AUTOMATIC GENERATION OF DATA MERGING PROGRAM CODES

Keywords: Data Merging Meta-model, Data Integration, Model Driven Engineering, Model Driven Data Integration, Automatic Model Transformation, Automatic Program Code Generation.

Abstract: Data merging is an essential part of ETL (Extract-Transform-Load) processes to build a data warehouse system. To avoid rewheeling merging techniques, we propose a Data Merging Meta-model (DMM) and its transformation into executable program codes in the manner of model driven engineering. DMM allows defining relationships of different model entities and their merging types in conceptual level. Our formalized transformation described using ATL (ATLAS Transformation Language) enables automatic generation of PL/SQL packages to execute data merging in commercial ETL tools. With this approach data warehouse engineers can be relieved from the burden of repetitive complex script coding and the pain of maintaining consistency of design and implementation.

1 INTRODUCTION

A Data Warehouse (DW) is a collection of integrated subject-oriented databases chosen to support the decision making process (Kimball et al, 2002). Building a DW involves processes that combine data with various formats and present a unified view of the data, extracting data from different sources and cleansing inappropriate data. DW has become a very popular choice for many enterprise systems, such as business intelligence and more enterprise systems data needed to be added to the data warehouse. To support the growing demands of DW development, ETL (Extract-Transform-Load) processes have supported a systematic framework for the extraction of the data from heterogeneous data sources, and its transformation; cleansing, converting, and loading them into the data warehouse. According to (March et al, 2007), ETL processes are not only important for design and maintenance of DW but also key contributers to the success of DW projects. Various approaches have been proposed in order to improve the ETL engineering.

Applying Model Driven Engineering (MDE) to ETL processes is one of the promising approaches. The approach reduces the complexity of ETL design by decoupling data and meta-data, and improving communication between domain experts and developers through the use of graphical model design. It also increases productivity due to the reduced amount of handcrafted coding and of code rework at the maintenance phase. This is achieved by first defining an abstracted model, then transforming it into program codes. Thus ETL working codes can be derived and maintained from well defined ETL models, described in abstracted level and gradually mapped into concrete level.

A number of these MDE approaches have been proposed either as a UML extension or as their own graphical notation for conceptual ETL data mapping design (Mora1 et al, 2004), (March et al, 2007). A meta-model for process has also been proposed to apply MDE to the workflow and scheduling in DW (Bohm et al, 2008). Muñoz et al have proposed not

just a design model, but a whole conceptual data integration framework (Muñoz et al, 2009). However most of these works address the whole ETL process and do not consider the problems which need to be addressed in each DW building phase. Furthermore, they have rarely demonstrated how to integrate industrial standards in their approaches. A more detailed review of previous works is dicusssed in section 6.

In this paper, we mainly focus on a model driven data merging approach to address problems in the data merging domain. Based on a real case study of a DW development project we propose a data merging system to generate executable merging codes from conceptual design. A Data Merging Meta-model (DMM) was proposed for design of merging models at conceptual level. Common Warehouse Meta-model (CWM), an industrial standard for data warehouse modeling, was also used for design of merging models at physical level (CWM, 2008). The proposed system provides transformation of DMM into CWM. By using the standard, it allows not to be bound to a particular tool but instead the use of any DW development environment. Through this system data warehouse engineers can develop a unified data schema by creating abstractions that help them program in terms of their design intent rather than the underlying computing environment. The executable data merging codes can be obtained from CWM merging models as ETL tool vendors provide code generation from CWM.

The rest of this paper is structured as follows: Section 2 presents model driven data warehousing, providing both the general approach and ours. The proposed data integration framework and merging meta-mode are also described. Section 3 shows our implementation works, illustrating the system architecture, target meta-model, CWM and, transformation rules. A case study to which we applied the proposed model driven approach is introduced in Section 4. Finally related works are given in section 5 and conclusions in section 6.

2 MODEL DRIVEN DATA INTEGRATION

The whole data warehousing processes can be divided into four phases; (1) analyzing and understanding data in the different data sources, (2) preparing and collecting data into staging area, usually one physical platform, (3) combining data through data cleansing, merging, and transformation, which covers most ETL processes, (4) finally customizing data into different presentation according to application purposes (Rahm et al, 2000). Through each data process, data sources are gradually reformatted and moved into target schemas. The processes can be easily executed and maintained by controlling data from models within a model driven approach.

In this section, we introduce general model driven approach with two representative methods and discuss our own approach which is implemented utilising the general approach.

2.1 General Model Driven Approach

Model Driven Engineering (MDE) is a software engineering methodology that uses models as primary artefacts to drive the entire development process through model transformations. Over the years model based development has gained rapidly increasing popularity across various engineering disciplines. The representative two approaches are presented in this section.

2.1.1 Model Driven Architecture

Model Driven Architecture (MDA) is the first initiative of MDE which uses UML as modeling language, OCL (Object Constraint Language) and, QVT (Query/View/Transformation) as model transformation language (OCL, 2008), (Kleppe et al, 2003). It is launched by the Object Management Group (OMG) in 2001 and mainly focuses on forward engineering, such as producing codes from abstract and human-elaborated modeling diagrams, separating design from architecture. The design addresses the functional requirements whilst the architecture provides the infrastructure addressing non-functional requirements like scalability, reliability and performance. Decoupling design and architecture allows system developers to choose the best and most fitting models in both domains.

MDA uses the Platform Independent Model (PIM) which represents a conceptual design to realize the functional requirements. PIM is translated into one or more Platform Specific Models (PSMs) that a computer can run. Accordingly model transformations which support conversion between PIM and PSM are particularly important for the realization of MDA.

Most software development IDEs support MDA by providing UML modeling and code generation from the UML models, but there are many critics that believe UML is too generic to describe domain specific problems. Another direction is to develop

domain specific languages designed to solve common model transformation tasks. Indeed, this approach has been widely taken recently by the research community and software industry. As a result a number of model transformation languages have been proposed (Marcos et al, 2006), (Greenfield, 2004).

2.1.2 Eclipse Modeling Framework

Eclipse is one of the most popular IDEs, providing convenient pluggable architecture. It also provides a meta-meta-model called ecore and its own modeling framework for MDE (Dave et al, 2008). This framework generates the model development environment automatically. Developers can design their own models and transform them into target models once a specific domain model is designed as a meta-model based on ecore.

In addition to this, there are several open source plug-ins that facilitate model driven development based on Eclipse modeling framework with various functionalities. For example, ATLAS Model Weaver (AMW) extends eclipse modeling framework for model to model conversion Macros, 2006). It enables a developer to combine different models together and generate a new model by establishing relationships between models using their weaving meta-model. For model transformation, it also provides a transformation language, called ATL, correspondent to QVT (Query/View/Transformation) of OMG (Allilaire et al, 2006).

2.2 Our Approach

We applied MDA to the whole data integration processes by designing PIM models in each DW development phase. PIM models then transformed into PSM models and real codes. Since existing ETL tools do not provide PIM modeling for data merging, we proposed a data merging PIM meta-model which allows conceptual design and model transformation into existing ETL standard. The ATLAS transformation language and toolkit have been used for the implementation of the transformation.

2.2.1 Data Integration Framework

It is well known that conceptual models (PIM) provide not only guidance on how to integrate actual data but also an automated generation of real code, ready for execution according to MDA viewpoints. In this context, transformations between PIMs and PSMs, and between PSMs and real codes are necessary for each modeling phase of DW. For data integration, it is also required to define and use different models for each data integration phase: data source model, extraction model, merging model and customized model.

In general, modeling starts from the highest abstraction layer and descends to the concrete codes layer. However, most Data Source PIMs and PSMs can be derived from real data sources through reverse transformation as existing data sources have their own schema or structure by which PSM is drafted. Extraction PIMs are usually designed on the basis of Data Source PIMs analysis and transformed into PSMs and program codes in turn later. Merging PIMs are commonly designed after building the data cleansing strategy and then transformed into PSMs and merging execution codes. Based on unified a data model, Customized PIMs are also built in order to present data in a different way. Figure 1 outlines the models and their relationships in different abstraction levels such as different data warehousing phases.

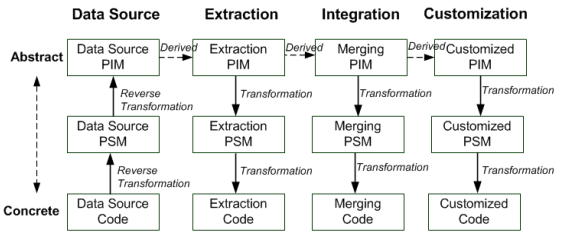

Figure 1: Models and Transformations between the models.

Most of the data modeling tools support reverse engineering that automatically transforms physical data schema into its physical ERD model or its logical UML model as like they do forward engineering for automated transformation of PIM into PSM and into real code. Most of the DW vendors also supply ETL modeling and model interchange mechanism between different ETL platforms and BI systems by implementing the Common Warehouse Meta-model (CWM) specification. However, many researchers have reported that CWM is not sufficient for conceptual modeling since it tightly bounds to a physical layer (Vassiliadis et al, 2002). Furthermore, data processing can not be designed effectively in UML as it is unable to express data mapping, requiring defining relationships between attributes. This problem is solved through using a data merging met-model for conceptual design (PIM) and transforming it into CWM (PSM). In our previous work, we had proposed the conceptual data merging meta-model

and rules to support manual transformation of merging PIM into PL/SQL scripts (Kim et al, 2009). In this work, we discuss how we have implemented these transformation rules proposing a data merging system. The transformation rules have been extended to support automatic conversion of the proposed meta-model into the commercial standard meta-model, CWM.

2.2.2 Data Merging

In this paper, we concentrate on model driven data merging. Data merging in data warehousing includes combining and moving data into target schema as well as creation of new data schema in order to provide a unified view. Data schemas and data combing rules can model entities that describe attributes of each data entity and relationships of the entities. In particular, a data merging model must show how to move data from existing source data entities into new target data entities. Since a data entity is a set of data attributes, not only relationships between data entities but also relationships between data attributes should be addressed for data merging.

Data merging modeling starts from investigating overlapped data from each data source. Once corresponding pairs of duplicated data are identified, a number of design issues lead to concerns including whether to preserve the duplicated data or how to keep data consistency between indirect references as well as direct ones. However, once a decision of how to merge the data is made, the actual merging can be simple repetitive routines in abstraction. The abstractions can be represented as three patterns; Join, Union, and Association. Join keeps all data from one leading data source and copies data, excluding duplicated parts with the leading one from the other data sources. Union combines all data from each data sources without discarding any data. Association only updates relationship constraints between data sources and target. They are described as DMType model elements in the proposed conceptual data merging model.

2.2.3 Proposed Data Merging Meta-model

We propose a Data Merging Meta-model (DMM) to support data merging design in the early stage of DW development. It describes merging models at conceptual level based on UML and rule description. A model includes model elements from different data sources and their relationships. These relationships of meta-data realize data mappings that describe how to move each source data to the target one. Figure 2 describes our DMM.

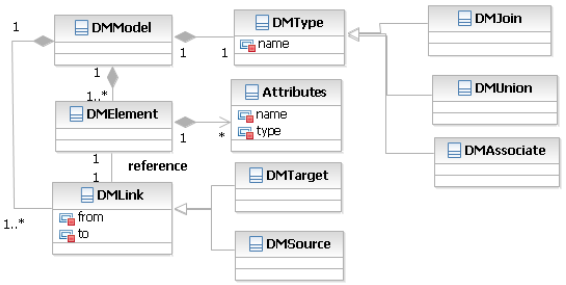

Figure 2: Data Merging Meta-model.

The root element of the model, DMModel, is composed of several elements; DMType, DMElement, and DMLink. Description of each element is following in Table 1.

Table 1: DMM elements.

DM Type	A base model of DMJoin, DMUnion and DMAssociation, which determines the merging method. A data mapping rule script attached on the DMType specified details of data mapping and their order.
DM Join	A type of merging which finds a joint data set of all linked source elements and moves the data into a target element.
DM Union	Moves all data from each source elements to a target element according to their order.
DM Associate	Replaces association of source elements to the target.
DM Element	Represents model elements including both source and target.
DM Link	Shows relationship and directions of data mapping.
DM Source	Inherits DMLink to identify source elements.
DM Target	Inherits DMLink to identify a target element.

Using this model data merging in DW can be designed abstractly; an example of a simple data merging between two meta-data is presented in Figure 3. Here two school model elements from a student record management system and a course marketing system are shown respectively. They contain exactly the same data structure but are differentiated by the reference to the faculty object named CM_Faculty. It means that not only data itself, but also other things such as the object reference and data constraints have to be considered when the two elements are merged. We merged them using DMJoin defining UE_School as a leading data source and describing detailed attribute

mapping as a rule shown below. This rule can be expressed with graphic notation such as an arrow in more advanced graphic editor.

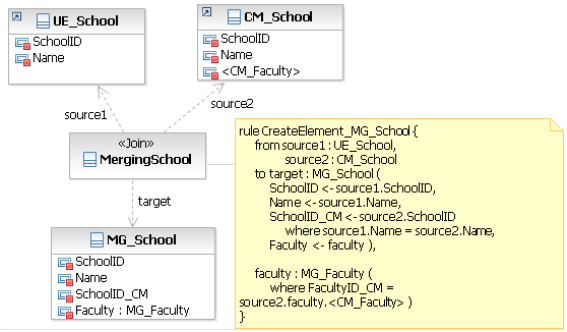

Figure 3: An Example of Data Merging PIM.

The ruleCreateElment_MG_School describes how to map the attributes of source elements; UE_School and CM_School, to the target element; MG_School. The rule has a set of {sources, targets} and the targets have a set of {target element name, a set of attributes mapping}. Attribute mapping is expressed with an arrow directing from a source attribute to a target attribute. In this example, DMJoin moves only overlapped data sets of CM_School. If the merging type is DMUnion, it would move the first source element into a target element on the 'insert' basis and the others on the 'update and insert' basis. All data from UE_School inserted into MG_School then CM_School data updated School_ID attribute of existing data set only if the same Name attribute data is found in existing data. As <CM_Faculty> is an object reference, not only data value but also an object constraint must to be changed. The reference object is changed from CM_Faculty to MG_Faculty in this example.

3. IMPLEMENTATION

In our approach, data merging process is launched by designing a conceptual merging model in DMM. This model is then automatically converted into a CWM model by executing the implemented transformation engine. Then the executable merging program is finally created through importing the generated CWM model into an ETL tool. In this section, we discussed the implementation detail including system architecture, CWM specification and, transformation rules.

3.1 System Architecture

We implemented a data merging system, including the transformation engine based on ATL toolkit and the engine exports generated from the CWM models as file format. All processes and architecture are illustrated in Figure 4.

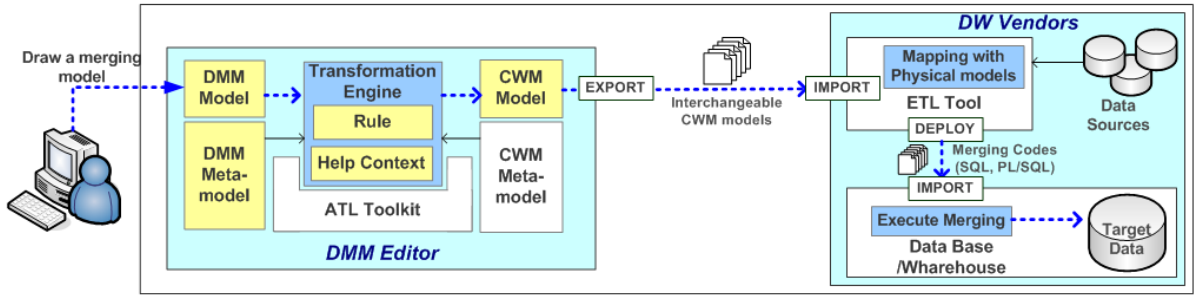

Figure 4: Data Merging System Architecture.

The DMM Editor takes the DMM model as an input and generates the CWM model. This requires both the DMM meta-model and the CWM meta-model, interpreted and deployed as ecore format. Based on these ecore models transformation rules have been implemented. The rule component of the Transformation Engine container consists of rules for mapping of DMM into CWM, the Help Context component comprising of the functions and utilities needed for type checking, condition management etc. Details of transformation rules are presented in Section 3.3.2.

DMM Editor exports the CWM model according to interchangeable CWM model specification which DW vendors can import. The imported model contains both skeletons and logics inside to execute data merging but is not bound with actual schemas of data sources. Therefore the additional work to bind it with physical data schemas and allow synchronization between them is necessary. After this, the merging codes are generated, deployed and executed into the target platform.

3.2 Common Warehouse Meta-model

CWM is a specification describing objects and relationships in the context of data warehousing. Since data warehouses pull in data from many different digital sources, CWM includes a comprehensive set of data models for data structures such as relational databases, flat files, and XML.

OMG announces that MOF bridges the gap between dissimilar meta-models by providing a common basis for meta-models. Consequently, the models described by DMM can be interchanged with the models conform to CWM since both are MOF-conformant.

CWM was designed in line with the aim of providing interchange of all warehouse meta-data that describes all warehouse data element. This includes data sources, transformations, data targets, and all warehouse processing elements including scheduling, status reporting and history recording. Thus the meta-model specification of CWM cover all warehousing areas: from the foundation of data types and type mapping, to the management of the warehouse process and operation. For the entire meta-model, we have only referenced the parts related to data merging. For example, Figure 5 shows relational meta-model of CWM to describe data sources and data targets. It presents the attributes of tables, columns and data types, and the relationships between them (CWM, 2008).

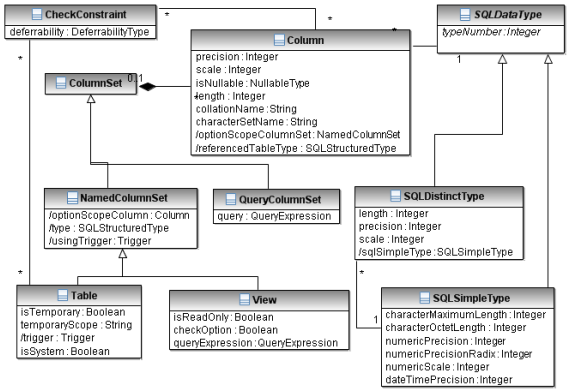

Figure 5: Part of Common Warehouse Meta-model.

3.3 Model Transformation

MDE can be completed through constant model transformations from abstract level to concrete one. As mentioned in Section 2.1, there are several MDE initiatives that suggest their own meta-model and transformation language. (Frédéric et al, 2006) summarizes the main characteristics of representative transformation languages; QVT and ATL, comparing their technology and functionality in architectural view, to helping software developers compare and select the most suitable languages and tools for a particular problem. The reasons we decided to implement ATLAS architecture are: abundant data, steady maintenance, and support of transformation development toolkit, although QVT is considered as an industrial standard in MDA. This section presents ATL and the transformation of DMM into CWM using ATL.

3.3.1 ATLAS Transformation Language

ATL provides both the language for description of model transformations and the toolkit for execution of the model transformations. The architecture for ATL toolkit is shown in Figure 6. It was developed on the top of Eclipse platform, aiming to offer ways to produce a set of target models from a set of source models. Source meta-model and target meta-model should first be defined subsequently target instance model is generated from input source model using ATL. The transformation rule between source model and target should be written in ATL language.

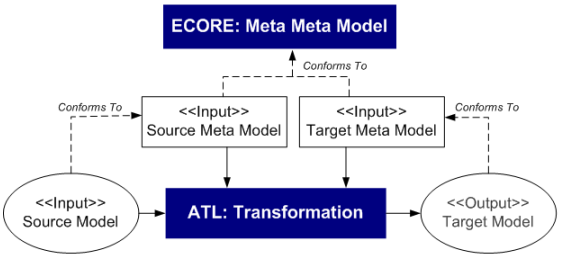

Figure 6: ATLAS Toolkit Architecture.

ATL language is used to create an ATL module that describes and executes transformation in the toolkit. Besides its header, an ATL module is composed of a set of ATL rules. Each rule defines the way of transforming an input element into a target element. A rule is composed of an InPattern and OutPattern. The InPattern declares a typed variable that corresponds to the rule input element. During the execution of the ATL transformation this variable corresponds to the source element currently being matched. The OutPattern declares a typed variable which corresponds to the rule output element. The OutPattern also specifies a set of Binding elements. A Binding describes how a given feature (an attribute or a reference) of the target element is initialized. This initialization must be

specified as an OCL expression (Allilaire et al, 2006).

3.3.2 DMM2CWM Transformation

Converting DMM into CWM means that *DMElements* are mapped to a relational data element. For example, source *DMElement* references existing data table, whilst a target *DMElement* creates new data schema. The full description of transformation rule is listed in Table 2.

Table 2: Transformation Rule.

DM Model	Transformation Rule
DMElement	-If *DMElement* is connected with *DMSource* link, generate a reference to an existing table. - If *DMElement* is connected with *DMTarget* link: create new table schema including primary key and foreign key constraints. -If an attribute of *DMElement* is not a primitive type, change table constraints on foreign key to reference a proper element.
DMUnion	-Create data mappings as much as the number of *DMSource* links. -According to the mapping order in rule script, each data mapping from a source to a target is transformed into each attribute connection between source and target elements in turn. - If attributes of source and target are not of the same type, insert data type change function before mapping data.
DMJoin	-Create a data mapping using *joiner* entity to merge source elements -From rule script, joining conditions and mapping sequence are determined.
DMAssociation	-Change target table schema. -Update target table schema to reference a source table with foreign key constraint.
DMLSource/DMTarget	- No correspondent transformation. Just indicate whether a linked *DMElement* is a source element or a target one.

To automate this model transformation, we implemented a transformation module using ATL. At first we created both input and output ecore models from DMM and CWM in UML. These ecore models are recognized as meta-models of input and output respectively, for the transformation. Then the transformation rules in Table 2 were implemented in ATL language as partly shown in Figure 7. DMElement is converted into Table element, DMType to Transforamtion element, and An Assocation to Link element, for example. Once an input merging model is designed, the correspondent output model is generated automatically by executing this transformation in the ATL runtime toolkit.

```
module dmm2cwm;
create OUT : Cwm from IN : Dmm;

-- create target table from DMElment
rule DMElement2Table {
    from
        src : Dmm!DMElement (not s.isSourceType())
    to
        tab : Cwm!Table (
            name <- src.name
            attribute   <- col
            checkConstraint <-con
        ),
        col : Cwm!Column foreach (e in src.attributes)
            name <- src.attributes->getName (e)
            checkConstraint <- src.attributes
                ->collect(e | e.constraints)
        )
        con : Cwm!CheckConstraint
```

Figure 7: DMM to CWM Transformation.

4 A CASESTUDY

We have applied the model driven data integration approach to a data warehouse development project in Thames Valley University. Different data sources from current university systems (such as the library system, student administration, or e-learning) have been integrated into the data warehouse system to provide a unified data view for a personalised student academic intervention system, based on data mining. In the project we have collected 3 years institutional historical data to build a DW and to predict individual student performance and dropout rate. As well as the suitability of the course or module for student intervention. The details of the case study were introduced in (Kim et al, 2009).

Based on the case study, we designed our DMM meta-model, and applied the model and its model transformation to the case study experimentally. In this section, we demonstrate our data merging approach throughout the example of merging two school data entities in Figure 3. The model in Figure 3 was entered into DMMtoCWM transformation engine and then the output CWM model was generated by the engine. The converted CWM is shown in Figure 8. According to the ruleCreateElment_MG_Faculty, UE_School element and CM_School are mapped to MG_School.

DMJoin is mapped into Joiner operation as well. The platform we used is Oracle Warehouse Builder.

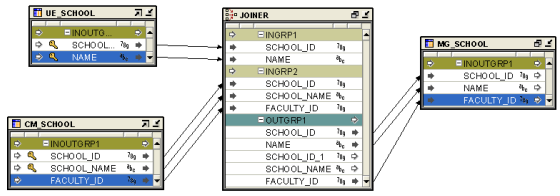

Figure 8: An Example of Data Merging PSM.

Once PSM is imported into ETL tool, each model has to be bound with actual data table manually. Through this process the actual data type is determined and additional conditions and logics can be added. Then executable codes are derived from the PSM. The following script in Figure 9 shows a part of PL/SQL packages which is generated from Oracle Warehouse Builder.

```
.... INSERT
       /*+ APPEND PARALLEL("MG_SCHOOL") */
     INTO
       "MG_SCHOOL"
        ("SCHOOL_ID",  "NAME",  "FACULTY_ID")
        (SELECT
       "UE_SCHOOL"."SCHOOL_ID" "SCHOOL_ID",
       "UE_SCHOOL"."NAME" "NAME",
       "CM_SCHOOL"."FACULTY_ID" "FACULTY_ID"
     FROM
       "UE_SCHOOL"  "UE_SCHOOL",
       "CM_SCHOOL"  "CM_SCHOOL"
     WHERE
       ( "UE_SCHOOL"."NAME" =
         "CM_SCHOOL"."SCHOOL_NAME" )......
```

Figure 9: An Example Merging Code

5 RELATED WORKS

Several researches have been proposed to overcome the challenges in designing of data integration in the context of MDE. In this section, we present a brief discussion about some relevant approaches.

In (March et al, 2007), MD2A (Multi Dimensional Model Driven Architecture) is suggested as an approach for applying the MDA framework to one of the stages of the DW development: multidimensional (MD) modeling. The authors defined MD PIM, MD PSM and necessary transformations. Although the suggested framework and models covers formalized MDDI, the designed models do not properly address data merging.

For conceptual modeling of data mapping, (Vassiliadis et al, 2002) suggests an ETL mapping model with their own graphic notation. on the other hand, (Mora1, Vassiliadis, and Trujillo, 2004) extends UML to model inter-attribute mapping at the attribute level. A conceptual model can be identified with a PIM in the context of MDA since it describes the necessary aspects of the application independently of the platform on which it will be implemented and executed (Kleppe et al, 2003). Although both of works presents the mapping between data source and target in different levels of granularity, they do not cover linking to PSM which is usually transformed from PIM.

(Muñoz et al, 2009) proposes the model-driven generation and optimization of integration tasks using a process-based approach. The approach models data integration process in high abstraction level in order to raise portability and lower maintenance effort. Although it provides modeling whole integration process rapidly, it does not consider details of each integration process modeling such as data mapping.

Furthermore, several automated data merging approaches are also researched in order to reduce human intervention for data merging through extraction of combined meta-data from source data or source meta-data in (Konigs, 2005) and (Embley et al, 2004). Particularly, (Fabro et al, 2008) and (Marcos et al, 2006) describes semi-automated model transformation using matching transformations and weaving models which can be applied on generation of merging model as well.

6 CONCLUSIONS

In this paper, we have presented a data merging system that aims to provide consistency between design, implementation, and automatic codes generation through creating abstract models in the early stage of a project. Physical models and executable codes from the abstracts will be generated. Through model transformation into CWM, the proposed conceptual modeling does not become isolated from the commercial systems, instead it shows a possibility to be extended and integrated with the existing industrial standards. The proposed meta-model and merging system was evaluated through a case study in Thames Valley University. A graphic modeling tool is being developed, with the aim to improve user interface through the conversion of rule scripts into graphic notations.

With this approach, data warehouse engineers can easily focus on data merging design being separated from concerns of physical environments, then integrate the design into ETL tool considering physical infrastructure at this stage. Executable program codes then can be derived from ETL tool finally. In this way, ETL design can be supported and well maintained systematically in model driven framework promising the success of DW development project.

ACKNOWLEDGEMENTS

REFERENCES

Allilaire, F., Bzivin, J., Jouault, F., and Kurtev, I., 2006. *ATL: Eclipse Support for Model Transformation*. In Proceeding of the Eclipse Technology eXchange Workshop (eTX) at ECOOP.

Bezivin, J., 2005. *Model-based Technology Integration with the Technical Space Concept*, In Metainformatics symposium 2005.

Bohm, M., Habich, D., Lehner, W., and Wloka, U., 2008. Model driven development of complex and data intensive integration processes, MBSDI 2008, CCIS 8, pp.31-42

CWM, 2008. *Common Warehouse Metamodel*, Object Management Group. http://www.omg.org/technology/documents/modeling_spec_catalog.htm

Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks, 2008. *Eclipse Modeling Framework*. Addison-Wesley Professional

Embley, D.W., Xu, L., and Ding, Y., 2004. *Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned*, SIGMOD Record, Vol. 33, No. 4

Fabro, D.D.M. and Valduriez, P., 2008. *Towards the efficient development of model transformations using model weaving and matching transformations*, Conference of Software and Systems Modeling.

Frédéric Jouault and Ivan Kurtev, 2006. *On the Architectural Alignment of ATL and QVT*

Greenfield, J., 2004. *Software factories: Assembling applications with patterns, models, frameworks and tools*. In GPCE, page 488.

Kim, H., Zhang, Y., Oussena, S., and Clark, T., 2009. *A Case Study on Model Driven Data Integration for Data Centric Software Development*, In Proceedings of ACM First International Workshop on Data-intensive Software Management and Mining.

Kimball, R. and Ross, M., 2002. *The Data Warehouse Toolkit*, John Wiley & Sons. 2nd edition.

Kleppe, A., Warmer, J. and Bast,W., 2003. *MDA Explained. The Model Driven Architecture: Practice and Promise*. Addison-Wesley, Reading.

Konigs, A. 2005. *Model Transformation with Triple Graph Grammars*. Model Transformations in Practice Satellite Workshop of MODELS 2005. Montego Bay, Jamaica.

Marcos, D.D.F., Jean B. and Patrick V., 2006. *Weaving Models with the Eclipse AMW plugin*, Eclipse Modeling Symposium.

MOF, 2008. Meta Object Facility, Object Management Group. http://www.omg.org/mof.

Mora1, L.S., Vassiliadis, P., and Trujillo, J., 2004. *Data Mapping Diagrams for Data Warehouse Design with UML*, volume 3288 of Lecture Notes in Computer Science, pp 191-204.

Muñoz, L., Mazón, J., and Trujillo, J., 2009. *Automatic generation of ETL processes from conceptual models*. In Proceeding of the ACM Twelfth international Workshop on Data Warehousing and OLAP.

OCL, 2008. *Object Constraint Language.* Object Management Group. http://www.omg.org/technology/documents/formal/ocl.htm.

Rahm, E., and Do, H. H., 2000. *Data Cleaning: Problems and Current Approaches*, Journal of IEEE Data Engineering Bulletin, volume 23.

March, S. and Hevner, A., 2007. *Integrated decision support systems: A data warehousing perspective*. Decision Support Systems, 43(3):1031-1043.

Vassiliadis, P., Simitsis, A,. and Skiadopoulos, S., 2002. *Conceptual Modeling for ETL Process*, ACM Fifth International Workshop on Data Warehousing and OLAP 2002.