UCLEAN: A REQUIREMENT BASED OBJECT-ORIENTED ETL FRAMEWORK

Payal Pahwa[1], Shweta Taneja[2] and Garima Thakur[3]

[1, 2]BPIT, Guru Gobind Singh Indraprastha University, Delhi
[1]pahwapayal@gmail.com, [2]shweta_taneja08@yahoo.co.in

[3]USIT, Guru Gobind Singh Indraprastha University, Delhi
[3]thakur_garima_27@yahoo.co.in

ABSTRACT

Data warehouse is used to provide effective results from multidimensional data analysis. The accuracy and correctness of these results depend on the quality of the data. To improve data quality, data must be properly extracted, transformed and loaded into the data warehouse. This ETL process is the key to the success of a data warehouse. In this paper we propose a conceptual ETL framework for an object oriented data warehouse design, the framework is called UCLEAN. This framework takes into account the concept of requirements of the users .The data is extracted from different UML sources and is converted into a multidimensional model. It is then cleaned and loaded in the data warehouse. We validate the effectiveness of the framework through a case study.

KEYWORDS

Data warehouse, ETL, Data cleaning & UML

1. INTRODUCTION

A data warehouse is defined as a subject-oriented, integrated, time variant, non-volatile collection of an organization's digitally stored data that augments the decision making body within the organization [1]. It is a repository of consistent historical data that can be easily accessed and utilized in order to facilitate reporting and trend analysis.As the data in a data warehouse is used for analysis and decision making purposes, it must be clean data. Data cleaning is crucial for a wide variety of applications in many industries. Data is increasing at an explosive rate, so a task to keep data correct and consistent is required. The main cause of dirty data comes from many basic mistakes such as mistaken data entry, missing fields, typos, etc. Data Cleaning is the most integral part of data transformation which is mainly done using ETL tools. ETL refers to Extract-Transform-Load process which consists of three phases. During extraction phase, data is gathered from multiple heterogeneous information sources. This gathered data has numerous discrepancies like- different naming conventions, errors, redundancies, inconsistencies, etc. Such data is called *dirty data.* So, before feeding this dirty data into the warehouse we need to pre-process the data. This is done in the Transformation phase, where we transform values of inconsistent data, clean "bad" data and filter redundant data. And finally, in the Loading phase we load the cleaned data into the data warehouse.

Another important aspect of data warehouse is designing a data warehouse. Designing can be carried out using various sources.UML is one of them. Unified Modelling Language (UML) has become an industry standard for object modelling during analysis and design steps of software development in [3], [4]. We have used UML in our design which is a widely accepted modelling language. So, we need not learn new notations and methodologies. Another excellent feature of UML is that it is an extensible language, that is, it provides features like stereotypes, tagged values etc. to add new elements to domains like business modelling, web applications etc. In this paper we describe a conceptual ETL framework for an object oriented data warehouse design, *UCLEAN*. This framework takes into account the concept of requirements of the users .The data is extracted from different UML sources and is converted into a multidimensional model. It is then cleaned and loaded in the data warehouse.

The paper is organized as follows. Section 2 presents the related work done in this field. In Section 3, we have proposed the ETL framework in which our case study is discussed. Finally the conclusion is stated.

2. LITERATURE REVIEW

In [2], the goal-oriented requirement elicitation technique is combined with UML model concepts to enhance the designing process of a data warehouse. In [3], a survey of various techniques of DW designing and requirement analysis is presented. A unique approach has been discussed in [4] that are based on OO concepts of modelling the requirements of a data warehouse so that the completeness, consistency, traceability and reusability of requirements and other phases of design process can become cost effective. A goal-oriented requirement elicitation and analysis approach has been addressed in [5] where goals have been used to classify various concerns in DW design into functional and non-functional.

A framework to map a UML class diagram into a multidimensional snowflake model by illustrating an example of a D.I.E.T case study is designed in [6]. [7] Throws light on the process of DW design and architectural development by highlighting important aspects to be covered while constructing a DW. A UML profile based on conceptual level usage of DW has been discussed in [8]. Here, the usage has been classified into four perspectives and more emphasis has been put on the user's role in it. A graph based modelling techniques has been designed in [9] for conceptual design of object oriented multidimensional models. In [10], an object-oriented approach for extraction, transformation and loading process for data extraction is discussed. Also, a metadata-driven approach has been suggested for ETL process design in [11]. In [12], ETL process has been demonstrated as a state-space where each ETL workflow is signified as a state and the state space is created by set of state transitions. Another unique way of extraction and loading of data in real-time environments has been addressed in [13] by introducing a concept of ETL idle time. This timestamp is used to distribute data loading by extracting few sources in real-time manner and rest in conventional manner.

In [14], certain algorithms for detection and elimination of approximate duplicates in a data warehouse are shown. [15], highlights the limitations of algorithms presented in [14] along with the appropriate solution to deal with it. A sequential token-based framework for data cleaning has been designed in [16]. Its more elaborated version has been provided in [17]. In [18] , the merge-purge approach to handle duplicate records by employing a sliding window technique is described. The data quality issues have been classified into single-source and multi-source in [19]. Moreover, the discussion has been extended to schema-level and instance-level. The duplicate detection and eradication techniques have been extended to fuzzy duplicates in [20]. In [21], it is claimed that apart from elimination of duplicates, the integration process also performs the transformation of data into the form desired by the target application. But, nobody has modelled the entire ETL framework and applied data cleaning on it, as we have done. In our framework, we have collected data from different sources and made a multidimensional model for the design of a data warehouse. The model is represented by a snowflake schema. In the previous research, nobody has done it. The main advantage of our method is that it is based on a well known standard modelling language. So, designers can avoid learning a new language for multidimensional systems. To the best of our knowledge, we are the first people to represent a UML snowflake diagram that integrates heterogeneous UML data sources.

3. UCLEAN: PROPOSED FRAMEWORK

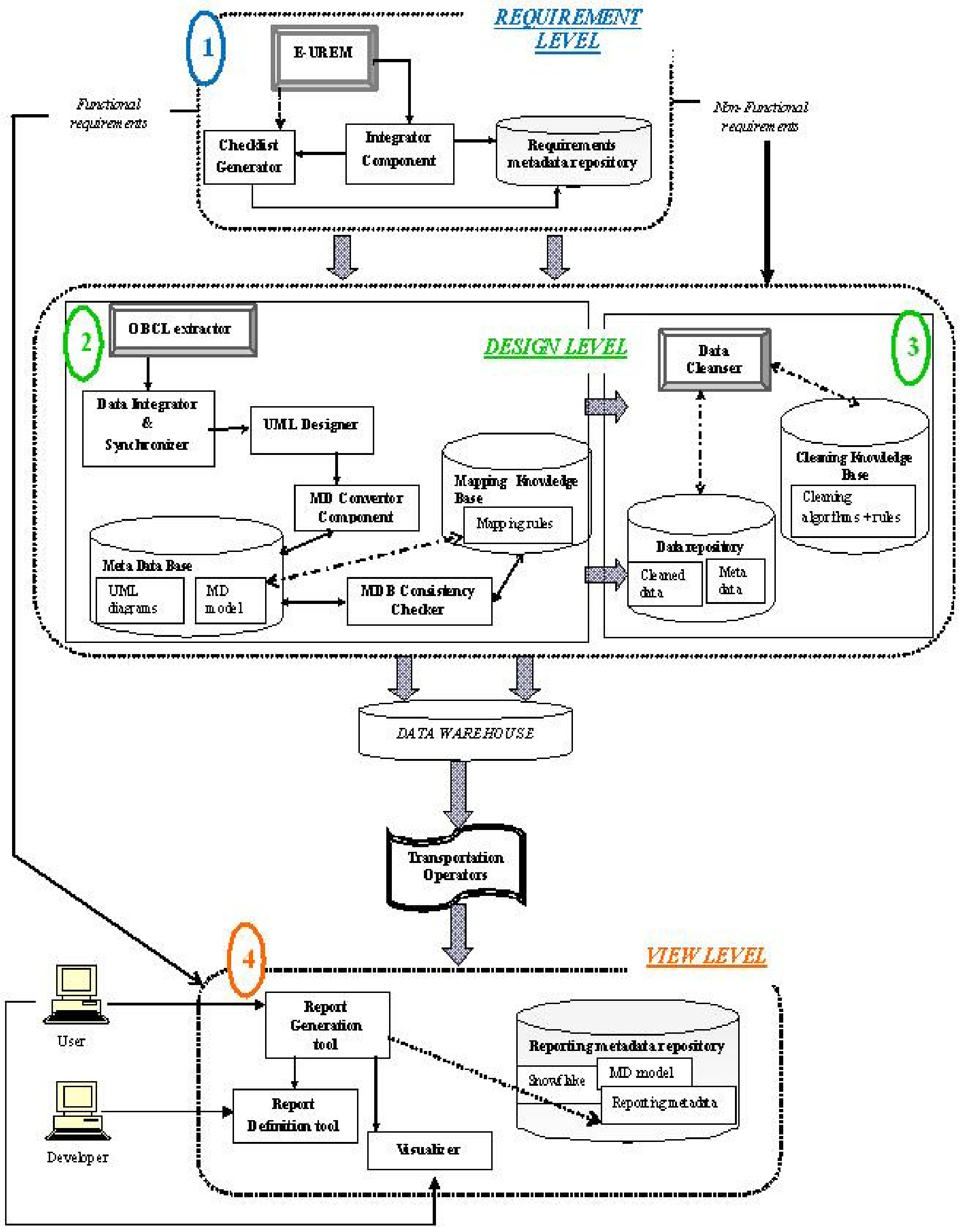

Figure 1. Proposed UCLEAN framework

3.1. Components of the Framework

The framework is divided into three levels namely- Requirements level, Design level and View level. At the *Requirement level*, the user requirements are elicited and classified into functional and non-functional requirements. The next level, that is the design level, consists of two phases. In the first phase, we begin with the DW designing on the basis of non-functional requirements that come from the previous level. In the second phase, we perform data cleaning in order to eliminate the redundancies from the data integrated from heterogeneous multiple information sources. And finally the view level, several reports are defined and executed based on the functional requirements specified by the user in the first level. Each level comprises of a number of components to manage particular tasks along with comprehensive metadata repositories to speed up the whole process.

Now, let us discuss the working of the above mentioned levels and their respective components in more details:

Table 1. Components of the framework

1. Requirement Level	
E-UREM	This model is a variant of UREM model described in [2]. By means of E-UREM we gather the requirements from the users and after a thorough analysis categorize them as functional & non-functional.
Integrator	This component integrates the collected requirements.
Checklist Generator	It generates a checklist of all the integrated requirements so that omissions and redundancies can be taken care of before starting with the designing process.
2. Design Level PHASE I	
OBCL Extractor	This is a significant component of the design level. It helps in extracting major objects and classes from data gathered from multiple operational systems prevailing in an organization.
Data Integrator & Synchronizer	It integrates the objects and classes extracted from multiple sources of information. Also, it synchronizes them in order to resolve conflicts that might be generated due to differences in formats, schema structure, naming conventions etc
UML Designer	The finalized objects and classes are modeled as UML class diagrams by this component. We can also create object diagrams by instantiating the classes designed in the class diagrams. In this paper, we mainly focus on class diagrams.
MD Convertor Component	The UML diagrams are converted to multi-dimensional models represented in the form of snowflake schemas. The conversion is done by applying certain mapping rules that help in mapping the classes in facts and dimensions [2].
MDB Consistency Checker	This component is responsible to keep a check on the violation of the mapping rules. It checks that all the mapping rules [2] are appropriately utilized and the resulting MD model is in accordance with those rules.
3. Design Level PHASE II	
Data Cleanser	This component comes into picture in second phase of design level. It is used to perform data cleaning in order to recognize exact and approximate duplicates in the data so that we enhance the data quality by elimination all the redundancies and inconsistencies (homonyms and synonyms).
4. View Level	
Report Generation Tool	This element generates customized reports based on the definition specified by the users.
Report	It defines and customizes the reports to be presented to the user.

Definition Tool	
Visualiser	This component enables the users to view the reports, UML diagrams and MD models in the desired format. Multiple views can be generated and visualized at a given point of time.

3.2. Metadata Support

We have provided metadata and repositories at every level of the framework to augment different tasks occurring at a particular level. Detailed discussion on them has been given below.

Table 2. Metadata at different levels

4. Requirement Level	
Requirement metadata repository	It stores all the metadata related to requirement gathering and analysis. Requirement Checklists are also stored here.
5. Design Level PHASE I	
Metadata Base (MDB)	All the UML class diagrams and the Multi-dimensional models are stored here for any future reference.
Mapping Knowledge Base (MKB)	The mapping rules that help in converting the class diagrams into their corresponding Multi-dimensional models are stored in this repository.
6. Design Level PHASE II	
Data Repository	After applying cleaning techniques all the cleaned and transformed data are stored in this repository. Their source relations, class diagrams, etc. are also stored here.
Cleaning Knowledge Base (CKB)	The techniques being used for data cleaning and data & calculations related to their operations are stored here. Example- match scores, thresholds, etc.
7. View Level	
Reporting Metadata Repository	This repository stores all the data required for reporting and analysis by the users at their end. Generated reports, their definitions, reporting metadata, UML diagrams, snowflake model, etc. can be viewed with the help of visualize component.

3.3. Framework Operation

3.3.1. Requirement Level

Requirements can be classified into functional & non-functional requirements. In context of a DW, functional requirements specify *what* data is to be stored in it, while non-functional requirements specify how this information should be provided to facilitate reporting and analysis in a correct manner. We use E-UREM model for requirement elicitation and analysis. This model is a variant to UREM model which has been described in detail in [2], [3]. In this paper, we will just give a brief about it and focus more on our model, E-UREM.

E-UREM utilizes certain techniques for gathering the requirements and their analysis so that they can be classified as functional & non-functional requirements.

Table 3. E-UREM features and approaches [2], [3]

Functional Requirements	
User-driven	• Information requirements are gathered from different business users and then integrated and made consistent to have unified schemata. • Bottom-up technique.

	• Active involvement of the users. • Dependent on knowledge of business users
Supply-driven	• It begins with collection and analysis of operational data sources in order to have an idea of available data. • Low participation of the users and their requirements. • Bottom-up technique • Resulting Schemata is stable as it is based on quality of the transactional data which does not changes often like user requirements.
Goal-driven	• Revolves around the business strategies and objectives formulated by the top-management • Involves top-management only. • Top-down approach • Tales care of progressive goal refinement from the top-management people.
Non-Functional Requirements	
Scenario-Based	• Various scenarios are provided to different users, stakeholders, top-level management people and others based on real life situations. Their feedbacks are collected and examined so that the specified expectations and constraints can be converted into requirements for addition into the checklists. • Some "soft-scenarios" (based on dummy situations or assumptions) are also generated. "Soft scenarios" represent those scenarios that do not have any straightforward definitions but help in what-if analysis for gathering alternate solutions to conflicting requirements.

After the requirements have been elicited, analyzed and categorized as functional or non-functional, the *integrator component* integrates all those requirements. Then, the collected requirements are fed into the *checklist generator* which creates a requirement checklist clearly specifying the functional & non-functional requirements that will be needed in the later stages of the process. The checklist can be stored in the repository. In addition to it, the checklist is again reviewed by the users and the feedback is fed into the integrator so that it can perform relevant additions, deletions and modifications to keep the checklist in the most consistent state. Other relevant data related to the requirements is stored in the *requirement repository.*

3.3.2. Design Level (Phase I)

After the requirements have been integrated and categorized, the non-functional requirements are utilized in the designing of the data warehouse.

The designing procedure begins from this particular phase. *OBCL extractor* identifies the various objects and classes from the heterogeneous operational data. Since, data coming from various operational sources might differ in formats and schemas, thus, we need to make them consistent. The *data integrator & synchronizer* component performs this job. It synchronizes the objects and classes belonging to the same real world situation under unified schema and structure. Then, these classes are represented in UML class diagram by the *UML designer.* It clearly states the relationship existing between the different classes and their instances (object diagram not discussed in this paper). Furthermore, these classes and relationships between them are mapped to snowflake schema (multi-dimensional models) by applying certain mapping rules that have been described in details in [2]. We describe this whole process by means of a University Management System case study.

Case Study: University Management System primarily helps in the development, implementation and maintenance of essential university administrative systems. A University offers various academic programs like- B.Tech, M.Tech, etc. Each program comprises of numerous departments as shown later. Employees of the university can be teaching or nonteaching staff members. The teaching faculty can further be classified into the regular faculty and guest faculty.

The different classes identified by the OBCL extractor and the corresponding UML class diagram are shown below:

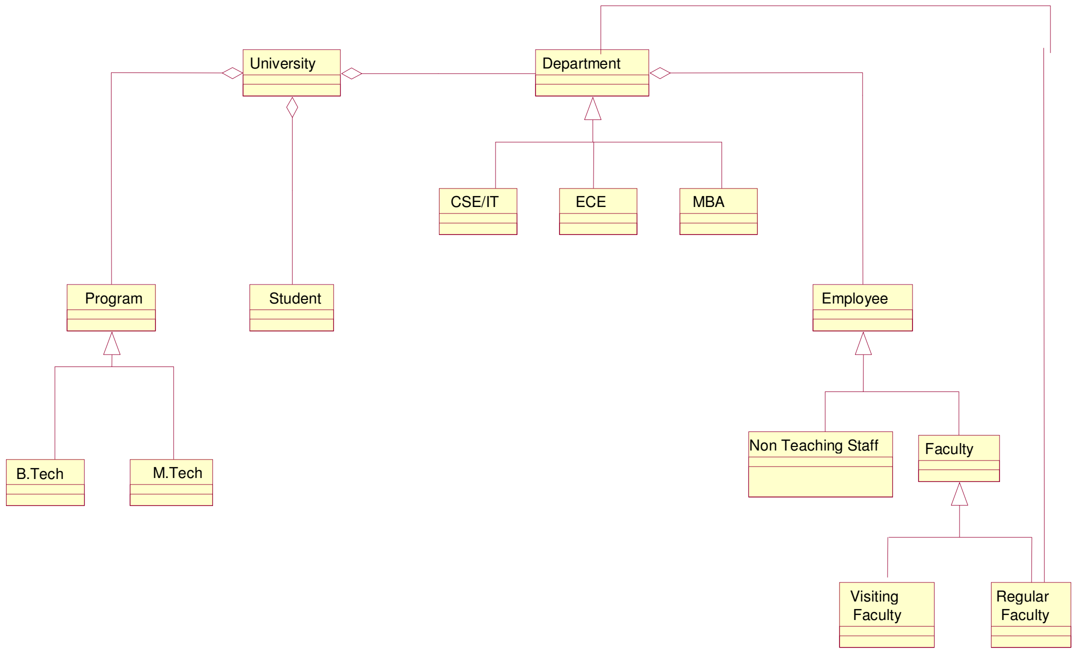

Figure 2. Class diagram for University Management system

The mapping rules that enable us to create snowflake schema from the class diagram have been depicted in the following table:

Table 4. Mapping rules [2]

UML diagram components	**Snowflake schema**
UML classes	Facts & dimension tables
Aggregation classes	Dimension hierarchies
Class attributes	Measures/ dimension attributes
Generalizations	Aggregation levels

These mapping rules are stored in *mapping knowledge base.* After the application of mapping rules as described above we get the resulting snowflake schema as shown below:

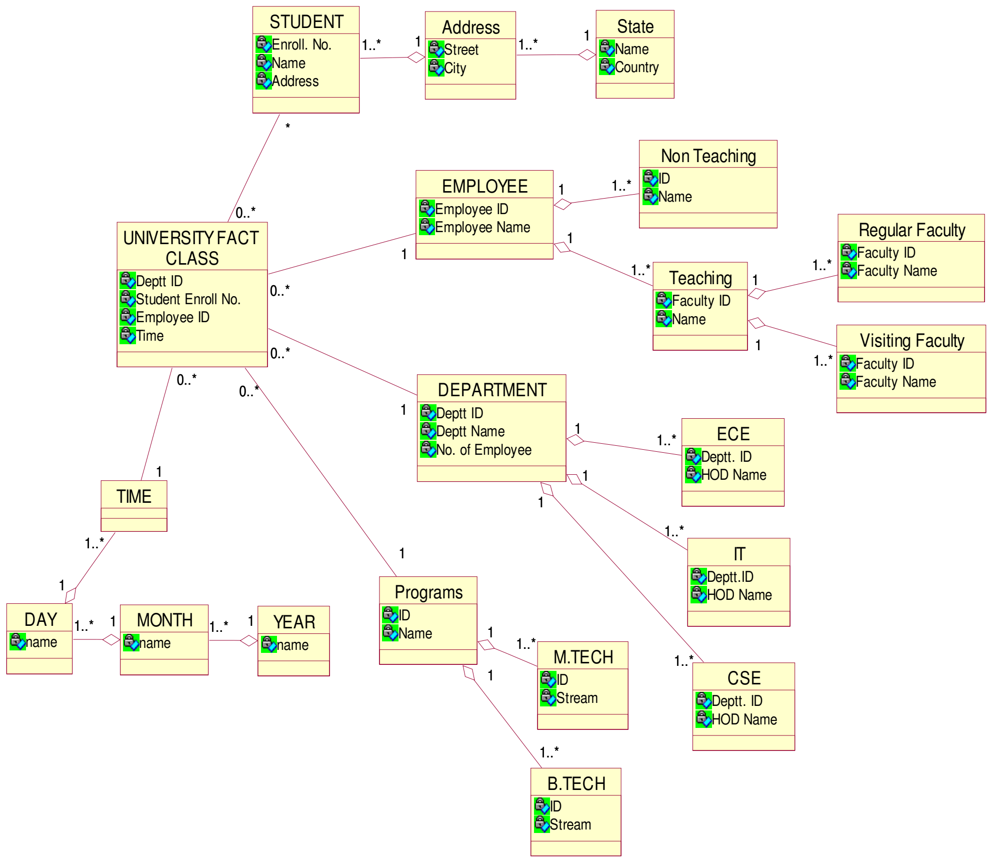

Figure 3. Snowflake schema for University Management System

All the UML diagrams and snowflake models generated during this phase are stored in the *meta knowledge base.* In order to keep a check on the mapping rules and their consistency we have introduced an element in our framework called *MDB Consistency checker.* This component is responsible for maintaining uniformity and stability amongst the UML diagrams, snowflake models and the mapping rules. This step verifies that we have applied the apt mapping rule appropriately on the required UML diagram component. Moreover, it examines that the mapping rules are updated correctly.

This phase is analogous to extraction process of the ETL paradigm for conventional data warehouse design.

3.3.3. Design Level (Phase II)

While performing the integration and mapping processes in the previous stage certain errors, redundancies or inconsistencies might creep up in the data. These kinds of errors are often termed as '*dirty data'*. So, in this phase of the design level we will pre-process (clean) the dirty data in order to improve the quality of data to be loaded into the warehouse. Hence, this phase becomes the transformation stage of our proposed framework. The snowflake model generated in the first phase is fed into the *DW Cleanser* component which applies certain techniques to recognize the dirty data (both exact and approximate duplicates) and eliminates them in order to make the data consistent and clean.

Two repositories have been provided in this phase of design level. The *data repository* contains all the cleaned and transformed data which will be loaded in the data warehouse. And, the *cleaning knowledge base* stores the techniques, algorithms and other relevant data that aids in the identification and removal of dirty data.

Once the data is cleaned and stored in the repository then from there it is loaded into the data warehouse.

3.3.4. View Level

The loaded data is now available to the users for reporting and trend analysis via *transportation operators*. They transport data from the warehouse at the user's end in the form of multiple views. After that the users can view the data from different dimensions according to their needs. The *report generating component* takes care of a variety of user's needs and produces customized reports based on the functional requirements specified by them. The report definitions are handled by the *report definition tool* that checks the format, availability and consistency of information to be used for generating views for numerous users. Necessary metadata is stored in a repository called *reporting metadata repository*.

4. CONCLUSION

ETL process is an integral component of data warehousing environment. In this paper we have presented UClean, a novel theoretical framework for ETL process. The framework is divided into different phases. In the initial stage the requirements specified by the users are taken into account for designing and Object-oriented data warehouse. These stated requirements are then categorized as functional and non-functional requirements. In the next phase, non-functional requirements are used for extracting object-oriented constructs (objects & classes) in order to design the UML diagrams. We have also suggested mapping rules to convert UML sources to multi-dimensional models. UML data is then loaded in the warehouse after the process of data cleaning so that the quality of stored data is improved. Lastly, functional requirements are used in order to facilitate customized reporting and analysis for the users at their end. Comprehensive metadata and repositories have been provided to augment the operation of framework at every phase. We are in the process of implementing the framework using JAVA at the front-end and Oracle 10g at the back-end.

REFERENCES

[1] W. Inmon & R. Hackathorn, (1994) *Using the data warehouse*, Wiley-QED Publishing, Somerset, NJ, USA

[2] Gupta, Chauhan, Kumar & Taneja, (2011) "UREM-A UML-Based Requirement Engineering Model for a Data Warehouse", In Proceedings of the *5th National Conference; INDIACom-201, Computing For Nation Development,* New Delhi, India.

[3] Golfarelli, M, (2009) "From User Requirements to Conceptual Design in Data Warehouse Design-A Survey", DEIS, University of Bologna.

[4] Lu, C.W., Chu, W.C., Chang, C & Wang, C, (2007) "A Model-based Object-oriented Approach to Requirement Engineering (MORE)", *31st Annual International Computer Software and Applications Conference*, IEEE.

[5] Lamsweerde, A, (2001) "Goal-Oriented Requirements Engineering: A Guided Tour", In Proceedings of the *5th IEEE International Symposium on Requirements Engineering*, August, pp. 249-263.

[6] Pahwa, P & Taneja, S, (2011) "Object-Oriented Multidimensional Modeling Using Extended UML Features", In Proceedings of *International Conference on Upcoming Trends in Information Technology (UTIT'11)*, Ludhiana, India.

[7] Rifaiea, M., Kianmehrb, K. & Alhajjb, R, (2008) "Data Warehouse Architecture and Design", IEEE.

[8] Stefanov, V. & List, B, (2007) "A UML Profile for Modeling Data Warehouse Usage", In Proceedings of *ACM Foundations and Applications ER 2007 Workshops CMLSA*, FP-UML, ONISW, QoIS, RIGiM, SeCoGIS, New Zealand.

[9] Sarkar, A., Choudhury, S., Chaki, N. & Bhattacharya, S, (2009) "Conceptual Level Design of Object Oriented Data Warehouse: Graph Semantic Based Model", *INFOCOMP Journal of Computer Science*, pp. 60-70.

[10] Pei, Y., Xu, J. & Wang, Q, (2010) "One CWM-based Data Transformation Method in ETL Process", IEEE.

[11] Simitsis, A., Vassiliadis, P. & Sellis, T, (2005) "Optimizing ETL Processes in Data Warehouses", In Proceedings of the *21st International Conference on Data Engineering*, IEEE.

[12] Krishna, R. & Sreekanth, (2010) "An Object Oriented Modeling and Implementation of Web Based ETL Process", *IJCSNS International Journal of Computer Science and Network Security,* Vol.10 No.2, February.

[13] Golfarelli, M., Maio, D. & Rizzi, S, (1998) "The Dimensional Fact Model: A Conceptual Model for Data Warehouses", *Int. Journal of Cooperative Information Systems* (IJCIS).7, 215-247.

[14] Monge, A, (1997) "Adaptive Detection of Approximately Duplicate Database Records and the Database Integration Approach to Information Discovery", In Proceedings of the *SIGMOND 1997 workshop on Data mining & knowledge discovery* .

[15] Pahwa, P., Arora, R. & Thakur, G, (2010) "An Efficient Algorithm for Data Cleaning", *International Journal for Knowledge- Based Organizations*, IGI Global, Vol 4.

[16] Tamilselvi, J. & Saravanan, V, (2008) "A Unified Framework and Sequential Data Cleaning Approach for a Data Warehouse", *International Journal on Computer Science and Network Security*, Vol 8.

[17] Tamilselvi, J. & Saravanan, V, (2009) "Detection and Elimination of Duplicate Data Using Token-Based Method for a Data Warehouse: A Clustering based Approach", *International Journal of Computational Intelligence Research*, Vol. 5.

[18] Hernandez & Stolfo, (1998) "Real-World Data is Dirty: Data Cleansing and the Merge/Purge Problem", *Data Mining and Knowledge Discovery*, 2(1), 9-37.

[19] Hang-Hai, D. & Erhard, R, (2000) "Data Cleaning: Problems & Current Approaches". *IEEE bulletin of the technical committee on Data Engineering*, 24, 4.

[20] Redman, T, (1998) "The impact of Poor Data Quality on the Typical Enterprise", *Communications of the ACM*, 41(2), 79-82.

[21] Shahri, Shahri, S., Hellerstein, J. & Raman, V, (2001) "Potter's Wheel: An interactive Data Cleaning System", In proceedings of *International Conference on Very Large Databases*.

[22] Elmasri, R. & Navathe, S.B, (2000) *Fundamentals of Database Systems,* Addison Weasely Pub Co. ISBN 0201542633.

[23] Ponniah, P, (2001) *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, pp 402.

[24] Inmon, W, (1991) *Building the Data Warehouse*, Weasely Publications, pp 23 (1991)

[25] Wikipedia, http://en.wikipedia.org.

[26] Booch, Rumbaugh, & Jacobson, (2001) *The Unified Modeling Language User Guide*, Addison-Wesley Longman, p.482.

Authors

Dr. Payal Pahwa

Dr. Payal Pahwa is presently a professor in Bhagwan Parshuram Institute of Technology, I.P. University, Delhi, India. She received her Ph.D. in Data Warehousing from Jawaharlal Nehru University, Delhi, India. Her areas of interest are Data Warehousing and Database Management Systems.

Shweta Taneja

She is a senior lecturer in Bhagwan Parshuram Institute of Technology, I.P. University, Delhi, India. She is pursuing her Ph.D. from Delhi Technical University, Delhi, India. Her areas of interest are Data Warehousing and Database Management Systems.

Garima Thakur

She has completed her M.Tech in Information Technology from USIT, I.P. University, Delhi, India. She has completed her B.Tech in Computer Science from the same university. Her areas of interest are Data Warehousing and Database Management Systems.