Developing Agricultural B2B Processes Using Web Services

S. KARETSOS[1], C. COSTOPOULOU[1], O. PYROVOLAKIS[2] and L. GEORGIOU[3]

[1]Informatics Laboratory, Agricultural University of Athens
75 Iera Odos str., 118 55 Athens, GREECE
http://e-services.aua.gr
[2] Hellenic Naval Academy
Terma Chatzikyriakou, 185 37 Pireaus, GREECE

[3]School of Informatics, University of Wales, Bangor
Dean Street Bangor Gwynedd, LL57 1UT, Bangor, UNITED KINGDOM

Abstract: - The objective of this work is to develop business-to-business processes of electronic markets using Web services in order to facilitate the execution of these processes in different electronic markets. The main contribution of this approach is the promotion of interoperability, just-in-time integration, and reduction of complexity. In specific, the Cooperation, Orchestration and Semantic Mapping of Web Services (termed as COSMOS) tool, which is an integrated development environment that enables the creation, design and modification of executable business processes based on the Business Process Execution Language, is used for the integration of business-to-business processes of a Virtual Agricultural Market into a fully functional Web-based environment.

Keywords: - Web services, BPEL, Business-to-Business, Electronic Markets

1 Introduction

The evolution of Information and Communication Technologies (ICT) brought new opportunities to enterprises and organizations, and changed the way of doing effectively and efficiently business. As a result, numerous electronic markets (e-markets) are continuously being deployed. An e-market can be considered as an information system intended to provide market participants with online services that facilitate information exchange and support activities related to business processes. It can support the phases of information search, negotiation, settlement, as well as, after-sales support [1].

A plethora of e-markets are operating in the agri-food sector (termed in the rest of this paper as agricultural e-markets). Agricultural e-markets can serve as an additional trade and marketing channel for agricultural firms (producers, processors, retailers, agribusinesses, wholesalers, brokers etc.), also providing them the opportunity to extend the chain of their suppliers. It is important to note that agricultural e-markets demonstrate different degrees of e-commerce adoption. For instance, there are e-markets that provide only product catalogue information (e.g. Tomatoland.com), e-markets that also provide transaction settlement (e.g. Burpee.com), and more sophisticated e-markets that support online negotiations (e.g. Agrelma.com or XSAg.com).

One of the major challenges in the electronic business (e-business) community is how to efficiently and reliably develop and maintain e-market solutions through the integration of existing application and systems [2]. Enterprises spent huge amounts of economic resources trying to integrate various non-compatible software systems and applications in order to automate their business processes and to collaborate with their business partners [3]. Internet-based software components available to their users (known as Web services) have been gaining popularity for developing business integration solutions. Web services are considered to be the key to revolutionizing how business will use the Internet to operate and interact with one another in the future.

In [4] it is stated that the term Web services means different things to different people. In this paper, we use the definition of Web services as stated by the World Wide Web Consortium (W3C) Web Services Architecture Group: “a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable

format (specifically Web Services Description Language - WSDL). Other systems interact with the Web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages, typically conveyed using Hypertext Transfer Protocol (HTTP) with an Extensible Markup Language (XML) serialization in conjunction with other Web-related standards" [5]. Web services are described, published, localized and invoked over a network and provide standardized means for service-based, language and platform independent interoperability between different and distributed software systems.

WSDL, in essence, allows for the specification of the syntax of the input and output messages of a basic service, as well as other details needed for the invocation of the service. WSDL does not, however, support the specification of workflows composed of services. In this area, the Business Process Execution Language for Web Services (BPEL4WS or BPEL) has the most prominent status [6]. BPEL is an XML-based language for enabling task-sharing across multiple enterprises using a combination of Web services. BPEL is based on SOAP, and WSDL and provides enterprises with an industry standard for business process orchestration and execution.

With Web services expected soon to be available as digital goods in e-markets, mechanisms necessary to facilitate their proper implementation will play a critical role. Within this context in this paper, firstly we present the Cooperation, Orchestration and Semantic Mapping of Web Services (COSMOS) tool [7]. COSMOS has been developed from one of the paper's authors and enables the design, creation, and modification of executable business processes based on BPEL. Second, we propose a set of Web services (expressed in BPEL) that support triangular business processes (demand, supply and transport) of agricultural e-markets, using digital intermediation services. In specific, a case study for agricultural B2B processes of a Virtual Agricultural Market (VAM) are expressed as BPEL processes, using COSMOS. The key contribution of the proposed approach is that Web services can be used by any similar agricultural e-market.

The structure of the paper is as follows: in the next section an overview of the BPEL is given. Afterwards, we present the COSMOS environment, describing its basic components, architecture and capabilities. Next, the fourth section provides an overview of VAM and two particular business processes are developed as Web services using COSMOS, providing also the BPEL code. Finally, some conclusions are given.

2 An Overview of BPEL

The concept of Web services is to use XML defined protocols, namely the SOAP for communication, the WSDL for description and the Universal Description, Discovery and Integration (UDDI) of software services over the Internet for discovery. Figure 1 presents a generic architecture of Web services.

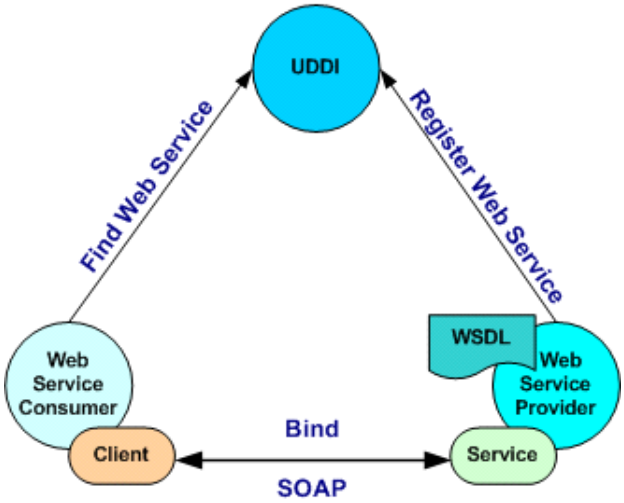

Fig.1: Web Services Architecture

Web services provide a basic one-way or request-response mechanism that can be used by two systems to communicate. Its standards are open, cross platform, and fully aligned with Internet standards and technologies. However, it is widely recognized that the interaction of several or many Web services is often required to create business value. This has led to several initiatives to create languages to express and define business processes that coordinate Web services [8].

BPEL is an XML based language that models the behaviour of Web services in a business process interaction. It is a language that models both the orchestration and choreography aspects of a business process (Fig. 2). Orchestration refers to the actual execution of a business process. It controls the flow of the various activities internal to a business process, like invocation of Web services, messages handling, business logic and rules. On the other hand, choreography describes the interfaces and the communication protocol between two or more independent business processes. It tracks the message sequence between Web services in an abstract manner [9].

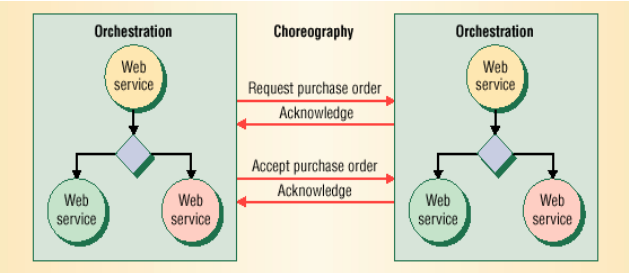

Fig.2: Web Services Orchestration and Choreography

BPEL seems to win the race for standardisation and global acceptance against other competitive initiatives. The main benefits of BPEL are:

- *Support of state-full conversations:* with the correlation mechanism of BPEL a process instance can be identified from parts of the data of the messages it handles. The correlation mechanism is responsible in deciding if an incoming message should create a new process instance or if it is a response to a previously started instance.
- *Managing of exceptions and transactions integrity:* the fault handlers of the BPEL allow the catching of runtime errors and their handling. Also the adoption of compensating transactions makes possible the notion of long-running transactions.
- *Composition of Web services:* each BPEL process is expressed as a Web service. In that way a process can invoke other processes and can also be invoked from other processes [10].
- *Rich collection of activities:* BPEL provides a rich collection of activities for the execution of many actions. It provides XML elements for Web services invocation and receive-reply, flow decision points, loops, time and message triggers of actions, data handling and messages inquiry.
- *Incorporation of standard XML protocols:* a BPEL process defines itself and its communication interface with the partners using the WSDL.

Even though BPEL is one of the most promising and industry-adopted Web services orchestration and choreography initiative today, it has also limitations and weaknesses. The most important limitations are the following:

- *Complexity:* even for modelling a simple process, the BPEL definition is extremely large and complex. Advanced workflow patterns are either very difficult or complicated or practically impossible to be implemented because of the resulting complexity of the produced process and the undocumented behaviour of some complex flow structures.
- *Not clear semantic:* the semantic of BPEL for advanced construct is not always clear. There are semantic gaps and the result is not implicit predictable [10].
- *Lack of data transformation and manipulation capability:* the lack of data handling functionality like integers and float numbers arithmetic and basic strings manipulation, adds more complexity to the business process. Instead of providing this basic functionality, BPEL forces the designers of a business process either to implement and invoke Web services, which will provide the necessary data handling functions or to use the data manipulation capabilities of the XPath standard with its difficulties and restrictions[11,12].
- *Supports only automatic fault handling:* when a fault occurs, the BPEL engine terminates the process. The language provides only the capability of the declaration of some actions to be performed before the process instance terminates. But in the real business world it does not happen that way. It should be possible to let a human actor to decide what should happen after the occurrence of an error and if the process should be terminated or not.
- *Lack of time-out and fault-handling in <invoke> activities:* there is no provision for processes waiting to invoke a temporary not responding Web service. Indeed, is not even possible to assign a fault handler for specific <invoke> constructs [10].
- *No direct support for fundamental workflow Patterns:* fundamental workflow patterns like multi-merge, discriminator, arbitrary cycles, interleaved parallel routing, milestone, multiple instances with priori runtime knowledge, and multiple instances without priori runtime knowledge are not directly supported by BPEL or are very difficult and error-prone to be implemented [10].
- *No direct support for all types of asynchronous communication:* publish, subscribe and broadcast types of asynchronous communication are not direct supported by BPEL.
- *Violation of XML syntax and conventional rules:* BPEL allows theoretical use of the character '<' in expressions as relational operator, but according to the XML specification this character is strictly illegal.
- *Dependency on no-standard protocols:* BPEL depends on non standard addressing protocols.

Indeed BPEL adds a non-standard extension to WSDL in order to define essential structures.

3 The COSMOS Environment

COSMOS is an integrated development environment for the design and creation of business processes based on the BPEL language. The goal of COSMOS is to provide a complete environment that would allow the user to design, create, code, verify and deploy a business process based on BPEL. The concept for the COSMOS deployment stems from the evaluation findings of BPEL and existing design tools.

The existing BPEL design tools are either very developer-oriented and tied to the BPEL tags instead of process concepts, or very manager-oriented and general, without basic features of BPEL, like fault handling or a real execution notion. The evaluated tools have not business process and workflow tasks orientation and do not provide an unambiguous, simple, and easy way to a user without knowledge of the BPEL language, to design and execute a business process. They are something more than just simple BPEL editors with a graphical environment, and they are not business process design tools which will help the user to think, design and implement a business process using Web services. It is worth noticing that none of these tools refer to basic business processes concepts.

The principal idea behind COSMOS software development process is that end-user applications, need and use some fundamental services hidden to the user which are responsible for communicating with the lower services provided by a platform, environment, network or operating system. The COSMOS development process considers that a software application can be conceptually approached as a combination of the following layers (Fig. 3):

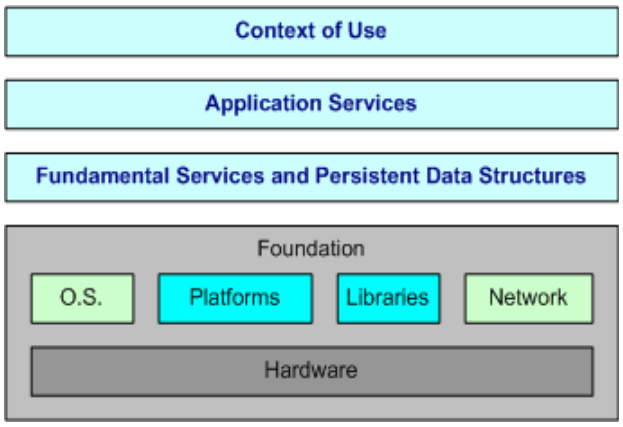

Fig.3: Software Applications Conceptual Architecture

- *Context of use:* this layer describes the interface of the application and its semantics. How, where, from whom and why, the application will be used. An application may communicate directly with other applications or interact with humans. The knowledge domain in which the application is used and the target group, are also parts of the context of use.
- *Application services:* each application provides actually some services to its users. These high level services compose the application services layer and usually are provided by collaborating software components.
- *Fundamental services and persistent data structures:* this layer consists of the general, low level and reusable software services used by the above layers in addition to the data structures used by the application services. The fundamental services are reusable classes and wrappers of persistent data structures.
- *Foundation:* this layer is the base on which the application is build. It is the underlying, operating system, framework, platform, libraries, network, and hardware. The services of this layer are the building blocks of the above layer.

COSMOS addresses the needs of two broad categories of users, namely managers and developers. In COSMOS, a business process can be described considering two different views: the manager view and the developer view. Each view is realized with different capabilities. The manager view provides a visual design environment with drag n' drop capability for the specification of the activities of a business process. It is represented by using a diagram containing information about the business process in a graphical way. The developer view follows the manager view. It provides automatically generated BPEL code for the business process as well as an XML editor for further BPEL coding. The usage of COSMOS environment is as simple as could be without unnecessary extra functionalities that could confuse users. The spirit of simplicity and formality influenced the requirements of the application.

4 Web Services for B2B Agricultural E-markets

In this section, the COSMOS environment is used for describing online agricultural B2B processes in BPEL in order (a) to promote interoperability by minimizing the requirements for shared understanding among different agricultural e-

markets, (b) to enable just-in-time integration, and (c) to reduce complexity by encapsulation. More specifically, the case of modelling agricultural B2B processes in VAM is discussed so as to facilitate the execution of these processes in different agricultural e-markets. The VAM system is an agricultural B2B e-market that supports triangular business processes namely, demand, supply and transport of agricultural products, using digital intermediation services. The market participants and their roles in the traditional agricultural supply chain are as follows [13]:

- *Producer*: is a farmer that produces agricultural products and is interested in selling them as quickly as possible (after harvest), without delay.
- *Seller:* is interested in selling agricultural products acquired from producers. Agricultural co-operatives, agribusinesses, food companies, retailers, and exporters are considered to be sellers.
- *Wholesaler:* is acting as an intermediary for the provision of matching services between demand and supply. Exporters, importers, producers, sellers, buyers, middlemen, brokers, distributors, agricultural co-operatives, auctioneers and commission merchants constitute wholesalers.
- *Buyer:* is interested to purchase agricultural products from producers, sellers or wholesalers, and then to resell them to the consumers. This participant comprises retailers, supermarkets, agribusinesses, food companies, agricultural co-operatives, and importers.
- *Consumer:* purchases agricultural products from producers or buyers. This participant can be distinguished as individuals or collective consumers (e.g. restaurants, hotels, hospitals).
- *Transportation firm:* carries agricultural products from producers, sellers or wholesalers to buyers. This participant includes local and medium-sized transport companies, and very large carriers.

For the description of the agricultural B2B process in VAM, the Unified Modelling Language (UML) is used. The UML business modelling concentrates on the business processes that will be generally supported by the VAM system. It describes the structure and dynamics of the business processes around the system. In specific, it concerns the identification of actors (anyone or anything that is external to the business but interacts with it), and use cases (a group of related workflows within the business that provide value to the actors). UML business modelling results in the use case and the activity diagrams. A use case diagram illustrates use cases and actors for business processes, as well as the interactions between them. Actors are represented as stick figures and use cases are shown as ovals. An activity diagram is used to describe the workflow through a particular use case. It consists of action states, activity states and transitions between them [14]. Figure 4 shows a UML high-level use case diagram of VAM. The VAM actors are:

- *Provider actor:* who is interested in selling agricultural products using the VAM system, and supplies the VAM system with information related to provider information, and production forecast information.
- *Customer actor:* who is interested in buying agricultural products using the VAM system, and provides it with information related to customer contact information (e.g. name, address, telephone, e-mail), customer demand information.
- *Transport firm actor:* who is responsible for delivering the goods after successful matching and negotiation process, and provides the VAM system with transport firm information.

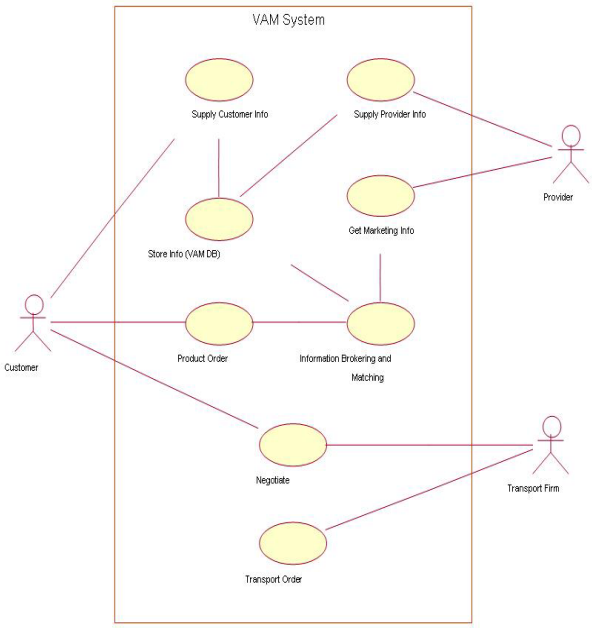

Fig.4: A Use Case Diagram of VAM

According to the VAM system's functionality, use cases are the following:

- *Supply customer info:* it is performed by the customer actor, and provides contact information and demand information to the VAM system.
- *Supply provider info:* it is performed by the provider actor, and provides the VAM system, firstly with contact information and actual field information, and next with estimated or actual production information.

- *Store info:* it is performed by the system through collection and storing both customers' and providers' given information, in the VAM Database (DB).
- *Get marketing info:* when this use case is executed, VAM informs the providers about regional, national and European market trends, and customers' preferences of products.
- *Product order:* It is performed by the customer actor, who expresses the acceptance or rejection of VAM product.
- *Information brokering and matching:* when this use case is executed, VAM aggregates and combines product information, matches the providers' production information and the customers' demand information, and then makes offers to the customers.
- *Negotiate:* when this use case is executed, a negotiation takes place between the customer actor and the transport firm actor about the terms of the payment (e.g. method of payment) and the physical delivery of the products.
- *Transport order:* after a successful matching and negotiation process, the transport firm actor is responsible to transport the order according to the agreed terms.

Figure 5 shows the activity diagram of the supply customer info use case (that corresponds to a business process). Initially, the customer provides his personal data that is need to be validated by the system. In case of successful login, the customer provides data for the demanded products. In the opposite case, the login process starts from the beginning.

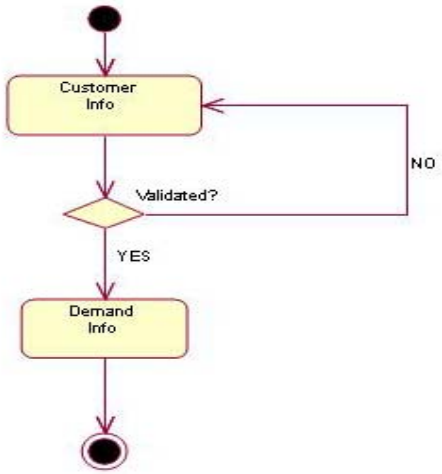

Fig. 5: Activity Diagram of the Supply Customer Info Use Case

Figure 6 shows the activity diagram of the product order business process. In the product order business process, the customer selects a provider from the resulting catalogue after the provision of the demand information, and then sends to the system the order for checking. The system checks the availability of the requested items. If the requested quantity of items item is available then, the system informs the customer who will proceed with the final confirmation of the order. Otherwise (shortage of quantity) the customer has two options, to continue by selecting another provider form the resulting catalogue or to drop the order.

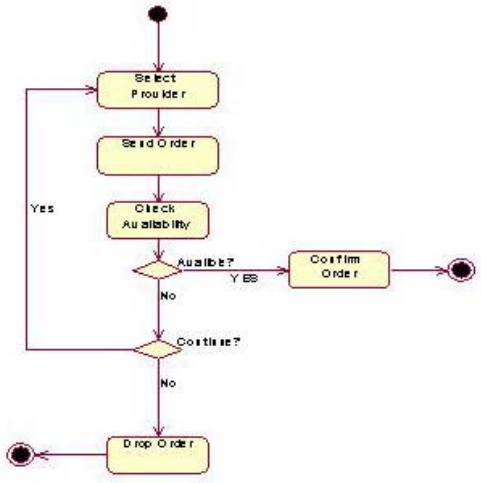

Fig. 6: Activity Diagram of the Product Order Use Case

In the following, the COSMOS environment is used for modelling the Supply Customer Info and Product Order business processes. These processes are expressed as BPEL processes. First, the manager view of COSMOS is used for designing these processes. For brevity reasons, the manger view of the product order business process is presented (Fig. 7). Second, the COSMOS developer view is used in order to express the business processes as BPEL processes. In this phase, BPEL code is automatically generated for each business process. Table 1 shows a part of the generated BPEL code of the final executable files of the supply customer info business process. Similarly, Table 2 shows a part of the BPEL code of the product order business process.

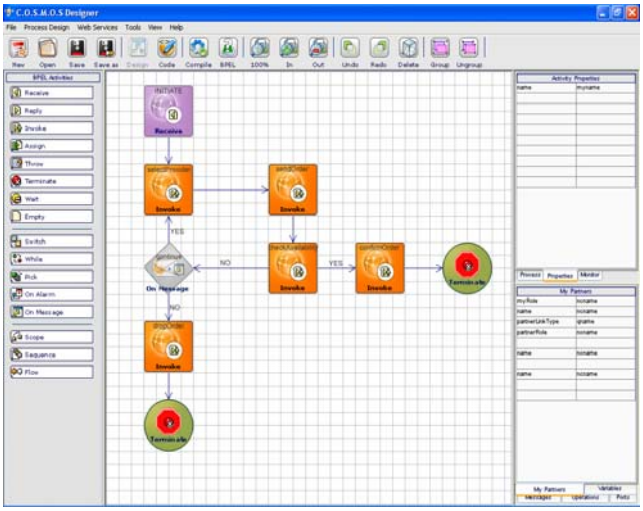

Fig. 7: Design of the Product Order Business Process in COSMOS

According to the BPEL characteristics, the supply customer info and the product order BPEL processes can be expressed as Web services. The developed services can be used by similar agricultural e-markets. Such services can lead to interoperability among different e-market systems, and enhanced user's capabilities for accessing different e-markets.

Table 1: Part of BPEL Code of the Supply Customer Info Business Process

```
<receive name="initiate" partnerLink="customer"
portType="customer:Customer"
                operation="process"
                variable="ncname" createInstance="yes" />
<invoke name="customerInformation"
partnerLink="customer" portType="customer:Customer"
                operation="process"
                variable="ncname" createInstance="yes" />
<invoke name="myname" partnerLink="ncname"
portType="qname"
                operation="process"
                inputVariable="username"
                outputVariable="validation"/>
<switch name="username">
          <case
condition="bpws:getVariableData(username,'validate'='yes'
)">
                    <sequence name="yes">
<invoke name="acceptInformation" partnerLink="broker"
portType="broker:Broker"
```

Table 2: Part of BPEL Code of the Product Order Business Process

```
<sequence name= "order">
<sequence name= "receiveInput" partnerLink="client"
                portType= "order:Order" operation="process"
                variable= "orderContentIn"
createInstance="yes" />
<sequence name= "callSelectProvider"
partnerLink="customer"
                portType= " customer:Customer"
   operation="process"
                inputVariable= "orderContentIn"
   outputVariable= "orderSelectProviderOut" />
<sequence name= "callSendOrder" partnerLink="customer"
                portType= "customer:Customer"
   operation="process"
                inputVariable= "orderContentIn"
   outputVariable= "orderSelectProductOut" />
<sequence name= "checkAvailability" partnerLink="broker"
                portType= "broker:Broker"
   operation=" checkAvailability"
                inputVariable= "orderSendOrderOut"
   outputVariable= "orderCheckAvailability" />
```

4 Conclusions

Information systems researchers develop Web services hoping that, in a near future, these services will be widely offered in e-markets [4]. In this direction, this paper presents the development of two business processes (i.e. supply customer info, product order) of an agricultural B2B e-market (termed as VAM), as Web services. Similarly, Web services have been developed for the rest of the VAM business processes such as supply provider info, get marketing info, information brokering and matching and negotiation. This is achieved using a COSMOS environment which has been proposed by one the of paper's authors. With the use of Web services in systems such as VAM a business process is externalize in a standard way, making it available to other e-markets. In the future work, Semantic Web service technologies such as the Ontology Web Language for Services (OWL-S, formerly DAML-S) will be used to develop such business processes in order to describe them in a semantic way.

References:

[1] Grieger, M., Electronic marketplaces: A literature review and a call for supply chain management research, *European Journal of Operational Research*, Vol.144, 2003, pp. 280–294.

[2] Huang, Y., Chung, J.-Y., A Web services-based framework for business integration solutions,

Electronic Commerce Research and Applications Vol.2, 2003, pp. 15-26.

[3] Nagappan, R., Skoczylas, R., Sriganesh, R.P., *Developing Java Web Services: Architecting and Developing Secure Web Services Using Java*, Wiley Publishing, 2002.

[4] Bui, T., Gachet, A., Sebastian, H.J., Web Services for Negotiation and Bargaining in Electronic Markets: Design Requirements, Proof-of-Concepts, and Potential Applications to e-Procurement, *Group Decision and Negotiation*, Vol.15, 2006, pp. 469-490.

[5] W3C, Web Services Architecture Working Group - Web Services Architecture, *http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/*

[6] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K., Bringing Semantics to Web Services: The OWL-S Approach, *Semantic Web Services and Web Process Composition*, Vol.3387, 2005, pp. 26-42.

[7] Georgiou, L., *Business Process Design based on Web Services*, Diploma Thesis, School of Informatics, University of Wales, Bangor, 2004.

[8] Putte, G., Jana, J., Keen, M., Kondepudi, S., Mascarenhas, R., Ogirala, S., Rudrof, D., Sullivan, K., Swithinbank, P., *Using Web Services for Business Integration*, IBM, 2004.

[9] Peltzer, D., *XML Language Mechanics and Applications*, Addison Wesley, USA, 2003.

[10] BEA, IBM, Microsoft, SAP AG, Siebel Systems, Business Process Execution Language for Web Services, *http://www-128.ibm.com/developerworks/library/specification/ws-bpel/*.

[11] Leymann, F.Roller, D., Thatte, S., Goals of the Business Process Language Specification. *http://xml.coverpages.org/BPEL4WS-DesignGoals.pdf*.

[12] W3C, XML Path Language, *http://www.w3.org/TR/xpath*

[13] Costopoulou, C., Lambrou, M., Karetsos, S., A Multi-Agent System for Internet Middlemen in B2B Environment: A Case Study in Agribusiness, *Journal of Applied Systems Studies* (to appear).

[14] Saleh, K., Documenting electronic commerce systems and software using the Unified Modelling Language, *Information and Software Technology*, 2002, vol. 44, pp. 303-311.