A Domain Specific Language for Digital Libraries' Interoperability

João Edmundo, José Borbinha

INESC-ID, Rua Alves Redol 9, Apartado 13069,
1000-029 Lisboa, Portugal
IST – Department of Information Science and Engineering, Instituto Superior Técnico,
Lisbon Technical University, Portugal
{joao.edmundo, jlb}@ist.utl.pt

Abstract. Humans, in order to create, share and improve knowledge on business processes, need a common, readable and preferably visual notation. In addition, since Internet is more and more a platform for global application sharing, the requirements for Web applications concerning workflow execution, interaction, aesthetics and Web service integration are steadily increasing. In this paper we propose a Domain Specific Language, implemented as an extension of the standard BPMN 2.0 language specifically to the domain of digital libraries' interoperability, and use it to define and execute data harvest processes. To support it, we designed an architecture and implemented a real case of a computational environment based on the *jBPM* and GWT frameworks. This approach allowed us to provide a web-based, natural and flexible way not only to create and manage domain specific processes, but also to monitor each process execution, allowing process managers to understand the process history, its current state and possible future execution. In our opinion, creating specific languages for domains, and putting activity of interest in its visualized context, makes the user knowledge more comprehensive.

Keywords: Process Orchestration; BPMN 2.0; Specialization; Workflow Infrastructure; Domain Specific Language;

1 Introduction

Libraries, archives, museums and other cultural heritage organizations face the need to share their resource description metadata in international initiatives such as Europeana[1], TEL[2] and EuDML[3]. In these scenarios, having a system not supporting natively the commonly required OAI-PMH[4] protocol is an important constrain. Commercial and open-source solutions for this problem exist, but the first imply investments not always possible, and using open-source software might require some

[1] Europeana - http://dev.europeana.eu/
[2] TEL - The European Library- http://www.theeuropeanlibrary.org
[3] EuDML- European Digital Mathematics Library- http://www.eudml.eu/
[4] OAI-PMH - Protocol for Metadata Harvesting - http://www.openarchives.org/pmh/

technical expertise for local customization, often not found in the staff of those organizations. Also, new emerging scenarios for transfer not only of data sets but also the contents referenced by these data sets (for example, the harvesting of the full-text of the documents described in the data sets) require the support for more sophisticated harvesting and aggregation processes.

REPOX [4] is an open-source framework to address that problem. It was designed and developed for convenient usage by both intended data providers and service providers, requiring little technical knowledge and effort, thus supporting a fast start process (installation and configuration). It is focused on common metadata interoperability scenarios, offering not only the publication and harvesting, but also support for metadata transformation. Consequently, REPOX is also a convenient tool for service providers. However, the initial releases of REPOX only provide metadata harvesting services that are used within built-in processes. These processes cannot be shared between REPOX installations neither edited nor extended (to harvest objects' contents, for example) without programming knowledge. So, a new system's design is necessary, for which not also new frameworks to develop web applications and interfaces must be explored, but also find new ways to define and orchestrate these harvesting processes.

Overall, this research seeks to develop a solution to orchestrate business processes using a Domain Specific Language (DSL) based on the BPMN 2.0, in scenarios of data aggregation and systems interoperability in digital libraries.

2 Related Work

Many people consider Business Process Management (BPM) to be the "next step" after the workflow wave of the nineties. BPM is a management approach focused on aligning all aspects of an organization with the wants and needs of clients. It is an approach that promotes business effectiveness and efficiency while striving for innovation, flexibility, and integration with technology. As a mean to visually represent BPM, the Business Process Modeling Notation (BPMN) was created which is based on a flowcharting technique very similar to activity diagrams from UML, tailored for creating graphical models of business process operations. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation [13]. The current version of BPMN specification is 2.0[5] . It not only defines a standard on how to graphically represent a business process like BPMN 1.x, but also includes execution semantics for the elements defined, and an XML format on how to store process definitions.

Flexible and innovative business processes are one of the key elements that enable modern organizations to succeed. According to Janis Barzdins et al. [1], and Marjan Mernik et al. [8], there is a growing need to consider new issues when implementing tools for domain specific languages with an orientation to the business process management. Although general languages can cover a large variety of cases, they add unnecessary complexity to specialized systems [1]. Therefore, specific languages for narrow business domains are required. As a result, some solutions presented by Steen

[5] BPMN 2.0 - http://www.omg.org/spec/BPMN/2.0/

Brahe et al. [3] and Momotko [9] show a set of guidelines used when defining a DSL, and present techniques for DSL creation based on BPMN, through the use of colors and custom icons.

To find a suitable approach to define and execute our DSL, we studied some workflow technologies. Although several commercial products like Microsoft's *BizzTalk*[6] and IBM's *WebSphere* [6] are widely used in the BPM domain, some open-source solutions also start to appear. These solutions like *jBPM* [7] and *Activiti* use BPMN 2.0[8] as the core process definition and execution language to their process engine, which gives them the flexibility to define complex processes. However, they still lack the process execution monitoring interfaces and modeling input interfaces, which are important issues for process managers [11] [2].

Finally, to create our web framework for process orchestration we searched for the right tool for the job. After comparing some web development frameworks like Prototype+*script.aculo.us* , jQuery, ExtJS, MooTools, Dojo, Google Web Toolkit (GWT) [7] [5], and ZK, we concluded that GWT, though it has a medium learning curve and ease of use, it provides a high performance and extensible framework to build complex UI interactions, eventually being the more fitting to our problem.

3 Proposed DSL

After analyzing the goals and requirements established for a digital library's interoperability and aggregation system in the Europeana Libraries project [12], the knowledge obtained while developing a new web visual interface for the REPOX 2.0, and using some DSL definition principles proposed by Steen Brahe et al. [3], we were able to design our DSL.

To characterize each concept within the DSL, a set of standard visual representations were used, and will be explained in this chapter.

Name	Icon	Description
One Data Record		A Data Record (also commonly called in the context of digital libraries Metadata Record) is a structure of information describing one information resource
One Record Set		A Record Set is a collection of Data Records
One Record Subset		A Record Subset is in itself a Record Set made as a subset of a Data Record Set
One Data Provider		A Data Provider is an entity that publishes one or more Data Sources: the publication of a Data Source corresponds to make available a Record Set
One Data Source		The way by which a Data Provider makes available a Record Set, which can happen by multiple means (a file available from an FTP or HTTP server, an OAI-PMH service, etc.)

Table 1. Information Entities in the DSL.

[6] BizTalk - http://www.microsoft.com/biztalk/en/us/default.aspx
[7] jBPM - http://www.jboss.org/jbpm
[8] BPMN 2.0 - http://www.omg.org/spec/BPMN/2.0/

These entities, which represent a piece of data or a group of pieces of data with a unique semantic definition (**Table 1**), allow the exchange of information between the DSL tasks described further in this section. Next, in **Table 2**, some operations/actions are represented using some standard visual symbols.

Name	Icon	Description
Create		Creation/Save/Edit a certain entity
Delete		Delete/Remove/Cancel an entity
Harvest/Get		The input of data into the system
Export		The output of data from the system
Schedule		The scheduling of a certain task
Transform		A transformation of a Data Record or a Data Set (usually, a conversion to a new format, according to a new schema). Can be specialized into a filter or data analysis transformation
Package		Grouping of several instances into one

Table 2. Description of Operations/Actions in the DSL.

Finally, the *Technology* concepts (names of the protocols applied in data harvest) used in this DSL are visually represented by their own written name since they don't have a standard symbol that represents them: **OAI**, **Z39.50** and **Folder**.

These previously described concepts lead to the visual representation of our tasks in the DSL, presented in the next section.

After describing the main concepts of the domain, defining the information entities, and choosing the visual representation of the concepts in our DSL, a set of tasks are proposed. We start by presenting the tasks related to Data Providers, in **Table 3**.The set of tasks regarding the Data Sources is proposed in **Table 4**. Finally **Table 5** shows the tasks for the Data Records.

Task	Icon	Description
Create/Edit Data Provider		Creates/Edits a Data Provider in the system through its name, country and description, returning a Data Provider entity
Delete Data Provider		Deletes an existing Data Provider (considering its identifier)

Table 3. DSL proposition tasks and their characteristics for Data Providers.

Task	Icon	Description
Create/Edit Data Source - OAI		Creates/Edits an OAI-PMH data source using an OAI-PMH repository urn and data set, and returns a Data Source entity.
Create/Edit Data Source - Z39.50		Creates/Edits a Z39.50 data source. It can be of type *Timestamp*, *Id List*, or *Id Sequence* according to the Z39.50 Type field.
Create/Edit Data Source - Folder		Creates/Edits a Folder data source. It can be of type FTP, HTTP or Folder as a pathname in an actual file system, according to the FType field.
Delete Data Source		Deletes an existing Data Source with a given identifier
Harvest Data Source		Initiates a harvesting session of a Data Source through its identifier, and return a Record Set entity with the harvested data
Harvest Data Source Sample		Initiates a harvesting session of a Data Source, ingesting only a pre-defined number of records.
Delete/Cancel Task		Deletes a scheduled or already running export or harvest task of a Data Source through its identifier
Export Data Source		Exports the records from a given Data Source using its identifier, and a user-defined number of records per file
Create/Edit Schedule Harvest		Creates/edits a scheduled Data Source harvest task starting on a given date and with a certain period
Create/Edit Schedule Export		Creates/edits a scheduled Data Source export task starting on a given date and with a certain period

Table 4. DSL proposition tasks and their characteristics for Data Sources.

Finally, in order to define and execute processes using this DSL, we developed a BPMN 2.0-based architecture which will be described in the next section.

4 Implementation and Solution

In this section we describe a possible solution to create an extensible architecture that enables the orchestration of business processes based on the DSL described in the previous section. Additionally, we explain how the development process was managed and why some of the choices were made.

4.1 Process Orchestration Architecture

After some analyses we decided to use the *jBPM* framework due to, among other features, its capability of running BPMN 2.0 defined processes and ease of extension to add new types of tasks. Also, the development of a web application was possible

Task	Icon	Description
Save Data Record		Saves one new record using its identifier, content and the Data Source identifier it should belong to, and return the new Data Record entity
Delete Data Record		Deletes one existing Data Record using its identifier
Publish Record Set		Publishes a Record Set to a database or file system
Get Record Set		Retrieves a Record Set from a Data Source with a given identifier, and returns the corresponding Record Set entity
Transform Record Set		Transforms a Record Set with a given schema to another schema. A specialization of a transformation might also represent the application of a filter or even a data analysis of a Record set
Package Data Records		Groups one or multiple Data Record into one Record Set entity
Harvest Data Record Full-Text		Harvests the full-text content of one given Data Record with a given identifier
Harvest Record Set Full-Text		Harvests the full-text content of a given Record Set using its associated Data Source identifier

Table 5. DSL proposition tasks and their characteristics for Data Records.

through the GWT, Ext GWT[9] and a GWT-SVG library developed by us. The following **Figure 1** represents the defined architecture for the process orchestrator.

The core of our Process Engine is the Process Orchestrator, composed by a Process Manager that launches new processes in the *jBPM* engine and the Process State Manager which monitors the state of each process (**Figure 1**). Both these managers share the access to a common list of processes, managed by the Process Planning. Each process consists in an orchestration of a set of Web Services that are registered within the *jBPM*. The Process Orchestrator is constantly monitoring all the running processes. The Process Orchestrator provides an interface so that new processes can be added. These processes can be defined visually using the DSL based on the BPMN 2.0 notation, which is then coded in XML, according to the format defined by the OMG – Open Management Group.

The web application is composed by a client side containing a Process Editor that supports the visual definition of new processes, and by a Process Instance Viewer that allows the runtime monitoring of each process instance. The processes defined on the client side are persisted in an extended BPMN 2.0, through the Process Definitions Manager. Initialized process instances are started through the Process Orchestrator, called by the Process Instances Manager.

[9] Ext GWT - http://www.sencha.com/products/extgwt/

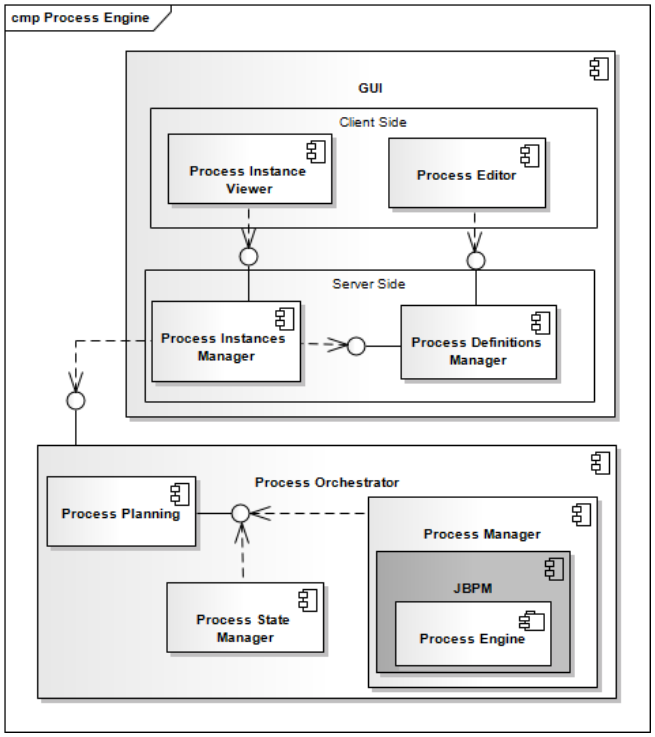

Figure 1. Process Engine architecture.

This architecture allowed us to create an extensible web process framework that enables process orchestration from its definition to execution and runtime monitoring.

4.2 BPMN 2.0 Extension

Being this DSL an extension of BPMN 2.0, we started by choosing which BPMN 2.0 main components we should use to help define our processes. As a result, and according to Michael Muehlen et al [10] analysis on usage of BPMN on different areas, we noticed that the Sequence Flow, the Parallel Gateway, and the Start and End Events were the most popular BPMN components used by BPMN modelers, so we decided to include them to our created components (Section 3).

In order to create a DSL that would be executable in the *jBPM* engine we developed a BPMN 2.0 extended language based on the BPMN 2.0 semantic definition[10]. In addition to the BPMN 2.0 data, all our process components have position and state, used for process runtime monitoring, and additional input data for each task in the process.

To define a process, the user is presented with a standard BPMN editor interface with the drag and drop grid in the middle, the available tasks on the left side, and on the right side the selected task's properties. On the properties panel, our solution adds

[10] BPMN 2.0 XSD- http://www.omg.org/spec/BPMN/2.0/20090502/Semantic.xsd

a customized input interface for each task type, granting the user the capability of specifying the input of each task while defining the process.

To successful monitor more than one process we present a table interface where the user can see all running process instances and their state of execution through each task's state. Therefore, to represent the current state of a task within a process, colors and stroke patterns were used (**Figure 2**). Such approach enables process managers to check quickly what the current status of the process is, improving their monitoring performance [9].

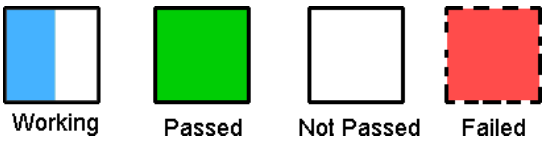

Figure 2. State color representation.

A more detailed view of each process instance can be accessed, where the user is presented with a single process and its log data, organized in a chronological manner, presenting a more complete analysis of what's happening in the process.

Overall, the proposed solution for our process engine uses the *jBPM* and GWT frameworks. The first one is used to run our extended BPMN 2.0 processes that represent the Proposed DSL described in section 3, and are defined through a BPMN 2.0 XML schema and visual notation extension. The GWT framework is separated in a server side that communicates with the *jBPM* and therefore manages process execution and perseverance. On the other hand, the GWT's client side manages the interfaces used for process modeling (where some BPMN 2.0 base components are used and several interface techniques for custom input during modeling are applied), process definition and instance management and process runtime monitoring (made more efficient through a color and stroke pattern system for each component state representation.

5 Practical Results

The idea for the verification of the hypothesis was to apply the developed DSL-based prototype to the REPOX framework used in scenarios like Europeana, TEL, EuDML and SHAMAN projects. The main purpose is to evolve the currently used "hard-coded" architecture into a service oriented architecture with a web human interface for process orchestration, and as a result, provide more flexibility using XML for process representation and improve user performance on process modeling and monitoring [9] through the use of a DSL, which is easier to use by domain experts [3].

We evaluated some simple harvest processes and more complex ones to test our solution. As an example of a complex process (**Figure 3**) we decided to define a process that contains the full cycle of data harvesting, from the creation of the data provider and their data sources, to their harvest and publishing, taking into account the success/failure of one of the harvests, and the existence of full-text on one of the harvested record sets.

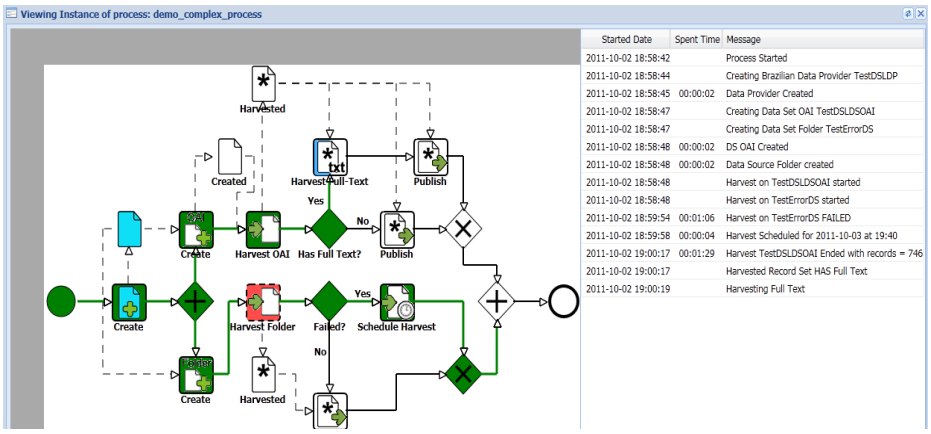

Figure 3. Complex process runtime monitoring.

The process represented in the previous figure begins by creating the base storage structures for data harvest (a data provider with two data sources of different types – OAI and Folder). This data source creation is done in parallel, which allows us to on one side, harvest the data from the OAI data source and then, after verifying the existence of full-text on the harvested record set, harvest it as well. On the other side, and since we are dealing with a system where unexpected problems can occur, we simulate a harvest failure which we deal with by scheduling it to a future date.

As shown in **Figure 3**, the running process shows detailed information on the right side log grid about each task's data, like for example the chosen name (*TestDSLDP*) and country (*Brazil*) of the data provider, the names of the created data sources (*TestDSLDSOAI* and *TestErrorDS*), harvested record count (*746*), scheduled harvest date and time (*03/10/2011 at 19:40*), the *TestErrorDS* ingest status, and all the task's start dates and spent times. Finally, analyzing the process' log data and the REPOX's web interface (**Figure 4**) we can confirm that the data obtained during the process' execution corresponds to the one represented in REPOX, and therefore showing the successful integration between our solution and the REPOX framework.

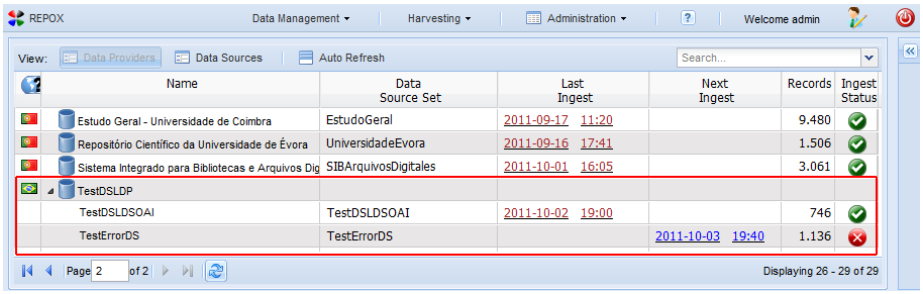

Figure 4. REPOX web interface used in the TEL project after the previously described complex process' execution.

6 Summary and Future Work

Although domain-specific languages (DSLs) are costly to design and implement, and require the learning of a new language with limited capability (only applied to its domain), they allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. Consequently, domain experts themselves can understand, validate, modify, and often even develop DSL programs. As a result, a good DSL can enhance quality, productivity, portability and reusability [8].

In general, we were able to create a DSL that enables digital libraries interoperability when integrated with a web service architecture for process orchestration.

Our proposed solution:

- Eliminates the need for technical knowledge in the creation of business processes through a process modeler based on a BPMN 2.0 extended visual notation;
- Increases user performance in process monitoring through a color system used to represent a process' tasks state;
- Increases flexibility in process exchange and through the use of a persistent format like XML to represent each process;
- Is extensible enough to easily create new tasks, with each task representing a web service handler, and its actual behavior created through the definition of this handler, its visual notation, and output and input data.

Overall, our results suggest that our solution is a valid approach to support digital library interoperability, at least in real scenarios like Tel, Europeana, EuDML and SHAMAN.

In the future, the process modeler should provide additional BPMN primitives like Human Tasks. These kinds of tasks, as the name suggests, only complete after human order. They are important to define scenarios where a human decision must be made within a process. Also, new visual notation approaches like Momotko [9] should be considered, in which additional runtime information is inserted directly on the task's visual notation, and therefore enhancing process monitoring. Finally, although now we provide support to define complex transformations in the harvesting processes of the current REPOX version (2.0), our goal is to extend the use of our processes to all processes in REPOX, and enable the seamless exchange of process definitions between REPOX instances.

7 References

[1] Janis Barzdins, Karlis Cerans, Mikus Grasmanis, Audris Kalnins, Sergejs Kozlovics, Lelde Lace, Renars Liepins, Edgars Rencis, Arturs Sprogis, and Andris Zarins. Domain specific languages for business process management: a case study. *The 9th OOPSLA Workshop on Domain-Specific Modeling*, 2009.

[2] Stefan Betermieux and Birgit Bomsdorf. Finalizing dialog models at runtime. In Luciano Baresi, Piero Fraternali, and Geert-Jan Houben, editors, *Web Engineering*, volume 4607 of *Lecture Notes in Computer Science*, pages 137–151. Springer Berlin / Heidelberg, 2007.

[3] Steen Brahe and Kasper Østerbye. *Business process modeling: Defining domain specific modeling languages by use of uml profiles*, pages 241–255. Springer, 2006.

[4] Nuno Freire and José Borbinha. Metadata spaces: The concept and a case with repox. In *Research and Advanced Technology for Digital Libraries*, volume 4172 of *Lecture Notes in Computer Science*, pages 516–519. 2006.

[5] Federico Kereki. Web 2.0 development with the google web toolkit. *Linux J.*, 2009(178):2, 2009.

[6] M. Kloppmann, D. König, F. Leymann, G. Pfau, and D. Roller. Business process choreography in websphere: combining the power of bpel and j2ee. *IBM Syst. J.*, 43(2):270–296, 2004.

[7] Ryan McFall and Charles Cusack. Developing interactive web applications with the google web toolkit. *J. Comput. Small Coll.*, 25(1):30–31, 2009.

[8] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37:316–344, December 2005.

[9] Mariusz Momotko and Bartosz Nowicki. Visualisation of (distributed) process execution based on extended bpmn. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, DEXA '03, pages 280–, Washington, DC, USA, 2003. IEEE Computer Society.

[10] Michael Zur Muehlen and Jan Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 465–479, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Michael zur Muehlen and Michael Rosemann. Workflow-based process monitoring and controlling - Technical and organizational issues. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6*, HICSS '00, pages 6032–, Washington, DC, USA, 2000. IEEE Computer Society.

[12] Stefanie Ruehle, José Borbinha, and Gilberto Pedrosa. *Deliverable 4.1 - Requirements Infrastructure and Harvester*. Europeana Libraries: Aggregating digital content from Europe's libraries, 2011.

[13] S. White. Using BPMN to model a BPEL process. *BPTrends*, 3(3):1–18.