Semi Automatic Method for Designing ETL Processes

Ahmed KABIRI, Dalila CHIADMI
Computer Science Department, EMI School
Mohammed Vth University-Agdal
Rabat, Morocco
{akabiri, chiadmi@emi.ac.ma}

ABSTRACT: *Data warehouse (DW) has been widely recognized as an effective solution for integrating diverse sources. In this* environment, Extraction-Transformation-Loading (ETL) processes constitute the integration layer which aims to pull data *from data sources to targets, via a set of transformations. ETL is responsible for the extraction of data, their cleaning, conforming and loading into the target. At the beginning of DW project, one of the most important tasks is to construct conceptual design of ETL processes. Several solutions have been proposed to support ETL designer in this task. Nevertheless few methods have accompanied these proposals. In this paper, we present a method for modeling and organizing ETL processes based on our experience in real world. More precisely, we suggest modularizing an ETL process into six modules. Namely, Technical Check, Semantic Check, Standardization & Cleaning, Mapping, Particular Processing and Rejects Management Bus. The proposed approach takes four inputs and produces, exactly via ENGINE device, a conceptual model of ETL processes using a graphical notation of our framework KANTARA. Finally, we give guidelines how to construct our method inputs.*

Keywords: ETL, ETL Method, ETL Modelling, ETL Organization, DataWarehouse, BI

Received: 11 May 2014, Revised 20 July 2014, Accepted 29 July 2014

1. Introduction and Background

In order to develop a successful strategy, managers need constantly to know the situation of their business. For this reason, they have to analyze their own data (data collected from operational applications). The challenge is to make right decision and to minimize the risk of making bad decisions. To meet this need, enterprises as organizations build decision support systems (DSS).

From architectural point of view, a DSS consists of a stack of cooperating components as one can see in Figure 1:

- The DW is the central element that perpetuates the information produced by the DSS.
- The "*Reporting and analysis*" layer is the presentation layer. It accesses the DW via dedicated tools and displays the data in

several formats, often as dashboards or charts. This layer is the visible part to end users.

• ETL has the critical mission to feed the DW with data from disparate sources. The ETL layer is invisible to end-users since it is responsible for the operations taking place in the back stage of DW.

• Sources layer is the provider of DSS. Typical sources include databases, flat files and xml files.

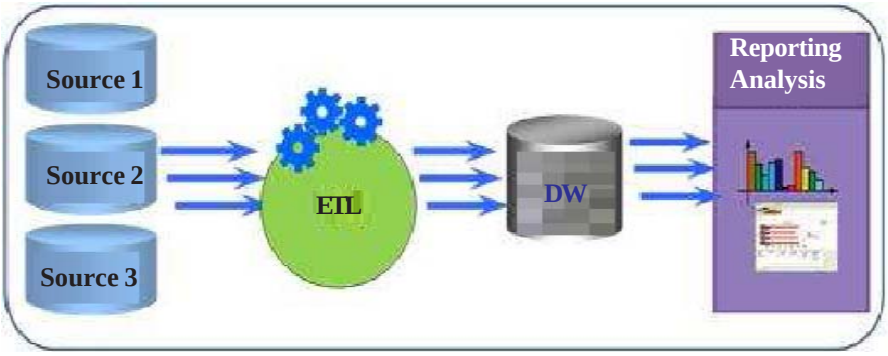

Figure 1. Classical DW Environement

It is widely recognized that building ETL processes is expensive in terms of time and money. It consumes up to 70% of resources [1], [3], [5], [6],[7],[8],[9]. A set of studies have shown this fact [5]. At the conceptual level, the ETL designer defines how to map sources attributes to the targets ones. Depending on the quality of data sources, he has to specify how to clean and how to conform data. For example, he states how to remove duplicates and how to standardize attributes values. To assist the designer in his task, the research community has suggested several proposals for modelling ETL processes [1], [11], [13], [14], [15], [16], [17].

In spite of the abundance of conceptual models proposals, only few methods have been suggested (as we will see in related works section). Here, we intend to enrich the field of conceptual modelling of ETL processes by proposing a helpful method for defining and organizing ETL processes. Namely, we suggest an approach which is framework-based aiming at:

• Enforcing reuse.

• Enhancing the quality.

• Organizing ETL processes

• Providing support and guidelines for designing ETL processes.

In a previous work [1], we have proposed a framework for conceptual modelling of ETL processes. Additionally in [2] we have shown our approach for maintaining these models. In current work, we complement previous ones by defining a method to build these models. We propose an organization (namely a modularization) of an ETL model (conceptual model). Then we define the requirements of our method (its inputs) in order to get ETL model (its output).

Our goal is to get a conceptual model of ETL processes using our notation presented in [1]. Still, our work is orthogonal to other modelling proposals. It can be applied to theme.

Outline: The remaining of this paper is organized as follows. Section 2 gives general information around ETL and precise some basic terminology while section 3 introduces and describes our ETL organization. Section 4 and 5 are dedicated to our method while section 6 presents related works. We conclude and present our future works in section 7.

In the rest of this paper, we don't make difference between ETL processes, ETL job, ETL. We do the same for ETL modelling, ETL design, conceptual modelling of ETL processes.

2. Backgroound

2.1 Architectural Design and Modelling Design

Previous section reports about the importance of ETL layer in DW environment (70% of whole budget time...). To build this important layer when a DW project (system) is launched, a decisive decision should be making about the architecture of this layer. Two choices are possible:

(a) Either to buy an ETL tool or

(b) To build ETL processes from scratch.

There is no consensus about this decision. A survey presented in [10] shows this disparity. Each policy has it pros and cons. While option (a) saves time, option (b) saves money. The intermediate option and solution will be open source tools like [18, 19].

Designing ETL processes by the intermediate of an ETL tool or programming language is a technical task. It is the matter of being familiar with processing or scripting paradigms. Furthermore, the technology is enough strong to handles any data processing (in the scope of structured data).

As a result, the challenge is more in modelling and specifying ETL processes. As Trujillo states, "*we argue that a model and*" method[1] "*are needed to help the design and maintenance of these ETL processes from the early stages of a DW project*" [17].

2.2 Overview of ETL Processes

An ETL process (also known as ETL job) integrates heterogeneous sources to feed a target (often a data warehouse). To this end, an ETL extracts data from multiple sources (flat files, databases, etc.) which may be local or distant. The extracted data pass through a sequence of transformations. At the end of the way, processed data are loaded into the target. In next paragraphs, we illustrate these ETL steps.

Extraction step has the problem of acquiring data from a set of sources. Often, sources come from operational applications with the possibility to use external sources for enrichment. The goal of an Extraction step is to access available sources, pull out the relevant data, and reformat such data in a specified format. Extraction step is performed over sources. Therefore, this step has the challenge to face sources instability besides their diversity.

Loading conversely to previous step, loading step has the problem of storing data to a set of targets. In DW context, targets are fact tables and dimension tables organized as a star scheme according to dimensional modelling. The goal of a loading step is to access available targets, writes the outcome data (transformed and integrated data) into the targets. This step can be a very time-consuming task due to indexing and partitioning techniques used in data warehouses [6].

Transformation is a broad term meaning all the data processing operations performed from sources to target (mapping attributes).. Such transformations include join, filter, sort, etc. operations. This is the critical phase in the whole process since it carries out the logic of business process instanced as business rules. Besides, from the first treatment to the last one, the structure of processed data is changing. The workflow schema is modified step by step, either by adding or removing attributes.

The Transformation step is the critical phase in the whole process because it carries out the logic of business process. In next section we review the process of building ETL , deals with syntax conflicts, semantic conflicts, etc

2.3 ETL Design Process

ETL as integration layer in DW configuration has to face several challenges. In [17], Trujillo et al propose a global process of ETL design. The authors propose a process composed on six actions:

1) Select sources
2) Transform the sources
3) Join the sources
4) Select targets
5) Map source to target attributes
6) Load data.

[1]The original word in citation is methodology

In the process above, the step 2 (transform the sources) is so broad that it can be interpreted in different ways. Particularly; for us; the step 3 (join sources) is a kind of transformation (step 2). Transformation step as fundamental stage where ETL adds value is refined by Kimball [6] as cleaning and conforming steps. Without these two specifications, the end users will suffer from poor quality of data delivered by ETL, and the ETL team will suffer from "*code, load, and explode*" phenomenon [10]. That is while ETL team implement ETL processes, they take further time to rework their jobs and to fix "*a large number of errors due to unanticipated values in the source data files*" [10]. To avoid the above problems, efforts are needed at early stages of ETL design. Namely conforming and cleaning issues should be solved and specified at conceptual level.

2.4 Definitions :Method and Methodology

Method and methodology are used interchangeably, but they are different and have different meanings establish a good starting we start by clarifying Thus and according to [22]:

- "*A method is an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products*".
- "*A methodology of information systems development is the systematic description, explanation and evaluation of all aspects of methodical information systems development*".

Stated differently, a methodology represents a theoretical analysis of methods while a method describe of making "*things*".

A method deals with two aspects: product Indeed, product is the result of applying a method while the process deals with "*the way*" to follow to get products [4]. Thus a method includes product and process quality of "*fabrication*" process influences the quality of the products.

Our goal in this work is to help designer in organizing and defining ETL specifications via a method. In section IV will find out what the product and the process are.

3. ETL Organization

In this section, we describe an organization of ETL processes. Such organization is based on ETL processes modularization that we highlight in sections A sketch some issues related to this organization.

3.1 Overview of ETL Organization

ETL modularisation consists on defining functional modules that should be distinguished in each ETL (organizational modules). In our approach highlighted in [24] we organize an ETL process as 6 parts or six sub ETL processes. The figure 2 depicts this modularization. More precisely, the gray box highlights these modules which can be regarded as a workflow. As ETL deals with sources and targets, the above ETL modules interact with theme as start and end points respectively.

Except Reject Store that depicts a repository to save reject data, all other boxes represent modules. Furthermore, all modules are mandatory except the last one (Processing that is why its lines are dashed). The black and the red links symbolize data ransition. Respectively data with status ok and ko. The former status holds when data performs all treatments included in a given module without errors. Otherwise the second status takes place.

Why this order in organizing modules? modules order of the above organization from what happens at logical level. Indeed, if we translate to logical level, there is a well known rule that states to push selections or filters in early stages, in order to avoid processing unnecessary or b Thus execution time is enhanced. This rule explains why we organize modules in this order (we agree that conceptual models leads logical one).

3.2 Impact on Logical Level

For precision, the organization above is at conceptual level while in logical or in physical level there are two options to perform data translation from a module to another options are:

- **Pipelining:** passing the output of individual module to next one without intermediate storing.
- **Intermediate storing:** conversely to previous technique, the outcome of a given module is saved in intermediate store. Thus

the next module gets data from intermediate store and not directly from its neighbour module. the next module gets data from intermediate store and not directly from its neighbour module.

Which one of the above techniques to use at logical level?
It is widely recognized that performance is one of the most important issues in ETL due to specific aspects of ETL (high volume of data or near real time requirements, etc…). In this perspective, the first option is useful.

In other side, when the ETL fails, especially when the volume of treated data is important, it is interesting to have intermediate stores (save points). Thus when ETL fails, it is possible to restarts data processing (ETL job) from last save point. However, one will lose in performance and overheads running time, when there is no crash. Therefore we need guidelines to choose which technique to use. Obviously, the second option is suitable if ETL has enough time otherwise the first option is not escapable.

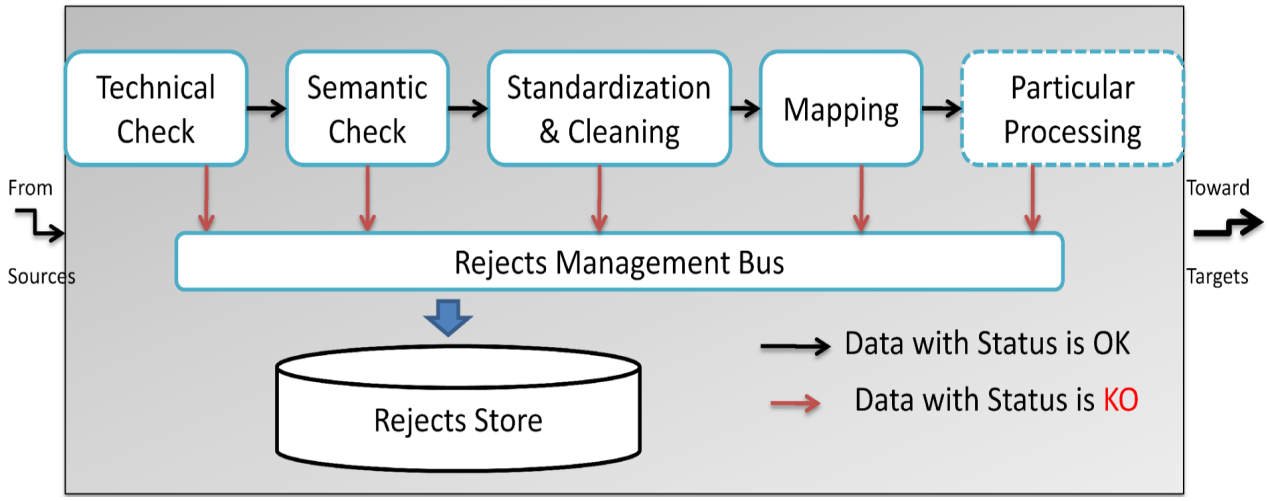

Figure 2. Modularization of ETL process

4. Organizational Modules of ETL

In this section, we explain each module and we argue its role. We consolidate this presentation with examples.

4.1 Running Example
We illustrate our proposal with an example of sale activity. We consider we have a source depicted in figure 3. We build different sub-parts of ETL process according to the organization presented above.

Goal: in order to show the importance of each module, we simulate the need to analyze the repartition of sales over cities in Morocco country. What would look like the final dashboard?

4.2 Technical Check
This module deals with data validation at technical level. Stated differently, it checks if what we get from sources is what we expect. Thus, this module regroups all technical checks that should be done over data.

The rules to be checked at this level are constructed according to data types. Controls are performed attribute by attribute. Also depending on the nature of sources involved, controls over records (all attributes) may be necessary.

Building ETL processes without these checks will leads to ETL crashes. Clearly such situation is not welcome especially that these checks can be constructed automatically

4.3 Mapping
This module deals with data translation from sources to targets. In other words, it includes data conversion and calculation of target information. More generally, this module regroups rules in connection with target information and target format.

Once data pass through this module, the dataflow format is those of the target. Finally, let recall that mapping step is a systematic phase in data integration process as in ETL design process.

Figure 4(Mapping) shows the dashboard resulting when processing dataset of figure 2 with an ETL processes including Technical checks and Mapping modules. As one can see, resulting diagram doesn't' help to make decision according to the goal defined above.

4.4 Semantic Check

This module deals with data validation at semantic level. It checks if received data are semantically corrects. Consequently, this module regroups all functional checks that should be done over data.

This includes both the verification of attributes content and verification of attributes relationship. More generally, the rules to be checked at this level are constructed according to business rules.

Unlike first module, theses rules cannot be constructed automatically. Figure 4(Conform_Without_Cleaning) shows resulted dashboard when processing dataset of figure 2 with an ETL processes including Technical checks, Mapping and Semantic Checks modules but it remains not clean.

4.5 Standardization and Cleaning

This module deals with data standardization. In other words, it aims to ensure that sources "*talk the same language*" as targets. This includes resolution of syntax and semantic conflicts between involved sources. More generally, this module regroups cleaning rules to perform over data.

The rules to be stated at this level are constructed according to sources quality. This includes schemas and data problems.

Figure 4(Clean_Without_Conforming), shows resulted dashboard that is not conform when processing dataset of figure 2 with

IdSale	Product	Price	Quatity	City
1	PEN	1,3	2	rabat
2	PC	250	15	rabat
3	TV	200	8	casa
5	CHAIR	15	5	casablance
6	CHAIR	15	8	casa
7	PEN	1,3	20	casablance
8	CD	1	100	rabat
9	CD	1,1	130	casaBlance
10	PEN	1,3	2	meknes
11	PC	250	15	meknes
12	TV	200	8	fes
13	CHAIR	15	5	tanger
14	CHAIR	15	8	casa
15	PEN	1,3	20	meknes
16	PEN	2	300	paris
17	DESK	100	15,5	sale

Figure 3. Source DataSet of Sales

an ETL processes including Technical checks, Mapping and Standardization & Cleaning modules.

4.6 Particular Processing

As we said in the beginning of this section, Particular Processing module is optional. It aims to handle treatments which are not possible to express in other modules.

This module is useful too, for dispatching the process (after a common treatment) into branches where each branch feed a specific target. This module regroups specifics rules by target involved in the ETL process.

Specific issues related to targets, like archiving are handled at this level.

4.7 Reject Management Bus

This module manages rejects that is data that fails to pass from a module to next one. This module can be regarded as a service that should coordinate with other modules.

To illustrate the importance of this module, suppose an ETL process without reject management feature. Simply rejected data will disappear. No trace will be created. According to the example above, figure 4(Clean_and_Conform), shows the final diagram. But one can note that record at line 17 (in data source of figure 3) has disappeared. Thus implementing ETL processes without rejects management will lead to loss of data. Stated differently with business terminology, rejected data may represent the lost of money or the lost of information.

Still, why there will be rejects? At business level, there will be direct statements to eliminate some data. Such data are not useful from business point of view; thus losing theses data is not "*dangerous*".

At logical level too, there are several reasons to reject data. Indeed, not all data will arrive at the destination, since it may be incorrect. Indeed, sources may contain erroneous data, which should be treated carefully by fixing errors and defects that they contain. Otherwise, this kind of data has to be filtered and rerouted to reject store. Losing theses data is "*dangerous*".

In next section, we give further information about this store dedicated to receive rejected data. We will present what information to store and in which format.

4.8 Reject Store

All rejected data; either for functional or technical reasons; should be traced and stored. In fact, rejected data will be recycled. That is a manual or even, if possible, an automatic or semi automatic processing of these data (coming from reject store) is required.

Table 1 shows a part of metadata we manage for rejected data. The first line gives the names of these metadata. The econd line states there types while the last line explains there functional meaning. Thus and in connection with previous modules (presented in last section), each module must format rejected data and supply such information.

5. Inputs and Output of Our Method

In this section, we describe a method for constructing ETL models. Such method is based on four inputs rules illustrate in sections B, C, D and E. While in describe its output. The mechanism of converting inputs into output is presented in conclude this section by implementation issues.

5.1 Method Overview

Defining a method consists on defining product and process. In this work, product is ETL model (conceptual model) and process is ENGINE device present in next sections. Figure 5 depicts this method.

In order to conceive an ETL process and according to the organization presented above, in our approach we assume to have some inputs. Figure 5 highlights these inputs with four main building blocks. The four blocks on the l inputs while on the right we find the output (product). Between both sides, there is ENGINE device that transforms the inputs into output.

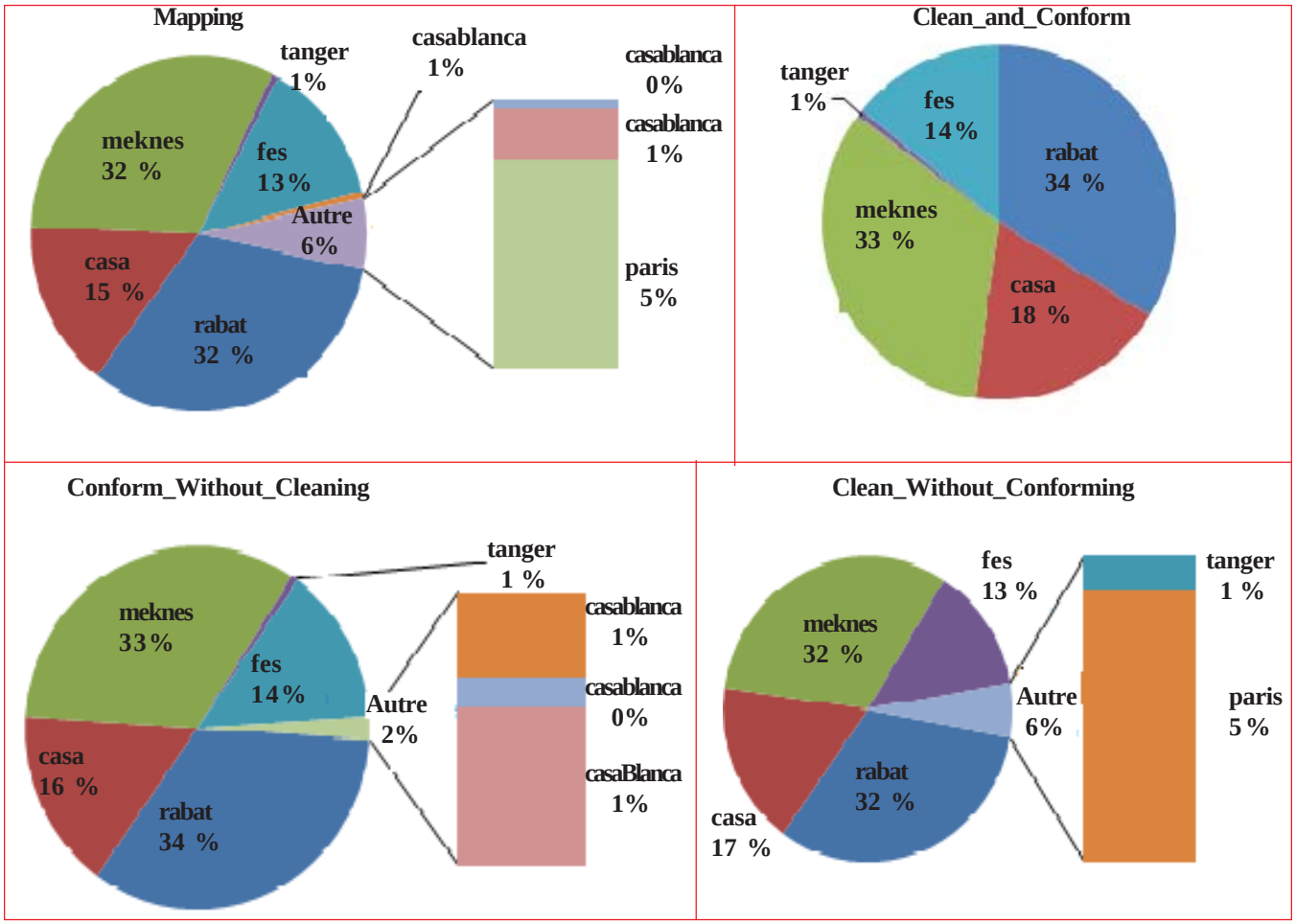

Figure 4. Resulting Dashbords

In connection to modules organization defined previously; table 2 summarizes the correspondence between and the input blocks of ENGINE device.

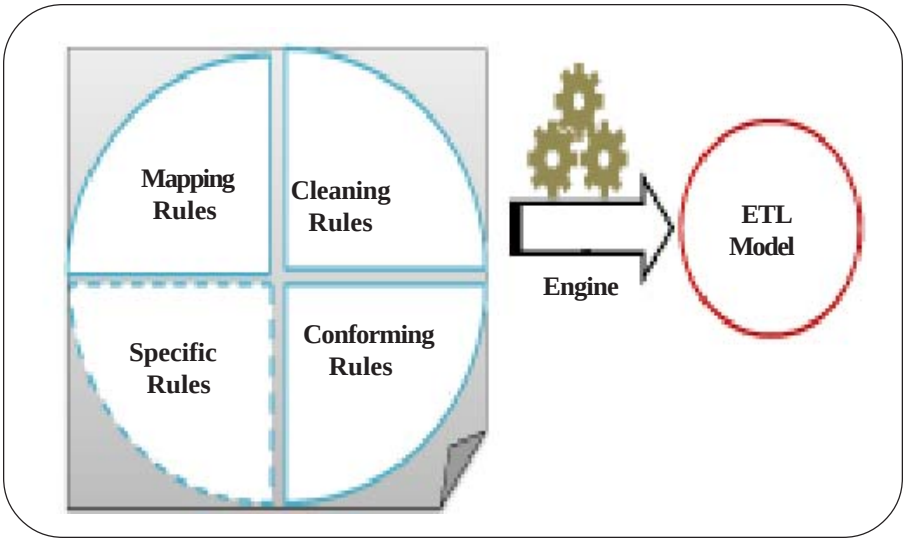

Figure 5. Overview of our method

Metadata Name	Type	Comment
ETL Process Name	String	This information is helpful to discriminate between data (rejected records) that belongs to each ETL process. This supposes there is a global repository for receiving all these data.
Module name	String	It traces the module where the event takes place. According to modularization above, this info takes values (“*Technical checks*”, “*Semantic Checks*”, “*Standardization & Cleaning*”, “*Mapping*”, “*Particular-Processing*”). It is a hierarchy over rejected data that is useful to decide on reprocessing these data
Component name	String	Traces; with more granularity; where the reject event takes place. The component can be interpreted, as a function or an operator. This metadata helps to recognize where to get more details if necessary.
Motif	String	Explicit message describing the reason for rejecting data. This metadata answers the question why there reject event. It guides to correct data.
Record in rejected format	String	Data being rejected. This is useful to extract intermediate information calculated during data travel before the reject event. Such intermediate information will help to decide what to do with rejected data.
Record Identifier	Number	Technical information to identify initial record from sources without any transformation. This more helpful if rejected data will be resend to sources for investigation
Flow Identifier	Number	Since there will be several passes of ETL processes, it is desirable to distinguish between multiple passes. Each pass corresponds to new periodic data from sources

Table 1. Meta Data of Reject Store

In the table 2, when the cellule is marked with (x) symbol, it means that ETL subpart (the line in table2) contains rules specified by block rules (column of table 2). Let note that Technical check module is systematic. Particular processing module that is optional depends on the presence of Specific rules block. The difference with other modules is that their sources blocks are quasi systematic which is not the case for Specific rules block.

Module \\ Block	Mapping rules	Cleaning rules	Conforming rules	Specific rules
Technical Check				
Semantic Check			x	
Standardization and Cleaning		x		
Mapping	x			
Particular processing (optional)	x	x	x	x

Table 2. Modules and Input rules correspondance

In spite of it not visible in the table above, and according to the modularization of ETL process adopted in section A, the Reject Managements Module is indirectly connected to all rules blocks. Each block will be further discussed in the paragraphs that follow.

5.2 Mapping Rules

Mapping rules are a set of transformations defining the mapping of attributes (information) available in a specific sources format to the attributes (information) used in the target definition. The main purpose of mapping rules is matching source data into target view.

The scope of transformations used for this purpose is broad. Typical mapping rules used include:

- Simple attribute mapping, where an attribute of the source is assigned to an equivalent one in the target.
- Constant value assignment, where a constant value assigned to an attribute of the target.
- Mapping via formulas, where predefined functions are used to feed an attribute of the target.
- Complex mapping, to state specifications for complex transformations.

Figure 6 illustrate these examples of rules where the column on the right represents output attribute while the column represents mapping expression. As one can note on the left column, there are some keywords denoting keywords of mapping language like LOOKUP, CONCATE.

5.3 Cleaning Rules

Cleaning rules are a set of transformations defining the enforcement of sources attributes. In other words, their main purpose is converting the source-system-specific representation of data into target representation and standardizing data provided by sources. As an ETL process can include multiple sources, Cleaning rules include solving conflicts between these sources. Problems will occur at two levels: schema and data level.

Data cleaning in the ETL scope is a complex issue which depend on many parameters. Data quality of sources and type of sources are examples of theses parameters. Thus the Scope of cleaning rules is so broad. But typical ones include:

- Technical conversions between source attributes and the target ones taking in account the differences in term of type size and the format.
- Statements about missing data. Here we identify attributes that has to be corrected and in which case. Often a constant value is assigned to missing data.
- Deduplication where selection criteria are defined to discriminate between similar (duplicate) records. see figure 3).

5.4 Conforming Rules

Conforming rules are a set of transformations performing data validity and integrity checks. This includes the enforcement of attributes relationship that is to check the logic relationship between attributes. Another main purpose of conforming rules is the trace of business rules (transformations from a business point of view) contrarily to cleaning rules which deal with technical aspects.

The content of Conforming Rules depends on business rules governing the domain of the datawarehouse system being built. However, at high level basic conforming rules include

- Lookup of referential data (master data) attributes. That is the assignment of master data attributes (source of enrichment) to target attributes or even the to the added (new) attributes.
- Discrimination of valid data and invalid data. Here, Conforming rules states the filtering rules of data based on expressions to match or criteria to satisfy…etc.
- Business rules including business constraints that state the relationship that should hold between attributes.Business rules can be simples as complexes on the nature and the scope of the being created rule. Obviously the business domain complexity has its impact.

Mapping	Output field
S1. Custld	Customer Key
UPCASE (S1. Fname)	FirstName
LOWERCASE (S1.Lname)	LastName
S1.City	City
LOOKUP (S2.Mail) by S1. Fname=S2.Nom and S1. Lname= S2.prenom	Email
CONCAT (S1.Country Code,S1.phone)	PhoneNumber

Figure 6. Example of Mapping Rules

5.5 Specifc Rules

Specific Rules are a set of transformations defining the specific treatments that are not possible to express in other modules. Specific aspects of rules can be either technical or functional. Thus, we distinguish the two following categories of these rules.

• **Functional Specific Rules:** this category depends on functional aspects rocess and its fashion of design. connexion to sales arena feeding Customers dimension. When specifying this ETL process it is possible to dictate to "*identify new customers and send them to X application*." X application may be for example an application that represents and applies enterprise process for customers.

• **Technical Specific Rules:** This category regroups technical aspects which are specific to targets involved in a certain ETL process. Basically, these rules deal with how an ETL process will deliver data to its customers (its dependents processes) as cooperative processes define an SLA (service Line Agreement) that can be regarded as a contract between them, additional checks and data formatting will be required before loading data (delivering data). Another significant aspect of specific rules is related to the way of managing the target changing dimensions. Indeed, depending on the need to track history of changes (for dimensions) a strategy to handle changes is required. Techniques like type 1, type 2 are used for this purpose [25].

Specific Rules is the last input block of our method. Next section deals with its output.

5.6 ETL Model

ETL model is the resulting output that is a combination of components defining the conceptual model of a given ETL process. Components are complex classes managing a set of metadata. They are part of the meta model of KANTARA that is our framework for managing ETL processes. More details are available in [1].

For model representation, this model is based on graphical notation presented and detailed too in [1]. the figure 4 depicts this notation which is based on following are facts:

- Extract component: symbolizes the extraction step and manage a set of metadata to handle the issues of this step.
- Similarly, Transform and Load components symbolize the transformation and load steps respectively.
- Note a free zone text.
- Linker a simple line to link between components, it represents flow transition from one component to another;
- Parameters correspond special text area.

5.7 Engine

Engine is the convertor of inputs into output. It transforms all information coming from previous building blocks (input rules)

into ETL model using graphical notation of figure 7. resulting model is organized according to the organization showed in previous section.

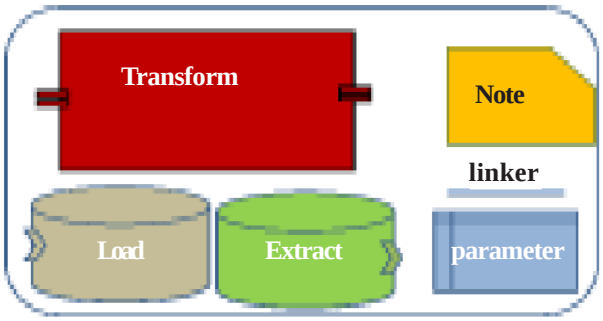

Figure 7. KANTARA Notation Used for ETL Design

Engine is based on an algorithm and has the following form: $E = (LG, MP, NT, CL, CR, SR)$, Where

- ***LG*:** A mapping language that is a set of functions or operators $\{f_i\}$
- ***MP* :** mapping rules that are a set of rules noted as $Y_i = f_i(X_1, X_2 \ldots\ldots Xn)$ where Y_i and X_i denote respectively output and input attributes
- ***NT* :** Notation, that is a set of components or operators denoted $\{C_i\}$
- ***CL* :** Cleaning rules,
- ***CR*:** Conforming rules,
- ***SR* :** Specific rules.

At high level this algorithm (Engine) is structured as below.

Step 1: Use mapping rules (insert sources, targets…)

Step 2: Enrich with conforming rules (insert filter…)

Step 3: Enrich with cleaning rules (insert transformation...)

Step 4: If Specific rules holds then use them for enrichment
Else go to step 5.

Step 5: Create and generalize connection to reject module (instantiate metadata of table 1).

All steps are expressed using LG (mapping features and they perform their tasks by inserting notation (NT) features. Finally, each step defines mapping between input and output attributes.

In our method, we have used our notation for modelling ETL process. However, for the same goal, it is possible to use another one (UML notation for example). To meet this need, the feature *Engine* should be adapted to this new notation. Still its whole process is intact.

Figure 8 gives details about step 1 of Other steps are constructed similarly. Below we details this part of ENGINE algorithm.

Step One Explanation:

1) Line 1 describes the inputs of step 1 algorithm. Indeed Y_i denotes an output attribute. Let recall that $Y_i = f_i(X_1, X_2, X_n)$ where X_i denote input attributes. *T* specifies target which will be populated by the given ETL process.

2) Line 2 describes the variables involved in this algorithm part. Thus, *list_sources* indicates list of involved sources *while list_transf* indicates list of involved transformation which will be inserted.

```
1.  input = list of mapping rules   Yi;  a target T
2.  Variables:  list_sources, list_transf.
3.  for each Yi      b b
        a)    Add to list_sources                    (Yi)
        b)    if Yi                  (Sj,Sj+1,...) add to list_transf  Join(Sj , Sj+1,...)
                                  (list_sources, list_transf)
                                  (list_transf)
6.  for each source Sj in list_sources , add "Extract(Sj)" component.
7.  for each source Sj in list_sources , add "Technical_Check(Sj)" component.
8.  for each source  Sj, if                  Sj) Then insert Split (Sj).
9.  for each element e in list_transf Add "Join(e)"  component.
10. If                  (list_transf) are included then add Merge (list_transf) component.
11. Add Mapping component
12. If                Add "Load (T)" component
```

Figure 8. Step 1 Details of Engine Algorithm

3) Line 3 parses each mapping rule feeding Y_i by:

a) Identifying and storing involved sources in the variable *list_sources.*

b) If Y_i involves many sources, then the flow should be converged to feed the target *T*. In other words, When a set of sources $\{S_j\}$ contribute to feed Yi then we merge between these sources by including " *Join*"operator**.**

4) The function *Remove_duplicates_in* (*list_sources*,*list_transf*) at line 4, works on variables by eliminating redundant information.

5) At line 5, the function *Keep_large_joins* (*list_transf*) aimsto optimize more by reducing the number of "*Join*"operators to insert. The idea is to keep *Joins* which involves maximum of sources.

6) In line 6, we attach each source S_j involved in ETL process to Extract-component which deals with extraction step according to our notation presented above.

7) For each source *Sj* that has been translated into extract component (line 6) we associate *Technical_Check(Sj)" component* which states technical checks to perform over source S_j. Technical-Check is a type *Transform component.*

8) Line 8 estimates for each involved source, if it is necessary to insert "*Split*" *component* that replicates the flow into many sub flows. Indeed, the function *non_serieable_Joins(Sj)* evaluates for each source taken as parameter *Sj* if it contributes to many "*Joins*". Whenthis condition holds it evaluates too if theses "*Joins*" are parallels or not. Thus we insert or not the "*Split*" component.

9) In line 9, we attach each "*Join*" involved in ETL process to *Join component* which is a type of Transform component.

10) Line 10 acts conversely to line 8. It estimates if it is necessary to insert "*Merge*" *component* that concatenatesmany sub flows into single flow. Indeed, the function *Independent-Joins* (*list_transf*) evaluates if it many "*Joins*" are included. When this condition holds it ENGINE device. evaluates too if theses "*Joins*" are parallels or not. Thus, in last case we insert *Merge* component.

11) Line 11 includes a *Mapping component* which is a type of Transform component. It handles mapping rules.

12) Line 12 estimates if it is time to insert "*Load*" *component* that deals with the target *T* loading. Indeed, the function *the-end()* evaluates if others steps(cleaning rules, conforming rules...) of *ENGINE* are activated or not. When others steps are not activated; "*Load*" component is inserted otherwise it will be inserted in other steps.

The algorithm above deals with step 1 of ENGINE device. It transforms mapping rules input into ETL model. In other words, it rewrites mapping rules into ETL model using a given notation. Others steps of ENGINE device if activated enrich the resulting model. They are constructed similarly to step 1. Obviously they perform different tasks.

5.8 Implementation and Illustration

Our method has two aspects:

- ETL organization (see figure 2)
- ENGINE device (see figure 5)

The empirical validation has been focused on these aspects. A prototype has been implemented using Java language based on Eclipse platform. We have used EMF (Eclipse Modelling Framework) [23], an Eclipse plugging for the purpose of creating the classes and there relations. Management of graph objects is performed by GEF (graphical Modelling Framework) [23], an Eclipse plugging providing functions and procedures to processes what EMF produces.

In reference to figure 6 which shows an example of mapping rules; figure 9 shows what ENGINE device produces when running over this input. By the way, others input rules are not activated. Simply in this example, ENGINE runs with mapping rules input only.

5.9 Consutructing Inputs

ENGINE device takes four inputs as we have seen in previous sections. In current section, we show how we deal with theses inputs. We give guidelines for constructing these inputs.

The idea is to analyze all mapping rules for cleaning and conforming involved attributes. Input attributes as output ones are compared according to four dimensions which are Type, Size, Format and Nullabilty. Table 3 summarises this analyze.

Let recall that a mapping rule is denoted as (1): $Yi = fi\ (X_1, X_2 \ldots\ldots X_n)$. Starting from this rule, we build automatically the following table. Thus the formula (1) becomes

(2) $Yi = f_i\ (\ r_{01}(X_1), r_{02}\ (X_2), \ldots\ldots\ldots\ r_n\ (X_n))$

Where r_i are derived functions solving conflicts between input and output attributes according to these four dimensions. Derived functions (r_i) are part of mapping language.

For example let consider the following mapping rule with its associate table.
(3): *FirstName* = *UPCASE* (*S1.Fname*)

By checking table 4 according to size column, one can see the divergence between input and output attributes. Thus the mapping rule expressed in (3) becomes:
(4): *FirstName* = *UPCAS E*(*SUBSTRING* (*S*1.*Fname*, 20))

Another Example About Cleaning and Conforming Rules

In reference to figure 3 which shows an example of dataset of a source in sales activity. We use this source and we consider it is the source labelled *S*1 in mapping rules presented in figure 6. For concision, we consider this example of mapping rule *City* = *S1.City*. By analyzing the content of city column in dataset of *S*1 (via a data profiling tool for example), one can define these rules

- Reject (S1) records where S1.*City* = "*paris*"
- Create a source S3 structured(Occurrences, City)
- *LOOKUP*(*S*3.*City*) by *S*1.*City* = *S*3.*Occurence*

Figure 10 shows what ENGINE device produces when running over mapping rule input and taken in account cleaning and conforming rules derived above.

6. Related Works

Data integration: Data integration is an active area of search. Several solutions have been suggested to same issues. In one hand, there are virtual data integration solutions represented by mediators like WASSIT [26] and TSIMMIS [27]. A mediation

	Type	Size	Format	Null ability
Code Operator on 4 digit	1 or 0	1 or 0	1 or 0	1 or 0
Yi	*type-Yi*	*size_y*	*format-y*	Y/N
X1	*type-*X_1	*size_x*	*format-x*	Y/N
X_n	…	…	….	…

Table 3. Input And Output

	Type	Size	Format	Nullability
CodeOperator	1	0	0	0
FirstName	String	**20**	—	N
S1.Fname	String	**25**	—	N

Table 4. Input And Output

system usually has a global schema, which provides the user with a uniform interface in order to access information stored in the data sources. This category of approaches has the advantage to access to data source on fly. However, rewriting step involved in mediation process is complex and requires non negligible time to dispatch the query before to gathers results and combine it again. Depending on the query complexity and data volume, mediator response for user query lasts.

In other hand, data integration with data warehouse approaches, supplies flexible solutions regarding response time. During data-warehousing process, ETL layer is responsible to gather data from disparate sources.

ETL Design and Modelling: A plethora of commercial ETL tools [20], [21] as well as a set of open sources [18], [19] exist. Both of them offer graphical interface to build ETL system. However, they do not supply deep support to conceive ETL processes. They focus on technical and running aspects more than designing ones. On the other side, research community enriches the field of conceptual modelling of ETL system with several approaches [1],[11],[14],[15],[16],[17].

These proposals differ on formalism and technologies used. For example, [11] present a model and a notation associated with constructs for modelling ETL activities that they translate to logical model through [13]. There are also approaches based on semantic web technologies like ontology [14], [15]. Many propositions, UML-based, enrich the literature [16], [17]. These approaches, above mentioned, are precious and interesting. But they are not accompanied with methods. Therefore, our current work is orthogonal to theme and enrich theme.

ETL Methods: As we said previously there is confusion between method and methodology. Thus and according to definitions listed above (see [22]), we deal with methods.

In spite of the abundance of conceptual models proposals (as shown in previous paragraph), only few methods have been suggested. Indeed, the authors of [12] show their method and report about Kimball method that is "*mainly grouped as a set of tips and technical guidelines*". The method of [12] authors aims to produce attributes mapping between sources and targets involved in a given ETL process. To meet this goal, the proposed method (that they call methodology) is spread over four steps: (1) identification of data stores; (2) candidates and active candidates for the involved data stores; (3) attribute mapping (4) annotating the diagram (conceptual model of ETL process) with runtime constraints.

Compared to other proposal, our method has the advantages modularization and enforcement of reuse. First we organize ETL model into six modules (see figure 2) particularly reject module which is often omitted. Secondly, our approach is semi automatic. In current work, we give guidelines to construct the inputs of our method. Exactly this is where we reuse the output of pre analysis step.

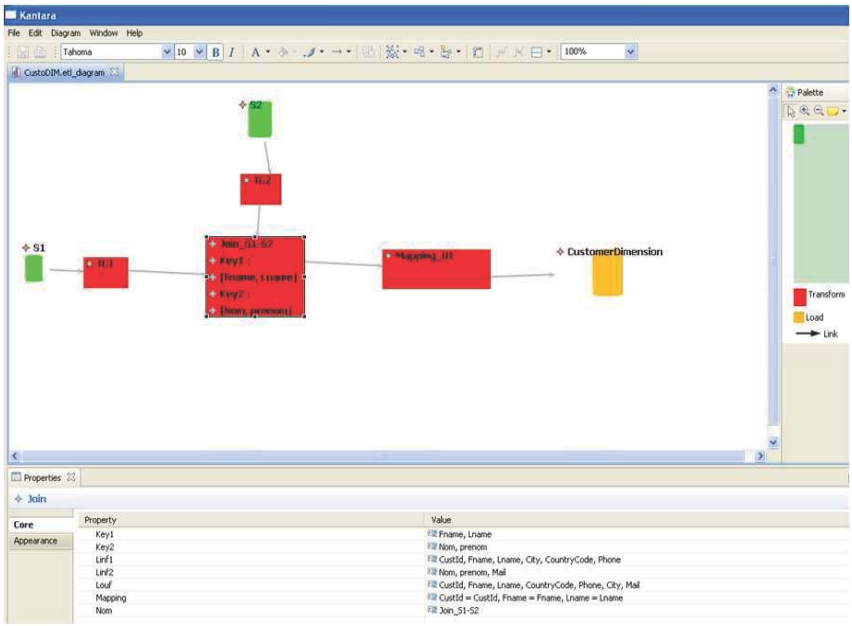

Figure 9. ENGINE output based on mapping rules only

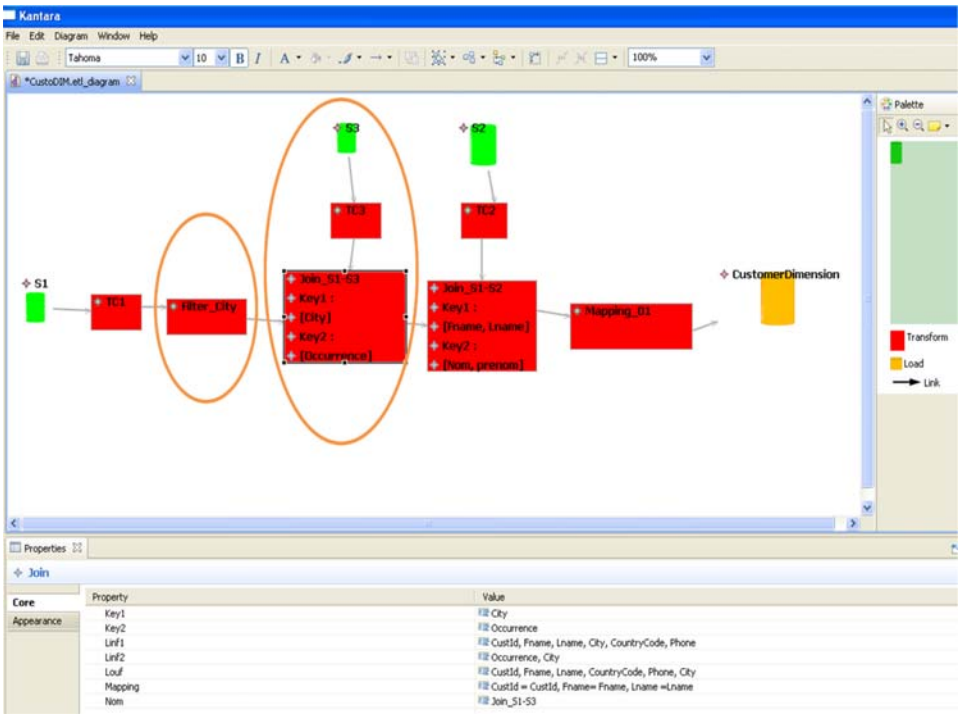

Figure 10. ENGINE output enriched with cleaning and conforming rules

7. Conclusion and Future Works

Data warehouse constitute an effective solution for integrating diverse sources. In this environment, ETL processes constitute the integration layer which aims to pull data from data sources to targets, via a set of transformations. ETL processes are widely recognized as complex and expensive. Therefore and in order to overcome this situation and simplify these processes, a model and a method become necessary. In this perspective, several works have dealt with conceptual modelling of ETL processes. However little effort has been dedicated to propose a method to organize and automate these processes.

In this paper, we have presented our method for modelling and organizing ETL processes. Firstly we have shown the functional modules that should be distinguished in each ETL. Exactly, we suggest modularizing an ETL process into six modules. Namely, Technical-Check Semantic-Check, Standardization-&-Cleaning, Mapping, Particular-Processing and Rejects-Management-Bus for which we have defined a set of metadata to manage.

The proposed approach takes four inputs (mapping rules, conforming rules, cleaning rules and specific rules) and produces, exactly via ENGINE device, a conceptual model of TL processes using a graphical notation of our framework KANTARA.

In order to construct the inputs of our method we have suggested guidelines. Particularly, we proposed to analyze all mapping rules for cleaning and conforming involved attributes.

The method we exposed in this work is based on our experience in real world. Currently we work on enriching our method by heading it by a profiling tool. We aim to simplify the input rules construction. In future works, we will focus on transition to logical model using both our model and method.

References

[1] Kabiri, A., Wadjinny, F., Chiadmi, D. Towards a Framework for Conceptual Modelling of ETL Processes . *In* : Proceedings of The first international conference on Innovative Computing Technology (INCT 2011). Communications in Computer and Information Science, 241, p 146-160.

[2] Kabiri, A., Wadjinny, F., Chiadmi, D. (2011).Towards a Matrix Based Approach for Analyzing the Impact of Change on ETL Processes. IJCSI International Journal, 8 (4,) July, http://www.ijcsi.org/papers/IJCSI-8-4-2-430-437.pdf. ISSN (Online): 1694-0784, ISSN (Print): 1694-0814.

[3] Vassiliadis, P.,Simitsis, A.(2009). EXTRACTION, TRANSFORMATION, AND LOADING., http://www.cs.uoi.gr/~pvassil/publications/2009_DB_encyclopedia/Extr act-Transform-Load.pdf

[4] Colette Rolland. «L'ingénierie des méthodes : une visite guidée». e-TI - la revue électronique des technologies d'information, Premier Numéro, 25 octobre 2005, http://revue-eti.net/document.php?id=726

[5] Simitisis, A., Vassiliadis, P., Skiadopoulos, S., Sellis, T. (2007). DataWarehouse Refreshment, Data Warehouses and OLAP: Concepts, Architectures and Solutions, IRM Press.

[6] Kimball, R., Caserta, J. (2004).The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning,Conforming, and Delivering Data, Wiley Publishing, Inc.

[7] Inmon, W., Strauss, D., G.Neushloss. (2007). DW 2.0 The Architecture for the next generation of data warehousing, Morgan Kaufman.

[8] Golfarelli, M. _(2009). Datawarehouse life-cycle and design, *In* :Encyclopedia of Database Systems, L. Liu, T. Özsuzsu (Eds.), Springer..

[9] Adzic, J., Fiore, V., Sisto. (2007). Extraction,Transformation,and Loading Processes, Data Warehouses and OLAP: Concepts, Architectures and Solutions, IRM Press.

[10] Eckerson, W., White, C. (2003). Evaluating ETL and Data Integration Platforms, TDWI REPORT SERIES, 101communications LLC.

[11] Vassiliadis, P., Simitsis, A., Skiadopoulos, S. (2002). Conceptual modelling for ETL processes. *In*: Proc. Of the 5th ACM Int Workshop on Data Warehousing andOLAP,DOLAP02.

[12] Simitsis, A., Vassiliadis, P. (2003). A methodology for the conceptual modelling of ETL processes. In J. Eder,R. Mittermeir, B. Pernici, editors, CAiSE Workshops, 75 of CEUR Workshop Proceedings. CEUR-WS.org.

[13] Simitsis, A. (2005). Mapping conceptual to logical models for ETL processes. *In* : Proc. of the 8th ACM Int.Workshop on Data Warehousing and OLAP,DOLAP'05.

[14] Skoutas, D., Simitsis, A. (2006). Designing ETL processes using semantic web technologies. DOLAP06.

[15] Skoutas, D., Simitsis, A. (2007). Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *Int. Journal on Semantic Web and Information Systems*.

[16] Muñoz, L., Mazón, J., Trujillo, J.(2009). Automatic Generation of ETL processes from Conceptual Models. *In* : Proc. of the th

ACM Int. Workshop on Data Warehousing and OLAP, DOLAP'09.

[17] Trujillo, J., Luj¶an-Mora, S. (2003). A UML Based Approach for Modelling ETL Processes in Data Warehouses. *In*: Song, I.-Y., Liddle, S. W., Ling, T. W., Scheuermann, P., editors, ER, 2813 of Lecture Notes in Computer science, Springer.

[18] Talend Open Studio, www.talend.com

[19] Vanilla Open source, www.bpm-conseil.com

[20] Informatica, http://www.informatica.com/FR/Pages/index.aspx

[21] IBM InfoSphere DataStage, http://www- 01.ibm.com/software/data/infosphere/datastage/

[22] Sjaak Brinkkemper , Method engineering: engineering of information systems development methods and tools. *Information and Software Technology* 38 (1996) 275-280.

[23] Eclipse WebSite, http://www.eclipse.org

[24] Kabiri, A., Chiadmi, D. A Method for Modelling and Organazing ETL Processes. *In* : Proceedings of The second international conference on Innovative Computing Technology (INTECH 2012). *Communications in IEEE*, 241, p 146-160.

[25] Kimball, R., Ross. M.(2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd ed. Wiley Computer Publishing.

[26] Wadjinny, F., Gounbark, L., Chiadmi, D., Benhlima, L., Moujane, A. (2009).Query processing in the WASSIT mediation framework. in the seventh ACS/IEEE International Conference on Computer Systems and applications (AICCSA), May 10-13Rabat, Morocco.

[27] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.,Widom, J. (1997). The TSIMMIS project: Integration of heterogeneous information sources, *Journal of Intelligent Information Systems*, 8 (2) 117-132.