MYETL: A JAVA SOFTWARE TOOL TO EXTRACT, TRANSFORM, & LOAD YOUR BUSINESS

MICHELE NUOVO

The project follows the development of a Java Software Tool that extracts data from Flat File (Fixed Length Record Type), CSV (Comma Separated Values), and XLS (Microsoft Excel 97-2003 Worksheet file), apply transformation to those sources, and finally load the data into the end target RDBMS. The software refers to a process known as ETL (Extract Transform and Load). Those kinds of systems are called ETL systems.

The analysis involved research on the theory behind the ETL process as well as the theory behind the various phases of the applied methodology. Also an in-depth look at the design and architecture of the software has been made. To create a complete design needed to be used for the implementation, different techniques and diagrams were used to visualise and refine ideas: UML class diagrams, System Architecture Diagrams, Physical Data Model, and Project Timeline.

The implementation of the project involved the translation of the system architecture into working software using the Extreme Programming Methodology and the Java programming language. A mapping algorithm module and design patterns have been used in the implementation phase. A transformation syntax has been defined to achieve data transformation.

The testing of the software was done in the form of a unit test. A formal test plan was prepared to ensure that the main features of the system worked as defined. An error handling code implementation has been developed to avoid an unexpected crash of the system and to communicate to the user problems or errors.

1. INTRODUCTION

The Extract Transform and Load (ETL) system is the bottom line of the data warehouse. Before those systems, the business application typically had their own database that was supporting their activities. No other systems had access to this database, so it becomes an information island (Dictionary.com, 2013). The more a business was growing, the more the information island was growing rapidly because of more departments using those applications.

When a business automated those systems, more data become available, and the analytical value of the available data was soon discovered. However, to analyse that data was very complicated due to the incompatibilities among the different systems. The infrastructure created the need of collecting, analysing, and exchanging all the data and it should provide a unified view of the enterprise data was the data warehouse (gravic.com). To manage and solve the problem of the initial load of this data into the data warehouse and keep them updated, Extract Transform and Load (ETL) utilities were developed. The purpose of those utilities is to extract the data from different sources, transform them into a common format and load them into a data warehouse (Oracle®, 11 Overview of Extraction, Transformation, and Loading).

The backbone of the data warehouse architecture is constituted by ETL processes. However, ETL is not useful only for the refreshment of data warehouses. In fact, new applications have emerged with the advent of Web 2.0. Those applications integrate data which are dynamically obtained via web-service invocations to more than one source into an integrated environment. Google Maps (<http://maps.google.com/>), a web mapping service application and technology provided by Google, or Yahoo Pipes (<http://pipes.yahoo.com/>), an interactive feed aggregator and manipulator, are two examples. Under the hood, the philosophy for their operation is 'pure' ETL. Furthermore, with the evolution of the technology, interest is moving to types of data that do not necessarily follow the traditional relation format, as XML, biomedical, multimedia data, and so on (Vassiliadis and Simitsis, 2007).

Although the ETL processes are well known in the computer science field, various issues still remain open. The most important problem is the standardisation: in the market, there exists several tools that provide ETL functionality but each of these tools follows a different approach for the modelling and representation of the different steps. To create a globally accepted paradigm of thinking on this topic is an issue for the academic community (Vassiliadis and Simitsis, 2007).

The aim of this project is to build a working prototype of Java Software which allows the user to extract data from the defined sources, apply the defined transformation on those data and finally load them into a target Teradata data mart that will store the data for Business Intelligence (BI) purpose. Examples of BI tools are MicroStrategy, IBM Cognos, or Informatica which are used to produce business reports on a data mart (bi-tools.org). Various phases have been involved including research and analysis of the theory behind the ETL process, design of the System Architecture and Software Graphical User Interface (GUI), implementation in Java programming language of the defined design using the chosen methodology and testing of the implemented code. Finally a User and Maintenance Documentation has been created to give assistance and to describe the practice overview to the final users of the developed system.

2. METHODOLOGY

Agile software development is a group of software development methods. They are based on iterative and incremental development. In this development requirements and solutions evolve with the collaboration between developers and functional teams. The Agile Methodology defines the iterative approach, evolutionary development and delivery, and inspires flexible and rapid response to changes. The Agile Manifesto (Beck, et al., 2001) is a formal announcement of four key values and 12 principles for approaching software development in an iterative way. The chosen method for implementing the MyETL Java Software Tool is Extreme Programming (XP) in order to better address the problems of project risk and because XP is set up for small groups of programmers.

A. EXTREME PROGRAMMING (XP)

Extreme Programming is an Agile Method where the customer and the development team are highly involved between them. The customer drives the development creating user stories. A user story is a high-level definition of a requirement that contains the necessary information for the developers to estimate the effort to implement it. The development team delivers in an iteratively way the user stories through continuous programming, testing, and planning. The software is delivered very frequently, usually from 1 to 3 weeks. In Figure 1 a typical XP project flow is represented.

Figure 1: The highly iterative XP project flow

XP is suited best for smaller development teams. It is a fast, aggressive delivery model. It requires high collaboration and minimal documentation. For these reasons it has been chosen for the development of the MyETL Java Software Tool. Agile methodologies are an alternative to waterfall or sequential methodology for project management.

3. SOFTWARE DESIGN

Software design is the process to understand constraints and business goals, customer needs, and technologies useful for planning a software solution and create business value. The main task of the design stage is to produce the plans necessary for software production to proceed. The figure that is responsible to produce this plan is the software designer (Budgen, 2003). To produce the MyETL software design, different communication channels were used to understand business goals. The theory behind the ETL process has been studied, and it will be part of the domain knowledge. The elaboration of these communication channels produces the System Architecture that is shown in the MyETL system architecture section.

This architecture contains the following design concepts:

- *Modularity*: The software is divided into independent components called modules. Each module has its own behaviour and purpose. Those modules are able to communicate with each other when the different application layer of the architecture needs to exchange information between them.
- *Abstraction*: Each module reduces the information content in order to keep in possession only information that is appropriate for its purpose.
- *Data Structure*: A logical relation between individual modules and data.
- *Information Hiding*: Modules are designed in a way that the information inside them is not accessible to other modules that do not need them (Pressman, 2009).

Figure 2: Communication channels for the software designer to produce the software design

A. MYETL SYSTEM ARCHITECTURE

MyETL System Architecture is composed of 3 Layers: Presentation Layer, Application Layer, and Database Layer. The purpose of those layers is to separate how the information is represented from the user interaction with it.

Using the architecture in Figure 3, the three layers can communicate in both directions using specific modules to retrieve data from the database layer, apply the business rules/transformation on the extracted data, and finally display the data in a GUI representation. Once the data is displayed, the user can easily manipulate and store them again, after modifications are applied, in the database layer.

Figure 3: MyETL system architecture diagram

User interface logic changes more frequently than business logic. If a new user interface page or layout needs to be added or changed, with the architecture proposed, redistribution of the application is not necessary. Using another type of architecture, for example one in which Presentation code and business logic are embedded in a single object, will from one side decrease the code lines number, but on the other side, each time is needed to change how the data will be displayed, and the developers need to change the logic behind the modules and test it again. This operation can require a lot of effort and separate the layers that will be more efficient and less time consuming for the development of new enhancements.

Next, the 3 layers of the MyETL system architecture will be discussed in more detail using a modelling language named: UML – Unified Modelling Language (Hamilton and Miles, 2006).

B. PRESENTATION LAYER

The presentation layer is where the data is presented to the final users. This layer is composed of Graphics Unit Interface (GUI).

The presentation layer is responsible for delivering and formatting information from the application layer for further processing or display. It contains the main graphic interface of the MyETL software, which manages all the graphics units available and is accessible for the final user.

C. APPLICATION LAYER

The application layer is where the business rules are applied to the data. This layer is composed of modules that have the purpose to elaborate or transform the data retrieved from the database layer.

The Application Layer is the core of the software. Most of its functions are visible to the user through the presentation layer. For example, when the available list of tables in the repository are shown in the repository tree interface, the data is retrieved from the database layer and formatted in order to be visible adapted to fill a tree graphical representation.

D. DATABASE LAYER

The database layer is where the data is stored. This layer is composed of modules that are responsible for database connections and makes data available and reachable for the application layer.

E. DESIGN PATTERNS

The layers previously reported are designed using design patterns. A design pattern is a documented best practice or core of a solution that has been applied successfully in multiple environments to solve a problem that recurs in a specific set of situations. It can be seen as an encapsulation of a reusable solution that has been applied successfully to solve a common design problem.

The different modules in the 3 layers are designed to respond to the following design patterns:

- *Private Methods*: The purpose of this design is to provide a way of designing module behaviour so that external modules are not permitted to access to the data/operations that is meant only for the internal use. This design is used in the presentation layer to initialise the GUI components and elements.
- *Accessor Methods*: The purpose of this design is to provide a way of accessing an object's state using specific methods. This design facilitates information hiding and the module result to be more maintainable. This design is used in the application layer to provide consistent data to the presentation layer.
- *Singleton*: The purpose of this design is to provide one and only one instance of a given module during the lifetime of an application. This design is used in the database layer to be sure to have just one connection object in the application for each database.

4. IMPLEMENTATION

Most of the effort in the implementation phase went into the application layer, which is considered the core of the software. Also the presentation layer takes a lot of effort to make visible to the user all the operations behind the process and make them easily manageable through the graphic user interface.

The software was implemented using Java programme language and Eclipse IDE for Java Developers version Juno Service Release 1. There were certain criteria that were considered before to finalise this decision. First of all the language should be easy and fast to develop. In fact, the relatively short project time requires a faster development. Second, it needs to have graphical capability, because a graphic unit interface is required for the user to manage and interact with the software. Furthermore, it has to be object oriented and portable. Object oriented because of the reusability of the objects inside the programme, and portable in order to be used independently from the platform by the maximum number of users.

A. USER STORIES

A user story captures the 'who, what, and why' of a requirement. It is used instead of a large requirements document and they are written by the customer.

A user story is in the format of about three sentences of text in a customer terminology, which means without technical-syntax. It also drives the creation of the acceptance tests that are used to verify if the user story has been correctly implemented. The difference between the user story and the traditional requirements specification are mainly two:

- *The level of detail:* The user story should only provide detail to make a sensibly low risk estimation of how long the story needs to be implemented. When the story will be implemented, developers will go to the customer and receive a detailed description of the requirements face to face.
- *Focus on user needs:* Specific technology, algorithms, and data layout should be avoided. The story has to be focused on user needs and benefits instead of specific GUI layouts.

Each story is estimated by the developers to determine how long it will need for the implementation. Usually a story will get 1, 2, or 3 weeks in an 'Ideal Development Time'. With the phrase 'Ideal Development Time', it is meant that in the development there will be no distractions, no other assignments, and the developer know exactly what to do.

When the user story estimation is up to 3 weeks, it means that the story needs to be broken into multiple stories, while when a story is under 1 week it means that it has to be integrated in another user story. To have a good release plan there should be around 80 stories with a margin of 20 stories plus or less (ExtremeProgramming.org, User Stories). The source viewer module is responsible for the selection and preview of the data inside the sources files. The estimation for the user story in Figure 4 is 3 weeks and it involves one developer.

Source Viewer User Story

As a user, I want to extract data from files and store them into the software Repository so that any users can view, delete or modify the reference to those data or refresh the entire content of the Repository.

1. The "Add Table", "Rename Table", "Delete Table", "Refresh Repository" buttons will be permanent items on the main GUI of the Software.
 - i. When adding a table, a GUI should appear for file selection.
 - ii. When deleting a table, the list of available tables in the Repository will be automatically updated.
 - iii. When renaming a table in the Repository of the Software, a dialog to insert the new table name will be displayed.
 - iv. When refreshing the Repository, the whole available tables list in the Repository will be refreshed.
2. File formats can include .xls, .csv and .dat.
3. The "Load" button inside the GUI for selecting the file to upload and a preview of the data must be provided.
4. Once a file has been chosen, it will be uploaded into the Repository of the Software by pressing the "OK" button.
5. When a file is added to the Repository, it should graphically appear in the available tables list of the Repository.

Figure 4: Source viewer user story

B. MAPPING ALGORITHM MODULE

In the application layer is an implemented mapping algorithm module. This module is responsible for auto mapping columns between the source and the target tables. The auto mapping is based on the similarity of the two strings. The similarity of two strings is defined as the minimum number of single-character edits required to change one word into the other. For the auto mapping module the Levenshtein distance algorithm has been implemented.

C. LEVENSHTSTEIN DISTANCE

The Levenshtein distance algorithm gives high quality string matching. The algorithm is also referred as edit distance algorithm. It calculates the minimum number of changes that are necessary to modify one given string in another given string. The way used to calculate this changes is a matrix with the size $(L1+1) \times (L2+1)$, where $L1$ and $L2$ are the length of the first and second given string. The matrix is filled from the upper left to the lower right and each horizontally or vertically jump corresponds to a change. The result number in the lower right corner is the Levenshtein distance between the given strings.

In Figure 5 is reported the entire matrix calculation for the comparison between the following strings: "meilenstein" and "levenshtein".

		M	E	I	L	E	N	S	T	E	I	N
	0	1	2	3	4	5	6	7	8	9	10	11
L	1	1	2	3	3	4	5	6	7	8	9	10
E	2	2	1	2	3	3	4	5	6	7	8	9
V	3	3	2	2	3	4	4	5	6	7	8	9
E	4	4	3	3	3	3	4	5	6	6	7	8
N	5	5	4	4	4	4	3	4	5	6	7	7
S	6	6	5	5	5	5	4	3	4	5	6	7
H	7	7	6	6	6	6	5	4	4	5	6	7
T	8	8	7	7	7	7	6	5	4	5	6	7
E	9	9	8	8	8	7	7	6	5	4	5	6
I	10	10	9	8	9	8	8	7	6	5	4	5
N	11	11	10	9	9	9	8	8	7	6	5	4

Figure 5: An example of how the algorithm works in the comparison of "meilenstein" and "levenshtein" made with an Excel file

The similarity between the given strings is 4, in fact 4 changes need to be applied to let the two strings be the same (Carsten). Different design patters were used during the software design.

D. JAVA SINGLETON

When it is necessary to have exactly one instance of a class and this instance is required to be accessed from different points from different classes, the Singleton design pattern is used.

Figure 6: Singleton class diagram

A Singleton class maintains a private static reference to themselves and returns this reference from a static instance () method. The singleton instance is created just when the getInstance() method is called for the first time. In this way it is ensured that the instance is created just when it is needed.

The following classes in the MyETL software have been implemented using the Singleton pattern:

- *HSQLDB_Connection*: It is responsible for providing one and only one connection for the software to the HSQLDB Repository.
- *TERADATA_Connection*: It is responsible for providing one and only one connection for the software to the target Teradata RDBMS.
- *Mylcons*: It is responsible for providing one and only one access to the software's icons. This class instantiate all the used icons for buttons, background image, and so on (Geary, 2003).

5. EVALUATION

A. LIMITATION

The MyETL Java software presents some limitations. The first limitation is related to the connection to the target RDBMS. A login form to input the connection parameters is not implemented. This limits the user to connect only to the embedded Teradata schema. Another limitation is related to the repository connection. The repository is embedded in the software and only one user exists. In the case of multiple users, they will share the same repository's data. The next limitation is related to the mapping frame. Currently, it is not possible to save the mapping for the source table, and it forces the user to create the mapping every time he is using the software. Furthermore, is not possible to join between them two source tables. This limitation creates a 1 to 1 relation through the sources and the target tables. The last limitation is related to the target tables in the Teradata database. All of the target tables must be empty since a FastLoad mode has been used in order to improve loading performance time.

B. FURTHER DEVELOPMENT

On the further development of the MyETL software tool the following enhancements are planned:

- Create a login form to let the software connect to different Teradata Data warehouse. Currently, it is just possible to connect to the embedded define Teradata RDBMS data warehouse (Version 1.1).
- The possibility to eliminate duplicates rows inside the data from the software repository. A button "CLEAR SOURCE" will be added to the toolbar of the MyETL frame (Version 1.1).
- Increment the number of available sources, adding the possibility to import also Extensible Markup Language (XML) (Version 1.2).
- Manage users and different workspaces in the software repository environment. In this way different users will have their own workspace on which work on (Version 1.3).
- Have a possibility to save the created mapping frame object, where transformation and mapping column are showed (Version 1.4).
- Have a possibility to apply the mapping joining multiples source table (Version 1.5).
- Create a splash screen image that appears while the programme is loading (Version 1.6).
- Have a possibility to export the created mapping into a XLS file. (Version 1.7).
- Have a possibility to show all the data table contents and not just a sample set as currently possible (Version 1.8).
- Change the load phase in the target database using UPSERT, a combination of an UPDATE and an INSERT, instead of INSERT (Version 1.9).
- Have a possibility to connect also to the Oracle database as Target RDBMS (Version 2.0). Have a possibility to connect to MySQL database as Target RDBMS (Version 3.0).

6. CONCLUSION

It is clear therefore the importance of the Extract Transform and Load (ETL) systems in the business environment. The business applications need an infrastructure to manage all of the data among the different systems. The ETL system is the answer to this need. Researching and collect information and knowledge about the theory behind an ETL process (section III) was a challenging and interesting part. It helps me to learn how to identify trustable sources and gives me a practical example about the importance to well understand the topic before to start any implementation. The importance to follow a methodology in the software development was faster clear from the beginning of the development. The Extreme Programming (XP) methodology is used because it is fast, suited for smaller teams, and it has an aggressive delivery model. This is exactly what this project needed due to the development time and the size of the team composed only of me.

Creating the design (section IV), the test plan (section VI), and plan a roadmap to increase the capabilities of the software was an interesting challenge that made me aware about all the aspects of a software development. The Java programming language used for the implementation of the software (section V) was in my plan since the beginning of the project due to its powerful and object-oriented language. Implementing the software using Java gave me the opportunity to improve my knowledge about the Java design patterns practically while applying some of them. Defining limitations (section VII) taught me how to set a perimeter for the application's domain and how to expand it.

In conclusion, the project aim was to build a fully working prototype of Java software reflecting the ETL infrastructure system. The software can extract data from Flat File, Comma Separated Values and Microsoft Excel. String Transformation, Mathematical Transformation, Aggregation Transformation and Arithmetic Transformation can be applied to the extracted data which can be loaded into a final data mart on a relation database management system.

Due to these software features and the results of the tests, the MyETL Java Tool meets all of these goals and define a plan to extend those objectives.

A. FURTHER CONSIDERATION

Regarding personal objectives developing and integrating all the required elements to build MyETL Java software was a very involved process. A careful approach has been taken in this phase to optimise performance, integrate components, acquiring the knowledge to use all of them and facing integration problems. It may be interesting to investigate and research more in others useful components to integrate in the software and in the application of additional design pattern to improve performance in the execution of the code. ■

REFERENCES

- Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., and Thomas, D. (2001) *Manifesto for Agile Software Development*. Available at: <http://agilemanifesto.org/> (Accessed: 24 January 2013).
- bi-tools.org. (n.d.) *Business Intelligence software tools*. Available at: <http://bi-tools.org/> (Accessed: 1 May 2013).
- Budgen, D. (2003) *SOFTWARE DESIGN* (second edn). Edinburgh: Pearson Education Limited.
- Carsten, S. (n.d.) *The Levenshtein-Algorithm*. Available at: <http://www.levenshtein.net> (accessed: 15 February 2013).
- Dictionary.com. (2013) *information island*. Available at: <http://dictionary.reference.com/browse/information+island> (Accessed: 1 May 2013).
- ExtremeProgramming.org. (n.d.) *User Stories*. Available at: <http://www.extremeprogramming.org/rules/userstories.html> (Accessed: 13 February 2013).
- Geary, D. (2003) *Simply Singleton*. Available at: <http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-designpatterns.html> (Accessed: 15 February 2013).
- gravic.com. (n.d.) *Part 3 - The History of Business Intelligence*. Available at: <http://www.gravic.com/shadowbase/uses/historyofbusinessintelligence.html> (Accessed: 1 May 2013).
- Hamilton, K. and Miles, R. (2006) *Learning UML 2.0* O'Reilly Media.
- How to write meaningful User Stories*. (2010) Available at: <http://www.subcide.com/articles/how-to-write-meaningful-user-stories/> (Accessed: 13 February 2013).
- Oracle®. (n.d.) *11 Overview of Extraction, Transformation, and Loading*. Available at: http://docs.oracle.com/cd/B19306_01/server.102/b14223/etlover.htm (Accessed: 1 May 2013).
- Pressman, R. S. (2009) *Software Engineering: A Practitioner's Approach*. McGraw-Hill.
- Vassiliadis, P. and Simitsis, A. (2007) *EXTRACTION, TRANSFORMATION, AND LOADING*. Ioannina: Department of Computer Science. [Online]. Available at: http://www.cs.uoi.gr/~pvassil/downloads/ETL/SHORT_DESCR/08SpringerEncyclopedia_draft.pdf (Accessed: 1 May 2013).