A NEW REVERSE ENGINEERING APPROACH TO CONVERT FORM FILL FORMAT DOCUMENT INTO UML CLASS DIAGRAM

Mohammad I. Muhairat [1] And Akram Abdel Qader [2]

[1] Department of Software Engineering, Al-Zaytoonah University of Jordan, Amman, Jordan
[2] Department of Multimedia Systems, Al-Zaytoonah University of Jordan, Amman, Jordan

ABSTRACT

This paper propose a new reverse engineering approach to convert a form fill format document into a set of related tables that can be used to generate the entity relationship diagram. A relationship between the set of tables is generated. In addition, the entity relationship diagram will be converted into a UML class diagram. However, this approach will be very helpful for researchers and practitioners in Software Engineering field, since most of the reverse engineering approaches are based on source code. This approach is tested by using several word form fill format documents and the results show a high accuracy rates comparing with the forward engineering.

KEYWORDS

UML, Class Diagram, Word Document, Normalization, (RE) Reverse Engineering, (FE) Forward Engineering.

1. INTRODUCTION

At present time, the growth of using information technology, make the researchers focusing on reverse engineering methods and techniques to reduce the efforts of producing new software from legacy systems or manual processes. Software re-engineering is the examination of gathering information, analysis, and alteration of an existing software system to reconstruct the infrastructure for new systems in a new form. Software re-engineering is reorganizing and modifying existing software systems to make them more maintainable [1].

The reversed engineering approach of the software re-engineering process is used to identify the system's components and their interrelationships to create representations of the system in another form of its design and specification [2]. In addition, the reverse engineering technique is widely used to reconstruct or recover design systems [13].

The reverse engineering is reconstruction or decomposing existing code, analyzes it to start the redesign process through the use of UML notations where the class diagram is drawn to clarify the new system process flow.

Although software reverse engineering is used in software maintenance process, it is applicable to many problem areas. In the context of software engineering, as defined in [24], reverse engineering is "the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction". Another definition is provided by IEEE [25], "reverse engineering is the process of extracting software system information (including documentation) from source code".

In this paper, the reverse engineering approach is used to construct the UML class diagram from a form fill format document. The proposed approach will be very useful in the field of reverse engineering and in business based system construction. Most applications and models in this area were based on source code in web application [14]. The increase of business automation and the huge data forms that need to be automated increase the need to build a software approach to construct pre-processing of normalized tables as a base of start point in the automation process.

The main contributions of this paper are:

- The flexibility of building pre-processed normalized tables from a form.
- A dynamic model that will deal with and language used in the form template.
- Extremely low cost of adapted software for different applications.
- Efficient and cheap means of gathering information from a form fill format document.

This paper is organized into four sections: section 2 discusses the related work and literature review, section 3 describes the proposed approach, section 4 the conclusion, and the suggested potential future work.

2. RELATED WORK

Many research projects were conducted to apply the reverse engineering approaches to reuse the infrastructure of the manual or existing system in a new form.

Many researchers were use the existing code of the system, in [15] Tran, et al. proposed an approach of reverse engineering based on the reuse of existing process code. Others research where done using a hybrid reverse engineering approach that combining metrics and program visualization as in [16]. Other reverse engineering approaches proposed to recover design pattern information from source code [17]. Mohammad, et al. in [5] they propose a reverse engineering approach to process the GUI to reconstruct the class diagram and verify it by using petri nets models. Other approaches use the GUI to perform testing [18]. Many researches were done to enhance web applications as in [18, 19, and 20].

Others maintain web sites [9], and to automate the construction of sequence diagrams for dynamic web applications [10].

Agarwal ans Sinha [3] perceived that the class diagram and interaction diagram is an easy, and user friendly notations. Te, eni et al [4] affirms that over fifty three percent of software projects use class diagrams. The implementation of a class diagram is in direct relation with most object oriented programming languages such as Visual Basic.Net, Java, and other languages, where each class diagram construct an interface or code class.

Chu et al. [26] introduced four phases of reverse engineering processes based on source code parsing. The first phase: context parsing phase to analyze the source code and extract syntactic and semantic. The second phase, they analyzing the components that matched the specifications of the extraction model are copied from the sources and stored partially in a component repository as in [27]. In the third and fourth phases, they apply design recovering methods by combine both structural and knowledge representation from the previous phases.

Finally, in a recent research, Akram, et al. [21] propose a reverse engineering approach using a pattern recognition technique to reconstruct a class diagram from distributed graphical user interfaces (GUI).

In this paper a reverse engineering approach is used to transform a form fill format document into a class diagram through the use of word processing recognition.

2. THE PROPOSED APPROACH

The proposed reverse engineering approach for constructing a class diagram from a form fill format document will consist of the following processes:

1. Capturing process: This process will capture form fill format document components and store them in a temporary table-not normalized.
2. Normalization process: This process will scan all records in the temporary table and normalized them. Also, a relationship will be defined and the result will be stored in a new normalized table.
3. Mapping process: This process will translate the normalized table to entity relationship diagram.
4. Translation process: This process will translate the entity relationship diagram to class diagram.

Figure 1 shows the above four processes and the inputs and outputs for each process.

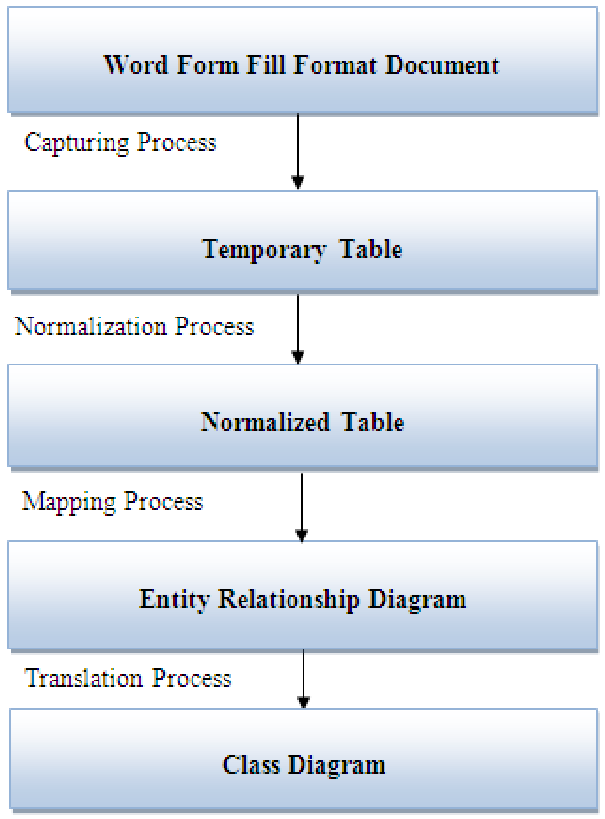

Figure1. The Proposed Reverse Engineering Approach

2.1. Capturing Process

As demonstrated, the capturing process is the first one and it will scan the form fill format document and identify each component and store its information in a table called temporary table. However, Figure 2 demonstrates the algorithm of capturing process.

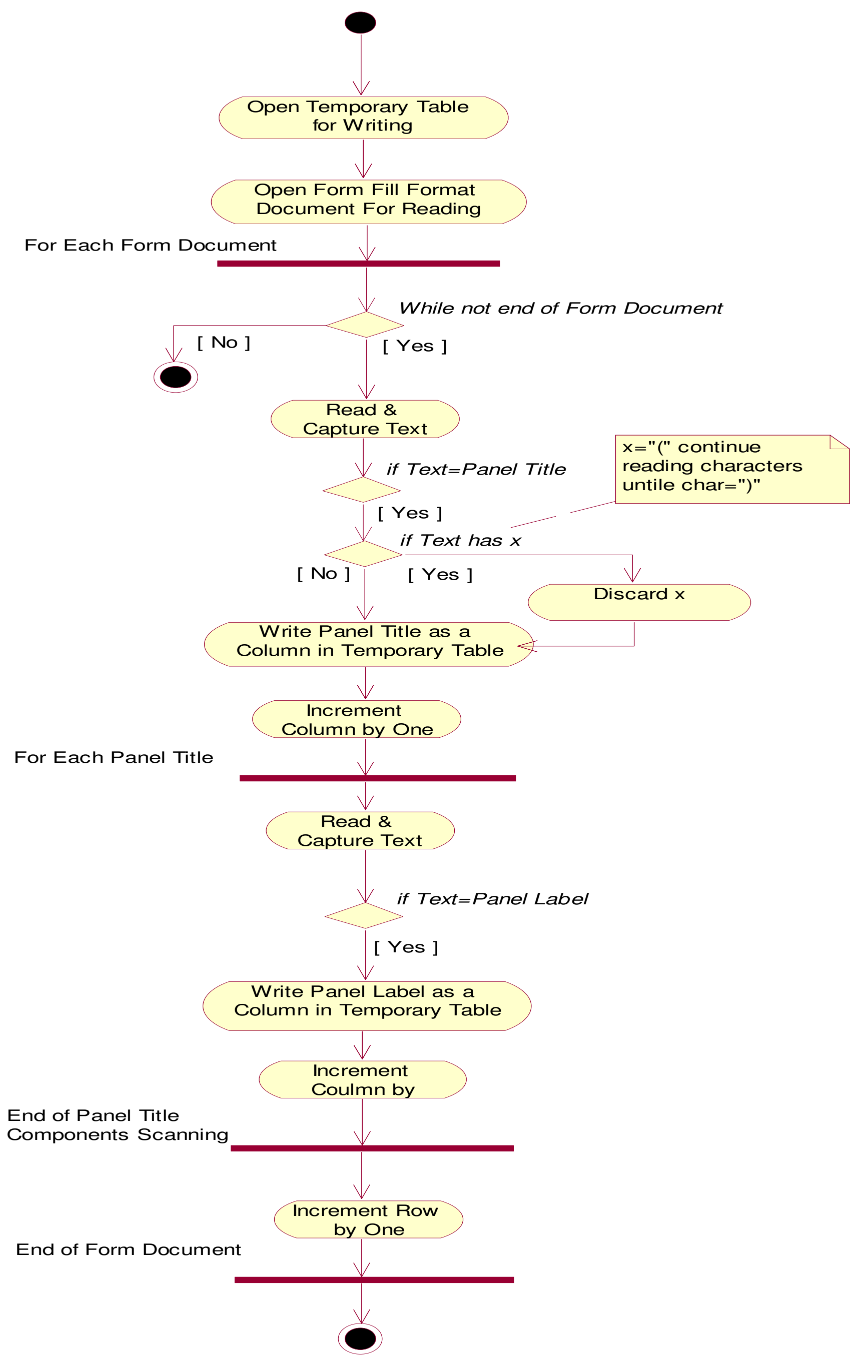
Open Temporary Table for Writing
Open Form Fill Format Document For Reading
For Each Form Document
While not end of Form Document
[No]
[Yes]
Read & Capture Text
if Text=Panel Title
x="(" continue reading characters untile char=")"
[Yes]
if Text has x
[No]
[Yes]
Discard x
Write Panel Title as a Column in Temporary Table
Increment Column by One
For Each Panel Title
Read & Capture Text
if Text=Panel Label
[Yes]
Write Panel Label as a Column in Temporary Table
Increment Coulmn by
End of Panel Title Components Scanning
Increment Row by One
End of Form Document

Figure 2. Capturing Process

The result of executing the capturing process will be stored in a temporary table as demonstrated in table 1. This table may contain redundancy data, such as column names. The Explanation of table 1 contents will be discussed to ensure the understand ability of it.

Table 1: Temporary Table

CN_i	CN_{i+1}	…	CN_n

Where is:
CN: Column Name,
i: Column index, and,
n: Maximum Number of Columns.

2.2. Normalization Process

To discard the redundancy data in Table 1, we proposed the second process to translate the temporary table into normalized table based on using the normalization process.

The Normalization Process consists of two algorithms:

- **First algorithm is The Translation Algorithm:**
  The translation algorithm is used to translate a temporary table to a normalized table, in this phase the process will discard the redundancy problem. The translate algorithm is represented in figure 3.

- **Second algorithm is The Relationship Algorithm:**
  The relationship algorithm is used to define the relationship between the tables that resulted from translation algorithm. The process follow of the relationship algorithm is represented in figure 4.

After executing normalization algorithms the result will be a normalized table as demonstrated in table 2.

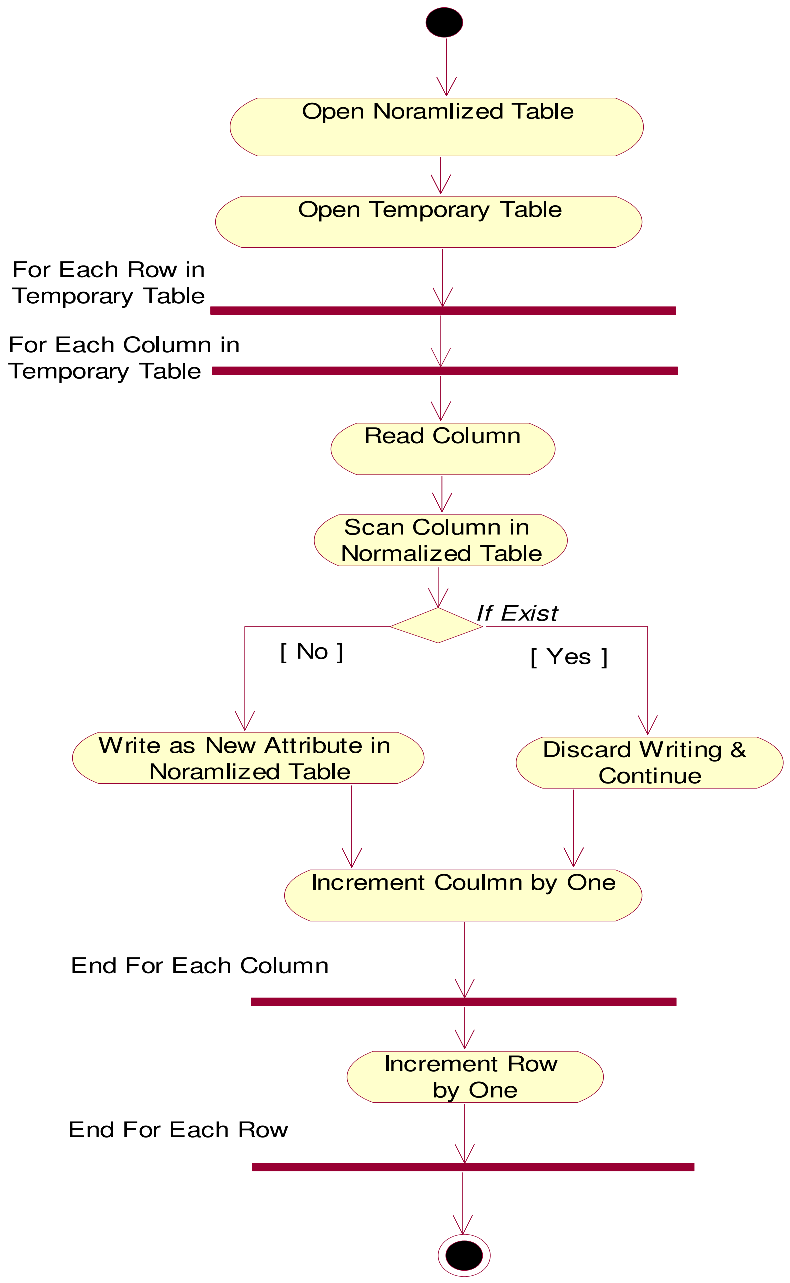
Open Noramlized Table
Open Temporary Table
For Each Row in Temporary Table
For Each Column in Temporary Table
Read Column
Scan Column in Normalized Table
If Exist
[No]
[Yes]
Write as New Attribute in Noramlized Table
Discard Writing & Continue
Increment Coulmn by One
End For Each Column
Increment Row by One
End For Each Row

Figure 3. The Normalization Process (Translation Algorithm)

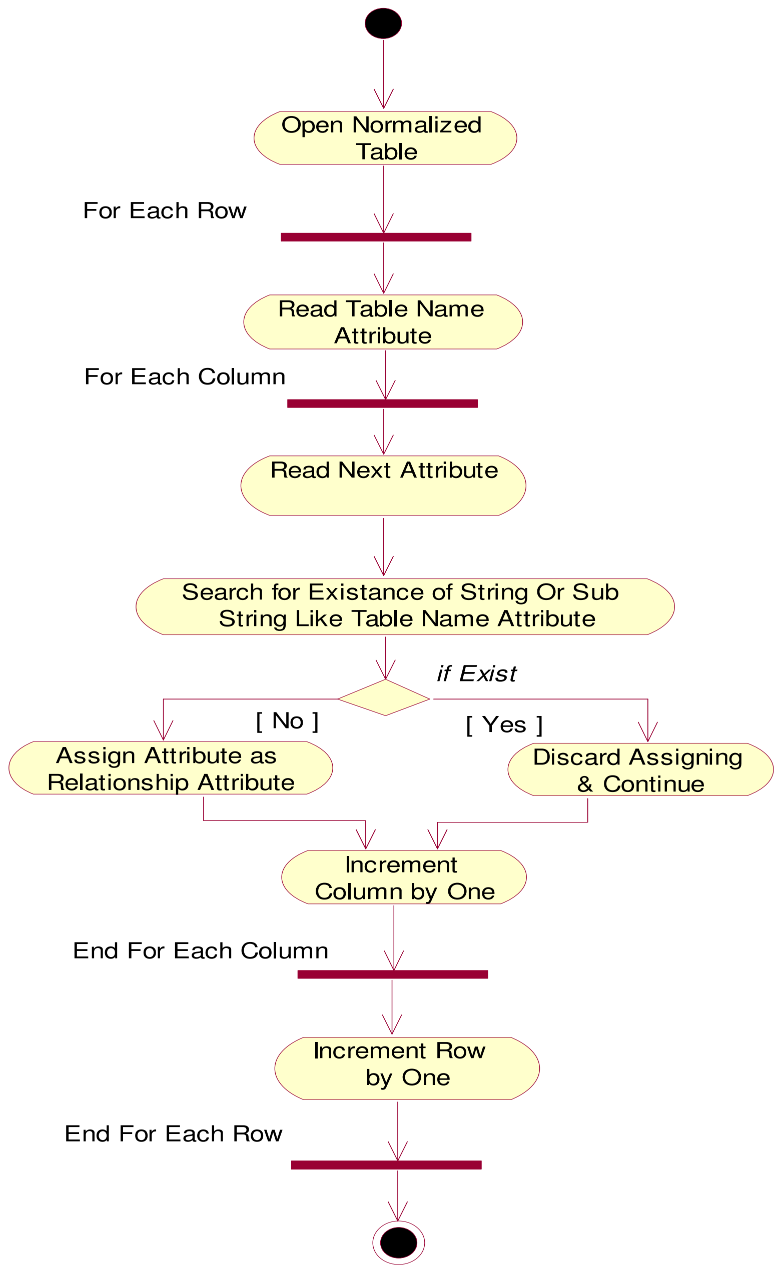

Figure 4. The Normalization Process (Relationship Algorithm)

2.3. Mapping Process

The third process will be the mapping process. In this process we proposed an algorithm to construct and build an entity relationship diagram. The mapping process will deal with the normalized table that contains the table of tables in addition to the relationships between them. By scanning this table using the proposed algorithm the result will be a database schema as demonstrated in figure 5.

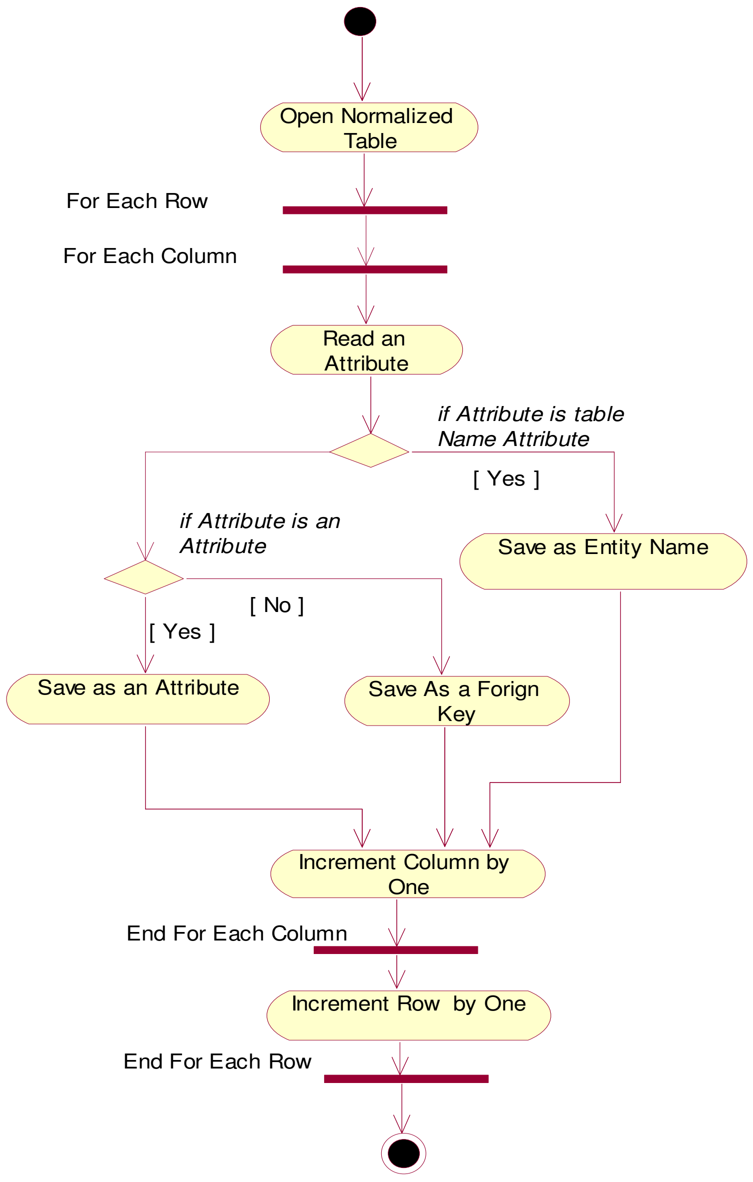

Figure 5. Mapping Process

2.4. Translation Process

The last process will be the translation process. In this process we proposed an algorithm to construct and build the class diagram. This process will utilized the mapping process results to generate the class diagram from each table created in section 2.3. The translation process is shown in figure 6. Besides that, the class diagram constructed will be translate the table name into class name and table fields into class attributes and we add the main default operational functions as (Insert, Update, Delete and Search), this class diagram is shown in the figure 7.

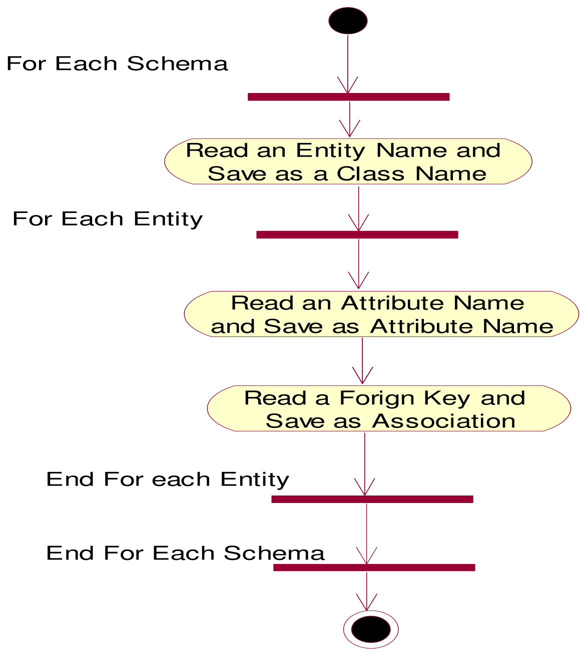
For Each Schema
Read an Entity Name and Save as a Class Name
For Each Entity
Read an Attribute Name and Save as Attribute Name
Read a Forign Key and Save as Association
End For each Entity
End For Each Schema

Figure 7. Translation Process

Tablename→ Classname
Attribute1 Attribute2 . AttributeN
Operational functions → class operation (Insert, Update, Delete and Search)

Figure 7. Class Diagram

3. CONCLUSIONS AND FUTURE WORK

This paper discussed a new reverse engineering approach to convert a form fill format document in a word format into a set of tables. These tables are also converted into UML notation as a class diagram. The proposed approach shows a significant improvement in the reverse engineering techniques and methods, and it is suitable to be used as an automatic tool to generate class diagram from a form fill format document.

The future work will focus on building a CASE tool to automate the proposed approach and to verify all processes using Petri Net formal method.

REFERENCES

[1] S. Ian, Software Engineering, 8th edition. Addison Wesley, New York, NY, USA, 2007.

[2] L. Chih-wei, C. William, C. Chih-hung, C. Yeh-ching, L. xiaodong, Y. hongji, Reverse Engineering, Department of Computer science, De Montfort University, Leicester, England.

[3] R. Agarwal, and A. Sinha, "Object Oriented Modeling wuith UML: A Study of developers' Perceptions", Communication of the ACM Vol.46, No.9, 2003, pp.87-294.

[4] D. Te'eni, R. Gelbard, M. Sade, Increasing the Benefit of Analysis: The Case of systems Shat support Communication, in Proceedings of the 11th International Conference of the Associations Information and Management (AIM'06), June 8-9, 006, pp. 13-27.

[5] M. Mohammad, A. Rafa, A. Belkacem, From Graphical User Interface To Domain Class Diagram: A Reverse Engineering Approach. Journal of Theoretical and Applied Information Technology, 2011.

[6] B. Kumar, Optical Pattern Recognition, Prentice hall, USA, 2003.

[7] R. Mills, K. Hamilton, Learnning UML 2.0, 1st edition, O'Rielly Media, USA, 2006. [2].H. Tran, U. Zdun, and S. Dustdar, View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs, in Proceedings of the International Conference on Software Reuse (ICSR'08), Beijing, China, May 25-29, 2008, pp.233-244

[8] Ranorex, Web Testing, online: http://www.ranorex.com/support/user-guide-20/web-testing.html, visited on April 22, 2009.

[9] Bright-Hub, Sniffing Data with Ettercap for Linux and Windows, online:http://www.brighthub.com/computing/smb-security/articles/35545.aspx, visited on April 22, 2009.

[10] QFS, Facts & Features, online: http://www.qfs.de/en/qftest/, visited on April 23, 2009.

[11] J. Pu, H. Yang, B. Xu, L. Xu, and W. C. Chu,Combining MDE and UML to Reverse Engineer Web Based Legacy Systems, in Proceedings of the 32nd Annual IEEE International Computer on Software and Applications (COMPSAC'08), Turku, Finland, July 28 - August 1, 2008, pp. 718-725.

[12] B. Kumar, Optical Pattern Recognition, Prentice-Hall, USA, 2003.

[13] H. El Bouhissi, M. Malki, and D. Bouchiha, A Reverse Engineering Approach for the Web Service Modeling Ontology Specifications, in Proceedings of the 2nd International Conference on Sensor Technologies and Applications (SENSORCOMM'08), Cap Esterel, France, August 25-31, 2008, pp. 819-823.

[14] M. H. Alalfi, J. R. Cordy, and T. R. Dean, Automated Reverse Engineering of UML Sequence Diagrams for Dynamic Web Applications, in Proceedings of the International Conference on Software Testing, Verifica tion and Validation (ICSTW'09), Denver, CO, USA, April 1-4, 2009, pp. 287-294.

[15] H. Tran, U. Zdun, and S. Dustdar, View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs, in Proceedings of the International Conference on Software Reuse (ICSR'08), Beijing, China, May 25-29, 2008, pp.233-44

[16] S. Demeyer, S. Ducasse, and M. Lanza, A Hybrid Reverse Engineering Approach Combining Metrics and Program Visualization, in Proceedings of the 6th Working Conference on Reverse Engineering (WCRE'99), Atlanta, GA , USA, October 6-8, 1999, pp. 175-186.

[17] A. Alnusair, and T. Zhao, Towards a Model Driven Approach for Reverse Engineering Design Patterns, the 2nd Workshop on Transforming and Weaving Ontologies and MDE (TWOMDE'09), Denver, Colorado, USA, October 4-9, 2009.

[18] A. Memon, I. Banerjee, and A. Nagarajan, GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing, in Proceedings of the 10th Working Conference on Reverse Engineering (WCRE'03), Victoria, BC, Canada, November 13-16, 2003, pp. 260-269.

[19] S. Weijun, L. Shixian, and L. Xianming, An Approach for Reverse Engineering of Web Applications, in Proceedings of the International Symposium on Information Science and Engineering (ISISE'08), Shanghai, China, December 20-22, 2008, pp. 98-102.

[20] G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, and U. De Carlini, WARE: a Tool for the Reverse Engineering of Web Aapplications, in Proceedings of the 1st International Conference on Web Information Systems Engineering (ICWEISE'00), Hong Kong, China, June 19-21, 2000, pp. 241-250.

[21] J. Pu, H. Yang, B. Xu, L. Xu, and W. C. Chu, Combining MDE and UML to Reverse Engineer Web-Based Legacy Systems, in Proceedings of the 32nd Annual IEEE International Computer on Software and Applications (COMPSAC'08), Turku, Finland, July 28 - August 1, 2008, pp. 718-725.

[22] Chen, Peter, "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems 1 (1): 9–36, 1976. doi:10.1145/320434.320440.

[23] Paul Beynon-Davies "Database Systems" Houndmills, Basingstoke, UK: Palgrave 2004.

[24] Chikofsky, E.J. and Cross II, J.H. 1990. Reverse Engineering and Design Recovery: A Taxonomy. IEEESoftware. 7(1): 13-17.

[25] Institute of Electrical and Electronics Engineers. Standard for Software Maintenance. New York, IEEE Std. 1219-1998. 1998.

[26] Chu, W.C., Lu, C.W., Chang, C.H., and Chung, Y.C. 2001. Pattern-Based Software Re-Engineering. In Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing.

[27] Pankratius, V. and Stucky, W. 2005. Aspect-oriented reengineering of e-learning courseware. Emerald. 12(5): 457-470.