Pattern-Based ETL Conceptual Modelling

Bruno Oliveira[1], Vasco Santos[2], and Orlando Belo[1]

[1] ALGORITMI R&D Centre, University of Minho, Braga, Portugal
id4103@alunos.uminho.pt, obelo@di.uminho.pt
[2] CIICESI, School of Management and Technology, Porto Polytechnic, Felgueiras, Portugal
vsantos@estgf.ipp.pt

Abstract. In software development, patterns and standards are two important things that contribute strongly to the success of any system implementation. Characteristics like these ones improve a lot systems communication and data interchange across different computational platforms, integrating processes and data flows in an easy way. In ETL systems, the change of business requirements is a very serious problem leading frequently to reengineer existing populating processes implementations in order to receive new data structures or tasks not defined previously. Every time this happens, existing ETL processes must be changed in order to accommodate new business requirements. Furthermore, ETL modelling and planning suffers from a lack of mature methodology and notation to represent ETL processes in a uniform way across all implementation process, providing means to validate, reduce implementation errors, and improve communication among users with different knowledge in the field. In this paper, we used the BPMN modelling language for ETL conceptual modelling, providing formal specifications for workflow orchestration and data process transformations. We provide a new layer of abstraction that is based on a set of patterns expressed in BPMN for ETL conceptual modelling. These patterns or meta-models represent the most common used tasks in real world ETL systems.

Keywords: Data Warehousing System, ETL Systems, ETL Conceptual Modelling, Verification, and Validation, ETL Patterns and Meta-models, and BPMN.

1 Introduction

In software development, the use of standard solutions facilitates a lot the development of complex systems. In real world applications, commercial tools use them to decompose solutions based on common software patterns that define their behaviour as well as ensure communication with other patterns/systems. Patterns represent templates or sets of rules that have the ability to specify how a particular problem can be solved, considering particular applications scenarios framed in different real-world contexts. These patterns allow for reusability and improve the general quality of the system, reducing potential negative impacts or incorrect software development processes. Thus, the use of software patterns improves code reusability, minimize the impact of requirements changing, and consequently reduce development costs.

A. Cuzzocrea and S. Maabout (Eds.): MEDI 2013, LNCS 8216, pp. 237–248, 2013.

ETL (Extract-Load-Transform) systems are a very particular type of software that is considered a critical component in any data warehousing system implementation. Design and implement an ETL system imply the specification of a set of construction rules related to specific decision-making processes, which are typically very volatile in terms of base requirements, since business requisites frequently change to cover and accommodate additional operational areas, leading to an increasing business process complexity. Even with the use of standard solutions in data warehousing systems implementation, it's always necessary to provide specific decision-making processes reflecting especially business analysis metrics of agents involved. This happens simple because different people having different ways of acting and different ways of thinking, which are linked, in some way, to a specific set of particular decision-making processes [1]. Moreover, operational systems have specific data schemas supporting very different operational business requirements. The definition of specific data extraction procedures from different data sources structures will be also considered, which may represent an additional effort because extraction mechanisms can be limited by obsolete technology or extraction mechanisms that still are used. All this, make ETL components reuse a difficult task to achieve [1]. Thus, it is easy to understand Kimball, when he stated that the implementation process of a ETL system consumes usually about 70% of the resources required to implement a data warehouse [2]. Moreover, it is known that the success of a data warehousing system depends heavily on the adequacy of its populating system, which imposes an extreme care and concern by the ETL development team in its planning, design, architectures, and implementation.

Proprietary tools generally support ETL modelling and implementation tasks, providing their own methodologies and notations. The use of such tools complicates ETL implementation and maintenance, since it imposes additional efforts for the ETL team that needs to understand their own specificities, and not standard tasks and mechanisms. Proprietary notations also limit communication with non-technical users due to the fact that the models produced are generally very detailed with issues related especially to the runtime environment of the tool. Moreover, if we need to change the ETL supporting tool, the effort to do that will be considerable. It will be need to build the ETL process from scratch (or almost). With these problems in mind, several authors have made relevant efforts to provide more standard methodologies and notations for ETL modelling, which are generally based on new notations or with some meta-model extensions of existing standard notations. However, we consider that there is still a lack of a complete methodology covering the whole ETL development cycle, providing an easy and understandable notation for all user profiles, allowing its mapping for a more detailed model for execution, and providing its validation before proceeding to a final implementation of the ETL system.

This paper presents another approach to ETL conceptual modelling that help to alleviate the negative effects of a less suitable planning for the development of an ETL system, providing a very practical way for ETL conceptual modelling based on a set of patterns that were built using *Business Process Model Notation* (BPMN) [3]. These patterns represent and characterize some of the most common tasks used on real world ETL processes, providing the necessary steps to implement a specific procedure that can be applied in many different application scenarios. Each pattern

acts like a black box that receives some input parameters, and produces the respective output using a set of pre-defined operations.

The choice of BPMN notation for pattern representation was mainly due to its simplicity representing and modelling business processes, coupled with its power of expressiveness, which makes BPMN notation quite easy to apply in ETL systems application contexts [4–6]. Thus, it's possible to provide very descriptive models through the use of BPMN 2.0 orchestration elements that can be mapped posteriorly into execution primitives, providing a straightforward design process into execution primitives. After a brief exposure of some related work (Section 2), we present in section 3 a pattern-based meta-model representation; based on some of the most ETL common used tasks. Next, in section 4, we present a *Data Quality Enforcement* (DQE) pattern, proposing its control elements using XPDL (XML Process Definition Language) and its data process specified in *Relation Algebra* (RA). Finally, in section 5, we present conclusions pointing out some future research lines.

2 Related Work

Using BPMN for ETL modelling has been explored by Akkaoui since 2009 [4]. In their former work, Akkaoui and Zimányi explored first the use of BPMN for ETL modelling, showing some examples how BPMN, commonly used for business processes implementation, can be used for more specific processes like ETL processes. These authors considered that existing organizational business processes must be integrated with ETL processes, since business processes can help not only to identify key data but also to identify the appropriated opportunity-window to load it into the data warehouse. They also explored how conceptual models could be implemented through the use of BPEL (*Business Process Execution Language*), showing that BPMN notation can be successfully applied to ETL processes, providing an easy way to understand mechanisms that already are used by many organizations and known by several types of users. Later, Akkaoui [5] provided a BPMN-based meta-model for an independent ETL modelling approach, exploring also the necessary bridges for a model-to-code translation, providing its execution model for some commercial tools. More recently, Akkaoui [6] proposed a more complete BPMN-based meta-model for ETL modelling based on control process mechanisms and data process operations. The first one provides process orchestration, which can be accomplished by several BPMN elements such as events and flow control gateways; the second is related to specific operations allowing for the manipulation of data among several data sources that may be coordinated by control process elements. Based on these two perspectives and on an ETL classification tree presented in the same work, the authors provided a specific meta-model covering ETL control mechanisms and data operations. Based on Akkaoui former work, we already presented [7] a pattern oriented approach for ETL conceptual modelling. Despite basing our work on the ideas originally presented by Akkaoui et al. [4–6], many other important initiatives have been taken in this field highlighting important aspects that should be considered in any ETL process modelling. For example, the work of Simitisis and Vassiliadis presented a methodology [8–11] covering the main phases of ETL modelling, namely: conceptual, logical and physical. More recently, El-Sappagh

et al. [12] proposed the EMD (Entity-Mapping Diagram) framework that provide a meta-model and notation for conceptual modelling of ETL processes. In turn, Trujillo and Luján-Mora revealed an UML approach [13] proposing an extension of the notation elements through the use of a new set of constructs representing specific activities commonly used on ETL conceptual modelling. Despite the many contributions already made in this field, we believe that still is an absence in the field of a complete proposal that will permit the conceptual modelling of an ETL system, as an initial work and discussion of its main features, allowing posteriorly the generation of the corresponding logical model and the generation of a physical model with the possibility of being executed. We intend to do that, using a set of patterns, composed by a specific flow of activities, representing the most common and error prone tasks on ETL systems design and implementation.

3 A BPMN 2.0 Meta-model Extension for ETL Modelling

Using BPMN notation for ETL specification is quite interesting. It provides a useful and expressive notation for the ETL conceptualization stage, as well as supplies several orchestration mechanisms that allow for their subsequent instantiation in execution primitives. With BPMN it's possible to represent more abstract processes, hiding some specific implementation details of their practical implementation, which turns possible to map them into new models associated to some specific execution architectures. Therefore, in this paper we propose a pattern-oriented approach with the ability to represent the most common tasks used in ETL systems implementation. We formalized these patterns through the use of a set of BPMN meta-models, which are characterized using a set of pre-established activities and its correspondent data flows. These meta-models can be used for ETL conceptual modelling, contributing to improve the quality of the process, reducing potential design errors as well as the costs associated with the whole process implementation. With these purposes in mind, we propose a BPMN 2.0 meta-model extension in order to represent a family of meta-models that can be instantiated and implemented by a tool that supports BPMN specification. Through this, we can add a new layer to the existing BPMN meta-model, composed by a set of pre-defined ETL patterns - based on a specific set of input parameters representing a particular application scenario, each pattern produces the corresponding output data set based accordingly to its control and data process specification. Following the BPMN-X approach [14] proposed for BPMN extension model representation, in Fig. 1 it's presented an excerpt of the relationship between BPMN base elements and the ETL Tasks class of the ETL pattern layer extension that we are proposing here.

The BPMN-X UML-based approach complies with the base BPMN rules for meta-model extension [3], guaranteeing that original BPMN elements are not modified and ensuring compatibility with existing BPMN supporting tools. In a BPMN meta-model each activity can be characterized as atomic (*Task*) or non-atomic (*SubProcess*). The *SubProcess* class inherits the attributes and associations with other classes (model associations) from *Activity* and *FlowElementsContainer* superclass. Our proposal extends this last class, once meta-models we going to define for ETL process specification are composed by several operations and pre-established data flows. The model presented in Fig. 1 uses two stereotypes allowing for the identification of

original elements (<<*BPMNElement*>>) and extended elements (<<*ExtensionElement*>>). Using the <<*ExtensionElement*>> stereotype, we present the ETLTasks superclass, which is also a subclass of the *SubProcess* class. Detailing our *ETLTasks* superclass, we present in Fig. 2 the pattern-based ETL meta-model we propose, using the Ecore model provided by EMF framework from Eclipse[1], in order to represent all concepts related to the specification of each ETL pattern.

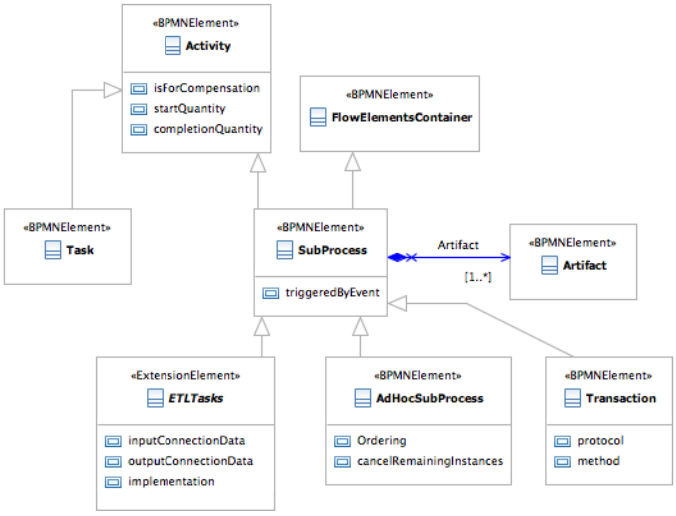

Fig. 1. The BPMN meta-model extension for ETL modelling

The meta-model of Fig. 2 presents several concepts related to pattern control elements and pattern data manipulation definition. In there, we included the *ConnectionParameters* class, which represents specific data that is used for input/output execution parameters and the *ConnectionMapping* class to specify data mappings about input parameters for data sources access and output parameters, where execution results from each pattern are stored. The *ETLTasks* class is the abstract superclass of the meta-model's classes representing here only a small example of a possible set of standard sub processes that can be included in the proposed meta-model, namely: *Load* – that represents load procedures for populating a data warehouse; DQE – which includes replace, unique, attribute decomposition and text transformation procedures; CDCLogFile – that applies for change data capture procedure based on a log file to capture relevant data that was changed on operational information systems; *SCDHistoryPreservation* – which represents a slowly changing dimension procedure that keeps records changing history in an auxiliary table; and Surrogate key procedures for a dimensional surrogate key generation process (*DimensionalSurrogateKey* class) and a surrogate key pipeline process (*SurrogateKeyPipeline* class). Each of these subclasses represents a standard ETL pattern integrating a specific set of pre-established tasks and data flow controls required for pattern definition and specification. In one of our previous articles [7] we

[1] `http://www.eclipse.org`

already presented a surrogate key pipeline pattern specification, and the necessary guidelines for its representation in a set of execution primitives. In the next section, we will describe some of most common DQE procedures, presenting its control and data process formal specification.

4 A BPMN Meta-model for Data Quality Validation

Data quality validation procedures are used to detect, remove errors and inconsistencies from data gathered on information sources. Given the nature of a data warehouse, it's critical to have correct data to feed it. This uniformity and cleaning procedures must be applied properly in some ETL's stage, in order to not affect the quality of decision-making processes. Data integration is highly influenced by the number of operational sources that participate in the ETL process as data providers, and by its schemas and inconsistencies that may exist. For that reason, specific procedures [15] must be used in all ETL transformation or cleaning steps, which allow for the specification of adequate strategies in order to eliminate problems related to data integration. Simpler tasks are typically used, such as: attribute decomposition; content attribute normalization; or duplicate elimination, as well as more complex processes that involve query ETL meta-data (mapping tables). These data dictionary tables are used for ETL support, storing necessary equivalences to support normalization and correction procedures. In order to specify these common procedures, in Fig. 3 we present the corresponding BPMN meta-model corresponding to a set of specific DQE procedures. This meta-model represents a general application of all data quality procedures presented previously in Fig. 2. However, for particular scenarios, this process can be simplified for including only a few of these procedures.

The DQE process starts by reading all necessary parameters that support the execution process: the audit table connection data in order to load and process target records, the mapping tables that contain the meta-data for ETL support, the target attributes for each table, the correspondence between all the attributes involved with, i.e. from audit table and data dictionary tables, the list of functions that will be applied and its respective attribute mappings, and the connection data of the target repository where records that were cleaned or conformed will be stored. Next, a loop sub-process (Fig. 3) starts applying specific data quality procedures. For this sub-process, two types of functions cab be identified: *pre-functions*, which are applied to a record prior the data replacement procedures; and *post-functions*, which are applied after the application of replacement procedures. Both type of functions represent simple procedures that must be applied to operational data, such as attribute decomposition and text transformations procedures. On the other way, post-functions, like duplicate elimination procedures, are only applied when replacing procedures are finished.

The replacement procedures are performed whenever it is necessary to standardize or correct (when necessary) data according to a given set of mapping rules stored in specific mapping tables. Thus, it's necessary to search these tables every time we need to get a replacement value. For a giver attribute, if no correspondences are found the process will continue following its specifications, i.e., data can be loaded to quarantine tables in order to be analysed latter, and approved automatically according to a set of specific rules when possible.

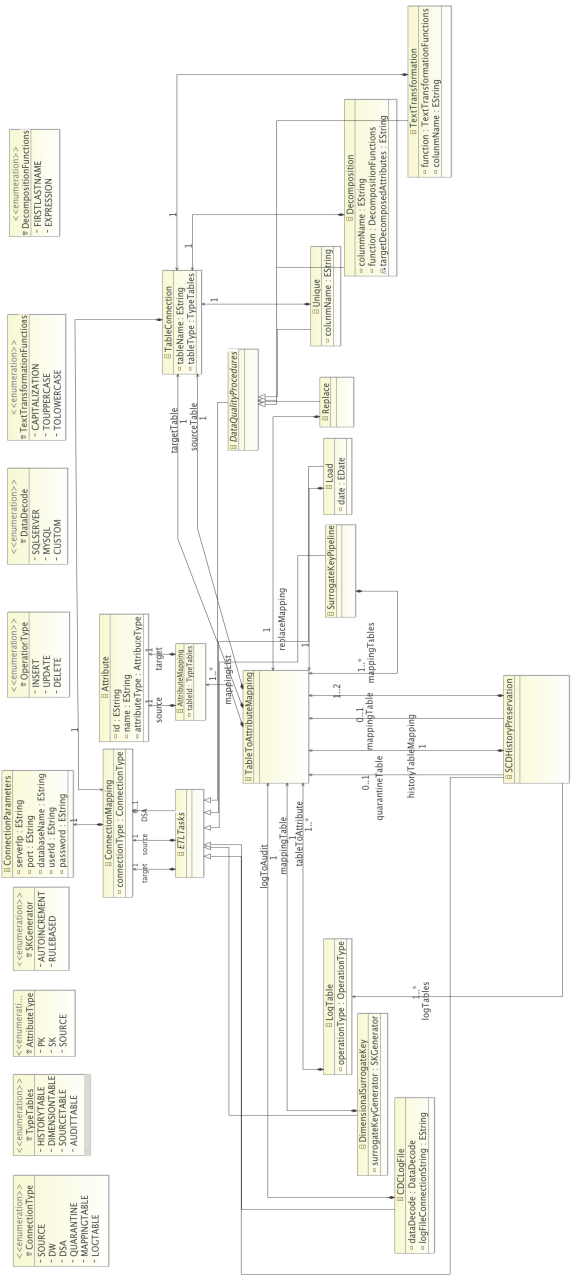

Fig. 2. The BPMN extension for a set of standard ETL patterns

The ETL conceptual model presented in Fig. 3 represents a single record for each sub process loop iteration. However, records are not necessarily pushed from external application one by one each time. The flows can be adjusted or reorganized according

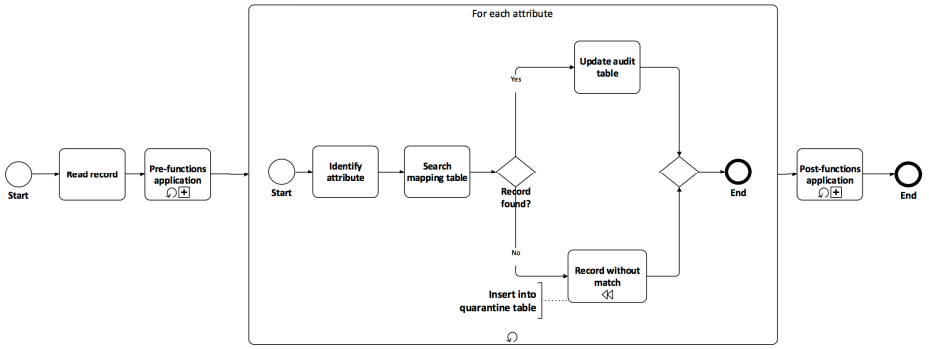

Fig. 3. A general BPMN conceptual model for a DQE procedure

to its physical implementation. However, in conceptual representation it is not needed to represent specific characteristics related to its implementation.

4.1 Workflow Control Specification

When modelling ETL processes we need to express the different control flows and data transformations between all the tasks that are embedded in them. The use of BPMN for ETL workflows, allows for a rich graphical representation of all the control flows established between activities we want to implement, as well as their own characterization and description. However, when defining a specific BPMN workflow as a standard pattern for ETL modelling, we must enforce some rules to be applied for the orchestration of flows and to allow the integration with some commercial tools. For that we selected XPDL (*XML Process Definition Language*) [16] specification, a XML based language, which can be used to formalize the process activities. Thus, we can specify how activities are executed and what is its execution sequence. Additionally, with XPDL specification we can also assure compatibility between BPMN workflow engines, which support the BPMN meta-model extension presented in Fig. 2. We must also consider that not all activities specified in a DQE pattern can be used in a real world scenario, simple because each ETL system has its own requirements. In a real process implementation, the XPDL file will be generated according to a particular pattern configuration, based on a specific application scenario.

4.2 Data Process Specification

With XDPL format we are able to formalize workflow orchestration elements and their interchanges across several BPMN-based tools. However, for each ETL workflow process we needed to specify a set of data transformations associated to the correspondent workflow tasks. Which means that BPMN allows for the specification of a number of steps (and its sequence) in order to achieve a specified goal. Data transformation procedures specify how data from operational sources will be

transformed in order to complete each workflow task. Thus, we specified and modelled all transformations associated with ETL model's tasks using *Relational Algebra* (RA) [17–19], providing thus all formal transformations needed for each DQE procedure used. All transformations are presented based on the same base audit table structure. Nevertheless, transformations can be applied sequentially. In RA an audit table can be represented using a general schema for data representation such as:

$$\boldsymbol{auditData} = \langle \boldsymbol{Att_1}, \ldots, \boldsymbol{Att_n} \rangle \tag{1}$$

In Eq. 1 we can see a representation of *auditData* identifying the source audit table used on DQE procedures, and a set of attributes $Att_1,\ldots,Att_n$ that composes the audit table. As described before, we categorize DQE procedures in **pre-functions**, attribute decomposition and text transformation procedures; **replacement procedures**, for data correction or data normalization; and **post-functions**, for the elimination of duplicated records. For the decomposition of attributes, we defined the following RA expression:

$$\boldsymbol{auditData\_d} \leftarrow \boldsymbol{\varepsilon}_{[\boldsymbol{Att\_d_1}=\boldsymbol{subs}(\boldsymbol{Att_n},\boldsymbol{1}),\ldots,\boldsymbol{Att\_d_n}=\boldsymbol{subs}(\boldsymbol{Att_n},\boldsymbol{n})]}(\boldsymbol{auditData}) \tag{2}$$

in which *subs* is a user-defined function that extracts a set of characters from an attribute. In this function, the first argument (Att_n) is the attribute to be parsed and the second on (*1..n*) defines which set of characters will be extracted (first, second, etc.). Lastly, the extended RA operator: ε [20] creates a new attribute based on data that is in other attributes, storing the result set in the *auditData_d* table. For text transformation procedures, such as upper, lowercase or capitalize conversion, we starting creating a new attribute based on the attribute to be transformed, for instance Att_n, and applying the respective transformation (*userFunction*) (Eq. 3). Then we remove the old Att_n attribute from table (Eq. 4) preserving the new one. Finally, we need to rename the new attribute to the old name restoring the original structure of the table:

$$\boldsymbol{Temp1} \leftarrow \boldsymbol{\varepsilon}_{[\boldsymbol{Att_y}=[\boldsymbol{userFunction}](\boldsymbol{Att_n})]}(\boldsymbol{auditData}) \tag{3}$$

$$\boldsymbol{Temp2} \leftarrow \boldsymbol{\pi}_{\boldsymbol{Att_1},\ldots,\boldsymbol{Att_m},\boldsymbol{Att_y}}(\boldsymbol{Temp1}) \tag{4}$$

$$\boldsymbol{auditData} \leftarrow \boldsymbol{\rho}_{\boldsymbol{Att_n}/\boldsymbol{Att_y}}(\boldsymbol{Temp2}) \tag{5}$$

Next, we need to specify some normalization procedures, such as name or acronyms replacement, and values correction procedures. Both of them require an auxiliary table (Equation 6) to support mappings among user pre-defined corrections and source data.

$$\boldsymbol{mappingTable} = \langle \boldsymbol{Att_n}, \ldots, \boldsymbol{Att_{n1}} \rangle \tag{6}$$

The conforming data task is represented by the result of a join operation, identifying the conformed abbreviation/correction that will be replaced (Eq. 7, 8 and 9), in the target table:

$$\boldsymbol{Temp1} \leftarrow \boldsymbol{auditData} \bowtie \boldsymbol{mappingTable} \tag{7}$$

$$\boldsymbol{Temp2} \leftarrow \boldsymbol{\pi}_{\boldsymbol{Att_1},\ldots,\boldsymbol{Att_m},\boldsymbol{Att_{n1}}}(\boldsymbol{Temp1}) \tag{8}$$

$$\boldsymbol{srcTempData} \leftarrow \boldsymbol{\rho}_{\boldsymbol{Att_n/Att_{n1}}}(\boldsymbol{Temp2}) \tag{9}$$

Afterwards, the identification of tuples that have attribute values with no match in the auxiliary table can be loaded to specific quarantine tables for future analysis and recovering. This behavior can be specified from both data normalization and data correction procedures (Eq. 10 and 11):

$$\boldsymbol{Temp1} \leftarrow \boldsymbol{auditData} \triangleright \boldsymbol{mappingTable} \tag{10}$$

$$\boldsymbol{quarantine} \leftarrow \boldsymbol{auditData} - \boldsymbol{Temp1} \tag{11}$$

For post-function duplicate elimination procedure, we must to identify all duplicate attributes and treated properly. Considering a particular attribute Att_n, which stores information that should be unique in the dimension, one possible method to identify duplicates is to find if the value of the attribute is in more than one tuple, not respecting the uniqueness rule (Eq. 12, 13, 14 and 15). If such records exist then they must be stored in some quarantine table for future analysis.

$$\boldsymbol{Temp1}(\boldsymbol{Attn, mycount}) \leftarrow \boldsymbol{Att_n} \sqsupset_{COUNT} \boldsymbol{Att_n}(\boldsymbol{auditData}) \tag{12}$$

$$\boldsymbol{Temp2} \leftarrow \boldsymbol{\sigma}_{\boldsymbol{mycount>1}}(\boldsymbol{Temp1}) \tag{13}$$

$$\boldsymbol{Temp3} \leftarrow \boldsymbol{\pi}_{\boldsymbol{Att_n}}(\boldsymbol{Temp2}) \tag{14}$$

$$\boldsymbol{tempDataDup} \leftarrow \boldsymbol{Temp3} \bowtie \boldsymbol{auditData} \tag{15}$$

After identifying the duplicates, they should be removed from the dimension table:

$$auditData\_U \leftarrow auditData - TempDataDup \tag{16}$$

5 Conclusions and Future Work

We have presented in this paper a pattern ETL modelling proposal based on the BPMN 2.0 notation. We described a meta-model that supports our ETL practical implementation, integrating a set of patterns composed by a group of tasks that we consider as standard in any common ETL system implementation. At this point it is very important to distinguish our proposal from other BPMN conceptual modelling proposals. As we have shown in one of our past papers [7], we consider the idea of applying BPMN to ETL conceptual modelling very interesting, particularly in what is concerned with mapping of conceptual models in execution primitives, witch can be accomplished not only using BPEL but also using BPMN 2.0. Additionally, not only the familiarity of the stakeholders of the organization with the BPMN notation is a great advantage, but also the integration and communication of ETL processes developed with other organizational processes already implemented is guaranteed [21]. The approach presented in this paper was essentially settled on a set of templates (or patterns) that are composed by several atomic tasks representing standard ETL processes commonly used on regular ETL systems, which includes tasks such as: change data capture, slowly changing dimensions with history maintenance, surrogate

key pipelining or DQE procedures. We think that with this approach we can reduce significantly potential inadequacies on the design and implementation of an ETL system. As an example of a pattern specification, we presented a description DQE pattern, including a set of procedures used on data integration processes. We propose a XPDL specification for process control formalization, providing its serialization and computer interpretation, allowing for the representation of all BPMN diagram concepts in an interchangeable format. Finally, and using RA, we also presented a specification of all elementary transformations that were needed to apply all data transformations associated with each DQE procedure.

In a near future we intend to provide a complete family of BPMN patterns specification at control and data process levels, in order to have the possibility to design and develop a complete ETL process, covering its main areas of planning and implementation. Additionally, we will extend our work to provide specific mapping rules allowing for BPMN–based ETL conceptual model translation to a logical model, letting its conversion and integration into existing ETL support tools, services and structures.

References

1. Weske, M., van der Aalst, W.M.P., Verbeek, H.M.W.: Advances in business process management. Data & Knowledge Engineering 50 (2004)
2. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data (2004)
3. OMG: Documents Associated With Business Process Model And Notation (BPMN) Version 2.0. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0 (2011)
4. El Akkaoui, Z., Zimányi, E.: Defining ETL worfklows using BPMN and BPEL. In: Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP, DOLAP 2009, pp. 41–48 (2009)
5. El Akkaoui, Z., Zimányi, E., Mazón, J.-N., Trujillo, J.: A model-driven framework for ETL process development. In: Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP, DOLAP 2011, pp. 45–52 (2011)
6. El Akkaoui, Z., Mazón, J.-N., Vaisman, A., Zimányi, E.: BPMN-Based Conceptual Modeling of ETL Processes. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 1–14. Springer, Heidelberg (2012)
7. Oliveira, B., Belo, O.: BPMN Patterns for ETL Conceptual Modelling and Validation. In: Chen, L., Felfernig, A., Liu, J., Raś, Z.W. (eds.) ISMIS 2012. LNCS, vol. 7661, pp. 445–454. Springer, Heidelberg (2012)
8. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, DOLAP 2002, pp. 14–21 (2002)
9. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: On the Logical Modeling of ETL Processes. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 782–786. Springer, Heidelberg (2002)
10. Simitsis, A., Vassiliadis, P.: A Methodology for the Conceptual Modeling of ETL Processes. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 305–316. Springer, Heidelberg (2003)

11. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M.: A framework for the design of ETL scenarios. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 520–535. Springer, Heidelberg (2003)
12. El-Sappagh, S.H.A., Hendawi, A.M.A., El Bastawissy, A.H.: A proposed model for data warehouse ETL processes. Journal of King Saud University – Computer and Information Sciences 23 (2011)
13. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 307–320. Springer, Heidelberg (2003)
14. Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: Extending BPMN 2.0: Method and Tool Support. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 59–73. Springer, Heidelberg (2011)
15. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin 23, 2000 (2000)
16. Shapiro, R.M.: XPDL 2.1 - Integrating Process Interchange & BPMN (2008)
17. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13, 377–387 (1970)
18. Özsoyoğlu, G., Özsoyoğlu, Z.M., Matos, V.: Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. ACM Trans. Database Syst. 12, 566–592 (1987)
19. Grefen, P.W.P.J., de By, R.A.: A Multi-Set Extended Relational Algebra - A Formal Approach to a Practical Issue. In: Proceedings of the Tenth International Conference on Data Engineering, pp. 80–88. IEEE Computer Society, Washington, DC (1994)
20. Baralis, E., Widom, J.: An Algebraic Approach to Rule Analysis in Expert Database Systems. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 475–486. Morgan Kaufmann Publishers Inc., San Francisco (1994)
21. Wilkinson, K., Simitsis, A., Castellanos, M., Dayal, U.: Leveraging Business Process Models for ETL Design. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 15–30. Springer, Heidelberg (2010)