The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring.

At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating* of *excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules.

ILLUSTRATIONS:
Software Agents

Web Searches Today

Semantic Web Searches

FURTHER INFORMATION

(The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Almost instantly the new plan was presented: a much closer clinic and earlier times—but there were two warning notes. First, Pete would have to reschedule a couple of his *less important* appointments. He checked what they were—not a problem. The other was something about the insurance company's list failing to include this provider under *physical therapists*: "Service type and insurance plan status securely verified by other means," the agent reassured him. "(Details?)"

Lucy registered her assent at about the same moment Pete was muttering, "Spare me the details," and it was all set. (Of course, Pete couldn't resist the details and later that night had his agent explain how it had found that provider even though it wasn't on the proper list.)

Expressing Meaning

Pete and Lucy could use their agents to carry out all these tasks thanks not to the World Wide Web of today but rather the Semantic Web that it will evolve into tomorrow. Most of the Web's content today is designed for humans to read, not for computer programs to manipulate meaningfully. Computers can adeptly parse Web pages for layout and routine processing—here a header, there a link to another page—but in general, computers have no reliable way to process the semantics: this is the home page of the Hartman and Strauss Physio Clinic, this link goes to Dr. Hartman's curriculum vitae.

The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users. Such an agent coming to the clinic's Web page will know not just that the page has keywords such as "treatment, medicine, physical, therapy" (as might be encoded today) but also that Dr. Hartman *works* at this *clinic* on *Mondays*, *Wednesdays* and *Fridays* and that the script takes a *date range* in *yyyy-mm-dd format* and returns *appointment times*. And it will

"know" all this without needing artificial intelligence on the scale of 2001's Hal or Star Wars's C-3PO. Instead these semantics were encoded into the Web page when the clinic's office manager (who never took Comp Sci 101) massaged it into shape using off-the-shelf software for writing Semantic Web pages along with resources listed on the Physical Therapy Association's site.

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present.

The essential property of the World Wide Web is its universality. The power of a hypertext link is that "anything can link to anything." Web technology, therefore, must not discriminate between the scribbled draft and the polished performance, between commercial and academic information, or among cultures, languages, media and so on. Information varies along many axes. One of these is the difference between information produced primarily for human consumption and that produced mainly for machines. At one end of the scale we have everything from the five-second TV commercial to poetry. At the other end we have databases, programs and sensor output. To date, the Web has developed most rapidly as a medium of documents for people rather than for data and information that can be processed automatically. The Semantic Web aims to make up for this.

Like the Internet, the Semantic Web will be as decentralized as possible. Such Web-like systems generate a lot of excitement at every level, from major corporation to individual user, and provide benefits that are hard or impossible to predict in advance. Decentralization requires compromises: the Web had to throw away the ideal of total consistency of all of its interconnections, ushering in the infamous message "Error 404: Not Found" but allowing unchecked exponential growth.

Knowledge Representation

For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Artificial-intelligence researchers have studied such systems since long before the Web was developed. Knowledge representation, as this technology is often called, is currently in a state comparable to that of hypertext before the advent of the Web: it is clearly a

good idea, and some very nice demonstrations exist, but it has not yet changed the world. It contains the seeds of important applications, but to realize its full potential it must be linked into a single global system.

Traditional knowledge-representation systems typically have been centralized, requiring everyone to share exactly the same definition of common concepts such as "parent" or "vehicle." But central control is stifling, and increasing the size and scope of such a system rapidly becomes unmanageable.

WEB SEARCHES TODAY

Moreover, these systems usually carefully limit the questions that can be asked so that the computer can answer reliably— or answer at all. The problem is reminiscent of Gödel's theorem from mathematics: any system that is complex enough to be useful also encompasses unanswerable questions, much like sophisticated versions of the basic paradox "This sentence is false." To avoid such problems, traditional knowledge-representation systems generally each had their own narrow and idiosyncratic set of rules for making inferences about their data. For example, a genealogy system, acting on a database of family trees, might include the rule "a wife of an uncle is an aunt." Even if the data could be transferred from one system to another, the rules, existing in a completely different form, usually could not.

Semantic Web researchers, in contrast, accept that paradoxes and unanswerable questions are a price that must be paid to achieve versatility. We make the language for the rules as expressive as needed to allow the Web to reason as widely as desired. This philosophy is similar to that of the conventional Web: early in the Web's development, detractors pointed out that it could never be a well-organized library; without a central database and tree structure, one would never be sure of finding everything. They were right. But the expressive power of the system made vast amounts of information available, and search engines (which would have seemed quite impractical a decade ago) now produce remarkably complete indices of a lot of the material out there. The challenge of the Semantic Web, therefore, is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported onto the Web.

Adding logic to the Web—the means to use rules to make inferences, choose courses of action and answer questions—is the task before the Semantic Web community at the moment. A mixture of mathematical and engineering decisions complicate this task. The logic must be powerful

enough to describe complex properties of objects but not so powerful that agents can be tricked by being asked to consider a paradox. Fortunately, a large majority of the information we want to express is along the lines of "a hex-head bolt is a type of machine bolt," which is readily written in existing languages with a little extra vocabulary.

Two important technologies for developing the Semantic Web are already in place: eXtensible Markup Language (XML) and the Resource Description Framework (RDF). XML lets everyone create their own tags—hidden labels such as <zip code> or <alma mater> that annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the script writer has to know what the page writer uses each tag for. In short, XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean [see "XML and the Second-Generation Web," by Jon Bosak and Tim Bray; Scientific American, May 1999].

The Semantic Web will enable machines to COMPREHEND semantic documents and data, not human speech and writings.

Meaning is expressed by RDF, which encodes it in sets of triples, each triple being rather like the subject, verb and object of an elementary sentence. These triples can be written using XML tags. In RDF, a document makes assertions that particular things (people, Web pages or whatever) have properties (such as "is a sister of," "is the author of") with certain values (another person, another Web page). This structure turns out to be a natural way to describe the vast majority of the data processed by machines. Subject and object are each identified by a Universal Resource Identifier (URI), just as used in a link on a Web page. (URLs, Uniform Resource Locators, are the most common type of URI.) The verbs are also identified by URIs, which enables anyone to define a new concept, a new verb, just by defining a URI for it somewhere on the Web.

Human language thrives when using the same term to mean somewhat different things, but automation does not. Imagine that I hire a clown messenger service to deliver balloons to my customers on their birthdays. Unfortunately, the service transfers the addresses from my database to its database, not knowing that the "addresses" in mine are where bills are sent and that many of them are post office boxes. My hired clowns end up entertaining a number of postal workers—not necessarily a bad thing but certainly not the intended effect. Using a different URI for each specific

concept solves that problem. An address that is a mailing address can be distinguished from one that is a street address, and both can be distinguished from an address that is a speech.

The triples of RDF form webs of information about related things. Because RDF uses URIs to encode this information in a document, the URIs ensure that concepts are not just words in a document but are tied to a unique definition that everyone can find on the Web. For example, imagine that we have access to a variety of databases with information about people, including their addresses. If we want to find people living in a specific zip code, we need to know which fields in each database represent names and which represent zip codes. RDF can specify that "(field 5 in database A) (is a field of type) (zip code)," using URIs rather than phrases for each term.

Ontologies

Of course, this is not the end of the story, because two databases may use different identifiers for what is in fact the same concept, such as *zip code*. A program that wants to compare or combine information across the two databases has to know that these two terms are being used to mean the same thing. Ideally, the program must have a way to discover such common meanings for whatever databases it encounters.

A solution to this problem is provided by the third basic component of the Semantic Web, collections of information called ontologies. In philosophy, an ontology is a theory about the nature of existence, of what types of things exist; ontology as a discipline studies such theories. 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.

The taxonomy defines classes of objects and relations among them. For example, an *address* may be defined as a type of *location*, and *city codes* may be defined to apply only to *locations*, and so on. Classes, subclasses and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. If *city codes* must be of type *city* and cities generally have Web sites, we can discuss the Web site associated with a *city code* even if no database links a city code directly to a Web site.

Inference rules in ontologies supply further power. An ontology may express the rule "If a city code is associated with a state code, and an

address uses that city code, then that address has the associated state code." A program could then readily deduce, for instance, that a Cornell University address, being in Ithaca, must be in New York State, which is in the U.S., and therefore should be formatted to U.S. standards. The computer doesn't truly "understand" any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.

With ontology pages on the Web, solutions to terminology (and other) problems begin to emerge. The meaning of terms or XML codes used on a Web page can be defined by pointers from the page to an ontology. Of course, the same problems as before now arise if I point to an ontology that defines *addresses* as containing a *zip code* and you point to one that uses *postal code*. This kind of confusion can be resolved if ontologies (or other Web services) provide equivalence relations: one or both of our ontologies may contain the information that my zip code is equivalent to your postal code.

Our scheme for sending in the clowns to entertain my customers is partially solved when the two databases point to different definitions of *address*. The program, using distinct URIs for different concepts of address, will not confuse them and in fact will need to discover that the concepts are related at all. The program could then use a service that takes a list of *postal addresses* (defined in the first ontology) and converts it into a list of physical *addresses* (the second ontology) by recognizing and removing post office boxes and other unsuitable addresses. The structure and semantics provided by ontologies make it easier for an entrepreneur to provide such a service and can make its use completely transparent.

Ontologies can enhance the functioning of the Web in many ways. They can be used in a simple fashion to improve the accuracy of Web searches—the search program can look for only those pages that refer to a precise concept instead of all the ones using ambiguous keywords. More advanced applications will use ontologies to relate the information on a page to the associated knowledge structures and inference rules. An example of a page marked up for such use is online at http://www.cs.umd.edu/~hendler. If you send your Web browser to that page, you will see the normal Web page entitled "Dr. James A. Hendler." As a human, you can readily find the link to a short biographical note and read there that Hendler received his Ph.D. from Brown University. A computer program trying to find such information, however, would have to be very complex to guess that this information might be in a biography and to understand the English language used there.

For computers, the page is linked to an ontology page that defines

information about computer science departments. For instance, professors work at universities and they generally have doctorates. Further markup on the page (not displayed by the typical Web browser) uses the ontology's concepts to specify that Hendler received his Ph.D. from the entity described at the URI http://www. brown.edu — the Web page for Brown. Computers can also find that Hendler is a member of a particular research project, has a particular e-mail address, and so on. All that information is readily processed by a computer and could be used to answer queries (such as where Dr. Hendler received his degree) that currently would require a human to sift through the content of various pages turned up by a search engine.

In addition, this markup makes it much easier to develop programs that can tackle complicated questions whose answers do not reside on a single Web page. Suppose you wish to find the Ms. Cook you met at a trade conference last year. You don't remember her first name, but you remember that she worked for one of your clients and that her son was a student at your alma mater. An intelligent search program can sift through all the pages of people whose name is "Cook" (sidestepping all the pages relating to cooks, cooking, the Cook Islands and so forth), find the ones that mention working for a company that's on your list of clients and follow links to Web pages of their children to track down if any are in school at the right place.

Agents

AGENTS

The real power of the Semantic Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services (including other agents) become available. The Semantic Web promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data come with semantics.

An important facet of agents' functioning will be the exchange of "proofs" written in the Semantic Web's unifying language (the language that expresses logical inferences made using rules and information such as those specified by ontologies). For example, suppose Ms. Cook's contact information has been located by an online service, and to your great surprise it places her in Johannesburg. Naturally, you want to check this, so your computer asks the service for a proof of its answer, which it promptly provides by translating its internal reasoning into the Semantic Web's unifying language. An inference engine in your computer readily verifies

that this Ms. Cook indeed matches the one you were seeking, and it can show you the relevant Web pages if you still have doubts. Although they are still far from plumbing the depths of the Semantic Web's potential, some programs can already exchange proofs in this way, using the current preliminary versions of the unifying language.

Another vital feature will be digital signatures, which are encrypted blocks of data that computers and agents can use to verify that the attached information has been provided by a specific trusted source. You want to be quite sure that a statement sent to your accounting program that you owe money to an online retailer is not a forgery generated by the computer-savvy teenager next door. Agents should be skeptical of assertions that they read on the Semantic Web until they have checked the sources of information. (We wish more *people* would learn to do this on the Web as it is!)

Many automated Web-based services already exist without semantics, but other programs such as agents have no way to locate one that will perform a specific function. This process, called service discovery, can happen only when there is a common language to describe a service in a way that lets other agents "understand" both the function offered and how to take advantage of it. Services and agents can advertise their function by, for example, depositing such descriptions in directories analogous to the Yellow Pages.

Some low-level service-discovery schemes are currently available, such as Microsoft's Universal Plug and Play, which focuses on connecting different types of devices, and Sun Microsystems's Jini, which aims to connect services. These initiatives, however, attack the problem at a structural or syntactic level and rely heavily on standardization of a predetermined set of functionality descriptions. Standardization can only go so far, because we can't anticipate all possible future needs.

The Semantic Web, in contrast, is more flexible. The consumer and producer agents can reach a shared understanding by exchanging ontologies, which provide the vocabulary needed for discussion. Agents can even "bootstrap" new reasoning capabilities when they discover new ontologies. Semantics also makes it easier to take advantage of a service that only partially matches a request.

Properly designed, the Semantic Web can assist the evolution of human knowledge as a whole.

A typical process will involve the creation of a "value chain" in which subassemblies of information are passed from one agent to another, each one "adding value," to construct the final product requested by the end user. Make no mistake: to create complicated value chains automatically on demand, some agents will exploit artificial-intelligence technologies in addition to the Semantic Web. But the Semantic Web will provide the foundations and the framework to make such technologies more feasible.

Putting all these features together results in the abilities exhibited by Pete's and Lucy's agents in the scenario that opened this article. Their agents would have delegated the task in piecemeal fashion to other services and agents discovered through service advertisements. For example, they could have used a *trusted* service to take a list of *providers* and determine which of them are *in-plan* for a specified *insurance plan* and *course of treatment*. The list of providers would have been supplied by another search service, et cetera. These activities formed chains in which a large amount of data distributed across the Web (and almost worthless in that form) was progressively reduced to the small amount of data of high value to Pete and Lucy—a plan of appointments to fit their schedules and other requirements.

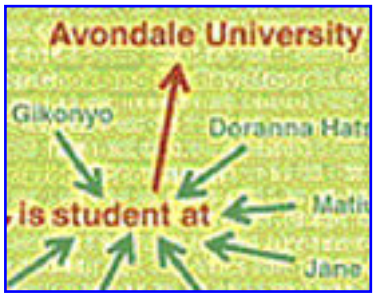
In the next step, the Semantic Web will break out of the virtual realm and extend into our physical world. URIs can point to anything, including physical entities, which means we can use the RDF language to describe devices such as cell phones and TVs. Such devices can advertise their functionality—what they can do and how they are controlled—much like software agents. Being much more flexible than low-level schemes such as Universal Plug and Play, such a semantic approach opens up a world of exciting possibilities.

For instance, what today is called home automation requires careful configuration for appliances to work together. Semantic descriptions of device capabilities and functionality will let us achieve such automation with minimal human intervention. A trivial example occurs when Pete answers his phone and the stereo sound is turned down. Instead of having to program each specific appliance, he could program such a function once and for all to cover every *local* device that advertises having a *volume control* — the TV, the DVD player and even the media players on the laptop that he brought home from work this one evening.

The first concrete steps have already been taken in this area, with work on developing a standard for describing functional capabilities of devices (such as screen sizes) and user preferences. Built on RDF, this standard is called Composite Capability/Preference Profile (CC/PP). Initially it will let cell phones and other nonstandard Web clients describe their characteristics so

that Web content can be tailored for them on the fly. Later, when we add the full versatility of languages for handling ontologies and logic, devices could automatically seek out and employ services and other devices for added information or functionality. It is not hard to imagine your Web-enabled microwave oven consulting the frozen-food manufacturer's Web site for optimal cooking parameters.

Evolution of Knowledge



ELABORATE, PRECISE SEARCHES

The semantic web is not "merely" the tool for conducting individual tasks that we have discussed so far. In addition, if properly designed, the Semantic Web can assist the evolution of human knowledge as a whole.

Human endeavor is caught in an eternal tension between the effectiveness of small groups acting independently and the need to mesh with the wider community. A small group can innovate rapidly and efficiently, but this produces a subculture whose concepts are not understood by others. Coordinating actions across a large group, however, is painfully slow and takes an enormous amount of communication. The world works across the spectrum between these extremes, with a tendency to start small—from the personal idea—and move toward a wider understanding over time.

An essential process is the joining together of subcultures when a wider common language is needed. Often two groups independently develop very similar concepts, and describing the relation between them brings great benefits. Like a Finnish-English dictionary, or a weights-and-measures conversion table, the relations allow communication and collaboration even when the commonality of concept has not (yet) led to a commonality of terms.

The Semantic Web, in naming every concept simply by a URI, lets anyone express new concepts that they invent with minimal effort. Its unifying logical language will enable these concepts to be progressively linked into a universal Web. This structure will open up the knowledge and workings of humankind to meaningful analysis by software agents, providing a new class of tools by which we can live, work and learn together.

PHOTOILLUSTRATIONS BY MIGUEL SALMERON

Further Information:

Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.
Tim Berners-Lee, with Mark Fischetti. Harper San Francisco, 1999.
An enhanced version of this article is on the Scientific American Web site, with additional material and links.

World Wide Web Consortium (W3C): **www.w3.org/**

W3C Semantic Web Activity: **www.w3.org/2001/sw/**

An introduction to ontologies:
www.SemanticWeb.org/knowmarkup.html

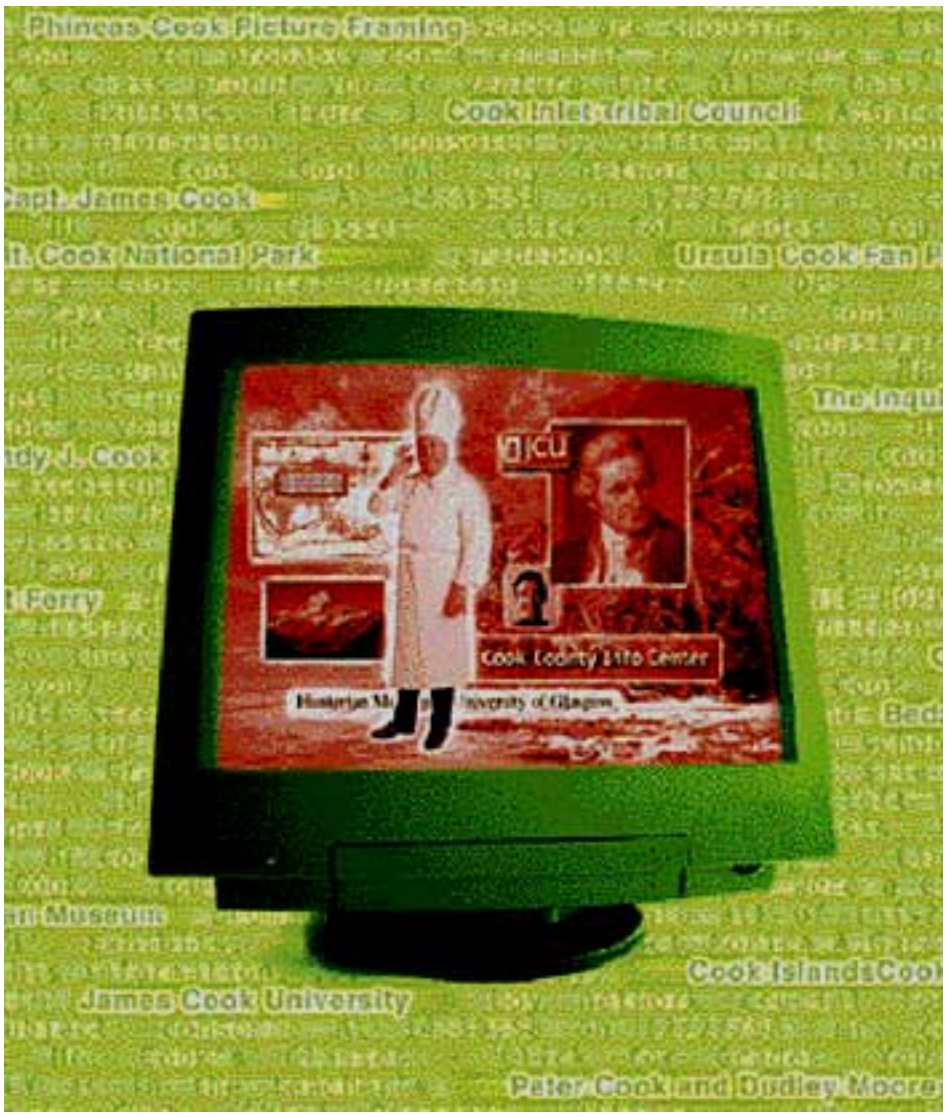
Simple HTML Ontology Extensions Frequently Asked Questions (SHOE FAQ): **www.cs.umd.edu/projects/plus/SHOE/faq.html**

DARPA Agent Markup Language (DAML) home page: **www.daml.org/**

The Authors

TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA are individually and collectively obsessed with the potential of Semantic Web technology. Berners-Lee is director of the World Wide Web Consortium (W3C) and a researcher at the Laboratory for Computer Science at the Massachusetts Institute of Technology. When he invented the Web in 1989, he intended it to carry more semantics than became common practice. Hendler is professor of computer science at the University of Maryland at College Park, where he has been doing research on knowledge representation in a Web context for a number of years. He and his graduate research group developed SHOE, the first Web-based knowledge representation language to demonstrate many of the agent capabilities described in this article. Hendler is also responsible for agent-based computing research at the Defense Advanced Research Projects Agency (DARPA) in Arlington, Va. Lassila is a research fellow at the Nokia Research Center in Boston, chief scientist of Nokia Venture Partners and a member of the W3C Advisory Board. Frustrated with the difficulty of building agents and automating tasks on the Web, he co-authored W3C's RDF specification, which serves as the foundation for many current Semantic Web efforts.

Web Searches Today



WEB SEARCHES TODAY typically turn up innumerable completely irrelevant "hits," requiring much manual filtering by the user. If you search using the keyword "cook," for example, the computer has no way of knowing whether you are looking for a chef, information about how to cook something, or simply a place, person, business or some other entity with "cook" in its name. The problem is that the word "cook" has no meaning, or semantic content, to the computer.

PHOTOILLUSTRATIONS BY MIGUEL SALMERON

Back to Article (The Semantic Web)

XML and the Second-Generation Web

The combination of hypertext and a global Internet started a revolution.
A new ingredient, XML, is poised to finish the job

by Jon Bosak and Tim Bray

Give people a few hints, and they can figure out the rest. They can look at this page, see some large type followed by blocks of small type and know that they are looking at the start of a magazine article. They can look at a list of groceries and see shopping instructions. They can look at some rows of numbers and understand the state of their bank account.

Computers, of course, are not that smart; they need to be told exactly what things are, how they are related and how to deal with them. Extensible Markup Language (XML for short) is a new language designed to do just that, to make information self-describing. This simple-sounding change in how computers communicate has the potential to extend the Internet beyond information delivery to many other kinds of human activity. Indeed, since XML was completed in early 1998 by the World Wide Web Consortium (usually called the W3C), the standard has spread like wildfire through science and into industries ranging from manufacturing to medicine.

The enthusiastic response is fueled by a hope that XML will solve some of the Web's biggest problems. These are widely known: the Internet is a speed-of-light network that often moves at a crawl; and although nearly every kind of information is available on-line, it can be maddeningly

XML Hyperlink

SIDEBARS:

New Languages for Science

Further XML links

FURTHER READING

RELATED LINKS

difficult to find the one piece you need.

Both problems arise in large part from the nature of the Web's main language, HTML (shorthand for Hypertext Markup Language). Although HTML is the most successful electronic-publishing language ever invented, it is superficial: in essence, it describes how a Web browser should arrange text, images and push-buttons on a page. HTML's concern with appearances makes it relatively easy to learn, but it also has its costs.

One is the difficulty in creating a Web site that functions as more than just a fancy fax machine that sends documents to anyone who asks. People and companies want Web sites that take orders from customers, transmit medical records, even run factories and scientific instruments from half a world away. HTML was never designed for such tasks.

So although your doctor may be able to pull up your drug reaction history on his Web browser, he cannot then e-mail it to a specialist and expect her to be able to paste the records directly into her hospital's database. Her computer would not know what to make of the information, which to its eyes would be no more intelligible than < H1 >blah blah < /H1 > < BOLD >blah blah blah < /BOLD >. As programming legend Brian Kernighan once noted, the problem with "What You See Is What You Get" is that what you see is all you've got.

Those angle-bracketed labels in the example just above are called tags. HTML has no tag for a drug reaction, which highlights another of its limitations: it is inflexible. Adding a new tag involves a bureaucratic process that can take so long that few attempt it. And yet every application, not just the interchange of medical records, needs its own tags.

Thus the slow pace of today's on-line bookstores, mail-order catalogues and other interactive Web sites. Change the quantity or shipping method of your order, and to see the handful of digits that have changed in the total, you must ask a distant, overburdened server to send you an entirely new page, graphics and all. Meanwhile your own high-powered machine sits waiting idly, because it has only been told about < H1 >s and < BOLD >s, not about prices and shipping options.

Thus also the dissatisfying quality of Web searches. Because there is no way to mark something as a price, it is effectively impossible to use price information in your searches.

Something Old, Something New

The solution, in theory, is very simple: use tags that say what the

information is, not what it looks like. For example, label the parts of an order for a shirt not as boldface, paragraph, row and column--what HTML offers--but as price, size, quantity and color. A program can then recognize this document as a customer order and do whatever it needs to do: display it one way or display it a different way or put it through a bookkeeping system or make a new shirt show up on your doorstep tomorrow.

We, as members of a dozen-strong W3C working group, began crafting such a solution in 1996. Our idea was powerful but not entirely original. For generations, printers scribbled notes on manuscripts to instruct the typesetters. This "markup" evolved on its own until 1986, when, after decades of work, the International Organization for Standardization (ISO) approved a system for the creation of new markup languages.

Named Standard Generalized Markup Language, or SGML, this language for describing languages--a metalanguage--has since proved useful in many large publishing applications. Indeed, HTML was defined using SGML. The only problem with SGML is that it is too general--full of clever features designed to minimize keystrokes in an era when every byte had to be accounted for. It is more complex than Web browsers can cope with.

Our team created XML by removing frills from SGML to arrive at a more streamlined, digestible metalanguage. XML consists of rules that anyone can follow to create a markup language from scratch. The rules ensure that a single compact program, often called a parser, can process all these new languages. Consider again the doctor who wants to e-mail your medical record to a specialist. If the medical profession uses XML to hammer out a markup language for encoding medical records--and in fact several groups have already started work on this--then your doctor's e-mail could contain < patient > < name > blah blah < /name > < drug-allergy > blah blah blah < /drug-allergy > < /patient >. Programming any computer to recognize this standard medical notation and to add this vital statistic to its database becomes straightforward.

Just as HTML created a way for every computer user to read Internet documents, XML makes it possible, despite the Babel of incompatible computer systems, to create an Esperanto that all can read and write. Unlike most computer data formats, XML markup also makes sense to humans, because it consists of nothing more than ordinary text.

The unifying power of XML arises from a few well-chosen rules. One is that tags almost always come in pairs. Like parentheses, they surround the text to which they apply. And like quotation marks, tag pairs can be nested inside one another to multiple levels.

The nesting rule automatically forces a certain simplicity on every XML document, which takes on the structure known in computer science as a tree. As with a genealogical tree, each graphic and bit of text in the document represents a parent, child or sibling of some other element; relationships are unambiguous. Trees cannot represent every kind of information, but they can represent most kinds that we need computers to understand. Trees, moreover, are extraordinarily convenient for programmers. If your bank statement is in the form of a tree, it is a simple matter to write a bit of software that will reorder the transactions or display just the cleared checks.

Another source of XML's unifying strength is its reliance on a new standard called Unicode, a character-encoding system that supports intermingling of text in all the world's major languages. In HTML, as in most word processors, a document is generally in one particular language, whether that be English or Japanese or Arabic. If your software cannot read the characters of that language, then you cannot use the document. The situation can be even worse: software made for use in Taiwan often cannot read mainland-Chinese texts because of incompatible encodings. But software that reads XML properly can deal with any combination of any of these character sets. Thus, XML enables exchange of information not only between different computer systems but also across national and cultural boundaries.

An End to the World Wide Wait

As XML spreads, the Web should become noticeably more responsive. At present, computing devices connected to the Web, whether they are powerful desktop computers or tiny pocket planners, cannot do much more than get a form, fill it out and then swap it back and forth with a Web server until a job is completed. But the structural and semantic information that can be added with XML allows these devices to do a great deal of processing on the spot. That not only will take a big load off Web servers but also should reduce network traffic dramatically.

To understand why, imagine going to an on-line travel agency and asking for all the flights from London to New York on July 4. You would probably receive a list several times longer than your screen could display. You could shorten the list by fine-tuning the departure time, price or airline, but to do that, you would have to send a request across the Internet to the travel agency and wait for its answer. If, however, the long list of flights had been sent in XML, then the travel agency could have sent a small Java program along with the flight records that you could use to sort and winnow them in microseconds, without ever involving the server. Multiply this by a few million Web users, and the global efficiency gains become dramatic.

As more of the information on the Net is labeled with industry-specific XML tags, it will become easier to find exactly what you need. Today an Internet search for "stockbroker jobs" will inundate you with advertisements but probably turn up few job listings--most will be hidden inside the classified ad services of newspaper Web sites, out of a search robot's reach. But the Newspaper Association of America is even now building an XML-based markup language for classified ads that promises to make such searches much more effective.

Even that is just an intermediate step. Librarians figured out a long time ago that the way to find information in a hurry is to look not at the information itself but rather at much smaller, more focused sets of data that guide you to the useful sources: hence the library card catalogue. Such information about information is called metadata.

From the outset, part of the XML project has been to create a sister standard for metadata. The Resource Description Framework (RDF), finished this past February, should do for Web data what catalogue cards do for library books. Deployed across the Web, RDF metadata will make retrieval far faster and more accurate than it is now. Because the Web has no librarians and every Webmaster wants, above all else, to be found, we expect that RDF will achieve a typically astonishing Internet growth rate once its power becomes apparent.

There are of course other ways to find things besides searching. The Web is after all a "hypertext," its billions of pages connected by hyperlinks--those underlined words you click on to get whisked from one to the next. Hyperlinks, too, will do more when powered by XML. A standard for XML-based hypertext, named XLink and due later this year from the W3C, will allow you to choose from a list of multiple destinations. Other kinds of hyperlinks will insert text or images right where you click, instead of forcing you to leave the page.

Perhaps most useful, XLink will enable authors to use indirect links that point to entries in some central database rather than to the linked pages themselves. When a page's address changes, the author will be able to update all the links that point to it by editing just one database record. This should help eliminate the familiar "404 File Not Found" error that signals a broken hyperlink. The combination of more efficient processing, more accurate searching and more flexible linking will revolutionize the structure of the Web and make possible completely new ways of accessing information. Users will find this new Web faster, more powerful and more useful than the Web of today.

Some Assembly Required

Of course, it is not quite that simple. XML does allow anyone to design a new, custom-built language, but designing good languages is a challenge that should not be undertaken lightly. And the design is just the beginning: the meanings of your tags are not going to be obvious to other people unless you write some prose to explain them, nor to computers unless you write some software to process them.

A moment's thought reveals why. If all it took to teach a computer to handle a purchase order were to label it with tags, we wouldn't need XML. We wouldn't even need programmers--the machines would be smart enough to take care of themselves.

What XML does is less magical but quite effective nonetheless. It lays down ground rules that clear away a layer of programming details so that people with similar interests can concentrate on the hard part--agreeing on how they want to represent the information they commonly exchange. This is not an easy problem to solve, but it is not a new one, either.

Such agreements will be made, because the proliferation of incompatible computer systems has imposed delays, costs and confusion on nearly every area of human activity. People want to share ideas and do business without all having to use the same computers; activity-specific interchange languages go a long way toward making that possible. Indeed, a shower of new acronyms ending in "ML" testifies to the inventiveness unleashed by XML in the sciences, in business and in the scholarly disciplines [see box on opposite page].

Before they can draft a new XML language, designers must agree on three things: which tags will be allowed, how tagged elements may nest within one another and how they should be processed. The first two--the language's vocabulary and structure--are typically codified in a Document Type Definition, or DTD. The XML standard does not compel language designers to use DTDs, but most new languages will probably have them, because they make it much easier for programmers to write software that understands the markup and does intelligent things with it. Programmers will also need a set of guidelines that describe, in human language, what all the tags mean. HTML, for instance, has a DTD but also hundreds of pages of descriptive prose that programmers refer to when they write browsers and other Web software.

A Question of Style

For users, it is what those programs do, not what the descriptions say, that

is important. In many cases, people will want software to display XML-encoded information to human readers. But XML tags offer no inherent clues about how the information should look on screen or on paper.

This is actually an advantage for publishers, who would often like to "write once and publish everywhere"--to distill the substance of a publication and then pour it into myriad forms, both printed and electronic. XML lets them do this by tagging content to describe its meaning, independent of the display medium. Publishers can then apply rules organized into "stylesheets" to reformat the work automatically for various devices. The standard now being developed for XML stylesheets is called the Extensible Stylesheet Language, or XSL.

The latest versions of several Web browsers can read an XML document, fetch the appropriate stylesheet, and use it to sort and format the information on the screen. The reader might never know that he is looking at XML rather than HTML--except that XML-based sites run faster and are easier to use.

People with visual disabilities gain a free benefit from this approach to publishing. Stylesheets will let them render XML into Braille or audible speech. The advantages extend to others as well: commuters who want to surf the Web in their cars may also find it handy to have pages read aloud.

Although the Web has been a boon to science and to scholarship, it is commerce (or rather the expectation of future commercial gain) that has fueled its lightning growth. The recent surge in retail sales over the Web has drawn much attention, but business-to-business commerce is moving on-line at least as quickly. The flow of goods through the manufacturing process, for example, begs for automation. But schemes that rely on complex, direct program-to-program interaction have not worked well in practice, because they depend on a uniformity of processing that does not exist.

For centuries, humans have successfully done business by exchanging standardized documents: purchase orders, invoices, manifests, receipts and so on. Documents work for commerce because they do not require the parties involved to know about one another's internal procedures. Each record exposes exactly what its recipient needs to know and no more. The exchange of documents is probably the right way to do business on-line, too. But this was not the job for which HTML was built.

XML, in contrast, was designed for document exchange, and it is becoming clear that universal electronic commerce will rely heavily on a flow of agreements, expressed in millions of XML documents pulsing around the

Internet.

Thus, for its users, the XML-powered Web will be faster, friendlier and a better place to do business. Web site designers, on the other hand, will find it more demanding. Battalions of programmers will be needed to exploit new XML languages to their fullest. And although the day of the self-trained Web hacker is not yet over, the species is endangered. Tomorrow's Web designers will need to be versed not just in the production of words and graphics but also in the construction of multilayered, interdependent systems of DTDs, data trees, hyperlink structures, metadata and stylesheets--a more robust infrastructure for the Web's second generation.

Related Links:

What the ?XML!- An introduction to XML, the second coming of the web and how it creates a content oriented interface.

What is XML?

The Authors:

JON BOSAK and TIM BRAY played crucial roles in the development of XML. Bosak, an on-line information technology architect at Sun Microsystems in Mountain View, Calif., organized and led the World Wide Web Consortium working group that created XML. He is currently chair of the W3C XML Coordination Group and a representative to the Organization for the Advancement of Structured Information Standards. Bray is co-editor of the XML 1.0 specification and the related Namespaces in XML and serves as co-chair of the W3C XML Syntax Working Group. He managed the New Oxford English Dictionary Project at the University of Waterloo in 1986, co-founded Open Text Corporation in 1989 and launched Textuality, a programming firm in Vancouver, B.C., in 1996.