# INFRAWEBS BPEL-BASED EDITOR FOR CREATING THE SEMANTIC WEB SERVICES DESCRIPTION

Tatiana Atanasova

*Abstract: INFRAWEBS project [INFRAWEBS] considers usage of semantics for the complete lifecycle of Semantic Web processes, which represent complex interactions between Semantic Web Services. One of the main initiatives in the Semantic Web is WSMO framework, aiming at describing the various aspects related to Semantic Web Services in order to enable the automation of Web Service discovery, composition, interoperation and invocation. In the paper the conceptual architecture for BPEL-based INFRAWEBS editor is proposed that is intended to construct a part of WSMO descriptions of the Semantic Web Services. The semantic description of Web Services has to cover Data, Functional, Execution and QoS semantics. The representation of Functional semantics can be achieved by adding the service functionality to the process description. The architecture relies on a functional (operational) semantics of the Business Process Execution Language for Web Services (BPEL4WS) and uses abstract state machine (ASM) paradigm. This allows describing the dynamic properties of the process descriptions in terms of partially ordered transition rules and transforming them to WSMO framework.*

*Keywords: Semantic Web Services, BPEL, ASM, WSMO*

*ACM Classification Keywords: H.3.4 Systems and Software: Information networks*

A little semantics goes a long way.
*James Hendler*

## Introduction

The power of Web services can be realized only when appropriate services are discovered based on the functional requirements given by functional semantics. Including the functional semantics to the web services descriptions is a step forward to the Semantic Web.

The Semantic Web is an extension of the current World Wide Web. It builds on the current World Wide Web constructs and topology, but adds further capabilities by defining machine-processable data and relationship standards along with richer semantic associations [KM-GOV, 2005]. Semantic Web Services are integrated solution for realizing the vision of the next generation of the Web; they define semantically driven technologies for automation of the Web Service usage process.

The semantics of a Web Service is the shared expectation about the behavior of the service [W3C, 2004] and give more expressive descriptions about the service, for example, the intent of the service or kind of data being deal with.

Semantics for a service can be:

- Implied
- Expressed in human-understandable form
- Expressed in machine-readable form

As a step in the direction to the semantic services creation, accession and reusing within the Semantic Web the current INFRAWEBS research project [Nern et al, 2004] may be considered. The project uses semantics during the complete lifecycle of Semantic Web processes. One of the main components of INFRAWEBS project is SWU – Semantic Web Services Unit [Atanasova et al, 2005]. SWU is intended to converting web services from available descriptions and domain knowledge to the semantic ones and to Compose Web Services, through combining and orchestrating them in order to deliver added-value services.

The appropriate solution for realizing these aims may be found using semantic approach to describing and processing of Web Services. One of the initiatives in Semantic Web is WSMO – Web Service Modelling Ontology (www.wsmo.org). The INFRAWEBS project relies on the WSMO framework.

Current standards for service description (for example, WSDL, UDDI, BPEL) have no semantically marked up content. In [Cardoso, 2005] it is pointed that *"During Web service development data, functional and QoS semantics of the service needs to be specified… it is very important to use semantics at this stage."*

Several approaches have already been suggested for adding semantics to Web services. Semantics can either be added to currently existing syntactic Web service standards like UDDI or WSDL or services can be described using some ontology based description language like DAML-S [Patil et al, 2004]. Only in [Mandell, 2003] it is made an assumption for using BPEL as a base for semantic annotation by proposing to "take a bottom-up approach to integrating Semantic Web technology into Web Services". But they mainly focus on introducing a semantic discovery service and facilitate semantic translations.

In this paper an approach is proposed for migrating from BPEL to WSMO description by semi-automatic mapping BPEL constructions to WSMO descriptions of service interface using Abstract State Machine paradigm.

## BPEL

The Business Process Execution Language for Web Services (BPEL) at present is the most well-known language to specify and execute business processes, using Web Services as its technological basis. BPEL is built on top of WSDL and indirectly on SOAP and introduces a stateful interaction model that allows exchanging sequences of messages between business partners (i.e. Web Services).

The definition of business process in BPEL expresses business logic and describes process model elements in terms of business messages exchange and consists of:

- process partners to which web services are connected;
- variables, which define the state of the process;
- activities (basic and structural), which define the behaviour of the business process. The basic activities define tasks, which are accomplished in business process. The structural activities define the control flow.

BPEL is a block-structured programming language; it allows recursive blocks, but restricts definitions and declarations to the top level. BPEL document contains partner links, variables (message containers), and activities (process programs). Partners are external Web services. Activities are the actions or tasks performed within the business process and are the basic components of a process definition. Structured activities prescribe the order in which a collection of activities take place. Sequential control between activities is provided by *sequence*, *switch*, and *while*. Concurrency and synchronization between activities is provided by *flow*. Nondeterministic choice based on external events is provided by *pick*.

Variables are process instance-relevant data, providing their definitions in terms of WSDL message types, XML Schema simple types, or XML Schema elements. Variables allow processes to maintain state data and process history based on messages exchanged.

BPEL defines a mechanism for catching and handling faults. Fault handlers catch faults when they are thrown by other actions. There is also a compensation handler to enable compensatory activities in the event of actions that cannot be explicitly undone. BPEL does not support nested process definition.

It is possible to use abstract and executable BPEL processes. Abstract process is a business interaction protocols; executable process models actual behaviour of a participant in a business interaction and it may be compiled into invokable services. The structure of BPEL-based process is as follows:

```
<process name = "…">
    <partnerLinks>
    …
    </partnerLinks>
    <variables>
    …
    </variables>
```

```
        <flow>
            <links>
            …
            </links>
            <!-- activities -->
            …
        </flow>
    </process>
```

The possibility to represent the BPEL-based business process as directed graph helps for formal description of operational behaviour and extraction of abstract functional semantics [Farahbod, 2004].

## ASM and BPEL

In [Farahbod, 2004] an Abstract Syntax Tree is proposed to capture the complete structure of a BPEL process. This representation will be used for formalization of BPEL file by ASM.

First of all, let's recall a definition of functional (operational) semantics: *Operational semantics* consists in describing the steps of the computation of a program by giving formal rules to derive judgments of the form *{p, a} -> r*, to be read as "the program *p*, when applied to the input *a*, terminates and produces the output *r*".

Operational semantics is a way to give meaning to computer programs in a mathematically rigorous way [Wikipedia]. The operational semantics for a programming language describes how a valid program is interpreted as sequences of computational steps. These sequences are the meaning of the program.

For the formal definitions of functional semantics of programming languages Abstract State Machines (ASM) are now widely used.

ASM's definition is given in [Börger, 1999]:

- An ASM *M* is a finite set of rules for guarded multiple function updates;
- Applying one step of *M* (a set of rules) to a state (algebra) *A* produces as next state another algebra *A'* of the same signature obtained as follows:
  - First evaluate in *A* using the standard interpretation of classical logic all the guards of all the rules of *M*
  - Compute in *A* for each of the rules of *M* whose guard evaluates to true all the arguments and all the values appearing in the updates of this rule
  - Replace simultaneously for each rule and for all the locations in question the previous *A*-function value by the newly computed value
  - The algebra *A'* thus obtained differs from A by the new values for those functions at those arguments where the values are updated by a rule of M which could fire in A

A distributed ASM – DASM [Börger, 1997] is given by a set of agents and a program function *Mod* which assigns to each agent a module ("sequential" program) consisting of a finite number of so called transition rules of the form "If *Cond* then *Updates*", where *Cond* is any expression (of first order logic) and *Updates* is a finite set of function updates, i.e. of updates $f(t_1; : : :; t_n) := t$. The states of ASMs are arbitrary structures, i.e. domains with predicates and functions defined on them.

DASM generalize runs from sequences of moves of a basic ASM to partial orders of moves of multiple agents, each executing a basic ASM.

So ASM is usually a state represented by algebra and transition rule. The algebra consists of superuniverse divided into universes and a number of dynamic functions. The interpretation of the transition rule generally causes an update of the state by modifying the dynamic function or by inserting a new element into the universe.

The DASM for formalizing of BPEL may be defined. One such attempt is made in [Farahbod, 2004] as:

A DASM *M* has a finite set AGENT of autonomously operating *agents*.

- The set of agents changes dynamically over runs of *M*

- The behavior of an agent *a* in a given state *S* of *M* is defined by its program $program_S(a)$
- To introduce a new agent *a* in state *S*, a valid program has to be assigned to $program_S(a)$; to terminate *a*, $program_S(a)$ is reset to the distinguished value *undef*
- In any state *S* reachable from an initial state of *M*, the set of agents is well defined as AGENTS ≡ $\{x \in S : program_S(x) \neq undef\}$.

Asynchronous (distributed) ASMs generalize sequences of transitions of basic ASM to partial orders of transitions of multiple agents, each executing a basic ASM [Börger, 1997].

Modeling service contract as an abstract machine needs to define its dynamics as functions and its static as state variables. The idea is that the execution of BPEL produces a computation which is a sequence of states. Each next state in the sequence is the result of applying in parallel the updates of exactly one *rule* to the current state. The rule that is chosen to be executed is a rule whose guard is true in the current state. If there is more than one rule whose guard evaluates to true in the current state, one of them is chosen non-deterministically. If no guard is true, then the computation halts.

The structure of BPEL Abstract Machine consists of different types of DASM agents representing process instances and Activity agents - created dynamically by process agents for executing BPEL structured activities.

Each BPEL pattern {<on message> operation; logic of transition; final operation} corresponds to *transition.*

The full BPEL abstract machine is a DASM that has components, each of them is again a DASM. The [Farahbod, 2004] formalization of the key semantic aspects of BPEL in terms of a hierarchically defined BPEL Abstract Machine can be further developed for the aims of mapping such formalization to semantic services description using WSMO framework.

## WSMO

The Web Services Modeling Ontology framework - WSMO [WSMO Specification, 2005] considers the semantic description of Web Services as including Capability (functional) and Interfaces (usage).

WSMO Web Service Interfaces are concerned with service consumption and interaction. Service Interface Description uses Ontologies as data model. This means that all data elements interchanged are ontology instances and service interface represents evolving ontology. Sub-concepts of Service Interface are Choreography and Orchestration. In WSMO service interface description represents the dynamics of information interchange during service consumption and interaction and supports ontologies as the underlying data model.

Service Interface Description Model in WSMO framework consists of:

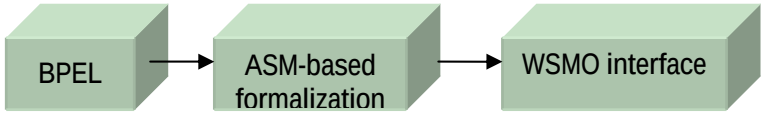
- Vocabulary Ω, that is ontology schema(s) used in service interface description and usage for information interchange: *in, out, shared, controlled.*
- States ω(Ω) that are a stable status in the information space, defined by attribute values of ontology instances;
- Guarded Transition GT(ω) representing state transition with general structure: *if (condition) then (action)* but different for Choreography and Orchestration.

The conceptual model of WSMO orchestrations and WSMO Choreography [Roman et al., 2005] may be described with ASM-based formal semantics.

So ASM is a conceptual base for describing of the formal semantics of non-semantic web services descriptions and the semantic ones. But there are two different layers of abstraction in BPEL-based ASM formalization and WSMO description of SWS interface. To construct a bridge between these representations it is proposed to provide Case-based mapping from BPEL to WSMO through ASM paradigm. As WSMO is still developing the proposed approach is the conceptual one. The proposed tool for semi-automatically mapping of BPEL functional semantic to WSMO description is a part of INFRAWEBS Case-Based Designer intended to semiautomatic conversion of non-semantic Web services to Semantic Web Services.

## BPEL-based Editor

The approach consists in constructing a bridge between two different levels of abstractions in the two ASM-based formalizations (Fig. 1). We have to find relation between BPEL and WSMO service model description. To develop the BPEL-based editor first it is necessary to formalize the semantics in imported BPEL-file.




Because of no formal definition of the semantics is provided by the BPEL so we need to explore the formal meaning of activities in BPEL using ASM paradigm. The semantics of BPEL describe how the language provides interactions. Modelling these semantics requires mapping between activities. An approach to mapping of BPEL to ASM is given in [Farahbod et al, 2004], but we need to enrich it by annotating it with ontologies.

The BPEL is a directed graph. N transitions from state $S_i$ are mapped. A link connects one source activity to one target activity. An activity can have multiple incoming and outgoing links that can be guarded. On the basis on the graph the mapping transitions, mapping states and connecting state skeletons can be made. Each activity is mapped to different kinds of agents in DASM formalization.

To make a matching to WSMO service interface let's consider:

*Data-flow aspect* - the concrete syntax of activity is reviewed; then a semantic mapping for the abstract syntax representation of this fragment of the activity is defined. This needs to include the mapping for basic control flow.

*Control-flow aspect* - The abstract definition of a web service (via WSDL as a port type with an operation) can be mapped to an activity since it comprises a self-contained functionality. The <*partnerLinkType*> definitions describe relationships between different activities without detailing the type of dependency (task, resource, goal).

*Data Handling* - variables hold message data and state information.

The proposed Functionality of the editor is as follows:

- load BPEL file,
- parse BPEL file,
- generate an ASM for each of the component processes, the component processes may be viewed either graphically or textually,
- load ontologies,
- matching/annotation,
- results of matching/annotation,
- export WSMO service interface (formal behaviour model).

The editor is been realizing as a plug-in to Eclipse (Fig. 2).

The following panes are proposed:

- *BPEL XML Tree View*
- *BPEL Process Activities List*
- *WSDL Service Description List*
- *ASM formalization*
- *Mapper/annotator*

At present we have modelled and formalized the web service compositions constructed in the BPEL standard specification. Further work is required with respect to transactional modelling. We are also working in decomposition based upon a resource model of how BPEL processes are composed to distributed requirements.

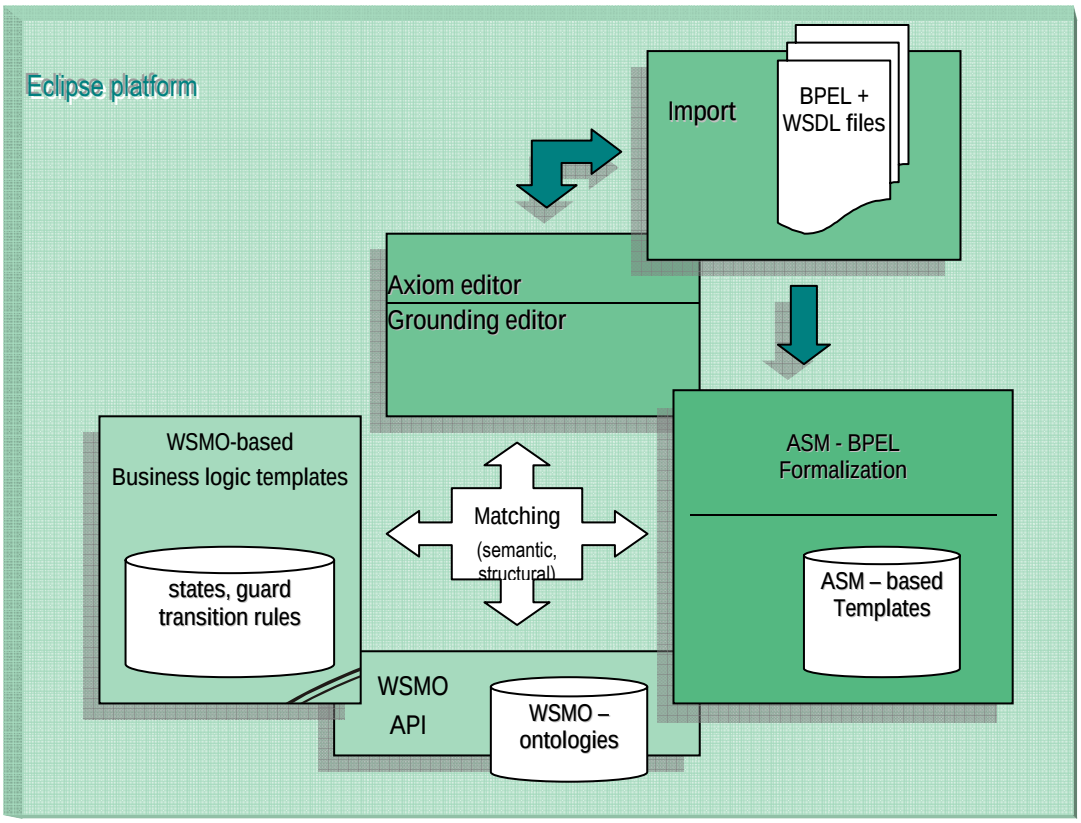



Fig.2

## Conclusion

BPEL provides the model and XML-based grammar that define the interactions between a process and its partners using Web Services interfaces. BPEL also defines the states and logic of coordination between these interactions and systematic ways of dealing with exceptional conditions. But BPEL itself cannot explicitly describe the content and meaning of a Web service/process. Formalizing BPEL by ASM aims to get strict meaning of the functional semantics of the BPEL-based Web service and to migrate to WSMO description of semantic Web service interface. By using ASM the semantics of BPEL may be defined by the work of an abstract interpreter whose behavior is expressed in terms of transition rules. The states of the interpreter are represented by a number of dynamic functions. The semantics of the BPEL components is describing by a corresponding transition rule which indicates in what way a dynamic function should be updated in the process of execution of its component. Mapping to WSMO service interface is proposed by matching/annotation a state with ontology, and guarded transition by transition rules that express changes of states from activities in BPEL-file with ontologies.

## Acknowledgement

This work is carried out under EU Project INFRAWEBS - IST FP62003/IST/2.3.2.3 Research Project No. 511723.

## Bibliography

[INFRAWEBS, 2004] http://www.fh-bochum.de/infrawebs/

[W3C, 2004] Web Services Architecture, W3C Working Group Note 11, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ February 2004.

[Cardoso, 2005] J. Cardoso and A. Sheth, Introduction to Semantic Web Services and Web Process Composition, in Semantic Web Process: powering next generation of processes with Semantics and Web Services, Lecture Notes in Computer Science, Springer, 2005

[Patil et al, 2004] A. Patil, S. Oundhakar, A. Sheth, K. Verma, METEOR-S Web Service Annotation Framework, WWW 2004, May 17–22, 2004, New York, New York, USA.

[Sivashanmugam et al. 2003] K. Sivashanmugam, J. A. Miller, A. P. Sheth, K. Verma, Framework for Semantic Web Process Composition, – Technical Report 03-008, LSDIS Lab, Dept of Computer Science, UGA. June 2003.

[Nern et al, 2004] H Joachim Nern, G. Agre, T. Atanasova, J. Saarela. System Framework for Generating Open Development Platforms for Web-Service Applications Using Semantic Web Technologies, Distributed Decision Support Units and Multi-Agent-Systems - INFRAWEBS II. In: WSEAS TRANS. on INFORMATION SCIENCE and APPLICATIONS, 1, Vol. 1, 286-291, 2004.

[Atanasova et al, 2005] Tatiana Atanasova, Gennady Agre, H Joachim Nern, INFRAWEBS Semantic Web Unit for design and composition of Semantic Web Services INFRAWEBS approach, In: Proc. 1st Workshop for "Semantic Web Applications" at the EUROMEDIA 2005, IRIT, Université Paul Sabatier, Toulouse, France, April 2005.

[Farahbod et al, 2004] Roozbeh Farahbod, Uwe Glässer and Mona Vajihollahi, Specification and Validation of the Business Process Execution Language for Web Services, Software Engineering Lab School of Computing Science, Simon Fraser University, Canada, Technical Report SFU-CMPT-TR-2003-06, Feb 2004

[Börger, 1999] Egon Börger, High Level System Design and Analysis using Abstract State Machines. Current Trends in Applied Formal Methods (FM-Trends 98). Springer LNCS 1641, 1999.

[Börger, 1997] Egon Börger, Integrating ASMs into the Software Development Life Cycle, Journal of Universal Computer Science, vol. 3, no., 603-665, Springer Pub. Co., 1997

[Gurevich, 1995] E. Börger (ed.): Yuri Gurevich: Evolving Algebras 1993: Lipari Guide, Specification and Validation Methods, Oxford University Press, 1995, 9-36.

[Mandell, 2003] Mandell, D.J., McIlraith, S.A.: Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In: Proc. of the 2nd Int. Semantic Web Conf. (ISWC), 2003.

[WSMO Specification, 2005] Roman, D.; Lausen, H.; Keller, U. (eds.): Web Service Modeling Ontology, WSMO Working Draft D2, final version 1.2, 13 April 2005.

[WSMO Choreography and Orchestration] Roman, D.; Scicluna, J., Feier, C. (eds.): Ontology-based Choreography and Orchestration of WSMO Services, WSMO Working Draft D14, 01 March 2005.

[Berners-Lee et al. 2001] Tim Berners-Lee, James Hendler, and Ora Lassila, “The Semantic Web”. Scientific American, 284(5):34-43, 2001.

[Zamulin, 2003] A. Zamulin, A state-based semantics of Pascal-like language, IIS, Novosibirsk, 2003

[KM-GOV, 2005] Semantic Interoperability Community of Practice –Introducing Semantic Technologies and the Vision of the Semantic Web, White Paper Series Module, KM-GOV, 2005

[Patil and Wagh, 2002] N. Patil & S. Wagh, Automating Business Process & SOA Testing using Optimyz WebServiceTester™, 2002.

## Author's Information

Tatiana Atanasova – Institute of Information Technologies, Acad. G. Bonchev 2, 1113 Sofia, Bulgaria, e-mail: atanasova@iinf.bas.bg