Conceptual-Physical Bridging – From BPMN Models to Physical Implementations on Kettle

Bruno Oliveira[1], Vasco Santos[1], Claudia Gomes[1], Ricardo Marques[2], Orlando Belo[1]

[1] ALGORITMI R&D Centre
Department of Informatics, School of Engineering, University of Minho
Campus de Gualtar, 4710-057 Braga, PORTUGAL
[2] WeDo Technologies
Centro Empresarial de Braga
Ferreiros, 4705-319 Braga, PORTUGAL

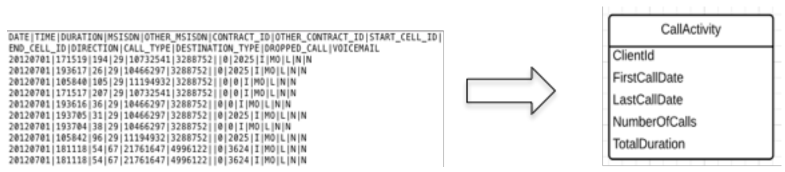
Abstract. Developing and implementing data-oriented workflows for data migration processes are complex tasks involving several problems related to the integration of data coming from different schemas. Usually, they involve very specific requirements - every process is almost unique. Having a way to abstract their representation will help us to better understand and validate them with business users, which is a crucial step for requirements validation. In this demo we present an approach that provides a way to enrich incrementally conceptual models in order to support an automatic way for producing their correspondent physical implementation. In this demo we will show how B2K (Business to Kettle) system works transforming BPMN 2.0 conceptual models into Kettle data-integration executable processes, approaching the most relevant aspects related to model design and enrichment, model to system transformation, and system execution.

Keywords: Data Migration Processes Modeling, BPMN 2.0 Conceptual Modeling, Domain Specific Languages, Metadata-driven Approaches, Kettle executable packages.

1 Introduction

Today, companies need to analyze large amounts of data that increases often exponentially day-after-day. Such tendency is revealed mainly due to the need that companies have to control the dimensions of analysis involved with their business contexts. Understanding such dimensions (an correlated variables) provides business monitoring and control, as well as ensures some real business advantages relatively to their most direct competitors. However, processing large amounts of data, frequently appearing in very disparate formats, it is a difficult task. The costs and the risks are higher and the success to do this appropriately is difficult to achieve. Thus, it is essential that stakeholders stay perfectly tuned, namely business users, which will interpret the results, and technical users, which will develop such data migration processes, ensuring the adequacy of the system to business requirements. The use of business process models is a common practice in many organizations, particularly using *Business Process Model and Notation* (BPMN) [1]. The BPMN notation was created to simplify modeling business processes and set a standard for the area. It has become a widely used notation on modeling business processes mainly because of its simplicity, expressiveness, and application scenarios. Moreover, BPMN presents a well-defined semantic structure, and provides an easy working platform.

A data migration processes can be considered as a specific business process, i.e. a data oriented workflow composed by several tasks logically related to transform and adapt raw data to a specific format. Current data migration tools provide strong constructs covering already several common transformations used on data migration processes. However, they are very specific tools, requiring very specialized staff to understand its specificities. To attenuate this advantage, we designed and implemented a tool with the ability to instantiate traditional BPMN models to specific Kettle [2] transformation models, providing an automatic mapping of the specification of conceptual models to some physical instances, which have the possibility to be executed directly in a commercial tool. To make this possible, we developed a *Domain Specific Language* (DSL) that allows for enriching conceptual BPMN conceptual models with the description of transformation tasks. Thus, the BPMN language provides the coordination of tasks in a workflow environment, while each task represents a transformation component specially configured using the DSL we developed. This way, task behavior can be generalized, simplifying processes development and facilitating communication between non-technical users without compromising its mapping to execution primitives. With this demo we demonstrate how this tool works transforming a BPMN conceptual model directly into an executable data migration system interpreted by Kettle.



```
DATE|TIME|DURATION|MSISDN|OTHER_MSISDN|CONTRACT_ID|OTHER_CONTRACT_ID|START_CELL_ID|
END_CELL_ID|DIRECTION|CALL_TYPE|DESTINATION_TYPE|DROPPED_CALL|VOICEMAIL
20120701|171519|194|29|10732541|3288752||0|2025|I|MO|L|N|N
20120701|193617|26|29|10466297|3288752||0|2025|I|MO|L|N|N
20120701|105840|105|29|11194932|3288752||0|0|I|MO|L|N|N
20120701|171517|207|29|10732541|3288752||0|0|I|MO|L|N|N
20120701|193616|36|29|10466297|3288752||0|0|I|MO|L|N|N
20120701|193705|31|29|10466297|3288752||0|2025|I|MO|L|N|N
20120701|193704|38|29|10466297|3288752||0|0|I|MO|L|N|N
20120701|105842|96|29|11194932|3288752||0|2025|I|MO|L|N|N
20120701|181118|54|67|21761647|4996122||0|3624|I|MO|L|N|N
20120701|181118|54|67|21761647|4996122||0|3624|I|MO|L|N|N
```

Fig. 1. A CSV file data source structure (left) and a target repository schema (right)

2 System architecture

Let us consider a simple data migration process. This process was built to read data in a CSV (*Comma-Separated Values*) file, containing phone calls activity data, into a relational table structured to receive some aggregated data (Fig. 1). The first to do is to model the process using BPMN in a workflow, which logic describes process execution constraints. The BPMN gateways and events allow for the description of specific rules in the coordination of all BPMN tasks represented. These tasks represent typical standard procedures or routines, which are composed by atomic clustered tasks representing a composite set of transformations configured based on a specific skeleton. For this demonstration we used BizAgi [3] to model the data migration process. Observing the BPMN model (Fig. 2), we see that the process starts with a data extraction task over a CSV file using a regular activity. All activities' names follow a convention using the # character, identifying activities as containers of tasks configured using the DSL proposed. The second task (*#DQE (Data Quality Enhancement)# - Normalize rows*) decomposes original data in order to homogenize the grain of the source file. Each row of the CSV file will generate two distinct records, representing the caller and the called client in a phone call record. Next, four BPMN tasks (*#Aggregate#*) are triggered to perform data aggregation operations. For that, two BPMN parallel gateways are used: the first one, a divergent gateway that determines the parallel execution of the aggregate tasks; and a convergent gateway to force the completion of all the parallel tasks in order to proceed with the remaining process activities.

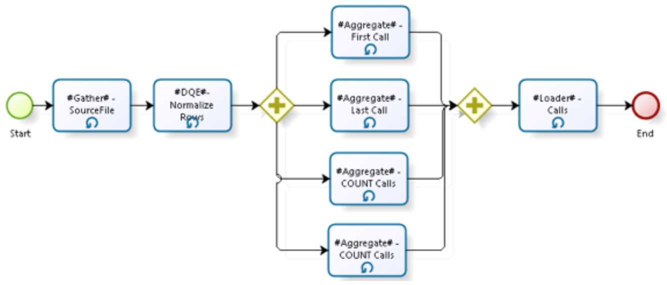


Fig. 2. The BPMN Conceptual model to support data migration for the scenario presented

Finally the process ends with the execution of a *#Loader#* container configured to load data into the relational table. For each activity we used a specific loop marker to mean that it should be executed iteratively based on a specific criteria – e.g. we can enforce incoming records to be processed individually or in groups in each iteration. We can also enrich tasks with additional BPMN elements to describe process execution at conceptual level. Data objects such as 'Data Input/Output', 'Data Storage' or 'Annotations' can be used to refine the process interpretation. After the enrichment process, it is necessary to configure tasks using the DSL proposed. Based on traditional migration processes architectures, we defined three categories of components, namely: 1) gathers, which are used in extraction operations – e.g. index access readings, change data captures, or log file data extractions; 2) transformers, which are used for data transformations – e.g. cleaning, conciliation or aggregation operations; and 3) loaders, which are used to load data into a target repository – e.g. bulk operations, slowing changing dimensions, or quarantine operations. These categories allow for grouping frequent tasks that can occur in all operational stages. In Fig. 2 we can see three typical composite tasks: a *Gather* task for extracting data from a CSV source; some *Transformers* tasks, DQE and Aggregate tasks to conform and summarize data; and a *Loader* task to put data into a data repository. Each one of these components has the ability to receive behavior specifications wrote in the DSL we developed using Xtext framework [4] and validated by its correspondent parser. Fig. 3 shows an excerpt of the DSL grammar. The task enrichment was defined to be compatible with several modeling tools. For the case of BizAgi, the DSL can be written using the documentation box available on the task properties panel or using a text annotation object. To demonstrate the application of the DSL grammar we prepared a small example (Fig. 4) to describe the internal behavior of the task #Gather# represented in Fig. 2. The block 'sources' is used to describe the metadata stored in the CSV file and the ‘target’ statement is a simple attribution, referencing that task output will be passed on-the-fly to the next workflow task.

```
Etl:
'use' (elements+=Pattern)+ 'on sources{' (source+=Data)+ '}'
('and target{' target=Data '}')?
('with mapping source{' (map+=Data)+'}')?  //keep it or not? Keep.
('options{'opt = Option '}')?;
//The existing pattern categories
Pattern:
Gather | Transform | Load;
//Gather parameters. It includes the data source and the configuration properties
to read the data from the sources.
Gather:
'Gather'('sort by'  sort = STRING ',')?(gatherFields+=Fields)*;
Data:
'BEGIN'(Path | Source)','type=' type=STRING 'END';
(…)
//Fields used to configure data source fields
Path:
'data=' src=STRING;
Source:
'name=' n=STRING ',' 'server=' s=STRING ',' 'database=' db=STRING ','
(…)
Fields:
('source' | 'target')?'fields{' (fd+=Field)+'}';
(…)
```

Fig. 3. An excerpt of the DSL

The BPMN diagram and the DSL are both serialized in a single document that will be analyzed by the system parser that serializes them in a XPDL (*XML Process Definition Language*) file [5]. This file contains information about all the existing connections in the BPMN model and in the configuration of each task. To guarantee system flexibility and avoid the proprietary formats of data migration tools, we used Acceleo to build Ecore models, under the MDA (*Model-Driven Architecture*) provided by Eclipse implementation, for describing each component skeleton we used, identifying all the metadata needed for process instantiation. Next, we built a specific and standard transformation template to encapsulate the conversion logic and convert the tasks internal structure to a XML serialization format supported by Kettle. Finally, the parser groups each construct in layers.

```
use Gather
   on sources{
      BEGIN
        data=CDR_Calls.csv,
        type=CSV
      END}
   and target:ON-THE-FLY
(…)
```

Fig. 4. Example of the DSL instantiation for a Gather task

The main DSL components: *Gather*, *Transformer* and *Loader* will be grouped together in a single Kettle job. If inner transformations associated to each component have only one composite task (such as Gather), then a Kettle transformation task will be generated. However, if several tasks were used (as it happens in the transform phase), a Kettle job is generated (Fig. 5). The loop marker that was used in all the tasks of the model represent a record-by-record processing that was configured for each transformation; the parallel execution of Aggregation tasks is performed using a job, and *the Launch Next Entries in Parallel* property. For each aggregation task data is copied to enable task parallelization, being its output stored in a temporary structure. The Loader task is responsible to accomplish load data from the temporary structure to the target table. The current version of the tool supports several common used data migration tasks, such as: gathers that support data extraction from CSV, XML and relational databases sources, transformers that support several cleaning, integration, surrogate key generation, and slowly changing dimensions procedures, and loaders that support several data loading strategies. Additionally, the tool supports log files management, error handling and data dictionary maintenance tasks. The development of conceptual models was tested with BizAgi and Eclipse BPMN modeler, both of them support different input schemas for BPMN representation.

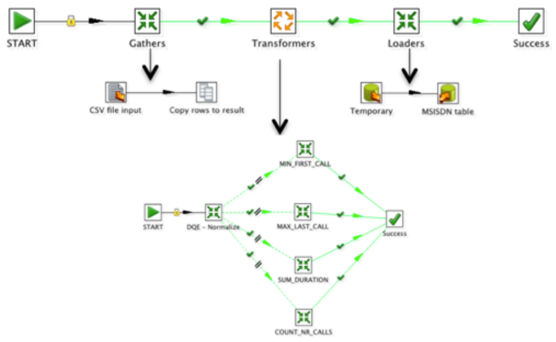


Fig. 5. An excerpt of the Kettle package generated

Acknowledgement

This work was developed under the project RAID B2K - RAID Enterprise Platform / NUP: FCOMP-01-0202-FEDER-038584, a project financed by the Incentive System for Research and Technological Development from the Thematic Operational Program Competitiveness Factors.

References

[1] OMG, O.M.G., Parida, R., Mahapatra, S.: Business Process Model and Notation (BPMN) Version 2.0. 2011.
[2] Kettle, Pentaho Data Integration, 2015. Available at [http://community.pentaho.com/projects/data-integration/]. Accessed on [12 June 2015].
[3] BizAgi, Business Process Management Software Platform, 2015. Available at [http://www.bizagi.com]. Accessed on [12 June 2015].
[4] Xtext, Language Development made easy, 2015. Available at [http://www.eclipse.org/Xtext/]. Accessed on [12 June 2015].
[5] Shapiro, R.M.: XPDL 2.1 - Integrating Process Interchange & BPMN, 2008.
[6] Acceleo, An implementation of the Object Management Group (OMG) MOF Model to Text Language (MTL) standard, 2015. Available at [http://www.eclipse.org/acceleo/]. Accessed on [12 June 2015].