IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ETL PROCESS DEVELOPMENT USING A MODEL-DRIVEN FRAMEWORK

[1]Author: Madduru Samba Sivudu, research scholar at the department of Computer Science & Engineering at Sri Satya Sai University of Technology & Medical Sciences, Sehore-MP.

[2]Author: Dr. Pankaj Kawadkar, Professor at the department of Computer Science & Engineering at Sri Satya Sai University of Technology & Medical Sciences, Sehore-MP.

Abstract

Since they supply the expert information with the expected incorporated and accommodated information from changed and dissipated information sources, ETL strategies are the spine part of an information distribution center. In any case, the development of an ETL interaction, particularly the plan stage, is as yet viewed as a tedious undertaking. This is expected fundamentally to the reality thatETL activities are regularly evolved in light of a particular innovation from the beginning. As a consequence, it's challenging to interchange and reuse techniques and best practises across projects usingvarious technologies We have done our best to the best of our abilities. There has been no attempt to coordinate the development of ETL processes by presenting a standard and integrated development plan. To address this issue, this article introduces a paradigm for model-driven ETL process creation. Our framework has two advantages: (i) utilization of merchant autonomous models for a uniform plan of ETL processes in light of the expressive and notable BPMN, and (ii) A substantial stage might be made via naturally changing over these models into the seller explicit code expected to do an ETL methodology.

Keywords:ETL, BPMN, Design, Model-Driven Engineering.

1 INTRODUCTION

an information stockroom is a "assortment of arranged, subject-situated data sets intended to assist with choice making"[2]. It is feasible to combine functional information into an assortment of vital pointers, or measures, that might be considered by tomahawks of interest, or aspects, by consolidating various information sources in the information distribution center. The permit to make computerized or print duplicates of all or part of this work for individual or study hall use is hence conceded without charge, given that duplicates are not produced or appropriated for benefit or business benefit and remember this notification and the entire reference for the first page..page. In any case, to replicate, republish, transfer to servers, or ship off mailing records: Quick and proficient business reports might be created from functional information sources on account of an information stockroom. The ETL (Extraction/Transformation/Loading) part is the foundation part that gives every one of the fundamental information to the information warehouses. The process of transformation and loading. As previously mentioned, most data warehouses employ a variety of disparate and dispersed data sources, which ETL methods enable to bring together. should consequently deal with these issues and guarantee that the data

warehouse's expected data quality is met.For starters, ETL process creation is time-consuming, complicated, and prone to failure. When it comes to the cost of a data warehouse project, the creation of ETL procedures is the most time-consuming and resource-intensive. Data quality and metadata and auditability standards are becoming more stringent, which makes integrating solutions more difficult, according to this research.

Since ETL processes are as yet planned in view of a particular seller apparatus, there are a number of drawbacks to the rising complexity of these procedures. Because of the time and effort required to integrate the new platform concept and the underlying languages, these technologies have a steep learning curve. ETL design utilising a vendor-specific tool is often done via a Graphical User Interface (GUI). (GUI) Thus, the design is almost always private. There are no processes in place to build the executable in a smooth way from a different platform since it can only be done inside the tool. In the past, there have been attempts to alleviate the situation. Consider building ETL procedures in a conceptual modelling stage.vendor-agnostic way [6, 15]. These suggestions help to improve the creation of ETL procedures by allowingassisting the designer and documenting the ETL methodstasks. However, they lack efficient techniques for producing vendor-specific code for ETL execution.transforming a method into a tangible platformThe first attempts to ameliorate this problem [6, 15] include a conceptual modelling step for designing vendor-independent ETL methods. These ideas aid in the creation of ETL processes by allowing for the documentation of ETL procedures as well as the support of designer duties. They, however, lack the ability to write vendor-specific code that can be used on a real platform to do ETL. An MDD paradigm for ETL procedures is presented in this research to overcome this problem. Robotized age of seller explicit code for some, stages is a significant point of this structure's plan. A reasonable portrayal of our methodology and the manner in which it incorporates with a current ETL stage might be displayed in Figure 1.

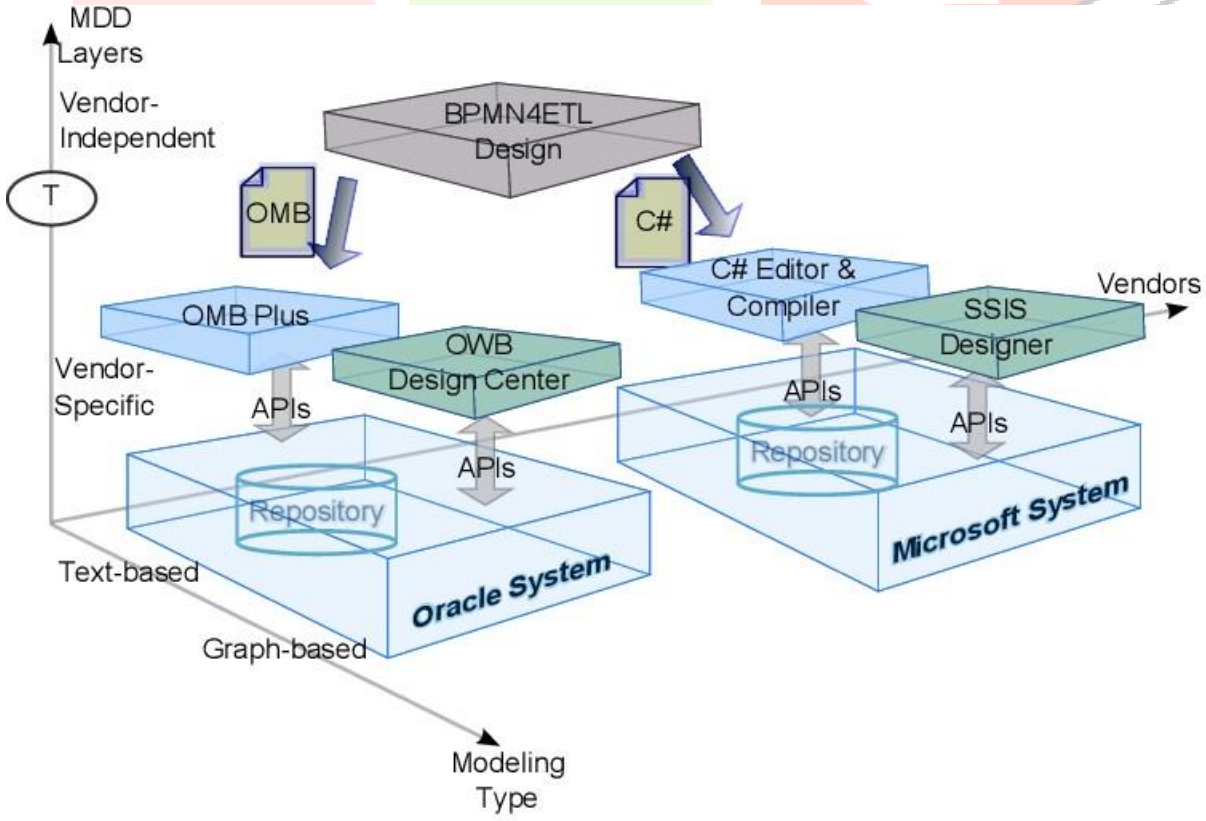

Figure 1: MDD approach for ETL process development

Platform-independent BPMN4ETL is used to generate ETL process models based on the Business Process Model Notation (BPMN). (BPMN). An Oracle Warehouse Builder (OWB) or SQL Server Integration Services (SIS)-specific solution may be built using this architecture (SSIS). In addition to typical graphical languages, these systems frequently allow developers and maintainers of ETL processes to use a fourth-generation programming language (4GL). Oracle Metabase or OMB are two examples of 4GLs that may be procedural or declarative. When a vendor-independent model and this vendor-specific code need to be transformed, OMG's standard model-to-text transformations2 are employed. Following is a breakdown of how we came up with the present structure. Our framework's vendor-independent component is based on a modified version of the BPMN4ETL metamodel for creating ETL processes [1]. To begin with, the ETL component's model-driven methodology was adapted from a broad data warehouse approach [4] to a specific and useful implementation. BPMN and MDD technologies, which allow for the design of any sort of ETL process, as well as the ability to automate any executable code, drive our framework. The following is a breakdown of the findings of the present research. ETL methods may now be created in a neutral metamodel, freeing designers to concentrate on their core competencies.

1.1 Reusability: The ETL process architecture may be utilised regardless of the ETL technology employed. This is a significant advantage since it saves wasting time implementing the same procedure several times. Creating ETL processes that are tailored to the specific requirements of different departments within the same firm may easily be shared. Also, the capacity to pre-characterize specific ETL subprocesses for regular utilization all through the association, no matter what the innovation used, is another tempting element. Subprocesses for a company's data might include the calculation and cleaning of trends for common strategic indicators.

1.2 Interoperability: Interoperability and data exchange across ETL processes may be made easier by combining diverse paradigms of ETL process models into a single design. A progression of model-to-message changes may likewise be done naturally to make code for various ETL apparatuses from models in our model-driven philosophy. A variety of benefits may be attributed to these changes: Technology-independent transformation templates are used to organise transformations into repeatable, repeatable patterns of behaviour. By gathering the relevant code for each template, we can easily add more target ETL tools to our framework. There is no longer a need for professional knowledge in this procedure because of the introduction of semi-automation.

• Validation: Before executing the produced code, the user has the opportunity to inspect and improve it.

• Performance: Picking the right instrument for the gig is basic, since not all incorporation issues are best tended to by similar arrangement of devices.

Next, each module may be implemented in its own tool. Additionally, modularity makes it possible to distribute work across many engines, which, when used in conjunction with the validation asset described below, may increase execution speed.

• Code quality improvement: when creating ETL procedures, the framework should allow for code augmentation by incorporating best practises and lessons gained.

For the remainder of this essay, it is set out as follows. It's time to get into the nitty gritty of the proposed MDD architecture in Section 3. We suggest a methodology for designing model-to-text conversions that is vendor-independent. Then, in Section 5, we illustrate how to design and apply this transformation paradigm for the Oracle platform. Section 6 wraps up the paper and looks ahead to future projects.

2 FRAMEWORK BASED ON MODELS

As previously stated, the current effort uses a Model-Driven Development (MDD) strategy to design ETL processes. Thus, the plan and execution phases of the ETL interaction might be achieved all the more quickly by utilizing this method to put together the various parts of this structure.

2.1 Framework based on MDD

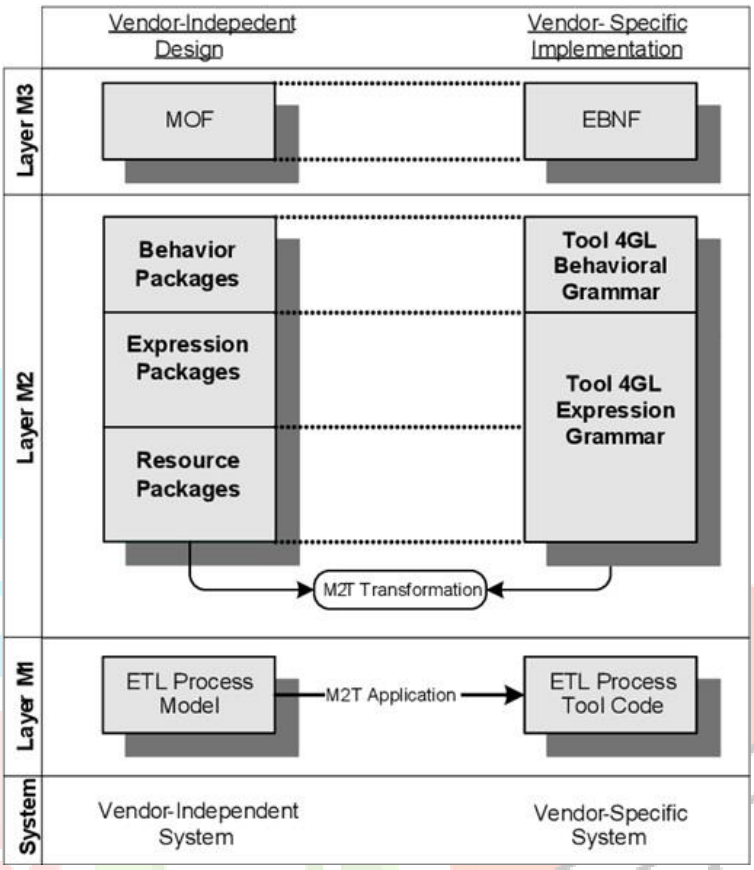

Figure 2: MDD layers for the ETL development framework

MDD is a product improvement approach in which complex models are worked before any source code is composed. Utilizing the MDD method (see Meta-Object Facilities (MOF)3), there are four significant layers characterized: the Model Instance layer (M0), the Model layer (M1), Meta-Model layer (M2), and Meta-Metamatmodel layer (M3). (M3). For the ETL cycle plan and execution, the Model Instance layer (M0) addresses this present reality framework. A merchant free graphical UI (GUI) and a seller explicit ETL motor (ETL) are instances of this. There are various changes applied to the ETL Process Model (M1) to produce the ETL Process Code. Subsequently, we've continued on from origination to execution. The Meta-Model layer of the BPMN4ETL metamodel portrays ETL designs during plan and a 4GL language during execution (M2). With regards to planning and carrying out, the M3 Meta-Model (M3) compares to the MOF meta-metamodel and Backus Naur Form separately (BNF).

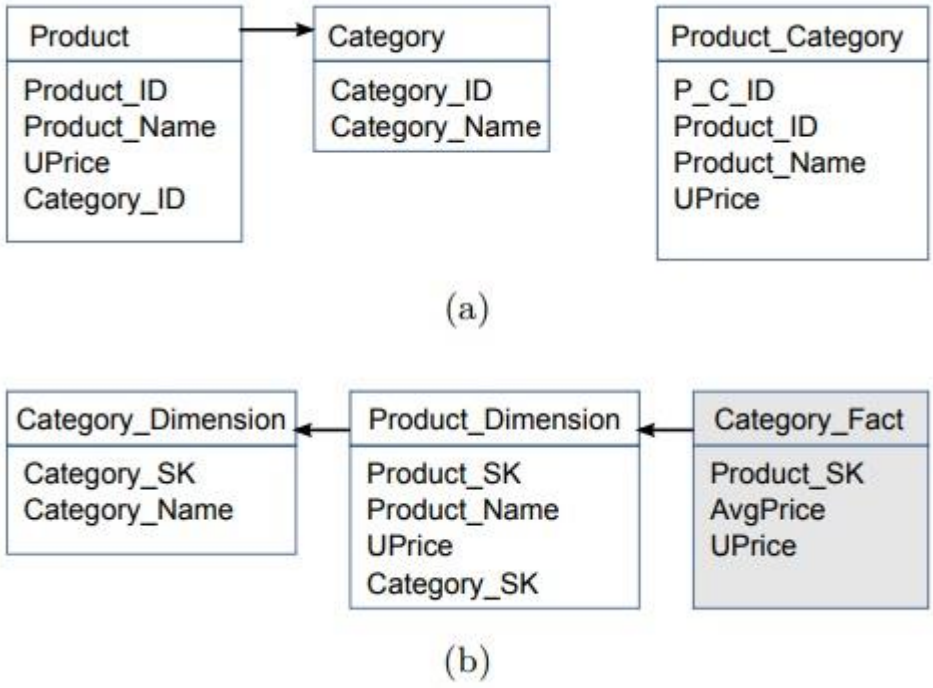

Figure 3: (a) Data source schemas; (b) Data warehouse target schema.

Because these layers establish the ETL semantics, we'll focus on them in the next sections. These two tiers are described using Fig. 3, which shows a small data warehouse sample. For this situation, a data set and a level document are used to take care of the information distribution center (see Fig. 3). The Common Warehouse Metamodel4 indicates the use of social and recordset ideal models for tables and documents, individually. Each thing in the information base has a comparing classification in the data set's Product and Category tables. Items and their classifications are consolidated into a solitary blueprint in the Product Category level record source, which conveys identical data. Item orders are portrayed in Fig. 3b of the objective information stockroom as a straightforward truth table Category Fact. The item pecking order is comprised of both the Category Dimension and Product Dimension tables. Design that is not dependent on a single vendor.

This part makes sense of the seller free plan side of the structure addressed in Fig. 2. The BPMN4ETL metamodel's constructs are used to develop the ETL model in this section.

2.2 Model BPMN4ETL (M1)

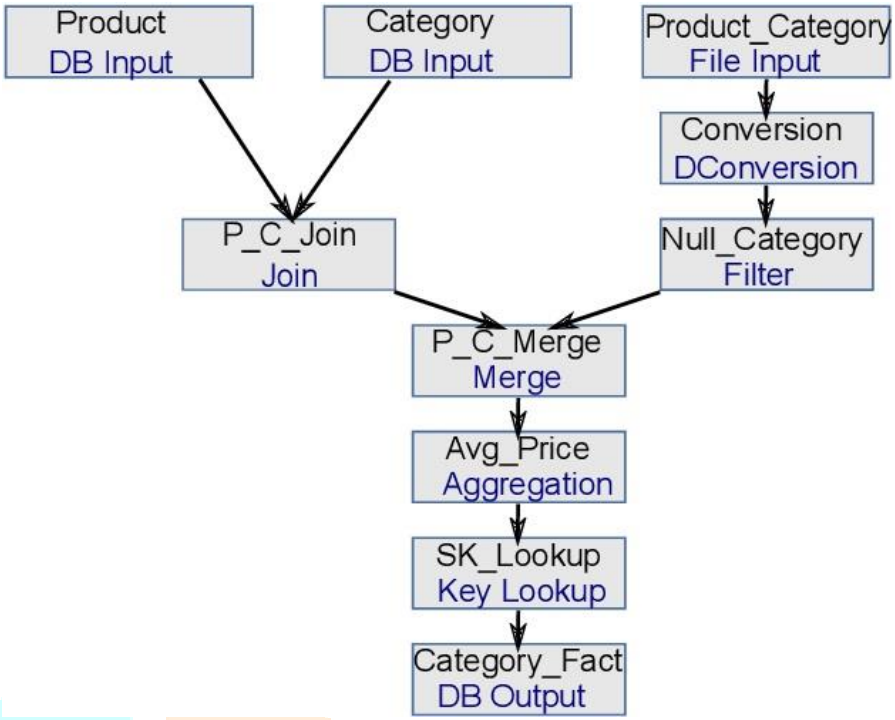

Figure 4: ETL process model for the Category Fact load

ETL, as displayed in Figure 4, is the cycle by which the required information from the different information sources is taken care of into an example information stockroom. There are many strides in this interaction, beginning with information extraction (e.g., DB Input) and document input (e.g., Filter, Join), and closing with information stacking (e.g., DB Output) into the vault. Because of the information in the classification table, we should utilize the VDerivation errand to make another field utilizing variable inference. Then there's the item, and afterward there'sData from various classes can be consolidated. Extraction of data from a file, on the other hand, creates fields of character data. As a result, translating various character types into their equivalents is a common practise when working with files. Therefore, the record ought to be sifted to prohibit invalid Product ID sections since there is no means to check this imperative inside the document, not at all like a table subsequently, consolidating both the information base and the file is conceivable. AvgPrice is the average price used to compute the fact table's target metrics. Finally, a key query is run before the Category Fact table is loaded to ensure that the transformed data and the old data are referentially intact. As a reminder, each assignment may be tailored to suit the needs of the student. by altering its actions in order to accomplish the intended outcome. The extraction and loading operations, for example, are important.a variety of scenarios, such as loading tactics andautomaticFurthermore, the transformations are made up of a variety of computations, branching, and merging activities, each of which may be adjusted using a range of parameters such as conditions and computations.input/output cardinalities, and Input and output numbers For the purpose of simplicity, these combinations have been omitted from Fig. 4.

2.3 Metamodel BPMN4ETL (M2)

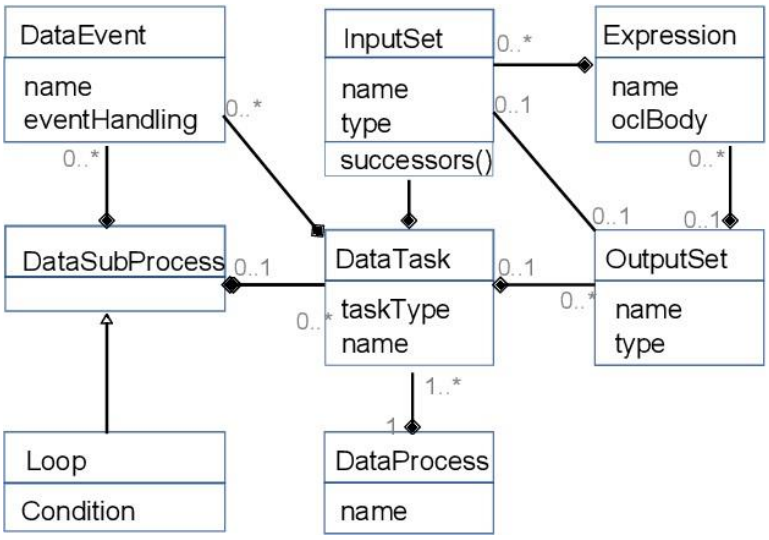

Figure 5: Excerpt of the data process package

Model ETL process in Figure 4 depends on the BPMN4ETL metamodel, as recently demonstrated (see Figure 5). The metamodel depends on BPMN, a true standard sentence structure for communicating business processes. Regardless of the technology used, BPMN offers a standardised and simple-to-understand framework for documenting business processes. Many fundamental features of business processes may be described using BPMN's event and container models. These models enable business processes to communicate with the outside world and track their boundaries. A variety of packages make up our BPMN metamodel. Using numerous DataProcesses, the ControlProcess records the whole ETL process of creating a data warehouse. Data processing may be seen in Figure 5, which shows a section of it.

Frequently, the entire information stream from a bunch of information sources to an objective information stockroom is displayed in a DataProcess outline. Several DataTasks from different taskTypes are required. variableDerivation, a join, a filter, a merge, and so on With the DataEvent class, users may customise their exceptions. Each data job requires a different approach. Furthermore, the following is a subsequence ofSubprocesses can be used to arrange data activities that have the same goal or comparable characteristics.The DataSubProcess class is responsible for this. both incoming and outbound Using the InputSet and OutputSet classes, you can express the incoming and exiting data activities. When a data task processes its input, it applies the expressions it has stored in this class. a collection of outputs is needed to create its output Additional metamodel packages, such as Condition, Computation, and Query, explain various expressions. Other programmes are used in ETL methods to capture crucial data storage properties. For non-tireless information in the ETL interaction, the Intermediate Data Store (IDS) bundle gives the stagging region to the information assets that are extricated/stacked.

3 TRANSFORMATIONS

It is possible to automatically construct ETL code by using transformations. These changes are described here as statements at the M2 layer in order to be done at M1 layer, which are metamodel and grammar compatible. ETL code generators may be guided by abstract patterns for such transformations, which we illustrate in this paper. The OMG's model-to-text (M2T) standard is used to represent the transformations.

3.1 Transformations from M2T

The input model components are coded using M2transformation T's templates. Metamodel-related code is included in each template's code segments. Static or dynamic code may both be used in this application. During execution, static code is really re-created. As a result of applying metamodel concepts, OCL

expressions are linked to dynamic code. The input model is used to compute these expressions during execution. An output file is constructed by iterating over the model components sequentially and appending the template code for each element to the output file.

Consequently, templates are utilised to convert between the BPMN4ETL metamodel and the OMB language. For example, the ETL tool's template code depends on this. A theoretical level for change formats might be helpful since it supplies the fundamental layouts and makes sense of how they're dealt with in like manner across ETL frameworks, as we'll illustrate.

3.2 Patterns of Transformation

Since ETL code for some, software tools follows a comparable way, we depicted examples of theoretical layouts, like the one displayed in Fig. 6. The sequence of template building may be better understood with the help of these patterns. The tool may then be used to generate the code semi-automatically. Utilizing the article situated worldview, each theoretical layout might be absorbed to a theoretical class, which can then be carried out in code. At long last, these examples may be changed to represent contrasts in programming dialects.

The abstract structure illustrated in Fig. 6 demonstrates how an ETL model might be used to generate new code. Other patterns deal with the upkeep and management of this code. The steps in this pattern algorithm are as follows: Core control process template addNewCP is built first, then its components are added incrementally. Analyzing the data is an essential step. For every ongoing information process, another vacant format addNewDP and all association layouts comparing to the connected assets addConnection are created as a result. This template is used for every resource that is part of the foreach loop's data process. Following the info model, the information task layouts addDTask are acquired in view of their sorts and associated with each other matchDTasks.

4 TRANSFORMATIONS IN ACTION

This segment makes sense of the Eclipse-based execution of our system, including the code age. It draws attention to a crucial area of the transformation code. OMB code may be generated using the ETL methodology shown in Figure 4. There are several tools in the Eclipse software development environment. a plug-in system that can be expanded. It includes the Eclipse Modeling Framework (EMF), which includes modelling, inter-model communication, and other model-driven programming features.code creation and transformations EMF is a more exact term.enables the creation of models and metamodels through the use ofEcore's tools. In practise, metamodels are built with the help ofEcore's meta-metamodel, which is the application ofEclipsed MOF. It is possible to create new metamodels using these metamodels.create the genmodel, which is a Java class library .

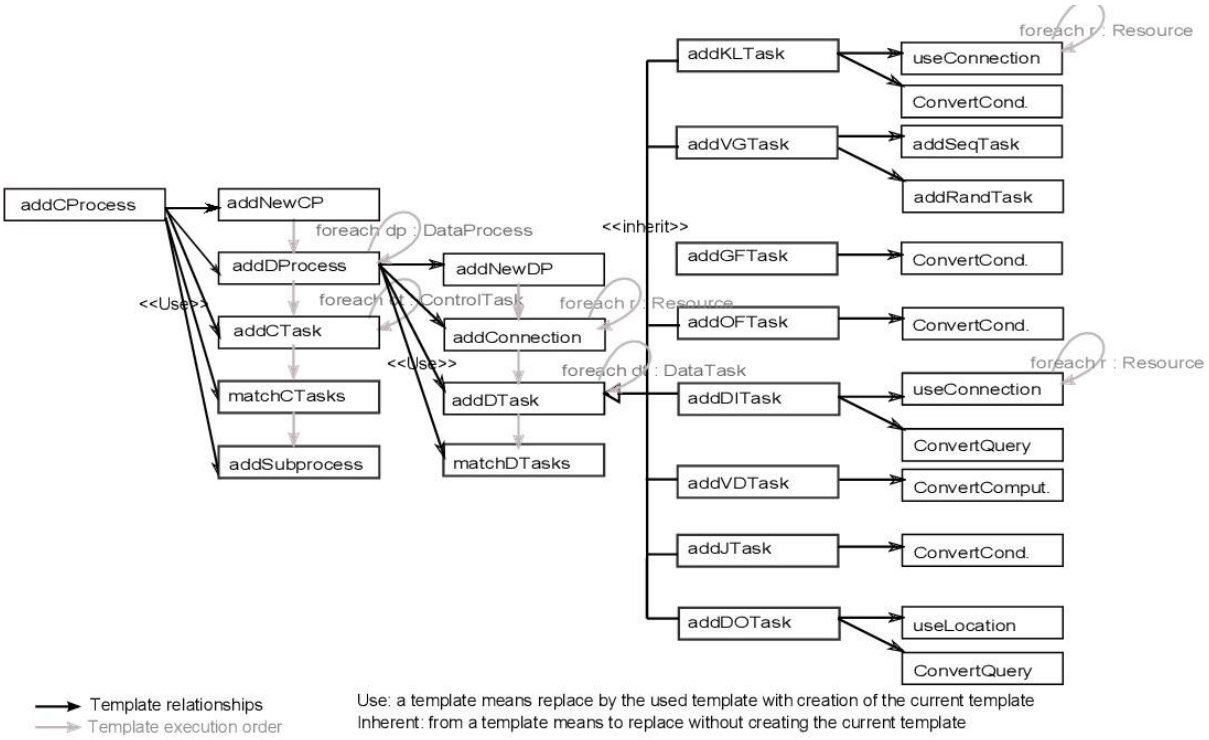

Figure 6: Abstract templates pattern for ETL process creation

5 CONCLUSION

ETL methods are still designed and modelled using vendor-specific technology, despite the fact that information distribution centers have been around since the mid 1990s. making it possible for them to be defined in line with the schemas described Concretized platforms Therefore, they must be created, implemented and maintained in line with the platform they are intended to be used for. ETL procedures may now be automatically developed in commercial applications platforms using a methodology that is vendor independent and vendor agnostic for the first time. The expressiveness of our language makes this feasible. Model-driven development tools from Eclipse and Acceleo are combined with a BPMN-based metamodel to generate code. Due to a lack of space, we have focused on Oracle's solution, however our approach considers Oracle and Microsoft as typical implementation and execution tools. Additionally, it offers a few additional methods for developers to aid them in generating code generators for newer platforms. Design and execution of the ETL process are currently covered by our framework. In the future, this framework might be expanded to include the analytical phase as well, which would be a significant accomplishment. Our method should improve the quality of the resulting code by: I improving the transformation implementation utilising certain existing knowledge-based techniques.

REFERENCES

[1] Z. El Akkaoui and E. Zim´anyi.Defining ETL worfklows using BPMN and BPEL. In Song and Zim´anyi [11], pages 41–48.

[2] W. Inmon. Building the Data Warehouse.Wiley, 2002.

[3] S. Luj´an-Mora and J. Trujillo. Physical modeling of data warehouses using UML. In I. Song and K. Davis, editors, Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP, DOLAP'04, pages 48–57, Washington, D.C., USA, Nov. 2005. ACM Press.

[4] J. Maz´on and J. Trujillo.An MDA approach for the development of data warehouses. Decision Support Systems, 45(1):41–58, 2008.

[5] A. Simitsis. Mapping conceptual to logical models forETL processes. In I. Song and J. Trujillo, editors, Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP, DOLAP'05, pages 67–76, Bremen, Germany, Nov. 2005. ACM Press.

[6] A. Simitsis and P. Vassiliadis. A methodology for the conceptual modeling of ETL processes. In J. Eder, R. Mittermeir, and B. Pernici, editors, Workshop Proceedings of the 15th International Conference on Advanced Information Systems Engineering CAiSE'03, CEUR Workshop Proceedings, pages 305–316, Klagenfurt/Velden, Austria, 2003. CEUR Workshop Proceedings.

[7] A. Simitsis and P. Vassiliadis. A method for the mapping of conceptual designs to logical blueprints for ETL processes. Decision Support Systems, 45(1):22–40, 2008.

[8] D. Skoutas and A. Simitsis. Designing ETL processes using semantic web technologies. In I. Song and P. Vassiliadis, editors, Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, DOLAP'06, pages 67–74, Arlington, Virginia, USA, Nov. 2005. ACM Press.

[9] D. Skoutas and A. Simitsis. Ontology-based conceptual design of ETL processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems, 3(4):1–24, 2007.

[10] D. Skoutas, A. Simitsis, and T. Sellis. Ontology-driven conceptual design of ETL processes using graph transformations. In Journal on Data Semantics XIII, number 5530 in LNCS, pages 122–149. Springer, 2009.

[11] I. Song and E. Zim´anyi, editors. Proceedings of the12th ACM International Workshop on Data Warehousing and OLAP, DOLAP'09, Hong Kong, China, Nov. 2009.ACM Press.

[12] C. Thomsen and T. Pedersen.pygrametl: A powerful programming framework for extract-transform-load programmers. In Song and Zim´anyi [11], pages 49–56.

[13] V. Tziovara, P. Vassiliadis, and A. Simitsis.Deciding the physical implementation of ETL workflows. In I. Song and T. Pedersen, editors, Proceedings of the 10th ACM International Workshop on Data Warehousing and OLAP, DOLAP'07, pages 49–56, Lisbon, Portugal, Nov. 2007. ACM Press.

[14] P. Vassiliadis, A. Simitsis, and E. Baikous.A taxonomy of ETL activities. In Song and Zim´anyi [11], pages 25–32.

[15] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis, and S. Skiadopoulos.A generic and customizable framework for the design of ETL scenarios. Information Systems, 30(7):492–525, 2005.

[16] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for ETL processes. In D. Theodoratos, editor, Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, DOLAP'02, pages 14–21, McLean, Virginia, USA, Nov. 2002. ACM Press.

[17] L. Wyatt, B. Caufield, and D. Pol. Principles for an ETL benchmark. In R. Nambiar and M. Poess, editors, Proceedings of the First TPC Technology Conference, TPCTC 2009, number 5895 in LNCS, pages 183–198, Lyon, France, Aug. 2009. Springer.