Applying MDA in Enterprise Application Interoperability: The PRAXIS Project[¶]

Vassilios Karakoidas[1], Stephanos Androutsellis-Theotokis[1], Diomidis Spinellis[1], Yannis Charalabidis[2]

[1] Athens University of Economics and Business, 76 Patission Str., Athens, 10434, Greece, {stheotok, bkarak, dds}@aueb.gr

[2] Singular Software SA, 23rd km Athens-Lamia Road, Ag. Stefanos, 14565, Greece, yannisx@singular.gr

Abstract: This paper elaborates on the application of the MDA approach for achieve interoperability among existing Enterprise Applications, such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems, through utilizing XML/B2B interconnection standards, developing components and modifying existing systems where necessary. We present the PRAXIS project design as a specific case study of this approach, describing the way in which interoperability-oriented high level model mappings are providing the means for guiding the model-driven development of interoperable Enterprise Applications. Specifically, we focus on the transformation processes that drive the Computation Independent System model into the Platform independent and finally the Platform Specific Model, including our final design decisions for the PRAXIS system.

The problem of Interoperability

Since the 1960s, Enterprise Applications development suffered from a lack of available technological solutions for organizing business processes [19]. To deal with this issue, software engineers worked in the direction of developing software in the areas of Relational Database Management Systems (RDBMS) and Rapid Applications Development IDE (RAD). Researchers also developed methodologies for formalizing the design of Information Systems such as the Soft Systems Methodology (SSM) [21] [22] [23] etc. Recently, IEEE provided a series of guidelines for efficient system design and development [20].

During the last two decades, many sophisticated systems were also developed to provide common ground for enterprise application development. These Enterprise Resource Planning systems (ERP) are widely used nowadays in the design of modern Information Systems (IS).

In the last few years, however, the focus has shifted from the development of software systems to the integration between them, in other words dealing with the problem of interoperability.

Interoperability is defined as [1]:

[¶] In Harald Kühn, editor, *Workshop on Ontology and Enterprise Modelling: Intgredients for Interoperability*, pages 76–84, December 2004.

"… the ability of two or more systems or components to exchange information and to use the information that has been exchanged".

This is a particularly complicated task. Most systems developed and in use today by organizations were deliberately alienated with each other, making it very complex to find common ways to share business related information.

Driving Forces of Interoperability

While working on the PRAXIS system [2], we made some interesting observations regarding the problem of interoperability between enterprise applications.

Figure 1 shows an attempt to categorise the driving forces that affect interoperability in Enterprise Applications development.

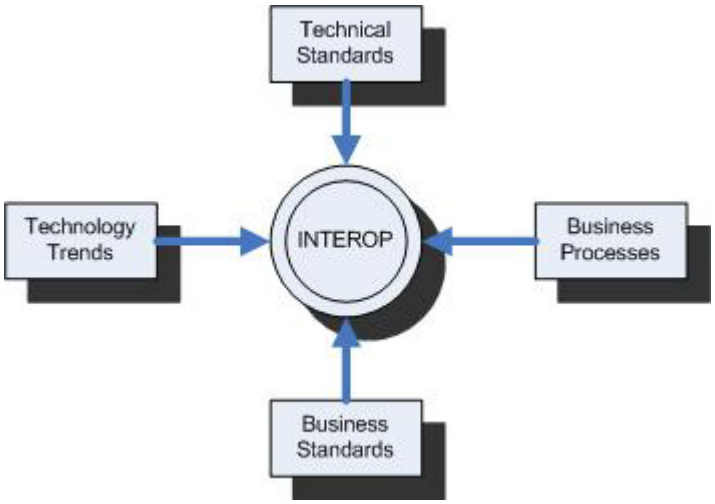

Figure 1 – Driving Forces of Interoperability in Enterprise Applications

Technical Standards: Enterprise Applications are based upon modern or legacy technical standards, such as XML [8], CORBA, and Java. In trying to make a system interoperable, one has to consider a wide set of standards in order to implement the solution.

Technology Trends: Technology trends pose serious restrictions to interoperability. In fact, it is common practice that the implementation of an interoperable system is attempted with the use of the most popular technology instead of the most effective one. For example, the way in which XML [8] and Web Services [13] affected Business 2 Business standards such as ebXML [9] and UN/CEFACT [10]. Similarly, according to recent research papers, ontologies are expected to be used as key technology for Enterprise Modeling [18].

Business Standards: These are often used as the common language between enterprise systems. Attempts to design an interoperable system

often depend on the usage of one or more business standard, for example, EDI [11], ebXML [9], RosettaNet [12] etc.

Business Processes: It is a common case for enterprises to achieve a level of interoperability in the business process level. In fact, in order to design such a bridge, one must reengineer a set of common business processes between the enterprises. In order to face interoperability issues, an enterprise must perform some “light” Business Process Rengineering (BPR). EbXML [9] is a widely accepted standard that is trying to solve this problem.

Considering the aforementioned facts, we attempted to find a formal and concrete way to design the PRAXIS system.

Case Study: The PRAXIS Project

The main objective of the PRAXIS system [2], which is particularly targeted towards SMEs, is to allow the interconnection of existing Enterprise Applications, such as Enterprise Resource Planning and Customer Relationship Management (CRM) systems, as well as their seamless interconnection with corresponding software applications of the public sector and financial institutions. This is achieved through the utilization of XML/B2B interconnection standards to support interoperability, the development components and the modification of existing systems where necessary.

The goal is to support a variety of transactions, ranging from invoicing and sales cycle support to bank and tax payments.

It is expected that the SMEs which will adopt the PRAXIS system will experience a significant reduction in the required time and the rate of errors involved in carrying out the various transactions, which in turn will offer an overall competitive advantage.

Using a Model-Driven Architecture approach

The Model Driven Architecture (MDA) concept aims to facilitate the design and development of wide-scale enterprise applications in an evolvable and flexible way, with particular emphasis on interoperability [3].

MDA defines an approach that separates the system functionality specification from its implementation on a specific technology platform. In this way, the system architecture can be language, vendor and middleware neutral.

Through the MDA approach, systems are modeled at the following three different levels of abstraction and platform/technology dependency:

- The Computation Independent (or business domain) Model (CIM) is one in which the computational details are hidden or as yet undetermined.

- The Platform Independent Model (PIM), expressed in UML [15], describes computational components and their interactions in a platform-independent manner. The PIM represents the logical view in which the composition and behaviour of all components (but not their implementation) are fully specified.

- The Platform Specific Model (PSM) is expressed in terms of the software development platforms, software standards and network protocols of the specification model of the target platform.

Since the PIM, by definition, does not contain technical details, it is envisaged that the PIM will be mapped to one or more platform-specific models (PSM). The PSM is a refinement of the PIM to target platforms such as Microsoft .NET (COM+), Enterprise JavaBeans (EJB) or the CORBA Component Model (CCM). The PSM represents the source code or its UML representation. There can be as many PSM as there are different implementations of a given PIM.

System design and modeling

Our system was designed using three levels of abstraction as defined in the MDA methodology.

Computation Independent Model Level

At the most abstract, Computation Independent System (CIS) Model level, the proposed system would be described as shown in Figure 2.

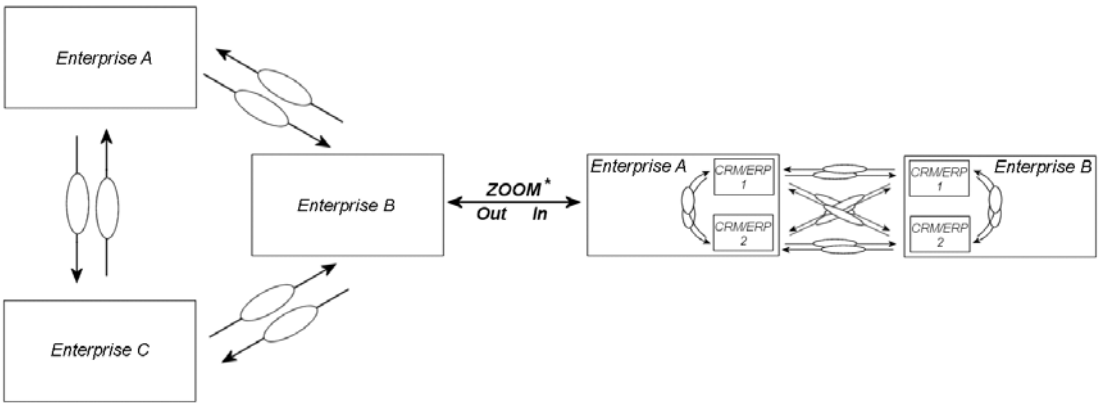

Figure 2 - ***The Computation Independent System Model for the PRAXIS system***

This abstract model illustrates the interoperability requirements for different software applications owned by the same or different enterprises that are expected to interact with each other. The various transactions schematically illustrated in Figure 2, entail a variety of interactions between the applications and lay out a set of complex functional and non-functional requirements for

the system to be designed and implemented, which must be examined at the lower modeling levels.

It should be noted that the CIM model is abstract enough to encompass any system architecture, including both client-server and distributed approaches. The use of MDA describes and represents interactions between entities, and not the way in which they will be implemented. The choice of architecture and components that will be utilized is up to the next model level.

Platform Independent Model Level

In refining the CIM model to produce the Platform Independent Model (PIM), the top-level system architecture needs to be designed. In the PRAXIS system design phase it was decided [2] to adopt a client-server architecture, consisting of a central server performing communication, mediation and store-and-forward activities, and a set of client applications that will run on the connected enterprise sites.

The server, which does not appear in the CIS model, needs to be part of the PIM model, in order to produce a valid system description that will be further refined at the platform-specific level.

Figure 3 illustrates the PIM level model for (a subset of) the PRAXIS system (simplified for clarity). The model is divided in two main modules, the server and client.

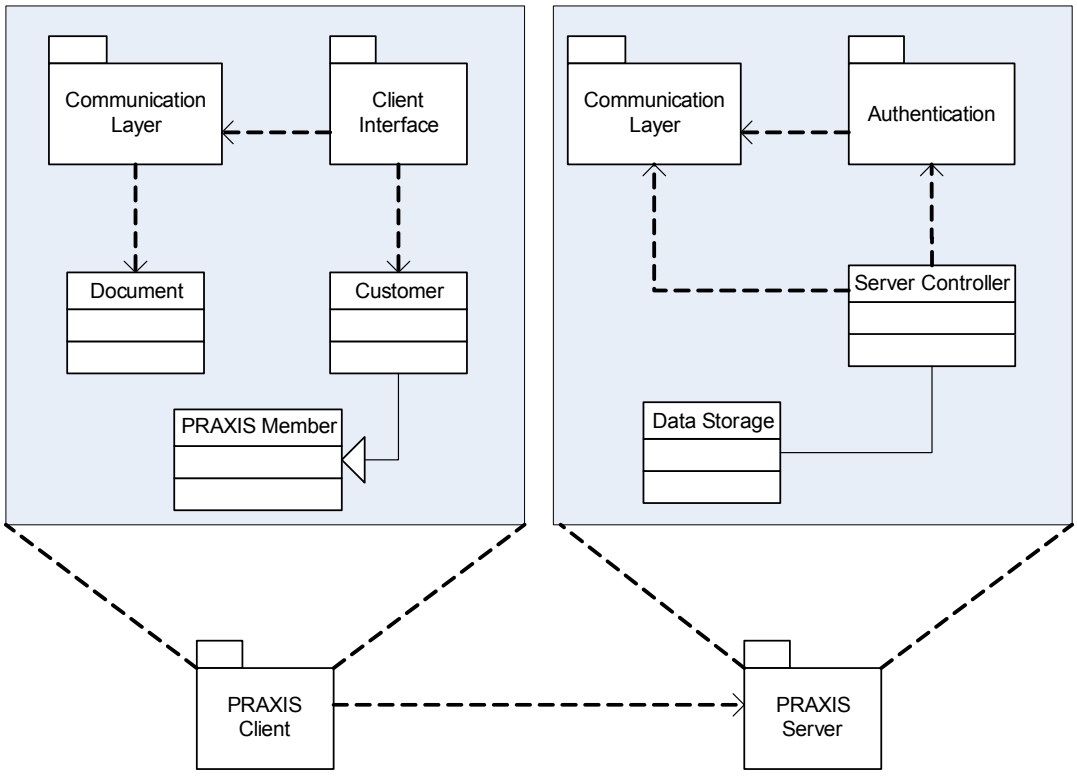

Figure 3 - ***Platform Independent Model for the PRAXIS system***

The PIM design is based on UML notation [15] and provides the bridge between the abstraction and the realization.

Platform Specific Model Level

The transition from the PIM to the PSM model consists of defining specific technologies and infrastructures for the various components of the system. These mappings fill the gap between the abstract model and the Platform Specific Model. The realization of our system is depicted in Figure 4.

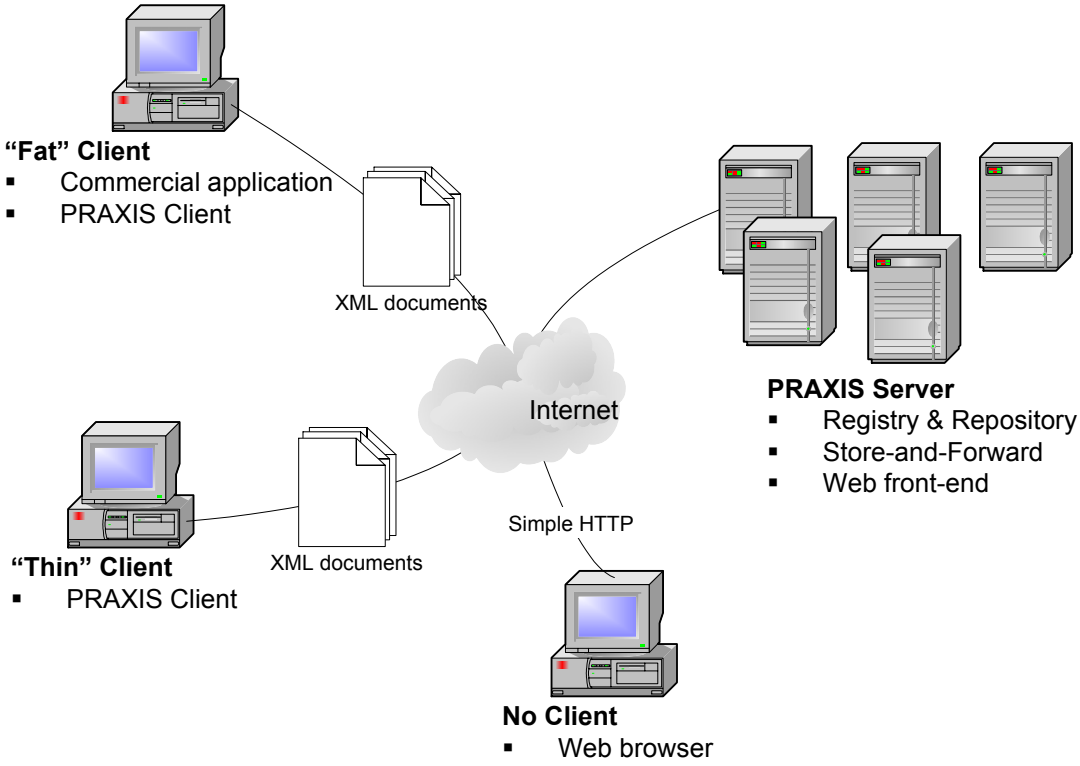

Figure 4 – PRAXIS System Architecture

In the case of the PRAXIS project, the central choices were the following:

- The server was implemented on the .NET platform, using a mixture of C++ and C# code.
- Data is stored in an MS SQL Server Relational Database.
- The following three types of clients were implemented:
  - A *Fat (Thick) Client*, based on an existing ERP system.
  - A *Thin Client, a* typical win32 application that implements the basic functionality of the system. This will be implemented in Visual Basic .NET.
  - A *Web Client*, a web application that provides limited access for enterprises, which do not have an ERP system installed.

These three types of clients were provided in order to reach out for SMEs that do not have the capability to support expensive ERP systems.

A more detailed architecture schema is provided in
Figure 5.

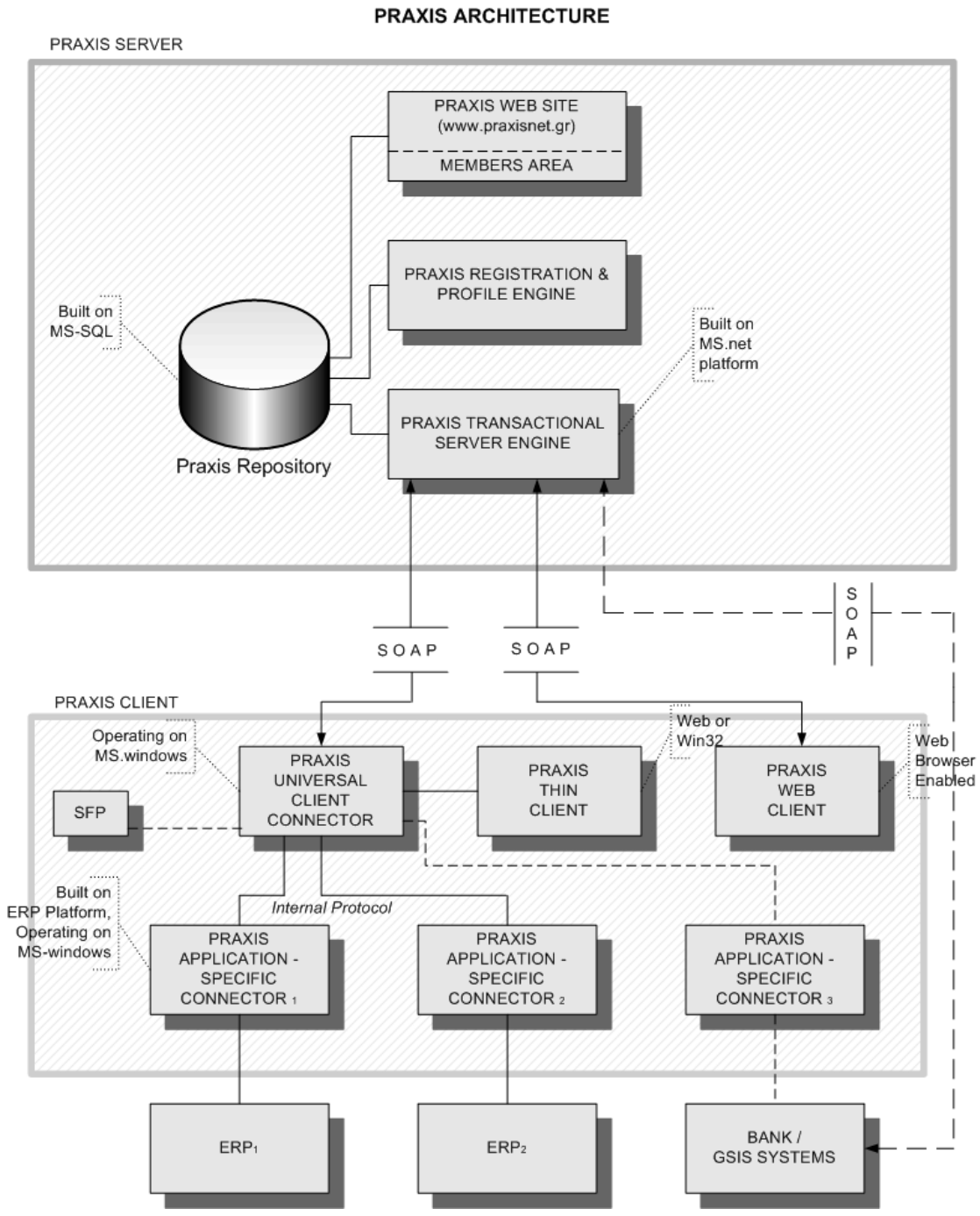

Figure 5 – PRAXIS system architecture in detail

The communication is layered as illustrated in Figure 5. The base protocol is based on SOAP. Other protocols that might be needed to achieve interoperability between distinct systems are supported through layering them on top of SOAP, making the necessary transformations where appropriate.

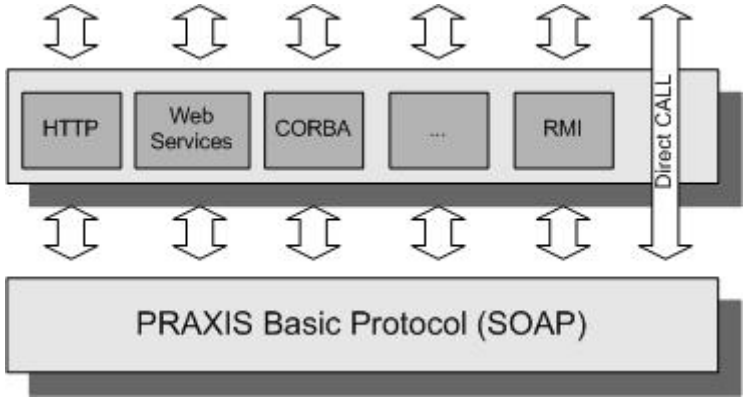

Figure 6 – PRAXIS Protocol Layers

Conclusions and Future Work

This paper has presented the application of the MDA approach in the design of a client-server application to support interoperability among enterprises. It has illustrated the refinement steps that lead from the more abstract to the technology-specific models, pointing out the specific patterns of the design. We are convinced that these patterns can be extracted and later used as common practice in similar situations. A direct extension of our work may include the employment of Ontology-based approaches to be defined in order to provide common knowledge [14] [16] [17] [18].

References

[1] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.

[2] Yannis Charalambidis, Vassilios Karakoidas, Stephanos Androutsellis-Theotokis and Diomidis Spinellis, *Enabling B2B Transactions over the Internet through Application Interconnection : The PRAXIS project*, e-Challenges 2004, Vienna, 27-29 October.

[3] Joaquin Miller and Jinshu Mukerji, *Model Driven Architecture*, July 2001, Available online at http://www.omg.org/mda/

[4] John Daniels, *Modeling with a sense of purpose*, IEEE Software, January / February 2002,

[5] Alan Brown, *An introduction to Model Driven Architecture Part I : MDA and today's systems*, January 2004, Available online at http://www-106.ibm.com/developerworks/rational/library/3000.html

[6] B. Selic, *The pragmatics of Model-Driven Development*, IEEE Software, Vol. 20, September 2003

[7] D. Frankel, *Model Driven Architecture : Applying MDA to Enterprise Computing,* Wiley Press, 2003

[8] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and Francois Yergeau. *Extensible markup language (XML) 1.0. Technical report*, W3C, February 2004. Available online at http://www.w3.org/TR/2004/REC-xml- 20040204

[9] ebXML Requirements Team. *ebXML requirements specification*

v1.06. Technical report, ebXML, May 2001. Available online at http://www.ebxml.org/specs/ebREQ.pdf

[10] UN / CEFACT. United nations centre for trade facilitation and electronic business. Available online at http://www.unece.org/cefact/

[11] UN / EDIFACT: UN directories for electronic data interchange for administration, commere and transport (UNECE). Available online at http://www.unece.org/trade/untdid/texts/d100 d.html

[12] RosettaNet. *Rosetta Implementation Framework (RNIF): Core specification* Technical report, RosettaNet, March 2002. Available online at http://www.rosettanet.org/

[13] David Booth, Hugo Haas, Francis McCabe Eric Newcomer, Iona Michael Champion, Chris Ferris David Orchard, *Web Services Architecture,* Technical Report, W3C, February 2004. Available online at http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[14] F. Vernadat, *Enterprise Modeling: objectives, constructs and ontologies*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[15] Berio, G., Petit , M., *Enterprise Modelling and UML: (sometimes) a conflict without a case*. In Proc. Of Concurrent Engineering Conference 03, July 26-30, Madeira Island, Portugal, 2003

[16] Wand, Y,*Ontology as a Foundation for Meta-Modelling and Method Engineering*. In Information and Software Technology, Vol. 38 (1996) 281-287

[17] G. Guizzardi, G. Wagner, *A Unified Foundational Ontology and some Applications of it in Business Modeling*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[18] M. Lenzerini, M. Missikoff: *Ontologies for Interoperability*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[19] Linthicum S.D., *B2B Application Integration – e-Business-Enable Your Enterprise*, Addison-Wesley, 2001

[20] IEEE Guide for Developing System Requirements Specifications Std 1233, *IEEE Standards Software Engineering: Volume One Customer and Terminology standards* 1999 Edition,Available online at http://www.standards.ieee.org/

[21] Checkland, P, *Towards a system-based methodology for real-world problem-solving*, Journal of Systems Engineering, Vol. 3, No. 2 - reprinted in *Systems Behaviour*, Third Edition (1981), The Open Systems Group, Open University Publishing, UK

[22] Checkland, P. & Scholes, J.(1990) *Soft Systems Methodology In Action,* John Wiley & Sons, Chichester UK

[23] Checkland, P. *Information, Systems and Information Systems: Making Sense of the Field,* John Wiley & Sons, Chichester UK.