International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Study and Assessment of Reverse Engineering Tool

Prof. Ketki Tiwari

Assistant Professor, Dept. of IT, Acropolis Institute of Tech and Research, MP, India

---

Abstract - *The conception of legacy codes is difficult to understand. Numerous commercial reengineering tools are widely available that have different working styles, and are equipped with their built-in capabilities and shortcomings. This paper presented a brief overview of a Reverse engineering concept by studying various existing reverse engineering tools. Paper consist of a study and assessment of various available tools in this field such as Imagix 4D, Columbus, Rigi, Solidfx and e.t.c..*

Key Words: Reverse Engineering, Imagix 4D, Columbus, Rigi, Bahuaus, Solidfx, etc.

1. INTRODUCTION

Reverse engineering concept arise from the problem of legacy system. Legacy system is an old method, technology, computer system, or application program, or being a previous or out of date computer system, yet still in use. Legacy system migration encircle many research areas. A single migration project could, quite accurately, address the areas of reverse engineering, business re-engineering, translation and schema mapping, data transformation, application development etc.

Reverse engineering is a process of restoring the design, requirement specifications and functions of a product from its code analysis. It forms a program database and generates information from this and also produce the necessary documents for a legacy system. It allows developers and maintainers to do things easily and saves money and time both. The Reverse engineering tools help software engineers to analyzing and understanding a complex software system (fig 1.1).

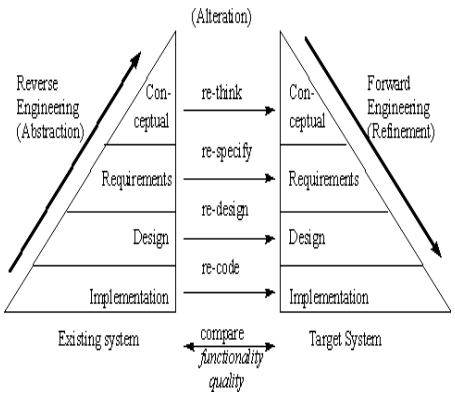

Fig 1.1 Reverse and Forward Engineering Concept

Reverse engineering is influenced by many things like Interfacing, Military or commercial purpose, to enhance documentation short coming, software modernization, product security analysis, bug fixing, creation of unlicensed, unapproved, duplicated, academic or learning purpose, competitive technical intelligence, saving money. For performing reverse engineering on a system many reverse engineering tools are available in the market.

In this paper we study various existing reverse engineering tools. The study of tool will help the developers to improve their mechanism of tools.

2. STUDY OF TOOLS

The reverse engineering community has presented many reverse engineering tools— including Bauhaus, Columbus, SNIFF++, Code Crawler, GUPRO, SolidX, Rigi, Defacto and IMAGIX 4D, Enterprise Architect, CPPX etc. We can divide tools on the basis of language, in which they can execute reverse engineering task. Basically, many of these reverse engineering tools have similar software architecture, including several components with standard functionalities: extractor, analyzer, visualizer, and repository. The extractor, analyzer, and visualizer components emulate the reverse engineering exercise of extraction, analysis, and synthesis, respectively. [1] We study some recognized reverse engineering tools each of which perform distinct category of reverse engineering:

Imagix 4D: Imagix 4D developed to present a solution for C, C++ and Java developers, It helps software that is huge, complex, obscure or old, by automating the examination and browsing of your code. You're adequate to achieve faster and more specific program understanding by more precise representation generated by the tool. This resulting in less software defects while requiring less investment of engineering resources in development process. Tool presents this crucial information on software in a 3D-graphical format which allows the user to quickly focus on particular areas of interest. [2]. Imagix 4D, alike Refine/C, present a superior parser and the generation of projects is quite supported by the tool (e.g. project solution by file, directory, makefile, reparse and incremental parsing, etc.).

The tool has a fine user interface and is very easy to use. It provides the most number of views of all the classified tools and it owns the best set of supporting capabilities (e.g. search engine, unified editor with highlighting and immense browsing capabilities, etc.). The generated views were sometimes too large and complex to be of real use, but because of a lot of efficiency to manipulate them (e.g. filters,

scopes, groups, etc.) they could be good tailored to the task or issue at hand. Queries for the source code that are provided tested very useful (e.g. to comprehend the program faster).

Another feature that is only provided by Imagix 4D is the automatic generation of documentation from the source code. It provides generating HTML documentation which can then be browsed. Document generation is not so good in Imagix 4D. Two major features that are missing are a way to extend the tool and to generate graphical views that are also useful in printed form. [2]. The capability of Imagix 4D to produce decision density and complexity metrics for a large code was found to be useful in getting to learn how much reengineering is required. But with all this advantages Imagix 4D offered limitations in dynamic visualizations, flow chart separation (large code) and parsing loops.[3].

Columbus: Columbus is capable to analyze large C/C++ code and scan to extract facts from them. Columbus is a frame work tool which provide project handling, data extraction, data representation, data storage, modification and visualization. The parsing of the provided source code via the C/C++ extractor plug-in of Columbus, which invokes a separate program called CAN (C++ Analyzer). CAN is a command line application for analyzing C/C++ code. This provides its integration into the client's make files and other configuration files thus accelerate automated execution in parallel with the formal software build process. In addition CAN accept one entire translation unit at a time (a preprocessed source file) and for files that are not preprocessed a preprocessor will be invoked. The clear results of CAN are the internal representation files, which are the binary saves of the internal representations built up by CAN during extraction. These files will be merged together by CANLink, the command line linker for CAN. CAN is able to produce templates at source level, which is capable using a two-pass technique in program analysis. The first pass only identifies the language constructs in connection with the templates (like a "fuzzy" parser) and instantiates them. The second pass then executes the complete analysis of the source code and creates its internal representation.[4].

CAN uphold the precompiled headers technique as well, which is widely used by compiler systems in order to cut down compilation time. This technique is useful especially in case of large projects. The parser is fault-tolerant (it has the capability to parse partial, syntactically incorrect source code), which means that it can carry on with the analysis from the next parsable statement after the failure [5]

Bauhaus: Bauhaus provides a wide range of features such as source code metrics, pointer analysis, side-effect analysis, data flow analysis, program slicing, code duplication methods, static tracing, query methods, source code navigation and visualization, object recovery, re-modularization, architecture recovery techniques, Bauhaus allows you resolve the pointers statically through an automatic analysis. For huge programs, it is so difficult to trace all the function calls and not to get lost. Bauhaus discover all side effects statically through an automatic analysis. Bauhaus analyze Uncover Side Effects, Control and Data Dependencies, Harmless pointers, Detect Code Clones, Measure Code Attributes, Lines-of-Code-per-Function Metric, Cyclomatic Complexity Metric, Nesting Metric, Detect Dead Functions, discover uninitialized variables, Static object tracing, Component mining.[6].

Rigi: Rigi is an interactive, visual and public domain tool Developed at the University of Victoria (Hausi A. Muller) under Rigi Research Project. It is designed in such a way to help you understand and re- document your software code. It uses parsers to read the source code of the subject software and develop a graph of extracted artifacts such as procedures, variables, calls, and data accesses. To control the complexity of the graph, an editor allows you to automatically or manually collapse related artifacts into subsystems.[2].

The initial phase of the reverse engineering process is the extraction of software artifacts, is language-dependent and necessarily involves parsing the source code and storing the artifacts in a repository. Rigi parsing system presently supports the programming languages C, COBOL, and a proprietary IBM system-programming language. It uses GRAS, a database specifically designed to represent graph structures, as a central depository to store the parsed artifacts. The software engineer can then modify the stored artifacts through an interactive graph editor. Primary resource flow graph generated by parsing system is static and uncommon. Rigi has a multiple pre-defined models (e.g., for C and C++) and containing parsers that can extract information from the subject system and store this information in Rigi's exchange format. [7].

Rigi's fact extractors target source code only. The C and Cobol parsers are based on Lex and Yacc. Also exists a C++ parser (vacpp parse) that is created on top of IBM's Visual Age compiler. There also exists fact extractors developed by other groups that support Rigi (e.g., Columbus's C/C++ parser and SHriMP's Java extractor).[7].

Enterprise Architect: EA is a extensible, multi-user, visual tool with a great feature set. Enterprise Architect presents reverse engineering support for a number of well known programming languages. However, the language that you are using is not supported, then you can write your own grammar for it, using the default Grammar Editor. You can then integrate the grammar into an MDG technology to allow both reverse engineering and code synchronization support for your target language. Enterprise Architect uses a deviation of Backus–Naur Form (nBNF) to incorporate processing instructions, the execution of which returns structured information from the parsed results in the form of an Abstract Syntax Tree (AST), which is used to produce a UML representation.

Solidfx: It present an integrated reverse-engineering environment (IRE) for C and C++. SolidFX was particularly designed to support code parsing, fact extraction, metric

computation, and interactive visual analysis of the output in the same way IDEs and design tools provide for the forward engineering pipeline. While the designing of SolidFX, we adapted and extended various existing code analysis and data visualization methods to render them scalable for handling code bases of millions of lines. It also used in various types of analyses of real-world industrial code bases, including maintainability and modularity assessments, detection of coding patterns, and complexity analyses.[8].

SolidFX IRE presents a set of interactive data visualizations, or views. These views are used as input and output to the reverse engineering operations: Users can select elements in the visualization and pass them 150 A. Telea et al. / Electronic Notes in Theoretical Computer Science 233 (2009) 143–159 as input to several operations, such as queries or metric engines, whose results can further serve as inputs for the views. The relative great success of SolidFX in several current projects seems to be due to the huge integration of its functions, provides under a uniform interface, and the possibility to execute complex analysis with minimal (or no) programming.[8].

SolidSX: It is an integrated tool for visual analysis of huge and complex software systems. It unify static code analysis (parsing and metric computation) and multiple linked views such as treemaps, table lenses, and hierarchical edge bundles in a specific environment, thereby reduce the work of software developers concerned in correlating various structure and metric aspects in understanding large software projects. It can be used fully standalone on C, C++, .NET/c#, and Java code bases, but is also incorporate in the Visual Studio IDE. The latter links code editors and visualization in both ways by mouse clicks, and is one of the first (and few) examples of truly unified visual analytics solutions in software development. The another feature of solidsx is a simple Python scripting interface which provides integration in other IDEs or toolchains, such as Eclipse, KDevelop, or Qt Creator. It was used in various industrial reverse-engineering and program comprehension projects.[9].

3. ASSESSMENT CRITERIA

Assessments or a judgment requires criteria or parameters on which we can perform this task. While doing literature review we got some authors describe criteria for the assessment of the reverse engineering tool. We took Berndt Bellay and Harald Galldefined evaluation criteria for our assessment drive which is the best suitable for our goal. It divides the criteria in four main categories: analysis, representation, editing/browsing and general capabilities to evaluate the capabilities of the above tools [Bellay and Gall, 1998].

Analysis: The parser is the main subsystem of each reverse engineering tool. The output of the parsed source code, i.e. the description from which all the views are created, depends on the capabilities of the parser. Portion of the source code that are not parsed or parsed incorrectly will affect all the produced views. [2] So in these criteria we described about the parser ability to parse the source code.

Representation or Import /Export: It is the ability of importing available projects and exporting to several formats of presentation. Representations are categories into textual and graphical reports, and properties of these reports, and are evaluated based on their usability. These are described as follows:

Textual (tabular, formatted, etc.): It presents a list with different textual reports, which are assessed for each tool on their operation and completeness. Graphical (2-, 3-dimensional): It presents a list with different graphical reports, which are assessed for each tool on their usability and completeness.[2].

Editing/browsing: The editing/browsing capabilities are important because the user usually switches the abstraction level from the generated visualization to the actual source code. The text editor/browser should therefore present means to help browsing the source code. [2]

General capabilities: common capabilities span a huge range from supported platforms to online-help. User interface, extensibility, storing capability, multi-user support, among other things [2]

4. CONCLUSION

Reverse engineering is very prominent area of research. Most of the organizations take concern in this field only because it save time, manpower and money. In this paper we evaluated features and capabilities of some distinguished existing reverse engineering tools such as Imagix 4D, Rigi, Bahuas, Columbus, Enterprise Architect, Solidfx and solidsx. These tools are very useful for the purposes of software maintenance, reengineering, re-documentation and code reuse. They present designer, programmer and maintainers many software quality capabilities for their work and for documenting their project. These include analyzing automatically the source code of a software system, representing the structure of this system at higher levels of abstraction such as call graphs, flow charts, control flow and class diagram. They also provide code parser, storing ability, representation features in large lines of code.

REFERENCES

[1] Holger m. Kienle, hausi a. Muller,"The Tools Perspective on Software Reverse Engineering Requirements, Construction, and Evaluation" Advances in Computers, December 2010.

[2] Berndt Bellay, Harald Gall, "An Evaluation of reverse engineering tool capabilities" Journal of Software Maintenance: Research and Practice,1998.

[3] Rashmi Yadav, Ravindra Patel and Abhay Kothari," Critical evaluation of reverse engineering tool Imagix 4D!" SpringerPlus (2016) 5:2111.

[4] Rudolf Ferencl, Arpad Beszedes, Mikko Tarkiainen and Tibor Gyimothy, "Columbus Reverse Engineering Tool

and Schema for C++", International Conference on Software Maintenance, IEEE 2002.

[5] Rudolf Ferencl, Arpad Beszedes, Mikko Tarkiainen and Tibor Gyimothy, "Fact Extraction and Code Auditing with Columbus and SourceAudit", Proceedings of the 20th IEEE International Conference of Software Maintenance (ICSM'04).

[6] Aoun Raza, Gunther Vogel, and Erhard Pl¨odereder, "Bauhaus – a Tool Suite for Program Analysis and Reverse Engineering", International Conference on Reliable Software Technologies Ada-Europe 2006.

[7] Holger M. Kienle, Hausi A. Muller," Rigi—An environment for software reverse engineering, exploration, visualization, and re-documentation" Science of Computer Programming 75 (2010) 247–263 elsevier.

[8] Alexandru Telea, Heorhiy Byelas, "A Framework for Reverse Engineering Large C++ Code Bases", Electronic Notes in Theoretical Computer Science 233 (2009) Elsevier.

[9] D. Reniers, L. Voinea1 and A. Telea," SolidSX: A Visual Analysis Tool for Software Maintenance", Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization (2010).

[10] Rashmi Yadav, Ravindra Patel and Abhay Kothari," Critical evaluation of reverse engineering tool Imagix 4D!" SpringerPlus (2016) 5:2111.

[11] Michele Lanza, "CodeCrawler — Lessons Learned in Building a Software Visualization Tool", Seventh European Conference on Software Maintenance and Reengineering, 2003. Proceedings., IEEE 2003.

[12] M.A.D. Storeyyz K. Wongy P. Fongz D. Hooperz K. Hopkinsz H.A. Muller," On Designing an Experiment to Evaluate a Reverse Engineering Tool", Proceedings of WCRE '96: 4rd Working Conference on Reverse Engineering,IEEE.