POWER TO CHANGE – HOW *XML* RESHAPED MODERN COMPUTING

DRAGOŞ MANGIUC
ACADEMY OF ECONOMIC STUDIES, BUCHAREST
6 PIAŢA ROMANĂ STREET, BUCHAREST 010374
mangiuc@gmail.com

Abstract:
Among the many Internet-based technologies which emerged in the last ten years, the Extended Markup Language (widely known as XML) was the only one able to change the face of distributed computing, e-commerce and web services forever and in a radical manner. Using XML nowadays has important consequences on all kinds of business processes, as financial, accounting and business XML applications become more and more popular. Web 2.0, the latest "wave" of Internet based services (expected to replace most desktop applications in the near future) is XML-centered and so the evolution of XML reached a new level, far from being over. This literature review research follows the evolution of XML and its objective is to assess this technology's impact on modern computing and business world during the nine years since its first version was adopted as a standard by the World Wide Web Consortium. The author has almost ten years of interest in this field, and also some previous results (papers and Ph.D. dissertation).

Key words: *XML, Web applications, Web services, computing, business process, Web 2.0*

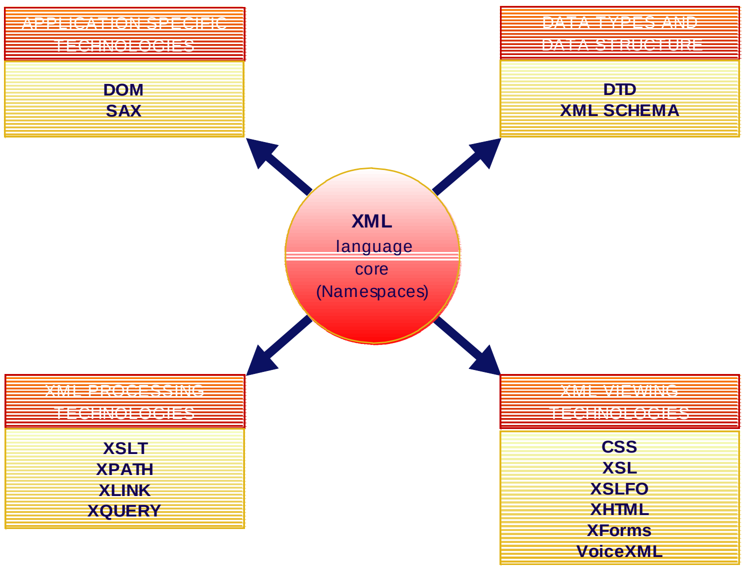
JEL classification: *M15*

INTRODUCTION

Extensive Markup Language (*XML*) is defined as a metalanguage (a language used to describe other languages) and was adopted as a standard eleven years ago by the *World Wide Web Consortium* (*W3C*), the main Web open standards promotion organisation [Raento, Mika 2003]. *XML* history is much like the *World Wide Web* history. The Web had a strong influence over many aspects of social life (work, leisure or human interaction), and *XML's* influence (as a data description language) on the form and content of distributed computing was very much alike. The main result of this influence resides in the new opportunities organisations have to get competitive advantages by adopting *XML* and Internet specific open standards as the foundation of their information systems. *XML* is basically a set of rules and principles for text data description meant to replace the legacy binary formats which are usually copyrighted and recognized as intellectual property of various developer companies. Reality proves that *XML* is now very much more than its intrinsic meaning. Since its adoption as a standard by the *W3C* in 1998 *XML* was the "engine" for the adoption of many other standards and specifications which changed the software development world.

As a data description language, XML is the core of a set of Web technologies which can be easily combined to respond to a large variety of requests (*Figure 1*). An important advantage of using *XML* nowadays is the great support it receives from the software development industry. The present software developer confronts a very large offer for support tools (such as *XML*-enabled Web browsers, *XML*-aware *DBMS* systems, *XML*-supporting operating systems) whose performances make *XML* data import and export an affordable and effective solution for small and mid-sized organizations. There is also a large choice of Web technologies which allow *XML* data parsing and transformation for Web browser presentation purposes, as well as generation of custom reports. Besides, all the widely used modern database management systems (relational and object-oriented) are endowed with native *XML* data import and export capabilities. *XML* omnipresence in the world of the Web led to important standardization efforts for the usage of *XML* along with standard Internet protocols such as *Hypertext Transfer Protocol, File Transfer Protocol or Simple Mail Transfer Protocol* [Luth, Jim 2002].

Figure 1 - *XML Core* and Related Web Technologies



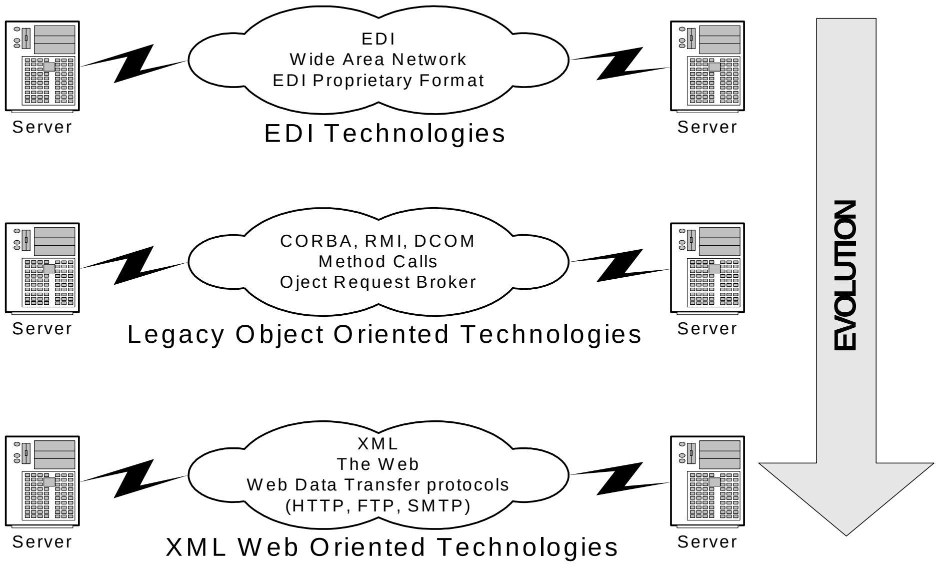
In order to determine *XML* and adjacent technologies' influence on modern software applications design and building, it is mandatory to observe this markup language by means of its main influence areas in the field of Web technologies.

1. *XML*'S INFLUENCE ON DATA STORAGE AND DATA PRESENTATION TECHNIQUES

Prior to *XML* most of the data formats were protected by copyrights and recognized as the intellectual property of their owners. Moreover, most data formats were strictly confined to a very limited number of applications able to understand process and convert those formats. Today domain-specific *XML* applications are presented as alternatives to legacy data interchange solutions (such as *Electronic Document Interchange* or *EDI*) facilitating data interchange in the business-to-business (*B2B*) sector of *e-commerce* and acting as a communication infrastructure for data distributed processing.

The main advantage of *XML* is data-independence. *XML* only describes data structures and is not linked to any operating system, programming language or data transport protocol. As a result, developers don't have to stick to any programming language, operating system or Web browser in order to use the data inside a large set of interconnected platforms. Data can flow freely from one system to another, free from all the constraints that tightly coupled platforms and transport methods used to set. *Electronic Data Interchange* was based on a proprietary binary data format and could be used only inside a very expensive *Wide Area Network*. Legacy object-oriented technologies used to transfer data based on a set of method calls which held transferred data as arguments. Using *XML*, a large set of transport technologies can be used for data transport (*Figure 2*). Consequently, transport protocols as *HTTP* had tremendous influence on adopting *XML* as a standard and proved themselves as alternatives to legacy technologies (as *CORBA, RMI* or *DCOM*) which are not functional under the *TCP/IP* protocol [Bucci G. and others, 2005]. *XML* avoids these issues entirely, being data-oriented and leaving technical details as a task for the supporting technologies.

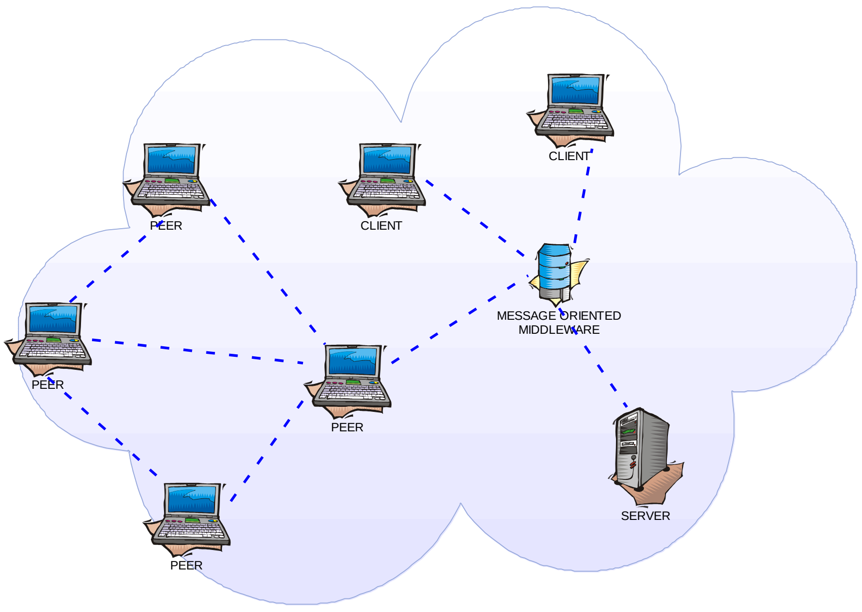
Figure 2 - The Evolution of Data Transport Technologies



2. XML'S INFLUENCE ON APPLICATIONS' ARCHITECTURE

XML's influence on applications' architecture usually consists in a shift from tightly coupled systems like *CORBA* or *DCOM*, having their own data transport protocols and steady infrastructure to loosely coupled systems based on standard Web protocols such as *TCP/IP*. Even if older transport protocols provided fast and quite error-free communication inside their own network they proved to be almost unable to communicate with third-party applications or to import or export data through *World Wide Web*. Building access bridges between tightly coupled applications is not impossible, but it is a complex and extremely expensive process which also involves disclosure of sensible and secret internal information about the two systems, such as data structures, data transfer encryption algorithms or access rights policy. This process adds a new and very complex layer to an already complex infrastructure. Loosely coupled application systems provide universal connectivity, reaching one of the most important goals of the data automatic processing [SOAP Tech Report, 2003]. Using *TCP/IP* as a data transport protocol computer systems are able to initiate ad-hoc connections based on open Web protocols. The loosely coupled Web architecture is extremely flexible allowing for the creation of new architectures based on *middleware* or *peer-to-peer* communications (*Figure 3*). *XML* plays an important role in this new architectural paradigm by means of a new protocol called *Simple Object Access Protocol* or *SOAP*. This protocol provides a set of *XML* tags used for *XML* data transport through the Web, based exclusively on Web protocols. The free data flow between various applications and platforms is an objective as old as client-server computing and its fulfillment led to an unprecedented popularity of message servers and applications able to link the participants in a message-based communication. The message servers are widely known as *Message-Oriented Middleware* (*MOM*) and have a growing importance for the "extended enterprise" concept because they guarantee message delivery and allow message multi-broadcast [Korhonen, Markku, 2000].

Figure 3 - The New Web Architecture Schema



3. *XML*'S INFLUENCE ON COMPUTER SOFTWARE SYSTEMS

The incredible speed of the changes depicted above points to a third area of *XML* influence, which is the *XML* influence on the way complex computer software systems are designed and built. Computer software systems were designed for decades as monolithic structures meant to solve punctual problems. It is widely accepted that the more problems a software system tends to solve, the less flexible, upgradable and adaptive to technology changes that system is. About 15 years ago, a new software system building model appeared, and simplicity was it's main feature. Instead of defining complex requests packages and requiring the building of complex applications in order to fulfill the requests, this new approach was based on creation of simple modules able to interconnect and also connect to other existent and future modules.

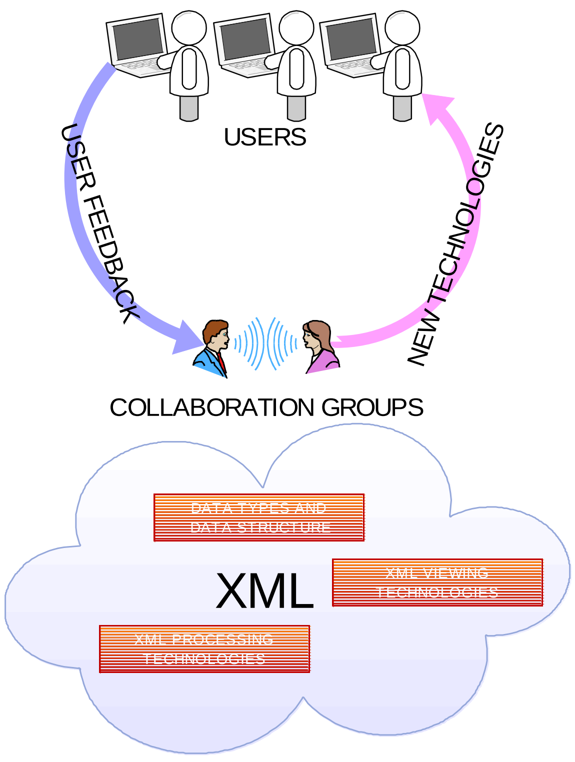
The most popular system of this new breed was without any doubt the *World Wide Web*, the most complex infrastructure ever built, able to interconnect networks and software applications with unprecedented speed and efficiency. The *WWW* is nowadays a conglomerate of support technologies (*HTTP*, *HTML*), Web browser applications (*Internet Explorer*, *Mozilla*, *Opera*) and an over 30-year old communication technology, the TCP/IP protocol (*Figure 4*).

4. WEB SERVICES

Web services are an extremely important component of "the extended enterprise". The use of Web services tends to simultaneously offer the solution for two important issues:

- Transforming the *World Wide Web* to become the perfect platform for business-to-business *e-commerce* interactions (most often these are software to software interactions).
- Keeping the object-oriented technology most important advantage: the components reuse.

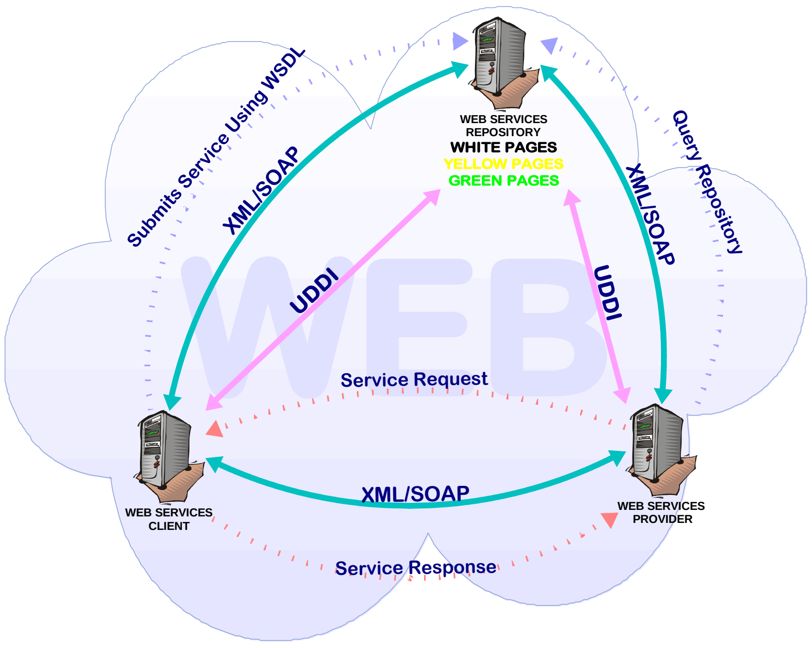
Figure 4 - *XML* Influence on Computer Systems



According to some authors Web services are the most important evolutionary step in the history of the *World Wide Web*: the Web metamorphosis from a network centered on providing services for human users to a network centered on providing services for software applications [Booth, David and others, 2003]. Web services area of applicability is almost unlimited. Web services are present everywhere on the web, from the most simple *blog* or virtual store software to complex systems such as airline booking systems or real-time weather information systems. The specific Web services architecture certifies web developers that their applications are able to collaborate with other developers' applications if necessary, in order to build a complete business process. Web services take the object-oriented goal to build software from reusable components to a new level. Only when it comes to Web services, it is not the object-oriented nature of the modules involved, but the interconnectivity that counts.

From a corporate perspective, a Web service is both a business process and a set of protocols able to search, find and interconnect with other services provided by the Web for the benefit of a corporation. By building their services on standard Internet protocols like *HTTP* or *SOAP* (*Simple Object Access Protocol*), developers can focus more on the content of data to exchange and less on the ways to transport data from source to destination. In order to facilitate even more the process of Web services development *SOAP* was "equipped" with a container (called *XML envelope* or *wrapper*) able to transport XML data and a set of conventions for the remote procedure calls. This feature allows Web services providers to publish the call syntax for their own services, so as a client application could use the services without needing detailed information about the programming language or the execution platform used to build the services. As depicted in the following figure (*Figure 5*) Web services and *SOAP* build a new type of distributed computing, based on *XML* for data definition and on *HTTP* and *SOAP* for data transfer. Specialized Web services as *Universal Description Discovery* and *Integration* (*UDDI*) or *Web Services Description Language* (*WSDL*) provide tools for the search and discovery of the needed Web services and also for the connection to the Web services found.

Figure 5 - Web Service Schema



Universal Description Discovery *and Integration* (*UDDI*) is a *XML*-based, platform independent library (or *repository*) which allows worldwide Web services and business processes to be published on the *Internet*. *UDDI* also contains information concerning the way Web services and software applications interact on-line. This repository is structured as a set of three components:

- *White pages* – provides addresses, contact information and identification elements;
- *Yellow pages* – provides a Web services classification, based on the standard industry taxonomy;
- *Green pages* – provides technical information concerning an organization listed Web services;

UDDI is one of the core Web services and supports querying by *SOAP* messages and it grants access to *WSDL* documents which describe the protocol and syntax used to interact with a certain listed Web service. In order to fulfill its goal, *UDDI* is endowed with a few redefined data types:

- **businessEntity**: information about the publishing party which may contain various types of contact information as well as one or more businessService entities.
- **businessService**: description of a set of services which may contain one or more bindingTemplates.
- **bindingTemplate**: information necessary to invoke specific services which may encompass bindings to one or more protocols, such as *HTTP* or *SMTP*.
- **tModel**: technical "finger print" for a given service which may also function as a namespace to identify other entities, including other tModels.

Web Services Description Language (*WSDL*) is an *XML*-based language used to describe Web services. It basically describes communication methods for a given Web service using *XML* language. The Web service is described as a collection of network nodes or *ports* and *WSDL* provides a standard document format for their description. Therefore, the abstract definition of ports and messages is separated from their effective use (*instantiation*), allowing definitions reuse. A port is defined by attaching a logical address to a network address and a service is defined as a collection of ports. The messages are abstract descriptions of the data to be exchanged and port types are abstract collections of supported operations. This is how *WSDL* describes a service's public interface and a client application willing to use the web service only has to read the *WSDL* document to find out what are the functions offered by the Web service server. The client application can eventually use *SOAP* to call the functions listed in the *WSDL* document.

5. XML AND DISTRIBUTED COMPUTING

Distributed computing went through important changes during the last few years. The "classic" networks built on the client-server paradigm which dominated the scene in the eighties and the nineties are now put away by a new type of network built around the World Wide Web. Instead of proprietary communication standards, this new network type uses open standard protocols and it is based on the revolutionary data description language, *XML*. The main effect of this "mutation" is the important change in the way organisations look at distributed computing: instead of focusing on the network support and data transport technologies, enterprises are now focused on improving and strengthening relations with customers and business partners, using the existing Web support. This direction was predicted since the beginning of the nineties, when a research group built a „prototype" of the future Internet, using a file request protocol called the *Hyper Text Transfer Protocol* or *HTTP* and a software application called *browser* for retrieval and presentation of *HTML* files. The result was what we call today the *World Wide Web* [Berners-Lee, Tim; Mark Fischetti 1999]. But the real turning point in taking the Web serious as a reliable support for distributed computing was the rise of *XML* and its adoption as a standard in the late nineties. Consequently, most of the major players in the field rushed to incorporate the change as a component of their future plans. Even Microsoft initiated a large re-engineering process shifting from an operating systems and desktop applications production company to a *XML*-based Web services provider. This is fully reflected by their .Net software development platform [Maguire, Steve, 1993]. The .Net platform is Web service-centered and managed to radically change the way big software producers build corporate software applications. In order to stand ground in front of the new „reengineered" *Microsoft*, *Sun* and its flagship product, *Java*, the most widely used enterprise software programming language [Perrone, Paul J., Chaganti, Venkata, 2000] started offering complete *XML* and Web services support, as an essential request for keeping *Java's* leading position as an enterprise software platform. Both the standard *Java* platform (*J2SE*) and its industrial strength „big brother" (*J2EE*) were endowed with *XML* support.

CONCLUSIONS

The combined effect of *XML* language's simplicity and Web technologies' potential opened new ways to look at data transfer and data communication as well as the emergent software applications architecture model based on the Internet and its support protocols. The impact of *XML* on modern computing is complex and could be synthesized as follows:

- *XML* greatly reduced the general dependence of software application on proprietary applications and data formats.
- *XML* introduced a new communication model for e-commerce applications in the business-to-business area, reducing these systems' dependence on legacy *Electronic Document Interchange* (*EDI*) systems.

- *XML* managed to shift from tightly coupled systems based on *CORBA*, *RMI* or *DCOM* object-oriented technologies to loosely coupled systems based on the Internet and *SOAP*.
- *XML* managed to shift from the object-oriented architecture to the freshly emerged service-oriented architecture.
- *XML* allowed the quick and efficient development of Web services as a search, find and connect technology for the Internet based applications and services.
- *XML* was the engine of the evolution from monolithic applications meant to respond to a large set of requests to a simpler approach and a new software model which reaches its objectives by interconnecting well-defined limited-scope components.

ACKNOWLEDGEMENTS

This paper is also a part of the Id_1866/2008 (766/2009) research project, called "*Research in the Field of Modeling And Building Organizational Memory. OMCAAF – A New Methodological Framework for Financial and Accounting Cognitive Acquis Capitalization*", funded on the basis of a national competition conducted by the National University Research Council (CNCSIS) within the Romanian Ministry of Education.

REFERENCES

1. Berners-Lee, Tim; Mark Fischetti (1999). Weaving the Web: Origins and Future of the World Wide Web. Britain: Orion Business
2. Booth, David and others (2003), "Web Services Architecture", W3C Working Draft 8 August 2003, available on-line at http://www.w3.org/TR/2003/WD-ws-arch-20030808/
3. Bucci, G.; Ciancetta, F.; Fiorucci, E.; Gallo, D.; Landi, C (2005), "A low cost embedded Web service for measurements on power system", Proceedings of the 2005 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2005. VECIMS 2005, 18-20 July 2005
4. Korhonen, Markku (2000), "Message Oriented Middleware", available on-line at http://www.tml.tkk.fi/Opinnot/Tik-110.551/1997/mqs.htm
5. Luth, Jim (2002), "OPC brings XML-DA to the factory floor", available on-line at http://www.iconics.com/news/feature_display.asp?Article=ConSol-0702.htm
6. Maguire, Steve (1993), Writing Solid Code: Microsoft's Techniques for Developing Bug-Free C Programs, Microsoft Press
7. Perrone, Paul J., Chaganti, Venkata (2000), Building Java Enterprise Systems with J2EE, SAMS Publishing
8. Raento, Mika (2003), "The XML Metalanguage", available on-line at http://mikie.iki.fi/teaching/xml_s03/handouts-lect1.pdf
9. SOAP Technology Report (2003), available on-line at http://xml.coverpages.org/soap.html