

ETL Standard Processes Modelling

A Novel BPMN Approach

Bruno Oliveira and Orlando Belo

ALGORITMI R&D Centre, University of Minho, Campus de Gualtar, 4710-057, Braga, Portugal

Keywords: Data Warehousing Systems, ETL Conceptual Modelling, Specification, Validation, Analysis and Testing of ETL Systems, BPMN Meta-model Extensions.

Abstract: ETL systems modelling have been a topic quite explored by researchers in Data Warehousing. However, we believe that there isn't yet a convinced and simply approach that provides the necessary bridges to validate conceptual and logical models and testing them before its real implementation. In this work we explore the use of BPMN for ETL conceptual modelling, presenting an extension to the BPMN 2.0 meta-model and notation to support modelling and visualization of ETL activities. We intend to provide a set of BPMN meta-models especially designed to map standard ETL processes, providing the necessary bridges to translate conceptual models into their correspondent implementation testing correctness and effectiveness of its execution. For this particular work, we specially designed a standard ETL process – Change Data Capture based on log files – to demonstrate the viability and effectiveness of the approach presented.

1 INTRODUCTION

Today, the development of an information system is a challenge and often a complexity work engaging personnel with knowledge and expertise in different areas (Scacchi, 2001). Software engineers use to make some initial drafts in order to get a first system overview and a set of preliminary requirements, which need to be validated with all project development team's members and future users. Basically, such drafts (and models) describe the system that we want to implement, regardless of the methodology (or technology) used in its implementation, revealing the most elementary project needs in a very clear and precise way (Losavio et al., 2001). Typically, models are implemented with the use of well-defined languages, with specific syntax, meaning and notation, which are very suitable for computer interpretation (Kleppe et al., 2003).

The design and implementation of an ETL (Extract-Transform-Load) process (Kimball and Caserta, 2004) usually involve the development of very complex tasks imposing high levels of interaction with a vast majority of components of a data warehouse (DW) system architecture. Reusability it's hard to accomplish when developing ETL systems, because each one of them populates

differently a DW, following as expected, a specific set of decision-making requirements and a specific target user community. It's easy to see that different ways of thinking produce also different dimensional models, which implies frequently to change ETL systems design during its life (Weske et al., 2004). Considering this features, as well as the importance and adequacy of an ETL system in the success of a DW, we design and developed a set of meta-models using Business process Model Notation (BPMN) (OMG, 2011) specifications for some of the most commonly ETL processes used in real word applications - we use to refer to them as ETL standard processes. In a previous work (Oliveira and Belo, 2012), we already specified conceptually an ETL standard process: the *surrogate key pipelining*. We started by developing a conceptual definition of the case through the use of BPMN notation, mainly due to its clarity and simplicity, validating it posteriorly by running the model with semantics and constructs provided by BPMN 2.0 specification. However, in this paper, our goal is a little bit different. We intend to provide a set of meta models that can be defined as ETL containers of operations specifying the most current standard ETL tasks (*change data capture, surrogate key pipelining, slowly changing dimensions*, and others), which can be used to create a conceptual model for an ETL

system. For that, we extended BPMN 2.0 standard meta-model and notation in order to represent an extended family of BPMN patterns that allow us to receive standard tasks and to build a complete ETL system more easily.

After a brief presentation of some related work (section 2) about ETL conceptual modelling, we present our proposal extending BPMN 2.0 meta-model in order to represent standard ETL patterns (section 3). Latter, in section 4, we discuss a visual extension to BPMN diagrams, with a concrete example, and their application in a typical ETL process, discussing their features and describing one of the most relevant standard ETL processes (change data capture, based on log files), in order to demonstrate the use and utility of the ETL modelling approach designed and developed (section 5). Finally, we finish the paper presenting results evaluation (section 6), conclusions and future work (section 7).

2 RELATED WORK

The use of models or meta-models (models that specify other models) enables the formalization of concepts, minimizing misinterpretations and ambiguous aspects, enhancing readability and efficiency in the entire implementation cycle of a particular system. Considering ETL system implementation as a very critical task in the success of a DW, several authors proposed different kind of methodologies to support process conceptualization. ETL process specification has been done mainly using proprietary notations imposed by current ETL design and implementation tools.

Vassiliadis and Simitsis (Vassiliadis et al., 2002a); (Simitsis and Vassiliadis 2003); (Vassiliadis et al., 2002b); (Vassiliadis et al. 2003) proposed a new set of elements particularly designed to specify the natural characteristics of an ETL process. In that work, the authors proposed a conceptual, logic and physical modelling for ETL processes that reduces system complexity, decreases implementation time, and increases its reliability. In turn, Trujillo and Luján-Mora (Trujillo and Luján-Mora, 2003) adapted the UML language to fit ETL conceptual modelling requirements. They raised interesting aspects that should be taken into consideration when developing conceptual and logical models for ETL processes. Nevertheless, we consider that still exists a lack of a complete proposal that lets ETL conceptual modelling to be produced easily and generate automatically executable models, providing

source code that could be executed in some platform.

As the basis of this paper, it is the work of Akkaoui and Zimanyi (Akkaoui and Zimanyi, 2009), which proposed an ETL conceptual design using BPMN, and shown how such model could be implemented through the use of Business Process Execution Language (BPEL). These authors considered an ETL system as a special type of process model, and provided a specific set of BPMN components to represent a typical ETL process to correspondent BPMN model process, thought the use of a model-driven framework (Akkaoui et al., 2011). More recently, Akkaoui (Akkaoui et al., 2012) extended his previous works, proposing a BPMN a ETL classification (control and data) objects and a BPMN-based meta-model extension based on a set of components related to the ETL design, such Core meta-model for data process model representation. Other authors explored also the BPMN conceptualization idea for modelling ETL systems. Wilkinson (Wilkinson et al. 2010), for instance, presented a method to guide BPMN specifications in the definition of conceptual models of ETL systems.

Despite the many contributions done so far, we still view a lot of research work to do if we think about validation and operationally of practical implementations of abstract models. We recognize that conceptual specifications and its validation would be essential, in order produce an effective executable instance of the model, providing its conceptualization to non-technical users and its validation by technical users.

3 META-MODEL EXTENSIONS

In this paper it is proposed a BPMN 2.0 extension to define a set of BPMN patterns especially oriented to build, from scratch, a complete ETL system. Each pattern has the ability to represent a specific ETL process that we recognize as standard. We believe that the specification of a conceptual model for ETL standard processes in BPMN is quite appropriate, once it simplifies the implementation process as well increase its own quality of construction, reducing errors, and consequently the ETL project costs. In these cases, the use of the BPMN notation is, in fact, very interesting, showing that it is possible to construct properly conceptual models, which are easy to read and to understand. BPMN also provides the necessary bridges to translate conceptual models to more detailed ones using BPEL or BPMN 2.0

execution semantics. Thus, we can avoid the major drawback that usually exists between modelling abstract models and their consequent implementation, providing conceptual validation through executable process instances.

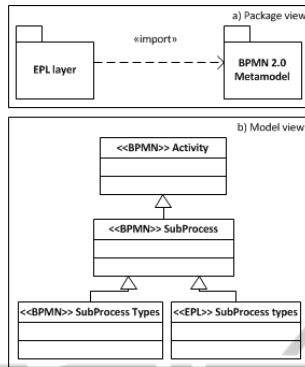


Figure 1: a) A package diagram, and b) The relationship between BPMN standard elements and EPL elements.

Through the extension mechanism of BPMN 2.0 meta-model, it is possible to represent specific concepts related to planning and implementation of an ETL process, and extend these concepts to pre-defined elements of BPMN notation (OMG, 2011). This turns possible to represent specific tasks that enable the construction of ETL processes, by the addition of new elements of a new application domain in the original meta-model. Using UML, it is possible to represent pre-defined elements of the BPMN notation and the elements that characterize the proposed extension, using a simple package diagram (Figure 1a). The package that contains the definition of the extension - *ETL Pattern Layer* (EPL) - depends on the package that represents the BPMN meta-model since it inherits the elements of the original notation. Therefore, through the use of stereotype «import», the basic elements of the BPMN meta-model are used for the representation of the concepts EPL package.

The definition of meta-models representing standard ETL operations comes as an addition to a layer that allows extending the original element: sub process BPMN notation. Using the stereotype «BPMN» to identify unique elements in the BPMN notation, and the stereotype «EPL» to identify elements in the proposed extension, we present in Figure 1b) an excerpt of the meta-model produced to represent the ratio of standard BPMN elements with the elements proposed in the extension. The extension proposed extends the class *SubProcess*, once the meta-models that we intend to define in the ETL process are composed of several operations that

given a particular set of input parameters produce an output data set using a pre-defined set of tasks.

Stropi (Stropi et al., 2011) have identified two factors that limit the development of extensions to BPMN 2.0 meta-model notation, namely the lack of: a) methodologies that support the creation of extensions based approach provided by the language; and b) a graphical notation that allows the representation of extensions. To reduce such lacks, Stropi presented an approach based on a *Model-Driven Architecture* (MDA), enabling the graphical representation of extensions using the UML language, being its structure based on the principles defined by the extension mechanisms of BPMN 2.0 notation, using a specific model to do that: the BPMN-X. Once setting the template extension is possible to transform it into XSD documents allowing their interpretation for the tools that support the BPMN specification. In Figure 2, we can see the conceptual model representation of the extension domain, previously presented in Figure 1, using BPMN+X model construction rules.

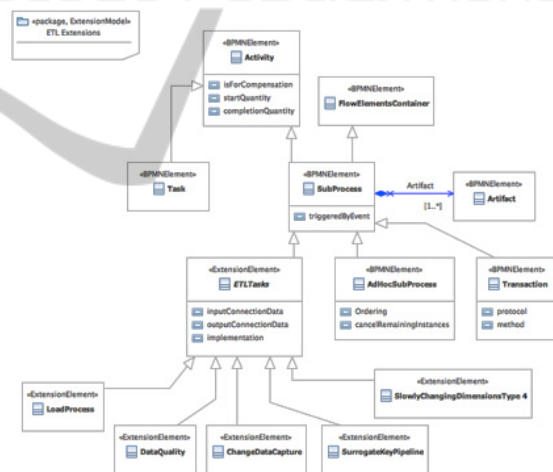


Figure 2: The meta-model extension for ETL conceptual modelling.

The syntax and semantic elements of BPMN+X used are based on the specification of the BPMN extension mechanism. In Figure 2 we see two of them identifying original elements of the BPMN notation: «BPMNElement» and «ExtensionElement». Through the stereotype «BPMNElement» we represent the original classes of the BPMN meta-model that are extended, or that contextualize the extended elements (Figure 2). Each ETL standard process activities is regarded as a new type of *SubProcess*, which given a set of input data produces a result accordingly with a specific predefined set of atomic activities (*Task*). The

SubProcess class inherits from *Activity* and *FlowElementsContainer* super-class, the attributes and associations with other classes (model associations) allowing for the definition of the elements in a diagram, which may be refined in specific processes such as *AdHocProcesses*. Using the stereotype <<ExtensionDefinition>>, we defined a new subclass in the model extension for the *SubProcess* class: the *ETLTasks*. In this subclass were included attributes that let to set all the input parameters related to the access of the data source where the sub process will be applied (*InputConnectionData*), as well as the output parameters where the result of the sub process will be materialized (*OutputConnectionData*). The attribute specifies the technology to be used in the execution of the sub process. In a similar way as OMG done for the class *Service Task*, defined in the original meta-model, we propose web service architecture as the technology to implement it. The *ETLTasks* class is the abstract superclass of the classes that represents only one example of a set of standard sub processes, namely: *LoadProcess*, *DataQuality*, *ChangeDataCapture*, and *Slowly ChangingDimensionsType4*. Each of these subclasses represents a standard ETL model integrating a specific set of elements that is required for the specification of a given task. This set will be available through an inheritance mechanism of the superclass *ETLTasks*.

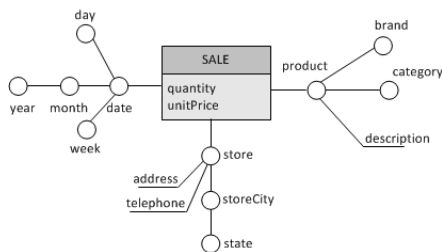


Figure 3: The star-schema Sales.

4 EXTENDING BPMN

Each designed meta-model integrates a set of specific activities representing a standard process that appears usually in most of DW populating processes. Using the BPMN notation for ETL conceptual modelling combined with the meta-model already presented, a user can "drag" a set of specific notation elements as a simple and unique task, which simplifies a lot ETL modelling activities. In order to demonstrate the usefulness of this

approach, we show in Figure 3 an example of a star schema (Kimball and Ross, 2002) modelled conceptually using the notation proposed by Golfarelli (Golfarelli and Rizzi, 2009). This dimensional model has three distinct dimensions tables and a fact table including two measures: quantity and selling price of a product - a very simple schema indeed, but even so it allows us to analyse the sales of products in a particular set of stores over time. Once specified the DW's storage structure it's necessary to define the initial ETL process, adjusting it in order to keep the repository updated properly. An incorrect or incomplete specification of ETL activities will affect seriously the quality of data stored in the DW, influencing negatively the quality of subsequently decisions.


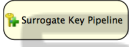
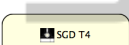
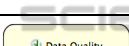
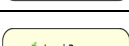


Figure 4: Sample population process for Sales schema.

Considering a high level view of the ETL process, it's possible to specify the entire process simply instantiating classes that represent sub processes constituted by atomic tasks. Using the notation presented in Table 1, which represents the instantiation of each subclass of the class *ETLTasks* of the BPMN meta-model, we present in Figure 4 a regular ETL process that can be applied for populating the star-schema of Figure 3. After starting the process, is used a divergence parallel gateway that initializes (in parallel) the *change data capture* tasks for each of the data sources, analysing available log files, identifying new records, updated records, or deleted records. Next, in our case, for source 1, we also specified the procedure for history preservation, related to the dimensional attributes of dimension tables classified as *slowly changing dimensions with history maintenance*. The sub process: *slowly changing dimensions type 4* (SCD T4) stores current data in two different tables: a dimension table, which have the most up-to-date information, and a history table that keeps track of all the changes occurred in the dimension table. In our example, we consider that SCD T4 process also applies surrogate key assignments in order to obtain dimensional keys for each dimension. In both sources is necessary to identify common records extracted from sources and generate surrogate keys in order to maintain the independence of the data used. The *surrogate key pipeline* sub process assigns surrogate keys (unique identifiers) in order to identify data extracted from a specific source, using

as support several mapping tables. Once concluded the surrogate keys assignment process, it is defined a convergence gateway that starts the *Data Quality* analysis process upon completion of the two source's streams. The *Data Quality* sub process is responsible for insure data integrity and consistency through the use of specific procedures. Finally, the *load process* loads data to the target repository, establishing the necessary correspondences between temporary storage structures and destination storage structures.

Table 1: The new BPMN extension's elements.

Element	Description
 Change Data Capture	This sub process identifies the changes made in a given structures that is responsible to feed a specific dimension of analysis.
 Surrogate Key Pipeline	The <i>surrogate key pipeline</i> sub process assigns surrogate keys (unique identifiers) to identify the data that comes from a specific source, using appropriated mapping tables.
 SGD T4	The <i>slowly changing dimensions type 4</i> sub process stores current data in two different tables: a dimension table, which have the most updated information, and a history table for keeping a track of all the changes occurred in the dimension table.
 Data Quality	This sub process applies a set of procedures, which are responsible to check integrity and consistency of data before being loaded to the DW.
 Load Process	The sub process: <i>load process</i> loads data to target repository, establishing the necessary correspondences between temporary storage structures and destination storage structures.

Although the ETL process presented being limited to the set of standard processes considered in the generated meta-model (mainly for simplicity), we can verify that ETL process modelling provides a more abstract level for users, encapsulating a wide range of elementary operations that are integrated in model activities (Figure 4). Thus, the construction of the ETL workflows simplifies process implementation, and reduces implementation errors, because we used pre-established mapping procedures for model conversion into to execution primitives. Following the general guideline of Oliveira and Belo (2012), we describe in the next section the process of specification of capturing data in data sources, a process that is commonly known as *change data capture* (CDC).

5 THE CDC SUB PROCESS

A DW must be a subject, non-volatile and time-variant repository Inmon (2005), i.e., once entered in the DW data shouldn't change in any form and sources' data should be maintained as when gathered in order to capture business requirements changes and discover business trends. Thus, an ETL process

must be capable to detect changes in the sources during the occurrence of business operations. CDC methods are used to detect changes on sources' data that occurred since the last extraction, capturing those changes to reflect latter in the DW. CDC techniques are one of the most difficult tasks to reflect and represent in ETL design. However, there are several possible methods to implement a CDC process. From using intensive comparison data processes between sources' and DW's data to the use of triggering mechanisms over source tables, controlling insert, update and delete events, there are a lot of different ways to capture data changes. One of the most used currently, provably due to its non-intrusive aspects, it's the use of DBMS systems' transaction logs, which keep records about all modifications made on operational systems. Thus, if we scan and process the contents of a transaction log, it is possible to identify changes that occurred during some period in an operational system database.

This way of doing does not affect transactional data and does not imply the implementation of additional mechanisms to transactional systems – it's a non-intrusive CDC technique. Nevertheless, additional efforts need to be made in order to develop an application that analyses the log file and identify data modifications occurred. Furthermore, each DBMS implements a specific transaction log structure, which increases complexity of implementation. In this work, we used a transaction log (Table 2) of a Microsoft SQL Server 2008 to demonstrate the application of a high-level model, providing a formal specification and standardization of a CDC standard ETL process. In there, LSN (Log Sequence Number) represents a unique identifier for each record inside transaction log that we only represent by a simple integer value. Transaction ID (T.ID) identifies modifications that belong to the same Transaction ID. So we know if they are related or not to each other. Each transaction begins with LOP_BEGIN_XACT operation and ends with LOP_COMMIT_XACT. Between these two rows can exists many LOP_MODIFY_ROW operations, representing a specific type of modification. Each LOP_BEGIN_XACT represents a specific type of transaction described in transaction name column: INSERT, UPDATE or DELETE. All this needs to be decoded in order to interpret and analyse data in a more readable way. Using the BPMN notation to represent all data extraction steps from log based CDC process, we introduce in Figure 5 a standard meta-model for general ETL use. Generally, a CDC process begins loading the required data parameters

necessary in subsequence steps, such as the server name and the source of the table data where we want to capture data changes, *data staging area* (DSA) connection data, the name of the audit table that will store all extracted records and attributes with variation, which need to be captured (Load Properties activity). Next, transaction log must be read based on the previous LSN read (if we have it). That is, instead read all records from the log file, process reads transaction log based on previous LSN.

Table 2: An excerpt of a Microsoft SQL Server 2008 transaction log.

C. LSN	T.I D	Operation	Transaction Name	Alloc Unit Name	RowLog Contents
1	1	LOP_BEGIN_XACT	INSERT	NULL	NULL
2	1	LOP_INSERT_ROWS	NULL	dbo.produc t	0x10...
3	1	LOP_COMMIT_XACT	NULL	NULL	NULL
4	2	LOP_BEGIN_XACT	UPDATE	NULL	NULL
5	2	LOP_MODIFY_ROW	NULL	dbo.produc t	0x63...
6	2	LOP_MODIFY_ROW	NULL	dbo.produc t	0x63...
7	2	LOP_COMMIT_XACT	NULL	NULL	NULL
...

The BPMN model of Figure 5 represents one expanded sub process and a collapsed sub process that was zoomed (due to space limitations) in order to show all their contents. Iteratively, expanded sub process handle all the transactions contained in the transaction log. Through *Read Transaction* activity, each transaction is passed to the *Process* transaction sub process, in order to read all operations inside transaction. The *Process* transaction sub process identifies the type of transaction received: *Insert*, *Update* and *Delete*, and for each of them represents loop structure that read all records and extracts contents to be decoded in *Decode contents* activity. After processing all transactions from transaction log, for each transaction changed rows are inserted into audit table for ETL processing. When all transactions are processed, it is necessary to preserve last LSN read for further CDC processes.

After the initial specification of a high-level model, we develop the necessary bridges to translate it to a more detailed one, providing its execution through BPMN 2.0 specific constructs. In order to carry out an appropriated mapping between the high-level model and the executable model we need to ensure some characteristics related to the execution of tasks. For that, we used some orchestration elements provided by BPMN 2.0 for services that

allow the coordination of multiple tasks. In order to turn the model of Figure 5 in an executable process, we use BPMN specifications and a service-oriented approach, which allowed us invoking methods through a set of services provided by web services. Once coordinated, these services allow the application of the necessary transformations for the implementation of the CDC process through the use of an intermediate layer of stored procedures. These ensure services interaction with all transaction log structures that need to be accessed.

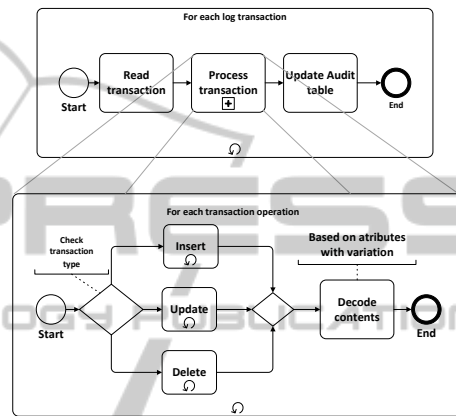


Figure 5: A BPMN pattern for the transaction log file CDC.

Beyond the specification of all activities represented in the conceptual model, the definition of the executable model also requires an additional specification of tasks because it will has the right characteristics to support its own implementation. For example, activities that communicate with a web service must be defined as service tasks. This type of activity is performed by the system without human intervention, and it is used regularly to maintain communication with external interfaces.

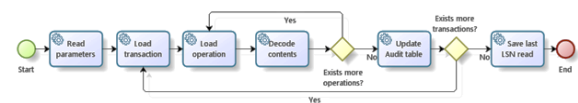


Figure 6: The executable CDC BPMN model.

In Figure 6 it is presented a proposal to convert the model of the CDC process (Figure 5) to BPMN 2.0 that can be executed using BizAgi. The implementation of the model is based on a data storage structure especially conceived for this process with a SOA (*Service Oriented Architecture*) layer, which acts as a bridge between the model and the BPMN storage structure. All *Service* tasks have a web service method associated. The methods are

responsible to invoke the most appropriated stored procedure in order to perform its related activity over log file or audit table. The stored procedures were created using Dynamic SQL and provide a readable view to transaction log. The process begins reading configuration parameters of the implementation, which involves usually access and operational data, such as the server connection, access credentials, processing data, the name of the source tables to process and the target audit table. All this metadata can be filled in the BizAgi configuration environment using specific constructed forms. The activities: Insert, Update and Delete from the collapsed sub process transaction (Figure 5), can be omitted in the execution model specification, since *Load* transaction task retrieves records to the BPMN runtime environment and identifies the respective performed operation.

The interaction with the log file and the records of the audit table is coordinated by the BPMN process that runs using the methods implemented in the web service layer. For each operation contained in transactions, *Load* operation task extracts contents from records through the use of specifics Web services. Next, *RowContents* columns need to be decoded using *Decode Contents* task that invokes another web service to decode data. Once decoded, data are inserted into an audit table. The looping structure was implemented by means of two gateways, which operate as two conventional structured cycles. These looping structures are modelled using gateways with transitions, which allow repeating a set of predefined activities accordingly to some particular condition. Finally, LSN read is stored by *Save last LSN read* task into DSA, in order to use it for further executions. BizAgi provides a set of structures especially oriented to support BPMN models execution. After model definition it is possible to construct a special oriented data structure to receive the information used in all the activities included in the model, as simple input data or as support business rules. In any model's structure, is possible to define entities and relationships as we usually do in a relational model. We use these structures to store temporary transaction records, in order to process them (e.g. sending them to another task) one by one and put them in respective audit table.

6 ANALYSIS AND EVALUATION

The use of meta-models for formal specification of

ETL standard tasks defines a level of abstraction that allows simplifying and enhancing the quality of any ETL design and implementation process. With the BPMN meta-model extension, we propose the inclusion of a set of sub process packages, which have a set of predefined atomic operations that allow a high level conceptual representation and specification of ETL processes, simplifying and accelerating their design and implementation processes, while reducing complexity and minimizing construction errors. Meta-models definition allows consolidating and parameterizing of the ETL development, which have, as we know, a strong impact on the success or failure in the life cycle of any data warehousing system.

Currently, tools that support the creation and maintenance of ETL processes, based on proprietary notations and methodologies, clearly aimed at technical users who have to use other mechanisms in order to communicate with "non-technical" users. Using the constructs and the semantic richness offered by BPMN notation, it is possible conceptually to define a new set ETL processes, which simplifies the communication between users and ETL designers that use to be responsible for managing business processes. The conversion of a conceptual model in an execution model is possible using the constructors and semantics implemented in BPMN 2.0 meta- model, such as a structure based on web services that enables the implementation of the tasks set out in the conceptual model. Each meta-model may be considered as a black box, which based on the specification of a set of parameters, that is performed by set of operations and consequently produce a output result. Through the use of DBMS transaction log file, the CDC process is able to capture relevant data from sources in order to load it to target DW. The interpretation of transaction log file allows a very simple way to identify changes in data sources, hiding to end-users the inherent complexity of such processes. As a consequence, they allow the simplification and optimization of this kind of activities in the planning and execution of any ETL process.

7 CONCLUSIONS

Akkaoui and Zimanyi (Akkaoui and Zimanyi, 2009) assumed that an ETL process can be seen as a particular case of a business process, and proposed the use of generic processes platform independent for running ETL processes (Akkaoui et al., 2011). The clear and simple notation of BPMN 2.0

facilitates a lot understanding and communication among users and offers an easy discussion platform to technical users who are responsible for implementing the system (Akkaoui and Zimanyi 2009). Specifying a new set of meta-models, like the ones presented in this paper, including new types of sub processes that represent specific tasks typically used in the implementation of ETL processes - *slowly changing dimensions, surrogate key pipelining, data quality assurance processes, or change data capture*, among others – it is helpful in any ETL specification and implementation process. Each of these models, consists of a finite set of pre-established atomic activities, which contributes significantly to improving the quality of construction, and reduce operational errors, resulting in a significant reduction of implementation time and costs. The meta-models referred and presented here represent only a subset of the models we intend to implement in a near future, in order to demonstrate effectiveness of the BPMN 2.0 meta-model extension presented and discussed in this paper.

REFERENCES

- Akkaoui, Z. El et al., 2011. A model-driven framework for ETL process development. In *DOLAP '11 Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*. pp. 45–52.
- Akkaoui, Z. El et al., 2012. BPMN-Based Conceptual Modeling of ETL Processes. *Data Warehousing and Knowledge Discovery Lecture Notes in Computer Science*, 7448, pp.1–14.
- Akkaoui, Z. El & Zimanyi, E., 2009. Defining ETL workflows using BPMN and BPEL. In *DOLAP '09 Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*. pp. 41–48.
- Golfarelli, M. & Rizzi, S., 2009. *Data Warehouse Design: Modern Principles and Methodologies*, McGraw-Hill. Available at: <http://books.google.pt/books?id=R7qqNwAACAAJ>
- Inmon, W. H., 2005. *Building the data warehouse*, Wiley. Available at: <http://books.google.pt/books?id=EeVQAAAAMAAJ>.
- Kimball, R & Ross, M., 2002. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, Wiley. Available at: <http://books.google.pt/books?id=2OCbq8Azdm8C>.
- Kimball, Ralph & Caserta, J., 2004. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*,
- Kleppe, A. G., Warmer, J.B. & Bast, W., 2003. *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley Professional.
- Losavio, F., Chirinos, L. & Pérez, M.A., 2001. Quality Models to Design Software Architectures. In *TOOLS '01 Proceedings of the Technology of Object-Oriented Languages and Systems*.
- Oliveira, B. & Belo, O., 2012. BPMN Patterns for ETL Conceptual Modelling and Validation. In *20th International Symposium on Methodologies for Intelligent Systems*.
- OMG, 2011. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0. In Documents Associated With Business Process Model And Notation (BPMN) Version 2.0.
- Scacchi, W., 2001. Process Models in Software Engineering. In J.J. Marciniak (ed.), *Encyclopedia of Software Engineering*, 2 nd Edition, John Wiley and Sons, Inc.
- Simitsis, A. & Vassiliadis, P., 2003. A Methodology for the Conceptual Modeling of ETL Processes. In *The 15th Conference on Advanced Information Systems Engineering (CAiSE '03)*. pp. pp. 305–316.
- Stroppi, L. J. R., Chiotti, O. & Villarreal, P.D., 2011. Extending BPMN 2.0: Method and Tool Support. *Lecture Notes in Business Information Processing*, 95(59-73).
- Trujillo & Luján-Mora, S., 2003. A UML Based Approach for Modeling ETL Processes in Data Warehouses. *Conceptual Modeling - ER 2003 - Lecture Notes in Computer Science*, 2813, pp.307–320.
- Vassiliadis, P. et al., 2003. A framework for the design of ETL scenarios. In *Proceedings of the 15th international conference on Advanced information systems engineering*. Berlin, Heidelberg: Springer-Verlag, pp. 520–535. Available at: <http://dl.acm.org/citation.cfm?id=1758398.1758445>
- Vassiliadis, P., Simitsis, A. & Skiadopoulou, S., 2002a. Conceptual modeling for ETL processes. In *DOLAP '02 Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*. pp. 14–21.
- Vassiliadis, P., Simitsis, A. & Skiadopoulou, S., 2002b. On the Logical Modeling of ETL Processes. In *International Conference on Advanced Information Systems Engineering (CAiSE)*. pp. 782–786.
- Weske, M., Aalst, W. M. P. van der & Verbeek, H.M.W., 2004. *Advances in business process management. Data & Knowledge Engineering* 50, 50(1–8).
- Wilkinson, K. et al., 2010. *Leveraging Business Process Models for ETL Design*. *Lecture Notes in Computer Science*, 6412/2010(15-30).