

1

XML: Extending the Enterprise

Extensible Markup Language (XML) is a simple data description language with profound implications. It affects how we build software and how we think about distributed systems. Surrounding XML is a family of standards and technologies that have opened up new possibilities for exchanging information across the World Wide Web and building communication infrastructures.

XML derives much of its strength in combination with the Web. The Web provides a collection of protocols for moving data; XML represents a way to define that data. The most immediate effect has been a new way to look at the enterprise. Instead of a tightly knit network of servers, the enterprise is now seen as encompassing not just our traditional networks but also the Web itself, with its global reach and scope.

In this chapter we look at XML's role in the expanding view of the enterprise. From its beginnings as a language for describing vertical industry data, XML has blossomed to include not only horizontal applications but also protocols that challenge the conventional wisdom about how to do distributed computing and that open the door to new ways of discovery and connection. Web services represents the most recent visible effect of this change.

In this chapter we also try to establish the context for the changes brought about by XML and the Web. To place events in perspective we explore three technology revolutions: data, architecture, and software. Together the changes in these areas are fostering new ways of thinking about the enterprise and about application development in general. The new paradigm that emerges is the result of global Web-based communication infrastructure, the

Chapter 1 XML: Extending the Enterprise

ability to describe data with XML, and the emergence of XML-based protocols such as Simple Object Access Protocol (SOAP) that contribute to a fabric of loosely coupled distributed systems. As we'll see, these forces are driving initiatives such as Microsoft's .NET and Sun's Java 2 Enterprise Edition (J2EE) that are looking at XML as the bridge between the more traditional space of tightly coupled object systems and the more freewheeling loosely coupled space of the Web.

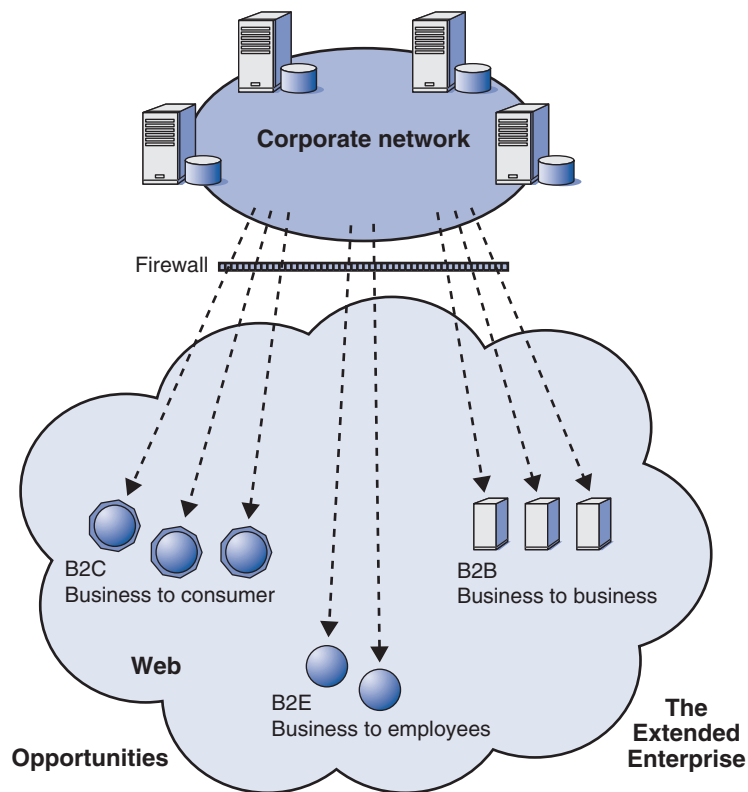
Extending the Enterprise

The Web has opened up new possibilities for engagement.

The story behind XML is very much the story of the Web. In just a few short years the Web has affected almost every aspect of our lives, from work, to play, to social interaction. However, until recently, the Web's impact has been primarily on individuals, providing a quick and efficient way to check email, search for information, and buy things online. The global scope of the Web and its possibilities have not gone unnoticed by companies looking to gain competitive advantage. This global connectivity, coupled with accepted Internet standards for communication, has spawned new ideas about how to leverage this new capability.

The extended enterprise includes B2C, B2B, and B2E interaction.

As Figure 1.1 illustrates, there are three major aspects to extending the enterprise from a relatively constrained network to the broad reach of the Web. The most commonly considered aspect is the business-to-consumer (B2C) connection, exploiting opportunities that abound in online commerce. Another area is the business-to-employee (B2E) connection, adding efficiencies in operations and customer contact by using the Web instead of proprietary networks. A third area, and one of particular interest to businesses trying to survive in competitive environments, is the business-to-business (B2B) connection made possible by the Web. Together, these opportunities are driving what is seen as the extended enterprise, a mix of traditional networks and the loose space of the Web.



The Role of XML

Figure 1.1 The extended enterprise combines traditional networks with the power of the Web, opening up new opportunities in B2B, B2C, and B2E commerce.

But to turn these possibilities and visions of global connectedness into a reality requires data, not only for the consumer, but for employees who need data and information to do their jobs wherever they are, and for the B2B transactions that are the essential ingredient for partners and suppliers. That's where XML comes in.

The Role of XML

XML is a metalanguage (literally a language about languages) defined by the World Wide Web Consortium (W3C), one of the main organizations driving the push to open Web standards. In its simplest sense, XML is a set of rules and guidelines for describing

Data is the key to the extended enterprise.

XML is a specification for defining new markup languages.

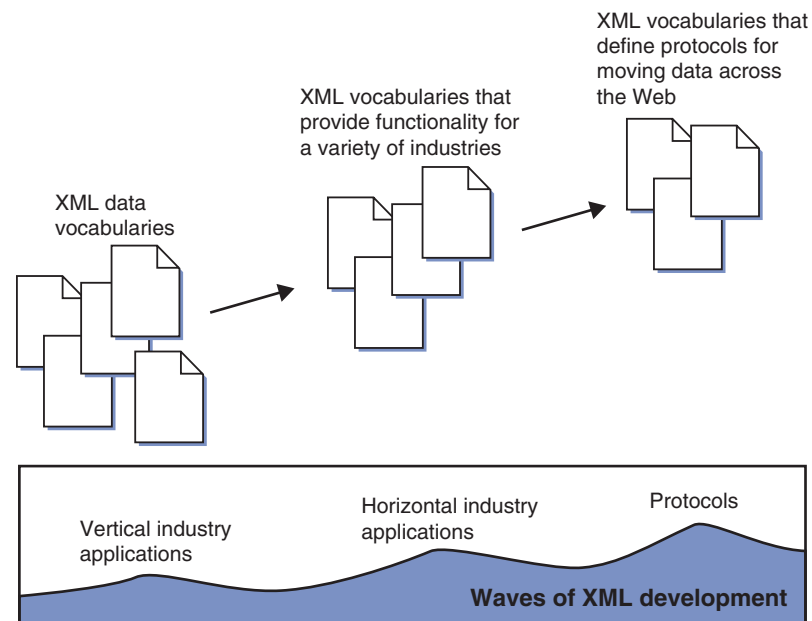
Chapter 1 XML: Extending the Enterprise

structured data in plain text rather than proprietary binary representations. However, as a phenomenon, XML goes beyond its technical specification. Since its standardization by the W3C in 1998, XML has been the driving force behind numerous other standards and vocabularies that are forging a fundamental change in the software world.

XML has enabled industry vocabularies and protocols.

In its short history, XML has given rise to numerous vertical industry vocabularies in support of B2B e-commerce, horizontal vocabularies that provide services to a wide range of industries, and XML protocols that have used XML's simple power of combination to open up new possibilities for doing distributed computing. As Figure 1.2 shows, XML's influence has been felt in three waves, from industry-specific vocabularies, to horizontal industry applications, to protocols that describe how businesses can exchange data across the Web. One of the key developments has been SOAP, the protocol

Figure 1.2 XML has been widely used as a language for a variety of applications ranging from vertical industry vocabularies, to horizontal industry applications, to protocols.



that has opened the Web to program-to-program communication and, as we'll see in Chapter 5, is the basis for Web services.

XML: Just Tags?

XML is simple. Technically, it's a language for creating other languages based on the insertion of tags to help describe data. However, XML is actually more than just tags. To see what we mean by this, let's begin with a simple XML data description.

XML supports user-defined languages that add meaning to data.

XML is a combination of tags and content in which the tags add meaning to the content. The following is a simple XML markup of customer information. Start tags such as <Name> begin an element that contains the actual data. End tags such as </Name> mark the end of an element definition.

```
<Customer>
  <Name>John von Neumann</Name>
  <PhoneNum>914.631.7722</PhoneNum>
  <FaxNum>914.631.7723</FaxNum>
  <E-Mail>Johnny@cd.com</E-Mail>
</Customer>
```

However, elements are only one way to describe data. It's also possible to represent the data using attributes within a single element:

```
<Customer name="John von Neumann" phone="914.631.7722"
fax="914.631.7723" email="Johnny@cd.com"/>
```

In both these examples, the data is the same, but the form is different. There are several important basic points to observe about these definitions.

- ❑ XML allows data to be stored in either elements or attributes.
- ❑ Elements and attributes can be named to give the data meaning.

Chapter 1 XML: Extending the Enterprise

- ❑ Start tags and end tags define elements that are the basis for XML tree-structured representations of documents.
- ❑ Elements can contain text data and/or other elements.

While XML is more than just a few simple rules, its essential aspects can be grasped and understood quite easily. What's important about XML is what it brings to enterprise computing and how it is affecting many aspects of software development and e-commerce.

Several aspects of XML have contributed to its success.

The XML Advantage

XML has had an impact across a broad range of areas. The following is a list of some of the factors that have influenced XML's adoption by a variety of organizations and individuals.

- ❑ XML files are human-readable. XML was designed as text so that, in the worst case, someone can always read it to figure out the content. Such is not the case with binary data formats.
- ❑ Widespread industry support exists for XML. Numerous tools and utilities are being provided with Web browsers, databases, and operating systems, making it easier and less expensive for small and medium-sized organizations to import and export data in XML format.
- ❑ Major relational databases now have the native capability to read and generate XML data.
- ❑ A large family of XML support technologies is available for the interpretation and transformation of XML data for Web page display and report generation.

Much of XML's success stems from what it does not address.

XML: Design by Omission

In addition to XML's explicit advantages, it's important to realize that much of XML's widespread success and use derives more from what it does *not* address. There are three key design elements that by omission contribute to XML's success:

1. No display is assumed. Unlike HTML, XML makes no assumptions about how tags will be rendered in a browser or other display device. Auxiliary technologies such as style sheets add this capability.
2. There is no built-in data typing. DTDs and XML Schema provide support for defining the structure and data types associated with an XML document.
3. No transport is assumed. The XML specification makes no assumption about how XML is to be transported across the Internet. This has opened the door to creative ideas about delivering XML by means of HTTP, FTP, or Simple Mail Transfer Protocol (SMTP).

These design-by-omission principles can be reformulated by saying that XML explicitly limits the scope of ambitions to maximize interaction with other technologies.

XML and the Web

XML's capability to work with other technologies has greatly expanded the possibilities for navigating the space of the new extended enterprise. Figure 1.3 shows how XML may be used to communicate directly with partners and suppliers. Instead of exchanging data about purchases and orders either manually or over proprietary networks, data vocabularies can be defined using XML and delivered from server to server using standard protocols such as HTTP or FTP.

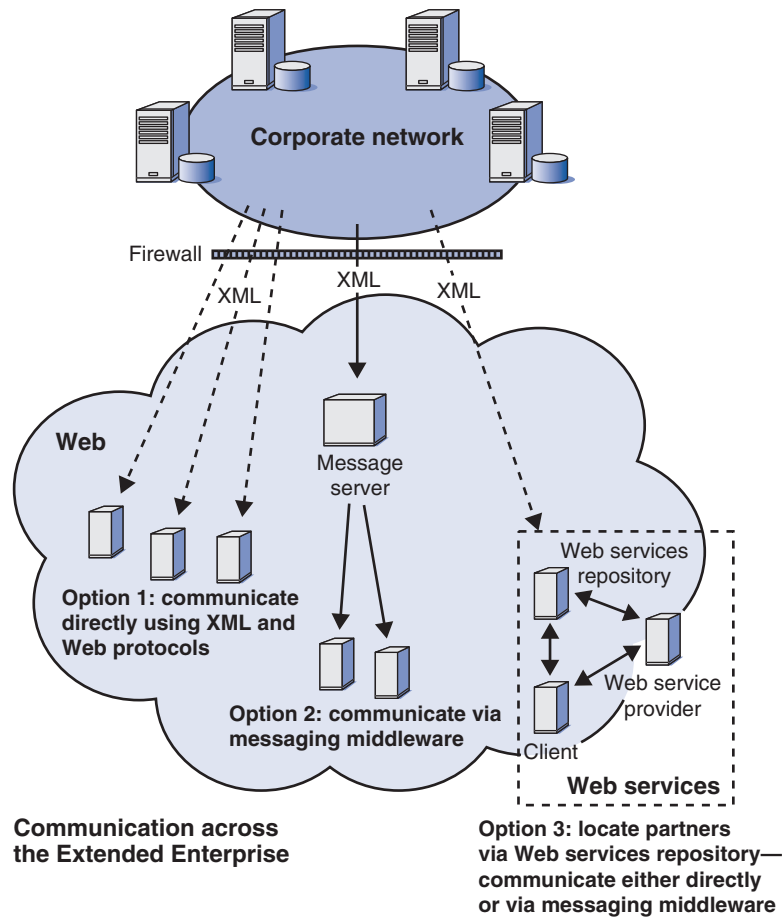
Associated with this ability to move data freely across the Web is the rise in the use of messaging servers and software that sit between conversational participants. These servers, supporting what is known as Message Oriented Middleware, are playing an increasingly important role in the new extended enterprise by providing guarantees of delivery and the ability to broadcast communications to multiple recipients.

XML integrates with standard Web protocols such as HTTP and FTP.

Messaging middleware supports the asynchronous delivery of XML.

Chapter 1 XML: Extending the Enterprise

Figure 1.3 XML fits into the fabric of a new architecture built around Internet protocols and loosely coupled systems.



Web services makes B2B direct connections feasible over the Web.

A third aspect of the new extended enterprise is the emergence of Web services. For some, Web services represents the next evolutionary step for the Web, extending it from a network that provides services to humans to one that provides services to software looking to connect with other software. Web services is an ambitious initiative that is moving the Web to new levels of B2B (that is, software-to-software) interaction while trying to fulfill object technology's promise of reusable components from a service interface

perspective. In Chapter 5 we take a closer look at Web services, but for now it's important to see the growing momentum behind Web services as an effort (but by no means the only effort) to structure B2B activity over the Web.

SOAP

SOAP is the XML glue that lets clients and providers talk to each other and exchange XML data. As Figure 1.4 shows, SOAP builds on XML and common Web protocols (HTTP, FTP, and SMTP) to enable communication across the Web. As we discuss in more detail in Chapter 4, SOAP brings to the table a set of rules for moving data,

SOAP is an XML-based protocol.

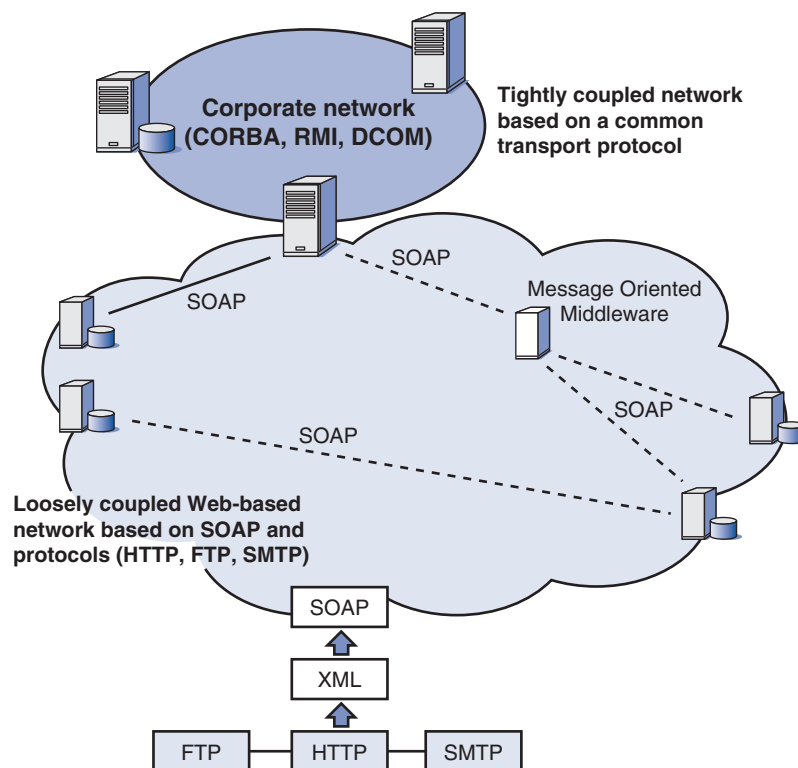


Figure 1.4 SOAP, an XML-based protocol, gains its global scope through the power of combination with Internet protocols such as HTTP, FTP, and SMTP.

Chapter 1 XML: Extending the Enterprise

either directly in a point-to-point fashion or by sending the data through a message queue intermediary.

SOAP opens up new options for distributed computing.

One of the main implications of SOAP is a change in how we think about distributed computing. Prior to SOAP, there were three basic options for doing distributed computing: Microsoft's Distributed Component Object Model (DCOM), Java's Remote Method Invocation (RMI), or the Object Management Group's Common Object Request Broker Architecture (CORBA). These technologies are still in widespread use today. Their drawback is that they limit the potential reach of the enterprise to servers that share the same object infrastructure. With SOAP, however, the potential space of interconnection is the entire Web itself, which is why there is such intense interest in technologies that can leverage the power of SOAP. One of these technology efforts is Web services.

Web Services

Web services builds on a SOAP foundation.

Web services is both a process and set of protocols for finding and connecting to software exposed as services over the Web. By assuming a SOAP foundation, Web services can concentrate on what data to exchange instead of worrying about how to get it from point A to point B, which is the job of SOAP. To make things even easier, SOAP also defines an XML envelope to carry XML and a convention for doing remote procedure calls so that a service can advertise "call me here" and a program will be able to do so without concern for language or platform. Although SOAP may be used with a variety of protocols, the only bindings specified in the proposed SOAP specification are for HTTP.

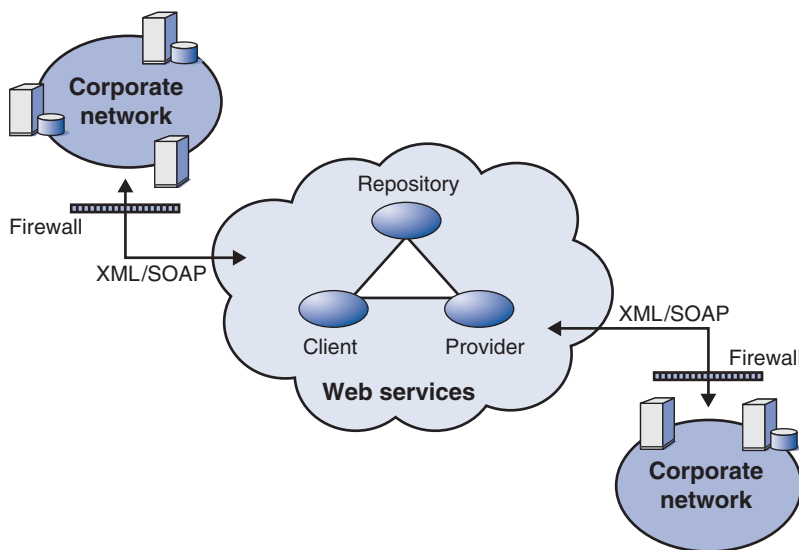
Web services facilitates software interaction.

A Web service can be anything from a movie review service, to a real-time weather advisory, to an entire hotel- and airline-booking package. The Web services technical infrastructure ensures that

services even from different vendors will interoperate to create a complete business process. Web services takes the object-oriented vision of assembling software from component building blocks to the next level. With Web services, however, the emphasis is on the assembly of services that may or may not be built on object technology.

As Figure 1.5 illustrates, the interconnections opened up by the Web make possible a new way of interacting through the registration, discovery, and connection of software packaged as Web services. As we'll see in Chapter 5, there are three major aspects to Web Services:

- ❑ A *service provider* provides an interface for software that can carry out a specified set of tasks.
- ❑ A *service requester* discovers and invokes a software service to provide a business solution. The requester will commonly invoke a remote procedure call on the service provider, passing parameter data to the provider and receiving a result in reply.



*Web services =
repository + client
+ provider.*

Figure 1.5 The Web services framework provides protocols and a process for clients to discover and connect to web-based services.

Chapter 1 XML: Extending the Enterprise

- A *broker* manages and publishes the service. Service providers publish their services with the broker and requests access those services by creating bindings to the service provider.

.NET and J2EE

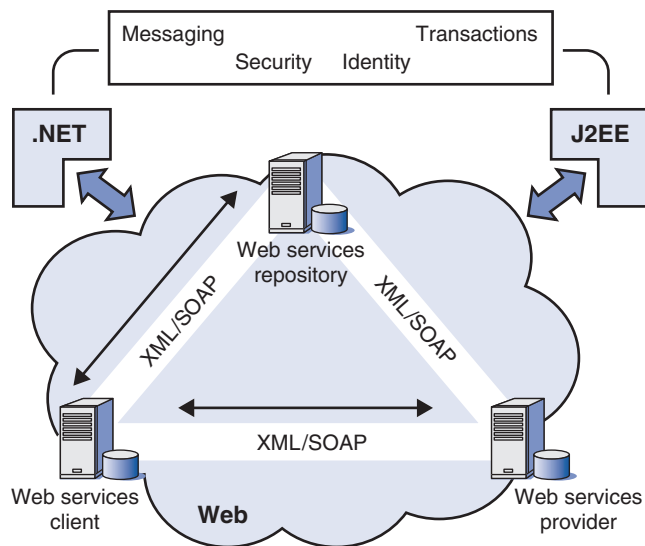
Messaging, transactions, security, and identity are key ingredients for Web commerce.

.NET and J2EE are options for building an extended enterprise.

While Web services represents one way to exploit the power and reach of the Web, there is a caveat. Web services and SOAP-based connections do not currently have the key building blocks for industrial-strength Web-based e-business. As Figure 1.6 shows, what's needed is support for the four pillars of Web-based e-business: messaging, transactions, security, and identity.

As we'll see in Chapter 6, messaging, security, transactions, and identity are the essential ingredients in managing interactions across the extended enterprise. In the current software world, these capabilities are being provided from two directions. On one side is Microsoft's .NET, a Windows-centric framework for extending Windows-based

Figure 1.6 .NET and J2EE add capability for messaging, security, identity, and transactions to loosely coupled networks based on SOAP.



networks into the extended space of the Web. On the other side is J2EE, a Java-centric specification that is being implemented by several companies including Sun, IBM, BEA, HP, Oracle, and others. While we'll defer until Chapter 6 taking a look at how these competitive forces are playing out, it's important to keep the big picture in mind as we explore each of these concepts in subsequent chapters.

Now that we've seen where we'll be going, let's step back and take a broader look at some of the forces at play in the rapidly changing world of the Web. As we do so, we'll fill in some details that will help in subsequent chapters and explore how XML is playing an integral part in three revolutions.

XML: The Three Revolutions

At the beginning of this chapter we outlined several areas in which XML's impact has been felt. To understand the changes that are occurring in today's software world, it's helpful to look at XML in the context of three revolutions in which XML is playing a major role.

The three revolutions: data, architecture, and software.

As Figure 1.7 illustrates, the three areas of impact are data, which XML frees from the confines of fixed, program-dependent formats; architecture, with a change in emphasis from tightly coupled distributed systems to a more loosely coupled confederation based on the Web; and software, with the realization that software evolution is a better path to managing complexity than building monolithic applications. In the following sections we'll explore each in more detail.

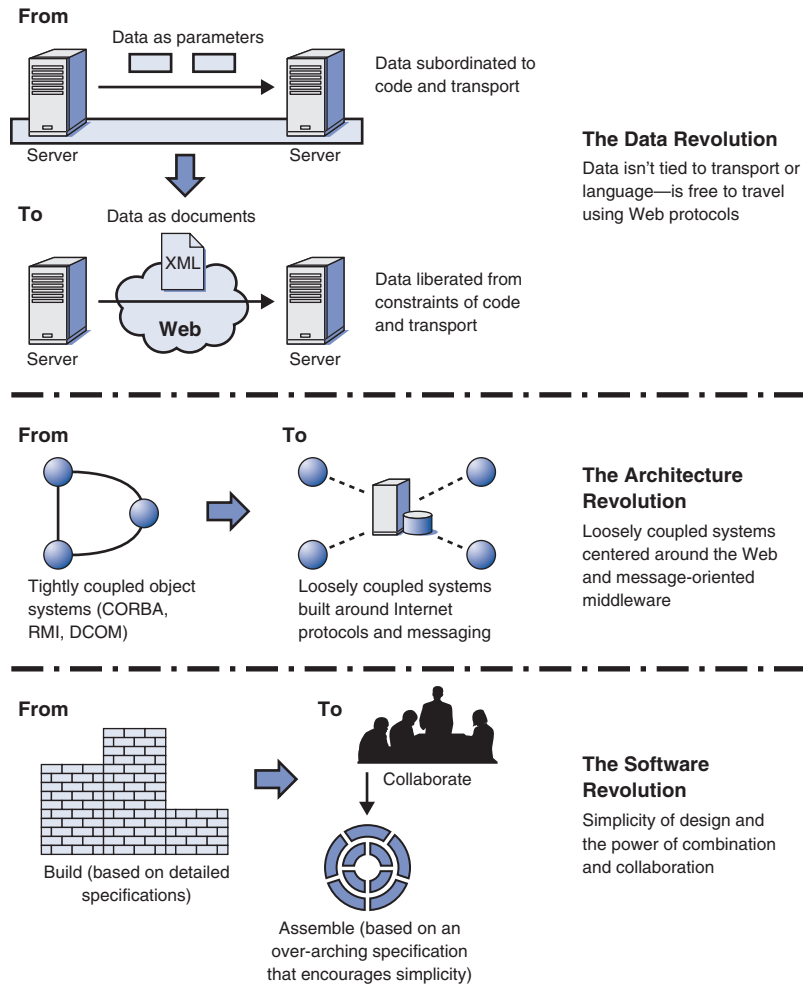
The Data Revolution

Prior to XML, data was very much proprietary, closely associated with applications that understood how data was formatted and how to process it. Now, XML-based industry-specific data vocabularies provide alternatives to specialized Electronic Data Interchange (EDI)

Data is now free to travel the Web.

Chapter 1 XML: Extending the Enterprise

Figure 1.7 The three XML revolutions: data, architecture, and software.



solutions by facilitating B2B data exchange and playing a key role as a messaging infrastructure for distributed computing.

XML enables the creation of program-independent data formats.

XML's strength is its data independence. XML is pure data description, not tied to any programming language, operating system, or transport protocol. In the grand scheme of distributed computing this is a radical idea. The implication is that we don't require lock-in

to programmatic infrastructures to make data available to Web-connected platforms. In effect, data is free to move about globally without the constraints imposed by tightly coupled transport-dependent architectures. XML's sole focus on data means that a variety of transport technologies may be used to move XML across the Web. As a result, protocols such as HTTP have had a tremendous impact on XML's viability and have opened the door to alternatives to CORBA, RMI, and DCOM, which don't work over TCP/IP. XML does this by focusing on data and leaving other issues to supporting technologies.

XML: Origin and Cultures

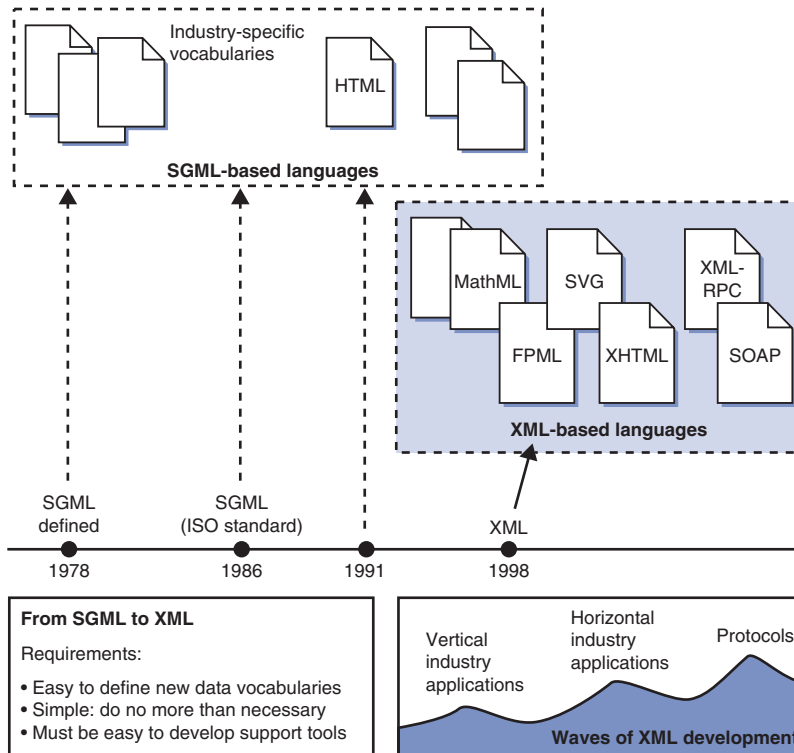
Although XML is a relatively new technology, its lineage extends back over several decades. Approved by the W3C in 1998, XML is an effort to simplify the Standard Generalized Markup Language (SGML), which, until XML, was the ISO standard for defining data vocabularies. Technically, XML is a subset of SGML designed to facilitate the exchange of structured documents over the Internet. Although SGML, which became an ISO standard in 1986, has been widely used by organizations seeking to structure their documents and documentation (for example, the General Motors parts catalog), its pre-Web complexity has been the main stumbling block to its widespread use and acceptance by the Web community. Figure 1.8 illustrates the relationship between SGML and XML and shows some of the languages derived from each.

XML's origins are in SGML.

The designers of XML took the best parts of SGML and, based on their experience, produced a technology comparable to SGML but much simpler to use. In fact, simplicity and ease of programming were requirements imposed by the W3C on the Working Group responsible for the final XML specification.

Chapter 1 XML: Extending the Enterprise

Figure 1.8 XML is the successor to SGML. Both are metalanguages that are used to define new data-oriented vocabularies.



XML has emerged from a document culture.

The Code, Data, and Document Cultures

To understand XML's impact on the computing world, it's useful to place XML in perspective. As Figure 1.9 shows, XML comes out of a document culture that is distinct from the code and data cultures that are the hallmarks of the mainstream computer industry. The code culture is characterized by a focus on programming languages, beginning with FORTRAN and evolving through Algol to C, C++, and Java. The data culture is characterized by COBOL, data processing, and databases. Both the data and code cultures carry with them a built-in propensity to view the world through either a code or a data lens. From a code perspective, data is something to be transported by

XML: The Three Revolutions

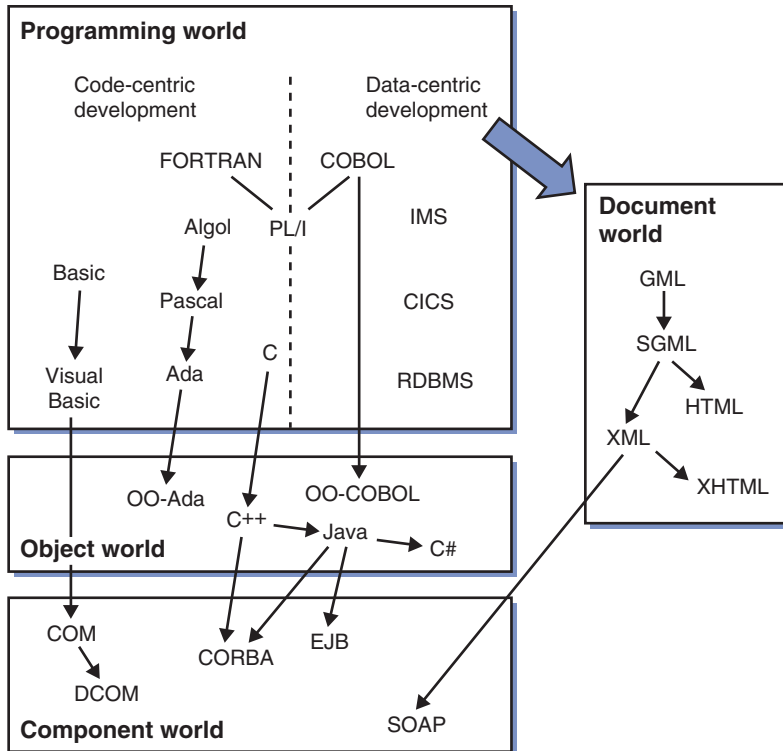


Figure 1.9 Evolution: from programming languages to objects to components.

procedure calls. From a data perspective, data is something to be stored in databases and manipulated.

The late 1980s and early 1990s saw code and data combine in the form of object-oriented languages such as C++, Smalltalk, Java, and Object COBOL. And yet, object technology was only a partial answer. As practitioners in the data world had long realized, transactions—the ability to update multiple databases in an all-or-none manner—are essential to serious industrial-strength enterprise applications. Because component frameworks provide transactions as a service to applications regardless of language origins, the playing field quickly shifted from objects to components. Thus infrastructures

Code and data have defined systems thinking.

Chapter 1 XML: Extending the Enterprise

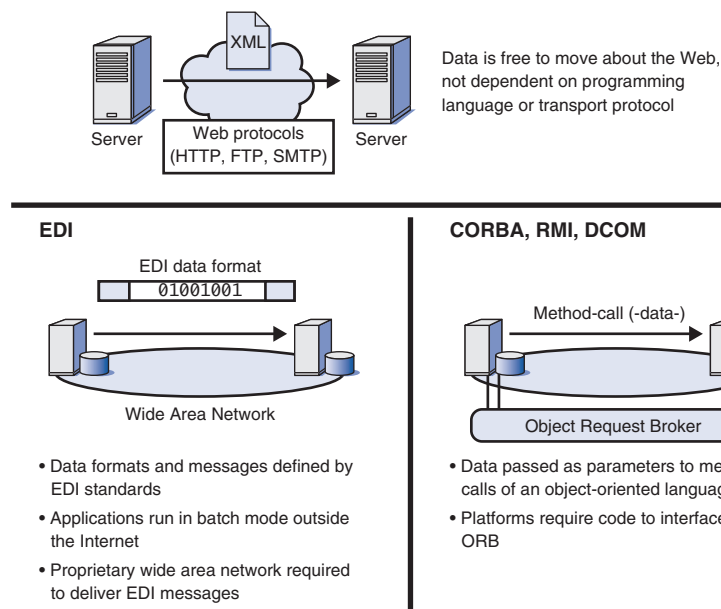
such as CORBA, DCOM, and Enterprise JavaBeans (EJB) provide interconnection, security, and transaction-based services for extending the enterprise. In the mid 1990s, components were the only way to extend legacy. However, XML changed the rules of the game.

XML opens up options for treating code as data.

XML's emergence from the data-oriented document culture has forced a rethinking about application development, particularly for those accustomed to thinking of building applications from a code-based perspective. What XML brings to the computing world is a technology that allows data to be freed from the constraints created by code-centric infrastructures. Instead of requiring data to be subordinated to parameters in a procedure call, XML now permits data to stand on its own. More radically, it allows code to be treated as data, which has been the driving force behind using XML for remote procedure calls. As Figure 1.10 illustrates, XML offers an alternative

Figure 1.10 XML in combination with Web protocols allows data to be independent of network, programming language, or platform.

The Data Revolution



to both EDI and technologies such as CORBA, RMI, and DCOM that lock data transfer into underlying networks and object infrastructures. It is this change in perspective that is driving the widespread use of XML across the entire computing industry and opening up new patterns of interaction, including Web services.

The Architectural Revolution

Together these XML-based technology initiatives open up new possibilities for distributed computing that leverage the existing infrastructure of the Web and create a transition from object-based distributed systems to architectures based on Web services that can be discovered, accessed, and assembled using open Web technologies. The focal point of this change in architectural thinking has been a move from tightly coupled systems based on established infrastructures such as CORBA, RMI, and DCOM, each with their own transport protocol, to loosely coupled systems riding atop standard Web protocols such as TCP/IP. Although the transport protocols underlying CORBA, RMI, and DCOM provide for efficient communication between nodes, their drawback is their inability to communicate with other tightly coupled systems or directly with the Web.

Loosely coupled Web-based systems, on the other hand, provide what has long been considered the Holy Grail of computing: universal connectivity. Using TCP/IP as the transport, systems can establish connections with each other using common open-Web protocols. Although it is possible to build software bridges linking tightly coupled systems with each other and the Web, such efforts are not trivial and add another layer of complexity on top of an already complex infrastructure. As Figure 1.11 shows, the loose coupling of the Web makes possible new system architectures built around message-based middleware or less structured peer-to-peer interaction.

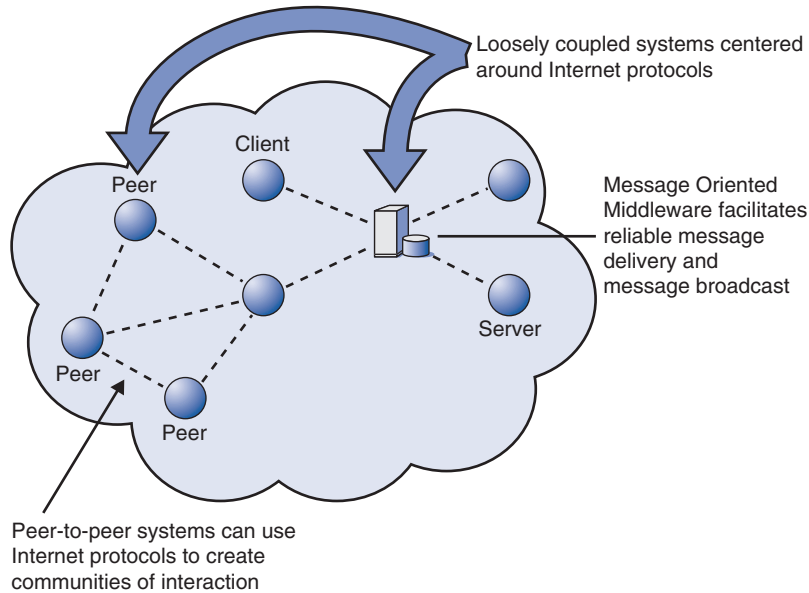
Simplicity and the ability to combine different standards are driving forces behind W3C deliberations.

XML and the Web have enabled the loose coupling of software components.

Chapter 1 XML: Extending the Enterprise

Figure 1.11 XML in combination with Web protocols has opened up new possibilities for distributed computing based on message passing as well as peer-to-peer interaction.

The Architecture Revolution



The Software Revolution

XML is also part of a revolution in how we build software. During the 1970s and 1980s, software was constructed as monolithic applications built to solve specific problems. The problem with large software projects is that, by trying to tackle multiple problems at once, the software is often ill-suited to adding new functionality and adapting to technological change. In the 1990s a different model for software emerged based on the concept of simplicity. As Figure 1.12 illustrates, instead of trying to define all requirements up front, this new philosophy was built around the concept of creating building blocks capable of combination with other building blocks that either already existed or were yet to be created.

The Web is an example of the power of combination.

A case in point is the Web. After decades of attempts to build complex infrastructures for exchanging information across distributed networks, the Web emerged from an assemblage of foundational

XML: The Three Revolutions

The Software Revolution

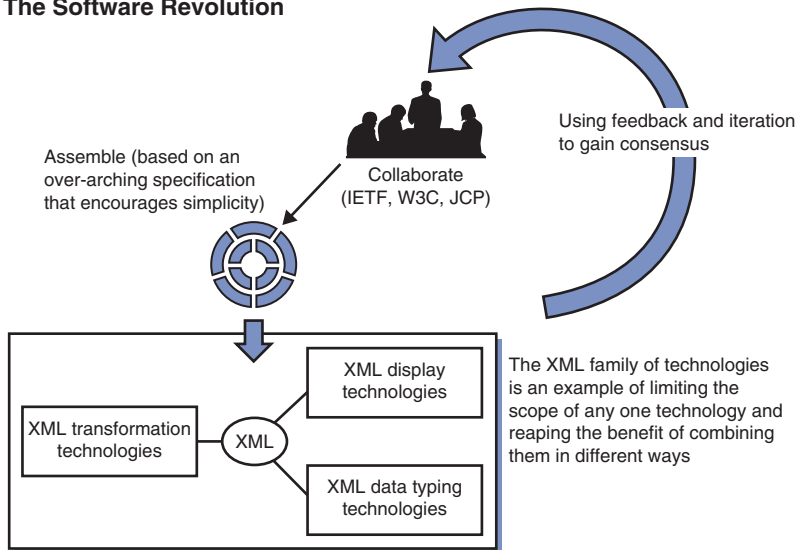


Figure 1.12 The software revolution: simplicity and collaboration.

technologies such as HTTP, HTML, browsers, and a longstanding networking technology known as TCP/IP that had been put in place in the 1970s.

Figure 1.13 illustrates how the Web as we know it was not something thought out in strict detail. Each of the contributing technologies focused on doing one thing well without inhibiting interconnection with other technologies. The essential idea was to maximize the possibility of interaction and watch systems grow. The result is the Web, a product of the confluence of forces that include the Internet, HTML, and HTTP. Let's now look at how these same forces of combination and collaboration are driving the revolution in software.

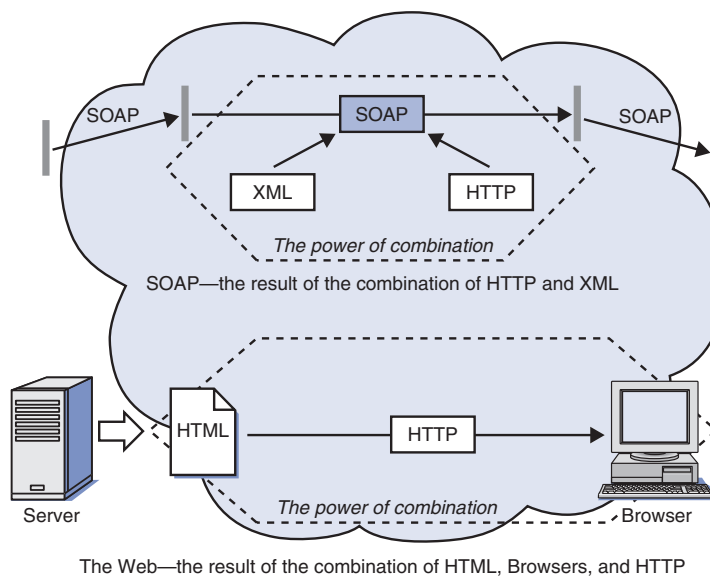
Software and Surprise

One byproduct of this new way of thinking about software combination is the element of surprise. Conventional software built around an ongoing series of requirements poses few surprises (except if it comes in under budget and on time). The Web, however, was different. It

Combination often leads to surprise.

Chapter 1 XML: Extending the Enterprise

Figure 1.13 The Web itself is an example of combinatoric simplicity in action. HTTP, a simple protocol, combines with browser technology to give us the Web as we know it today.



Design Principles

While Tim Berners-Lee gets the credit for assembling the pieces that ultimately composed the Web, a look at what Berners-Lee has to say in a 1998 article entitled “Principles of Design”¹ sheds some light on the thinking behind the software revolution. In this article, Berners-Lee focuses on several fundamental principles that are driving a new way of creating software.

- **Simplicity:** Often confused with ease of understanding, simplicity refers to the ease with which a concept or construct achieves its goal.
- **Modular design:** When you want to change a system (and change is inevitable), modular design lets you

¹ Tim Berners-Lee, “Principles of Design,” 1998, <http://www.w3.org/DesignIssues/Principles.html>.

introduce change with minimal impact on the workings of other system components.

- ❑ *Decentralization*: Systems should be constructed so that no one element serves as a single point of failure for the entire system.
- ❑ *Test of independent invention*: This involves a thought test. If someone else had invented your system, would theirs work with yours? This property means that in design, you should try to do one thing well and avoid having to be the center of the universe.
- ❑ *Principle of least power*: Computer science from the 1960s to the 1980s put great effort into constructing systems as powerful as possible, systems that tried to do it all. The principle of least power asserts that less powerful solutions ultimately are better suited for analysis and manipulation by applications yet to be invented. One hundred years in the future, software will probably have an easier time figuring out the content of an XML document than of a C++ program.

took just about everyone by surprise. Like a chemical reaction, the elements reacted in combination, giving rise to totally new structures.

Another example of the power of combination and surprise is Napster, a radical way of distributing music over the Internet. Napster relied on peer-to-peer connectivity rather than centralized distribution. Napster wasn't the result of a team of dedicated software professionals, but was created by a twenty-something upstart drawing on the power of assembly. The music industry will never be the same.

Napster was a surprising Web-based application.

Collaboration is being applied to requirements and design.

Combination and Collaboration

The power of combination is finding its way not only into software construction but up the development chain to software specification and design. Rather than hoping to meet the needs of users, design is now more collaborative, bringing in stakeholders early to ensure maximum feedback and the benefits of collaborative thinking. Figure 1.14 illustrates how this collaborative model is used by the W3C, the Internet Engineering Task Force, and Sun in its Java Community Process.

Figure 1.14 Part of the software revolution includes collaboration on specification and design. Examples include the Internet Engineering Task Force, the W3C, and Sun's Java Community Process.

Collaboration in Software Specification and Design

Internet Engineering Task Force



- Formed in 1992
- Charter: build an international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet's architecture and operation
- Standards include those for Internet infrastructure including security, routing, and user services

World Wide Web Consortium



- Formed in 1994
- Charter: lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability
- Standards (published as W3C Recommendations) include XML, XSLT, CSS, HTML, XHTML, DOM, and XML namespaces

Java Community Process



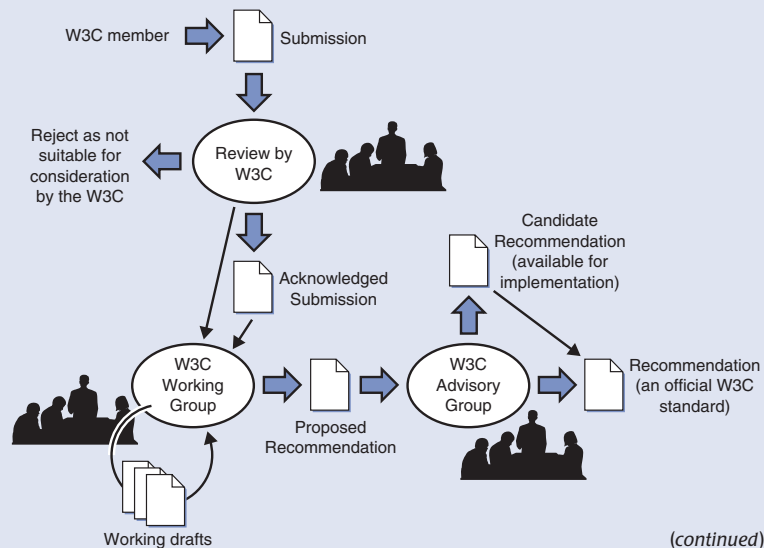
- Formed in 1995
- Charter: develop and revise Java technology specifications, reference implementations, and technology compatibility kits
- Standards are based on Java Specification Requests which describe both proposed and final specifications for the Java platform. There are currently over 100 JSRs including standards for Java 2 Enterprise Edition and Java Server Pages

The W3C

Regarding standards from the W3C, it's important to realize that the word "Recommendation," in W3C parlance, means final specification or standard. Understanding the W3C's process in moving from idea to Recommendation is important in tracking where the Web is going. Having the status of an approved Recommendation means that software vendors and developers can be confident that the technology described in the Recommendation has general industry-wide consensus.

There are several steps along the W3C path from submitting a proposal to ultimate approval as an official Recommendation, as Figure 1.15 illustrates.

- **Submission:** Any W3C member may submit a document to the W3C for possible review. A Submission indicates only that the document has been placed in a W3C in-box. It says



The W3C is the driving force behind open Web standards.

Figure 1.15 The W3C approval process from Submission to Recommendation.

The W3C (continued)

nothing about what the W3C thinks about it. The next step is for the W3C to determine whether it warrants further consideration as an Acknowledged Submission or should be dropped. This decision is based on whether the Submission is within the scope of the W3C charter.

- ❑ **Note:** A Note is a W3C document that has followed a formal submission process and gets an official date stamp. It carries no commitment on the part of the W3C to pursue the work any further.
- ❑ **Acknowledged Submission:** A Submission or Note that has been reviewed by the W3C becomes an Acknowledged Submission, which results in the formation of a Working Group, typically composed of the member group that authored the original Submission plus any other interested parties. The Working Group is tasked with producing Working Drafts that go up for public review.
- ❑ **Working Draft:** Working Groups produce Working Drafts. A Working Draft is a document in progress. When consensus is reached within the Working Group, a Proposed Recommendation is released. Often a Working Draft will be implemented by vendors who provide feedback to the Working Group about the viability of the proposed idea.
- ❑ **Proposed Recommendation:** The Working Group's consensus is formulated in a Proposed Recommendation that is sent to the W3C Advisory Committee for review.
- ❑ **Candidate Recommendation:** For complex proposals, the W3C Advisory Committee may release the document as a Candidate Recommendation, which indicates that there is consensus within the Working Group but that the W3C would like additional public review and feedback,

particularly from implementers of a specification. These developers also get a head start in bringing the technology to market before it acquires Recommendation status.

- ❑ **Recommendation:** A Recommendation represents consensus within the W3C that the idea is ready for prime time. Developers can be confident that a Recommendation will remain stable and that software can be built around it.

Summary

The simplicity of XML in combination with the Web has opened up new possibilities for moving data and for building new application architectures centered around common Internet protocols. Some of the changes brought about by XML include:

- ❑ Reduced dependence on proprietary data formats for applications
- ❑ A new way to do B2B data exchange using XML instead of the formats defined by traditional EDI systems
- ❑ A shift from relying on tightly coupled systems such as CORBA, RMI, and DCOM to a more loosely coupled Internet-based framework centered around XML and SOAP
- ❑ A change in focus from object-oriented to service-oriented software
- ❑ The emergence of Web services as technology for discovering and connecting to Internet-based services
- ❑ A move away from monolithic applications that attempt to do it all to a more organic software model that derives new capabilities from the combination of well-defined, limited software modules
- ❑ The consolidation of the software industry around two competing architectures, Microsoft's .NET and Sun's J2EE, specifications implemented by many of the major middleware vendors including IBM, Sun, BEA, Oracle, HP, and others

Chapter 1 XML: Extending the Enterprise

Placed in context, these changes reflect a major shift in the software industry from monolithic applications to applications built up from constituent pieces in an environment that fosters open, collaborative development.

Resources

Article

Jon Bosak and Tim Bray, "XML and the Second-Generation Web," *Scientific American*, May 1999.

Two Web pioneers discuss how hypertext and a global Internet started a revolution and how XML is poised to finish the job. A convenient online version of their article with hyperlinks is available at <http://www.scientificamerican.com/1999/0599issue/0599bosak.html>.

Web

<http://www.simonstl.com/articles/civilw3c.htm>

"An Outsider's Guide to the W3C," authored by Simon St. Laurent, offers a list of frequently asked questions about the W3C, providing insights into its workings and philosophy.

<http://www.w3.org/DesignIssues/>

Hosts a broad-ranging collection of personal notes and articles written by Tim Berners-Lee over the past decade describing the architectural design philosophy behind the Web.

<http://www.oasis-open.org/cover/xml.html>

"The Cover Pages," an encyclopedic Web site of news and articles about every aspect of XML maintained and updated daily by Robin Cover.

<http://www.webservices.org/>

A vendor-neutral site with hundreds of links to news and articles about all aspects of Web services.