Business Process Design based on Web Services: The C.O.S.M.O.S. Environment

LOUKAS GEORGIOU
School of Informatics
University of Wales-Bangor
Dean Street Bangor Gwynedd, LL571UT
UNITED KINGDOM

ODYSSEAS I. PYROVOLAKIS
Hellenic Naval Academy
Terma Chatzikyriakou
185 39 Pireaus
GREECE

Abstract: Today's Enterprises and Organizations are facing the problem of Software Integration and the difficulty to efficiently design and implement their Business Processes using cutting-edge Information Technology. The problem investigated in this paper is the creation and design of executable Business Processes using technologies and standards such as XML, Web Services, and Java. Our proposal, the C.O.S.M.O.S. (Cooperation, Orchestration and Semantic Mapping of Services) environment, is an environment that enables the user to create, design and modify executable Business Processes following the BPEL specification, an XML based language for the Orchestration and Choreography of Web Services.
C.O.S.M.O.S. environment, attempts to be a complete and easy to use application for the design of Business Processes. It provides two perspectives for the users: a Manager-oriented interface, that includes a graphical enabled design environment of Business Processes and a Developer-oriented that provides an advanced XML Workflow Editor and functionalities such as validation and deployment of the BPEL Workflow.

Key-Words: Web Services, Software Engineering, Business Process Modeling, workflow, IDE, XML, BPEL

1 Introduction

The evolution of Information Technology brought new opportunities to enterprises and organizations, and changed the way of doing effectively and efficiently business. On the other hand the continuously increasing demand for better, faster and smarter software systems, and the plethora of offered solutions by different vendors brought a software systems' integration nightmare. The enterprises spent huge amounts of money trying to integrate various non-compatible systems and applications in order to automate their business processes and to collaborate with their partners. That situation known as the Application Integration Crisis is one of the most critical Information Technology issues.

In the past many inspired ideas such as CORBA, Java RMI and Microsoft DCOM applied in order to solve the problem of software integration [1]. All of them were promising technologies but their main drawbacks were issues concerning security, complexity, vendor's proprietary standards and the absence of interoperability between these diverse protocols.

The great penetration of the Internet and the adoption of the XML technology enabled the creation of a new software integration technology widely known as Web Services. A quite good definition for Web Services is that "they are self-described and modular business applications that expose the business logic as services over the Internet through programmable interfaces and using Internet protocols for the purpose of providing ways to find, subscribe and invoke those services" [1].

One of the main open issues of Web Services is their process flow and control. Providing solution for applications interoperability is not enough, and there is a need for the specification and implementation of executable Business Processes [2].

In this paper the creation and design of executable Business Processes using technologies and standards such as XML, Web Services, and Java is investigated. Our proposal, the C.O.S.M.O.S. (Cooperation, Orchestration and Semantic Mapping of Services) environment, is an IDE (Integrated Development Environment) that enables the user to create, design and modify executable Business Processes based on the BPEL specification, an XML based language for the Orchestration and Choreography of Web Services.

2 Web Services & Business Modeling

The main characteristic of Web Services is the adoption of the XML technology as the core building block. This made Web Services independent from operating systems, platforms, vendors and programming languages [2]. The data

exchanged between Web Services and consumers are defined with XML. Even the description of Web Services and the protocols used by Web Services are defined with XML.

Another feature of Web Services is the operational and architectural model they use, known as SOA (Service Oriented Architecture) [1]. SOA is an evolution from the computer-based and object-oriented service model into a new type of applications where everything is encapsulated as a service that can be invocated over the Internet.

The concept of Web Services is to use XML defined protocols for communication (SOAP) , description (WSDL) and discovery (UDDI) of software services over the Internet (Fig.1).

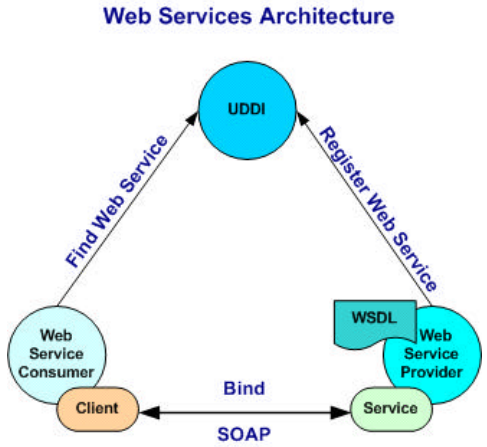

Fig.1, Web Services Architecture

Web Services seems to be the most promising technology of the near future in the area of software integration [4]. The greatest software vendors are investing huge amounts of money and effort to support this new technology.

Even though the future seems to belong to them, Web Services are still a new technology and in addition their inherited drawbacks, like the exchanged messages size (because of XML) and performance, there are some very important open issues such as process flow, transaction coordination, and message routing . A plethora of proposed specifications and protocols [5] trying to solve the various issues of Web Services, lead to the danger of loosing interoperability. For this reason the research community established the Web Services Interoperability Organisation (WS-I) which published the Basic WS-I Profile.

2.1 Business Process Execution Language (BPEL4WS)

Web Services promise the interoperability of applications in a vendor independent way. Only this is not enough for the systems integration between departments, organizations and partners, and for the implementation of automated and flexibly Business Processes and Workflows.

The Business Process Execution Language for Web Services (BPEL for short), an initiative from IBM, Microsoft, Siebel Systems, BEA and SAP, models the behaviour of Web Services in a business process interaction. It is the only specification today that models both the Orchestration and Choreography aspects of a Business Process.

Orchestration refers to the actual execution of a Business Process or Workflow. It controls the flow of the various activities internal to the process, like invocation of Web Services, messages handling, business logic and rules. On the other, Choreography describes the interfaces and the communication protocol between two or more partners. It tracks the message sequence between Web Services in an abstract manner.

BPEL is an XML based language that provides support for both executables (Orchestration) and abstract (Choreography) business processes [6], and is the technology used for the implementation of business processes design and execution in this project.

BPEL describes a business process using two aspects: Orchestration and Choreography (Fig.2) [6] and is the only specification today doing so.

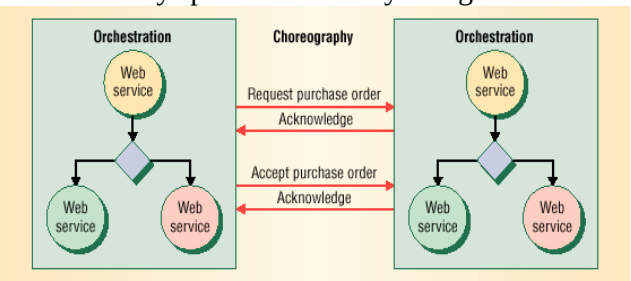

Fig.2, Web Services Orchestration and Choreography

2.1.1 BPEL Goals

According to the authors of the specification, the BPEL language has the following goals, which formed the base of their work [7]:

Web Services as the Base. The entities interacted in the business process are Web Services. The Web Services are defined using the Web Services Description Language (WSDL) and the interaction is abstract. That means that the BPEL uses the abstract interface of a Web Service to interact with it and not the actual reference to it.

XML as the Form. The Business Process is modelled using an XML based language. It describes a process with XML tags and in that way it gains portability. Any vendor can build its own implementation of design tools or execution engines (parsers) on the BPEL.

Common set or Core Concepts. Because each aspect of a Business Process (Orchestration and Choreography) needs specialized extensions for its expression, the BPEL provides a set of common XML elements for the core concepts required by both the external (abstract) and internal (executable) views of a Business Process.

Control Behaviour. BPEL provides two control regimes each inherited from its ancestors WSFL and XLANG: Hierarchical (a characteristic of XML) and Flow-based (a characteristic of Flowcharts).

Data Handling. The BPEL provides a limited set of data manipulation functions that are needed to define process-relevant data and control flow. Advanced data manipulation should be kept outside the process and implemented as a specific implementation's extensions or as invoked Web Services.

Properties and Correlation. BPEL defines a mechanism for identification of process instances. That mechanism is based on the data of partners' messages and can keep track of exchanged messages belonging or referencing to the same instance of the process (state-full communication).

Lifecycle. Support for implicit creation and termination of process instances is the basic lifecycle mechanism. When a message arrived, a process created and when the process reaches its terminal activities then it terminates.

Long-Running Transaction Model. BPEL defines a long-running transaction model that is based on compensation techniques and an adaptation of the Sagas and open nested transaction mechanism for processes.

Modularisation. A BPEL Business Process is itself expressed as a Web Service. So a Business Process can invoke other Business Processes or be used by them.

Composition with other Web Services Functionality. BPEL relies on compatible Web Services standards and standards proposals. If a needed standard does not exist then it is developed for the needs of BPEL as part of the language or as a different specification.

2.1.2 Features and Limitations

As noticed, BPEL is the only language today that supports both Orchestration and Choreography of Web Services. Indeed it seems to win the race for standardisation and global acceptance against other competitive protocols. The main advanced features of the language 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 [8].
- *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 the most promising and industry-adopted Web Services Orchestration and Choreography initiative today, it has also limitations and weaknesses with the most important being the following:

- *Complexity*. Even for modeling a simple process, the BPEL definition is extremely large and complex. Advanced workflow patterns are either very difficult and complicated or practically impossible to be implemented because of the resulting complexity of the produced process and the undocumented behavior 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 [8].
- *Overlapped Constructs*. Lack of Orthogonality is one of the most serious drawbacks of BPEL. There are many interleaved constructs and attributes of the language. Also some of them must be replicated leading to semantic redundancy.
- *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 [7] or to use the data manipulation capabilities of the XPath standard with its difficulties and restrictions [8].

- *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 occurring 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 [8].
- *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 [8].
- *No direct support for all types of Asynchronous Communication* [8]. The Publish/Subscribe and Broadcast types of asynchronous communication are not direct supported by BPEL.
- *Violation of XML Syntax and Conventional Rules.* The BPEL specification allows theoretical use of the character '<' in expressions as relational operator [8], but according to the XML Specification this character is strictly illegal .
- *Dependency on no-standard Protocols*. BPEL depends on the non standard WS-Addressing protocol for addressing. Indeed BPEL adds a non-standard extension to WSDL in order to define essential structures.

3 Development methodology

The findings of the evaluation of the BPEL and existing design tools formed the basis of the goals to be accomplished with the development of the C.O.S.M.O.S. Environment. The C.O.S.M.O.S. Environment is an application for the design and creation of Business Processes based on the BPEL specification.

During the development of the prototype of the environment some problems and difficulties were recognized. The first one was the difficulty in maintaining the consistency between the UML Class Diagrams and the Java Code. Because the round-trip engineering is not supported, a lot of time is wasted in every change of the design or the code. Another problem is the amount of time for the analysis and design of the software application before the actual code writing.

Current design methodologies require a lot of diagrams and when the design of the implementation begins, the good knowledge of the implementation programming language and its features is necessary. Furthermore in design methodologies there is no clear distinction between analysis and design phases when the diagrams are produced. The same analysis diagrams become gradually the design diagrams.

Today's well known software development methodologies are trying to address and solve some or all of the above problems emphasizing in some direction. For this reason the C.O.S.M.O.S. Environment developed with a different methodology, which took into account the best practices of the existing well-known methodologies. This methodology named "C.O.S.M.O.S. Software Development Process" or "C-SDP".

3.1 The C.O.S.M.O.S. Software Development Process (C-SDP)

The principal idea behind C-SDP 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.

C-SDP considers that a software application can be conceptually approached as a combination of the following layers (Fig.3):

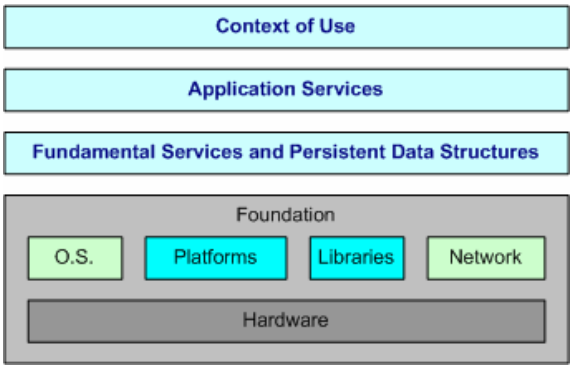

Fig., 3 Software Applications Conceptual Architecture

- **Context of Use**. 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 & 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**. 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.

The main advantage of the C-SDP is that even though the requirements of a software application are not yet fully specified and documented in the early stage, it is possible for the engineers to work in parallel on the different layers of the application. The activities of the C-SDP, which were used in the development of the C.O.S.M.O.S. Environment, are illustrated in Fig.4.

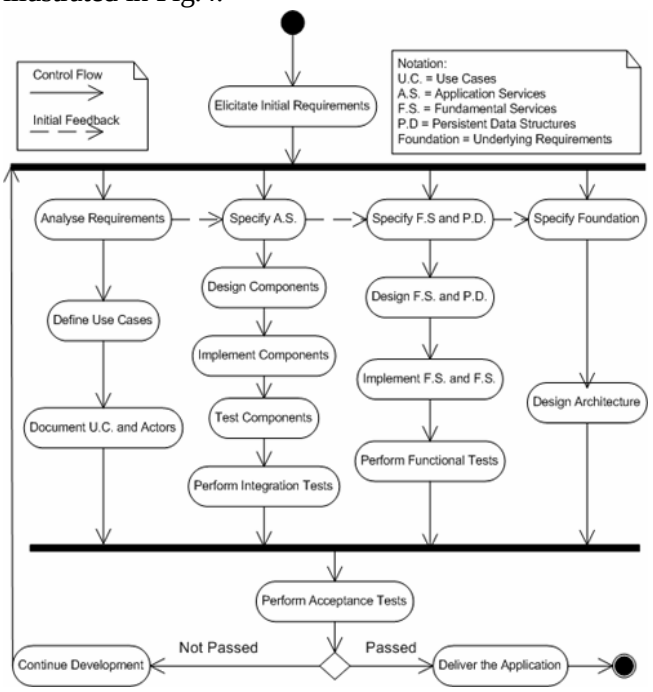

Fig.4, C-SDP Activities

4 The C.O.S.M.O.S. Environment

The goal of the C.O.S.M.O.S. Environment is to help its user to design and create an executable Business Process. The application should provide a complete environment that would allow the user to create, design, code, verify and deploy a Business Process based on the BPEL specification.

Since it is difficult to address the needs of different categories of users like Managers and Developers, C.O.S.M.O.S. Environment, in respect of the different levels and areas of knowledge of its users, uses two "views": The Manager's Perspective and the Developer's Perspective. Each one provides the services needed for each category of users. The first one is a visual design environment with drag n' drop capability of the activities of a process, and the second one provides an XML editor for BPEL coding.

In addition the application should be as simple as could be without unnecessary extra functionalities that could confuse the users. The spirit of simplicity and formality influenced the requirements of the application.

4.1 Specification and System Architecture

The specification of the application services led to the required components and the architecture of the C.O.S.M.O.S. Environment. The application is based on a layered and components-based architecture (Fig.5), which enabled flexibility for future improvements and addition of new components or replacement of the existing.

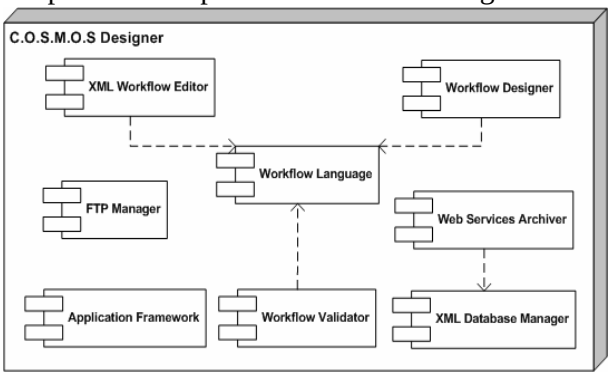

Fig. 5, C.O.S.M.O.S. Environment Components Diagram

The main components of the application are:

- **Application Framework**. Bound and controlled the components of the application. It is also responsible for the graphical user interface (GUI) of the application.
- **Workflow Designer**. Provides visual representation of a Business Process as a Flowchart and enables the user to create and design business processes.

- **XML Workflow Editor**. An XML Editor that supports XML Workflow Design Languages. Providing code coloring, indentation and a code assistant specific to the implemented workflow language. Currently only the BPEL language is supported.
- **Workflow Validator**. Validates the produced business process. It is based on the XML Schema of the implemented workflow language.
- **Workflow Language**. Provides the actual implementation of the used Workflow Language. Currently only the BPEL language is implemented.
- **FTP Manager**. It is responsible for the uploading of files into an FTP server.
- **Web Services Archive**. Provides an interface for administering libraries of ready to use Web Services.
- **XML Database Manager**. Provides an interface for manipulating data, stored in XML Databases.

5. Conclusion & future work

With C.O.S.M.O.S. Environment, Managers and Developers can create or change their Business Processes using a friendly and easy to learn environment. C.O.S.M.O.S. Environment enables its users, to design an executable Business Process, create a new one from available templates, and upload them to a BPEL engine for execution. Managers can work on a visual environment with drag n' drop functionality producing a flowchart which represents a workflow or business process of their company. Developers can use an XML editor for the coding of BPEL processes and the creation of more advanced workflows.

Business Process Design based on Web Services, and the Web Services Orchestration / Choreography are exciting areas for challenging and innovative research. The ambition of C.O.S.M.O.S. environment is to become the foundation of further research in this and relative areas.

Ongoing research in the C.O.S.M.O.S. Environment includes the performance improvement of existing components and inclusion of new functionalities. The vision is that C.O.S.M.O.S. Environment along with C-SDP would become the starting point of the research on the design and development of a new processes framework for Web Services that would incorporate a new XML Workflow Language, the *C.O.S.M.O.S. Workflow Language*, which should overcome the limitations and the weaknesses of the today's implementations along with an Execution Engine of this language with monitoring, security and quality of service guarantee mechanisms.

References:

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

[2] Langdon, C. S. The State of Web Services, *IEEE Computer*, Vol. 36, No 7, 2003, pp. 93-94.

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

[4] Chung, J. Y., Lin, K. J. & Mathieu, R. G. Web Services Computing: Advancing Software Interoperability. *IEEE Computer*, Vol. 36, No 10, 2003, pp. 35-37.

[5] Turner M., Budgen D. & Brereton P. Turning Software into a Service, *IEEE Computer*, Vol. 36, No 10, 2003, pp. 38-44.

[6] Peltz, C. Web Services Orchestration and Choreography. *IEEE Computer*, Vol. 36, No 10, 2003, pp. 46-52.

[7] Frank Laymann, Dieter Roller & Satish Thatte, Goals of the BPEL4WS Specification, 2003.

[8] BEA, IBM, Microsoft, SAP AG & Siebel Systems, *Business Process Execution Language for Web Services Version 1.1*, May 2003.

[9] P. Wohed et al., Analysis of Web Services Composition Languages: The Case of BPEL4WS, Springer-Verlag, 2003.