Distributed Graphical User Interfaces to Class Diagram: Reverse Engineering Approach using Pattern Recognition

Akram Abdel Qader[1], Khaled Musa[2]

[1][2]Faculty of Science and Information Technology,
Al-Zaytoonah University of Jordan
[1]Akrama@zuj.edu.jo , [2]Dr.khalid@zuj.edu.jo

ABSTRACT

The graphical user interfaces of software programs are used by researchers in the soft-ware engineering field to measure functionality, usability, durability, accessibility, and performance. This paper describes a reverse engineering approach to transform the cap-tured images of the distributed GUIs into class diagram. The processed distributed GUIs come from different and separate client computers. From the distributed GUIs, the inter-faces are captured as images, attributes and functions are extracted and processed through pattern recognitions mechanism to be stored into several temporary tables corresponding to each client's graphical user interface. These tables will be analyzed and processed into one integrated normalized table eliminating any attribute redundancies. Further, the normalized the one integrated table is to create a class diagram.

KEYWORDS

Reverse Engineering, Graphical User Interfaces (GUIs), Pattern Recognition, Optical Character Recognition (OCR), Unified Modeling Language (UML), Class Diagram.

1. INTRODUCTION

Software re-engineering is the examination analysis, and alteration of an existing software system to recreate it in a new form. Software re-engineering is reorganising 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].

The reverse engineering technique is widely used to reconstruct or recover design systems [13]. The reverse engineering is reconstruction or decomposing existing code, analyze 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.

In this paper, the reverse engineering approach is used to construct the UML class diagram from the distributed Graphical User Interfaces (GUIs) [11]. 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 uses 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.

The process of transformation a GUI into class diagram was proposed using petri nets models [5]. In this paper a reverse engineering approach is used to transform distributed GUIs into class diagram through the use of OCR, temporary tables, and the normalization.

This paper is organized into four sections: section 2 discusses the related concepts used in the process, section 3 describes the proposed approach, section 4 is the conclusion, and the section 5 discusses suggested potential future work.

2. RELATED CONCEPTS

This section introduces the related concepts used in this paper that contributes to the addressed reverse engineering approach. The concepts used are the Optical Character Recognition (OCR) and Class diagram.

2.1 Pattern Recognition

Pattern Recognition is the classification, which attempts to assign each input value to one of a given set of classes such as determining whether a given email is "spam" or "non-spam". Pattern recognition algorithms generally aim to provide a reasonable solution for all possible GUI inputs, and a "fuzzy" matching of inputs is done. This is compared using pattern matching algorithms, which look for the matches of the inputs with the pre-existing patterns. The results of pattern recognition comparison is very accurate due to the set of algorithms that matches the scanned inputs to the existing set of characteristics to determine what type of object it is [6].

2.2 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is an external device that uses software to analyze and electronically translate scanned images to recognize scanned images of handwritten, typewritten or printed text into machine-encoded text [6].

The new OCR engines add the multiple algorithms of neural network technology to analyze the stroke edge, the line of discontinuity between the text characters, and the background elements. Allowing for irregularities of printed ink on paper, to be matched with a known characters and makes a best guess to which characters they belong [12].

The OCR approach used is to capture the attributes that are located within the GUI labels, textboxes, and its functions such as buttons.

2.3 Class Diagrams

The class diagram used in the object oriented software design is a static structure diagram and is a type of the Unified Modeling Language (UML). The class diagram notations describe the structure of a system by showing the system's classes, and their attributes, operations or methods, and the relationships among the classes [7].

The UML design class diagram illustrates software class definitions with simple attributes and methods [14]. Class diagrams model class structure and contents using design elements such as classes, attributes and functions or operations. The class diagram in Figure 1 illustrates the common components of class name, class attributes, and class operations.

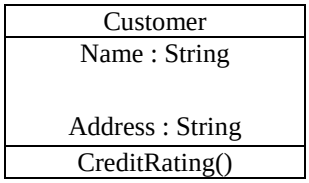


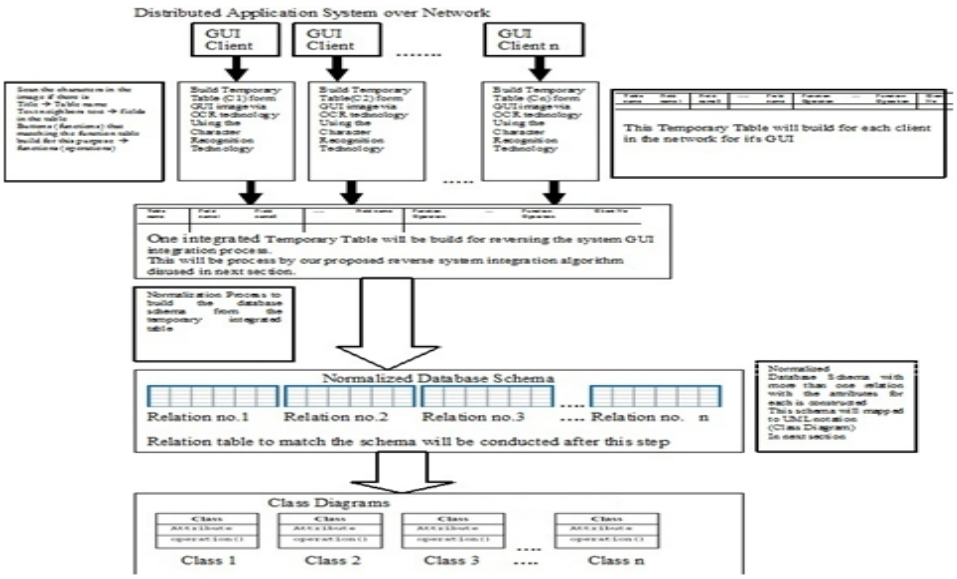
Figure 1. Class Diagram

The class diagram demonstrates the characteristics of an interface and its class name such as Customer, the attributes used within the interface such as Name and Address, along with the operation constituted in the interface such as checking the customer CreditRating. The class diagram of a software application is comprised of classes and a diagram depicting the relationship between each of these classes.

3. PROPOSED APPROACH

The reverse engineering approach is discussed in this paper is capturing and collecting from each client graphical interface over a network. These collected GUIs are to be transformed into to a class diagram following the below process, figure 2, and will be discussed thoroughly in later sections.

Figure 2. Reverse Engineering Process GUIs to Class Diagram



3.1 Capturing Method

Using pattern recognition methodology is to be used in capturing the GUI of each client on the network on the distributed systems. The pattern recognition capturing mechanism addressed in figure 3, is to distinguish table names, fields, and functions for each client graphical user interface.

Capturing Methodology

For each client in the network

Scan the image form and contribute the following

For each form title → define table name
For each label then space then text → define attribute name (field1)
For each label then space → define table name or function name (after scanning in key functions table)
For each labels or buttons → define function name after scanning in key functions table
For each Tab scan table names if not found → add table name
For each form build new record in **captable** (capturing table) (row data)

ForEnd

Figure 3. Capturing Methodology

Capturing elements from all graphical user interfaces is done by looking for any related items on each client graphical user interface as in figure 4, samples of GUIs.

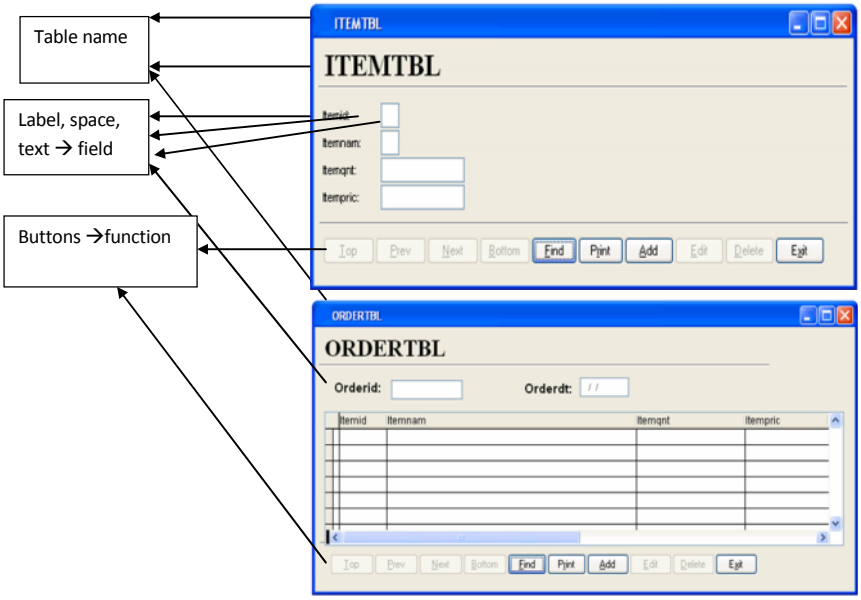


Figure 4. Samples of GUIs

Hence, capturing is recognized. Captured images to extract captured table names, fields, and functions using capturing analysis methodology applying a set of rules to all interfaces are done. The captured images will follow a set of rules regulated in a predefined operation table (Oprtable) to be able to match the different operations or functions to specific operation.

The Oprtable consists of predefined data for all functions and attributes in the graphical user interfaces, figure 5, where several operations will be justified by a common operation or function such as (Save, Add, Submit, New, Insert, Create) which will be replaced with (Save) and etc.

Oprtable	
Operation	Operation Name
Save, Add, Submit, New, Insert, Create	Save
Delete, Cancel, Erase	Delete
Update, Modify, Change	Update
Search, Find, Explore, Navigate, Select, Read, Print	Search

Figure 5. Operation Table (Optable)

Comparing functions in all images in the client graphical user interfaces with the attributes in the images through recognition pattern mechanism will create a Captable that includes all functions and attributes captured from all distributed graphical user interfaces. The capturing images are analyzed using the below analysis methodology, figure 6, to create the Captable and normalized table (Normtable).

Figure 5: Predefined Operation table (Oprtable)

Analysis methodology

Scan all record in **captable** and normalize the tables as the following steps

1- Create new table names as **normtable** with the following fields
(table name,field1,field2, field3,….. fieldn, function1, function2,…,functionn, relation, relation table)

2- Start from the first record in **captable** compare with normtable
Scan for table name
if not found add new record
Scan for fields
If not found()
Scan functionkey()
If not functionkey()
Add fields to that table
Else
Add function to that table
Endif
Else
Return table name
Add relation field to that table
Endif
Else
Return table name
Add relation to that table for different table name in (normtable and captable)
Endif

3- If not end of table goto step 3

When analysis is complete, a client temporary capturing table (Captable) is created for each client graphical user interface to store class attributes and their functions, figure 7.

The capturing table (Captable) for each client will contain all elements captured from each client graphical user interface.

Captable

Table name	Attr1	Attr2	Attr3	Attr4	…	Func1	Func2	Func3	…
Itemtbl	Itemid	itemnam	itempric	itemqnt		Add	Edit	Delete	…
Ordertbl	orderid	orderdat	itemid	itemnam	…	Add	Edit	Delete	

Figure 7. Capturing Table (Captable)

3. GENERATE A CLASS DIAGRAM

The created captable that corresponds to the scanned GUIs is to be combined and integrated into one integrated table normalized table to aid the reverse engineering system process.

On the integrated normalized table, normalization process will be set to generate a normalized database (Normtable), figure 8. The normalized database (Normtable) will consist of several relations each with its own attributes. Based on the normalized relations, a UML Class diagram notations are created. From the created Captable, a normalized table will be created (Normtable) to eliminate any redundancies captured in the Captable. While the Captable process flows to normalization, the functions are compared to the operations located in the Oprtable to eliminate any redundancies. The normalized table will be created with all attributes by eliminating redundancies.

The attributes will be matched to specific operation or function located in the portable to create class diagram which contains attributes and their functions or operations. The process flow that will allocate a finalized class diagram is addressed below in figure 6. The process flow starts with opening the Captable, opening the Norrmtable, and using the portable. Following the logic of the process flow will create the class diagram in the allocated system tables that contains system attributes and their functions or operations. Each Class consists of attributes and operations or functions where each corresponds to the previous normalized database relations attributes.

Normtable

Table name	Attr1	Attr2	Attr3	Attr4	…	Func1	Func2	Func3	Relation	RWT
Itemtbl	Itemid	itemnam	itempric	itemqnt		Add	Edit	Delete	….	-
Ordertbl	Ordered	orderdat	itemid	itemnam	…	Add	Edit	Delete	1toM	Itemtbl

Figure 8. Normalized Table (Normtable)

4. CONCLUSION

The proposed graphical user interfaces of software programs are used by researchers in the software engineering field to measure functionality, usability, durability, accessibility, and performance. This paper proposed a reverse engineering approach to transform the captured images of the distributed clients GUIs into class diagram. From the distributed GUIs, the interfaces are captured as an image, and attributes are extracted and processed through pattern recognitions to be stored into several temporary tables called “Captable” corresponding to each client graphical user interface. These tables will be analyzed and processed into one integrated normalized table called “Normtable” eliminating any attribute redundancies. Further, the normalized integrated table will assist in creation of class diagram.

5. FUTURE WORK

A future work can be done by rebuilding a successful relational database from normalized table and class diagrams to create an application forms in any programming language that are normalized and verified from the proposed process.

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.