POIESIS: a Tool for Quality-aware ETL Process Redesign

Vasileios Theodorou[1] Alberto Abelló[1] Maik Thiele[2] Wolfgang Lehner[2]

[1]Universitat Politécnica de Catalunya
Barcelona, Spain
{vasileios,aabello}@essi.upc.edu

[2] Technische Universität Dresden
Dresden, Germany
{maik.thiele,wolfgang.lehner}@tu-dresden.de

ABSTRACT

We present a tool, called **POIESIS**, for automatic ETL process enhancement. ETL processes are essential data-centric activities in modern business intelligence environments and they need to be examined through a viewpoint that concerns their quality characteristics (e.g., data quality, performance, manageability) in the era of Big Data. **POIESIS** responds to this need by providing a user-centered environment for quality-aware analysis and redesign of ETL flows. It generates thousands of alternative flows by adding flow patterns to the initial flow, in varying positions and combinations, thus creating alternative design options in a multidimensional space of different quality attributes. Through the demonstration of **POIESIS** we introduce the tool's capabilities and highlight its efficiency, usability and modifiability, thanks to its polymorphic design.

1. INTRODUCTION

The increasing volume of available data, as well as the requirement for recording and responding to multiple events coming from participants within Big Data ecosystems that are characterized by the 3Vs (volume, variety, velocity) [3], pose a serious challenge for modern data-centric processes. As such, Extract-Transform-Load (ETL) processes are becoming more and more complex, while there is a growing demand for their real time responsiveness and user-centricity.

It has recently been proposed that to tackle complexity, the level of abstraction for ETL processes can be raised. ETL processes have been decomposed to ETL activities [6] and recurring patterns [1] as the main elements of their workflow representation, making them susceptible to analysis for process evaluation and redesign.

Manual modification of ETL processes in order to improve their quality characteristics is error-prone, non-trivial, time-consuming and it suffers from incompleteness, inefficiency, and ineffectiveness. According to our experience with individuals with computer science expertise, most common mistakes during this manual process are wrong configuration of ETL operations, incomplete exploitation of quality enhancement options and wrong placement of optimization patterns.

It is apparent that there is a need for an automatized process of ETL quality enhancement, as it would solve many of the above-mentioned issues. Analysts should be in the center of this process, where the large problem space is automatically generated, simulated and displayed in an intuitive representation, allowing for the selection among alternative design choices.

In this paper we present our tool **POIESIS**, which stands for **P**rocess **O**ptimization and **I**mprovement for **E**TL **S**ystems and **I**ntegration **S**ervices. Using a process perspective of an ETL activity, our tool can improve the quality of an ETL Process by automatically generating optimization patterns integrated in the ETL flow, resulting to thousands of alternative ETL flows. We apply an iterative model where users are the key participants through well-defined collaborative interfaces and based on estimated measures for different quality characteristics. POIESIS implements a modular architecture that employs reuse of components and patterns to streamline the design. Our tool can be used for incremental, quantitative improvement of ETL process models, promoting automation and reducing complexity. Through the automatic generation of alternative ETL flows, it simplifies the exploration of the problem space and it enables further analysis and identification of correlations among design choices and quality characteristics of the ETL models.

The remainder of this paper is organized as follows: In Section 2 we provide some background for ETL quality analysis and redesign; in Section 3 we provide an overview of the system and finally, in Section 4 we showcase an outline of a demonstration of our tool.

2. QUALITY-AWARE REDESIGN OF ETL

2.1 ETL Quality Characteristics

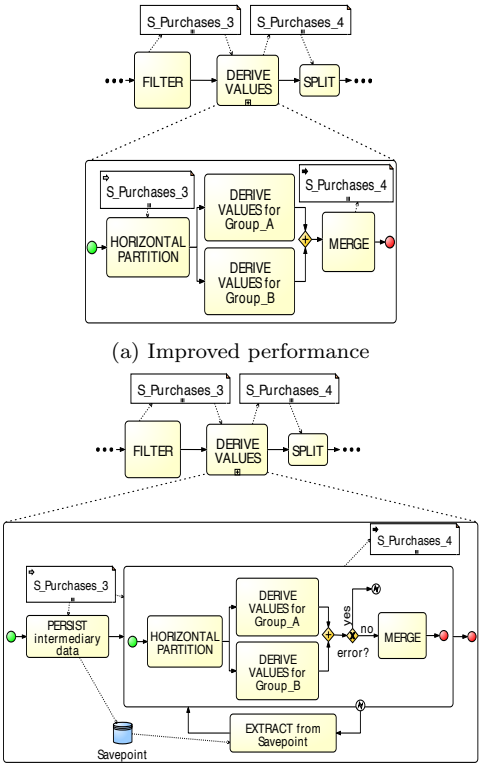
ETL processes need to be evaluated in a scope that brings them closer to fitness to use for data scientists. Therefore, apart from performance and cost, other quality characteristics, as well as the trade-offs among them should be taken under consideration during ETL analysis. In Fig. 6 we show a subset of ETL process quality characteristics and measures that we have extracted from existing literature, in our previous work [4]. There are two types of measures: ones that derive directly from the static structure of the process model

and those that are obtained from analysis of historical traces capturing the runtime behaviour of ETL components.

Characteristic	Measure
performance	• Process cycle time • Average latency per tuple
data quality	• Request time - Time of last update • 1 / (1 - age * Frequency of updates)
manageability	• Length of process workflow's longest path • Coupling of process workflow • # of merge elements in the process model

Figure 1: Example quality measures for ETL processes

Based on such measures, it is possible to conduct a multi-objective analysis and make design decisions according to user preferences on different quality characteristics, which can often be conflicting.



(a) Improved performance

(b) Improved reliability

Figure 2: Generation of FCP on the ETL flow

2.2 Addition of Flow Component Patterns

An initial ETL flow can be modified with the addition of predefined constructs that improve certain quality characteristics, but do not alter its main functionality. We refer to these constructs as Flow Component Patterns (FCP) and their integration can take place on different parts of the initial flow, depending on the flow topology. For example, in Fig. 2, we illustrate how different quality goals can cause the generation of different FCP on the ETL flow. In the first case, the goal of improving time performance of the process, results in the generation of horizontal partitioning

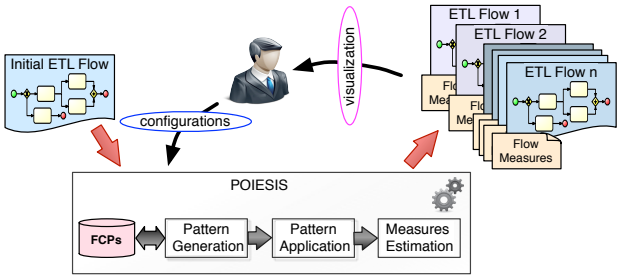


Figure 3: POIESIS architecture

and parallelism within a computational-intensive task and in the second, the goal of improving reliability brings about the addition of a recovery point to the sub-process. Another example would be the goal of improved data quality that would result in crosschecking with alternative data sources.

Central to our implementation is the notion of *application point* of a FCP, which can be either a node (i.e., an ETL flow operation), or an edge or the entire ETL flow graph. As examples, a valid application point for the *ParallelizeTask* pattern is a node that can be replaced by multiple copies of itself and a valid application point for the *FilterNullValues* pattern is an edge on which a filter operation can be added. The entire ETL flow graph as application point serves for the case of process-wide configuration and management operations that are not directly related to the functionality of specific flow components. Examples of the latter include the application of security configurations (encryption, role-based access etc.), management of the quality of Hw/Sw resources, adjusting the frequency of process recurrence etc.

We model the ETL process as one graph G with graph components (V, E), where each node (V) represents an ETL flow operation, and each edge (E) represents a transition from one operation to a successor one. We also assume that there is a set P of available FCP, $P = P_E \cup P_V \cup P_G$, each of which can either be applied on a node, an edge of G, or the entire graph, in order to improve one or more quality characteristics of the ETL flow.

After the application of all the FCP, a number of nodes and edges is added to the initial graph. This process can be repeated an arbitrary number of times and a new Graph is created every time.

It is apparent that the complexity of this analysis is factorial to the size of the graph. Thus, manual configuration of the ETL flow appears inefficient and error-prone, being dependent not only on the users' cognitive abilities but also on characteristics and dynamics of the flow that are hard to predict. Therefore, the need for defining adequate automated mechanisms and heuristics to produce and explore alternative designs and to optimize the ETL flow is evident.

3. SYSTEM OVERVIEW

In [5] we have presented an architecture for user-centered, declarative ETL (re-)design. **POIESIS** is an implementation of the *Planner* component of that architecture. The main functionality of this component is the automatic application of Flow Component Patterns (FCP) on an existing ETL process flow and the architecture of our approach can be seen in Fig. 3. **POIESIS** takes as input an initial ETL

flow and user-defined configurations. Utilizing an existing repository of FCP models, it generates patterns that are specific to the ETL flow on which they are applied. Thus, it produces alternative ETL designs with different FCPs and varying distribution of them on the ETL flow, while keeping the data sources schemata constant. It also estimates defined measures for various quality attributes and illustrates the alternative flows, as well as the corresponding measures to the user through an intuitive visualization.

The internal representation of the FCPs is in the same format as the process flow on which they are deployed. Thus, they can be considered as additional flow components which are positioned at valid application points of the process flow. For example, the *FilterNullValues* pattern is itself an ETL flow consisting of only one operation — a filter that deletes entries with null values from its input. When the *FilterNullValues* pattern is deployed on the initial ETL flow, it is interposed between two consecutive operations. The *FilterNullValues* ETL flow is then configured according to the properties and characteristics of the initial ETL flow as well as the exact application point, ensuring the consistency between data schemata, run-time parameters etc. The same idea is generalized for more complex FCPs or for their more elaborate implementations (e.g., data enrichment additionally to data removal in the described example). In those cases, more detailed configurations might be required to be predefined, such as the access points and data models of additional data sources and processing algorithms of operations.

Our main drivers throughout the development of this component have been the objectives of extensibility and efficiency. In this direction, we followed a modular design with clear-cut interfaces and we employed well-known object-oriented design patterns. The model that was used internally to represent the ETL process flow and allow for its modifications was the ETL flow graph. Each node of this graph represents an ETL flow operation and each directed edge represents a transition from one operation to a successor one.

As a consequence, one strong point of our implementation is that it allows for the definition of custom, additional FCPs, tailored to specific use cases. The applicability of a FCP on the complete ETL flow or some part of it, is decided based upon specific conditions that form the applicability prerequisites, such as the presence or not of specific data types in the operation schemata (e.g., numeric fields in the output schema of preceding operator). Each FCP is related to a particular set of prerequisites that have to be satisfied conjunctively to determine a valid application point. Apart from these strict conditions, there are also *heuristics* to determine the fitness of FCPs for different parts of the ETL flow. For example, according to such heuristics, the addition of a checkpoint is encouraged after the execution of the most complex operations of the ETL fow, in order to avoid the repetition of process-intensive tasks in case of a recovery. Similarly, the application of FCPs related to data cleaning is encouraged as close as possible to the operations for inputing data sources, to prevent cumulative side-effects of reduced data quality. Thus, as opposed to manual deployment, our tool guarantees that all of the potential application points on the ETL flow are checked for each FCP and it can be customized to select the deployment of patterns based on custom policies based on different heuristics.

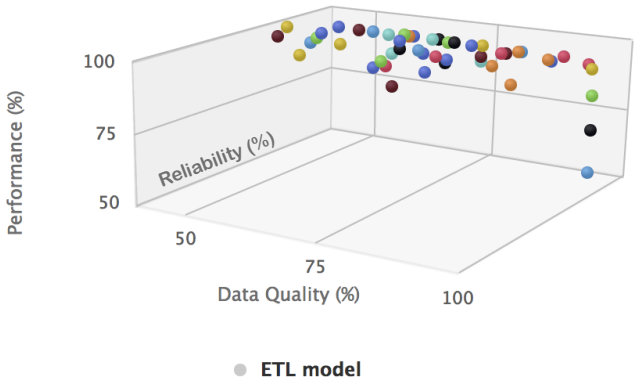


Figure 4: Multidimensional scatter-plot of alternative ETL flows

What is unique about **POIESIS** is that the redesign process takes place in an iterative, incremental and intuitive fashion. A large number of alternative process designs is automatically generated and these can be instantly evaluated based on quality criteria. Moreover, through a highly interactive UI, the user at any point can interact with a visualization of the ETL process and the estimated measures for each of the alternative designs.

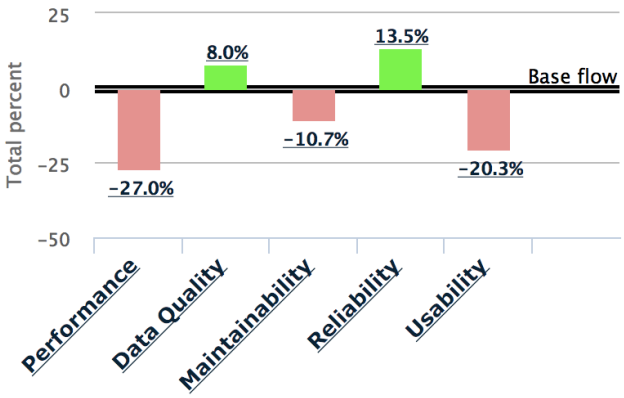


Figure 5: Relative change of measures for an ETL flow, compared with the initial flow as a baseline

The first step is to import an initial ETL model to the system. This model can be a logical representation of the ETL process and we currently support the loading of xLM [7] and PDI[1], but more options will be available in a future version. Subsequently, the user can select the preferred processing parameters, i.e., choose which FCP can be considered in the palette of patterns to be added to the flow, and select the deployment policy for the patterns. It is important to notice at this point that the user can configure the various patterns and even extend them to create custom patterns for future use. The same also stands for the deployment policies, which can be configured according to the user-defined prioritization of goals, as well as the set of constraints based on estimated measures.

Next, after generating and applying relevant FCPs on the ETL flow, the Planner presents to the user a set of potential

[1]`http://community.pentaho.com/projects/data-integration/`

designs in a multidimensional scatter-plot visualization (see Fig. 4), together with quality measures (by clicking on any point on the scatter-plot). The scatter-plot points presented to the user are only the Pareto frontier (skyline) of the complete set of alternative designs, based on their evaluation according to the examined quality dimensions, where larger values are preferred to smaller ones. For example, considering the quality dimensions shown in Fig. 4, for one design *ETL1*, if there exists at least one alternative design *ETL2* offering the same or better *performance* and *data quality*, and at the same time better *reliability*, then *ETL1* will not be presented to the user.

The presented measures (see Fig. 5) show on a bar-graph the relative change on the metrics for each quality characteristic, denoting the estimated effect of selecting each of the available flows, compared with the initial flow. Apparently, the processing and analysis of the alternative process designs is a process intensive task, mainly due to the large number of alternative flows that have to be concurrently evaluated. Therefore, we employ Amazon Cloud[2] elastic infrastructures, by launching processing nodes that run in the background and enable system responsiveness.

When the user selects (clicks on) any of the bars on the measures graph, the corresponding composite measure "expands" to more detailed measures, providing the user with a more in-depth monitoring view. Based on measures and design, the user makes a selection decision and the tool implements this decision by integrating the corresponding patterns to the existing process flow. These patterns are in the form of process components and the Planner carefully merges them to the existing process [2]. Subsequently, new iteration cycles commence, until the user considers that the flow adequately satisfies quality goals [4].

4. DEMO WALKTHROUGH

In the demonstration of **POIESIS** we will use two initial ETL processes based on the TPC-DS[3] and TPC-H[4] benchmarks. These processes contain tens of operators, extracting data from multiple sources. Their logical representation in *xLM* format will be loaded in the system and the automatic addition of Flow Component Patterns in different positions and combinations on the initial flows, will result in thousands of alternative ETL flows, with different quality characteristics.

Using these processes as input data to our system, we will show the capabilities of our tool in an interactive demo, consisting of the following parts:

P1. In the first part of the demo, users will interact with the visualizations of our tool's GUI. In particular, they will be able to scroll over/click on any point on the scatterplot that depicts alternative ETL flows on a multidimensional space of different quality characteristics. By selecting one point — corresponding to one ETL flow — the process representation and the measures for this flow will appear on the screen. Users will then be able to view details about the ETL flow, as well as click on any measure so that it expands to more detailed composing metrics.

P2. The second part aims at illustrating how the processing parameters can be configured in order to produce different collections of alternative flows. Thus, users will be allowed to choose which of the available Flow Component Patterns will be used and which policy will be followed for their deployment.

P3. Finally, users will be guided through defining their own Flow Component Patterns, quality metrics and deployment policies, by extending and pre-configuring the existing ones. They will be able to save their custom processing preferences, adding them to the palette of available patterns for future execution. Examples of the FCPs, which our palette currently includes, together with the quality attribute that they are intended to improve, are as follows:

FCP	**Related quality attribute**
RemoveDuplicateEntries	Data Quality
FilterNullValues	Data Quality
CrosscheckSources	Data Quality
ParallelizeTask	Performance
AddCheckpoint	Reliability

Figure 6: Available FCPs

Acknowledgements. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC).

References

[1] Castellanos, M., Simitsis, A., Wilkinson, K., Dayal, U.: Automating the loading of business process data warehouses. In: EDBT. pp. 612–623 (2009)

[2] Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Integrating ETL Processes from Information Requirements. In: DaWaK. pp. 65–80 (2012)

[3] Russom, P.: TDWI best practices report: Big data analytics. Tech. rep., The data Warehousing Institute (01 2011)

[4] Theodorou, V., Abelló, A., Lehner, W.: Quality Measures for ETL Processes. DaWaK (2014)

[5] Theodorou, V., Abelló, A., Thiele, M., Lehner, W.: A Framework for User-Centered Declarative ETL. In: DOLAP (2014)

[6] Vassiliadis, P., Simitsis, A., Baikousi, E.: A taxonomy of ETL activities. In: DOLAP. pp. 25–32 (2009)

[7] Wilkinson, K., Simitsis, A., Castellanos, M., Dayal, U.: Leveraging business process models for ETL design. In: ER, pp. 15–30 (2010)

[2] `http://aws.amazon.com/ec2/`
[3] `http://www.tpc.org/tpcds/`
[4] `http://www.tpc.org/tpch/`