submitted to "The Internet Encyclopedia," Hossein Bidgoli (editor), John Wiley & Sons

Machine-To-Machine Communication for Electronic Commerce

Franz J. Kurfess,[1] Leon Jololian,[2] and Murat Tanik[2]

[1] Computer Science Department
California Polytechnic State University
San Luis Obispo, California, USA
Email: fkurfess@csc.calpoly.edu

[2] Computer Science Department
University of Alabama at Birmingham
Birmingham, Alabama, U.S.A.
ljololia@eng.uab.edu, mtanik@uab.edu

Abstract

Many commercial activities rely on services performed by or with the help of computer systems. This often requires the exchange of information between computers, with precisely defined formats and protocols. While the Electronic Data Interchange (EDI) protocol has been in use for some time, the wide adaptation of the Internet and the World Wide Web has initiated more flexible methods of exchanging such documents. Many of these methods utilize the eXtensible Markup Language (XML), and various frameworks such as RosettaNet or ebXML are being put in place to. They are often combined with Web services, supported by technologies such as Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI), and the XML-based Web Services Description Language (WSDL). This contribution discusses methods, protocol and technologies used for the exchange of data, information, and knowledge among computer-based systems. Since the technical aspects of communication and interaction protocols are already reasonably well established, the emphasis here lies on the semantic aspects of machine-to-machine communication: How can computers interpret the contents of documents sufficiently well to perform the activities on these documents required by the respective business processes?

Table of Contents

Introduction

Machine to machine communication (M2M) is a fundamental issue whose resolution is critical to our ability to apply computers to a wider range of problems in the realm of business, manufacturing, science, private life, and others. One example is to link the meter that measures the usage of electricity in a household to the computers at the power company in order to generate monthly bills automatically. Another example is to connect the computers of a manufacturing plant to the raw material suppliers to automate the on-time delivery of goods and lower the inventory costs. Where computers have been applied successfully to automate particular tasks, M2M will allow us to increase the level of computer automation by allowing information gathered or generated by these individual tasks to be shared. Furthermore, M2M will allow us to automate processes whose tasks may be distributed. For successful implementation of M2M, three fundamental areas must be addressed: Communication protocols, Semantic protocols, and Interaction protocols. To make basic (syntactical) communication possible one needs to adhere to a common communication protocol, guaranteeing that the information packages transmitted are formed and transported according to the rules of the protocol. To make a meaningful (semantic) communication possible one needs to adhere to a common semantic protocol. The semantic protocol makes sure that the content of the information is structured in such a way that the parties involved can utilize it. To be able to interact, basic rules of engagement should be laid out and adhered to, as specified in a common interaction protocol.

Motivation

Most transactions between business partners are accompanied by an exchange of documents that capture the relevant aspects of the transaction. Examples of such documents are purchase orders, invoices, or bills of lading, but often also ones that conduct financial transactions, such as checks or money orders. Traditionally, these documents have been paper-based, and delivered physically from the sender to the recipient through direct delivery or intermediate services such as mail. Nowadays, most of these documents are generated and processed with the help of computers, even if the actual delivery of the documents is still in its physical form. The delivery and processing of paper-based documents accompanying business transactions can cause various problems: There is a major delay during the transmission of the document from sender to recipient; documents may be damaged or lost during the various stages of transport; documents have to be handled physically at various stages; the data contained in the documents may have to be entered into their computer system by the recipient, and possibly by intermediaries. This leads to delays in the transactions themselves, to increased costs for the sender and the recipient, and to uncertainties about if and when the documents were received. Especially since these transactions are eventually processed via computers in many organizations anyway, the benefits of eliminating the physical delivery of documents have become ever greater. Of course this also generates problems of its own, mostly related to compatibility across computer systems and applications, trust and security, and the willingness and capability of business partners to eliminate physical delivery. One of the cornerstones in exchanging documents between computers is the role of protocols and standards that specify exactly how these documents are structured, encoded, and transmitted in order to enable computers to process the documents without or with only minor computer intervention (such as the authorization of a transaction, for example).

Protocols in generals can be defined as a set of conventions or rules. It has been an engineering practice to break the communication task into layers of protocols responsible for layers of sub-tasks. In this model, each layer also has its own protocol. Generally, such a layered protocol set is called protocol suite or architecture. The presentation and complexity of the M2M process is reduced by viewing the tasks of M2M from the angle of our three types of protocol classes. It should be noted, however, that we are not proposing another layered-architecture, we are simply introducing a classification of existing conventions and protocols for the purpose of a clear presentation of M2M.

In this chapter we will address Machine-to-Machine communication, with and emphasis on the exchange of documents that accompany or constitute business transactions. After a brief overview of general aspects of M2M, we will examine the role of Electronic Document Interchange (EDI), an early set of standards and protocols for the exchange of documents via computers that defines the structure of such documents. Then we will discuss more recent approaches using the above conceptual protocol classification, distinguishing between communication, interaction, and semantic protocols. This is different from the layered communication architecture generally discussed in the literature (for example, the ISO-OSI layered model [ISO-OSI]), which concentrates on the technical aspects of communication protocols at various levels.

M2M Essentials

For two parties to communicate successfully, they need to have an agreement about the way in which the information between them is transmitted (the communication protocol{ XE "communication protocol" }), they should have a common understanding of the contents of the messages, and there has to be an awareness of the context in which the messages can be exchanged and understood. At this level, the emphasis is shifted from the transmission of data and information to the exchange of more complex and meaningful abstract structures, such as documents. Structured document types representing invoices or purchase orders, for example, will provide the context needed to process the information and thereby lay the basis for meaningful M2M communication [Banerjee & Kumar, 2002]. The establishment of communication and interaction protocols is a necessary condition for progress toward semantic communication between machines. At the level of communication protocols, consortia of standards organizations, vendors, and users of communication devices establish guidelines and standards for communications between machines. Some of the major organizations contributing to this process are ISO (International Standards Organization) [http://www.iso.org/], CCITT/ITU (The Consultative Committee for International Telephony and Telegraphy/International Telecommunication Union) [http://www.itu.int/], ANSI (The American National Standards Institute) [http://www.ansi.org/], IEEE (The Institute of Electrical and Electronic Engineers) [http://www.ieee.org/], EIA (The Electronic Industries Association) [http://www.eia.org/], and ETSI (European Telecommunications Standards Institute) [http://www.etsi.org/]. At every layer of the communication architecture, consortia of interested parties have developed numerous commonly used communication protocol standards. Among these communication standards are the ISO-OSI (International Standards Organization-Open System Interconnect) seven layer model [ISO-OSI], ATM (Asynchronous Transfer Mode) [Siu & Jain, 1995], HTTP (HyperText Transfer Protocol) [Albert, 2000] used for the World Wide Web (WWW), and TCP/IP (Transmission Control Protocol/Internet Protocol) [Comer, 1995], the collection (or suite) of networking protocols that have been used to construct the global Internet. Obviously, there are numerous publications discussing various aspects of these communication protocols. The goal of this contribution is to "connect the dots" in the sense that to accomplish meaningful M2M communication one needs to essentially understand the semantic protocols and the contributing rules and standards towards establishing a context in which the messages can be exchanged, understood (albeit in a very limited way) and processed by computers with no or very limited human interaction.

Electronic Data Interchange (EDI)

Based on efforts dating back to the 1960s and 1970s, there are now two major standards that govern the exchange of documents between computers, typically referred to as Electronic Data Interchange{ XE "Electronic Data Interchange" }, or EDI{ XE "EDI" \t "*See*" }{ XE "EDI" \t "*See* Electronic Data Interchange" } [Chan, 1997]. One standard has been developed under the auspices of the American National Standards Institute (ANSI) [http://www.ansi.org/], which chartered the Accredited Standards Committee X12 to develop a specification for the electronic transmission of documents. This standard is referred to as ANSI ASC X12{ XE "ANSI ASC X12" \t "*See*" }{ XE "ANSI ASC X12" }, and describes the information that needs to be included in a document, the structure of the document, and the use of codes and identification numbers that describe specific elements in those documents. On a global basis, the United Nations established the United Nations Electronic Data Interchange For Administration, Commerce and Transport (EDIFACT{ XE "EDIFACT" }) group [UN/EDIFACT], which also involves the International Standards Organization (ISO) [http://www.iso.org/] and the United Nations Economic Commission for Europe (UNECE) [http://www.unece.org/]. The EDIFACT standard is a combination of the ASC X12 standard and the Trade Data Interchange (TDI) standard used in Europe.

Both the ASC X12 and the EDIFACT standards explicitly define the structure of documents (such as a purchase order, invoice, shipping notice, etc), plus the format of data segments (roughly a line in a document, with information such as the ID number of an item, its description, the quantity, price and total amount for that line) and individual data elements.

The transmission of EDI documents starts with the translation of the original document generated on the sender's computer system, usually with the help of an EDI translator component. This document is then packaged into an EDI envelope, and transmitted via modem or the Internet. The actual transmission may be directly from the sender to the recipient, or through intermediaries that set up Value-Added Networks (VANs) with electronic mailboxes for their customers. At the recipient's side, the document is extracted from the EDI envelope, translated into a format compatible with the recipient's computer system and application, and then processed accordingly.

One of the fundamental problems for EDI is its inflexibility. Since it is developed with a very broad scope, it must govern a large variety of documents. On the other hand, the standard bodies prescribe and control the detailed structures of the documents, leaving little room for interested parties to develop their own, more appropriate solutions for tasks that may be specific for their particular domain. This flexibility is one of the major attractions for approaches based on XML, which will be discussed below. An integration of EDI and XML is the goal of the XML/EDI working group{ XE "XML/EDI working group" } [Bryan, 1998]. Although a substantial part of the technical aspects of EDI can be handled by appropriate computer programs or with the help of intermediaries, the implementation of EDI is a substantial task that may challenge the resources and capabilities of an organization. On the other hand, it can offer long-term benefits that quickly justify the initial costs and efforts, freeing up resources for advanced tasks than re-entering data from paper documents.

Semantic Protocols

Communication, interaction, and semantic protocol{ XE "semantic protocol" }s collectively are sufficient to achieve meaningful and context dependent message exchange. It is essential to understand semantic protocols in the context of M2M communication. Communication and interaction protocols have a longer history of use, and naturally fall into their places once semantic protocols are understood. Therefore, we will introduce in some depth semantic protocol standards and procedures that collectively constitute a basic set for M2M communication. Our discussion starts with the eXtensible Markup Language (XML) [Bray et al., 1998], which provides the basis for a number of electronic business frameworks such as RosettaNet or ebXML. Then we will examine the role of ontologies and metadata for semantic protocols. From this basis, we will explore the use of these technologies and concepts for machine-to-machine communication.

Extensible Markup Language{ XE "Extensible Markup Language" } (XML)

XML{ XE "XML" \t "*See* Extensible Markup Language" } is a markup language for documents containing structured information [Goldfarb & Prescod, 2002]. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to arbitrary documents. XML allows the definition of tags for domains and applications. These tags describe certain aspects of parts of a document, such as the <H1> … </H1> tag to identify a heading in a HTML document. There are two major differences that distinguish the use of tags in XML and HTML: First, HTML tags are used primarily for syntactical purposes, such as formatting, whereas XML tags are intended to impose a meaningful internal structure on a document. Second, the set of tags that can be used in HTML is restricted, and defined in the HTML standard set by the W3C [http://www.w3c.org/] governing body of the World Wide Web. XML allows interested parties to define their own set of tags, based on their particular needs. This provides much greater flexibility, but still requires an agreement about the sets of tags used in a particular domain, or among a network of parties that want to establish communication.

XML document{ XE "XML document" }s consist of sets of nested open and close tags, and tags can have attribute-value pairs. Figure 1shows the tags of a document representing an invoice; a complete XML document also has some information about the version of XML used, and a reference to the Document Type Definition{ XE "Document Type Definition" } (DTD{ XE "DTD" \t "*See* Document Type Definition" }) or schema that defines the tags (see also *Figure 3*, p. 23). A valid XML document corresponds to a labeled tree, with a tag for each node.

XML Document Tags
<**Invoice**> <**Buyer**> Smith, Inc. </**Buyer**>

<**Ordernumber**> 0001923</**Ordernumber**>

<**ItemNumber**> 36-0198QA. </**ItemNumber**>

<**Quantity**> 3 </**Quantity**>

<**UnitPrice** >85.26 </**UnitPrice**>

<**Total** > 255.78 </**Total**>

</**Invoice**>

Figure 1 Tags in a simple XML document

For a particular document, there usually exist several possible XML descriptions. XML is often used in conjunction with Document Type Definitions (DTDs), which specify admissible combinations of XML constructs, or XML Schema{ XE "XML Schema" } definitions, which also define a grammar for XML documents, but are more flexible than DTDs.

XML is often used for the following purposes:

i. As a serialization syntax for other markup languages;
ii. As semantic markup of Web pages, in combination with XSL style sheets to display the elements of a page appropriately;
iii. As a method to define a data exchange format in cases where the intended meaning is already established among the exchange partners.

For our purpose here, the latter two cases are the more interesting ones.

Electronic Data Interchange (EDI)

The necessity to exchange data and information in a clearly defined way was recognized by some communities quite a while before XML was developed [Goldfarb & Prescod, 2002]. One of the protocols used for commercial transactions is the Electronic Data Interchange{ XE "Electronic Data Interchange" } (EDI) format [Chan, 1997, UN/EDIFACT]. While EDI{ XE "EDI" \t "*See* Electronic Data Interchange" } provides a way to structure and annotate data to be exchanged, it has become clear that XML is more flexible, and probably also easier to use. On the other hand, EDI is so widely used that it won't simply be replaced by XML-based solution, leading to a co-existence and integration of both approaches [Bryan, 1998].

RosettaNet

The need for computer-supported information exchange in businesses that are part of supply chains led to the formation of RosettaNet{ XE "RosettaNet" } [http://www.rosettanet.org/], a consortium devoted to the definition of an electronic business framework [Goldfarb & Prescod, 2002]. Based on a dictionary of IT products, the consortium creates guidelines known as Partner Interface Processes (PIPs). The PIPs formalize the dialog between computer systems, and are based on and conducted in XML. RosettaNet is strongly supported by the information technology industry, and is in use in a variety of domains where the problem of supply chain misalignment is especially eminent.

ebXML

On an even larger scale, the United Nations are involved in an effort to standardize terminology, the exchange of information through messages, and codes that are used to identify products and businesses. ebXML{ XE "ebXML" } [http://www.ebxml.org/] is an electronic business framework{ XE "electronic business framework" } based on XML. This is a substantial undertaking, and involves several separate specifications that together constitute the framework. On the other hand, the potential benefits of enabling computers to exchange information and execute business processes largely autonomously are also very tempting, especially with the backing of a major international organization.

Knowledge Exchange

The languages and protocols can be augmented by knowledge exchange{ XE "knowledge exchange" } capabilities. In this case, it is not sufficient to provide markups for entities to be exchanged. In addition to the markups for individual entities that describe how they are supposed to be handled, it is important to also convey information about the relationships between individual entities. Although the distinction is not always clear, the combination of items and their relationships is usually considered a critical aspect of knowledge, in contrast to data. In the area of Artificial Intelligence, various knowledge representation methods have been proposed and used, many closely related to rules, or derived from mathematical logic [Russell & Norvig, 1995]. Linguists, philosophers, and cognitive scientists, on the other hand, have been interested in the relationships of words and terms to each other, leading to the development of ontologies. Due to their critical importance for machine-to-machine communication, ontologies are discussed in more detail below.

Ontologies

An ontology{ XE "ontology" } provides an explicit formal specification for the terms used in a particular domain, and identifies relations among these terms [Gruber, 1993, Chandrasekaran et al., 1999, Gomez Perez & Benjamins, 1999, Uschold & Gruninger, 1996]. It can be used as a common framework among various parties interested in the domain{ XE "domain" }. In our context here, ontologies are typically used to establish a common framework of reference for the terms in a particular domain or task. The ontology identifies and formalizes the underlying structure of the information and knowledge about the domain. Due to the formalization, it can be represented and to some degree interpreted by machines, and enables the formal analysis of the domain. This allows an automated or computer-aided extraction and aggregation of knowledge from different sources and possibly in different formats (as long as the formats can be mapped to the ontology). So instead of using a tightly defined protocol that has to be strictly followed by all the participants in an information exchange, systems with different representation mechanisms and communication methods still can engage in communication. While the exchange of knowledge and information with the help of ontologies is very flexible, it is clearly not always the most efficient method; especially in situations where there are clear constraints on the format and contents of information to be exchanged, a less flexible protocol (such as EDI, for example) may be more appropriate. From a knowledge engineering perspective, ontologies can be very helpful with the reuse of domain knowledge, and for the separation of domain knowledge and software code that performs operations on that knowledge.

To a certain extent, ontologies can mirror class hierarchies, objects, relation, properties, and methods used in software development. The latter, however, usually reflect the perspective of software developers, whereas ontologies concentrate on aspects of the domain that are visible to all parties interested in a particular domain, in particular users of software applications.

From a human perspective, ontologies establish a shared understanding of a domain, based on the representation of a shared conceptualization of a particular domain. With respect to communication between machines, ontologies form the basis for a semantic interpretation of the terminology used to exchange information. In contrast to syntactical exchange schemes such as XML, where the intended meaning of terms must be established beforehand between the partners engaged in the communication, ontologies can be used to associate meaning to the terms used in the communication. This is critical for machine-based communication in e-commerce, allowing flexible information exchange protocols among the computer systems of business partners. The establishment of such a mechanism also enables better vertical integration of markets in e-commerce.

Purpose

From a knowledge-oriented perspective, ontologies provide reusable descriptions of relevant concepts in a domain. This is especially important in domains that are under constant evolution, such as the domain of computer software. An ontology can provide a flexible framework for the description, classification, and organization of software components of packages, allowing constant updating as new components are added. In the area of knowledge and information retrieval, ontologies enable the enhancement of search engines for syntactically different, but semantically similar words. This may not only yield more and better results for searches initiated by humans, but can lead to personalized knowledge management agents. Such an agent knows the context in which the user is looking for information, the formulations and results of previous queries, and the personal preferences of the user for the presentation of information, and thus is able to present the newly found knowledge in such a way that it is easy to use by a human.

Within the context of the design and development of complex systems, e.g. the establishment of e-commerce market places, the development of an ontology can provide a framework for the actual implementation work, as well as for the later usage in machine-to-machine communication. Especially since it is likely that the systems to be built will need to be expanded by incorporating additional concepts on a regular basis, it is very cumbersome to have to augment the internal structure of the software, such as the class hierarchy, to reflect these changes. The ontology can also be used to enhance the compatibility with related systems. In addition, it provides a learning environment for developers, allowing them to become familiar with the domain and the terminology without access to domain experts{ XE "domain experts" }.

Terminology

The definitions of the terms used to discuss ontologies here are mainly based on [Noy & McGuiness, 2001].

Ontology{ XE "Ontology" }: A formal and explicit specification of the concepts in the domain and their relationships with each other. The backbone of an ontology is often a hierarchy, although links "across" the hierarchical structure are usually also employed.

Concept{ XE "Concept" }: A distinguishable, meaningful entity in the domain, usually associated with a name that is used to identify the concept. Concepts are often also referred to as classes.

Relation{ XE "Relation" }: A connection between two or more concepts. Relations usually have names that indicate their intended meaning, and may have specific properties or restrictions.

Property{ XE "Property" }: A feature{ XE "feature" } or attribute{ XE "attribute" } that describes an aspect of a concept or relation. Properties are also referred to as slots{ XE "slots" } or roles{ XE "roles" }.

Facet{ XE "Facet" }: A restriction or constraint placed on a property. Facets are also referred to as role restrictions{ XE "role restriction" }.

In order to distinguish our discussion from the corresponding software development effort, we will use the terms concept instead of class, property instead of role, and facet instead of slot. The notion of concepts is the central focus of our ontology, and concepts are the most visible entities in it. Concepts together with their relations are often displayed as graphs, where nodes represent the concepts, and links the relations. Another common visual representation is as a tree, reflecting the hierarchical backbone of the ontology. In this representation, the ontology can be navigated in a similar way to the file system on a computer.

Design and Development Approach

A basic approach to develop an ontology{ XE "ontology" } for a domain is suggested in [Noy & McGuiness, 2001] and consists of the following steps:

1) Determine the domain and scope of the ontology
2) Consider reusing existing ontologies
3) Enumerate important terms in the ontology
4) Define the concepts and concept hierarchy{ XE "concept hierarchy" }
5) Define the properties of concepts (slots)
6) Define the facets of the slots
7) Create instance{ XE "instance" }s

Of particular relevance in our context are steps 3-6; they correspond to the identification of tags that should be provided for documents, and their specification. The actual creation of specific documents corresponds to Step 7.

Extensible Ontologies

For many domains, the vocabulary used changes over time. This leads to the necessity of creating the respective entities to capture these new terms. Consider our above example of an invoice: Although it is clear that our example doesn't cover all relevant terms, even a very elaborately worked out collection of tags would not have included a Web address field (URL) in the part that specifies the company's address as little as ten years ago. In combination with the definition of a new set of tags in XML, this requires the addition of new entries to an ontology [Swartout et al., 1996]. These entries may have to be added at lower levels, corresponding to domain-specific, detailed items, or at higher levels of abstraction when new areas are included. Ideally, these new concepts should be integrated into the ontology in collaboration between domain experts, knowledge engineers, actual users, and possibly other parties. In practice, this is unrealistic since it requires a revision of the underlying system, possibly involving the deployment of new versions to the customer, with all the delays and installation hassles. The other extreme is to let the user add concepts to the ontology as the need arises from the actual use of the system. This certainly is faster, but may lead to "wild" growth of the ontology since the user may not be aware of the underlying design principles chosen by the domain expert or knowledge engineer. The distributed nature of e-commerce exacerbates this problem even more: Without a flexible mechanism that allows the introduction of new terminology (e.g. for the description of new products or new procedures, or the addition of new partners), all systems involved will need to be updated individually. With a mechanism in place, the new entities can be distributed to all partners, and integrated into their existing frameworks.

Class Hierarchies

The classes in object-oriented software development as well as the concepts in an ontology are often arranged in hierarchical structures, indicating a similarity between the two approaches. The purpose for which the hierarchies are used, however, is quite different: Object-oriented systems rely on a class hierarchy{ XE "class hierarchy" } for the design of systems that exhibit a certain desired behavior, whereas a concept hierarchy{ XE "concept hierarchy" } used in an ontology serves as description of the relationships between concepts in a domain. Thus, the character of class hierarchies is more behavior-oriented, and that of ontologies more declarative.

Ontology Construction{ XE "Ontology Construction" }

For many applications, the construction of ontology from scratch may not be the best choice. In addition to being tedious for even mid-sized ontologies with more than a few handfuls of concepts, it might lead to complications in the case of distributed approaches or systems [Swartout et al., 1996]. An alternative is the use of existing ontologies as backbones, and the addition or modifications of concepts and relationships according to the specific domain. Thus, multiple teams can work on different parts of ontology, and separate ontologies that may have to interact have a common basic structure.

Identification of Relevant Terms and Concepts

One of the first steps in the construction of ontology is the identification of terms{ XE "identification of terms" } that need to be present. The traditional approach here is to consult a domain expert in order to elicit the most important terms and concepts. A core set is then used as "seed" for the ontology, and less important ones are integrated gradually. This approach is reasonable for the initial phase to guarantee a solid foundation, but may be too expensive or too tedious in later stages. Then, new concepts can be identified by the users of the system, who either integrate them on their own, or with the help of domain experts and knowledge engineers. Alternatively, the identification of new terms can be based on the search of computer-based documents that are used in the context of the system. System manuals, user guidelines, reports, or other documents are searched for terms that are not too general in nature, and thus may be relevant for the particular domain. In addition, software artifacts such as databases, data structures, source code, or requirements specifications can be subjected to such a search. The problem here is to eliminate general terms that do not have a specific meaning within the context of the domain, while not overlooking general terms that are used in a very specific sense.

For M2M, the identification of important terms is often done by standard organizations representing a particular domain, or by companies that offer such services on a commercial basis. In practice, the development of the corresponding ontology is often implicit, and is reflected in the DTD or schema that specifies the structure of documents in that domain. An alternative approach is to explicitly develop an ontology of the domain under consideration, and use that as the starting point for the specification of DTDs and schemata.

Addition of New Concepts

The most significant changes for the adaptation of an existing ontology to a particular domain will probably be achieved through the addition of new concepts. This may involve the introduction of new terms describing new concepts, or the expansion of already present terms to cover additional concepts. The crucial decision here is to determine the right place in the ontology for a new concept. This decision should be guided by the underlying principle used for the overall layout of the ontology. A domain expert should make the final decision, although this poses some practical problems with availability of experts and possible delays.

Element	Description
Domain{ XE "Domain" }	domain under consideration
Concept definition{ XE "Concept definition" }	clarification of the meaning for the respective concept
Documentation	explanatory text, images or other data, links to relevant source documents
Relation{ XE "Relation" }s with other concepts	links to other concepts, including link type
Hierarchy{ XE "Hierarchy" } information	super-concepts{ XE "super-concepts" }, sub-concepts{ XE "sub-concepts" }, sibling concepts{ XE "sibling concepts" }
Properties{ XE "Properties" } and facets{ XE "facets" }	important aspects (roles) of a concept or relationship, together with constraints and restrictions
Instance{ XE "Instance" }s	existing instances of the respective concept

Figure 2: Important properties of concepts in an ontology

At a first glance, it may seem that for many situations, an ontology in addition to a well-defined DTD{ XE "DTD" } or schema{ XE "schema" } is not necessary for machine-to-machine communication. The development of DTDs and schemata, however, has a substantial overlap with the development efforts for an ontology. Thus it is possible, with moderate overhead, to simultaneously develop the XML specification framework and the corresponding ontology. The availability of an ontology enables the treatment of more complex situations, e.g. in cases where communication involves the translation of documents relying on different DTDs, or the incorporation of unmarked text.

Metadata

Metadata{ XE "Metadata" } provide information about the actual data items themselves, such as the table names and field names in a database, or the names, and properties associated with a file in a directory. Metadata to some degree have been used for a long time to make the utilization of the actual data easier or more convenient: Sorting the files in a directory by size, creation date, or even by their names, relies on metadata. Especially in connection with XML, the role of metadata becomes more prominent. XML defines a method to mark up data items, and separates the internal structural aspects of a document from the way the document is presented to the user. The actual XML markers, or tags, however, are not very useful unless they are accompanied by a description of how they are supposed to be interpreted. This description is usually delivered in the form of a style sheet{ XE "style sheet" }, or through a schema{ XE "schema" }. The style sheet provides instruction for the role and interpretation of the specific markers, and two systems that communicate with each other must either use a common one, or use different ones that have a clearly defined relationship with each other.

Resource Description Framework{ XE "Resource Description Framework" } (RDF)

RDF [Brickley & Guha, 2000, Lassila & Swick, 1999, RDF] is a standard proposed by the W3C consortium to enable the exchange of metadata, and is intended to be used for the description of documents and resources on the Web. In contrast to XML, which is essentially a means to define grammars, and allows the syntactical characterization of documents, RDF is designed for the semantic characterization of documents. RDF employs object-attribute-value triples as basic building blocks: object O has an attribute

A with the value V. The commonly used notation A(O, V) indicates that there is a relation expressed through the attribute A between object O and the respective value V for the attribute. An RDF triple can also be viewed as a labeled edge A between the two nodes O and V. In its basic form, RDF offers a data model for metadata through its <object, attribute, value> triples. It does not provide specific capabilities to define domain-specific names and terms. This is done by the vocabulary definition facility RDF Schema{ XE "RDF Schema" }, which can be used for the specification of the terms to be used in the domain, and the types of objects to which the terms can be applied. RDF is also used for the specification of ontologies [Staab et al., 2000, Decker et al., 2000], thus constituting an important construction method for the Semantic Web{ XE "Semantic Web" }.

10/24/02

Exchanging Information and Knowledge between Machines

In its current incarnation, the Internet and the World Wide Web serve mainly as a sophisticated infrastructure that enables humans to perform business transactions more efficiently. In most cases, these transactions are initiated and supervised by humans, and carried out at varying levels of sophistication by machines. Even at this semi-automated stage, it is becoming clear that the underlying protocols such as HTTP and TCP/IP must be augmented by mechanisms that operate at a higher level of abstraction [Albert, 2000].

Using XML for Machine-to-Machine Communication

For many Web-based systems, XML{ XE "XML" } is already superseding HTML, due to its far higher flexibility, and the possibility to separate content from presentation. However, at its very core, XML is a mechanism that is intended for the syntactical characterization of entities to be transferred over a communication medium such as the Internet. It is essentially a mechanism for defining a grammar, and ensures syntactic interoperability between systems exchanging information. This means that the information to be exchanged is prepared in such a way that it is straightforward for the sending side to encode it in the common format, and just as straightforward for the receiving side to convert it from the common format into the one that it uses locally. In XML, this is done by defining tags that identify the entities to be exchanged. The definitions of the tags to be used are listed in a Document Type Definition{ XE "Document Type Definition" } (DTD) or XML Schema{ XE "XML Schema" }, and must be shared between all the partners involved in the exchange. The actual XML document then uses the tags to describe the values to be transmitted, and their arrangement. A simple example of a DTD with the instance of a corresponding XML document is given in Figure 3. Please note that there are many different ways of specifying the format for such a transaction, and the one used here is only intended to illustrate the basic concepts.

Document Type Definition (DTD)	XML Document
<!ELEMENT Invoice (Buyer,	<Invoice>

```
OrderNumber, Item, Quantity, Price,
Total)>
<! ELEMENT Buyer (#CDATA)>
<! ELEMENT OrderNumber
(#CDATA)>
<! ELEMENT ItemNumber
(#CDATA)>
<! ELEMENT Quantity (#CDATA)>
<! ELEMENT UnitPrice (#CDATA)>
<! ELEMENT Total (#CDATA)>
```

```
<Buyer> Smith, Inc. </Buyer>
<Ordernumber> 0001923</Ordernumber>
<ItemNumber> 36-0198QA. </ItemNumber>
<Quantity> 3 </Quantity>
< UnitPrice >85.26 </UnitPrice>
< Total > 255.78 </Total>
</Invoice>
```

Figure 3 XML Example with Document Type Definition (DTD)

The main advantage of using XML in such a situation is the availability of a precise format, which can be converted easily from or into specific applications by the parties involved. All parties involved, however, must have a common understanding of the meaning of all elements used in the DTD. For example, it is critical that there is an agreement on the currency used in the UnitPrice and Total elements of the Invoice. This common agreement can be achieved by establishing a jointly used DTD among the partners. If the consortium sharing the DTD has to be expanded, the situation may become more complicated. As long as the new partners can start using the same DTD (e.g. because they don’t have an established protocol), it is easy. If they already have a protocol or convention for the exchange of information, it becomes necessary to establish mappings or conversion mechanisms between the two domains. The basic problem is that the mapping has to be established on a semantic basis, i.e. between entities with the same intended meaning. In many cases, it will not be sufficient to perform this on a syntactical basis, i.e. by simply mapping elements from one DTD to another DTD. In the example above, a syntactic mapping would be between invoice DTDs that have exactly the same structure, but use different tags to indicate the respective elements (e.g. customer instead of buyer); this would be fairly easy to achieve. XML provides mechanisms to syntactically translate documents between different DTDs, e.g. through XSLT style sheets. The way an invoice is structured could be completely different, however, and it might be difficult to convert documents written in one format into the other format. This would require a reengineering of the source format, the establishment of the mapping from the source to the target format, and the actual implementation of the conversion mechanism.

In practice, standard bodies, professional societies, independent brokers, or major players in a particular field are often the driving forces behind the establishment of such protocols and conventions. From an implementation perspective, XML has the advantage of making the software mechanisms for parsing documents easy to reuse. XML has its limitations when the interpretation of the mechanism used to exchange information is not known in advance and shared by all parties involved, or if it has to be adapted and updated on a frequent basis.

Using RDF for Machine-to-Machine Communication

While RDF is not quite as popular as XML for information exchange purposes through the Internet, in comparison with XML it is significantly better suited for semantic interoperability. RDF uses <object, attribute, value> triplets as basic units. All objects are independent entities, and can be used as semantic units (they can be affiliated with some interpretation). Since domain models typically consist of objects and their relationships, RDF is a good match to encode such domain models. Whereas XML embeds information to be exchanged according to the syntactic rules lined out in the DTD, RDF operates at the meta-data level, providing a description of the structure of a particular domain through the objects and their relationships at a higher level of abstraction. RDF relies on declarative semantics, by providing interpretations of statements through a mapping to another, well-established formalism, such as types, classes, and subclasses. The identification of semantic-preserving mappings between RDF descriptions may still be relatively complex, but it can be done at the semantic level, which allows for higher reusability. Similar to the document type definitions (DTDs) or Schemas affiliated with XML, RDF has RDF Schema{ XE "RDF Schema" }s, a mechanism to define the vocabulary used. RDF Schemas are appropriate vehicles for the representation of ontologies, providing a natural encoding of the domain model in the formal framework. Using RDF Schemas, more specialized or richer languages can be defined on the basis of the RDF primitive constructs. On the other hand, RDF is not as widely used as XML, and at least initially the creation of RDF-based interchange mechanisms can be more difficult. While XML is rapidly gaining popularity in the Web community as an important step towards semantic integration, RDF is closer to the methods and techniques investigated by and used in the Artificial Intelligence community for the representation of knowledge [Decker et al., 2000].

Semantic Web{ XE "Semantic Web" }

The World Wide Web has had tremendous success in making information easily accessible to people by providing a set of mechanisms and tools to easily display and link interconnected collections of documents. The underlying protocols and languages, such as http and HTML, however, rely on humans to explicitly construct those documents and the links between them. It has not been designed for, and indeed is not very well suited for the automatic processing and manipulation of information by computers. The Semantic Web builds on the infrastructure provided by the World Wide Web, but augments Web pages with meta-data{ XE "meta-data" } in order to enable the automatic processing of information by computers [Berners-Lee, Hendler & Lassila, 2001]. Meta-data contain additional information about documents, and can be used by computers to obtain more information about the meaning of documents and terms. In combination with rules for reasoning about terms and documents, this enables computers to perform sophisticated Web services through intelligent agents such as information brokers, search agents, personalized information filters, etc. Although this will not be sufficient for computers to capture the meaning of terms, the Semantic Web offers an infrastructure for knowledge to be processed by computers, shifting much of the tedious scanning and "weeding out" of irrelevant information from humans to computers.

Knowledge Exchange Protocol{ XE "Knowledge Exchange Protocol" }s

Whereas the Semantic Web has gained a considerable amount of attention due to the popularity of the World Wide Web, related approaches to make knowledge more amenable to computers have been pursued for a long time in the Artificial Intelligence community, with expert systems as the most widely known instance of knowledge-based tools. The main concern of these approaches is the representation and manipulation of knowledge for reasoning purposes, and the exchange of knowledge between systems often is a secondary issue. Soon after the development and practical use of the main knowledge representation frameworks, it became evident that it would be very useful for such systems to communicate with each other. In the first place, this requires a common syntactical basis between the systems involved in the communication. Secondly, they must have a common understanding of the terms used in their exchanges of information. The syntactical problem can be solved reasonably well with moderate overhead: If the systems involved share the same knowledge representation{ XE "knowledge representation" } method, they can simply exchange fragments of their knowledge base. Otherwise, translation programs can be written to convert statements from one representation into another one. The more critical aspect is the second one, the common understanding of the terms: without the ability to attribute some meaning to the statements that are exchanged, the systems involved will face some difficulties. They must either be constructed in such a way that they automatically know how to deal with the terms in a statement, or they must have some way of determining the intended meaning of those terms. Humans ultimately provide this meaning; the main distinction here is the degree to which it can be automated. In some cases, such as EDI, messages to be exchanged must comply with a very specific format together with a prescribed vocabulary. This is only suitable for situations where the knowledge to be exchanged is very well structured, and the messages are usually fairly simple, albeit maybe voluminous.

A number of formal knowledge exchange language{ XE "knowledge exchange language" }s have been proposed and used in the AI community, and efforts are being made to integrate them with related approaches originating from the World Wide Web community. Some of the more popular ones are the Knowledge Interchange Format (KIF) [Genesereth & Fikes, 1992], and the Knowledge Query and Manipulation Language (KQML) [Labrou & Finin, 1996, KQML].

SHOE [Berners-Lee, Hendler & Lassila, 2001] is an example of a Web-based knowledge representation language that provides additional information about Web documents, thus rendering them more easily usable by computers. Explicitly designed for the exchange of knowledge between agents are so-called Agent Communication Languages. The DARPA Agent Markup Language (DAML) [http://www.daml.org/] enables agents to communicate with each other by providing directives for the exchange of messages, a format for the content of the messages, and guidelines on how to carry on a conversation.

Web Services{ XE "Web Services" }

In our discussion so far, the emphasis has been on the utilization of methods and technologies for the exchange of knowledge and information in such a way that computers can perform meaningful operations on the items they transmitted. While this is a necessary condition for machine-to-machine communication, it assumes that the parties involved already are well informed about what their partners have to contribute to the exchange. In some situations, this may be the case, e.g. the network of suppliers that regularly work with a particular car manufacturer. In general, however, such a well-established relationship with clearly defined products and services cannot be assumed to exist. In a traditional scheme, human involvement can of course provide the information to establish such a relationship between electronic business partners, and, once established, the techniques described in the previous sections still can be used favorably. This situation characterizes the current status of the Web: There are independent Web sites and stand-alone or loosely connected applications, depending on humans to bridge the gaps between these islands of functionality and data [Goldfarb & Prescod, 2002].As a simple example, take the purchase of a digital camera on the Web. As the user interested in buying a camera, you probably visit a number of Web sites that offer information about various models, such as consumer magazines or specialty Web sites with tests and comparisons, and the Web sites of the camera manufacturers. Then you may go to a Web site that provides you with price comparisons among various on-line stores, and finally you go to the Web site of the store you chose to purchase the camera from. On this Web site, you pay for your purchase with your credit card, which requires you to fill out a lengthy form with your address, credit card info, and so on. Although there is some support available through hyperlinks between Web sites, or from the Web browser that stores your address and credit card information, these are often crutches that fail easily. In combination with the methods and techniques described in the previous sections, *Web Services* (sometimes also referred to as XML Web Services, or e-services) enable computer systems to establish networks where data, information, knowledge, and affiliated products or services can be offered, brokered, and utilized with no or minimal human intervention, thus spanning the gaps between the islands of functionality and data.

Just like with the exchange of meaningful information, these services utilize some abstract methods to solve basic problems, and rely on specific techniques and technologies to implement appropriate and practical solutions.

In the rest of this chapter, we will describe the main concepts behind Web services, and discuss some of the more popular specific technologies that are used in systems that implement Web services. It is worth noting at this point that a substantial part of the approaches discussed here are in their early stages of deployment, and still have to pass the test of time and large-scale use. It is our belief, however, that the basic principles and main techniques are sound, and will be utilized more and more in the near future.

Basic Principles

Web services are build on top of many of the existing Web and Internet protocols such as HTTP and TCP/IP, and utilize XML as the underlying data representation framework. The basic idea is to integrate and coordinate multiple Web sites by providing an infrastructure that allows computer systems to search for, request, and provide services within their specific network. On such an infrastructure, systems can be built that interact seamlessly across individual programs or computer systems.

Web Service Technologies

The technologies used to implement Web services frequently rely on XML as their foundation. They help provide components that can be configured and reused into systems that coordinate multiple Web sites into what to the user appear as seamless online services [Goldfarb & Prescod, 2002]. In the following, we will briefly discuss a language that allows a systematic description of Web services, and two protocols for the discovery and access of services.

Web Services Description Language{ XE "Web Services Description Language" } (WSDL{ XE "WSDL" \t "*See* Web Service Description Language" })

As long as the number of services is relatively small, and it is known to the programmer or user where they can be obtained, programs can be written in such a way that they directly access a particular service at a specific machine. For example, a user can instruct a program to connect to a FTP (File Transfer Protocol) server, essentially by specifying the URL of that server. With a larger variety of services, some of them possibly offered at varying addresses, a more flexible approach needs to be embraced. The first step is the systematic description of Web services, specified in such a way that the services can be located and utilized without human intervention. This is the purpose of the Web Services Description Language (WSDL). WSDL uses XML{ XE "XML" } to specify various layers, with services as the highest layer, and complex or simple data types at the bottom [Goldfarb & Prescod, 2002]. A *service* comprises a set of operations that are accessible to potential clients. Examples could be services that provide up-to-date arrival times of flights, stock prices, or the status of orders. WSDL needs to be flexible in order to accommodate all kinds of existing and future services, while at the same time providing descriptions of services that can be used by computers and humans alike. Access to these operations is offered through *ports* specified as Web addresses (URLs). A *binding* affiliates a specific transport protocol with a port; this is similar to the way HTTP, FTP, and similar protocols are currently used. For more flexible scenarios, the Simple Object Access Protocol{ XE "Simple Object Access Protocol" } (SOAP) can be used. The three concepts of service, port, and binding describe the concrete implementation of Web services in WSDL. In addition, WSDL features the abstract definition of an interface for a Web services through the notions of port type, operation, message, part and type. The port type is the abstract interface to a set of operations provided by a particular service. The individual operations describe the behavior that can be expected from a service. They are characterized through their inputs and outputs, together with fault messages in case of errors. A service exchanges information via *messages* that contain XML structures. A message consists of one or more *parts* corresponding to elements specified in the XML Schema definition, or some other schema language.

Universal Description, Discovery, and Integration{ XE "Universal Description, Discovery, and Integration" } (UDDI{ XE "UDDI" \t "*See* Universal Description, Discovery and Integration" })

While XSDL provides the means to describe Web services, it does not offer much help in finding out where and who might offer a particular service. The discovery of services can be considered a service itself; the yellow pages of a phone book are an example of such a service. The creation of a directory of Web services is the goal of the Universal Description, Discovery, and Integration (UDDI) project [http://www.uddi.org/]. It uses WSDL to describe services offered by providers, and SOAP for the exchange of computer objects generated while the operations that constitute the service are performed.

Simple Object Access Protocol{ XE "Simple Object Access Protocol" } (SOAP)

SOAP is a specification that regulates the exchange of computer-generated objects [Goldfarb & Prescod, 2002]. Such objects can be used to describe properties and behaviors of entities represented in computer programs, ranging from simple ones such as the record of a single customer in a store, to the complex structure needed for a complete on-line store. Since most modern computer programming languages utilize objects, various methods to exchange objects between different programs and different computer systems have been developed. As long as this exchange takes place between programs developed in the same language, and running on computers with the same or compatible hardware and operating systems, there is no need for a separate protocol, as the internal representation is identical or compatible across programs and systems. In practice, it is also often desirable or necessary to perform such an exchange across programs and systems that use a different internal representation, requiring specific mechanisms to convert objects from one representation to another. Two relatively popular object communication specifications are the Component Object Model{ XE "Component Object Model" } (COM{ XE "COM" \t "*See* Component Object Model" }) [COM], mostly used with Microsoft products, and CORBA{ XE "CORBA" } [CORBA]. Since these models rely on the exchange of binary information, they require specific development and test tools, and have only been moderately successful for communication between separate computer systems. The alternative to the exchange of binary objects is to convert their contents into text-based messages, and transmit these from one machine to another. This leads back to the use of XML for the description of structured documents, and SOAP allows the specification of exchange methods for complex structured documents.

Intelligent Agent{ XE "Intelligent Agent" }s

Another frequently used term in a similar context as Web services is that of *intelligent agents*. An agent is a program that is mobile, and acts in a goal-based manner [Russell & Norvig, 1995]. It picks up percepts from its environment, selects an action that takes into account these percepts and its internal state, and performs the action, which results in some changes to the environment and its internal state. The critical difference between service and agent lies in the mobility and goal-based aspects: Services typically are offered and performed at a specific, fixed location, while agents may "roam" the network (provided the appropriate infrastructure is available). Moreover, services are usually programmed in a deterministic manner: They are under direct control of the user, and perform exactly those operations initiated by the user. Agents, in contrast, are given a goal, and use their own reasoning capabilities in order to achieve that goal. Of course, agents and services are not mutually exclusive: With the right set-up, agents may utilize services, act as brokers to identify suitable services, and provide services themselves. At some point, the distinction between the two may fade, and a particular system may incorporate aspects of both service and agent.

Conclusions

In this chapter, we addressed Machine-to-Machine (M2M) communication using a conceptual protocol classification that is different from the layered communication architecture generally discussed in the literature. The core of the discussion centered on semantic protocols since they constitute the heart of M2M, and are not as well known as the communication and interaction protocols. However, interaction protocols provide a capability extension to semantic protocols by establishing the standards for communication between machines. In order for two machines to communicate there must be agreed upon standards to specify how the exchange of information from one machine to another is to be carried out. A related issue is the discovery of information about the machines with which interaction is needed. This can be achieved through searches of network directories acting as yellow pages for information about machines and the services they can provide. Among these evolving standards are SOAP, UDDI, XML-RPC, and Web services frameworks.

While computers facilitate many tasks in our private and professional lives, they also contribute significantly to the generation of an overwhelming amount of information and knowledge accessible in digitized form. At a low, syntactical level, computers are remarkably efficient in performing tasks like searching for keywords in huge collections of documents. When it comes to the utilization of the content of documents, however, much of the work is still done by humans: We skim or read the documents to interpret their content, and then decide what to do with them. The essential tools and methods to make computers more effective at dealing with the content of documents either are available, or are in development now. By providing computers with additional information on the structure and content of digital documents, they can conduct far more complex transactions, without requiring a full understanding of the meaning of information conveyed in documents. The quick acceptance of XML as a flexible document markup language that can be used for various domains and tasks, and the development of the Semantic Web as an additional, content-oriented layer for the World Wide Web indicate that the potential for applying machine-to-machine communication to well-defined and novel tasks alike is being recognized. We believe this will result in better computer support for knowledge organization and management, and not only in yet another increase in the amount of information generated.

Glossary

Communication Protocol

A set of rules that describes the expected behavior of parties involved in the exchange of information. In this context, communication protocols refer to the way computer systems exchange messages over a network. Examples of communication protocols are wire protocols such as Ethernet, modem protocols (V.32, V.90), wireless protocols such as IEEE 802.11b, the Internet addressing protocol TCP/IP, or the HyperText Transfer Protocol (HTTP) of the World Wide Web. See also *interaction protocol, semantic protocol.*

Electronic Business Framework

The term framework in this context is rather vague, but usually covers design guidelines, best practice recommendations, vocabularies, and markup conventions for the annotation of documents. Examples of electronic business frameworks are ebXML supported by the United Nations, and Microsoft's BizTalk framework. These generalized, common frameworks are sometimes contrasted to vertical applications that are intended for particular domains or industries, such as RosettaNet.

Electronic Business XML (ebXML)

A standard supported by the United Nations intended to enable computers to perform international business transactions. It is based on XML, and comprises several different, but related specifications. Two key areas are the discovery of services and potential business partners through registries (e.g. via UDDI), and the actual exchange of documents between business partners through messages. ebXML messages contain documents consisting of core components that are linked to the actual business processes in the real world.

Electronic Data Interchange (EDI)

A set of standards that specifies the exchange of computer-based documents between business partners. It regulates the structure of the documents for various purposes such as purchase orders, invoices, or shipping notifications, and identifies commonly used codes and identification numbers. For more details, see the chapter on Electronic Data Interchange.

Extensible Markup Language *see* XML

Intelligent Agent

A computer program (software agent) or robot (physical or embodied agent) that performs tasks usually assumed to acquire intelligence. In contrast to conventional programs or systems, an agent does not follow a prescribed algorithm, but utilizes reasoning and learning to achieve goals related to tasks. For further information, see the chapter on Intelligent Agents.

Interaction Protocol

A set of rules that describes the expected behavior of the parties involved in a conversation. In this context, a conversation refers to the exchange of a series of messages that are grouped together, e.g. as part of a business transaction. For example, a party may inquire about the availability and cost of a particular service, receive a confirmation and quote from the provider, request the service, receive the result, confirm the receipt of the result, pay the provider, and receive a confirmation of payment from the provider. An example of an interaction protocol is the Simple Object Access Protocol (SOAP). In practice, the distinction between communication and interaction protocols is often blurred. See also semantic protocol.

Knowledge Exchange

In contrast to the exchange of data and information, where the interpretation is mostly left to humans, the exchange of knowledge between computers refers to the sharing of computational structures that represent knowledge in such a way that this knowledge can be interpreted and utilized in a meaningful way by the receiving system. This does not necessarily imply that computers "understand" the contents of these structures, but they must be able to perform activities such as business transactions that are dependent on the content of the documents. The exchange of knowledge usually is supported by a shared ontology that specifies the terminology, and by a knowledge exchange language or knowledge exchange protocol such as Resource Description Framework (RDF), the Knowledge Interchange Format (KIF), or the Knowledge Query and Manipulation Language (KQML). See also Ontology, Semantic Protocol.

Metadata

Additional information about the data in a document, such as the name of the field containing the actual data (e.g. "city" for the respective field in an address entry). Metadata are often used for the interpretation of the content of a document, and may also provide details about the context in which the document exists and is used. Annotations in XML or a similar language capture metadata in a specific format that allows the appropriate processing of the actual data in the document. See also Resource Description Framework, Semantic Protocol, XML.

Ontology

An ontology defines a common framework of reference for the terms used in a particular domain. It identifies the main concepts in the domain, describes their meaning, and captures the relationships between concepts in a systematic way. Ontologies are often built around a hierarchy of concepts in the domain, and visualized as a network of nodes (for concepts) and links (for relationships). In this context, ontologies are important for the semantic interpretation of documents: They provide an explicit formal specification of terms and relationships between terms. This enables computers to perform some reasoning about the contents of a document, which in turn makes more complex interactions between networked computer systems possible.

Resource Description Framework (RDF)

A model for metadata that is used as a convention for designing XML documents. It allows the processing of the actual data in the document in a more meaningful way by providing additional information about the intended meaning and usage of a data item. RDF is used in particular for the description of Web pages in the Semantic Web. It uses object-attribute-value triples A(O, V) as basic building blocks: object O has an attribute A with the value V indicating that there is a relation expressed through the attribute A between object O and the respective value V for the attribute. An RDF Schema can be used to formally describe the semantics of properties. An example of such a schema is the Dublin Core [http://www.dublincore.org/] that defines bibliographic metadata.

RosettaNet

The definition of electronic business interfaces in the information technology industry is the goal of the RosettaNet framework. The framework defines commonly used terms through a dictionary of IT products, and specifies standardized business processes through Partner Interface Processes (PIPs). In contrast to the generic ebXML framework, RosettaNet is vertical in the sense that it is specific to the domain of information technology. See also Electronic Business Frameworks, ebXML.

SCHEMA

In this context, a schema is a document type, providing a definition of the structure and elements for documents of that particular type. In XML, document type definitions (DTDs) and schemas (or schemata) are used for the same purpose, but the latter are more general, and offer more advanced capabilities for the treatment of documents. XML Schemas are written in the XML Schema Definition Language (XSDL). In a similar way, a RDF Schema is used to describe the semantics of the properties of metadata.

SEMANTIC PROTOCOL

A set of rules that describes the interpretation of the contents of messages or documents exchanged by the participants. Semantic protocols rely on communication and interaction protocols to facilitate the actual exchange of the messages. They are usually supported by metadata, providing additional information about a document and its content, and ontologies that describe the vocabulary used in the documents. Semantic protocols are often part of knowledge representation and exchange frameworks, such as the Resource Description Framework (RDF), the Knowledge Interchange Format (KIF), or the Knowledge Query and Manipulation Language (KQML).

SEMANTIC WEB

The Semantic Web enriches the World Wide Web by augmenting Web pages with additional data and documents that allow computers to make better use of the meaning conveyed in documents. These additional data, also referred to as metadata or semantic data, provide computers with more information about the data that make up the contents of a document. For example, they could add hyperlinks to a document that point to definitions of the key terms and their relationships to other terms as specified in an ontology. While this may not enable computers to develop a true understanding of the meaning of a document, computers will be able to perform tasks that depend on the meaning of the contents, rather than the rigid syntactical and data-oriented criteria that are used by search engines or other, more conventional technologies. To a limited degree, it will allow computers to reason logically about these documents and their contents. Thus, a computer-based system, possibly in the form of intelligent agents, will be able to conduct a substantial variety of commercial transactions that otherwise require human intervention.

SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

This protocol specifies a mechanism to directly exchange computer objects between systems on a network. Without such a protocol, objects representing possibly complicated data structures can not easily be exchange because they may have different internal representation formats that are dictated mainly by the programming language used. SOAP actually is a meta-protocol based on XML that can be used to define new protocols within a clearly defined, but flexible framework. SOAP was originally proposed by an industry group under the guidance of Microsoft, but is being supported by companies and organizations that traditionally have not been supportive of Microsoft.

XML (EXTENSIBLE MARKUP LANGUAGE)

The goal of XML is to enable information interchange in networked computer systems in a precisely defined, but also very flexible manner. Its basic idea is to annotate, or mark up, text or other data contained in documents with additional information (often referred to as meta-data) that help computers with the processing and interpretation of the contents of the document. Similar to HTML, XML uses so-called tags for the markup. While in HTML the set of tags is limited and fixed, and mainly used for formatting purposes, XML provides a framework for a common, extensible data representation, and for the formal specification of rules that apply to all documents of a particular type. It allows the definition of sets of tags that can reflect the needs of a particular domain or community. These sets of tags are typically defined in Document Type Definitions (DTDs), or in XML Schemas. This makes XML much more flexible than HTML, and also enables the use of the tags to help with the interpretation of the contents of a document by computers. Historically, XML was developed as a reaction to one major deficiency in HTML that became more and more critical as the World Wide Web grew: The fixed set of tags in HTML, or its lack of extensibility. XML is a simplified subset of the Standard Generalized Markup Language (SGML), with particular emphasis on the requirements of the World Wide Web. Over the last few years, XML has been embraced by many organizations and communities, and more and more standards and products based on XML are being used for the easy exchange of documents between computers.

References

[Albert, 2000] Steven Albert. Information and Knowledge Exchange: The Cases of the World Meteorological Organization and the World Health Organization, Knowledge and Process Management, 7:2, 65-75.

[Banerjee & Kumar, 2002] Snehamay Banerjee and Ram Kumar. Managing Electronic Interchanges of Business Documents. Communications of the ACM, vol. 45, no. 7, pp. 96-102, 2002.

[Berners-Lee, Hendler & Lassila, 2001] Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. Scientific American, pp. 35-43, May 2001.

[Bray et al., 1998] T. Bray, J. Paoli, and C.M. Sperberg-McQueen (eds.). Extensible Markup Language (XML) 1.0, W3C Recommendation, February 10, 1998, http://www.w3.org/TR/REC-xml.

[Brickley & Guha, 2000] D. Brickley and R. Guha (eds.). Resource Description Framework (RDF) Schema Specification, W3C Candidate Recommendation, March 27, 2000, http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

[Bryan, 1998] Martin Bryan (ed.). Guidelines for Using XML for Electronic Data Interchange. XML/EDI Working Group, available at http://www.xmledi-group.org/xmledigroup/guide.htm, 1998.

[Chan, 1997] Siu-cheung Charles Chan. Introduction to Electronic Data Interchange (EDI). Available at http://home.hkstar.com/~alanchan/papers/edi, 1997.

[Chandrasekaran et al., 1999] B. Chandrasekaran and John R. Josephson and R. Richard Benjamins. What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems, 14:1 20-26, 1999.

[COM, 2002] Microsoft COM Technologies – Information and Resources for the Component Object Model-based Technologies, available at http://www.microsoft.com/com, 2002.

[Comer, 1995] Douglas E. Comer. Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture. Third edition, Prentice Hall, 1995.

[CORBA] The Object Management Group (OMG) CORBA Web site at http://www.omg.org/corba

[Decker et al., 2000] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann and Ian Horrocks, The Semantic Web: The Roles of XML and RDF, IEEE Internet Computing, 4:5, 63—74, 2000.

[Genesereth & Fikes, 1992] M. R. Genesereth and R. E. Fikes (eds.). Knowledge Interchange Format Reference Manual, version 3.0. Stanford University, Computer Science Department, Technical Report Logic-92-1, 1992.

[Goldfarb & Prescod, 2002] Charles F. Goldfarb and Paul Prescod. XML Handbook. Fourth edition, Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[Gomez Perez & Benjamins, 1999] A. Gomez Perez and V. R. Benjamins. Applications of Ontologies and Problem-Solving Methods. AI Magazine 20:1,119—122, 1999.

[Gruber, 1992] T. R. Gruber. Ontolingua: A mechanism to support portable Ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

[Gruber, 1993] T. R. Gruber. A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220, 1993.

[HTTP, 1999] The Internet Society. Hypertext Transfer Protocol – HTTP/1.1. Available at http://www.w3.org/Protocols/rfc2617/rfc2616.html, 1999.

[ISO-OSI] International Organization for Standardization. Open Systems Interconnection, ICS Field 35.100, available at http://www.iso.org/.

[KQML] University of Maryland, Baltimore County. Knowledge Query and Manipulation Language (KQML) Web at http://www.cs.umbc.edu/kqml/.

[Labrou & Finin, 1996] A Proposal for a new KQML Specification. Yannis Labrou and Tim Finin, University of Maryland Baltimore County, TR CS-97-03, February 1997.

[Lassila & Swick, 1999] O. Lassila and Ralph Swick (eds.). Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, February 22, 1999, http://www.w3.org/TR/REC-rdf-syntax.

[Noy & McGuiness, 2001] N. F. Noy and D.L. McGuiness. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University, http://www.ksl.Stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.htm, 2001.

[RDF, 2002] Eric Miller, Ralph Swick, and Dan Brickley. Resource Description Framework (RDF), available at http://www.w3.org/RDF, 2002.

[Russell & Norvig, 1995] S. Russell and P. Norvig. Artificial Intelligence – A Modern Approach. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[Siu & Jain, 1995] K. Siu and R. Jain. A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management. Computer Communications Review (ACM SIGCOMM), vol. 25, no. 2, pp. 6-28, 1995.

[Staab et al., 2000] Steffen Staab, Michael Erdmann, Alexander Maedche, and Stefan Decker. An extensible approach for Modeling Ontologies in RDF(S). Technical Report 401, AIFB, University of Karlsruhe, March 2000.

[Swartout et al., 1996] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. Towards Distributed Use of Large-Scale Ontologies. Technical Report, Information Sciences Institute, University of Southern California, 1996.

[UN/EDIFACT] United Nations Directories for Electronic Data Interchange for Administration, Commerce, and Transport (UN/EDIFACT), available at http://www.unece.org/trade/untdid/welcome.html.

[Uschold & Gruninger, 1996] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, 3:2, 93—136, 1996.

Index