A BPMN-Based Design and Maintenance Framework for ETL Processes

Zineb El Akkaoui, Department of Computer & Decision Engineering (CoDE), Université Libre de Bruxelles, Brussels, Belgium

Esteban Zimányi, Department of Computer & Decision Engineering (CoDE), Université Libre de Bruxelles, Brussels, Belgium

Jose-Norberto Mazón, Department of Software and Computing Systems, University of Alicante, Alicante, Spain

Juan Trujillo, Department of Software and Computing Systems, University of Alicante, Alicante, Spain

ABSTRACT

Business Intelligence (BI) applications require the design, implementation, and maintenance of processes that extract, transform, and load suitable data for analysis. The development of these processes (known as ETL) is an inherently complex problem that is typically costly and time consuming. In a previous work, the authors have proposed a vendor-independent language for reducing the design complexity due to disparate ETL languages tailored to specific design tools with steep learning curves. Nevertheless, the designer still faces two major issues during the development of ETL processes: (i) how to implement the designed processes in an executable language, and (ii) how to maintain the implementation when the organization data infrastructure evolves. In this paper, the authors propose a model-driven framework that provides automatic code generation capability and ameliorate maintenance support of our ETL language. They present a set of model-to-text transformations able to produce code for different ETL commercial tools as well as model-to-model transformations that automatically update the ETL models with the aim of supporting the maintenance of the generated code according to data source evolution. A demonstration using an example is conducted as an initial validation to show that the framework covering modeling, code generation and maintenance could be used in practice.

Keywords: Code Generation, Conceptual Model, Data Warehouses, ETL Process, Maintenance, Model-Driven Development

DOI: 10.4018/jdwm.2013070103
INTRODUCTION

Organizational data used by BI applications come from heterogeneous and distributed sources that are integrated into a data warehouse (DW) (Inmon, 2002). To achieve this integration, the data warehousing process includes the extraction of the data from the sources, the transformation of these data (e.g., to correct syntactic and semantic inconsistencies) and the loading of the warehouse with the cleansed, transformed data. This process is known as ETL (standing for Extraction, Transformation, Load). It has been widely argued that the ETL process development is complex, error-prone, and time-consuming (Simitsis, 2008; Vassiliadis, 2009; Wyatt, 2009). Actually, ETL process development constitutes the most costly part of a data warehouse project, in both time and resources.

One of the main reasons for this is that, in practice, ETL processes have been traditionally designed by considering a specific vendor tool from the very beginning of the data warehouse project lifecycle. Unfortunately, commercial ETL tools have a steep learning curve, due to a lack of standard capabilities to be provided, e.g., they all provide different underlying languages with a wide spectrum of functionality features or complex wizards.

Some existing approaches address this problem by proposing a conceptual modeling stage for developing ETL processes in a vendor-independent manner (Skoutas 2009; Trujillo, 2003; Vassiliadis, 2005). These proposals successfully support the designer tasks, although they lack of effective mechanisms for automatically generate vendor-specific code of the ETL process to be executed into different platforms. Moreover, the increasing need of fresher analysis data and the evolving nature of organizational data pose new challenges for these proposed approaches. The lack of systematic technique for continuous update of ETL process increases significantly the development effort (Papastefanatos, 2009). Indeed, during the ETL process lifecycle, both the data sources and the analysis requirements are likely to evolve, the latter implying an evolution of the data warehouse. Such changes may lead to inaccurate ETL processes: (i) syntactically invalid ETL model and code; and (ii) inconsistent data output generated by the process to feed the data warehouse. To avoid this situation, the ETL process should be automatically updated to accommodate the evolution. However, in general, schema evolution is done manually and remains an error-prone and time-consuming undertaking, because the designer lacks the methods and tools needed to manage and automate this endeavor by (i) predicting and evaluating the effects of the proposed schema changes, and (ii) rewriting queries and applications to operate on the new schema.

To overcome these problems, the present work proposes a Model-Driven Development (MDD) framework for ETL processes. This framework aims at covering the overall ETL development process, including the automatic generation of vendor-specific code for several platforms. Further, the framework supports an automated maintenance capability of the process and its code in order to accommodate evolution of organizational data.

For creating and managing ETL processes, in addition to the traditional graphical languages, current platforms generally provide programming capabilities through specific languages, which can be scripting languages (e.g. Oracle Metabase or OMB) or imperative languages (e.g. C# for SQL Server Integration Services). In our framework, transformations between a vendor-independent model and such vendor-specific code are formally established by using model-to-text (M2T) transformations, an OMG standard for transformations from models to text (i.e. code). For evolving ETL processes, a set of model-to-model (M2M) transformations are iteratively applied on the original model to automatically derive the updated one. Finally, by applying our M2T transformations, the updated code can be derived.

The present framework relies on our previous work: the BPMN4ETL metamodel for designing ETL processes described in El Akkaoui et al. (2012) and ElAkkaoui and Zimányi (2009). The rationale behind this metamodel...
Related Content

Sequential Patterns Postprocessing for Structural Relation Patterns Mining
www.igi-global.com/article/sequential-patterns-postprocessing-structural-relation/1814?camid=4v1a

Vehicle to Cloud: Big Data for Environmental Sustainability, Energy, and Traffic Management
Alper Ozpinar and Serhan Yarkan (2016). Effective Big Data Management and Opportunities for Implementation (pp. 182-201).
www.igi-global.com/chapter/vehicle-to-cloud/157692?camid=4v1a
A Survey of Big Data Analytics Systems: Appliances, Platforms, and Frameworks
[www.igi-global.com/chapter/a-survey-of-big-data-analytics-systems/150204?camid=4v1a](www.igi-global.com/chapter/a-survey-of-big-data-analytics-systems/150204?camid=4v1a)

A Survey on Fuzzy Association Rule Mining
[www.igi-global.com/article/survey-fuzzy-association-rule-mining/75613?camid=4v1a](www.igi-global.com/article/survey-fuzzy-association-rule-mining/75613?camid=4v1a)