International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 10 No. 2 Feb. 2015, pp. 649-655

http://www.ijias.issr-journals.org/

Recovery of Metrics by using Reverse Engineering

Jalil Abbas[1], Rabia Mehdi[2], Sana-ul-Haq[3], and M. Mutahhar Saeed[4]

[1]Department of Computer Science & IT,
Govt. College University Faisalabad (Layyah Campus),
Layyah,Punjab, Pakistan

[2]Department of Mathematics,
Gomal University Dera Ismail Khan,
D.I.Khan, KPK, Pakistan

[3]Department of IECS,
UST Bannu University,
Bannu, KPK, Pakistan

[4]Department of Computer Science,
Qurtuba University, Dera Ismail Khan,
D.I.Khan, KPK, Pakistan

ABSTRACT: Reverse Engineering is the process which recovers the design artifacts of a software system by using its Size of source code, Available source code type, Abstraction level, Documentation type support. This research focus on the different case studies of recovery of Metrics and is elaborated by using the method of Reverse Engineering, to measure the complexity of the recovery of artifacts for the maintenance task. During this process of analyzing a subject system and in this way representation of a system is created at a higher level of abstraction. It represents an overview of the yield of reverse engineering and reviews cardinal achievement and areas of application, shedding light on key open research issues of the future.

KEYWORDS: Design artifacts, Source code, Abstraction level, Recovery, Documentation

1 BACKGROUND

Software artifact recovery is the ability that describes and follows the life of an artifact requirements, code, tests models reports and plans developed during the software lifecycle in both forward and back word. Artifact recovery can provide insight into system development and evolution in order to assist in both top bottom and bottom up program comprehension, impact analysis, reuse of existing software, which gives support in understanding the relationships existing within and across software requirements design and implementation [1]. Inadequate Metrics recovery is one of the main factors that constitutes to project overruns and failure [2]. Although several research and commercial tools are available that support Metrics recovery. The need to provide the software engineer with methods and tools supporting Metrics recovery has been widely recognized [3]. The rationale behind them is the fact that most of the software documentation is text based or contains textual descriptions and that programmer's use meaningful domain term to define source code identifiers. This research focus on the different Case studies of Anti viruses; Simple Machine Protect Antivirus Case Study, AviGen Antivirus Case Study, CS Antivirus Case Study and it generates valuable results.

Corresponding Author: Jalil Abbas

Reverse Engineering (RE) is the process of discovering the technological principles of a device, object or system through analysis of its structure, function and operation. It often involves taking something (e.g. a mechanical device, electronic component, or software program) apart and analyzing its workings in detail, used in maintenance or to try to make a new device or program that does the same thing without copying anything from the original[10]. Metrics are a set of quantifiable parameters which are used to measure the effectiveness of a project or undertaking. Values are obtained for the parameters for multiple instances of the same entity and they are compared and interpreted as to the change in the effectiveness.

2 MATERIAL AND METHODS

Reverse Engineering Abstraction Methodology (REAM) recovers the design artifacts and help to represent the artifacts of varying details at the domain, functional, structural and implementation Abstractions levels. This method consists of 5-models, High level, Functional, Architectural, Source Code and Mapping Models. The High Level Model use Object Oriented Techniques for the Complexities, which defines Classes and Functions through which we derive Metrics and different Line of Codes [9]. The Functional Model directly deals with the functionality of any system. The Architectural Model is developed using the High-Level Model, Functional Model and from the source code. An Architectural Model defines the structural relationship between the components that together creates the system [4].

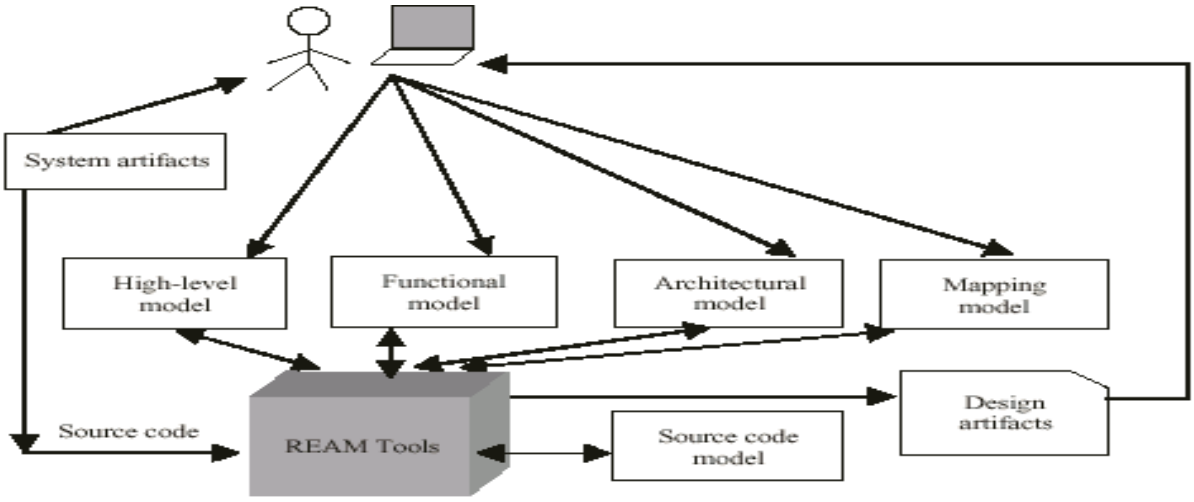


Figure 1: REAM System

2.1 ARTIFACT RECOVERY COMPLEXITY (ARC)

The artifact recovery complexity depend on size of source code, degree of source code type, abstraction level and the degree of available document support to recover the artifacts for the task at hand[7].

- Recover-ness: The task is to identify the components of the system and then relates with each other. Recover-ness is degree of the artifacts that are related to the maintenance task that we have performed in all over the system [5].
- Size of source code: Size of source code from which the artifacts are recovered for maintenance task, the table given below shows the total size of source code and description of source code.

Table 1: Source Code Size

Size category	Description
Small	Few thousand line of code
Medium	More than 1000 line of code
Large	More than 10,000 line of code
Very large	More than 1 Million line of code

- Available source code type: The table given below shows the available source code type and description of source code. It defines available source code types.

Table 2: Source Code Type

Available source code type	Description	Weight
Mix-mode.	The source code is written in multiple languages.	5
Dialects.	The available source code is written in different dialects.	4
Incomplete	The complete source code is not available.	3
Errors.	The code contain errors and cannot be compiled	2
Normal.	The source code exists in a single language	1

- Artifact abstraction level: These artifacts extracted at various levels of implementation, structural, functional and domain level for the recovery [6].

Table 3: Artifact Abstraction Level

Abstraction level	Description
Domain Level	High-Level entities describe the domain level information.
Functional Level	Understand the functionality of the software system that aids the reverse engineering process.
Structural Level	Structural relation between the individual components that together creates the system as a whole. e.g., metrics recovery architecture
Implementation level	Object oriented Classes and Functions.

- Documentation Type Support: Each and every thing documented has been defined separately. Table shows the degree to which the available document supports to recover the artifact for maintenance task.

Table 4: Documentation Type Support

Doc. Type Support	Documentation contain	Weight
No documentation	No document supports the task.	0
Minor	Only system/ component detail.	1
Medium	Some requirements, Design & implementation details exist for support.	2
High	Requirements, Design & implementation details support fully the recovery task.	3

2.2 DESIGN RECOVERY TOOL (DRT)

Design Recovery Tool (DRT) is a special REAM tool used for the source code extraction and high-level artifacts. User also uses this tool for the recovery of design artifacts. The tool allows the user to extract the artifacts from the source code. The main task is to identify different design artifacts. In this perspective there are some of design artifacts that are related to design and user will extract these artifacts from the source code during the Reverse Engineering process. Four levels of abstraction that scope the system artifacts are,

I. Architecture Artifacts

II. Requirement Artifacts

III. Design Artifacts

IV. Implementation Artifacts

The classes are used for different types of functionalities. Main purpose to show classes is that Differentiate between different types of classes and the functions. These classes have been taken separately from different levels of abstraction.

2.3 CASE STUDIES

Followings are the case studies which are conducted in this report with more explained form.

I. Simple Machine Protect (SMP) Antivirus Case Study

II. AviGen Antivirus Case Study

III. CS Antivirus Case Study

2.3.1 CASE STUDY -1

Simple Machine Protect Antivirus Case Study

Simple Machine Protect is portable antivirus software for your Windows Operating System, built to remove certain variant of virus, worm, Trojan and Spyware from your computer. Protect Simple Machine is just that a portable and free anti-virus application that fits on your USB pen and makes sure that the computer you are using is not infected by viruses, worms, Trojan horses or spyware. Created with Visual Basic, the program is incredibly simple to use and can analyze any folder or disk you choose [6]. Nowadays, one of the most important areas to scan is the memory and Simple Machine Protect also caters for this with a memory scanner which works alongside the disk scanner. The program warns you if it finds a virus with a very primitive "beeping" sound and it can sometimes repair them on the spot, depending on whether the database is equipped with a sufficient solution. At all times, the program lets you know at what stage the scan is at with a progress bar.

- Maintenance Task: It identifies the components of the system and relationship among them to understand the system for maintenance.
- Available Documentation Type Support : Available documentation type support (DTS) is available. Minor Documentation is available and its weight is 1.
- Available Source Code Type (SCT) : It is in Mix-Mode. Recovered system is implemented in a multiple languages. E.g. C, C++, Java, HTML and its weight is 5.
- Size of Source Code: Total size of source code (SSC) is 149,380 LOC. Source code category is Large. Its weight is 3.
- Abstraction Level: Different Artifacts of Simple Machine Protect Antivirus recovered at Functional level for the maintenance task i.e.,3.
- High Level Model: In High level Model we discussed Methodologies / Techniques facing complexity problems in the presence of utilizing UML (Unified Modeling Language) and how to solve the modeling language Metrics Recovery.
- Extraction: Class diagram is static and logical model included Functions, Methods, behaviors and attributes.

Table 5: SMP Case Study ARC table

Case Study No:	1
Total Size of Source Code (SSC)	3
Available Source Code Type (SCT)	5
Artifacts Abstraction Level (AL)	3
Available Doc. Type Support(DTS)	1
Artifact Recovery Complexity (ARC)=SSC +SCT+AL-DTS	3+5+3-1 = 10

2.3.2 CASE STUDY -2 AVIGEN ANTIVIRUS CASE STUDY

A comprehensive antivirus tool is created to ensure the security of your PC that will not be infected by all external threats.This software will protect your computer form various types of mal-ware, while still offering you the possibility to add viruses to the list or edit the virus definition

- Maintenance Task: It identifies the components of the system
- Available Documentation Type Support (DTS*) :* Available documentation support is available. Minor documentation type is available and its weight is 1.
- Available Source Code Type: Available source code type (SCT) is Normal implemented in a single language and its weight is 1.
- Size of Source Code: Size of source code is 220,535 and the category is Large .Its weight is 3

- Abstraction Level: Artifacts of AviGen antivirus are recovered at Implementation level for the maintenance task i.e. 1
- High-Level Model: High-level Model is developed using Functional descriptions.
- Extraction: Class diagram being a static model having different functions and methods.

Table 6: AviGen Case Study ARC Table

Case Study No.	2
Total Size of Source Code (SSC)	3
Available Source Code Type (SCT)	1
Artifacts Abstraction Level (AL)	1
Available Doc. type Support (DTS)	1
Artifact Recovery Complexity (ARC)=SSC +SCT+AL-DTS	3+1+1-1 = 4

2.3.3 CASE STUDY -3 CS ANTIVIRUS CASE STUDY

The departmental mail servers use Clam Antivirus to scan all incoming/outgoing email messages. This includes scanning email bodies and attachments against known virus signatures as well as blocking certain MIME types and names (file extensions). The software will also scan compressed attachments.

- Maintenance Task: Identify the components of the system
- Available Documentation Support: Minor documentation type support (DTS) is available and its weight is 1.
- Available Source Code Type: Available source code type (SCT) is in Dialects and written in different languages and its weight is 4.
- Source Code Size: Size of source code (SSC) is 370,425 and the category is large. Its weight is 3.
- Abstraction levels: Artifacts of CS Antivirus are recovered at structural level for the maintenance task. i.e. 2
- High-level Model: The Software engineer developed a high-level model using the experience and knowledge.
- Extraction: Recovered classes are presented in UML.

Table 7: CS Antivirus ARC value Table

Case Study #	3
Total Size of Source Code (SSC)	3
Available Source Code Type (SCT)	4
Artifacts Abstraction Level (AL)	2
Available Doc. type Support (DTS)	1
Artifact Recovery Complexity (ARC)=SSC +SCT+AL-DTS	3+4+2-1=8

3 RESULTS AND DISCUSSION

It is observed that maximum SMP value is 10 which is less than 13 and it shows that highest value of AviGen Antivirus and CS Antivirus are relative 4, 8 which is less than 13.

SSC	SCT	AL	DTS
Large	Mix-Mode	Functional Level	Minor
3	5	3	1

Putting values in Formula;

ARC = SSC + SCT+ AL - DTS

= 3 + 5 + 3 - 1

=11 – 1

ARC =10

Medium value: It is observed that Medium value is 8 which is less than 13.

SSC	SCT	AL	DTS
Large	Dialects	Structural level	Minor
3	4	2	1

Putting values in Formula;

ARC = SSC + SCT+ AL - DTS

= 3 + 4 + 2 - 1

= 9 – 1

ARC = 8

Lowest value: It is observed that the lowest value is 4 which are also less than 13.

SSC	SCT	AL	DTS
Large	Normal	Implementation level	Minor
3	1	1	1

Putting values in Formula:

ARC = SSC + SCT+ AL - DTS

= 3 + 1 + 1 - 1

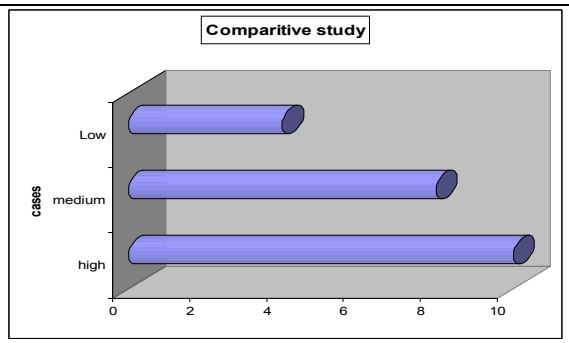
= 5 – 1

ARC = 4

The above said data can be represented in a graphical manner. Representation of this data in a Bar graph by using software SPSS.

Bar Graph

- Y-axis represents highest value 10 which I have calculated during my analysis and this value has been shown in the graph on Y-axis.
- Y-axis represents Medium value 8 which is to be found in my analysis and this value is visible in the graph on Y-axis.
- Y-axis represents lowest value 4 and this value is also visible in the graph on Y-axis.



4 CONCLUSION

This research describes and highlights a case study of Recovery of Metrics, which uses the methodology for reverse engineering that recovers the design artifacts of a software system. The critical analysis of ARC tool represents that each and every value of the case study is within the range of 0-13 and Simple Machine Protect Antivirus case study has the highest value which is 10, CS- Antivirus has Medium value i.e. 8 and AviGen Antivirus has the lowest value which is 4. This analysis and its conclusion are also represented graphically.

This research further focuses on the advance software re-engineering techniques or agile technology to performance based software quality measures. The reverse engineering method described in the future would be applied to a variety of frameworks in order to truly validate its potential.

ACKNOWLEDGEMENT

Authors are grateful to respectable Prof: Syed Ghulam Qasim Shah who reviewed our research work.

REFERENCES

[1] PALMER, J. D. Traceability In Software Requirements Engineering, Second Edition, R. H.Thayer and M. Dorfman, Eds. IEEE Computer Society Press, Los Alamitos, CA, 412–422, 2000

[2] DOMGES, R. AND POHL, K. Adapting traceability environments to project specific needs. Commun. ACM 41, 12, 55–62, 1998

[3] HOLAGENT CORPORATION, .RDD-100, http://www.holagent.com/products/product1.html, 2006

[4] Nadim Asif, Mark Dixon, Janet Finlay, George Coxhead "Recover the Design Artifacts", Proceedings of the International Conference of Information and Knowledge Engineering (IKE 02), pp 656-662, Las Vegas, June 24-27, 2002.

[5] M. Bechter, M. Blum, H. Dettmering and B.Stützel: "Compatibility models". Proc. 3rd Intl. Workshop on SW Engineering for Automotive Systems, pp.5-12, 2006.

[6] D. Binkley, Source code analysis: A road map. In *ICSE -Future of SE Track* 2007

[7] Asif, N. Design Artifacts Recovery Techniques, International Journal of Software Engineering (JSE). Vol. 1 No 1, 2007.

[8] Asif, N. Developing High Level Models for Artifacts Recovery and Understanding Using the Statistical Information, In proceedings of 8th Islamic Countries Conference on Statistical sciences (ICCS-VII) , Dec 19th –23rd , ISOSS Press, (2005C)

[9] J. Beck, "Interface sticing: a static program analysis tool for software engineering," PhD diss., Dept. Stat. & Computer Sci.,West Virginia Univ., 1993

[10] Asif, N. Reverse Engineering Methodology to Recover the Design Artifacts: A Case Study. In proceedings of International Conference on Software Engineering Research and Practice (SERP03) , 23rd –26th june, Las Vegas, Nevada,pp.932-938,2003