visualization of the Product Catalogue ontology.

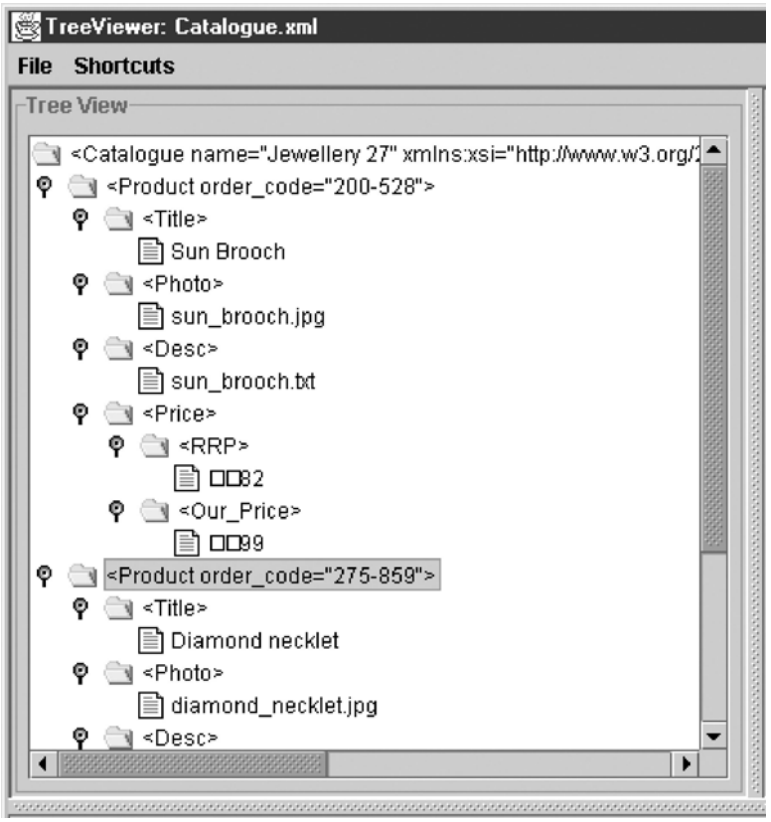


Figure 12.3 The DOM (Document Object Model) visualization of a Catalogue document.

visualization looks quite technical from the user's point of view. Since it contains the maximal number of structural components (nodes) that may be extracted from an XML document, it is very difficult for the user to traverse the node tree of a complex XML document. Nevertheless, as will be shown later, it is possible to use a DOM tree as a navigational aid within a graphical user interface.

Thus, the process of designing and parsing XML documents involves at least three native visualizations. They have more similarities than differences. These visualizations may be of use for creating visual interfaces that facilitate user interaction with XML metadata.

12.3 Technologies for Visualizing XML Documents

If the visualizations that are native to the XML development process can be useful in designing interfaces for XML applications, what about technologies and languages available for the implementation of visually rich interfaces? First it should be noted that an XML document as such has no specific visual form other than the plain text view. Indeed, an XML file (unlike an HTML file) does not contain any formatting tags and therefore requires specialist external files, or even applications, for its visual rendering.

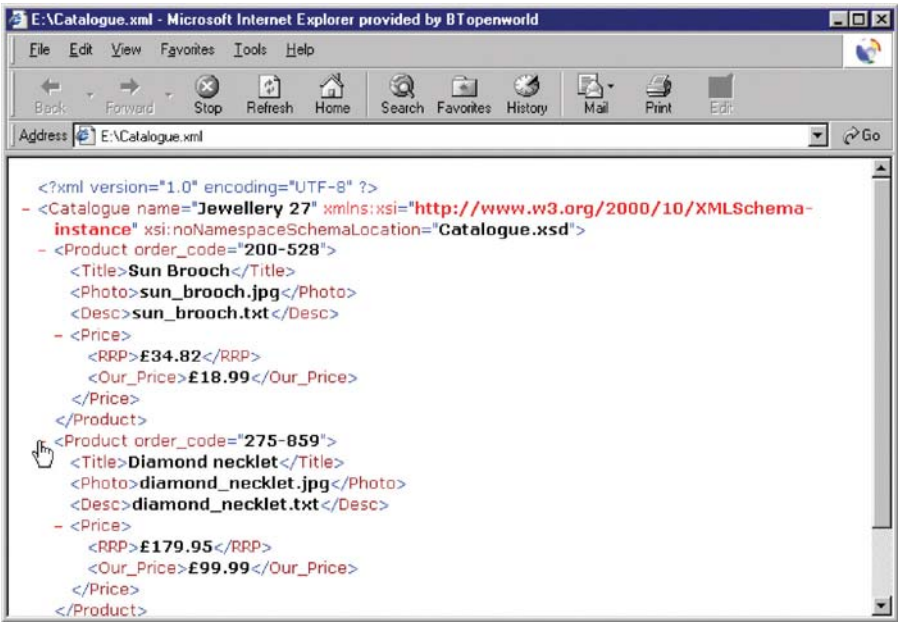


Figure 12.4 The rendering of a Catalogue document in Internet Explorer.

Figure 12.4 shows how the "Catalogue.xml" file appears in Microsoft Internet Explorer. The main difference between this view and a plain text view (for example, in Notepad) is that Internet Explorer allows collapsing and expansion of XML elements that have child nodes.

It is possible to format an XML document in order to display it in a browser as a normal HTML page that includes links, tables, a variety of fonts, colors, etc. Cascading Style Sheets (CSSs) are suitable for simple formatting, whereas the Extensible Stylesheet Language (XSL) allows more complex formatting and also transformation of XML documents. Although displaying XML as HTML or XHTML documents seems to be the main technique of their presentation on the Web (at least, in the foreseeable future), it does not provide the level of interactivity required for creating effective visual interfaces. It lacks direct manipulation features, which are hard to implement even using Dynamic HTML (DHTML).

If the HTML/XHTML rendering has a distinctly limited potential for contributing to visually rich interactive interfaces, what languages and techniques are more suitable? In principle, any programming language may be equally relevant, as XML is language independent. Although this is true in theory, the real-world situation is much more complicated. For various reasons, only a few languages will be of wide use on the XML-based Internet. An analysis of today's situation and trends reveals the two major players—Java and ActionScript.

It is no surprise that Java is the closest companion to XML. Like XML itself, Java is platform independent. This means that a combination of XML documents and Java applications or applets provides e-commerce developers with a universal and portable solution. Combining XML with any other programming languages (such as C or C++) may result in losing its cross-platform features. Another reason for using Java for developing XML applications is that Java has become the *de facto* number-one programming language, and so it is a natural choice for most programmers.

Sun Microsystems, Inc. (http://java.sun.com) provides developers with the following core Java technologies for XML: the Java API for XML Parsing (JAXP), the Java API for XML Messaging (JAXM), and the Java API for XML Data Binding (JAXB). Together, these three key Java technologies provide developers with a powerful and easy-to-use tool set. JAXP allows them to integrate any XML parser with a Java application in order to read, manipulate, and generate XML documents. JAXM enables B2B messaging through support of a variety of XML messaging methods, including the ebXML standard. JAXB offers two-way mapping between Java objects and XML documents, including the automatic generation of Java classes from XML schemas. With the delivery of JAXP, JAXM, and JAXB, Java increases its potential as the main programming language for XML development by offering even more simplified and integrated solutions.

While Java is the obvious choice for XML developers, the ActionScript programming language is hardly known to most of them at this moment. What is ActionScript and why can it be cited, along with Java, as one of the most important tools for the future of XML development?

ActionScript is a complete JavaScript-like language that allows Macromedia Flash developers to read, transform, and generate XML documents. The methods of the ActionScript XML object can be used for downloading XML data from a server, accessing and manipulating XML nodes, sending XML documents to a server, and displaying them within a Flash interface. Additionally, ActionScript provides a predefined XMLSocket object for establishing a continuous connection with a server.

Generally speaking, ActionScript offers the same opportunities for XML development as any other programming language. What makes it unique is that it is based on the Flash paradigm. Macromedia Flash is one of the most popular Web technologies. It allows designers to create reliable, dynamic, and graphically rich Web sites for entertainment and e-commerce. According to Macromedia, 7 of the top 10 visited sites on the Internet deploy Macromedia Flash authored content, and 98% of current online users are able to experience this engaging content immediately using a preinstalled Macromedia Flash Player (www.macromedia.com/flash). Although Flash began its life a simple vector-graphics and animation tool, it has evolved into a new platform for developing entire Web sites that is presently used by over half a million Web authors worldwide.

Thus, the Java and ActionScript languages seem to be the two main contenders for XML development. They can complement each other very well: Java is a choice for complex and serious B2B applications; Flash ActionScript is good for graphically rich interactive Web pages, portals, and standalone e-commerce applications. Although Java applets and Flash movies have the same mission as interfaces for XML-based applications, Flash technology offers a simpler and more engaging solution. The history of the Internet (e.g., Java applets versus GIF animations) shows that such solutions have a greater chance of survival.

Thus, Macromedia Flash technology is at present the best choice for creating visually rich and highly interactive interfaces because multimedia, animation, and interaction is the true nature of this Web leading technology. Since adding an embedded XML parser and the ActionScript language with its specialist XML objects, Macromedia Flash has become the most powerful tool for developing real-life XML-based Web applications. For example, Figure 12.5 shows an application we created using Flash and its ActionScript. This application can read our example Product Catalogue XML file and display it in an attractive and interactive visual form that enables the user to browse the Catalogue.

Figure 12.5 The rendering of a Catalogue document using Flash.

12.4 A Generalized Interface for Visualizing XML Metadata and Their Structural Relationships

There are several possibilities for designing graphical user interfaces to XML documents based on existing or novel visualization techniques. We have tried to develop a new interface that enables effective visual interaction with XML metadata. Our research was based on the above analysis of the native XML visualizations as well as some further logical and experimental exploration of the possibilities of their mutual use and integration. The result of the investigation was embodied in a novel interface that we call the Generalized Document Object Model Tree Interface (G-DOM-Tree Interface).

This Interface is based on the XML DOM visualization, generalized and enhanced to make it more suitable for effective visual interaction with metadata. Basically, it is a part of a DOM tree where all logical and structural components of an XML document are represented as nodes. The main difference from a DOM tree is that our model borrows from a standard DOM only its recurring metadata structure. By implementing this, it becomes very similar to an XML schema or a UML model. But, at the same time, the G-DOM-Tree model inherits the structural organization of a DOM rather than of a schema or an UML diagram.

Figure 12.6 shows the G-DOM-Tree representation of the Product Catalogue. Both types of metadata (elements and attributes) are represented as nodes that show element/attribute names and their places within the XML document structure. Each

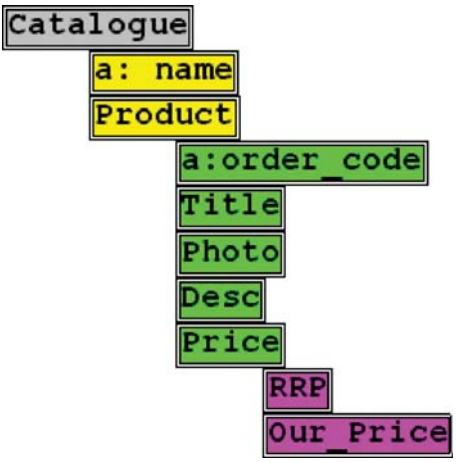


Figure 12.6 The generalized DOM tree model of the Product Catalogue.

attribute has the status of a special child element of its parent element. The letter and colon "a:" in front of a node name is the only difference between an attribute and an element. To make the organization of the node tree more apparent, each structural level is indented and also marked with a different color.

The features of the G-DOM-Tree view that make it an efficient and promising visualization of XML metadata include:

- The ontology of an XML document is represented in a comprehensive form, which, at the same time, is much more compact than a UML model or an XML schema.
- It is easy to read and understand because of its indentation-based form that matches the recurring structural pattern of an XML document and also because of the use of the document tag names.
- All attributes are displayed as specialist children elements marked with the "a:" prefix, making it easy to transform an element into an attribute and back simply by adding or deleting the attribute prefix "a:".
- From a technological point of view, this visualization can be easily implemented as a Flash movie or Java applet, essentially by exploiting their techniques for generating dynamic menus.

Thus, the Generalized DOM Tree is a novel technique for visualizing ontologies of XML documents on the fly, within Web-based user interfaces. Its practical use in a variety of XML applications may therefore be expected.

12.5 A Web-Based Ontology Translator for E-Commerce Documents

To implement the concept of the G-DOM-Tree-based visualization, a working prototype of a simple Ontology Translator has been developed using Macromedia Flash technology. This application can read XML files and generate dynamically their Generalized DOM Trees. Any interface object can be manipulated directly. By dragging and dropping an object (i.e., a framed element or attribute name) on a specific target (another interface object), the user triggers the underlying ActionScript that can

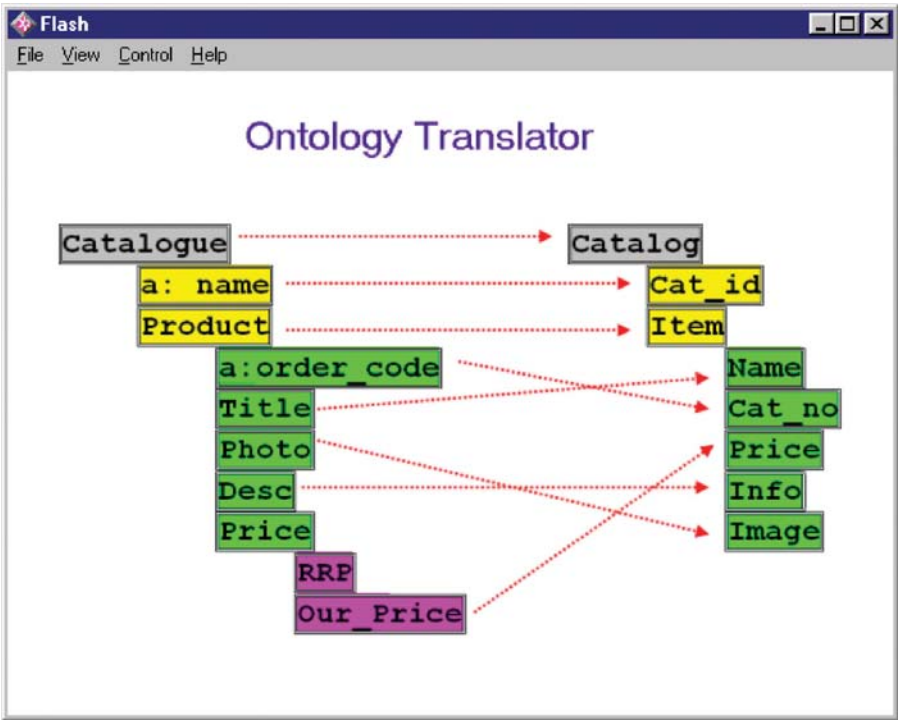


Figure 12.7 Matching ontologies using direct manipulation within an interactive Flash interface.

rename, add, and delete appropriate nodes in the XML documents directly or write an XSLT stylesheet for all documents of this type.

The following example, shown in Figure 12.7, illustrates and explains the use of the Ontology Translator.

Suppose there are two collections of XML documents on the Web that describe the same domain using different XML dialects. In other words, they are employing two dissimilar sets of tags, most or all of them synonymous to human domain experts, but not to computers.

Comparison of the ontologies of the two Product Catalogues reveals the following main differences and similarities:

- The root element names ("Catalogue" – "Catalog") are written in British English and American English respectively.
- The metadata tag "Product" has the same meaning in the first catalogue as the "Item" in the second one.
- The same synonymy exists between the following pairs of tags: "Our_Price"—"Price," "Desc"—"Info," "Photo"—"Image."
- The nodes "a: name"—"Cat_id" and "a: order_code"—"Cat_no" are synonyms, but they have different status (an attribute versus an element in each pair respectively).

Now imagine that you need to establish Web-based real-time communication between the two XML-based languages, for example, in order to merge the "British" and "American" catalogues. It is quite obvious that as of today there is no software like

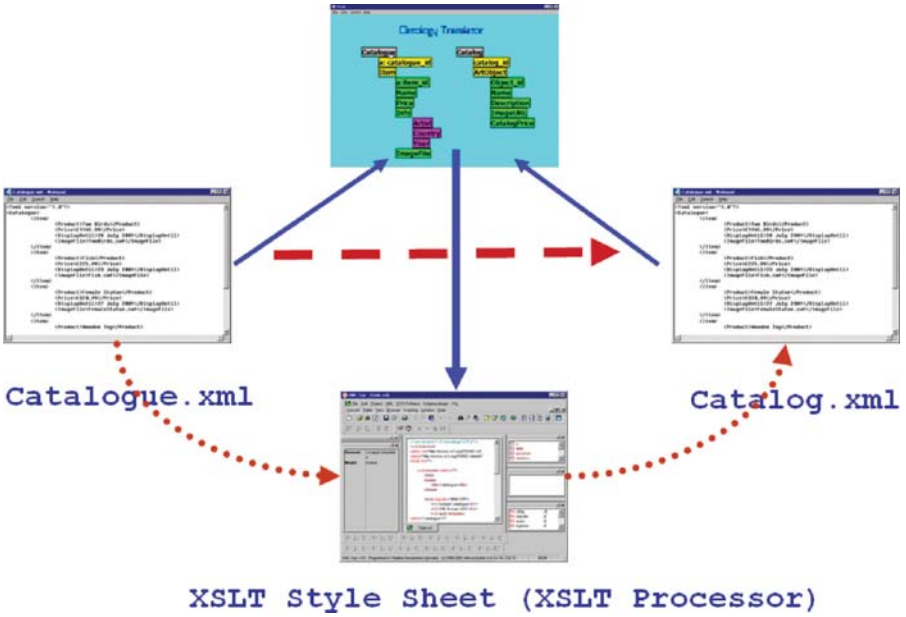


Figure 12.8 The mechanism of communication between the two types of catalogues that is based on an XSLT stylesheet generated as a result of visual interaction with the ontologies.

"intelligent agents" that could accomplish this task for you. A straightforward way is to develop an XSLT stylesheet for these documents but it requires time and technical knowledge.

The Ontology Translator allows any person, expert in the domain, but without XML or XSLT knowledge, to map the two ontologies easily and efficiently. For example, dragging the "Catalogue" element and dropping it onto the "Catalog" can rewrite the name of the root tag in the "American" Product Catalogue or, more generally, establish an XSLT rule for translating one element into another. Deleting or adding the "a:" changes the node status in terms of "element – attribute" not only on the interface but also in the XML document itself. Figure 12.8 shows the mechanism of establishing a mutual understanding by mapping the two ontologies using direct manipulation of their metadata elements. As a result, an XSLT stylesheet is generated that enables an automated translation from "British" Product Catalogues into "American" ones.

Thus, the Ontology Translator is a prototype application that enables visual interaction with ontologies of two XML-based domain-specific languages in order to establish communication between them. This Web application implements both the G-DOM-Tree Interface and the Flash XML technology. It allows domain experts to work effectively in order to solve real-life problems of interoperability of Web-based XML documents.

12.6 Future Work

Our research to date has focused on the possibility of developing interactive visually rich interfaces that enable Web users to access and manipulate XML document

ontologies. Native XML visualizations were analyzed and, on the basis of their integration, a novel and efficient interface for displaying XML document ontology (the Generalized DOM Tree) has been developed. It has been implemented in an Ontology Translator working prototype using Macromedia Flash XML technology. Our research and prototype implementation has shown that visual interaction with XML ontologies provides a comprehensible but efficient way of dealing with the problem of semantic interoperability of XML documents within Web-based applications. The next logical step in the development of G-DOM-Tree-based interfaces is to use them within some real-life e-commerce applications to reveal aspects that need to be enhanced or rethought. A Java version of the interface may be required for its integration with Java-based B2B applications.

12.7 References

Baeza-Yates, R., Lemus, R., Ponceleon, D., Srinivasan, S. (2003). WISDNA: An information visualization paradigm for XML. *First Latin American Web Congress (LA-WEB'03),* November 10–12, 2003, Santiago, Chile, p. 205.

Berder, M., Klein, R., Disch, A., Ebert, A. (2000). A functional framework for Web-based information visualisation systems. *IEEE Transactions on Visualization and Computer Graphics,* 6(1):8–23.

Bonfire, M.E., Casadei, G., Salomoni, P. (2000). Adaptive intelligent hypermedia using XML. *Proceedings of the 2000 ACM Symposium on Applied Computing 2000,* vol. 2, pp. 922–926.

Card, S., MacKinlay, J., Shneiderman, B. (Eds.) (1999). *Readings in Information Visualisation: Using Vision to Think.* Morgan Kaufmann.

Chen, C. (2004). *Information Visualisation* (2nd Edition). Springer Verlag.

Gruber, T.R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition,* 5(2):199–200.

Korhonen, R., Salminen, (2003). A visualization of EDI messages: Facing the problems in the use of XML. *Proceedings of the 5th International Conference on Electronic Commerce Table of Contents.* ACM Press, NY, pp. 465–472.

Ware, C. (2004). *Information Visualisation: Perception for Design* (2nd Edition). Morgan Kaufmann.

Yu, Y., Beyls, K., D'Hollander, E. (2004). Performance visualizations using XML representations. *Information Visualisation, Eighth International Conference (IV'04).* IEEE Computer Society: Los Alamitos, CA, pp. 795–800.

Chapter 13

Back Pain Data Collection Using Scalable Vector Graphics and Geographical Information Systems

Gheorghita Ghinea, Tacha Serif, David Gill, and Andrew O. Frank

13.1 Introduction

According to a Department of Health survey, in Britain back pain affects 40% of the adult population, 5% of which have to take time off to recover (Boucher, 1999). This causes a large strain on the health system, with some 40% of back pain sufferers consulting a GP for help and 10% seeking alternative medicine therapy (Boucher, 1999). Due to the large number of people affected, back pain alone cost industry £9090 million in 1997/8 (Frank and De Souza, 2000), with between 90 and 100 million days of sickness and invalidity benefit paid out per year for back pain complaints (Frank and De Souza, 2000; Main, 1983; Papageorgiou et al., 1995). Back pain is not confined to the UK alone, but is a worldwide problem: in the United States, for instance, 19% of all workers' compensation claims are made with regard to back pain. Although this is a lot less than the percentage of people affected by back pain in the UK, it should be noted that in the United States not all workers are covered by insurance and not all workers will make a claim for back pain (Jefferson and McGrath, 1996). Any improvement in the way that patients with back pain can be analyzed should therefore be viewed as one potentially capable of significantly saving both benefit expenditure and lost person-hours.

The problem with back pain is that "there exist no standardised clinical tests or investigations by which all people with low back pain can be evaluated" (Papageorgiou et al., 1995). Nor will there ever be, as different people have different pain thresholds and will be affected differently. It is also difficult for medical personnel to know what has caused the back pain, as there are potentially many different causes behind it (Frank and De Souza, 2000).

Due to the debilitating effect that back pain has on society, our research aimed to find a method that would allow correlations and patterns to be found between patients' data, and therefore allow the medical world to draw conclusions as to the cause and effect of back pain. Within our research we devised and implemented four ways of visualization and accessing back pain datasets, all of which would enable the user to carry out deeper analysis on a dataset than is usually possible using standard database queries. This is the case as the human is able to compare and contrast information diagrammatically far faster and to a higher degree than just relying on statistical or numerical values.

13.1.1 Back Pain Questionnaires

The main medical work that is done to resolve back pain tends to be with patients that have chronic back pain. However, these patients may have developed psychological and emotional problems, due to having to deal with the pain. Because of these problems, patients can have difficulty describing their pain, which can lead to problems during the treatment. In some patients, the psychological problems may have aided the cause of the back pain, by adding stress to the body, or the stress of the back pain may itself have caused psychological problems (Ginzburg et al., 1988; Hildebrandt et al., 1988; Main, 1983; Mann III et al., 1992; Parker et al., 1995; Ransford et al., 1976; Uden et al., 1988; Von Baeyer et al., 1983). It is because of this factor that patients suffering from back pain are usually asked to fill out questionnaires of different types in order to help the medical staff, not only to know where the pain is located but also to identify the patients' mental state before treatment begins. The main questionnaires used for this purpose are:

- The Modified Somatic Perception Questionnaire (MSPQ), which assesses somatic anxiety (Main, 1983)
- The Roland and Morris (1983) questionnaire, which is used to measure the patient's back pain-caused disability
- The Zung (1965) questionnaire, which assesses depression via the respondent giving answers to 20 questions using a self-rating scale

In addition, the patient is usually required to mark on a diagram of a human body, where the pain is located and the type of pain. This type of diagram is known as a "pain drawing" and forms the primary focus of our chapter. Accordingly, the structure of the chapter is as follows: the next section looks at pain drawings in more detail, examining the different types used in practice and their scoring methods, and finishes by highlighting limitations of current approaches. The subsequent section examines the feasibility of various technological solutions to overcome these limitations, and is followed by a description of the implementation of these solutions in practice. Finally, the developed solutions are then compared with respect to one another and the set of requirements they set out to fulfil, and conclusions subsequently drawn.

13.2 The Pain Drawing

Pain drawings, an example of which (used in our study) is shown in Figure 13.1, have been successfully used in pain centers for over 45 years (Palmer, 1949) and act as a simple self-assessment technique, originally designed to enable the recording of the spatial location and type of pain that a patient is suffering from (Ohlund et al., 1996; Parker et al., 1995; Rankine et al., 1998). They have a number of advantages, including being economic and simple to complete, and can also be used to monitor the change in a patient's pain situation (Ohnmeiss et al., 1995). Over the years, different ways of evaluating and using pain drawings have been suggested.

Ransford et al. (1976) concluded that the pain drawings could be used not only as a location and pain recorder, but also as an economical psychological screening instrument to see if a patient would react well to back pain treatment. As previously mentioned, back pain can be caused by psychological and emotional problems, as

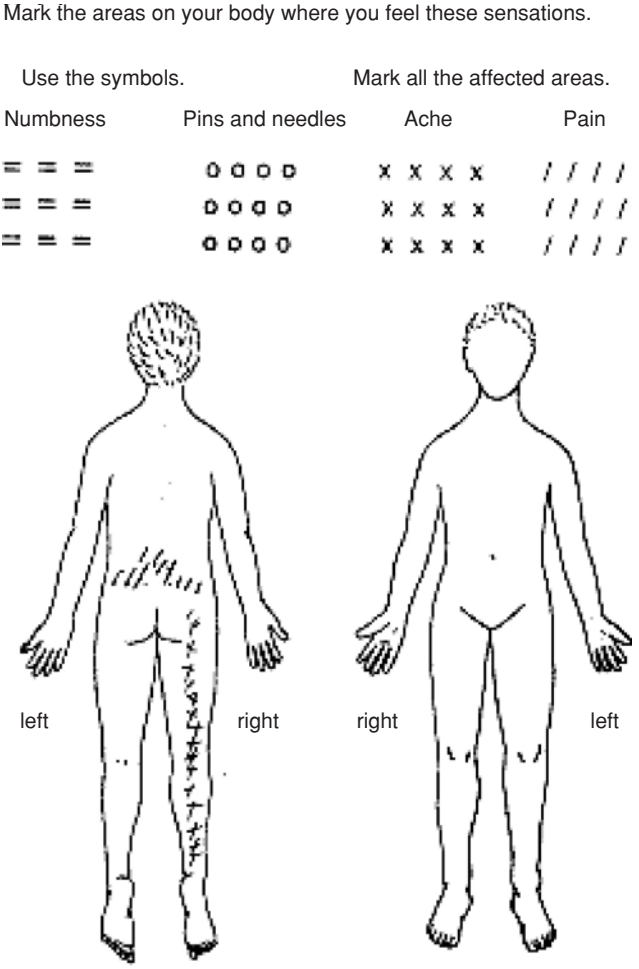


Figure 13.1 Sample pain drawing.

well as occupational factors, and hence medical treatment alone may not remove the cause of the pain, making the patient no better (Chan et al., 1993; Hildebrandt et al., 1988; Uden et al., 1988).

In order to evaluate the patient's psychological state the Minnesota Multiphasic Personality Inventory (MMPI), a standard American psychological questionnaire, can be used (Waddell et al., 1980). This has been proven in a double blind study to indicate hypochondriasis (Hs) and hysteria (Hy) scores for patients, factors that have both been linked to treatment outcomes(Wiltse and Racchio, 1975). However, the MMPI is expensive and takes on average around one and a half hours to complete, also requiring the respondent to understand English to a high school level in order to be able to complete the questionnaire (Von Baeyer et al., 1983). Therefore, research was done by Ransford et al. (1976) to link the pain drawing with the MMPI, and see if it would be capable of acting as a simple screening device, filtering those in need

of further psychological evaluation. This worked by using a scoring system for the pain drawing, which gave points for abnormalities in the pain drawings (drawings that did not match accepted patterns of pain). If this score was greater than three, the patient could be psychologically distressed. Ransford et al. (1976) found that they could predict 93% of the patients that needed further psychological evaluation just by looking at the patient's pain drawing, a conclusion later corroborated by Chan et al. (1993), and to a lesser extent by Von Baeyer et al. (1983). The latter concluded that while relationships between the pain drawing score and the Hs and Hy scores in the MMPI were present, the magnitude of these relationships was much smaller than that published by Ransford et al. (1976).

Mann III et al. (1992) have tried to link the pain drawing to causes of pain. In their paper, five lumbar spine disorder categories were used, and analysis was made of the amount of patients that were correctly identified. The identification was done using a number of systems, including statistics, expert, and computer evaluations. It was found that physicians averaged 51% accuracy, while the computer systems achieved 48% accuracy. They conclude that computer systems could be used to help with the diagnosis of back pain patients and could be used to bring down the costs of this type of evaluation.

13.2.1 Scoring Methods

In order to try and link the pain drawing to either psychological or emotional causes of pain, several scoring systems have been developed. With the *grid method*, a grid overlay is placed over the pain drawing (Gatchel et al., 1986). The grid is designed so that each cell is approximately the same size. By using the grid, unskilled testers could calculate the amount of surface area that was in pain. *Sensation scoring*, on the other hand, allows not only the placement of pain to be noted but also the type of pain (e.g., numbness, stabbing, pins and needles) in the respective area. This type of scoring is done using a key, thereby allowing more information to be collected, and could act as an aid to the clinic as to what the cause of the pain is.

In the *simple body region* approach the body is broken down into very simple regions, in order to indicate large areas that were in pain (Ohnmeiss et al., 1999), whereas in the *body region* method the pain drawing is broken down into regions using anatomical landmarks, such as joints (Margolis et al., 1986). In contrast, *visual inspection* uses trained evaluators, who look at the pain drawings and from their experience are able to say what they believe to be wrong with the patient, or if psychological testing is needed (Chan et al., 1993). Lastly, *penalty point methods* work by awarding points for every unnatural placement of pain on a pain drawing. Different areas and rules are made so that there is a weighting depending on the irregularities in the drawing. If more points than normal are scored, then the person in question may have a psychological problem that needs addressing (Ransford et al., 1976).

Many investigations have been carried out to assess the various processes involved with data collection and dissemination using pain drawings. These include the patient's ability to redraw the pain drawing after a time delay (Ohnmeiss, 2000), as well as the drawing evaluator's ability to interpret the information the same way each time (Hildebrandt et al., 1988; Margolis et al., 1988; Von Baeyer et al., 1983). Further investigations have also been carried out into the ability of two different evaluators to agree on their interpretation of the pain drawings (Chan et al., 1993; Uden et al., 1988). The findings showed that patients remain constant in the way that they fill in

pain drawings, even if the time between filling them in is quite lengthy. It has also been found that this was correct across all scoring systems used in pain drawings except sensation scoring (Ohnmeiss, 2000). As far as the ability of evaluators to agree on their interpretation of the pain drawing, Chan et al. (1993) report agreement between evaluators 78% of the time, and between evaluators and a registered nurse 74%. This, they claim, proves that the pain drawing is as reliable as any other commonly used process.

13.2.2 Pain Drawings—Conclusions

The consensus of the literature seems to be that the pain diagram is a powerful tool in the role that it is designed for, namely to record the spatial location and pain type. However, pain drawings are usually stored in a paper format, which allows no further evaluation of the data that are stored upon it and makes searching through the data somewhat an arduous task. To compound the issue, if information regarding the pain drawing is stored, usually it is simply just a numerical result of statistical analysis that has been carried out on it, inevitably resulting in a loss of information. This is due to the fact that current systems that are used for analysis of the pain drawings and the associated questionnaires revolve around statistical packages, such as Excel and SPSS, which are incapable of handling diagrammatic data. Thus, although diagrammatic data are collected, they are not used as the key component to the data analysis tools. This is somewhat a problem, as people will find it easier to show through a diagram the way that they feel, instead of answering closed questions in questionnaires. Such data cannot therefore be used to their full potential and, in particular, cannot be used in helping with queries within the dataset.

In our work, we have sought to alleviate this problem and have investigated various technological solutions that use the pain drawing as an actual aid to the analyzing of the dataset. Furthermore, in our approach, we have enhanced data management by digitally storing the data in ways that allow it to be analyzed easier, and have employed user-friendly visual techniques for data querying. Lastly, recognizing the importance in healthcare of distributed systems such as the World Wide Web providing ubiquitous information, all our approaches use Web-based technologies or technologies that could be implemented in this manner if required.

13.3 Back Pain Data—Technological Solutions

The back pain drawing that a patient completes can be stored in one of two ways: either the image can be scanned, or the image can be subjected to *regionalization*. In the latter case, the image is firstly broken down into regions, and only information relating to those regions of the human body affected by pain is recorded. The drawback of this approach is that pain location is *generalized*: for instance, numbness in the hand might be generalized to numbness in the whole arm, if that is the smallest region encompassing the hand. However, regionalization leads to simple *image maps*: digitized drawings, broken down into regions, each region being hyperlinked using HTML (the HyperText Markup Language, used for writing Web pages) to a specific document, such as a patient's medical records. Therefore each region becomes an image map hotspot linking to records that relate to that region. Use of image maps allows for easy data cross-examination, a feature absent if the image is simply scanned. However, scanning an image does allow for the drawing to remain intact, with precise

indications of the location and types of pain. It can, if it is scanned to appropriate storage formats such as GIF or JPEG, be hyperlinked to the rest of the dataset.

In our approach, Web-based image maps were constructed in one of two ways: either by using a GIF image (broken down into regions using Macromedia's Fireworks package) in conjunction with HTML, or by using Scaleable Vector Graphics (SVG), a new technology that allows zoomable and panable image maps to be designed. Both types of image maps can be used together with Active Server Pages (ASP) to dynamically update and present data. In our work we have implemented GIF image maps on both conventional and ubiquitous computing platforms. Moreover, we have also used Geographical Information Systems (GIS), a specialist analyst tool, which inherently works on image maps. These technologies are now presented in more detail.

13.3.1 SVG

SVG is a new markup-based vector graphics format, which the World Wide Web consortium (W3C) has made into an open Web standard. SVG is an XML styled tag language that allows a text file to store the way that a vector graphic should be displayed. This has advantages over raster-based formats such as GIF and JPEG, which store data about each of the pixels in the image, resulting in large files. Another problem associated with raster images is that zooming tends to turn the image blocky. Vector graphics, on the other hand, just store the end coordinate points that define lines, thus allowing the creation of polygons that not only can be color filled, but that *may have their own attributes*. The fact that only data about the end points of a line are stored in such images makes zooming and panning of the image at any size possible. Moreover, the storage requirements of this type of image are also less than the raster images (Boye, 1999). Lastly, the fact that SVG is XML-based, and is therefore a text-based graphics language with semantic markup, enhances the ability of search engines to hunt through the format and extract meaningful information for the end-user.

13.3.2 ASP

ASP allows for dynamic content to be used on the Web. The text document that is used to build the Web page contains either Visual Basic or Javascripting and requests the server to carry out some functions, such as database data retrieval or updates, before the HTML page is built dynamically, at runtime. Once the server has carried out its operation, the instructions for laying out the Web page are sent to the client. One of the main uses of this technology is to allow Web pages to interface, and get results from, a database (VB123, 1999). In order to make use of this technology, the ASP queries and the database have to be stored either on an Internet Information Services Server or Personal Web Server, each of which can carry out queries on the database and return the information requested.

13.3.3 GIS

With the advent of computing and information systems, the analysis of complex geographical datasets and their related databases and flat files has been greatly enhanced

by GIS technology (Bernhardsen, 1992). GIS tools such as ArcView allow the user to visualize data that may have gone unseen in spreadsheets, charts, and other types of reports (ESRI, 1999). GIS, however, does not need to be a single system, as it can be made up of a number of different hardware and software components, each performing a role in the storing and integration of digital images and related geographical data, thereby allowing for fast information retrieval (Bretas, 1996).

Using GIS, several methods of analysis can be carried out on the data, such as selection by geographic criteria for the spatial dataset, or using standard database functions such as sum, maximum, minimum, average, frequency distribution, and standard deviation on the nonspatial data held in the database. As most GIS are built using relational databases, SQL (Standard Query Language) statements can also be used in such systems (Bretas, 1996). As the system is visual, it removes the complexity of paper files or large spreadsheets and allows users to point and click in logical ways through the datasets (Theodore, 1998). For instance, if an area of the visual image is selected, the area that has been selected will be highlighted and all corresponding data in the related tables will also be highlighted and vice-versa.

In order to build a GIS, a base map is used, where every point, line, and area has been given a unique identification code. These codes can then be linked to the database by inserting a new linking attribute into the database. The GIS software then automatically builds all the links that are needed for the system to work.

13.3.4 Visualization for Mobile and Embedded Applications

Microsoft Embedded Visual Basic is a language specifically geared to help developers build applications for the next generation of communication and information-access devices running Windows CE.

Once GIF-image maps are implemented and ported on a mobile computing platform, a developed application can read the coordinates of the pain locations from a touch-sensitive screen and using ADOCE 3.0 (Active Data Objects for CE) connect to a local Microsoft Pocket Access database file (Figure 13.2). Through this connection, the application saves the pain coordinates and patient questionnaire data to the database. When the user is within wireless Internet coverage, the application can then use Winsock CE 3.0 (Windows CE Sockets) to send a connection request to the Internet Information Server, which then connects to the hospital database through

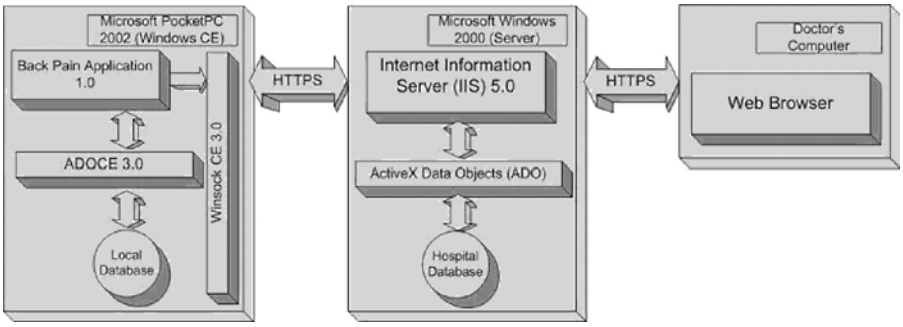


Figure 13.2 Mobile and ubiquitous data capture system architecture diagram.

Open Database Connectivity (ODBC). The hospital server hosts a patient database, which is accessed by the Internet Information Server using ActiveX Data Objects (ADO).

The doctor's interface to the system is made of dynamically created Active Server Pages (ASP), which can be accessed using any conventional Web browser running on a computer connected to the Internet. Thus, after successful authorization, medical personnel can download a particular patient's data to their personal computer. This is achieved through the ASP code dynamically creating an SQL query to the database, the results of which are presented dynamically on the viewed Web page.

13.4 System Requirements and Development

In our research, we have worked with back pain data provided by NHS Northwick Park Hospital. Here, medical staff currently collect data relating to patients with back pain by the use of a number of questionnaires including the three questionnaires described earlier in the chapter, as well as pain drawings, where the pain drawing is stored on paper and the questionnaire results are stored in a spreadsheet. The latter allows for a small degree of analysis; however, any complex queries are out of the question. Moreover, as the diagrammatic data are stored in paper format, it only allows for simple one-to-one comparisons and makes cross-examination and data recovery between patients difficult.

After consultations with medical staff at NHS Northwick Park Hospital it became clear that they needed a system that would allow back pain data to be able to be analyzed fully. In order to achieve this goal, both the questionnaire data and the diagrammatic data would have to be converted into more analyzable methods of digital data storage using technologies such as HTML, SVG, ASP, GIS, and Embedded Visual Basic, which would allow for easy cross-examination and recovery of the data. The developed solution should allow for medical personnel and patients to enter new data, as well as to be able to examine the information stored in the system, using a user-friendly visual-based interface. Moreover, the medical staff interviewed stressed that it would be an advantage if the developed system could provide the ability not only to allow the use of the back pain diagram to select the appropriate records, but, conversely, also to enable the highlighting on the diagram of the region(s) corresponding to the selected records. Lastly, it would also be of use if the system could be run over an intranet or the Internet and, indeed, if a mobile solution for collecting data was implemented.

Based on these requirements, a total of eight different design solutions were initially suggested. These were then assessed as to what their practicality was and it was decided to narrow down the number of solutions to four, in order to allow Northwick Park the flexibility of choosing between the competing designs. Thus, the following systems were implemented:

- ASP related database accessed by a GIF image map
- ASP related database accessed by an SVG image map
- GIS-based solution
- Wireless enabled Personal Digital Assistant (PDA) solution

As the issue of relating the digitized pain drawings to the corresponding datasets was primordial in our work, we shall now describe our approach in achieving this goal, followed by a more detailed examination of each developed system.

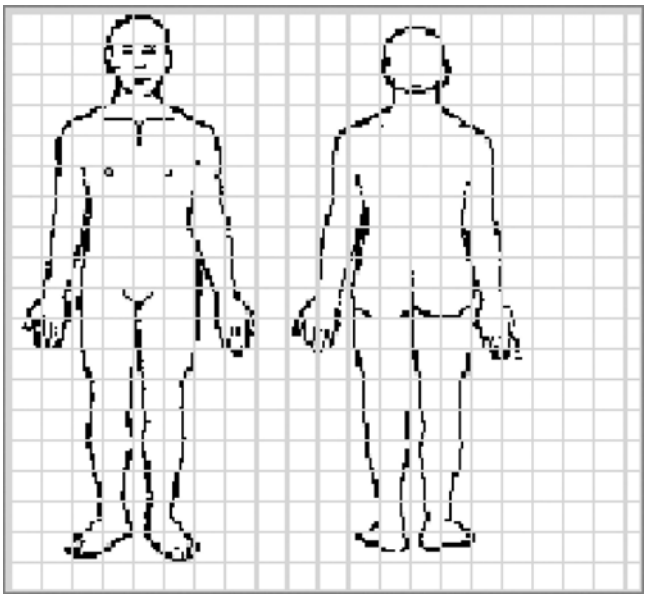


Figure 13.3 Grid approach used in our work.

13.4.1 Regional Diagram for Visual Interaction

In order to allow the digitized diagrams to be interactive with the data set, thereby allowing the display of information relating to patients that had a particular type of pain in a specific region, the pain drawing itself had to be broken down into regions. It was decided that the best way to split the human body into regions was to use either a dermatome map or one based on a grid approach. The reason for choosing the dermatome map was that most medical staff would understand what dermatomes represent and their mapping on the human body. On the other hand, the grid method of regionalization is conceptually simple and easy to implement in practice. Indeed, both approaches give a simple system of body regionalization that everyone would already understand or where simple training would get them to the level of understanding needed. While in the grid approach 10×10 pixel regions were used (Figure 13.3), in the dermatome one, in order to allow for a better resolution of pain regions, the body was split into quarters, thus leaving a rear left and right, and a front left and right for each dermatome. All such regions had to be made into hotspots for SVG, GIF systems as well as GIS, with all regions following a standard code, described in Table 13.1 and graphically depicted in Figure 13.4.

Table 13.1 Dermatome regions and their abbreviations

RRT1	Rear Right Dermatome T1
RLC2	Rear Left Dermatome C2
FRL1	Front Right Dermatome L1
FLS2	Front Left Dermatome S2

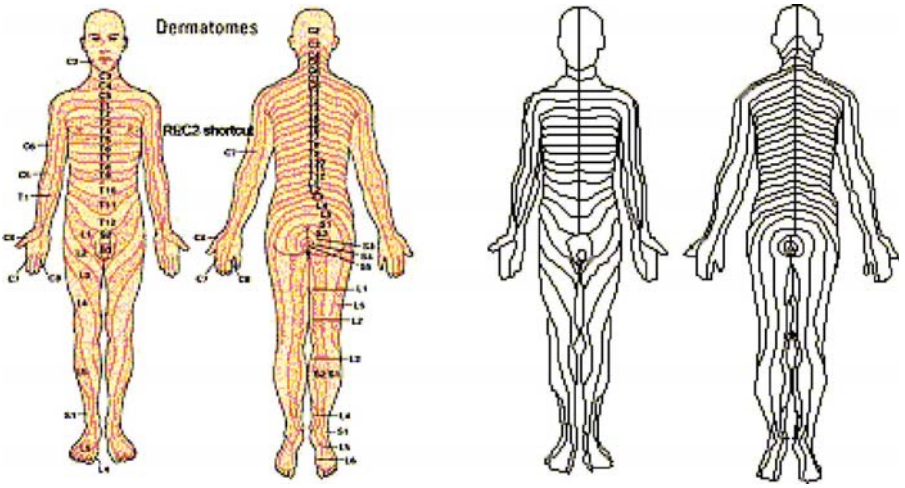


Figure 13.4 The original dermatomes used in our work, and the corresponding regions used within the research.

13.4.2 ASP with a GIF Image Map

With this design, the GIF image map contains links that open other options, displayed in frames on the Web page. These allow a selection of the type of pain and data required, relating to the selected dermatome. When one of these is selected, ASP queries stored on a Web server would then operate on the resident back pain database using SQL, before then returning the result in an HTML Web page, as shown in Figure 13.5. As the original scanned images are also stored on the Web server, this approach also allows for any generalization errors to be checked for.

13.4.3 ASP with SVG Image Map

This implementation works the same way as the previous one; however, instead of having a GIF image map for use as the shortcut to the ASP queries, it uses an SVG image map, comprised of four dermatome drawings, one for each different type of pain. Because the system uses four different SVG images of the dermatomes, it allows users to select the pain type that they wish to review from a visual representation instead of a hyperlink (as is the case with the preceding solution). SVG aids this, in that although the diagrams are relatively small, they can be zoomed and panned to allow the user to click directly on the region needed. All possible queries, relating to a specific area and pain type, were stored on the PWS on the machine, in ASP format. XLink functions were then used so that if a region was selected, a Web page would be loaded with hyperlinks to all the different tables that may contain information relating to patients that had pain of the specified type and in the specified region. Once one of these hyperlinks was selected the ASP would run and load a new page, which would show the resulting data, as depicted in Figure 13.6.

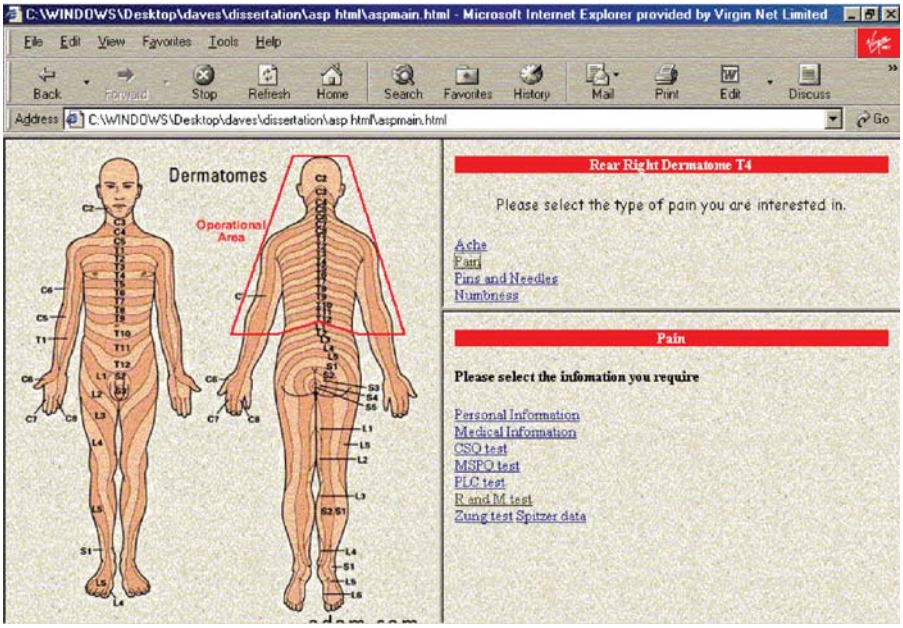


Figure 13.5 ASP with GIF image map—system snapshot.

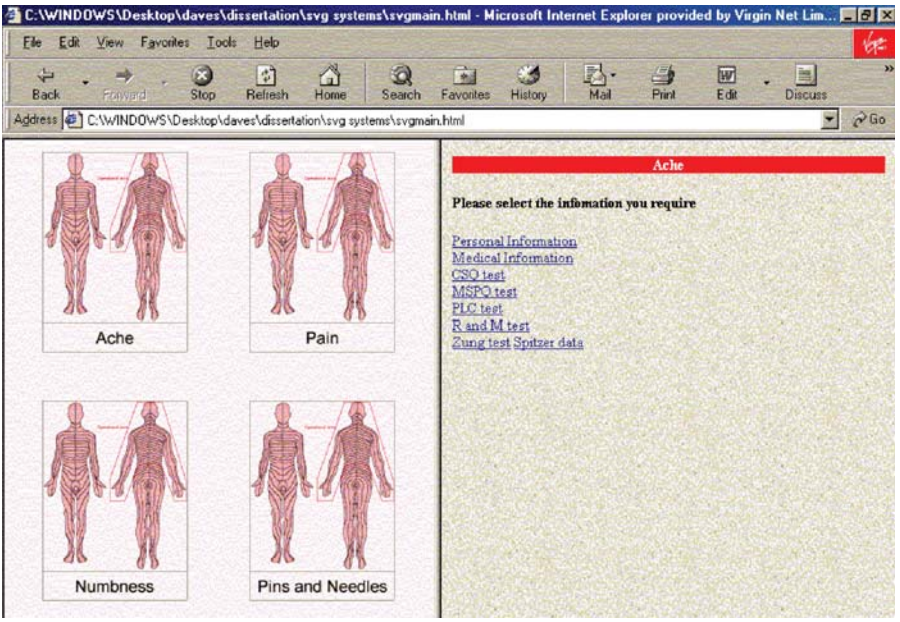


Figure 13.6 ASP with SVG image map—here the user selects region and type of pain from the graphic, which in turn gives the user the option of choosing the type of data needed.

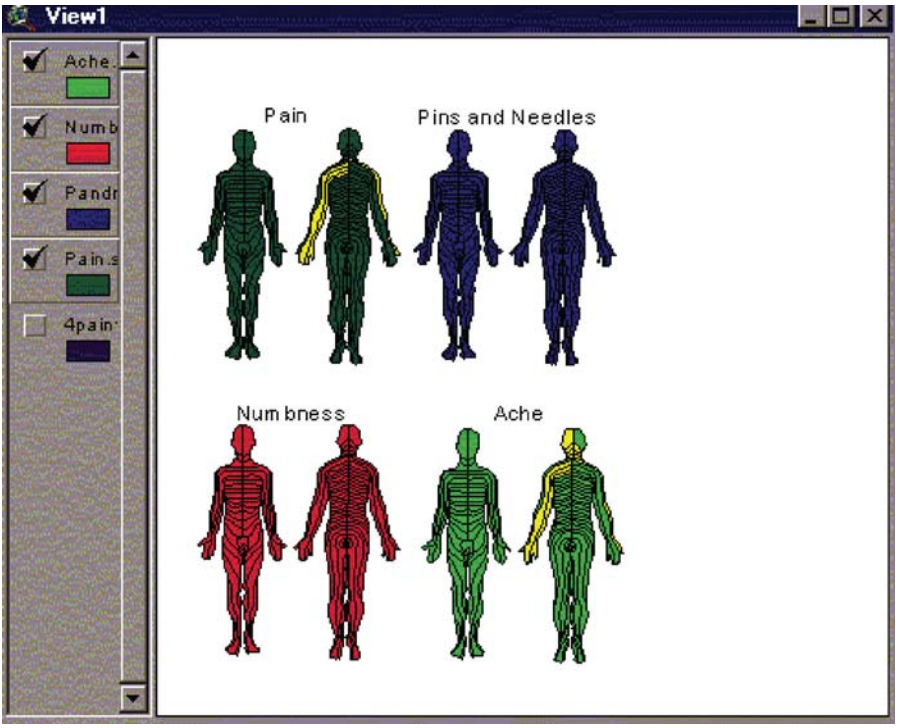


Figure 13.7 Simultaneous selection of dermatomes and types of pain in the GIS solution.

13.4.4 GIS

The GIS solution was created by digitizing the outline of each dermatome, stored as spatial coordinated polygons. This was then copied four times, one for each type of pain. Each polygon that was thus created in this manner was then given a unique identification number and therefore could be easily linked to the database via its keys. Each polygon was also given its dermatome code name (e.g., "RRT1"), which related to the corresponding region. A new flat file table was created; this contained the four types of pain across the top, and each row then contained the study number and the area of pain.

This table stored under different attributes the areas of pain that each patient suffered from (Figure 13.7). Each of the different attributes were then linked to the corresponding dermatome image, as well as the rest of the database, as shown in Figure 13.8.

13.5 Wireless-Enabled PDA Solution

We have also developed a wireless-enabled, ubiquitous solution that uses the pain drawing as an actual user-friendly visual aid to the input and analysis of back pain datasets. While our solution is generic and applicable to all back pain sufferers that have access to wireless technology, we have specifically targeted wheelchair users due to

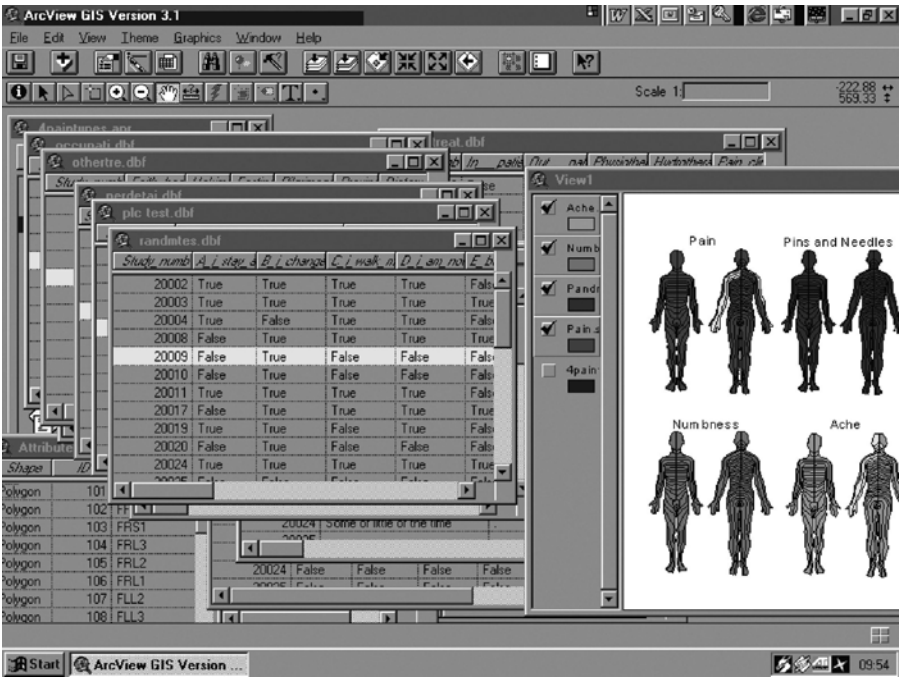


Figure 13.8 GIS solution—snapshot showing the pain areas and details of patient 20009.

their severe mobility limitations (which might mean that they might not, for instance, easily have access to a desktop-based computer) and their dynamic pain patterns, which are now easily logged by the developed application. In so doing, we specifically address the issue of pain variability in time, as identified by Gibson and Frank (2004), and our application can thus also be used as a data gathering tool for this still incompletely understood phenomenon, the solution of which has potentially important implications in the monitoring of the effectiveness of back pain treatment and medication.

The Back Pain Application was implemented on an HP iPAQ 5450 PDA with 16-bit touch-sensitive transflective thin film translator (TFT) liquid crystal display (LCD) that supports 65,536 color. The display pixel pitch of the device is 0.24 mm and its viewable image size is 2.26 inches wide and 3.02 inches tall. It runs Microsoft Windows for Pocket PC 2002 (Windows CE) operating system on an Intel 400 Mhz XSCALE processor and contains 64 MB standard memory as well as 48 MB internal flash ROM. The Web server was implemented on an Intel Pentium III running at 1 GHz, with 512 MB RAM and a 50 GB hard disk. In our work, a 10 Mbps D-Link DWL-700AP wireless access point was used.

The underlying structure of our application is based on a three-tier wireless system model (Figure 13.9) where the three main components are a mobile, wireless-enabled device, a Web server with scripting capability, and a backend database.

In this model the patient inputs, on a wireless-enabled device (in our case, a PDA), pain information. This is done at specific time intervals, as requested by clinicians, and the information is saved to a local backend database. Each pain type is represented with a different shape and color; a yellow square for numbness, a black cross for

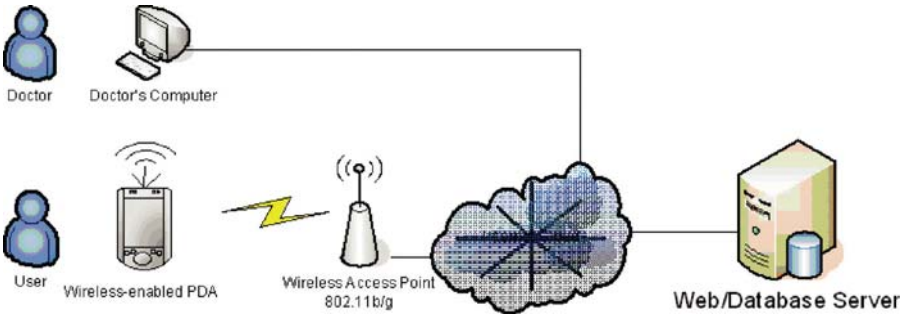


Figure 13.9 Wireless system model.

pins and needles, a red triangle for pain, and a blue circle for ache. The colors and symbols were chosen so that potentially a selected location could have all four pain types displayed concurrently and clearly. Subsequent to pain type selection, tapping on the screen creates one of the shapes mentioned that symbolizes the pain type and its location on the overall body. The patients can add as many pain symbols as they want to present their pain type and location. In case of erroneous pain type or pain location selection, the user can delete any symbol from the diagram by selecting the same pain type with the target symbol and then tapping on it. When the user has finished inputting his or her pain information of the diagram, he or she then taps on the *Save Pain Points* button and saves the pain type and location data, as well as the particular time at which these were gathered, on the PDA database (Figure 13.10).

Whenever the user is within a wireless-enabled zone, he or she then connects (Figure 13.11) to a Web/Database server via a wireless access point, using the Hypertext Transfer Protocol over Secure Socket Layer (HTTPS). Moreover, the connection between the PDA and the wireless access point is itself secured through the use of 128-bit Wired Equivalent Privacy (WEP) encryption.

Upon receiving such requests, the server responds back and asks for appropriate authorization. After this has been successfully completed, the data are then uploaded to the hospital server. The clinician then uses his or her computer to log onto the Web server and downloads information regarding any specific patient and their pain pattern from the database for further analysis.

13.6 Solutions Review and Comparison

We now review how the developed systems satisfied the original requirements set by Northwick Park Hospital, followed by a detailed comparison between the solutions. Accordingly, all of the implemented solutions were able to store patient questionnaire data, and the ASP systems also stored the scanned pain drawings, linking the database relations with the corresponding pain drawings. All of the developed systems, with the possible exception of the GIS solution, where supplementary training is required to be able to take advantage of its full functionality, use standard, familiar Web interfaces and point-and-click/tap functionality to remove the complexity of the database from the user. Adding new patient records is straightforward in all of the solutions, as is patient data input. Moreover, all the information relating to a patient can be edited or deleted, including the pain drawing, though with the ASP-based systems this has to

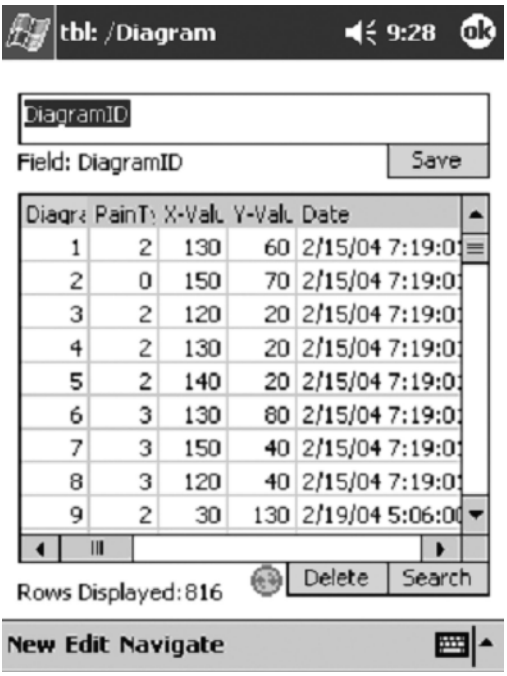


Figure 13.10 PDA back pain database.

be done by someone with access to the Web server. However, all other solutions work under a client-server framework and place their data on a Web server.

All of the systems allow the linking between the image maps and the information in the database, thereby enabling the largest part of the query to be done graphically, as the region selection is done by point-and-click/tap instead of a complex query written

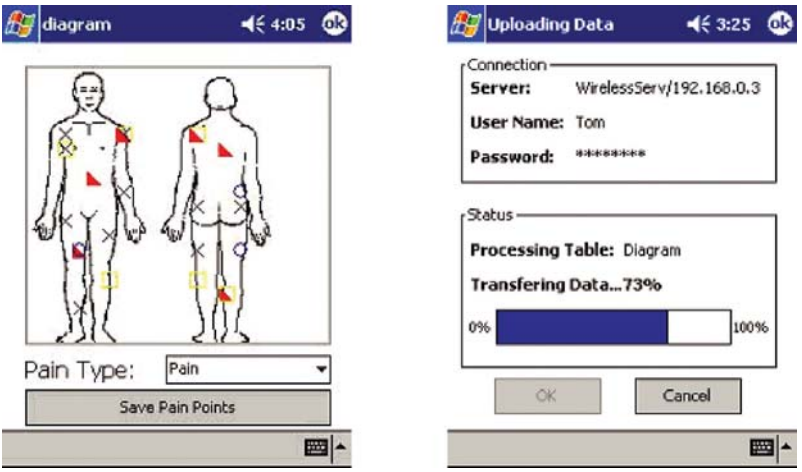


Figure 13.11 Collection and uploading of back pain data.

by the user. However, the only solution that highlights the regions on the image maps when corresponding data have been highlighted in tables is the GIS. This is part and parcel of the GIS functionality, as not only does a GIS show highlighted regions on the diagram, it also highlights all corresponding data on all of the tables related to them.

Last but certainly not least, all developed solutions allow the recording of time-varying pain data; in the case of back pain this is an insufficiently understood phenomenon and its recording and correlation with prescribed medication and treatments can ultimately lead to better diagnosis and management of back pain patients.

13.6.1 Comparisons

All of the developed solutions use image maps in order to allow users to select an area of interest to them. This is where the similarity between the systems ends, however. The GIF-based solutions use an image map in order to allow users to select the region of interest; the rest of the selection is done via hyperlinks, which allows the pain type and type of information to be selected. The main advantages of this system are that the users do not need to understand the underlying technologies employed; they just have to be able to use a Web browser or a PDA.

Although some of the patients who evaluated our solutions, especially those suffering from arthritis and/or poorer eyesight, did encounter difficulties in using the relatively small interface of the PDA, overall the participants agreed that the processes of recording pain data was a relatively easy one and that the ability to record data across time, irrespective of the particular location in which users found themselves, was indeed beneficial. Indeed, recognizing the mobility problems that many back pain patients endure, our developed PDA solution is WiFi enabled, thus facilitating remote, ubiquitous data access and management and absolving patients of the need to actually physically hand in their completed questionnaires.

The SVG-based solution works the same way as the GIF-based systems, but allows a more detailed zoomable and panable graphic system to be used, which contains an image for each of the different types of pain that the patient has described. It therefore replaces the hyperlink pain selection process with the corresponding choice being made via the dermatome image selection. After consultations with specialists from Northwick Park Hospital it was decided that, due to the enhanced functionality of its interface, the SVG solution is the more adequate one of the two.

Both GIS and the PDA-based solutions allow the user to select as many areas of interest as he or she wants at the same time. This means that, when using this solution, patients that had pain in the left leg and left arm could be selected concurrently. If the user made this selection by pointing and clicking it would return all the results (i.e., pain in leg only, pain in arm only, and pain in both).

Moreover, while the desktop and SVG-based solutions only allow users to request certain datasets (i.e., personal datasets, medical, etc.), with the PDA and GIS systems users can also query the underlying database. This allows users to construct queries such as "Show me the position of pain for everyone that answers question 6 of the MSPQ in a particular manner."

Finally, while the other solutions only allow users to look at each dataset as a separate entity, as normally users would look at one table at a time, GIS highlights on each table the information regarding the currently selected record(s). This therefore allows users to carry out a much more in-depth type of analysis using the GIS than can be done

with the other methods, which automatically makes this solution the most suited for finding patterns. However, this system needs a high level of training compared with the rest of the systems, but the rewards are a lot higher. The main drawbacks of this system, though, are its complexity and setup costs.

13.7 Conclusions

Back pain is one of the most prolific health problems within the population and costs industry lost revenue due to the amount of days people have to take off in order to recover. For several decades now hospitals have been collecting data in order to help analyze back pain patients. These have included several different psychometric and after-treatment questionnaires, as well as pain drawings that are used to record position and type of pain. However, little or no work has been done in which pain drawings and questionnaires are used to check for data patterns between patients. Even though there was positional data, no one seemed to have devised any system that made use of the positional data to link pain areas to the corresponding datasets.

In our work, we proposed a number of solutions to this problem, four of which have been presented in this chapter.

For desktop-based data collection, the most appropriate solutions were found to be an SVG-enabled system, which used ASP for remote data access, and one based on GIS technology. The former allowed users to use Web technology and pan around the drawings, before clicking for the data that they wished to view. The GIS made full use of the Geo Spatial analysis tool in order to view and analyze the data. We have also implemented a solution for the ubiquitous collection of back pain data based on a pain drawing interface. Furthermore:

- GIS technologies can be used in none geographical fields, as long as the dataset contains or can contain, spatial data. In this case the GIS was used to map the body. This allows the whole spatial analysis tool set to be used on none geographical data, and helps the user to find correlations or patterns which may have taken a lot longer using standard glance methods. Queries such as: "Does a patient with pain in both limbs feel worse than a patient with pain in only one, and is (s)he more likely to take time off work?" can easily be undertaken using the GIS solution.
- SVG makes for better image maps than the standard GIF images. This is due to the scale and pan features of this new technology, which allow both the user to zoom in and get a better idea of the region areas, as well as the image to take up less space on the screen, as the whole image need not be shown at a large scale.
- The challenge of pervasive clinical data collection has only now become attainable through recent advances in mobile and wireless communication capabilities. Only by harnessing the combined features and capacities of such technologies will healthcare applications be able to offer stakeholders location-independent communications and computing. We have developed a proof of concept ubiquitous data collection application for back pain patients allowing anywhere/anytime data recording and seamless interconnectivity.

Moreover, our work has raised some interesting future research questions. Chief among these is the issue of body regionalization. Although in our work we have split the body region into dermatomes, is this the best grid possible for back pain analysis, especially bearing in mind generalization effects? Would a finer resolution grid, albeit resulting in a more complex and resource-hungry solution, be more beneficial?

Finally, work can also be done to explore how SVG can be combined with other XML-based medical markup languages in order to harness, on the one hand, SVG's powerful graphic capabilities with XML's enhanced search and analysis potential, on the other.

13.8 References

Bernhardsen, T. (1992). *Geographical Information System.* Viak IT, Norway.

Boucher, A. (1999). *The Prevalence of Back Pain in Great Britain in 1998*, Department of Health. Available: http://www.doh.gov.uk/public/backpain.htm.

Boye, J. (1999). *SVG Brings Faster Vector Graphics to Web.* Available: http://www.tech.irt.org/articles/js176/.

Bretas, G. (1996). *Geographic Information Systems for the Study and Control of Malaria. GIS for Health and the Environment.* Available: http://www.idrc.ca/books/focus/766/bretas.html.

Chan, C.W., Goldman, S., Ilstrup, D.M., Kunselman, A.R., O'Neill, P.I. (1993). The pain drawing and Waddell's nonorganic physical signs in chronic low-back pain. *Spine*, 18:1717–1722.

ESRI (1999). *Health: GIS Solutions for Health Sciences and the Business of Health Care. Environmental Systems Research Institute.* Available: http://www.esri.com/library/brochures/pdfs/heathbro.pdf.

Frank, A., De Souza, L.H. (2000). Conservative management of low back pain. *International Journal of Clinical Practice*, 55:21–31.

Gatchel, R.J., Mayer, T.G., Capra, P., Diamod, P., Barnett, J. (1986). Qualifications of lumbar function, Part 6: The use of psychological measures in guiding physical function restoration. *Spine*, 11:36–42.

Gibson, J., Frank, A.O. (2004). Pain experienced by Electric Powered Indoor/Outdoor Chairs (EPIOC) users: A pilot exploration using pain drawings. *Proceedings of the 2nd Meeting of the European Federation of Physical and Rehabilitation Medicine*, Vienna, May, p. 114.

Ginzburg, B.M., Merskey, H., Lau, C.L. (1988). The relationship between pain drawings and the psychological state. *Pain*, 35:141–145.

Hildebrandt, J., Franz, C.E., Choroba-Mehnen, B., Temme, M. (1988). The use of pain drawings in screening for psychological involvement in complaints of low-back pain. *Spine*, 13:681–685.

Jefferson, J.R., McGrath, P.J. (1996). Back pain and peripheral joint pain in an industrial setting. *Arch Phys Med Rehabil*, 77:385–390.

Main, C.J. (1983). The Modified Somatic Perception Questionnaire. *Journal of Psychosomatic Research*, 27:503–514.

Mann III, N.H., Brown, M.D., Hertz, D.B., Enger, I., Tompkins, J. (1992). Initial-impression diagnosis using low-back pain patient pain drawings. *Spine*, 18:41–53.

Margolis, R.B., Chibnall, J.T., Tait, R.C. (1988). Test-retest reliability of the pain drawing instrument. *Pain*, 33:49–51.

Margolis, R.B., Tait, R.C., Krause, S.J. (1986). A rating system for use with patients' pain drawings. *Pain*, 24:57–65.

Ohlund, C., Eek, C., Palmblad, S., Areskoug, B., Nachemson, A. (1996). Quantified pain drawing in subacute low back pain.*Spine*, 21:1021–1031.

Ohnmeiss, D.D. (2000). Repeatability of pain drawings in a low back pain population. *Spine*, 25:980–988.

Ohnmeiss, D.D., Vanharanta, H., Ekholm, J. (1999). The relationship between pain location and disc pathology: A study of pain drawings and CT/discography. *Clinical Journal of Pain*, 15:210–217.

Ohnmeiss, D.D., Vanharanta, H., Guyer, R.D. (1995). The association between pain drawings and computed tomographic/discographic pain responses. *Spine*, 20:729–733.

Palmer, H. (1949). Pain charts: A description of a technique whereby functional pain may be diagnosed from organic pain. *NZ Med J*, 48:187–213.

Papageorgiou, A.C., Croft, P.R., Ferry, S., Jayson, M.I.V., Silman, A.J. (1995). Estimating the prevalence of low back pain in the general population: Evidence from the South Manchester back pain survey. *Spine*, 20:1889–1894.

Parker, H., Wood, P.L.R., Main, C.J. (1995). The use of the pain drawing as a screening measure to predict psychological distress in chronic low back pain. *Spine*, 20:236–243.

Rankine, J.J., Fortune, D.G., Hutchinson, C.E., Hughes, D.G., Main, C.J. (1998). Pain drawings in the assessment of nerve root compression: A comparative study with lumbar spine magnetic resonance imaging. *Spine*, 23:1668–1676.

Ransford, A.O., Cairns, D., Mooney, V. (1976). The pain drawing as an aid to psychologic evaluation of patients with low-back pain. *Spine*, 1:127–134.

Roland, M., Morris, R. (1983). A study of the natural history of low back pain, Parts 1 and 2. *Spine*, 8:141–150.
Theodore, J. (1998). BodyViewer Extension for ArcView GIS. *ESRI Announces*
Uden, A., Astrom, M., Bergenudd, H. (1988). Pain drawings in chronic back pain. *Spine*, 13:389–392.
VB123 (1999). *Understanding How ASP Can Interrogate Your Database. VB123.com.* Available: http://www.VB123.com/toolshed/99_dbWeb/05ASPintro.htm.
Von Baeyer, C.L., Bergstrom, K.J., Brodwin, M.G., Brodwin, S.K. (1983). Invalid use of pain drawings in psychological screening of back pain patients. *Pain*, 16:103–107.
Waddell, G., McCulloch, J.A., Krummel, E.D., Venner, R.M. (1980). Nonorganic physical signs in low-back pain. *Spine*, 5:117–125.
Wiltse, L.L., Racchio, P.D. (1975). Preoperative psychological tests as predictors of success of chemonucleyolsis in treatment of the low back pain syndrome. *Journal of Bone and Joint Surgery (am)*, 57:478–483.
Zung, W.W.K. (1965). A self-rating depression scale. *Archives of General Psychiatry*, 32:63–70.

Chapter 14

Social Network Analysis on the Semantic Web: Techniques and Challenges for Visualizing FOAF

John C. Paolillo and Elijah Wright

14.1 Introduction

The Semantic Web promises to provide new applications for Internet users through the use of RDF metadata attached to various information resources on the Web. Yet it is somewhat unclear who will provide the metadata, or what will motivate people to provide it, let alone the exact nature of the applications the Semantic Web will ultimately support. What will the "killer app" of the Semantic Web be, and what shape will it take? An answer to this question may have already arisen, in the form of the Friend-of-a-Friend (FOAF) vocabulary. The FOAF project was begun in 1999 to explore the application of Semantic Web technologies (RDF/XML) to describing people's personal details: their professional and personal lives, their friends, interests, and other social dispositions. Its main product is the FOAF vocabulary, an RDF/XML namespace with elements defined for describing an individual's social sphere (Brickley and Miller, 2003).

Recently the FOAF vocabulary has been adopted by many large Web-logging ("blogging") and social networking software sites, such as LiveDoor, LiveJournal, and others. Weblogs, a recent Internet phenomenon, are diary-like sites usually consisting of entries in reverse-chronological order. Herring, et al. (2004) situate Weblogs as a genre bridging between extant media technologies and new forms of computer-mediated communication. The contribution of Weblogs to the Semantic Web comes from the design of the supporting software to automatically generate RDF/XML files including RSS feeds, and now FOAF.

The popularity of Weblog hosting sites, and their ability to automatically generate RDF, has had a large impact on the Semantic Web. Swoogle (swoogle.umbc.edu), at present the largest fully automatic Semantic Web document aggregator, currently lists 19 large Web-logging sites in its top-50 index of sites with Semantic Web content (LiveDoor is top-ranked with 9,473 documents in Swoogle and LiveJournal second with 7,690), and collectively blogging sites are responsible for 45% of the Semantic Web documents collected by Swoogle. Even from these sites, Swoogle crawls only a small fraction of the available FOAF documents: LiveJournal automatically generates FOAF files for each of its 6.7 million users, several times more than the number of documents in Swoogle.

The quantitative predominance of FOAF suggests that potential Semantic Web applications need to consider what kinds of additional utility FOAF can offer. On

blogging sites, FOAF supplements syndication metadata (author, title, topic, date, etc.) provided in RSS 1.0 with further detail about the authors of posts (interests, instant messaging IDs, contact information, etc.). FOAF is flexible enough, however, to be used in the context of social networking sites (Friendster, Orkut, etc.), where users post information about their relationships to other people they know, as an aid to finding new social contacts, jobs, life partners, etc. (see Boyd, 2004, for further details regarding Friendster). It is not clear what effect these sites are having, or whether they are beneficial to their users, but their immense popularity (all of the sites mentioned have millions of users) suggests many users do find them beneficial for some purpose. Since the application of FOAF is new, it is also likely that the information it encodes is not being used to its greatest potential. Hence, existing FOAF documents are a good place to learn about the possible effects of blogging and social network software sites, and to start to uncover their latent utility through the Semantic Web metadata they produce.

The scale of the information available in FOAF makes it challenging to work with. Past work on visualizing FOAF focuses on exploring networks on the level of individual actors (Mika, 2004; Mika and Gangemi, 2004). Early examples of such work could employ all of the FOAF then in existence (Dan Brickley, personal communication, September 2004). This is no longer true with the large-scale social networking and blogging sites that are now using FOAF. Moreover, the logic-based tools envisioned for the Semantic Web (Berners-Lee, 2001; Alferes, et al., 2003) typically have computational complexity issues that prohibit working with large actor networks on anything but the most powerful hardware. Practical use of FOAF will need to be in closer reach of the average user, requiring average hardware to accomplish. What then is the best approach to working with FOAF?

We develop here an approach to visualizing FOAF data that employs techniques of quantitative Social Network Analysis to reveal the workings of a large-scale blogging site, LiveJournal. Our analysis specifically seeks to ascertain if the interests that users express in LiveJournal are useful indicators of their social interactions on the site, as represented by their selection of friends. Information pertaining to interests and friends is extracted from a scutter crawl of FOAF and analyzed quantitatively using Principal Components Analysis to arrive at natural groupings of the users. The information is visualized in a series of reduced sociograms (Scott, 2000; Wasserman and Faust, 1994) and interpreted. Our observations reveal an interesting organization of social life on LiveJournal, and suggest modifications to the user interfaces of social networking sites that would potentially assist users in finding one another. In addition, the FOAF data explored here exemplify the challenges encountered in Semantic Web visualization more generally. The resolutions to the problems presented in this chapter provide a model for other practical applications of Semantic Web visualization.

14.2 XML, the Semantic Web, and FOAF

Sir Tim Berners-Lee, original architect and visionary of the Semantic Web, proposed a "stack" of technologies (Berners-Lee, 2001) in order to enable his vision of knowledge management through hypertext. By the mid- to late 1990s it was apparent that the World Wide Web was a success, though a chaotic one, and occasionally a syntactic nightmare. In that context, the XML project (for "eXtensible Markup Language") was initiated to provide an extensible, machine-interpretable language for storing,

communicating, and interpreting information. Since 1998, work done in XML has largely coalesced around using RDF (Resource Description Format) as a means of enabling metadata interoperability and compatibility. RDF, in a nutshell, is a language for defining metadata vocabularies. Items that have their metadata marked up using RDF may compatibly include terms from any of a variety of XML vocabularies. It is not necessary for applications to know in advance which vocabularies will be encountered, or which items may occur; the metadata defined in RDF can still be used to make inferences.

Within RDF, the "FOAF" (friend-of-a-friend) initiative undertaken by the World Wide Web Consortium (W3C) has focused on developing ways to describe both the properties of human beings—date of birth, age, real name, nicknames, contact information of various sorts—and their social relationships as expressed through their interests, group affiliations, common haunts, places of employment, etc. Socially, the most interesting relationship encoded in FOAF is foaf:knows—the notion that one person "knows" another. This relationship is rather coarse, in that the properties or the quality of the relationship are not necessarily expressed. In addition, foaf:knows is unidirectional rather than bidirectional. Individuals sometimes assert that they "know" someone who would not necessarily reciprocate the assertion. Such self-reported, directed social relations are commonly employed in social network analysis research, as they are rich in information about people's underlying social relationships, and many good methods are available for their analysis (Scott, 2000; Wasserman and Faust, 1994).

Typically, users of the FOAF schema would create a file that contains personal data, including e-mail address, location, interests, a list of friends, etc., and place that file in a Web-accessible location. Early FOAF data were produced by hand, usually by users interested in the technology. The early FOAF implementations were often brittle, due to nonstandard tag use and occasional lack of clarity in the specifications. The working schema of FOAF was extended several times to handle new problems, or when someone arrived at a new, useful conceptualization of how people could manage and share social metadata. More recently, large blogging and social networking sites have begun automatically generating FOAF directly from users' profiles stored in their databases, and typically export them at automatically generated addresses. Users generally edit their profiles using Web-based forms. Hence, most users of FOAF today are completely unaware of the technology's existence and their use of it.

FOAF files are sometimes indexed in what is called a "scutter plan," typically a Wiki page containing the URLs of a large collection of FOAF URLs. Networks of FOAF-encoded actors can then be created by using a scutter, or RDF crawler/spider, to follow the network of foaf:knows tags and store the associated files for later analysis. A scutter plan works reasonably well with small quantities of data, as its scope is restricted enough to be managed with limited resources.

The issues involving the scale of FOAF data are exemplified by the data we obtained for our analysis. Our data come from a scutter dump collected by Jim Ley (jibbering.com) containing approximately 700 MB of parsed RDF files harvested between March 3 and March 7, 2004, yielding more than 6.5 million RDF triples. We obtained the data as both raw RDF files and a 1GB MySQL database dump, which was modified for import into PostgreSQL. The dump consists of two main tables. The first of these is a comprehensive table of all of the RDF triples obtained; the second is a list of all of the URLs of FOAF files known to the scutter (including many that have not yet been retrieved), with information about the processing status of the file, the local cache name of the file, and a reference number used to identify the files that are the

Table 14.1 Distribution of URLs in the scutter dump

	LiveJournal		Other	
Date	Visited	Not Visited	Visited	Not Visited
March 3	663	0	1607	0
March 4	13940	121408	160	23
March 5	2810	17648	1279	130
March 6	11782	60844	0	6
March 7	4347	22650	0	1

sources of individual triples. Table 14.1 summarizes the distribution of the 259,298 records in the URL table in terms of the operation of the scutter.

The number of LiveJournal FOAF files visited during scuttering was more than 10 times larger than those from other sites. As the scuttering progressed the number of non-LiveJournal files visited dwindled to zero. This indicates a typical problem in using scuttering methods to characterize FOAF and other Semantic Web data generally. LiveJournal's design encourages users to elect friends who are also LiveJournal users to the exclusion of those who are not. In fact, before its user data were exposed as FOAF, LiveJournal did not permit people to designate friends that were not also LiveJournal users. Hence, a large social networking site like LiveJournal presents a kind of "black hole" from which a scutter will have little chance of escaping. Hence, scutter crawls are not representative samples of data in a statistical sense. In addition, there is a substantial social consequence, since we can see that large social networking sites effectively control their members' social capital. Truly free association is not possible if the representation of the social sphere is dictated by interests from outside the people they concern.

Scale is also an issue for the analysis of FOAF data. Typically, social network analysis requires that one construct an actor-actor matrix. For just the LiveJournal site, which has roughly 4.5 million actors, the complete sociomatrix would require 4.5 m $\times$ 4.5 m cells, or 20.5 terabytes of storage, just for the link structure of the network. For our scutter dump, there are 274,305 distinct actors, yielding a sociomatrix of more than 65 billion cells. When we consider that we want to store other data, such as the interests of the actors, etc., storage requirements climb even higher.

14.3 Analyzing LiveJournal FOAF

The set of FOAF data we have obtained is too large to visualize without further reduction. Consequently, we adopt a statistical approach to the analysis of the data, using Principal Components Analysis (PCA) and Hierarchical Cluster Analysis (HCA) as implemented in the statistical programming language and environment known as "R" (www.r-project.org). These techniques have a computational complexity that scales well enough to permit visualization of the global social patterns on modest dual-processor commodity machines. In addition, the analyses yield information that facilitates detailed exploration of the data down to the level of individual metadata elements. A summary of the FOAF namespace usage appears in Table 14.2.

Our analysis begins with the extraction of the relations of interest from the PostgreSQL database of RDF triples. This is accomplished in two steps. We first used a

Table 14.2 Counts of FOAF predicate usage from the scutter sample

Count	Predicate	Count	Predicate
1067568	interest	1016	surname
869916	nick	749	depicts
868538	weblog	685	jabberID
839934	knows	674	maker
40066	mbox_sha1sum	561	title
33172	page	452	workplaceHomepage
26385	dateOfBirth	403	schoolHomepage
15624	homepage	273	codepiction
13255	aimChatID	238	currentProject
6718	yahooChatID	235	img
6072	name	203	gender
4973	msnChatID	203	phone
4275	icqChatID	195	made
1603	depiction	195	regionDepicts
1436	mbox	143	workInfoHomepage
1416	thumbnail	109	lastName
1089	firstName	1213	77 items occurring 100 times or less

partial string-match to identify all triples with a predicate from the FOAF namespace (http://xmlns.com/foaf/0.1/) into a new table. Since many FOAF files use Dublin Core elements (e.g., for the interests), we did the same thing with the Dublin Core relations as well. This resulted in two tables with 3.81 million and 1.16 million triples respectively. We then tabulated the frequencies of each of the elements in the FOAF namespace to identify elements of interest for further examination. The most frequent FOAF elements in our data were the foaf:interest relation (1.06 million triples) and the foaf:knows relation (839,934 triples). Users' nicknames (foaf:nick, 869,916 triples) were used to resolve participant identities, and dc:title relations (1.11 million triples) were used to identify the object of the foaf:interest relation, as this was the strategy employed for identifying interests in LiveJournal; database JOIN operations were used accomplish these steps. The modified relations were exported as two-column text files for import into the R statistical computing environment.

Within R, both relations were reconstituted as binary incidence matrices. For the foaf:knows relation, we identified the 200 most common objects, and constructed an incidence matrix involving those and the 17,305 distinct subjects they occurred with. For the foaf:interests relation, we identified the 500 most frequent interests and tabulated their occurrence with 21,506 distinct subjects. Both incidence matrices were subjected to a column-wise z-score transformation, the effect of which is to dampen the influence of very frequent foaf:interests or foaf:knows objects. These transformed matrices were the input to PCA.

PCA was performed using the LAPACK functions for Singular Value Decomposition (SVD) available in R through the function `svd()`. The output of SVD is a pair of orthonormal matrices, representing the projections of the rows and columns of the original data into the space of its principal components, and a vector of singular values, which are weights that can be used in a matrix computation to reconstruct the original matrix or principal components scores, as desired (Basilevsky, 1994, Chapter 3). The SVD can be written in matrix notation as in the equation:

$$A = P(\lambda I)Q^T$$

where A is the original matrix, $= P$ and Q are the orthonormal matrices representing the row and column associations with the principal components respectively, and λ is the vector of singular values. P and Q must have the same number of columns r as the number of singular values retained in λ. I is an r-by-r identity matrix that permits P and Q to be reassembled into A as a matrix multiplication.

The utility of PCA is that it permits us to characterize the major dimensions of variation in the data, as determined by the correlations among foaf:interests and foaf:knows across the set of all actors. The vector λ is ordered by the size of the principal components (where size corresponds to the proportion of the variance associated with the component), and their respective values can be compared to identify a suitable number of dimensions that retains important variation in the data; methods for doing this include Bayesian, Markov Chain, Monte Carlo, and other computationally sophisticated methods (Basilevsky 1994, Chapter 4); we opted for the more heuristic "scree-plot" as a computationally tractable alternative; the scree plots suggest in the neighborhood of 10 to 20 principal components can be retained in both cases.

Once the principal components are found, we can calculate principal components scores for each of the columns or rows of the original data. These scores are useful in two ways. First, they can be used as input to Hierarchical Cluster Analysis (HCA) in order to identify groups of interests or users sharing similar behavior. Second, they can be plotted for visual inspection and interpretation. We proceeded by clustering the data before plotting. The projection of the data into two dimensions tends to place many unrelated points near each other, and this can be diagnosed if the clusters are color-coded in the plot. Figures 14.1 and 14.2 provide examples of this, where HCA was performed on the column scores of the foaf:interests data, to allow meaningful

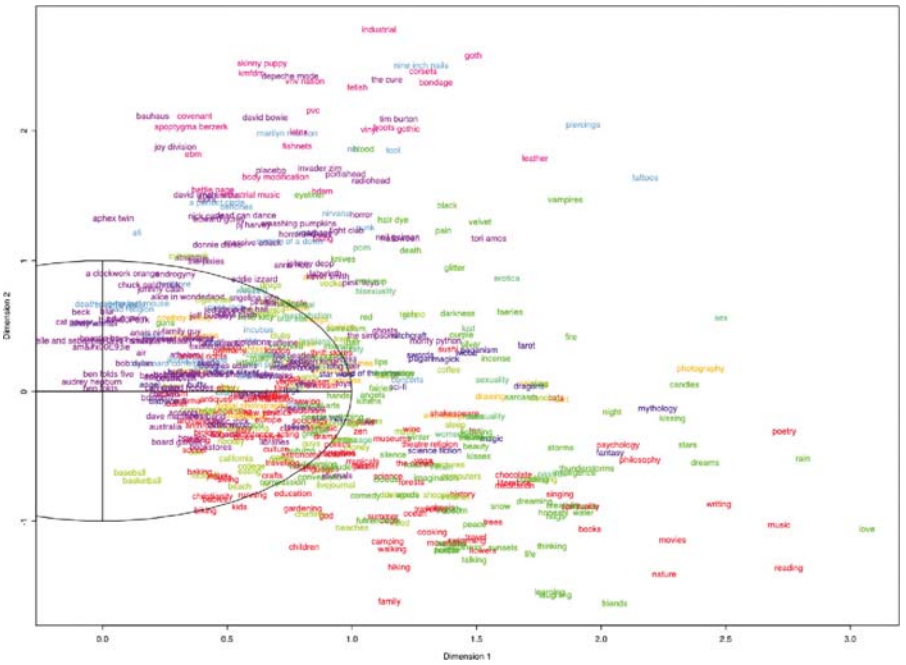


Figure 14.1 Plot of nine interest clusters on the first two principal components.

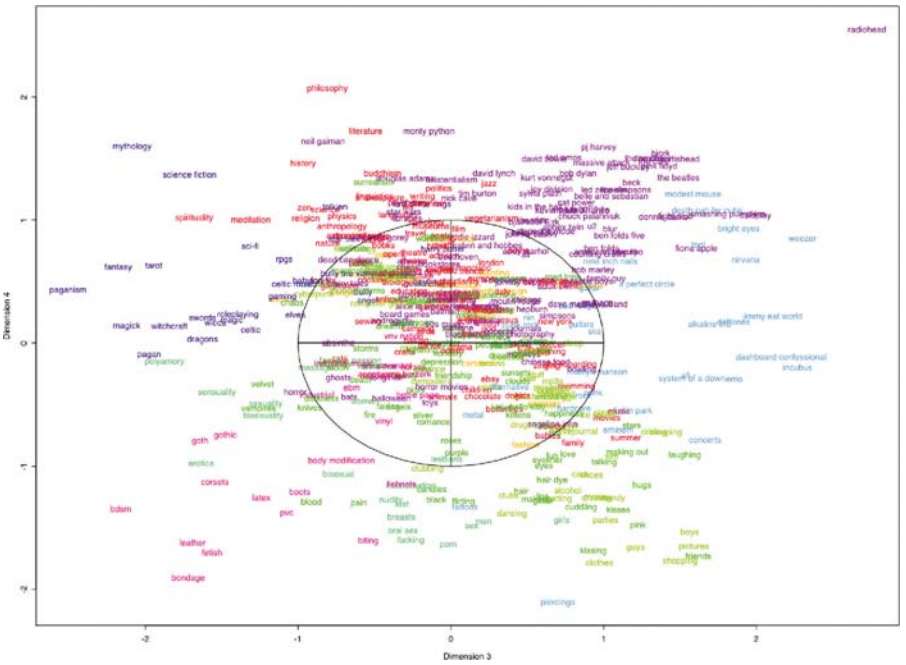


Figure 14.2 Plot of nine interest clusters on principal components 3 and 4.

clusters of the 500 interests to be viewed and interpreted. A cut of nine clusters (the largest number that can be practically recognized from a rainbow palette) was selected and plotted on the first four principal components (dimensions 1 and 2 in Figure 14.1, and dimensions 3 and 4 in Figure 14.2). The data can be explored by plotting them on different principal components, as different clusters and subclusters of interest are revealed. On each of our plots, we have added a unit circle and axes to clearly indicate the location of the origin in each plot.

In PCA, the first principal component tends to be sensitive to overall frequency; the same is true of the second principal component, usually to a lesser extent. Subsequent principal components are more reliable indicators of correlations among the interests in the incidence matrix. In the interest data, the most frequent interests load high on the first principal component. Music, being the most frequently expressed interest, is on the extreme right, as are art, books, movies, photography, reading, cats, computers, writing, and love, which are also very frequently indicated interests. The positive direction of the second principal component exhibits a large number of specific music interests (industrial, nine-inch nails, skinny puppy, marylin manson, radiohead, etc.).

Dimensions 3 and 4 exhibit similar patterns of differentiation. The right-hand-side interests concern aesthetic sensibilities (e.g., musical tastes, in the light blue and violet clusters) and general sociability, especially romance (boys, guys, kissing, cuddling, shopping, etc., in the green and yellow-green clusters). The left-hand side concerns fantasy (science fiction, paganism), philosophy (spirituality, tarot, meditation), and sexual fantasy/fetish (bondage, erotica, polyamory, etc.). Vertically, there is a split between body/sexual and romantic interests (bottom) and intellectual/aesthetic interests (top). The scatterplot reads like a map for these particular forms of user interest.

The cluster analysis also reveals useful information for interpretation. The nine clusters in our chosen cut can be roughly characterized as general interests (red), art (orange), nightlife (yellow-green), romance (green), sexuality (aqua), grunge music, tattoos, and piercings (sky blue), science fiction/fantasy (blue), music (violet), and goth subculture and sexual fetish (pink). Goth subculture is a primarily North American counterculture movement distinct from punk, which emphasizes morbidity, death, and to some extent androgyny. Goths, both female and male, also typically put great care into their appearance, spending hours dressing and applying makeup before appearing in public.

The clusters are not entirely pure (e.g., kurt vonnegut is found in the music cluster, etc.) but the labels are general characterizations based on an inspection of the complete cluster hierarchy that can be useful in developing interpretations of the data. Clustering at a lower level in the hierarchy can also be inspected to reveal patterns of interest. For example, god, Christianity, and jesus form a cluster together within the red cluster of general interests; this is certainly a coherent group of interests. Its closest neighbor is a cluster consisting of two subclusters: (a) religion, philosophy, psychology, sociology, spirituality, meditation, buddhism, zen, and (b) animal rights, feminism, activism, yoga, sewing, knitting, vegetarianism, and thrift stores. Again, an overall pattern of coherence can be identified that matches subcultural social patterns in offline behavior.

Clusters of actors on principal components scores from the foaf:knows incidence matrix indicate social positions, or equivalence relations among actors with common relations to other actors in the network. These are useful in examining the social structure of the foaf:knows network to identify patterns of centrality, power, and exchange among users of LiveJournal. Figure 14.3 illustrates this set of patterns at two levels of reduction in the form of a reduced sociogram, where by "reduced" we mean that equivalent actors have been aggregated into a single node for the purposes of display.

Figure 14.3, left, shows nine clusters, which are disaggregated into 77 clusters on the right, keeping similar subclusters together in space and preserving the color-coding of the larger clusters they belong to (note that the spring layout algorithm does not preserve the position of the clusters in general, just their pattern of connectivity). A strong pattern of interaction is evident in both network graphs, in which members of

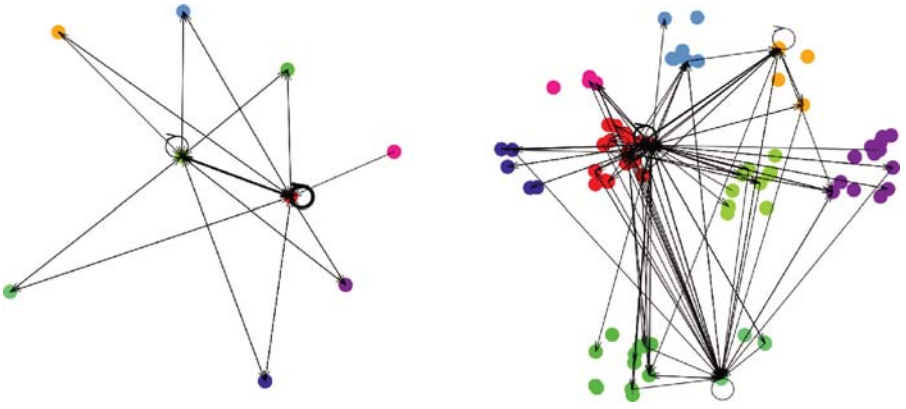


Figure 14.3 Social positions in the foaf:knows relation revealed at different levels of clustering.

the red cluster are seen to be most central: there are strong self-links within it, and all other clusters link strongest to the red cluster as well. In addition, the yellow-green cluster shows fairly strong links to the red cluster in the left-hand side of Figure 14.3, and ties to all of the other clusters as well. However, this pattern of linkage disappears on the right-hand side of Figure 14.3, suggesting that the yellow-green cluster's patterns of relationship are not at the level of group-to-group in the disaggregated clusters. The threshold at which links are displayed is partly responsible for this difference in appearance, but the level of resolution, in terms of numbers of clusters across which links are aggregated, is also probably responsible, as there are many more groups among which an actor's relationships can be scattered. Hence the different levels of resolution reveal different patterns in the social structure of the network. Thus, the yellow-green cluster has a much less tight social organization than the red cluster, and other clusters are peripherally associated with the network-central red and yellow-green clusters.

These observations suggest that LiveJournal has a distinct core of social activity, at least as far as we have sampled it. Other participants are more peripheral. We do not see strong evidence of partitioning of LiveJournal into different camps, although this question bears closer scrutiny. Having found strong patterns of social structure on LiveJournal, we next ask if they are related to the users' expressed interests in any way: do the users' interests assist the social structuring of interaction on LiveJournal? To address this question, we used HCA to identify groups of users based on their principal components scores for the distribution of interests. Once identified, the foaf:knows relations among members of these groups were again aggregated. Our hypothesis is that if the interest groups are coherent social entities, we should see a predominance of self-ties within them, perhaps alongside a more global pattern of relationship and interaction among the interest groups. If not, only a global pattern of interaction would be visible.

Figure 14.4 visualizes, again at two levels of clustering, the social relations of groups of actors with shared interests. Although the palette is the same as for the previous visualizations, the categories and their color-coding are not shared, as they represent clusters based on different information. Color-coding is preserved, as well as general position in space, between the left-hand and right-hand images of Figure 14.4. Again we see a network with a strong central core and a periphery; on the left of Figure 14.4,

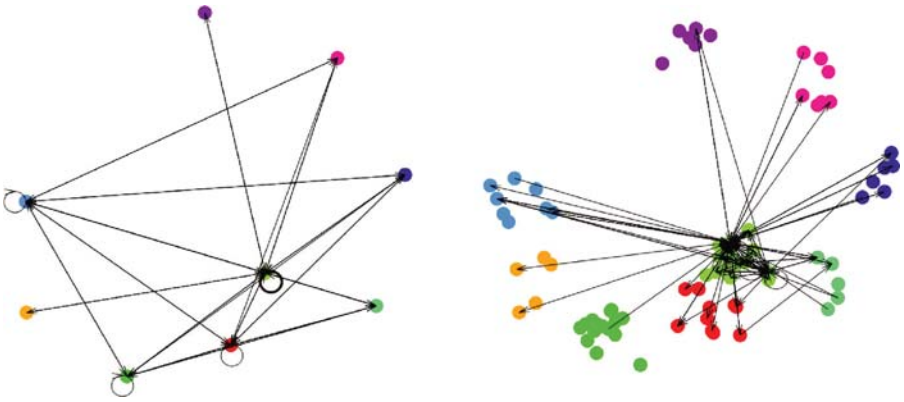


Figure 14.4 Social relations of shared interest groups at two different levels of clustering.

the central group this time is yellow-green, with the strongest self-links and links to all of the other groups. Again there is a second group that is linked above the threshold level to most of the other groups, but the strong reciprocal pattern we saw on the left of Figure 14.3 is not evident. There is clearly different social information in Figures 14.3 and 14.4.

Regarding our hypothesis about the social relations and interests, we find some evidence for cohesion in the diagram on the left of Figure 14.4, in that four out of the nine groups exhibit some degree of self-linkage. This evaporates, however, in the disaggregated groups in the diagram on the right of Figure 14.4, where only the yellow-green groups retain their self-links. Note also that many foaf:knows relations appear to point outward from this central group, and relatively fewer inward. These patterns suggest a highly articulated core of interests around which social cohesion is built among a central set of participants on LiveJournal; other groups with somewhat more diffuse interests, and having lower overall social cohesion, occupy the periphery of the network.

Having established that the social life of LiveJournal can be described meaningfully in terms of the shared interests of its users, the natural question to ask is which interests are shared, and how they contribute to the social structuring of LiveJournal. Here, we are interested if we can identify clusters or subclusters of relations that are associated with any of the different clusters above. To investigate this, we computed centroids of each of 76 interest clusters and each of the 77 interest-based social groups on each of the principal components scores. We then computed Euclidean distances between each interest group and interest cluster pair, from the two sets of factor scores. These distances were visualized as a bimodal network, with the interest groups (people) represented in the same relative positions they occupy in Figure 14.4, and the interest clusters (interests) positioned in a ring outside them. The interest clusters are sorted and color-coded according to the same scheme as in Figures 14.1 and 14.2, so their relationship to these plots can be more easily identified.

This visualization is presented as Figure 14.5, where interpretive labels of the nine interest clusters have been added to assist the viewer. As suggested by our interpretation of Figure 14.4, we note a broad range of interest clusters characterizing the central yellow-green groups, but markedly fewer interests characterizing the peripheral interest groups. Each subgroup within the yellow-green group has somewhat different links to the subclusters of interests, although there is substantial overlap among particular subinterests in the music, general, nightlife, and romance interests. Only a very few of the peripheral interest groups have links of any strength to any specific clusters of interests. Note that this is the first pattern to emerge from the network at any level of link threshold: any threshold low enough to increase the number of links to peripheral groups causes almost all of the interest clusters to be connected to all of the yellow-green subgroups.

These observations confirm our impression that the yellow-green group's interests are more coherent, both in terms of relation to general spheres of interest as well as in social structuring, than the other peripheral interest groups. At the same time, the relations between interest subclusters and subgroups are very highly articulated, such that different subgroups of users can be meaningfully distinguished in term of their interests.

We do not yet know specifically which interests are responsible for which structures, or which users are members of which groups, but this information could be obtained by careful collation of our statistical analyses with the foaf:knows and foaf:interests database tables. In future research, we hope to examine in particular members of the

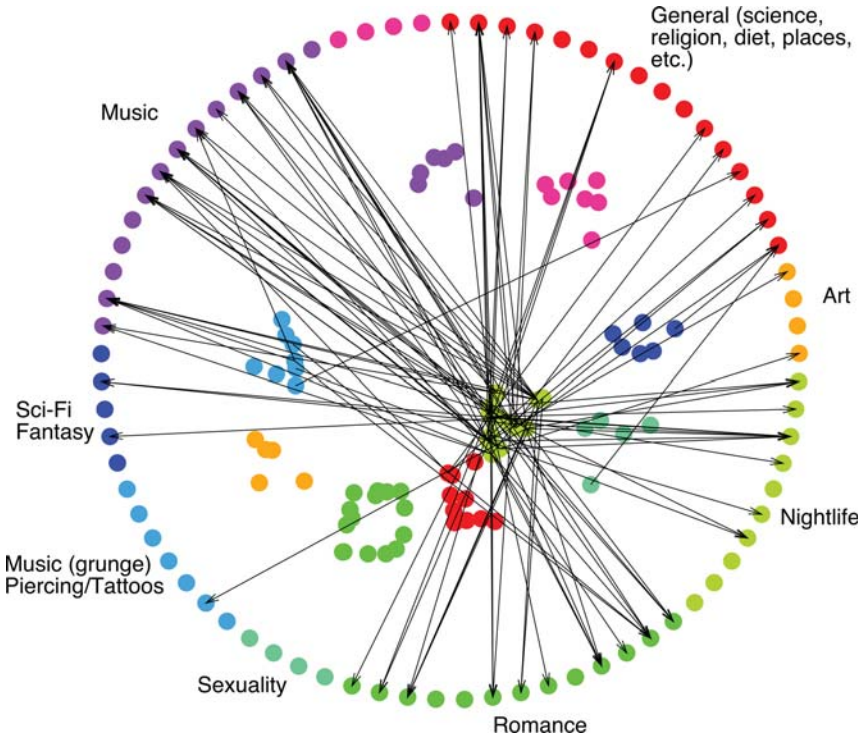


Figure 14.5 Relation of interest clusters to groups of actors with shared interests.

yellow-green group to ascertain if their online behavior comports with and/or further illuminates our findings here.

14.4 Discussion and Conclusions

Our visualizations of Semantic Web social metadata suggest that statistical and quantitative approaches have much to contribute to the understanding of how these new technologies are used, and what sorts of changes might improve them. While Semantic Web metadata are meant to provide standardized ways of annotating information, we see in the application of the FOAF relations studied here a very rich structuring of the data that is not readily captured in ontologies and yet which is very close to the meaning of social life in the online environment of LiveJournal.

The foaf:interest relation is especially instructive in this regard. It is unlikely that a useful ontology of interests could be devised that would express the same social meanings and correlations as the interest clusters discovered here. These clusters are discovered within markup that is essentially uncontrolled—the object of the dc:title relation, which accepts any literal value. Moreover the empirically discovered interest clusters are subject to change, as people reorganize their social relations and reorient themselves to the new social realities they create. An interesting research project in the Semantic Web applications of FOAF would be to try to expose some of this dynamic, changing, and emergent structure for use in inferences. This will require means beyond

the sort of logical deduction envisioned by the architects of the Semantic Web's current form (Berners-Lee, 2001; Alferes, et al., 2003).

Our visualizations also suggest that LiveJournal's user profile interface, which allows users to select their interests in a fairly open-ended way, nonetheless permits users to usefully organize themselves according to their interests. Patterns of interest and user interaction are intricately interwoven, suggesting that the design of the site is at least partly successful in its goal of fostering online, interest-based communities of users.

At the same time, it must be noted that these effects are most pronounced for a central core of the users in our sample; outside of this core, both social relationship and the coherence of interests become more diffuse. Consequently we must regard the vast majority of the ties indicated through foaf:knows or foaf:interests as "weak ties" in the sense of Granovetter (1973). Strong ties, such as those between people who know each other well or who spend a great deal of time together, are only found in the central core of the network. This pattern closely resembles that observed on the Internet Relay Chat (IRC) channel #india (Paolillo 2001), where a predominance of weak ties was also observed. In fact, a central aim of computer-mediated communications systems like Weblogs and IRC is, in essence, to amplify weak ties.

The consequences of this for the structure of social interaction and its potential outcomes bear considering. The nature of social interaction through Weblogs is different from that of face-to-face communication. Since the subscribership of community Weblog and social networking Web sites is booming, this suggests a more general change in the way that social relations are enacted, at least among the users of such sites, if only because of the amount of social interaction they have in the online context as opposed to in more traditional contexts.

Moreover, as is evident from the nature of the expressed interests of the users, the majority of LiveJournal users are young (between the ages of 15 and 30), a life-stage in which people are at their most sociable, and a great deal of social development occurs (Degenne and Forse, 1999; Forse, 1981). It is important to ask what these changes in social behavior might mean for the future of social life—a change in the social development of even a few million youth could have an impact on future trends. Data mining and visualization of the social metadata made available through FOAF is a valuable way in which we can study and begin to understand such issues.

Another possible application of these results would be to use them to inform the design of user interfaces of community Weblogs and social networking sites. User interfaces could present visualizations of a site's social metadata in visualizations similar to those employed here. By manipulating the visualization, its level of resolution, the arrangement of clusters, etc., users would be able to locate themselves within the social life of the site, navigate through it, and find other users or even interests they were not previously aware of by following links among the clusters and groups. While such visualizations are more complex and harder to generate than social visualizations proposed for other communication modes by Viegas and Smith (2004), Donath, et al. (1999), and others, they complement those approaches by providing both detailed and manageable access to the complex link structure of an online social space. Such interfaces might do more to strengthen the ties among members of peripheral groups, whose structuring around interests is currently less coherent than the core.

We must keep in mind, however, that the young users of blogging and social networking sites like LiveJournal are also vulnerable to social manipulation—through sexual harassment, pornography, commercial marketing, surveillance by law enforcement, legal intimidation, and other means—by forces that are arguably in a better position to exploit Semantic Web metadata than the users of the Weblog services that

generate it. The observations made here—ethically permissible as research because of the public nature of the data—could easily be used to target people for marketing purposes (many of the interests have a commercial aspect), for lawsuits about copyright violation (because of associations between certain musical tastes and illegal downloading), or for political surveillance (because of expressed antiwar, pro-drug, or otherwise minority views). Whether our aim is to provide improved communication and community technologies, or to conduct research and advance our knowledge of the interaction of social and technical systems, we must not lose sight of the human experiences behind the statistics and metadata that we employ. It is our responsibility to wield these tools carefully as we explore the world of social metadata through visualization.

14.5 Acknowledgments

The authors would like to thank other members of the Blog Research on Genre project at Indiana University, and participants at the FOAF Workshop, Galway 2004, for their interest and support of this research.

14.6 References

Alferes, J., Damásio, C., Pereira, L. (2003). Semantic Web logic programming tools. *Lecture Notes in Computer Science,* 2901:16–32.

Basilevsky, A. (1994). *Statistical Factor Analysis and Related Methods: Theory and Application.* New York: Wiley.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American,* 284, May: 34–43.

Boyd, D. (2004). Friendster and publicly articulated social networks. *Proceedings of CHI 2004, Connect: Conference on Human Factors and Computing Systems.* ACM Press, New York.

Brickley, D., Miller, L. (2003). FOAF vocabulary specification. Technical report, RDFWeb FOAF Project.

Degenne, A., Forse, M. (1999). *Introducing Social Networks.* Thousand Oaks, CA: Sage Publications.

Donath, J., Karahalios, K., Viegas, F. (1999). Visualizing conversation. *Proceedings of the 32nd Hawaii International Conference on System Sciences.* IEEE Computer Society, Los Alamitos, California.

Forse, M. (1981). La sociabilite. *Economie et Statistique,* 132:39–48.

Golbeck, J., Hendler, J. (2004). Inferring reputation on the Semantic Web. *Proceedings of the 13th International World Wide Web Conference (WWW2004).* ACM Press, New York.

Granovetter, M. (1973). The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380.

Herring, S., Scheidt, L., Bonus, S., Wright, E. (2004). Bridging the gap: A genre analysis of Weblogs. *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37).* IEEE Computer Society, Los Alamitos, California.

Mika, P. (2004). Bootstrapping the FOAF-Web: An experiment in social network mining. *First International Workshop on FOAF, Social Networks, and the Semantic Web*, Galway, Ireland.

Mika, P., Gangemi, A. (2004). Descriptions of social relation. *First International Workshop on FOAF, Social Networks, and the Semantic Web*, Galway, Ireland.

Paolillo, J. (2001). Language variation on Internet Relay Chat: A social network approach. *Journal of Sociolinguistics*, 5:180–213.

Scott, J. (2000). *Social Network Analysis: A Handbook* (2d Edition). Thousand Oaks, CA: Sage Publications.

Viegas, F., Smith, M. (2004). Newsgroup crowds and authorlines: Visualizing the activity of individuals in conversational cyberspace. *Proceedings of the 37th Hawaii International Conference on System Sciences.* IEEE Computer Society, Los Alamitos, California.

Wasserman, S., Faust, K. (1994). *Social Network Analysis: Methods and Applications.* Cambridge: Cambridge University Press.

Chapter 15

Concluding Remarks: Today's Vision of Envisioning the Semantic Future

Vladimir Geroimenko and Chaomei Chen

In the second edition of this book, we have explored some significant topics in the emerging research area of visualizations for the Semantic Web. A new version of the Web is a vision that is incredibly quickly becoming a reality. Visual techniques and metaphors that the second-generation Web is bringing to life depend very much on its future implementations and building blocks. No one can predict how the real-life development of the Semantic Web will turn out, and what novel visualization techniques will be needed. Because of this, drawing a more or less detailed picture of the new research area will be possible only as a result of several years of thorough exploration, and we hope that with our pioneering book we help to make it happen sooner. As with any other novel area, we were trying our best to investigate the topics that seem to be of primary importance at this moment in time. We think that in general we have achieved our goal and that the reader will not judge us too severely for not presenting those aspects that could not be covered today because they require research that can be done only after the main structure of the Semantic Web has been implemented.

The next-generation Web will be rich both semantically and visually. Since XML and related technologies separate content from presentation rules, any form of presentation has "equal opportunities" and therefore visually rich ones have a better chance of being implemented in semantic human-computer interfaces. The intensive use of semantics will change the nature of the visual aspect of the Web. In addition to today's images, pictures, and animations, the future Web will enable the user to visualize far more conceptual things such as the meaning and structure of Web data. The next generation will be not only the Web of formalized meanings, understandable by machines, but also the Web of visualized meanings that increase human understanding of Web data. In other words, computers will be able to comprehend the future Web because it includes machine-processable semantics; humans can better comprehend this Web because of the use of visualized semantics.

Visually rich 2D and 3D worlds of meanings, common to the Semantic Web, will comprise a variety of different levels and forms of metadata—from simple XML vocabularies to complex OWL ontologies. They will allow the user to interact with data and metadata in an efficient and desirable way (for example, navigate, manipulate, transform, zoom, filter, etc.). If a picture is worth a thousand words, a semantic visualization is worth a thousand metadata tags. Visual interaction with Web data, metadata, schemas, and ontologies will be effectively used for a wide variety of purposes such

as searching the Web using semantically enhanced search engines, customising Web pages and sites on the fly, instructing autonomous software agents, navigating through huge sets of data and metadata, accessing any part of any Web document, and so on.

The Semantic Web will also change the nature of images that we are used to seeing on the current Web. Since they will be presented in SVG, X3D, or other XML-based formats, it will be possible to put them to an active use, similar to any other XML documents. This means that computers will be able to understand the meaning of not only Web data but also Web images. Both computers and humans can access any parts of such images, "recognize" them by their tags, find them on the entire Web, compose new images on the fly using XSLT, and so forth. For example, a software agent will be able to find an appropriate map of a city you plan to visit, and then put on it all images of your places of interest, link them to available Web resources, and so on. Moreover, it can refine an existing X3D virtual model of the city in question by leaving only the streets you have to use, and by putting more visual signs and aural clues. Our book *Visualising Information Using SVG and X3D: XML-Based Technologies for the XML-Based Web* is a conceptual sequel to the first edition of this book. It was published in 2004 by Springer (ISBN: 1-85233-790-7) and is the first research monograph that explores the current and future applications of the new-generation Web graphics.

As of today, the future of the Semantic Web looks extremely bright. It will be a Web of meanings. These meanings will be formalized in a smart way to make them automatically processable by machines. The same meanings will be very often visualized in an effective way to help humans to comprehend them. Such a Web will make sense.

Index