BPEL - based Web Services Composition

Elena Ivanova

Abstract: *Technologies for web services choreography and orchestration are considered. These technologies and standards have appeared as a natural growth of web service technologies. The main features and elements of BPEL standard are presented, because of its wide acceptance. It can be used both for choreography and for orchestration. The set of major characteristics of software orchestration tools are discussed. A case study about particular service orchestration software suits, based on BPEL, is presented.*
Keywords: *Web Services, Choreography, Orchestration, BPEL*

1. Introduction

Technologies for web services choreography and orchestration are considered. These technologies and standards have appeared as a natural growth of web service technologies. These technologies address the need to find, group and compose the web services available and form more complex and meaningful business process. To do so the component services should be equipped with suitable interfaces, providing interoperability and location transparency.

There is a large amount of standards for web services composition. Nevertheless, it is clear that BPEL standard dominates, having in mind all the software tools for composition based on it.

The paper is organized as follows. The next section explains briefly two aspects of web services compositions – choreography and orchestration. Available standards for choreography and orchestration are described. In the following is overview of the main features of BPEL standard. After that the attention is paid on the software tools providing means for orchestration. A particular service orchestration software suit is presented in the fifth section. The paper finishes with some concluding remarks.

2. Web Services Choreography and Orchestration

Choreography refers to global, multiparty, peer-to-peer collaborations where the component entities interact in long-lived stateful and coordinated way regardless of any programming model or supporting platform used. The W3C glossary [5] gives the following definitions of the term "choreography":

- *"A choreography defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state.*
- *Web Services Choreography concerns the interactions of services with their users. Any user of a Web service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. Transactions among Web Services and their clients must clearly be well defined at the time of their execution, and may consist of multiple separate interactions whose composition constitutes a complete transaction. This composition, its message protocols, interfaces, sequencing, and associated logic, is considered to be a choreography."*

Choreography languages (examples are WSCI, WS-CDL, etc) are mainly descriptive and cannot be directly executed. These languages are necessary to be mapped to an orchestration language in order to be executed.

Common characteristics of choreography languages are summarized in the following [18]:

- they concentrate on the specification of the *observable behaviour* of communicating services;
- they describe which messages are exchanged between services in a specific situation;

- they do not show *how* the services should be *implemented* so as to exchange the messages.

Short descriptions of choreography languages are presented below.

WS-CDL (Web Services – Choreography Description Language) [21] is an XML-based choreography language that allows defining from a global viewpoint the observable behaviour of different business entities. WS-CDL is a choreography language for which orchestrations can be implemented using BPEL, XPDL, etc. It is not an executable language. WS-CDL is based on XML, XML-NS, XML-S XPointer and WSDL.

WSCI (Web Service Choreography Interface) [21] was submitted as a note to W3C for the work on WS-CDL by BMPI and a set of companies. It was mainly created by BEA, Intalio, SAP and Sun. WSCI is an XML-based interface definition language that describes the flow of messages exchanged by a Web service participating in choreographed interactions with other services. WSCI describes the dynamic interface of the Web service participating in a given message exchange by means of reusing the operations defined for a static interface specified using WSDL.

WSCL (Web Services Conversation Language) [21] is a simple conversation definition language. The purpose of WSCL 1.0 is to provide and define the minimal set of concepts necessary to specify conversations. The expectation is that WSCL will be extended for more complex Web services frameworks; for example, multi-party conversations, quality of service attributes, transactions, or composition of conversations. WSCL has never reached the status of a standard. It was published as a W3C Note in 2002 without any further improvements.

There are languages, like BPEL and BPSS, which can be used both for choreography and orchestration. These will be described with orchestration languages.

Term Orchestration refers to coordinating of multiple tasks. So, in terms of services Orchestration means coordinating of several services (i.e. their relations and consequence of execution) so that the result is a complex service or a process composed by these services. The W3C glossary [5] gives the following definitions of the term “orchestration”:

- “*An orchestration defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. I.e., an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal*”.

Requirements for Web Services Orchestration

In order for web services to be composed into a complex service or business process, there are a set of requirements which should be addressed. Several works and papers deal with such requirements [3, 4]. The main of them can be summarized as follows:

- ability to invoke services in an asynchronous manner – the system has to enable services to be invoked concurrently, not only sequentially;
- ability to manage transaction and compensation – this is crucial for long-running compound services – in case of a failure in a single component service, the system has to provide adequate compensation;
- exception handling – how the system will behave when some error occurs;
- security and reliable messaging – security is a key point in all the layers in web service technologies;
- support the separation of abstract process logic and concrete web services used – it is necessary for building dynamic and flexible processes.

Almost all available languages for web services orchestration have mechanisms respecting the aforementioned requirements.

Orchestration focuses on the behaviour of the composed services as a whole. Orchestration languages (e.g. BPLM, BPEL, XPDL, BPELJ, jPDL, etc) are executable languages and define a runtime environment for their execution. Short descriptions of these languages are presented below in alphabetical order.

BEPLJ (BPEL for Java) [13] - It is a combination of BPEL and Java proposed by IBM and BEA that allows sections of Java code, called Java snippets, to be included in BPEL process definitions. Snippets are expressions or small blocks of Java code that can be used for things such as: loop conditions, branching conditions, variable initialization, Web service message preparation, logic of business functions etc. BPELJ introduces a few minor changes to BPEL as well as several extensions in order to fit BPEL and Java conveniently together. There is an effort to standardize BPELJ as part of JSR 207.

BPEL (Business Process Execution Language) [17] - This is a convergence of WSFL and XLANG, written by developers from BEA, IBM, SAP, Siebel and Microsoft. It is an XML-based language, depending on WSDL, XML-Schema and XPath, providing a language for the formal specification of business processes and interaction protocols. It supports both abstract and executable processes. In BPEL, a business process is composed of elements ("activities") that define activity behaviors, including the ability to invoke Web services and control flow, and to compensate when errors occur. The resulting business process is exposed as one or more Web services.

BPML (Business Process Modelling Language) [9] – This is a textual service-oriented process modelling language, based on XML, XML-NS, XML-S, XPath and WSDL. It provides an abstracted execution model for collaborative and transactional business processes. It is a strict superset of BPEL, although BPEL have been endorsed by for BPMI phase 2.0 instead of BPML. A process is viewed as a series of activities and an activity represents a component that performs a specific function. Activities can be composed into complex activities. Activities execute within a context which is transmitted from parent to child. In particular the context allows two activities to share properties. Properties constitute the data flow of BPML.

BPSS (Business Process Specification Schema) [12] - BPSS provides a standard framework (language) for business process specification. As such, it works with the ebXML Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) specifications to bridge the gap between Business Process Modelling and the configuration of ebXML compliant e-commerce software.

jPDL (jBPM Process Definition Language) [14] - It is an XML based process execution language running in the Open Source JBoss jBPM (Java Business Process Management) workflow management system. Java actions can be triggered from the language.

JSR 207 (Java Specification Request) [15] is a draft JCP (Java Community Process) specification that is defining metadata, interfaces, and a runtime model that enables business processes to be easily and rapidly implemented using the Java language and deployed in J2EE containers. It will support tasks commonly encountered when programming business processes, e.g. parallel execution and asynchronous messaging.

Orch [1], [10] - This is a programming language and system for orchestrating distributed services. Orc has a strong theoretical foundation that supports modular composition and analysis of concurrent programs. The Orc model [9] assumes that basic services, like sequential computation and data manipulation, are implemented by primitive sites. Orc provides constructs to orchestrate the concurrent invocation of sites to achieve a goal – while managing time-outs, priorities, and failure of sites or communication.

XLANG [16] - It is an extension of WSDL, providing both the model of an orchestration of services and collaboration contracts between orchestrations. It has been standardized by Microsoft and has been used as a base by BPEL. XLANG, like BPML, were designed with an explicit -calculus theory foundation. WSFL and XLANG have converged to BPEL.

XPDL (XML Process Definition Language) [18] - It is an XML-based language from the Workflow Management Coalition (WfMC) for defining business processes. XPDL was based around a common set of functions for work distribution found in most workflow products. The XPDL's syntax is specified by an XML Schema.

WSFL (Web Services Flow Language) [13] - It is an XML based language, standardized by IBM, for the description of Web Services compositions as part of a business process definition. It relies and complements existing specifications like SOAP, WSDL, XMLP and UDDI. WSFL considers two types of Web Services compositions: the first type specifies an executable business process known as a flowModel; the second type specifies a business collaboration known as a globalModel.

3. BPEL – main features

In its early versions BPEL was named as WSBPEL or BPEL4WS - Web Services Business Process Execution Language or Business Process Execution Language for Web Services.

BPEL is:

- is an XML-based language designed to enable task-sharing for a distributed computing;
- is written by developers from BEA Systems, IBM, and Microsoft;
- combines and replaces IBM's WebServices Flow Language (WSFL) and Microsoft's XLANG specification;
- depends on WSDL, XML Schema, and XPath;
- supports both abstract processes and executable processes:
    1. Abstract processes are useful for specifying expected protocols and publicly visible behaviors without too much detail;
    2. Executable processes contain enough detail to fully specify execution.
- BPEL is essentially a layer on top of WSDL, with WSDL defining the specific operations allowed and BPEL defining how the operations can be sequenced.

In BPEL, a business process is composed of elements ("activities") that define activity behaviors, including the ability to invoke Web services and control flow, and to compensate when errors occur. The resulting business process is exposed as one or more Web services. The BPEL elements are listed below.

- Partners / Partner Links - the different parties that interact with the business process
- Variables - Data variables used by the process; Persistent for long running interactions; Defined in WSDL types and messages
- Correlation Sets - Set of properties shared by all messages in a correlated group
- FaultHandlers - Activities that must be performed in response to a fault;
- CompensationHandlers - allowing the process designer to implement compensation actions for certain irreversible errors in business (wrapper for a compensation activity);
- EventHandlers - invoked concurrently if the corresponding event occurs;
- Activities (Flow Logic) - the actions that are being carried out within a business process. There are two types of activities:
    - basic activities (<receive> <reply> <invoke> <assign> <throw> <terminate> <wait> <empty>);
    - structured activities (<sequence> <switch> <while> <pick> <flow> <scope> <compensate>).

The following figure 1 shows the basic structure of a BPEL document.

```
<process name="TheProcess">
  <partners>
          <!-- lists the external web services invoked from within the workflow -->
  </partners>
  <variables>
          <!-- specifies the data elements that flow within the workflow -->
  </variables>
  <correlationSets>
          <!-- specifies bindings for a set of operations to a service instance -->
  </correlationSets>
  <faultHandlers>
          <!-- lists the elements to catch faults -->
  </faultHandlers>
  <compensationHandler>
          <!-- specifies the elements that implement compensating actions in the case
  of transaction rollback -->
  </compensationHandler>
  <eventHandlers>
          <!-- for receiving external events to the workflow -->
  </eventHandlers>
  <sequence>
          <!-- the workflow execution logic -->
  </sequence>
</process>
```

Fig.1 Basic structure of a BPEL document

It is very hard to itemize all the software tools for service orchestration based on BPEL. The list is quite long. Some of these products are mentioned in the following Table 1.

Table 1: BPEL based software tools

ActiveBPEL Engine	Bexee
ActiveBPEL Designer	Biztalk Server
ADONIS	Cape Clear Orchestrator
Apache Agila	iGrafx BPEL
BEA WebLogic	MidOffice
Oracle BPEL Process Manager	PXE
Parasoft BPEL Maestro	SAP NetWeaver Exchange Infrastructure

4. Software Orchestration Tools

The special attention in this section is paid on the tools for web service orchestration, because they provide a run time environment for execution of the business processes.

Service orchestration tools are usually part of complex workflow applications (applications for automation of business processes). A large amount of service orchestration software tools is available. Our studies came across more than 90 products. The list, of course, is not complete, but it is sufficient to illustrate the importance of the developments in such application area. The products have different level of maturity and they are presented both as market available products and as an open source software suits.

The main functional characteristics of service orchestration tools can be derived from the Workflow Reference model [7], which defines the main components and interfaces within the workflow architecture. In short the five WfMC interfaces can be characterized as follows. For details refer to [2].

- *Interface 1* refers to exchange of workflow descriptions, for example models, choreographies, or orchestrations, between modelling tools and other modelling tools, repositories, workflow execution tools.
- *Interface 2* considers the invocation of workflow engines by workflow clients.
- *Interface 3* supports the invocation of applications performing the specified workflow activities.
- *Interface 4* supports workflow collaboration.
- Interface 5 supports the monitoring and administration of workflows including the life cycle management of distributed workflows.

The following functionalities are specific to the service orchestration tools:

- specification of executable processes – creating and specifying processes, in accordance with some orchestration standard like BPEL and XPDL;
- simulation – simulating and testing the process execution using sample data and breakpoints;
- deployment – deploying the process onto the appropriate server location;
- execution – executing of the deployed processes;
- administration and monitoring – managing and monitoring of the running processes.

There are tools, supporting all these functions, and others, supporting only the first or/and the second function. Thus, service orchestrating products can be divided into three categories, according to their major functionality: editors (supporting mainly the first function – specification and/or simulation), engines (supporting mainly the second function – deployment and execution) and complex tools (supporting all aforementioned functions).

5. Service Composition with ActiveBPEL software suit

A case study in this section will illustrate the described functionalities of particular service orchestration tools: ActiveBPEL Designer and ActiveBPEL Enterprise, the products of Active Endpoints [6, 8], one of the leading companies in this area.

ActiveBPEL Designer is a comprehensive visual tool (editor) for creating, testing and deploying composite applications, based on the BPEL standard. It is a plug-in to the Eclipse (www.eclipse.org) integrated development environment.

Interfaces – according to Workflow Reference Model [7] briefly described above.

Interface1: ActiveBPEL Designer provides interfaces at key process definition points:

- Create new BPEL activities via drag-n-drop from business operations defined in Web Services Description Language (WSDL) files;
- Generate BPEL-specific WSDL extensions for Partner Link Types;
- Generate BPEL-specific WSDL extensions for message properties and property aliases.

For modeling processes **ActiveBPEL Designer** uses its own graphical notation language.

In **ActiveBPEL**, one can build two kinds of business processes: Abstract and Executable.

The following figure 2 [6] shows that one can start the process either by using WSDL files in existence or by building an abstract BPEL process first and then create the WSDL files for it.

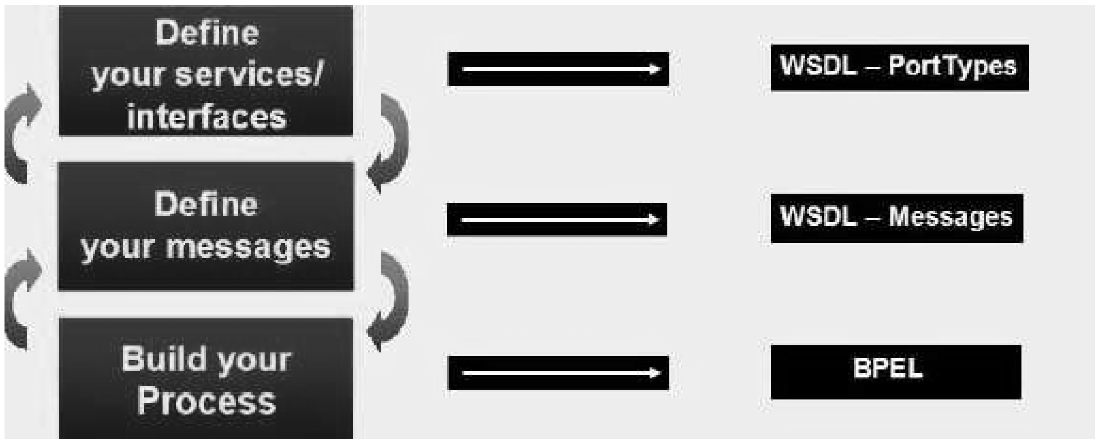


Fig.2. Creating processes [6]

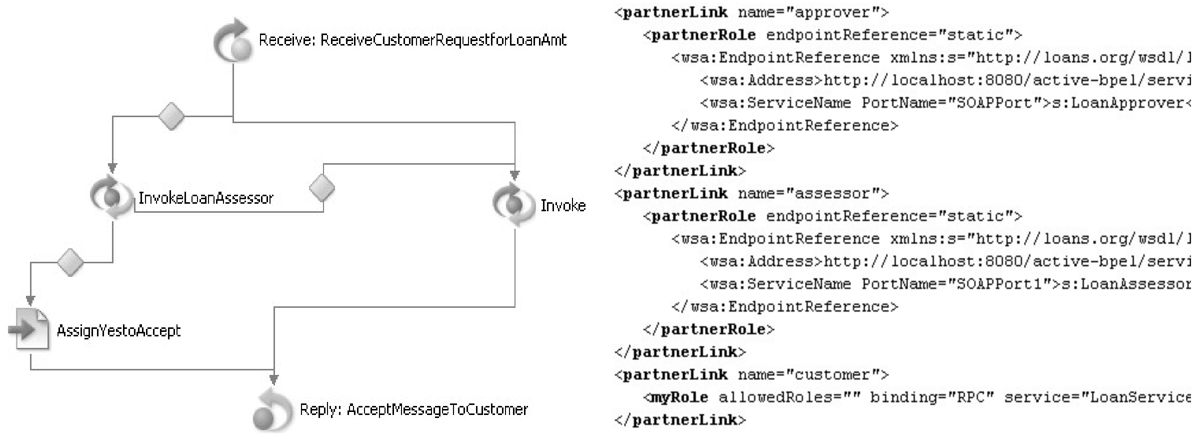
Interface2: ActiveBPEL Designer generates a *.bpel* file from the created diagram and generates *.wsdl* file for the diagram itself.

Supported standards: WSDL, BPEL, XML data files: A process can be created diagrammatically on the canvas and the Designer generates the corresponding BPEL code. WSDL files can be added to the Web References view for automatic discovery and organization of all pertinent information stored in existing WSDL.

Specific orchestration functions

Having in mind the five specific functions of orchestrating tools, outlined above, the tool covers three of them - specification, simulation and deployment of executable complex business processes.

Specification. Processes are built by choosing partners, services and operations, and defining how data flows among those entities. It is done by organizing icons on the Process Editor canvas, and at the same time the tool constructs valid BPEL code. Figure 3 presents a typical picture of an ActiveBPEL process on the left hand side, and the corresponding BPEL code snippet as generated by ActiveBPEL Designer on the right hand side.


Receive: ReceiveCustomerRequestforLoanAmt
InvokeLoanAssessor
Invoke
AssignYestoAccept
Reply: AcceptMessageToCustomer

```
<partnerLink name="approver">
   <partnerRole endpointReference="static">
      <wsa:EndpointReference xmlns:s="http://loans.org/wsdl/l
         <wsa:Address>http://localhost:8080/active-bpel/servi
         <wsa:ServiceName PortName="SOAPPort">s:LoanApprover<
      </wsa:EndpointReference>
   </partnerRole>
</partnerLink>
<partnerLink name="assessor">
   <partnerRole endpointReference="static">
      <wsa:EndpointReference xmlns:s="http://loans.org/wsdl/l
         <wsa:Address>http://localhost:8080/active-bpel/servi
         <wsa:ServiceName PortName="SOAPPort1">s:LoanAssessor
      </wsa:EndpointReference>
   </partnerRole>
</partnerLink>
<partnerLink name="customer">
   <myRole allowedRoles="" binding="RPC" service="LoanService
</partnerLink>
```

Fig.3 ActiveBPEL process

Simulation. The created process is simulated by using sample data and setting break points. Management of sample data is performed by adding data files for all WSDL messages for a convenient repository of test data across all processes using the messages. Multiple files can be added for various test scenarios. During simulation, it can be tested various execution paths using different data. Set breakpoints, step through or run the process. It can be performed remotely debug of a process running on the server.

Deployment. The necessary files for deployment are automatically packaged and deployed onto the appropriate server location. Deployment wizards guide the user to provide endpoint references for services used in your process. A process deployment descriptor provides error-free techniques for binding your services. Process is automatically deployed to the appropriate server location within a package that contains all required files.

ActiveBPEL Enterprise is a complete BPEL engine running either on top of a J2EE application server or standalone with a web servlet container. ActiveBPEL Enterprise servers are high performance BPEL servers that deliver many enterprise features including static analysis, process persistence, process versioning, extensive runtime web console, programmable Web Service and Java APIs plus diagram-based diagnostics.

The core of ActiveBPEL Enterprise is the ActiveBPEL Engine, which is an open sorce product. ActiveEndpoints is the first commercial BPEL product company to commit the source code of your core engine technologies into open source through ActiveBPEL initiative, giving the necessary power and control to understand the internal workings of ActiveBPEL Enterprise.

Interfaces - according to Workflow Reference Model

Interface1: ActiveBPEL Enterprise uses BPEL descriptions for the processes and WSDL descriptions for the services. They files can be created manually or using ActiveBPEL Designer.

Interface2: ActiveBPEL engine starts with the servlet container. In order for the engine to execute a BPEL process, several files must be packaged in a business process archive (.bpr) file for deployment. The basic business process archive includes:

- BPEL process (.bpel file)
- WSDL file(s) referenced in the BPEL process (.wsdl file)
- WSDL catalog (.xml file)
- Process deployment descriptor (.pdd file)
- Optionally, a partner definition file (.pdef file)

These files can be created manually or using ActiveBPEL Designer.

Interface3: Web Services applications with a valid partner link type in the associated WSDL can be invoked using generic SOAP client. ActiveBPEL Enterprise also includes a J2SE compliant mechanism for handling Web service requests and Web service invocations.

Interface4: The collaboration should be possible and be based on SOAP/WSDL standards.

Interface5: ActiveBPEL Enterprise provides comprehensive management and administration capabilities. Web-based consoles allow the configuration and administration of the server's execution environment and deployed and running processes.

Supported standards: WSDL and BPEL for specifying services, XML data files, SOAP for communication.

Specific orchestration functions

This tool covers three of specific functions – deployment, execution, and administration and monitoring of executable business processes.

Deployment. In order for the engine to execute a BPEL process, several files must be packaged in a business process archive (.bpr) file for deployment. The Deployment console pages allow the processes and their artifacts to be managed. Deployment functionalities include:

- Deployment of Business Process Archive (.bpr) files
- Deployment logs for each .bpr file for errors and warningsLists of all deployed processes with selection filters

- Detailed version information including the ability to modify specific data based on the version state (e.g., status, effective and expiration dates)
- Detailed information regarding other artifacts deployed with process
- WSDL files deployed in the BPR
- Global Partner definition files deployed to the server

Execution. ActiveBPEL Enterprise provides an enterprise-level business process execution engine running as an application in the Tomcat server. It can be used to execute BPEL compliant processes, to manage the server's configuration, to suspend or resume running process instances. The Engine console pages allow to be viewed and managed settings of the engine, main of which are listed in the following table:

Table 2: ActiveBPEL Engine settings

Engine Settings	**Description**
Configuration of ActiveBPEL Enterprise	• Enable and disable process instance logging • Turn on and off message validation • Enable or disable BPEL extensions • Set engine caching parameters
Detailed license information	Add and remove licenses
Persistent storage maintenance	• Prune completed processes by date • Prune deployment logs by date • Prune deployed process definitions
Detailed information regarding the version of ActiveBPEL Enterprise installed	Version and build numbers for all ActiveBPEL Enterprise components

Administration and monitoring. Web-based consoles allow the configuration and administration of the server's execution environment and deployed and running processes. Statistics data for deployed, active, completed, terminated processes are available. The Process Status console pages allow one to view and manage information associated with individual process instances. These pages provide the ability to sort and filter the information in ways that make easier to deal with only the most relevant process information. For running and completed processes one can use the consoles to perform the following tasks:

- Suspend, Resume and Terminate executing processes
- View the process execution state graphically, hierarchically, and textually
- Examine the content and status of process variables, activities and links

For the persistent queues that allow for the re-hydration of processes on server startup you can inspect the values of:

- Alarms -- events that will execute at a specific time as specified by executing processes
- Receives -- the result of a BPEL activity or Event Handler specifying that a message is expected

6. Conclusions

A lot of technologies are involved in the area of web services composition. Two of them - BPEL and XPDL, especially BPEL, have found wide acceptance, having in mind the software products, which are based on them. The list of software tools, supporting service orchestration, is quite long, and shows the importance of the developments in such application area. There are at least five major functional characteristics, specific to such kind of products – specification, simulation, deployment, execution and administration of executable processes.

A brief overview of standards for web services composition has been presented. Functionalities for services choreography and orchestration, supported by BPEL standard have been described. Software tools for orchestration have been discussed. Particular software suit, providing functionalities for service orchestration (editor and engine): ActiveBPEL Designer and ActiveBPEL Enterprise, has been illustrated.

References

[1] Misra J.and W. Cook, 2004, *Computation Orchestration: A Basis for Wide-Area Computing,* Lecture Notes for NATO summer school, held at Marktoberdorf in August 2004

[2] Hollingworth D.; *The Workflow Reference Model –10 Years On*; Workflow handbook 2004

[3] Peltz C., Web ser vices orchestration, available at http://devresource.hp.com/drc/technical_ white_papers/WSOrch/WSOrchestration.pdf

[4] Yushi C., L. E. Wah and D. K. Limbu, Web Services Composition - An Overview of Standards, Singapore Institute of Manufacturing Technology

[5] W3C, *Web Services Glossary*, W3C Working Group Note, 11 February 2004, available at http://www.w3.org/TR/ws-gloss/.

[6] ActiveWebflow™ Designer User's Guide

[7] WfMC: *Workflow Reference Model*; available at http://www.wfmc.org/standards/model.htm

[8] http://www.active-endpoints.com/

[9] http://www.bpmi.org

[10] http://www.cs.utexas.edu/users/wcook/projects/orc/

[11] http://www.developer.com/

[12] http:// www.ebxml.org

[13] http://www.ibm.com

[14] http:// jbpm.org

[15] http://www.jcp.org/en/jsr/

[16] http://www.microsoft.com

[17] http://www.oasis-open.org/home/index.php

[18] http://wfmc.org/

[19] http://www.omg.org

[20] www.visp-project.org/

[21] http://www.w3.org/TR/

Assistant prof. Elena Todorova Ivanova, BAS – ICCS, telл. 02 979 2774, e-mail: e_ivanova@hsh.iccs.bas.bg.

This research is partly supported by the European Commission, project №FP6-027178, and National Scientific Fund of Bulgaria, project № ВУ-МИ-108/2005.