# Semantic Orchestration of Services in E-Government

Marek Skokan[1], Peter Bednár[2]

[1]Faculty of Economics, Nemcovej 32, 040 01 Kosice, Slovakia
[2]Centre for Information Technologies, Letna 9, 040 01 Kosice, Slovakia
{Marek.Skokan,Peter.Bednar}@tuke.sk

**Abstract.** Services (e.g. governmental) can participate in a workflow, where the order of their invocation affects the overall operation. The observable behaviour of the service from the client's point of view is known as service choreography whereas how the overall functionality of the service is achieved in terms of co-operation with the other services (a full execution mechanism) is known as orchestration. There is just one available implementation of the execution mechanism based on abstract state machines for WSMO specification. In principle, this implementation is not suitable in case there is a need to navigate user during the process execution. An alternative solution based on Cashew workflow language was proposed to overcome this problem.

## 1 Introduction

The work presented in this paper was done within the context of the Access-eGov project. Access-eGov is EC funded 6th framework project (FP6-2004-27020), which aims at increasing the accessibility of public administration services for citizens and business users by supporting the interoperability among existing electronic and "traditional" government services.

For citizens and business users, Access-eGov will provide two basic categories of services. Firstly, Access-eGov will identify – depending on the needs and context situation (location, etc.) of the user – traditional and/or e-Government services (if available) relevant to the given life event (of the given citizen) or business episode (in case of businesses). Secondly, once the relevant services have been identified, Access-eGov will generate a "scenario" consisting of elementary government services. In most cases these scenarios will be probably of a "hybrid" nature – i.e. a combination of elementary traditional and e-services – which will lead to a requested outcome (e.g. to get a building permit, register a new company, etc.).

According to analysis of the expected functionalities, WSMO semantic framework for Web Services was selected as the main technology to implement the Access-eGov technological solution.

The paper is organised as following: an outline of the WSMO semantic framework is presented in Section 2, the Access-eGov model for orchestration and choreography is defined in Section 3. In Section 4 a brief description of one scenario from the Access-eGov pilot applications is presented. Then some examples of the orchestration

© Václav Snášel (Ed.): Znalosti 2008, pp. 215–223, ISBN 978-80-227-2827-0.
FIIT STU Bratislava, Ústav informatiky a softvérového inžinierstva, 2008.

interfaces for this scenario are presented in Section 5 to show how the hybrid scenarios are modelled with the proposed model.

## 2 Web Service Modelling Ontology

The Web Service Modeling Ontology (WSMO) is a conceptual model for describing semantic Web Services. WSMO consists of four major components: ontologies, goals, Web Services and mediators.

Ontologies provide the formal semantics to the information used by all other components. WSMO specifies the following constituents as part of the description of ontology: concepts, relations, functions, axioms, and instances of concepts and relations, as well as non-functional properties, imported ontologies, and used mediators. The latter allows the interconnection of different ontologies by using mediators that solve terminology mismatches.

A goal specifies objectives that a client might have when consulting a Web Service, i.e. functionalities that a Web Service should provide from the user perspective. In WSMO a goal is characterized by a set of non-functional properties, imported ontologies, used mediators, the requested capability and the requested interface (see the Web Services description).

A Web Service description in WSMO consists of five sub-components: non-functional properties, imported ontologies, used mediators, a capability and interfaces. The capability of a Web Service defines its functionality in terms of preconditions, postconditions, assumptions and effects. A capability (therefore a Web Service) may be linked to certain goals that are solved by the Web Service via mediators. Preconditions, assumptions, postconditions and effects are expressed through a set of axioms and a set of shared all-quantified variables. The service interfaces are described in the following chapter.

Mediators describe elements that aim to overcome structural, semantic or conceptual mismatches that appear between the different components that build up a WSMO description. Currently the specification covers four different types of mediators:

- *OOMediators* - import the target ontology into the source ontology by resolving all the representation mismatches between the source and the target;
- *GGMediators* - connect goals that are in a relation of refinement allowing the definition of sub-goal hierarchies and resolve mismatches between those;
- *WGMediators* - links a goal to a Web Service via its choreography interface meaning that the Web Service fulfills the goal; or links a Web Service to a goal via its orchestration interface meaning that the Web Service needs this goal to be resolved in order to fulfill the functionality;
- *WWMediators* - connect several Web Services for collaboration.

### 2.1 WSMO Choreography and Orchestration

The interface of a Web Service provides further information on how the functionality of the Web Service is achieved. It describes the behavior of the service for the client's point of view (service choreography) and how the overall functionality of the service is achieved in terms of cooperation with the other services (service orchestration).

A choreography description is semantically based on the Abstract State Machines (ASMs) [3] and consists of the states represented by ontology, and the if-then rules that specify (guarded) transitions between states. The ontology that represents the states provides the vocabulary of the transition rules ant contains the set of instances that change their values from one state to the other. The concepts of an ontology used for representing a state may have specified the grounding mechanism, which binds service description to the concrete message specification (e.g. WSDL).

For the Orchestration interfaces, it is planned by the authors to proceed as follows. The language will be based (note, that it is envisioned only, and the specification is not finished yet) on the same ASMs model as Choreography interfaces which - in order to link to externally called services or (sub)goals that the service needs to invoke to fulfil its capability - needs to be extended as follows:

- Goals and Services can be used in place of rules, with the intuitive meaning that the respective goal/service is executed in parallel to other rules in the orchestration
- The state signature defined in the choreography can be reused, i.e. external inputs and outputs of the service and the state of the choreography can be dereferenced also in the orchestration
- Additionally the state signature for the orchestration interface can extend the state signature of the choreography interface, with additional in/out/shared/controlled concepts which need to be tied to the used services and rules by mediators
- Respective WW or WG mediators need to be in place to map the in and out concepts defined in the orchestration to the respective out and in concepts of the choreography interfaces in the used services and goals, i.e. these mediators state which output concepts are equivalent to which input of the called service/goal and vice versa

## 3 Access-eGov orchestration and choreography model

Besides of the general requirements on the Choreography and Orchestration models for the semantic Web Services, the following requirements were identified as the critical for the Access-eGov project. The choreography and orchestration process model will be used:

a) to guide citizens to achieve specific goals, and
b) to coordinate activities performed by all actors - citizens, traditional public administration services and web services.

We have found that the current proposal of WSMO specification described in the previous chapter is not suitable for these objectives, because models based on state

machines are not structured in the way suitable for the interaction with the human actors. For these reasons, we have decided to design and implement workflow based extension to the WSMO specification. Besides of the previous objectives, the following facilities were identified as useful for a process model to provide support for modelling orchestrated scenarios:

- compatibility with the standard process modelling notation (i.e. BPMN) in order to visualize scenarios to the users and to use standard tools for modelling,
- compatibility with the proposed standard workflow modelling languages (i.e. WS- BPEL)

Our model is based on the workflow CASheW-s model [5] originally proposed for the OWL-S specification with the dataflow and WSMO mediation extensions. The general structure of the model is depicted on the following diagram.

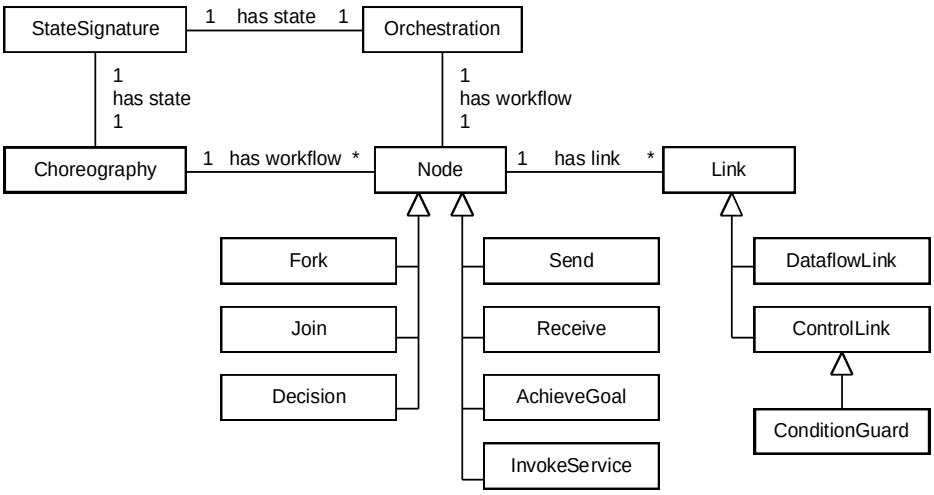


**Fig. 1.** The structure of the Access-eGov orchestration and choreography process model.

Access-eGov model reuses the state signature from the WSMO specification and replaces the ASMs transition rules with the workflow constructs. Shared ontology state signature allows reusing grounding of the input and output concepts to the communication protocols via WSDL. Workflow model consists of activity nodes. Node can be an atomic node (Send, Receive, AchieveGoal and InvokeService), or control node (Decision, Fork and Join).

Nodes are connected with control links, which represent control flow. Each node can have at least one arriving control link, only the root node which represent starting activity of the workflow doesn't have any arriving links. Number of outgoing links depends on the type of the node. Fork and decision nodes have many outgoing links; other nodes can have only one outgoing link, which determine subsequent activity in

the workflow. End nodes do not have any outgoing links (it is possible to have many end nodes, depending on branches in workflow).

Branching is defined with the Decision nodes. Cycles (while or do-while) are created with the decision node and backward control links which points to activity within the cycle (i.e. every activity can have many arriving control links). Decision node represents internal decision, which is evaluated by the execution environment (service requester cannot directly select branch (i.e. deferred decision pattern), but decision can be depended on the data received from the user). Decision node can contain many outgoing control links, which are guarded by logical conditions. It can contain one unguarded link, which represent *else* branch. All conditions should be disjoint, i.e. only one condition can be evaluated to true and subsequent node should be selected deterministically.

Workflow can have parallel threads created with Fork/Join nodes. Fork node has at least two leaving control links and each subsequent activity is executed in the parallel thread. Join synchronizes parallel threads arriving to the node (i.e. it waits until all activities from arriving links will be finished).

**Data pins and dataflow**

Activity nodes can require some data to be performed or can provide data required for other activities. Each node has assigned the set of data pins. The values of the data pin can be read or write by the execution environment or service requester (i.e. user) according to the pin mode. The semantic of pin modes is defined as:

- *Input* – data has to be provided by the service requester, i.e. user
- *Output* – (for choreography only) data is provided by the electronic service or in the case of traditional services, data is requested from the user
- *Inferred* – data is inferred from values of Input/Output or Controlled pins
- *Controlled* - data is assigned by the execution environment (i.e. copied with dataflow link from other nodes)

**Reference syntax**

This chapter defines reference syntax for orchestration and choreography process model. To keep the description concise, the elements that are already defined in [4] are not shown in the syntax. The grammar is specified using a dialect of Extended BNF which can be used directly in the SableCC compiler. Terminals are quoted; non terminals are underlined and refer to tokens and productions. Alternatives are separated using vertical bar '|', and are labelled with labels enclosed in curly braces. Optional elements are appended with a question mark '?'; elements that may occur zero or more times are appended with an asterisk '*'; and elements that may occur one or more times are appended with a plus '+'.

```
aeg_workflow = t_workflow aeg_node* aeg_controlflow? aeg_dataflow?;
aeg_node =
    {send} t_perform id? t_send log_expr nfp? |
    {receive} t_perform id? t_receive log_expr nfp? |
    {achievegoal} t_perform id? t_achievegoal [goal]:id nfp? |
    {decision} t_perform id? t_decision nfp? |
```

```
    {fork} t_perform id? t_fork nfp? |
    {join} t_join id? t_join nfp?

aeg_controlflow = t_controlflow aeg_control_links;
aeg_dataflow = t_dataflow aeg_dataflow_links;

aeg_control_links = aeg_control_link aeg_control_links?;
aeg_control_link = t_source [source]:id t_target [target]:id aeg_guard?;
aeg_guard = t_guard log_expr;

aeg_dataflow_links = aeg_dataflow_link aeg_dataflow_links?;
aeg_dataflow_link =
    t_source [source]:aeg_pin_reference
    t_target [target]:aeg_pin_reference;

aeg_pin_reference = id lbrace variable rbrace;
```

# 4 Life event description

Three life events constitute a base for pilot applications in Access-eGov. One of these life events is "Establish an enterprise". This pilot application will be deployed in City Hall of Gliwice in Poland. The overall procedure is described in detail in [1]. Services being used to establish an enterprise in Poland consist of five main tasks:

- Registration in the City Hall (local government).
- Registration in the Statistical Office
- Obtaining bank account number (this is performed within the next task)
- Registration in the Tax Office
- Registration in the Social Insurance Agency

## 4.1 Sub-goal Registration in Statistical Office

Registration in statistical office is the second step within this life event. This step is chosen because of explanation purposes. In that stage of process the user applies for receiving statistical number REGON. In both cases (private person and civil law partnership) the user fills in and applies RG-1 form and dependently on number of business activity types he additionally applies GR-RD form.

There is no registration fee in statistical office, thus after applying RG-1 form and other required documents the user receives REGON number within 7 days from application. When applying the form personally in the statistical office REGON number is assigned immediately during the visit in authority.

## 4.2 Expectations from the system

The following facts have to be taken into account:

1. The tasks within this scenario take place in different offices and at different times.
2. The user has to perform these tasks in the right order.

3. The specific user's situation determines actions within each task (parametrisation).
4. The main goal is to deliver complete information related to the services and to enable citizens to establish their enterprise via Internet (if possible).

The Access-eGov platform identifies the parameters of the eventual company from additional questions answered by the user and indicates next actions. Note, that how these questions are managed is not described in this paper.

## 4.3 Example

Formal description of the pilot Access-eGov application processes has been done. The following example is the formal description of the orchestration interface of the high level process that is connected with the life event "Establish an enterprise" in Poland.

```
interface EstablishEnterpriseLifeEventInterface
orchestration
   workflow
     perform n1_1 receive ?x memberOf Q1.
     perform n1_2 achieveGoal RegisterInLocalGovernmentGoal
     perform n1_3 achieveGoal RegisterInStatisticalOfficeGoal
     perform n1_4 achieveGoal RegisterInTaxOfficeGoal
     perform n1_5 achieveGoal RegisterInSocialInsuranceAgencyGoal

   controlFlow
     source n1_1 target n1_2
     source n1_2 target n1_3
     source n1_3 target n1_4
     source n1_4 target n1_5

   dataFlow /**/
```

By interpreting this formal description, first the batch of answers to the pre-defined questions (?x has to be instance of concept Q1) needs to be received from the user by the process. Then other sub-goals need to be achieved in the right order. As it can be seen one of these goals is *RegisterInStatisticalOfficeGoal*. Transitions in the *controlFlow* part express that all nodes are executed in a sequence. The *dataFlow* part is empty in this case, since there is no direct use of some variable between these workflow nodes.

The example bellow describes the choreography interface of Registration in the Statistical Office (the second step in the overall process).

```
interface RegisterInStatisticalOfficeInterface
choreography
   workflow
        perform n2_1 receive ?x memberOf Q3.
        perform n2_3 receive ?x memberOf FormRG_1.
        perform n2_4 decision
        perform n2_5 receive ?x memberOf FormRG_RD.
        perform n2_6 send ?x memberOf REGON.
```

```
controlFlow
     source n2_1 target n2_3
     source n2_3 target n2_4

     source n2_4 target n2_5 guard ?x[q1 hasValue moreThanThree].
     source n2_5 target n2_6
     source n2_4 target n2_6

dataFlow
     source n2_1{?x} target n2_4{?x}
```

By interpreting this formal description, first the batch of answers to the pre-defined questions (?x has to be instance of concept Q3) needs to be received from the user by this process (number of business activity types). Then the process needs to receive certain form (instance of concept FormRG_1). The decision node means that some of the following nodes are optional (only node n2_6 in this case - see the *controlFlow* part). Next, the process (might) need(s) to receive another form again (instance of concept FormRG_RD). Finally, the process sends the REGON number. The *controlFlow* part contains one conditional transition. The transition between the *decision* workflow node and the following *receive* workflow node depends on the answer to the question about the number of business activity types (the value of attribute q1 from concept Q3 is checked). The process can thus reach the final node right after this *decision* node or from the last *receive* node depending on the decision result. The *dataFlow* part specifies that the variable from the first node (n2_1 – the batch of questions) is equivalent with the variable from the *decision* node (n2_4).

## 5 Conclusion

The paper described an alternative solution for orchestration of WSMO services. The definitions of the WSMO choreography and orchestration interfaces are presented. The reason why these definitions are not sufficient for the purposes of the Access-eGov system is explained and the Access-eGov (alternative) orchestration interface is presented. The process of pilot application "Establish an enterprise" which will be running in City Hall of Gliwice in Poland is described in the textual form as well as defined in the formalism of the Access-eGov orchestration interface. The textual description of this process corresponds quite directly to the Access-eGov formal description of interfaces. Therefore it is much easier to present this process to the user than it is in case of WSMO choreography and orchestration.

## 6 Acknowledgements

This paper is supported by the project FP6-2004-27020 Access-eGov (Access to e-Government Services Employing Semantic Technologies), by project Nr. 1/4074/07 Methods for annotation, search, creation, and accessing knowledge employing meta-

data for semantic description of knowledge and by the project SEMCO-WS Semantic composition of Web and Grid Services APVV-0391-06.

## References

1. Klischewski, R. et al.: User Requirement Analysis & Development / Test Recommendations. Access-eGov Project (FP6-2004-27020) Report, 2006.
2. Roman, D., Scicluna, J., Fensel, D., Polleres, A., de Bruijn, J.: D14: Ontology-based choreography of WSMO services. WSMO working draft, DERI (2006) Available online at http://www.wsmo.org/TR/d14/v0.4/.
3. Yuri Gurevich, "Evolving Algebras 1993: Lipari Guide", Specification and Validation Methods, ed. E. Börger, Oxford University Press, 1995, 9-36.
4. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, D. Fensel: The Web Service Modeling Language WSML. WSML Final Draft v0.2, 2005. Available from http://www.wsmo.org/TR/d16/d16.1/v0.2/.
5. B. Norton, S. Foster, and A. Hughes.:A compositional operational semantics for OWL-S. In Proc. 2nd Intl. Workshop on Web Services and Formal Methods (WS-FM 2005), September 2005.