Task Clustering on ETL Systems
A Pattern-Oriented Approach

Bruno Oliveira and Orlando Belo
ALGORITMI R&D Centre, University of Minho,Campus de Gualtar, Braga, Portugal

Keywords: Data Warehousing Systems, ETL Conceptual Modelling, Task Clustering, ETL Patterns, ETL Skeletons, BPMN Specification Models, and Kettle.

Abstract: Usually, data warehousing populating processes are data-oriented workflows composed by dozens of granular tasks that are responsible for the integration of data coming from different data sources. Specific subset of these tasks can be grouped on a collection together with their relationships in order to form higher-level constructs. Increasing task granularity allows for the generalization of processes, simplifying their views and providing methods to carry out expertise to new applications. Well-proven practices can be used to describe general solutions that use basic skeletons configured and instantiated according to a set of specific integration requirements. Patterns can be applied to ETL processes aiming to simplify not only a possible conceptual representation but also to reduce the gap that often exists between two design perspectives. In this paper, we demonstrate the feasibility and effectiveness of an ETL pattern-based approach using task clustering, analyzing a real world ETL scenario through the definitions of two commonly used clusters of tasks: a data lookup cluster and a data conciliation and integration cluster.

1 INTRODUCTION

Extract, Transform, and Load (ETL) development is considered a very time-consuming, error-prone and complex task, involving several stakeholders from different knowledge domains. The amount of data extracted from (heterogeneous) data sources and the complexity associated with data transformation and cleaning tasks represent a significant effort in any data warehousing project (Kimball and Caserta, 2004). Thus, commercial tools with the ability to support ETL development and implementation have a crucial impact in the development of such processes, providing the generation of very detailed and documented models. However, such tools provide very specific notations that were developed to support very specific architecture choices. So, they use to produce very large populating processes, often difficult to understand, with granular activities associated to specific modelling rules. Even supporting composite constructs that represent common used routines, they are still very platform-specific since its implementation are not based in solid formalisms such as Relational Algebra (Santos and Belo, 2013) operators. Considering this, it is pretty obvious that such tools cannot be used for conceptual representation, since communication with non-technical users (e.g. business users and database administrators) would be compromised. In the development process of an ETL system we need to consider all the aspects related to developing phases as well as the evolution the system may suffer across time. Both are critical factors of success.

At early development phases, the business processes, and by consequence business users, plays an important role because they really know business from inside, providing requirements validation that are crucial to the feasibility of a decision support system. Several works addressed this important issue, proposing the use of well-know modelling languages such as *Business Process Modelling Language* (BPMN) (OMG, 2011) to support ETL conceptual modelling (Akkaoui and Zimanyi, 2009) (Akkaoui et al., 2011). BPMN is widely used to describe and implement common business processes. Choosing BPMN for conceptual modelling provides several advantages. BPMN notation disposes very expressive constructs, allowing for the representation of data flows in a large variety of ways, capturing both data and control flow constraints. However, such expressiveness represents a delicate problem, especially when an ETL conceptual model needs to be translated and interpreted by a machine. The

Oliveira B. and Belo O..
Task Clustering on ETL Systems - A Pattern-Oriented Approach.
DOI: 10.5220/0005559302070214
In *Proceedings of 4th International Conference on Data Management Technologies and Applications* (DATA-2015), pages 207-214
ISBN: 978-989-758-103-8

research community addressed these problems over the years, specifically in what is concerned to mapping BPMN processes to BPEL language (Ouyang et al., 2007). The detail and scope disparities between more abstract and concrete models represent a huge distance between two representations, simply because they serve different purposes. In order to simplify ETL development, we propose the application of a task-clustering technique to group a set of finer grain tasks into a collection of tasks, flows and control primitives, providing a method to organize them using layers of abstraction and supporting different detail to serve the several stakeholders in different project phases. The cluster categorization provides a way to identify and classify patterns that can be instantiated and reused in different ETL processes. Each pattern is scope independent and provides a specific skeleton that not only specifies their internal behaviour but also enables communication and data interchange with other patterns in a workflow context.

In this paper, we demonstrate the feasibility and effectiveness of the approach we developed and followed analysing a real world ETL scenario and identifying common tasks clusters. In section 2 we discuss its generalization and use as general constructs in an ETL package. Then, we demonstrate how activities can be grouped to build ETL conceptual models in different abstraction layers (section 3), presenting two ETL skeletons (data lookup and data conciliation and integration) exposing their configuration and behaviour in a way that they can be executed in a target ETL tool (section 4). Next, some related work is exposed (section 5). Finally, in section 6, we evaluate the work done, pointing out some research guidelines for future work.

2 ETL TASKS CLUSTERING

On-line Transaction Processing (OLTP) systems are responsible for recording all business transactions performed in enterprise operational systems. These are built to support specific business, which store operational data from the daily business operations. Therefore, they are the main data sources used by data warehousing systems. In more complex cases, data are distributed following the distinct business branches of a company. These data can be stored in sophisticated relational databases or in more simple data structures (e.g. texts or spreadsheets files). Due to this variety of information sources, problems on populating a data warehouse often occur (Rahm and Do, 2000).

Generally, tasks used on the extraction step (E) are responsible to gather data in the data sources and put it into a *data staging area* (DSA). The DSA is a working area where data is prepared before going to the data warehouse. For that, the DSA provides the necessary metadata to support the entire ETL process, providing, for example, support for data correction and data recovery mechanisms using domain oriented dictionary, mapping mechanisms or quarantine tables. Transformation and cleaning (T) procedures are applied posteriorly to data extraction, using the data that was already stored in temporary structures in the DSA. After this second step, data is loaded (L) to a target data warehouse, following schema rules, and operational and business constraints. Essentially, an ETL process represents a data-driven workflow representing a set of tasks and their associated control flows and business rules that together express how the system should coordinated. Typically, the commercial tools use workflows to the representation of very specific tasks that are frequently grouped together each time we want to represent a same procedure.

To reduce the impact of such situations, we purpose an approach (also used in others application scenarios (Singh et al., 2008) that allows us to organize ETL tasks into clusters and execute them as a single block. However, we went a little bit further formalizing a set of "standard" clusters representing some of the most common techniques used in real world ETL scenarios. Based on a set of input parameters, the tasks that compose a cluster should produce a specific output. In fact, we are creating routines or software patterns that can be used with the aim to simplify ETL development, reducing the implementation errors and time needed to implement ETL processes. With the ETL patterns identified, we propose a multi-layer approach to represent them, using BPMN pools to represent the several layers of abstraction, composed by several lanes representing the tasks that should be applied for each group of similar records to be processed.

To demonstrate the potential of our approach, we present a common ETL scenario, representing an ordinary data extraction process that using two different data sources prepare data through confirming and inserting it posteriorly in a data warehouse. The first source, a relational schema, stores data about flights (dates and flight time), travel locations (city and country) and planes (brand, model and type, and manufacturer data). The spreadsheet source represents a subset of the data structures that can be found in the relational schema.

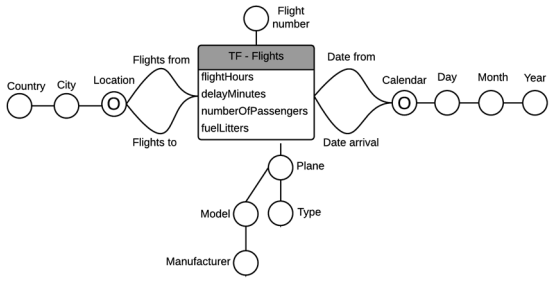

Figure 1: The "Flights" star schema.

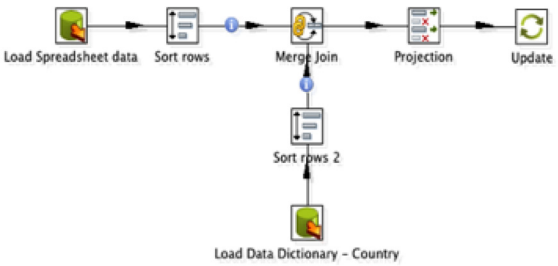

Figure 2: Implementation of a normalization process.

This example was selected to illustrate all the steps related to extract, conciliate, and integrate data into a data warehouse schema (Figure 1), revealing how we used our ETL pattern-based approach, using task clustering, in a real world ETL scenario. The data warehouse model is represented using the notation of (Golfarelli and Rizzi, 2009). It integrates four distinct dimension tables ("Calendar", "Planes", Flights", and "Locations") and a single fact table ("FT-Flights"), providing the necessary structures to analyze flights data of a particular set of company branches over time.

Once specified the data warehouse's storage structure, it is necessary to define the ETL process, adjusting it to the specificities of the system. An incorrect or incomplete specification of the ETL activities will affect seriously the quality of the data warehouse. Even in such simple example, the populating process of the data warehouse have dozens of tasks. Analysing each phase individually, several clusters of tasks with a finer grain detail can be identified - see Figure 2 where it is represented a miniminal subset of the ETL workflow process needed to populate the "Location" dimension, implemented in Kettle (Pentaho, 2015). This process represents the normalization of an attribute ("Country") that was collected in the spreadsheet data source, and simply replaces the abbreviation values (e.g. 'PRT', 'ESP', or 'FRA') used to identify countries by the correspondent full names (Portugal, Spain, or France). This transformation is done using a dictionary table having all the correspondences between countries full names and the abbreviation values for each country is used. This process represents a simple lookup operation that can be reused in many other similar situations. If we group all the tasks involved with into a data lookup cluster, including a well-defined description of their behavior, we have the ability to define a new ETL container, a *Data Lookup* (DL) cluster or pattern.

Additionally, the semantics of populating process from Figure 2 is strong related to the methodology followed by the Kettle ETL implementation tool - for example, before we can proceed to the *Merge Join* task, between the two data sets, we need explicitly to sort the data set by the attributes that will participate in the joining condition. To generalize the behaviour of this cluster, we need only to change its configuration parameters. All the rest remains. However, the configuration output metadata should be provided, i.e. a compatible target repository and its mappings to store execution results should be indicated.

Following the previous example, it is pretty clear that the physical approach does not help the ETL planning and development. The use of abstract models can reduce the effects of an unsuitable planning, providing task encapsulation for hiding processes complexity to end-users. Thus, users can focus in non-technical details, helping in early ETL development stages revealing the most critical business and operational requirements.

In this work we explored two important stages of the development of an ETL system: the ETL conceptual representation, using a set of abstract constructs that contributes to increase process interpretation, and the enrichment of conceptual representations, giving them enough detail to transform abstract constructs to a set of detailed tasks that can be mapped and executed in a commercial ETL implementation tool. ETL patterns establish a bridge between those models. Besides, they provide the necessary meaning to drill down more general constructs and support its (semi) direct execution. So, we can generate executable models using general templates describing how a problem must be solved independently of the context in which they will be applied. Besides their strong reusing abilities, patterns in workflow systems provide a very flexible way to specify and share communication protocols, increase data interchange across processes, and allow for incremental integration of new ETL patterns. All this will improve the quality of an ETL system solution.

3 A MULTI-LAYER APPROACH

In large ETL projects, conceptual models play an important role being very useful on preliminary development stages where users must validate business requirements. They help users having different knowledge to understand the meaning of concepts involved. Basically, BPMN it provides a very simply and powerful notation for process representation (very suitable for processes such as ETL), coupled with his power of expressiveness, implementation constructs, and control tasks provided. Additionally, companies use BPMN used to describe and implement internal business processes. So, generically, we can identify two advantages: 1) business users are already familiar with process construction language; and 2) existing business processes can be used and integrated with ETL process, taking advantage of existing routines/process logics. To support different abstraction levels, we can use BPMN collapsed sub processes that can be applied successfully to ETL processes. This provides process conceptualization, since complex constructs are decomposed into different levels. This is very advantageous for high-level users when presenting, discussing and understanding process concepts.

Considering the example used, we design a possible subset of activities needed for data extraction on the two referred information sources in order to feed some temporary data structures stored in the DSA (Figure 3). Figure 3a) represents the ETL layer 2, showing the level that can be derived based on the extraction phase included in the main layer (layer 1), representing the logic of the process related to the extraction of data in both sources. Two parallel BPMN gateways were used to indicate that tasks belonging to each flow could be executed in parallel. Two different BPMN sub-processes are also represented, namely the '*Extraction Tasks*' and the '*#CDC# - spreadsheet'*. The first process is a very ordinary BPMN sub-process, which hides the complexity of the data extraction process applied on the relational source. The second one represents an ETL pattern (we used the '#' symbol to distinguish it), which represents the logic of the differential loading process performed over the data collected.

The CDC pattern was defined as a composite task. However, for conceptual representation, the user does not need to know its composition, but rather the configuration for each parameter associated in order to enable it. In contrast with this pattern, the first extraction process can be easily decomposed (Figure 3b)), representing the data extraction from the tables "Planes" and "Location".

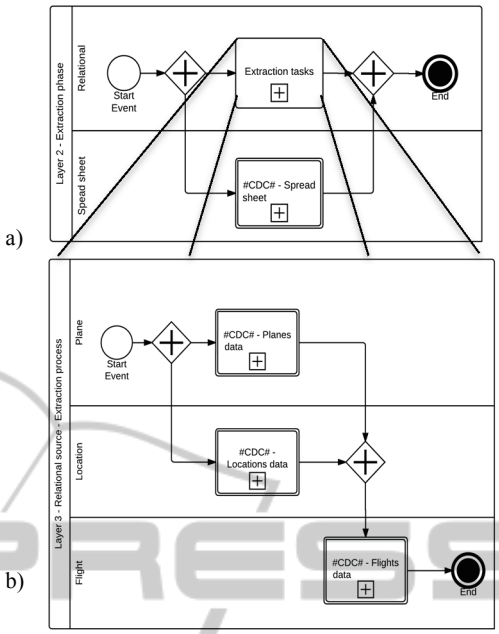

Figure 3: Representation of an ETL extraction phase - layers 2 (a) and 3 (b).

After the conclusion of both processes, the data stored in the table "Flights" is extracted considering DSA specific requirements. The composition of ETL layer is represented through a BPMN pool, where the lanes represent the roles of the processes associated with the ETL tasks. The transformation cluster can be represented using only software patterns (Figure 4). To consolidate and normalize data from each data source, we defined several BPMN lanes to represent the transformations of the entities that should be applied to each information sources.

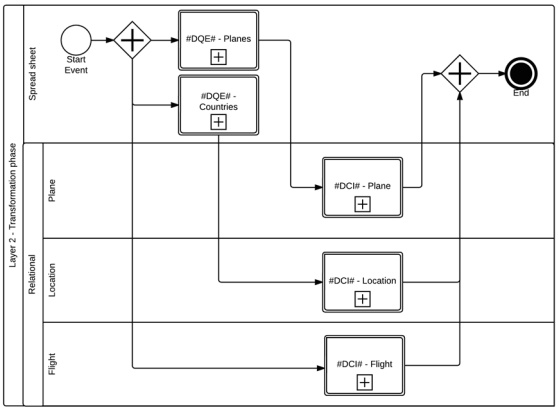

Figure 4: Representation of ETL Transformation phase (layer 2).

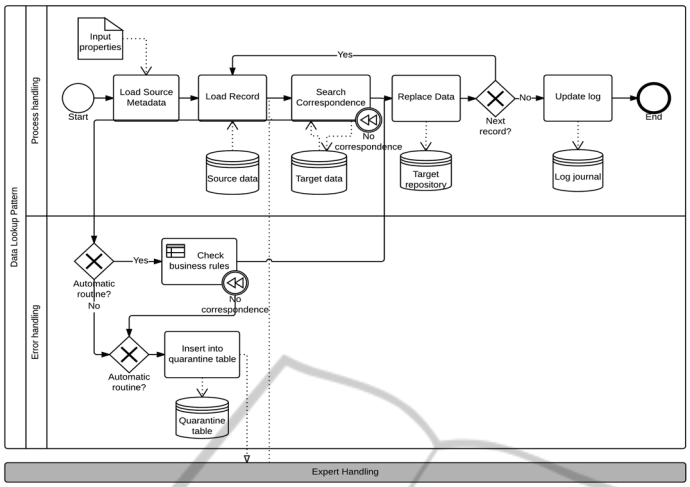

Figure 5: Generic algorithm for DL pattern.

The process starts with the execution of three concurrent flows. For the spreadsheet source, two DQE (*Data Quality Enhancement*) ETL patterns were used to apply data cleaning and normalization techniques over the tables “Planes” and “Countries”. Next, a set of DCI (*data conciliation and integration*) patterns was used in order to integrate data about planes, locations and flights, ensuring data integration and consistency between data sources. The DCI pattern involves several integration problems that uses other external patterns. For this particular case, a DCI pattern need to call a SK (surrogate key) pattern in order to generate the diemnsios surrogate keys. Upon completion of each flow, data are synchronized and the loading procedures activated.

The use of more abstract BPMN constructors, especially oriented to receive ETL requirements, provides a way to encapsulate a considerable set of repetitive tasks by grouping them into clusters that represent ETL patterns. As such, this not only allows for a simplified process modeling but also provides more guarantees of consistency to ETL processes. Using BPMN on ETL process specification is quite interesting. It supplies a useful and expressive notation at ETL conceptualization stage, allowing for the representation of powerful orchestration mechanisms on process instantiation. We can enrich BPMN models with some specific implementation details to support the automatic generation of physical models based on its correspondent conceptual representation. Eventual inconsistencies related to the expressiveness of BPMN constructs are minimized when we use ETL patterns.

4 PATTERNS SKELETONS

To enrich the ETL pattern approach we followed, we provide a set of patterns definitions that can be instantiated for specific ETL scenarios. We describe patterns as a set of tasks that need to be executed in a specific order. Such tasks can be atomic or composed, depending of the detail level associated to its implementation. We do not intend to provide a workflow engine to execute ETL processes. Instead, we want to present a generic way to describe ETL patterns to ensure their mapping and execution on a commercial ETL tool based on specific transformations templates. For that, the architecture and behaviour of each pattern must be described as a set of functional components, i.e. an input configuration, an output configuration, an exception-handling configuration, and a log-tracking configuration

We can view an ETL pattern as a single unit of work, representing several operations that can be used to transfer data between data sources. Since data representation is intrinsically connected to their data schemas and implementation technology, the use of ETL patterns allow the representation of data in several states, ensuring its consistency with target repository requirements. The ACID properties (atomicity, consistency, isolation, and durability) for database transactions provide a set of base characteristics that can also be considered in the context of ETL patterns. Additionally to instance level constraints, schema-level exceptions can occur, which can leads to a compensation or a cancelation event to handle appropriately such cases. For example, a normalization procedure applied on a

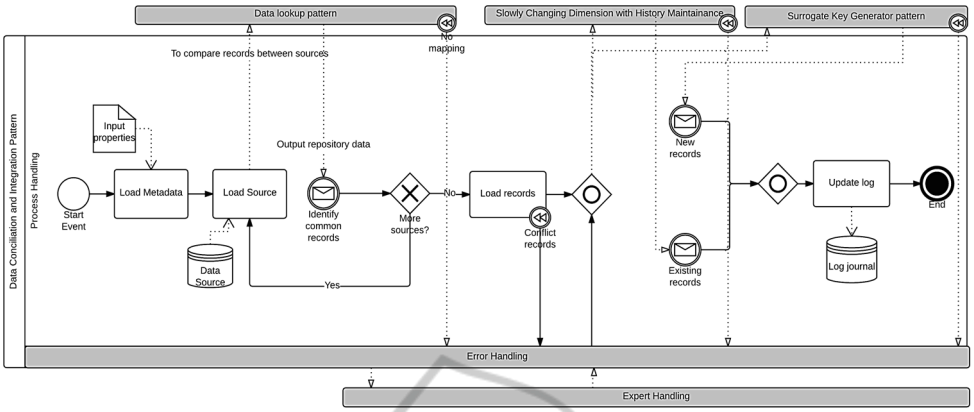

Figure 6: Generic algorithm for DCI pattern.

non-existent attribute should activate a cancelation event. However, if some specific records have unexpected values, that records can be stored in quarantine tables for further evaluation.

Generally, the behaviour of the process present previously in Figure 2 can be described appealing to a well-known procedure – a data lookup operation over a specific table in order to identify the correspondences that can be used to detect incorrect or missing values, correcting them automatically when possible.

The Figure 5 presents a generic algorithm for the DL pattern that is scope independent and can be used in single or composite contexts. The DL pattern starts by reading the input metadata that allows for it to identify all binding maps between source data and data stored in the data warehouse. Configuration data can be passed between activities at the process scope. Next, records from source data are read (*Load Record* activity). For each one of them, a target correspondence is verified using predefined matching attributes over the correspondent lookup table. A simple mapping: <*old value, new value*> is used to support *Search Correspondences* activity. If no correspondences are found for a given value, the process will identify an anomaly, and starts a compensation event in order to preserve on a specific quarantine table all the records without correspondence. These records will be posteriorly analyzed by a system administrator or, if possible, by a correction and recover mechanism.

A BPMN *Business rule* task was also used to represent automatic policies to handle specific application scenarios – e.g. to solve a non-existing record correspondence one may specified a rule to delete the record. Tasks like these fit completely on our implementation purposes, since rules can be added or changed without affecting other tasks, hiding its complexity to more general users.

The *Replace data activity* will be responsible to materialize processed data (i.e. records with correspondence) in the target repository specified in pattern output metadata. Finally, the DL pattern should update an ETL log journal with data output summarizing the entire process.

Internally, each pattern is responsible to preserve temporary data structures to handle intermediate results and materialize them if needed. Several BPMN artifacts were used to represent data with persistent characteristics beyond process scope. They represent the target schemas that each activity will interact. The use of DL patterns is quite very useful in the specification of the DCI procedure, which activities are formulated in Figure 6. Similarly to the DL pattern, the process starts with the configuration parameters specification that is needed to access to source data schemas. DCI's metadata also includes the binding rules that must be applied to common attributes, allowing for the preservation of relationships between fields with different names or ordinal positions. Next, the process handles each record according to a set of specific rules that define its integration in the data warehouse.

The Load source task is responsible to load the records collected on each data source providing the means to extract the records and use them as input to identify common records. For the representation of the DCI pattern, we used a BPMN collaboration diagram. Unlike the DL pattern, the DCI pattern need to communicate with the DL pattern and, depending on the requirements established, may communicate with the SK and SCD (slowly changing dimension) patterns in order to accomplish its function. This behavior was modeled using intermediate message flows events, which send messages to specific pools that represent patterns - pools are collapsed due to space limitations. Each pattern receives the request and internally executes the respective procedure.

The DL pattern is responsible to identify common records between each data source. This is necessary because the DCI pattern may receive data from two or more data sources. The dictionary table that preserves the binding rules must be queried to identify mappings between the data schemas of each source. The records that must be processed are never transferred in events messages. Instead, the messages exchanged store information about the input/output repository that keeps the records and the specificities that should be used to process them. As already referred, for any record without correspondence, a compensation event is launched internally in the DL pattern as a way to signalize a no-occurrence exception. The event can occur inside an external pattern, but it is always reported to DCI pattern.

The error handling used is very similar to the DL pattern. After processing all sources, a BPMN inclusive gateway is used to determine which paths will be taken next. For the new and updated records without conflicts among data sources selected by *Load records* activity, a surrogate key generation pattern is invoked for the generation of a new surrogate key and to update the correspondent mapping tables. For the updated records, a flow can be started to handle new versions of data using a slowly changing dimensions technique. The history preservation should be handled using a SCD pattern with a history table to keep old versions of data. However, in some cases records have multiple versions between sources, which lead to many redundancy problems. To solve these issues, a compensation event in *Load Source* activity was used to specify an alternate path to unlock these cases. For example, a set of priority rule can be defined to solve conflict records. Finally, the process ends with *Update log* activity that will store all operations performed during the process. Like the DL pattern, the representation of a DCI pattern also include the *Error handling* layer (also collapsed due to space limitations), which describes the error handling processes, not only fired by internal tasks but also by ETL patterns involved.

Following the ETL pattern context presented, both DL and DCI patterns can be categorized in different levels. The DL pattern can be categorized as a single pattern because its internal composition does not imply the choreography with other patterns. However, the DCI pattern is a composite pattern that needs to exchange messages with other patterns in order to accomplish a common goal. Moreover, with the basic configuration provided, the dynamic generation of instances following the base configuration provided will be facilitated.

5 RELATED WORK

In the field of ETL patterns, as far as we know, there is not much to refer. (Köppen et al., 2011) proposed a pattern approach to support ETL development, providing a general description for a set of patterns, i.e. aggregator, history and duplicate elimination patterns. In our opinion, this work present important aspects defining composition properties that are executed before or after each pattern. However, for our ETL approach, we believe that the formalization of these composite properties will limit ETL design, because populating processes represent very specific needs of a decision support system that can be handled in a large variety of forms. Considering the requirements of a conventional ETL process, we provided a way to represent each pattern using visual constructs using the BPMN language (Akkaoui and Zimanyi, 2009).

Alternatively, (Muñoz et al., 2009) proposed a MDA approach to derive ETL conceptual models modelled with UML activity diagrams to carry out an automatic code generation for ETL tools. Later, (Akkaoui et al., 2011) complemented their initial proposal providing this time a BPMN-based meta-model for independent ETL modelling. They explored and discussed the bridges to a model-to-text translation, providing its execution using some ETL commercial tools. Still using BPMN notation, (Akkaoui et al., 2012) provided a BPMN meta-model covering two important data process operations, and the workflow orchestration layer.

More recently, following the same guidelines from previous works, the same authors (Akkaoui et al., 2013) provided a framework that allows for the translation of abstract BPMN models to its concrete execution in a target ETL tool using model-to-text transformations. Based on these contributions, we are working on the physical representation of BPMN conceptual models in order to provide its execution using existing ETL commercial tools. Additionally, we want to explore its mapping using a set of model-to-code transformations, providing dynamic translation between our conceptual model to a serialization model that can be executed in specific ETL tool. Previously, we already explored several ETL patterns specification and its physical mapping, such as the SOA architecture (Oliveira and Belo, 2012; Oliveira and Belo, 2013).

6 CONCLUSIONS AND FUTURE WORK

In this paper we identified and discussed two ETL clusters as a way to group and organized a set of inter-related tasks. These clusters can be generalized and, when integrating specific configuration parameters, can form ETL patterns. These have the ability to represent a general solution for some typical ETL composite tasks, which are commonly used in conventional ETL processes – e.g. surrogate key pipelining, slowly changing dimensions, change data capture, or intensive data loading. Using the BPMN language, we presented a conceptual approach to model such processes applying ETL patterns. With this component-based ETL development approach, ETL designers and developers only have to take into account the interfaces that define the interaction and communications among ETL patterns. From a conceptual point of view, we consider that ETL models should not include any kind of implementation infrastructure specification. BPMN provides this kind of abstraction, since it allows for the representation of several levels of detail in a same process, fitting well the needs of the conceptual approach proposed, and contributing to reduce functional and operational errors, and decrease its overall costs. To prove the adequacy of our approach, we have presented a global specification of an ETL model using a set of ETL patterns was used. As future work we intend to provide an extended family of ETL patterns to build a complete ETL system, covering the integration of tasks that can be used in a regular ETL system.

REFERENCES

Akkaoui, Z. El et al., 2013. A BPMN-Based Design and Maintenance Framework for ETL Processes. *International Journal of Data Warehousing and Mining (IJDWM), 9.*

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 worfklows using BPMN and BPEL. In DOLAP '09 *Proceedings of the ACM twelfth international workshop on Data warehousing and OLA*P. pp. 41–48.

Golfarelli, M. & Rizzi, S., 2009. Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill.

Kimball, R. & Caserta, J., 2004. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data,

Köppen, V., Brüggemann, B. & Berendt, B., 2011. Designing Data Integration: The ETL Pattern Approach. The European Journal for the Informatics Professional, XII(3).

Muñoz, L., Mazón, J.-N. & Trujillo, J., 2009. Automatic Generation of ETL Processes from Conceptual Models. *In Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP. DOLAP* '09. New York, NY, USA: ACM, pp. 33–40.

Oliveira, B. & Belo, O., 2012. BPMN Patterns for ETL Conceptual Modelling and Validation. The 20th International *Symposium on Methodologies for Intelligent Systems: Lecture Notes in Artificial Intelligence.*

Oliveira, B. & Belo, O., 2013. Approaching ETL Conceptual Modelling and Validation Using BPMN and BPEL. In *2nd International Conference on Data Management Technologies and Applications (DATA).*

OMG, 2011. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0.

Ouyang, C. et al., 2007. Pattern-based translation of BPMN process models to BPEL web services. International *Journal of Web Services Research (JWSR*),5,pp.42–62.

Pentaho, "Pentaho Data Integration". Available at: http://www.pentaho.com/product/data-integration [Accessed March 16, 2015].

Rahm, E. & Do, H.H., 2000. Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin, 23, p.2000.

Santos, V. & Belo, O., 2013. Modeling ETL Data Quality Enforcement Tasks Using Relational Algebra *Operators. Procedia Technology, 9(0)*, pp.442–450.

Singh, G., Su, M. & Vahi, K., 2008. Workflow task clustering for best effort systems with Pegasus. Proceedings of the *15th international conference on Advanced information systems engineering.*