Workflow Description Language
Research Results

Poznańskie Centrum Superkomputerowo-Sieciowe

Poznań, 2011
Table of Content

1 Selection process and criteria ................................................................. 4
2 Workflows requirements ........................................................................ 5
  2.1 Introduction ......................................................................................... 5
  2.2 Overall description ............................................................................ 6
3 BPEL ....................................................................................................... 7
  3.1 General .............................................................................................. 7
  3.2 Language strength ............................................................................ 7
  3.3 Popularity ......................................................................................... 8
  3.4 Graphical representation .................................................................. 8
4 DPML ...................................................................................................... 9
  4.1 General .............................................................................................. 9
  4.2 Language strength ............................................................................ 9
  4.3 Popularity .......................................................................................... 10
5 GSFL ...................................................................................................... 11
  5.1 Language strength ............................................................................ 11
  5.2 Popularity .......................................................................................... 12
6 GWorkflowDL ........................................................................................ 12
  6.1 General .............................................................................................. 12
  6.2 Language strength ............................................................................ 13
  6.3 Popularity .......................................................................................... 13
  6.4 Graphical representation ................................................................ 13
  6.5 Workflow Engines ............................................................................ 14
  6.6 Workflow APIs/libraries ................................................................. 14
7 MoML ...................................................................................................... 15
  7.1 General .............................................................................................. 15
  7.2 Language strength ............................................................................ 15
  7.3 Popularity .......................................................................................... 16
  7.4 Graphical representation ................................................................ 17
  7.5 Workflow Engines ............................................................................ 17
  7.6 Workflow APIs/Libraries ................................................................. 19
8 OWL-S .................................................................................................... 20
  8.1 General .............................................................................................. 20
  8.2 Language strength ............................................................................ 20
1 Selection process and criteria

A proper workflow description language has to be chosen to ensure extended functionality and performance of Workflow Manager. A dedicated solution was designed for needs of EXPReS project. It was based on JSON – a light text format, formal subset of JavaScript language. Unfortunately maintaining this solution is no longer cost-effective, especially where it comes to addition of functionalities that were not foreseen at the beginning of implementation. Standards or de facto standards are another significant aspect applying to workflows domain. Considering problem in all its bearings a decision to introduce new language was taken. An elaborated study was conducted to choose solution best suiting NEXPReS requirements.

Before the research was started the possible solutions domain was narrowed down to three workflow languages group. For each group a representative set of languages was chosen and further studies conducted for them. The identified groups are:

- BPEL (BPEL itself and other BPEL-based languages)
- WebServices (workflow basing on WS idea, using WS mechanisms or format of description)
- Grid/Scientific (languages developed for Grid or scientific usage and communities)

Languages chosen for further research, grouped by types defined above are presented in Table 1.

<table>
<thead>
<tr>
<th>Group</th>
<th>BPEL</th>
<th>WS</th>
<th>Grid/Scientific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>BPEL + BPMN</td>
<td>OWL-S</td>
<td>DPML</td>
</tr>
<tr>
<td></td>
<td>SimPEL</td>
<td>XPDL</td>
<td>GSFL</td>
</tr>
<tr>
<td></td>
<td>BPELScript</td>
<td>xWFL</td>
<td>GJobDL</td>
</tr>
<tr>
<td></td>
<td>XPDL</td>
<td>WSCL</td>
<td>GWorkflowDL</td>
</tr>
<tr>
<td></td>
<td>YAWL</td>
<td>WSCI</td>
<td>MoML (Keppler)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSFL</td>
<td>SCUFL (Taverna)</td>
</tr>
</tbody>
</table>

**Table 1 Workflow languages’ groups**

In process of deciding whether given workflow description language is interesting for NEXPReS project the following preliminary criteria were applied:

- Is the project active/in development?
- Is there technical support available? (e.g. in form of Internet forum or mailing list)
- What is the language licence?

If a language has passed the preliminary criteria test it was then described and ranked in the following fields:

- Language strength (extensibility, loops and conditions support, data types, subworkflows)
- Popularity (usage in other projects, available technical support, automatic transformation to other workflow description languages)
- Libraries (parsers, graphical representation of workflow)
The outcome of research is presented in table 2 and detailed results and description of all languages studied are contained in the following sections of this document.

<table>
<thead>
<tr>
<th></th>
<th>Extensability</th>
<th>Language strength</th>
<th>Popularity</th>
<th>Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL</td>
<td>-</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>YAWL</td>
<td>-</td>
<td>+++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>XPDL</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GSFL</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GLWorkflowDL</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>MoML</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Scufl</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 2. Chosen workflow languages

As a result of described process of research MoML language was chosen as a new language for workflow description in NEXPReS. Main highlights of this solution are its extensibility, actor-based architecture and being a de facto standard for scientific communities of different domains. It is also worth to mention that vivid technical support groups exists available as a mailing lists. MoML is language used in Kepler project – very popular scientific project developing an open source libraries for parsing and graphical presentation of MoML structures.

2 Workflows requirements

[according to IEEE 830]

2.1 Introduction

2.1.1 Product Overview

Application for remote access to environmental tools and data acquired by those tools

2.1.2 Purpose

To provide users with possibility of remote usage of instruments, scheduling their work and accessing data they produce/acquire.

2.1.3 Scope

- Instruments configuration
- Adding new instruments
- Performing remote operations on instruments
- Definition of new workflows (involving instruments)
- Scheduling of workflow
- Submission of workflow
- Monitoring of workflow
- Access to data gathered/produced by workflow
2.2 Overall description

2.2.1 Product Perspective

2.2.2 Product Functions

- Log-in to system
- Role-based access to instruments and functionalities
- Configuration of instruments
- Adding new instruments
- Remote access to instruments – commands invocation
  - information of instrument state
  - information whether a command given can be executed in current state
- real-time access to instruments
  - data streaming
- defining of workflows (involving instruments)
  - graphical drag’n’drop editor
  - graphical controls to configure each instrument’s action
- scheduling of workflow
  - calendar access
  - resources reservation
- workflow submission
- workflow saving
- exporting/importing of workflow
- predefined workflows for further customisation
- monitoring of running workflow
- access to data acquired/produced by instruments

2.2.3 User Characteristics

- Advanced user
  - administrator – tech-savvy, he/she can add and configure new instruments, create, submit and monitor workflows as well as browse data produced by those workflows
  - scientist – he/she can create, submit and monitor workflows as well as browse data produced by those workflows

2.2.4 User Interfaces

- Graphical workflows editor
- Graphical data browser
- Graphical controls assisting in adding of new instruments
- Graphical controls for configuration of instruments
- Graphical monitoring of running workflow

2.2.5 Software Interfaces

- Web-based application
2.2.6 Communications Protocols

- LAN/WLAN - web services

3 BPEL

BPEL (Business Process Execution Language for Web Services) developed by IBM, BEA Systems, Microsoft and other companies, XML-based language for business processes definition in web services. BPEL4WS is the official acronym, but it is known as a BPEL for short.

3.1 General

3.1.1 General characteristic
- **XML-based**
- Business processes definition in web services
- It allows for definition of business processes and interaction protocols for completing different tasks
- It was developed mainly to extend WebServices model with business transaction support
- Different programing languages versions of BPEL exists

3.1.2 Basic language terms and structures
- Language type: Orchestration. *An orchestration is from one actor's point of view, where choreography looks at a global system and all the actors, and their interactions, without looking at any single actor's internals*

3.1.3 Historical context
- IBM - 2001 - WSFL
- Microsoft - 2001 - XLANG
- Merge of 2 languages: BPEL4WS - 2002
- In 2003 some companies (including IBM and Microsoft) send BPEL4WS 1.1 for standardisation process (OASIS)

3.2 Language strength

3.2.1 Node types

- *process*
  - data declaration
  - definition of actors relations
  - declaration of activity for execution
- *scope* – defines the visibility, allows for definitione of process’ logic. Process will not be visible outside of scope.
- *partnerLinks* – definitione of communication means with businesss partners. It is a two-way interaction: the process calls its partner, then the partner calls its process.
- variables – see below
- faultHandlers
aktywności – two types: structure and basic (receive, invoke, reply)

3.2.2 Data types
- variables – keep temporary data of process or results of partners communication.
- Types of data:
  - WSDL message types
  - XML schema simple types
  - XML schema complex types or XML schema elements

3.2.3 Extensibility
- yes

3.2.4 Conditional structures
- yes, if structure
- yes, while structure
- the same as in common programming languages

3.2.5 Loops
- yes, couple of types, like in common programming languages

3.2.6 Other
- Parallel processing
- Data manipulation
- Exception handling mechanism

3.3 Popularity

3.3.1 Is it standardised?
- It is a standard
- Versions available:
  - BPEL4WS 1.0 year 2002
  - BPEL4WS 1.1 year 2003
  - WS-BPEL 2.0 year 2007 (last update)

3.3.2 Interoperability
- BPEL is an XML-based language, therefore it can be easily transformed to other languages. There are different versions and extensions of BPEL available.

3.4 Graphical representation
3.4.1 Bpel for Eclipse

1. Programming language
   - different programming languages support, special support for Java.
2. Is the project alive?
   - yes, it is still in development. The last update action was taken in May 2009
3. Graphical presentation
   - graph representing business processes. User’s icons make no sense as the graph is operating on language terms/elements: invoke, sequence, etc.

3.4.2 Workflow Engines

- Many of engines available:

3.4.3 Workflow APIs/Libraries

- XML- and WSDL- based language, therefore a lot of editing tools and libraries are available

4 DPML

4.1 General

4.1.1 General characteristic

- XML
  - Discovery Process Markup Language. Discovery Net was one of the first scientific workflow systems, now replaced by Kepler and Taverna.

4.1.2 Historical context

- Aimed at scientific workflows where task can be defined as deterministic functions. It is now a historical language.

4.2 Language strength

4.2.1 Node types

- Nodes are mapped to tasks

4.2.2 Data types

- A key feature of the design of the system has been its support for data management within the workflow engine itself. This is an important feature since scientific experiments typically generate and use large amounts of heterogeneous and distributed data sets. The system was thus designed to support persistence and caching of intermediate data products and also to support scalable workflow execution over potentially large data sets using remote compute resources.
• A second important aspect of the Discovery Net system is based on a typed workflow language and its extensibility to support arbitrary data types defined by the user. Data typing simplifies workflow scientific workflow development, enhances optimization of workflows and enhances error checking for workflow validation. The system included a number of default data types for the purpose of supporting data mining in a variety of scientific applications. These included a Relational model for tabular data, a bioinformatics data model (FASTA) for representing gene sequences and a stand-off markup model for text mining based on the Tipster architecture.

4.2.3 Extensibility
• No data available

4.2.4 Conditional structures
• Conditional structures are available, no time constraints structures are available

4.2.5 Loop (for-each, for-n, while)
• No loops available, the flow has to be acyclic, mapped into XML structure.

4.2.6 Data- or Control-driven?
• A key contribution of the system is its clean separation between the data flow and control flow models of computations within a scientific workflows. This is achieved through the concept of embedding enabling complete data flow fragments to be embedded with a block-structured fragments of control flow constructs. This results both in simpler workflow graphs compared to other scientific workflow systems, e.g. Taverna workbench and the Kepler scientific workflow system and also provides the opportunity of applying formal methods for the analysis of their properties.

4.3 Popularity

4.3.1 Standard
• Not intended to be a standard, it was intended as a proof-of-concept

4.3.2 Community
• No community back-up. The is no current support

4.3.3 Usage in other projects
• Used mostly in EU scientific projects, i.e.:
  1. SIMDAT "http://www.simdat.org/"
  2. The BRIDGE Project "http://www.bridge-grid.eu/"
  3. The ARGUGRID Project "http://www.argugrid.eu/"
  5. InforSense Ltd. "http://www.inforsense.com/"

4.3.4 Interoperability
No data available

4.3.5 Other
The language is no longer in use.
5 GSFL

5.1 Language strength

5.1.1 Node type
- each WebService endpoint (WS) is considered as a workflow node
- technical details:
  - every WebService has to be defined in <ServiceProvider> list.
  - every WebService call is defined in <activityModel>
  - every GSFL may describe multiple workflows (considered as sequence of actions), each such sequence is called <exportedActivity> (described in <compositionModel>)

5.1.2 Data types
- Data types are limited to those that WebService/WSDL are able to handle
- Transferring data between services (Third-party transfer) is supported (OGSA, see more: dataSources, dataSinks)

5.1.3 Extensibility
- Extensibility is limited to WebService specification.

5.1.4 Conditional structures
- not supported directly,
- conditional structures can be simulated by defining sequence of WebService instantiation (lifecycleModel) and it's visibility:
  - session – each WebService call is an isolated instance,
  - application – instances of services are created for each workflow based on <lifecycleModel>, this gives the ability to control sequence of the WebService calls.

5.1.5 Loops (for-each, for-n, while)
- Language has no support for loops
- There are no constraints to have a cycle in WebService sequence calls.

5.1.6 Data- or Control-driven
- control-driven
- it's possible to define autonomous from workflow data flow
- more DAG than Petri-net

5.1.7 Subworkflows support
- available – this is a key feature, WebService as a whole workflow or just a single action,
- subworkflow is visible as a WebService, hence can be used as a single action from another workflow/WebService,
- workflow is just a WebService,

5.1.8 Other
- XML-based
• clear, flexible construction of workflow as a WebService composite
• usage limited to workflows only
• focused on integration with already existing workflows.

5.2 Popularity

5.2.1 Standard
• not a standard
• commonly used in grid projects
• OGSA Support, no activity since 2008,
• closed project – no longer under development (according to Globus site)

5.2.2 Community
• OGSA - working group in OGF – no activity since 2008.

5.2.3 Usage in other projects
• GridWorM – new language GWLang = GSFL + BPEL, project from 2007. - integrates
WS with non-WS jobs, little data available.

5.2.4 Interoperability
• No data available

6 GWorkflowDL
6.1 General

6.1.1 General characteristic
• XML
• High-Level Petrinets (nodes are restricted in number of processed tokens, token can
represent the data handle or the data itself.
• website
• XSD schema v. 2.1 + documentation
• Generic Workflow Description Language (G stands for Grid in some sources)
• workflow description language for the Grid.

6.1.2 Basic language terms and structures
• GWorkflowDL file consists of two parts:
  o general – describes workflow, data flow and control flow
  o middleware specific (as a extension) – describes specific actions to perform
    workflow in this middleware
• Petri-net namespace:
  o place
  o arc
  o transition
  o token
6.1.3 Historical context
- developed in K-WfGrid project

6.2 Language strength

6.2.1 Node types
- Nodes are representing workflow state, hence no need to type them
- General transitions are representing data operations, no need to type them

6.2.2 Data types
- XML based
- URL based (data handlers)

6.2.3 Extensibility
- Supports extensions for different middlewares.

6.2.4 Conditional structures
- Available, both logical and time-based

6.2.5 Loops (for-each, for-n, while)
- No support in language syntax
- Available by workflow construction – see workflow construction patterns

6.2.6 Data- or Control-driven?
- Tokens are representing data, actions (transitions) are triggered by tokens count.

6.2.7 Subworkflows support
- Yes

6.3 Popularity

6.3.1 Usage in other projects
- K-WfGrid
- CppWfMS

6.3.2 Interoperability
- Available conversions in CppWfMS:
  - from JDL
  - from Scufl

6.4 Graphical representation

6.4.1 GWUI - GridWorkflow User Interface
- Language
  - Java (servlets)
- Is the project alive?
  - No documentation, last available version 0.7.1, December 2008.
6.5 Workflow Engines

6.5.1 GWES – Grid Workflow Execution Service

- **Is the project alive?**
  - Last version (2.0.1) from July 2009.
- **Licence**
  - The Fraunhofer FIRST license agreement (link) - open source for non-commercial use
- **Extensibility**
  - No data available
- **Graphical presentation**
  - No data available
- **Other**
  - Graphical UI for GWES
  - No documentation available, no project description available

6.6 Workflow APIs/libraries

6.6.1 GWorkflowDL Toolkit

- **Programming language**
  - java
- **Functionalities**
  - Language modeled in Java
- **Static workflow parsing** – checks if workflow finishes in finite time
• **Is the project active**
  o Last post on discussion board in 2007
  o Last release (2.1) March 2010.

• **Licence**
  o The Fraunhofer FIRST license agreement ([link](#)) - open source for non-commercial use

• **Other**
  o [download](#) - pakiety gworkflowdl-X.X.X-src.tar.gz lub gworkflowdl-X.X.X-src.zip
  o [documentation](#)
  o [javaDoc](#)

### 7 MoML

#### 7.1 General

- **XML**
- Semantic can be easily changed
- Used only to describe connections between objects
- Semantic (which objects can be connected and how) – not defined in MoML. They are specified by a reference to a class object (default to Java language, supported by [PtolemyII](#), implementations in other languages possible). Each node and relation between them is an instance of one of those classes.

- **MoML specification**
- **dtd**
- **MoML as a MIME type**

#### 7.1.1 Basic language terms and structures

- 2 basic MoML elements:
  - class – class declaration, describes workflow semantic
  - model – describes structure

- model consists of:
  - entity – model’s object (in workflow this is a node)
  - relation – they describe type (semantic) of a relation between entities. Together with **link** and **port** they create connection between entities.
  - port – chosen for an entity, 0..n, can be of in, out or in/out type. Part of a connection.
  - link – Connection between port and relation. Together they form a connection between entities.
  - Optional elements used by render (i.e. link break points in graphical representation of a model). Ignoring elements not understandable by model.

- **connection** – connects entities: it consist of at least 1 relation, 2 ports and 2 links.
- **topology** – structure modeled by this particular MoML description.

### 7.2 Language strength
7.2.1 Node types

- not defined in language specification (no semantic there)
- defined by reference to particular class – class instances defined by user are nodes

7.2.2 Data types

- not defined in language specification (no semantic there)
- depends on the relation between objects (class instances – nodes) and relation mappings.
- allows adding in/out ports for a node entity

7.2.3 Extensibility

- very high, cause types are not defined by language

7.2.4 Conditional structures

- no definition in language specification
- can be easily added by external classes for entities or relations

7.2.5 Loops

- no definition in language specification
- can be easily added by external classes for entities or relations

7.2.6 Data or control-driven

- depends on semantic used

7.2.7 Subworkflow support

- yes

7.3 Popularity

7.3.1 Is it standard

- no

7.3.2 Community

- quite active:
  - mailing list for ML: [http://ptolemy.berkeley.edu/ptolemyII/ptIIfaq.htm#mailing%20list](http://ptolemy.berkeley.edu/ptolemyII/ptIIfaq.htm#mailing%20list)
  - active list: [https://lists.eecs.berkeley.edu/sympa/arc/ptolemy-hackers](https://lists.eecs.berkeley.edu/sympa/arc/ptolemy-hackers)
  - Kepler’s mailing list
7.3.3 Usage in other projects

- Very popular – a lot of documents describing integration with various other technologies: portal/web GUI, Flex, Fedora Commons, OGC (sensors)
- **Kepler**
- **Metropolis** – system modeling – last release from 2008
- **COMET** – web portal based on Kepler
- **Vine Toolkit** Workflow Engine – portal, based on Kepler, uses Flex library RaVis
- **Science Pipes** – portal (based on Java Script), analyses and visualizations of biodiversity data
- **project** which exposes Kepler as OGC Web Services
- **description** of sensor integration with Kepler
- **Fedora-Kepler Integration** – integration with Fedora Common, The goal of the project is to create a Kepler workflow that accesses data from a Fedora Repository, performs operations on the data and then stores the result back in a Fedora Repository. Uses Jython
- **Kepler + GUI in Flex**

7.3.4 Interoperability

- no support

7.4 Graphical representation

7.4.1 Ptolemy II: Vergil GUI

- *Programming language*
  - Java - Swing i Diva (last release 2001)
  - documentation
- *Is the project alive*
  - yes – active SVN communities
- *Licence*
  - "fairly liberal UC Berkeley copyright“ - Ptolemy II is free for academic and commercial use, requires to include UC Berkley licence in a product (which informs that UCB does not take any responsibility)
- *Functionalities*
  - Swing application – allows to show Ptolemy II model
- *Extensibility*
  - extensible – mapping of relations and entities to classes (which can be custom classes)
  - allows adding to MoML information about graphical representation (which will be ignored by workflow engine)
  - no plug-in mechanism
- *Graphical representation*
  - as a graph
  - allows adding custom icons, connection types, line breaks etc.

7.5 Workflow Engines
7.5.1 Ptolemy II

- **Is the project alive**
  - available version 8.0 beta
  - [www](http://chess.eecs.berkeley.edu/ptexternal)
  - access:
    2. SVN: [http://chess.eecs.berkeley.edu/ptexternal/](http://chess.eecs.berkeley.edu/ptexternal/)
  - bugtracker: bugs are sent to mailing list [https://lists.eecs.berkeley.edu/sympa/arc/ptolemy-hackers](https://lists.eecs.berkeley.edu/sympa/arc/ptolemy-hackers)

- **Licence**
  - "fairly liberal UC Berkeley copyright" - Ptolemy II is free for academic and commercial use, requires to include UC Berkley licence in a product (which informs that UCB does not take any responsibility)

- **Extensibility**
  - defining custom actors and directors- [description](#)
  - [description](#) how to model and build actors (for version 4.0)
  - [actorlibrary](#) (for version 4.0)

- **Integration with workflow language**
  - through model classes poprzez klasy modelowe (see below in Workflow APIs/Libraries – moml package)
  - parser produces object of type ptolemy.kernel.util.NamedObj, which is extended by ptolemy.actor.Manageriand ptolemy.actor.lib.jxta.MoMLSimpleApplication
  - from the manager documentation: A Manager governs the execution of a model in a domain-independent way. Its methods are designed to be called by a GUI, an applet, a command-line interface, or the top-level code of an application.

- **Other:**
  - [dictionary](#) (version 4.0)
  - terms:
    1. actor – defines action
    2. director – defines how actions are executed, 1 for a workflow (ie. sequential, parallel…)
  - main idea behind "actor-oriented design"
    1. components responsible for the actions executed by a workflow are actors; actors communicate with other actors
    2. main difference between actor-oriented and object-oriented design: object-oriented design focuses on execution flow, control is passed between objects; in actor-oriented design focus is on executing specific roles in parallel – data is passed between actors (and not control)
    3. well-defined interface sets:
      1. internal state
      2. behavior
      3. possible interactions
    4. actor interface specifies:
      1. ports (communication channels for actors)
      2. parameters allowing configuration of actor’s behavior
7.6 Workflow APIs/Libraries

7.6.1 Ptolemy II – moml package

- **Programming language**
  - Java
- **Functionalities**
  - MoML parser
  - parsing code
  - **ParsingMoML**

```java
public static NamedObj parseFile(File momlFile) {
    String wsLocalisation = momlFile.getParent();
    Workspace ws = new Workspace(wsLocalisation);
    MoMLParser parser = new MoMLParser(ws);
    NamedObj result = null;
    try {
        // 1st parameter is an URL of workspace base
        result = parser.parse(
            new URL("file://" + momlFile.getAbsolutePath()),
            "model1.xml",
            new FileInputStream(momlFile));
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
    // result is instance of CompositeActor
    return result;
}
```

- The parse() methods of the MoMLParser class read MoML data and construct a Ptolemy II model. They return the top-level model. The same parser can then be used to incrementally parse MoML segments to modify that model.
- easiest way to retrieve all infrom about a model is to parse model XML and execute `ptolemy.kernel.CompositeEntity.deepEntityList()` (from Ptolemy II FAQ)
- Model serialization to XML (MoML) -

- **Is the project alive**
  - yes
- **Licence**
  - same as Ptolemy II
- **Other**
  - sources: svn://source.eecs.berkeley.edu/chess/ptII/trunk/ptolemy/moml
8 OWL-S

8.1 General

8.1.1 General characteristic

- A language based on XML
- OWL-S is a semantic web markup language for web applications. It is built on Web Services standards, has JAVA-API. OWL-S is working on objects defined in WSDL, in WSDL-defined processes.
- Language aims to enable the following tasks:
  - automatic Web Service discovery;
  - automatic Web Service invocation;
  - automatic Web Service composition into a Workflow (WF) and interoperation.

8.1.2 Basic language terms and structures:

- service profile - a description what and how can be managed by the WS. This is mainly for human operator.
- process model – an information about service execution: input parameters output data, initial conditions
- grounding – a description of technical details about accessing service, e.g. protocols.
- Elements of description:
  - process:atomicProcess
  - process:compositeProcess
  - owl:Class
  - owl:ObjectProperty

8.2 Language strength

8.2.1 Node types

- Each node must be defined and described

8.2.2 Data types

- Data is dynamically processed, there is possibility to create custom data types.

8.2.3 Extensibility

- Theoretically OWL-S allows to be extended, but this is not documented.

8.2.4 Conditional structures

- There is possibility to define conditions:
  - if-then-else
  - custom conditions by implementing predicates.
8.2.5 Loops (for-each, for-n, while)

- RepeatWhile i RepeatUntil

8.2.6 Data- or Control-driven?

- It is probably control-driven
- Allowed predicates: Choice, Sequence, AnyOrder, Split, and Split-Join

8.2.7 Subworkflows support

- It is possible to wrap a sub-workflow into individual service described by WSDL.

8.2.8 Other

- This is mainly a language for processing semantic information.

8.3 Popularity

8.3.1 Is it standardised?

- This project seems to be not developed since 2006. Lack of perspectives for standardization and further development.

8.3.2 Community

- A mailing list exists: [http://on.cs.unibas.ch/owls-api/mail-lists.html](http://on.cs.unibas.ch/owls-api/mail-lists.html)

8.3.3 Usage in other projects

- Not found

8.3.4 Interoperability

- Not found

8.4 Graphical representation

8.4.1 OWL-S Editor

- Programming language
  - JAVA
- Is the project alive / supported?
  - probably not being developed since 2004.
- Licence
  - lack of information
- Functionality
external editor of semantic web, it allows to execute, edition and saving of workflow. It is not supporting drag’n’drop operations.

- **Extensibility?**
  - standalone application
- **Graphical presentation**
  - this is a state diagram.
- **Other**

### 8.5 Workflow Engines

#### 8.5.1 OVM (Owl-s Virtual Machine)

- **Is the project alive / supported?**
  - probably not being developed since 2005-2006
- **Licence**
  - no information
- **Extensibility**
  - No information
- **Integration with workflow language**
  - workflow is executed by Web Services on remote computers. An RDF (Resource Description Framework) instance have to be embedded in OWL description, that allows to store intermediate and output data.
- **Other**
  - this is an reference implementation of the OWL-S, comparison test performed in 2007 shown many implementation gaps (a model was not fully implemented).
  - [http://www.wsmo.org/TR/d35/v0.1/20070403/](http://www.wsmo.org/TR/d35/v0.1/20070403/)

### 8.6 Workflow APIs/Libraries

#### 8.6.1 OWL-S IDE (Eclipse plugin)

- **Programming language**
  - JAVA
- **Is the project alive / supported?**
  - project is not active since 2005.
- **Licence**
  - no information
- **Functionality**
  - it supports a process of semantic web creation, does not support graphic visualisation of the such a web.
- **Other**
9 Scufl

9.1 General

9.1.1 General characteristic

- XML based;
- Simple Conceptual Unified Flow Language strictly connected with Taverna (and not supported outside it) and in official documentation is described as ‘volatile’;
- myGrid (Taverna and xScufl language were created for it) is dropping the usage of Scufl and migrates to the Scufl2;
- Scufl is not an independent language definition, it is strictly connected with Taverna Model Objects and was constantly changed along with model changes;
- "Unfortunately there is no XSD or DTD defining XScufl. The data format is meant for internal use only by Taverna" – there is no reason for further research.

9.1.2 Basic language terms and structures

- XScufl- means: it is written in Scufl’s XML
- Scufl contains:
  - inputs – input for workflow data;
  - outputs – output for wf’s data;
  - processes – workflows nodes, they have input and output ports (which are not the same things as above input / output sets), a set of data links which are describing data flow;
  - coordination links – are the syntax for connecting input and output ports on processors or overall workflow source, describing control dependences that cannot be strictly derived from graph structure (it is connected with fact that Scufl is data-driven. Language of such a type is not able to describe dependencies existing between nodes, where is no direct data flow (e.g. one node is obligated to wait for another end his job)
- 3 layers of Taverna processing:
  - Scufl - a workflow language for linking applications, is at the abstraction level of the user (probably the client is an RPC application of Eclipse IDE)
  - an execution layer interpreting the Taverna Data Object Model – language model, strict implementation of iterations, errors handling, retry operation, etc.
  - an extensible processor plug-in architecture for the Freefluo enactor manages the low-level "plumbing" invocation complexity of different families of services – is executing workflow implemented as Taverna Data Object (Workflow Object Model - model + information about services used in wf)

9.1.3 Historical context

- myGrid

9.2 Language strength
9.2.1 Loops (for-each, for-n, while)

- loops in data-driven language are implemented as execution of 1 operation for many input data sets
- there are following types of data iterations:
  - map – if processor has 1 input port and is processing data x with output $f(x)$, then given set of input data $(x_1, x_2, x_3)$ produces $(f(x_1), f(x_2), f(x_3))$;
  - cross - if processor has more than 1 input port and is implementing method $f(x, z)$ then a set of input data $(x_1, x_2, (z_1, z_2))$ produces $(f(x_1, z_1), f(x_1, z_2), f(x_2, z_1), f(x_2, z_2))$;
  - dot - if processor has more than 1 input port and is implementing method $f(x, z)$ then a set of input data $(x_1, x_2, (z_1, z_2))$ produces $(f(x_1, z_1), f(x_2, z_2))$.

9.2.2 Data- or Control-driven?

- Data flow

9.3 Popularity

9.3.1 Usage in other projects

- Taverna

9.3.2 Interoperability

- Translation to the GWorkflowDL exists – it based on XSLT, is developed in CppWfMS project.

9.4 Workflow Engines

9.4.1 Freefluo workflow enactment engine

- an extensible processor plug-in architecture for the Freefluo enactor manages the low-level "plumbing" invocation complexity of different families of services

10 SimPEL

10.1 General

10.1.1 General characteristic

- script language
- based on JavaScript

10.1.2 Basic language terms and structures

- language structures:
  - **ns_id** - identifier of language elements (bloks, processes, variables)
- **block** – a piece of code, surrounded with {} brackets, contains structures e.g. `{reply();}

- **param_block** – parameterized block, e.g. { |a| reply(a); }  

- **body** – content

- process – a process definition, described in following way: `process : 'process' ns_id body;`

### 10.1.3 Historical context

- it was created as solution for long running asynchronous processes and as alternative to complicated BPEL language;
- developed by INTALIO company as an open source project

### 10.2 Language strength

#### 10.2.1 Node types

- nodes are processes

#### 10.2.2 Data types

- integers
- floats
- strings with escape sequences ("abc" or "\t\f\n") and XML (bar)
- Native arrays and hashes support is also planned (for the moment, XML partly fills that gap). Numbers have the same characteristics as in JavaScript. Integers are considered reliable (numbers without a period or exponent notation) to 15 digits.

#### 10.2.3 Extensibility

- by definition of processes

#### 10.2.4 Conditional structures

- if-else (operators that are allowed in expressions: ==, !=, <, >, <=, >=, &&, ||, +, -, *, /, ! and unary -)
- wait();

#### 10.2.5 Loops

- while
- specification mentions also until, for-each, for-all, but they are not implemented yet

#### 10.2.6 Other

- well integrated with RESTfull services, easy resource access

### 10.3 Popularity
10.3.1 Is it standardised?
- no

10.3.2 Community
- no

10.3.3 Usage in other projects
- unclear support from Apache ODE

10.3.4 Interoperability
- No information available

10.3.5 Other
- A few documents describing language are available and the mailing list is not working any more

10.4 Workflow Engines

10.4.1 Simplex 0.2 + Singleshot
- *Is the project alive*
  - not very vivid, a few documents available, mailing list is not working any more
- *Licence*
  - GPL v.3
- *Extensibility*
  - ?
- *Integration with workflow language*
  - All SimPEL processes run tightly integrated with their own JavaScript shell
  - It is possible to integrate it with user/3rd party libraries

11 XPDL

11.1 General

11.1.1 General characteristics
- XML-based
- XPDL is a workflow description language that has also the possibility to include information of workflow’s graphical presentation as well as information of how to run the workflow.
- It can cover all BPMN language features (BPMN is a subset of XPDL)
- The goal of XPDL is to store and exchange the process diagram, or specifically to allow one tool to model a process diagram, and another to read the diagram and edit, another to "run" the process model on an XPDL-compliant BPM engine, and so on.
11.1.2 Other

- Comparision of support for processing patterns:
  [http://is.tm.tue.nl/research/patterns/standards.htm](http://is.tm.tue.nl/research/patterns/standards.htm)

12 YAWL

12.1 General

12.1.1 General characteristic

- XMP-based language
- *Yet Another Workflow Language*. Business workflow language description. Build on Petri nets, extending them to implement some of the patterns ad higher abstraction level (those patterns are: synchronizing merge, discriminator, N-out-of-M join, multiple instance with no a priori runtime knowledge and cancel case). It is considered BPEL’s competition – the main differences are that YAWL supports ‘human’-actions, but lacks industry support an is not a standard. Only one implementation of YAWL specification is available.

12.1.2 Basic language terms and structures

- *Terms used by language*:
  - role;
  - actor;
  - task;

12.2 Language strength

12.2.1 Node types

- Each node is mapped to task, which can be an action execution, condition check or flow steering.

12.2.2 Data types

- Data is embeded in XML’s namespace (xs) and custom types can be build on top of basic ones:

  ```xml
  <xs:simpleTypename="CurrencyType">
  <xs:restrictionbase="xs:string">
  <xs:enumerationvalue="AUD"/>
  <xs:enumerationvalue="USD"/>
  </xs:restriction>
  </xs:simpleType>
  ```

12.2.3 Extensibility

- Custom/user language elements cannot be added.
12.2.4 Conditional structures

- Logical and time conditions can be defined. If desired functionality is not supported directly it can be implemented as a custom task of condition check.

12.2.5 Loops

- It is possible to create custom task that will act as a loop, which is not directly supported by the language.

12.2.6 Data- or Control-driven

- Depending on transition conditions defined it can be both - control- or data-driven.

12.2.7 Subworkflows support

- YAWL has a possibility of defining black-boxes which can be other YAWL definitions or independent Web Services.

12.2.8 Other

- Task description:

```xml
<task id="Produce_Freight_Invoice_595">
  <name>Produce Freight Invoice</name>
  <flowsInto>
    <nextElementRef id="null_597" />
  </flowsInto>
  <join code="xor" />
  <split code="and" />
  <startingMappings>
    <mapping>
      <expression query="&lt;FreightInvoice&gt;&lt;OrderNumber&gt;{/Payment/POrder/Order/OrderNumber/text()}&lt;/OrderNumber&gt;&lt;/Company&gt;{/Payment/POrder/Company/*}&lt;/Company&gt;&lt;/FreightCost&gt;{/Payment/POrder/FreightCost/text()}&lt;/FreightCost&gt;&lt;/FreightInvoice&gt;" />
      <mapsTo>FreightInvoice</mapsTo>
    </mapping>
  </startingMappings>
  <completedMappings>
    <mapping>
      <expression query="&lt;FreightInvoice&gt;{/Produce_Freight_Invoice/FreightInvoice/*}&lt;/Produce_Freight_Invoice/FreightInvoice/&gt;" />
      <mapsTo>FreightInvoice</mapsTo>
    </mapping>
  </completedMappings>
  <resourcing>
    <offer initiator="system">
      <distributionSet>
        <initialSet>
          <role>RO-326e7029-c9b5-4a7e-a6ea-d310bc43ca9d</role>
        </initialSet>
      </distributionSet>
    </offer>
  </resourcing>
</task>
```
12.3 Popularity

12.3.1 Is it standardised?

- It is not a standard and probably will never become it.

12.3.2 Community

- The project seems to be active and in development. Discussion forums are available.

12.3.3 Usage in other projects

- Some projects, none of them is famous:
  - first:utility and first:telecom in the UK, both companies part of the Impelloplc group of companies, providing energy and telecoms services respectively, have been collaborating with QUT for the past 2 years on the YAWL project. These companies have built software around the YAWL system providing a novel approach to page navigation for web based systems together with the more traditional use of choreographing long-lived business processes.
  - GECKO, a Germany-based software development and IT services company, aims to make use of the new YAWL 2.0 release for supporting resource-centric workflows in the context of an R&D project. The project addresses the application domain of clinical surgical suites and aims to provide support for peri-operative clinical processes based on the YAWL system as well as by building extensions to support e.g. cooperating workflows and non-human resource management.
  - The Knexus Research Corporation collaborates with the YAWL foundation to provide YAWL-related services to the US Naval Research Laboratory. MMT Seguros, a car insurance company based in Spain, is using YAWL to implement claims management and internal quality assurance processes.
  - The FERP Workflow System uses YAWL as part of its software. InsuraPro Workflow claims to use YAWL as its "mathematical footing". As stated in this paper the workflow management module of Axserion "includes concepts from YAWL". SWS has based the long term development strategy for its FlowConnect platform on YAWL and the workflow patterns. A Petri net design tool called Woped, developed at the Berufsakademie Karlsruhe, Karlsruhe, Germany, claims to have an interface to YAWL.
12.3.4 Interoperability

- No information on that available, but tools for transforming BPEL to YAWL exist.

12.4 Graphical representation

12.4.1 YAWL Editor

- Programming language
  - Desktop JAVA application, source code available.
- Is the project alive?
  - Seems to be alive – last release (2.1) was made in 2010, earlier: 2.0.1 in 2009, 2.0 in June 2008.
- Licence
  - GNU Library or Lesser General Public License (LGPL)
- Functionalities
  - Definition of new workflows, which can be saved in YAWL format, it has access to clipboard.
- Extensibility
  - No, which is a result of limitations in language extensibility
- Graphical presentation
  - Customisation is possible, graph has a fixed set of icons

12.5 Workflow Engines

12.5.1 YAWL Engine

- Is the project alive?
  - The project is alive – new releases are published, the last one from 2010-07-30
- Licence
  - GNU Library or Lesser General Public License (LGPL)
- Extensibility
  - Service-oriented architecture, services can be exchanged and user’s extensions implemented
- Integration with workflow language
  - The engine is run within Tomcat’s container and it is integrated through Web Services.