ISSN:0975-9646

Pooja Khanna / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2134-2138

Tool for Extraction of Functional Requirements from Legacy Code

Pooja Khanna

ASET, Amity Universiy
Lucknow, India

***Abstract*-A business rule specifies or restraints one aspect of business that is intended to verify business structure or determine the behavior of our business. Business rules often focus on access control issues. There author has proposed a re-engineering method that is implemented to extract a business rule from legacy source code. With time, generally the valuable documents of system are lost and in absence of documentation it becomes difficult to maintain such legacy codes. A case study is included to check the accuracy of the proposed method. The system is checked against the extracted rule with the functional requirement presented in the SRS of the case study.**

Keywords— **Re-engineering, functional requirement, business rule.**

1 INTRODUCTION

Due to rapid advancements in software industry, structure of a software system progressively degrades. Thus maintenance of large and a critical legacy system has become hard. A legacy system is one which is extremely valuable to an organization, performing key strategic functions. But maintaining such systems as to incorporate new functionalities or due to organizational policy changes become hard in the absence of proper documentation, qualified staff and other resources.

The cost of re-engineering a system is generally less than developing a new system. Sometimes what is required is, add some functionalities, change some policies, change the structure of system without changing functionalities, changing the architecture of the system to add some non functional requirements and for making these changes, it is worthless to develop a new system. But still it is a matter of debate that whether to go for forward engineering or for re-engineering.

2 OVERVIEW

The focus of this work is intended to develop a Re-engineering method to automate the extraction of business rules from Source code. Extracted business rules can be classified as structural, behavioral and constraint rules. In this, a program has been be taken as input and then candidate variables are identified.

The available tools left the task of identification of variables on the maintainer. And good approximation of variable is necessary of extraction of required business rule. The identified variable will be used for program slicing. The output of program slicing will be sliced segments which will contain the business rule. These sliced segments will be than given to presentation tool to present the rule in different views so that different stakeholder can easily understand the rules. To check the accuracy of the methodology authors included a case study of computer science department. The case study includes SRS and code of the system. The accuracy of the system depends on the data (approximation of variables) we choose for extracting business rules.

3 OBJECTIVE

In recent years maintenance has become major activity in software industry and as software system age, their original motivation gets lost. In the absence of requirement documents, it is very difficult to maintain a software system. The scenario gets worst in case of a business organization where business policies and decisions (Business Rules) are implemented through an information system. With the growth of the organization, policies and decisions also get changed and to incorporate those changes in the system we first require tracing out the point in source code where those rule (Business Rules) have been implemented.

Fig 1 shows that an organization provides documents that contain business rules for development of a system. With the time, various changes have been done on the system to maintain the system. This results in disparity among the document and the software, as proper documentation was not done along with the maintenance work.

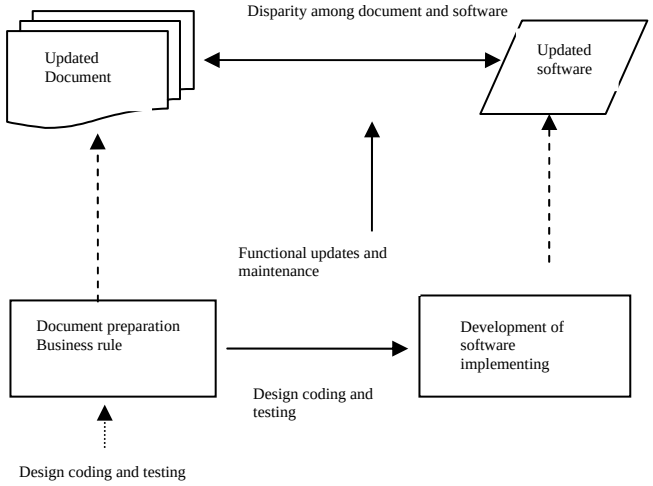

Fig 1 Re engineering Need

Reading a code is much difficult task than writing a code (program). As a result, extracting the information from a system, it may be structure, architecture, design, requirements of system, remains a challenging job. The

task of a software re-engineer is to bridge the gap between operational level at which a program is and the conceptual level, at which a business system works.

Re-engineering helps in better understanding of system even if documentation is not available. Re-engineering a system is generally less expensive than developing a new system . A re-engineering method can help in extraction of business rules buried in source code of legacy system. The extracted business rule must be at high level of abstraction so that every stakeholder of the system could understand the system. The fully automation of the system is a difficult task and system maintainers don't want a black box tool. They need a transparency so that they can well understand the processing of the system.

4 RELATED WORK

A B Earls, S M Embury and N H Turner have proposed a method [A B Earls, S M Embury and N H Turner ,2002] for manually extracting business rules from legacy source code. Goal of this work is to provide guidelines to the code readers of the source code to extract business rule in effective way in terms of time. Since no organization have extra manpower that could be engaged for long in extracting rules (functional requirements).The authors have also discussed the two already available methods for extracting business rule:

Program centric:Their aim is to discover the rules in an information system by analyzing program Code.Data centric:This involves analysis of data set produce by information system to extract business rules. In the paper, they have used method of finding error handling section and translating the conditions that led to error into business rules. Then they proposed two phase approach for the evaluation of method and for that they used operator service information system (OSIS) program. Paper proposed the business rule extraction viability factor (BREVF) sampling method that helps code reader to gain a quick and rough estimate of applying the method to a program. They also pointed the weakness of the method as the inability to extract business rules that are not violated. At the end they have mentioned about the need to develop more tools and methods for extraction of rules.

J Shao and C J Pound [Shao J and Pound C,1999] have proposed "Extracting business rules from information system". Different types of business rules and techniques to unlock these rules are proposed in this work. These are: Structural business rules Constraint business rules Behavioral business rules. According to the study of program and data centric methods , it is revealed that individual method is not sufficient to extract all rules. They proposed a data centered program approach to extract constraint type business rules. The approach presented here is consisted of four components: Parsing tool ,Schema tool, Extraction tool, Presentation tool.The work of this paper is in initial stages and they are working on prototype of the system or tool.Business requirements extraction (BRE) is a re engineering tool for extracting business requirements from COBOL legacy applications developed by software mining [cobolmining.com, BRE_brochure .pdf]. BRE is successful in extracting business rules from COBOL legacy application but the problem is the user should be an expert of the system who has a good knowledge about the domain as for extracting business rule he has to enter a filtering criteria which is comprise of a variable and statement no. BRE basically targets only COBOL application and no further enhancements for any other language is given. Many other tools are also available ww.csse.monash.edu.au,free_tool] like RiGi a software visualisation tool, developed for visualising graphs, with many other features, VCG a ``visualization tool for compiler graphs". And many more but almost every tool helps in better understanding of program through graphs only.

Panagiotis Lions, Philippe aubet, aurent Dumas, Yan Helleboid, Patrica Lejeune [Linos, P.; Aubet, P.; Dumas, L.; Helleboid, Y.; Lejeune, 1993] have proposed "CARE: An environment for understanding and reengineering C programs".Computer Aided RE-engineering (CARE) maintains a repository of control-flow and data flow dependencies of C program. It facilitates incremental program modification. This paper presents the architecture of CARE and empirical evaluation of CARE by modifying small and medium size C program.

Chih-Wei Lu, William C. Chu, Chih-Hung Chang, Yeh-Ching Chung, Xiaodong Liu and Hongji Yang [Chih-Wei Lu, William C. Chu, Chih-Hung Chang, Yeh-Ching Chung, Xiaodong Liu, Hongji Yang,2002] have proposed "Reverse engineering". This paper discussed about legacy system and the need of this system in reengineering. According to this work, reverse engineering can be a time consuming task in lack of proper tools. They have also mentioned that reverse engineering becomes difficult due to several factors, like: Inconsistent programming style.Lack of staff who know legacy system language, Poor documentation, Structure corruption. Paper discussed a reverse engineering process which include Component Analyzing, Design recover, Design reconstructing. Various tools are available for Component Analyzing phase which can easily extract artifacts like structure chart, variables, attributes, functions, program slices , call graph, data flow graph and control dependencies. Extracting original requirements is a tough job and this requires domain knowledge. So, generally all available techniques use structural and knowledge representation to get some high level information. Recovered domain knowledge plays an important role in Quick understanding of the back ground of the application in case of lost or inconsistent document. And can be reused to develop new application. In Design Reconstructing phase System model and design specification will be examined and integrated to reconstruct precise view of the design model, which will help in maintenance and re-development process.

Chia-Chu Chiang and Coskun Bayrak,Member IEEE [Chia-Chu Chiang; Bayrak , 2006] have proposed a paper "Legacy software Modernization". This paper discussed a semi automated approach for extraction of business rule from legacy source code. They suggested a component interconnection model that describes a standard method for communication of different component in different environment. They explained the need of re-engineering by

the problem of Y2K. According to this paper Adaptive maintenance is an activity that is done to adapt requirements due to up coming needs new software, technology, business policies etc. The paper has suggested three types of activities in adaptive maintenance: rewriting, wrapping and re-engineering.

Wrapping can add few functionalities to legacy system using middleware technology though inherent deficiencies of the system remain there. Rewriting legacy system means new functionality can be added but this is an expensive process and organization may not be interested in rewriting whole system. Paper has discussed a re-engineering method for software modernization. The goal of the paper is to build reusable components from legacy source code. Paper has discussed the program slicing technique for extraction of business rule.A list of output data items is selected from entity relationship diagram and data dictionary so that user can select a variable to start program slicing. After finding out the variable v, they searched the last statement of the program to start program slicing. As they use last statement of program as starting point for slicing, in case of large program it become time consuming task. After selection of output variable v and starting point p control flow graph and data dependency graph are used to get the program slice. Since the goal is to search the common code segment. They suggested data constraints and flow constraints to minimize the no slices extracted.

Raghavan Komondoor and Susan Horwitz [Raghavan Komondoor and Susan Horwitz,2001] have proposed a paper' Using slicing to Identify Duplication'. Paper has suggested a slicing technique to identify duplicate code. Understanding and maintenance of a program is often difficult because of duplicate code. Detected duplicate code is extracted into a separate new procedure, and call is made to the new procedure by the instances of the duplicated code. The paper describes the design and initial implementation of a tool. The approach followed use the program dependence graphs (PDGs) and program slicing for finding isomorphic PDG subgraphs that represent clones. The main advantages of the approach are that tool can find clones in which matching statements have been reordered, can find non-contiguous clones and clones that are enlaced with each other. The followed algorithm has three basic steps:

Step 1: Finding pairs of clones.

Step 2: Removal of subsumed clones.

Step 3: Combining pairs of clones into larger groups.

David W. Binkley and Keith Brian Gallagher [David W. Binkley and Keith Brian Gallagher , 1991] have proposed a paper 'Program slicing'. the paper has discussed a program slicing technique for both structured and non structured programs for Computing Slices they discussed some terminology .

A directed graph G is a set of nodes N and a set of edges <ni , nj> where ni and nj belongs to N.For edge (ni ; nj) € E, ni is an successor of nj G contains two special nodes, nstart which has no predecessors, and nend, which has no successors. A control flow graph for program P is a graph in which each node represents the flow of control in P. A edge between two nodes means that the result of the predicate expression at the node pointed by the edge will decide whether to execute the other node or not.Two sets associated with each node: REF (n), the set of variables whose values are used at n, and DEF (n), the set of variables that are defined at n. Program slice is a collection of those statements of P that define behavior of v at s for slicing criteria <v, s>.

5 PROPOSED APPROACH

First of all, we identify the variable that represents valid business rules. All variables of the system or program can be considered but they are large in amount and even all variables are not helpful in identifying business rules. The program is decomposed by analyzing their data and control flow on the basis of identified variables. Retrieved relevant code is then analyzed for extraction of different types of business rules [Hay D and Anderson Healy K,2000] like

Structural

Structural business rules work on one data object and change or derive another data object. So it specifies a relationship between data objects e.g.

An employee is a person

So there is a relationship between data objects that employee is a type of person.

Behavioral

In this on occurrence of an event, state of data objects changes. They are implemented as procedural programs e.g.

On withdrawal of money balance should be debited.

On withdrawal of money event value stored in balance data item will be changed

Constraint

In this valid state of an object is checked. This can be done by searching data base for valid state of data object e.g.

A student should not be allowed for exams with attendance below 70 percent

Status of student attendance will be checked before any processing. For extraction of these rules data centric or program centric approach separately is not sufficient. So I have used data and program understanding technique for extraction of business rules.

Fig 2 shows that extraction of business rule is not a fully automated process. Understanding a legacy system requires automated and manual analysis.

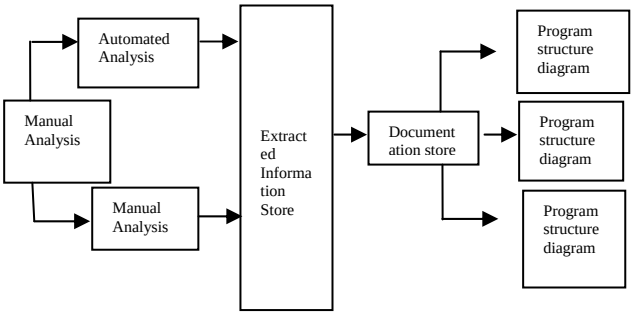

Fig 2 Analysis of legacy system

The system provides the user with flowchart of Individual Programs to better understand the flow of process steps. Extraction of Business Rules using is done using "Variable" filters. each variable is assignment an appropriate descriptions for better understanding and usage. The output as Business Rules are presented in Natural Language (ENGLISH)

6 METHODOLOGY

Business rule represents structure, behavior or constraints of an organization. So to implement a business rule through a program what is required is some input, conditions, assignments, results. Formally a business rule is a code segment in program where a input is checked based on some condition using conditional operators and result either assigned after calculation or in form of Boolean yes or no to output variable e.g.

```
IF HRS-WORKED > 42
    COMPUTE  TOTAL-SALARY = 42 * WAGE-RATE
      + ( HRS-WORKED - 42 ) * 2 * WAGE-RATE
ELSE
    COMPUTE TOTAL-SALARY = HRS-WORKED * WAGE-RATE.
```

In above code salary of an employee is calculated and it represents a structural business rule as if Hour worked is more than 42 hrs. Total salary will be 42 times wage rate plus extra amount of extra hrs is calculated by double of wage rate. To extract such business or any business rule we first have to identify the candidate variables, after that using program slicing we extract the code segment where possible business rule is implemented and then those code is presented in a form to maintainer.

So to extract a business rule, we first have to identify the data (out put variable) that implements the business rule. Now next step is to find the reference (Assignment reference) of the data in which it is used in the program.

There are various references of data in a program.

Usage reference
Conditional reference
Assignment reference

Assignment references are the place where data are created or altered, so they are the key place for extracting the business rules.

New_price = old_price + (old_price x sale_tax);

So here a business rule is implemented that new price will be old price of the commodity plus sale tax over the price of the commodity. But collections of these assignment statements is not a simple task as these assignment statements are scattered along the program. E.g.

```
1.  else if(a== la && f < lf) {
2.  if(ff[f] == ' ' )
3.  ans[lans] = '_';
4.  else
5.  ans[lans] = 'x';
6.  convert(a, f+1, abb, ff, la, lf, ans, lans+1 );
7.  return;
8.  }
9.  else if (a < la && f == lf )
10. return;
11. if(abb[a] == ff[f])
12. {
13. ans[lans]='M';
14. convert(a+1,  f+1, abb, ff, la, lf, ans, lans+1);
15. ans[lans]='x';
16. convert(a,  f+1, abb, ff, la, lf, ans, lans+1);
17. return;
18. }
19. if(ff[f] == ' ' )
20. ans[lans] = '_';
21. else
22. ans[lans] = 'x';
23. convert(a,f+1,abb,ff,la,lf,ans,lans+1);
24. }
25. else if(a== la && f < lf) {
26. if(ff[f] == ' ' )
27. ans[lans] = '_';
28. else
29. ans[lans] = 'x';
30. convert(a, f+1, abb, ff, la, lf, ans, lans+1 );
31. return;
```

Fig 3. Example program slicing

In above example if we concentrate on ans [] assignment reference to this variable is made at various lines like

```
 3.  ans[lans] = '_';
ans[lans] = 'x';
13   ans[lans]='M';
15   ans[lans]='x';
20   ans[lans] = '_';
22   ans[lans] = 'x';
```

So method used to extract business rule implemented with ans [ ] is we will find out the point where these assignment references are triggered. We have maintained a tree structure were we maintain a relation of assignments and triggering conditions. The assignment statements are captured using static forward program slicing. E.g

```
11. if(abb[a] == ff[f])
12.{
13  ans[lans]='M';
15 ans[lans]='x';
18 }
if(ff[f] == ' ' )
ans[lans] = '_';
else
ans[lans] = 'x';
```

These are the slices we get now the main task now is to represent these extracted program segments in a form that may be easily understood by a reader.

7 RESULT ANALYSIS

For analysis of the of the tool we have included case studies with its functional requirements. The accuracy of the system is checked by comparing the final output of the system with the functional requirement present in the SRS of the case studies after considering code for each case study.

Case Study 1

There are a set of classrooms in the computer science department. The department offers courses every semester, which are chosen from the list of department courses. An enrollment No is must for a course and could be for graduate students or undergraduate students. For each course, the instructor gives some time preferences for lectures.

Case Study 2

A railway reservation system from open source is considered as another case study. a reserve module with Few functional requirements is considered here.

The system is run against various modules of two different case studies .Table 2 list the results we got. For module sched_pg_pref( ) of CASE STUDY 1 we have checked the system for 3 different variables against Pref_lst no business rule implemented. But for TimeTable[][] we get one rule in both modules sched_pg_pref( ) and sched_ug_pref( ).Similarly in reserve() module from cas study 2 we run the tool for two variabes. and got results as 2 FR by each variables.

CS#	MDL/FUN	VAB_SET	TLE	BRC
1	sched_pg_pref( )	Pref_lst	3	0
1	sched_pg_pref( )	TimeTable[][]	1	1
1	sched_ug_pref( )	TimeTable[][]	2	1
1	sched_ug_pref( )	ConflLst	5	2
1	form_pref_list()	Token	0	0
1	form_pref_list()	New_pref	3	1
2	Reserve()	Passenger.to	7	2
2	Reserve()	Passenger.sex	4	2

Table 1 No of Business rules extracted successfully after each run.

Where:

CS# = CASE STUDY NO
MDL/FUN = MODULE/FUNCTION
VAB_SET = VARIABLE SELECTED
TLE =TOTAL LINE EXTRACTED
BR = BUSINESS RULE CONTAINED

8 CONCLUSION

The automation of re-engineering process is not an easy task and maintainers are not in favor of a tool which directly re engineers a system. As sometimes they are interested in extracting few information from the system in the absence of documentation. In this work we have studied the various available methods for extracting business rules from a legacy system. We have proposed a method that will help a maintainer in better understanding of the system. Though the system is able to traces the segments where businesses rules are buried but few statements are also extracted which are not of any practical use. Future enhancements include upgrading the system to control more complex programs as we checked the methodology for simple line code that does not includes loops and nested loops and more complex structures.

ACKNOWLEDGEMENT

The authors are very thankful to Mr. Aseem Chauhan, Chairman, Amity University, Lucknow, Maj. Gen. K.K. Ohri, AVSM (Retd.), Director General, Amity University, Lucknow, India, for providing excellent computation facilities in the University campus. Authors also pay their regards to Prof. S.T.H. Abidi, Director and Brig. U.K. Chopra, Deputy Director, Amity School of Engineering and Technology, Amity University, Lucknow for giving their moral support and help to carry out this research work.

REFERENCES

[1] Ian Sommerville 'Software Engineering' , 6th edition, 2000 Pearson Education.
[2] BYUNG-KYOO KANG and JAMES M. BIEMAN 'A Quantitative Framework for software Restructuring' ,journal of software maintenance and research volume 11 issue, page 245-284, 18 Aug 1999
[3] Hay D and Anderson Healy K: 'Defining business rules — what are they really?', White paper, The Business Rules Group, Version 1.3 (2000)
[4] A B Earls, S M Embury and N H Turner:' A method for the manual extraction of business rules from legacy source code' BT Technology journal Volume 20, Issue 4 (October 2002) Pages: 127 - 145 Year of Publication: 2002.
[5] Shao J and Pound C: 'Extracting business rules from legacy information systems', BT Technol J, 17, No 4, pp 179—186 (October 1999).
[6] Linos, P.; Aubet, P.; Dumas, L.; Helleboid, Y.; Lejeune, "CARE An environment for understyanding and re-engineering C program" Software Maintenance ,1993. CSM-93, Proceedings., Conference on Volume , Issue , 27- 30 Sep 1993 Page(s):130 – 139
[7] Harry M. Sneed & Katalin Erdos: 'Extracting business rules from source code', in Cimitile A and Mullar H (Eds):'proceedings of 4th Workshop on program comprehension', Berlin, Germany, EEE Computer society Press,pp240-247(March 1996).
[8] Chia-Chu Chiang; Bayrak, C. 'Legacy software Modernization' Systems, Man and Cybernetics, 2006. SMC apos;06. IEEE International Conference on Volume 2, Issue , 8-11 Oct. 2006 Page(s):1304 – 1309.
[9] Keith Brian Gallagher Keith Brian Gallagher:' Using Program Slicing in Software Maintenance IEEE Transactions on Software Engineering Volume 17, Issue 8 (August 1991) Pages: 751 - 761
[10] http://www.waset.org/pwaset/v16/v16-20.pdf. Raghavan
[11] Komondoor and Susan Horwitz ,Using Slicing to Identify Duplication in Source Code'Lecture Notes In Computer Science; Vol. 2126 Proceedings of the 8th International Symposium on Static Analysis Pages: 40 - 56 Year of Publication: 2001.
[12] Weiser M: 'program slicing', IEEE transactions on software engineering,10,no4,pp 352-357 (1984).
[13] X. P. Chen, W. T. Tsai, J. Joiner, H. Gandamaneni and J .Sun, "Automatic Variable Classification for COBOL Programs", Proc. of IEEE COMPSAC, 1994, pp. 432-437.
[14] Chih-Wei Lu, William C. Chu, Chih-Hung Chang, Yeh-Ching Chung,Xiaodong Liu, Hongji Yang: 'Reverse engineering', Handbook of Software Engineering and KnowledgeEngineering,2002
[15] http://www.agilemodeling.com/artifacts/useCaseDiagram.htm
[16] www.sei.cmu.edu/reengineering/lsysree.pdf
[17] .http://c2.com/cgi/wiki?WhatIsRefactoring
[18] www.cse.iitk.ac.in/JaloteSEbook/CaseStudies/CaseStudy1/SRS.pdf.
[19] http://en.wikipedia.org/wiki/Software_architecture.
[20] www.cobolmining.com/download/SM-BRE-Brochure.pdf.
[21] http://www.csse.monash.edu.au/~ipeake/reeng/free-swre-tools.html