Advances in Manufacturing Technology XXXIV
M. Shafik and K. Case (Eds.)

doi:10.3233/ATDE210012

Reverse Engineering Application Instruments and Code Reliability: A Comparative Study of Tools

Manaqib Ahmad Zeeshan[1], Salman Sagheer Waris[2]
Department of Mechanical Engineering,
Capital University of Science and Technology,
Islamabad, Pakistan

Abstract. Based on different scenarios (professional guidelines) practice of Software Reverse Engineering (SRE) is used to analyse the combined instructions system to extract information regarding design and implementation of either part of, or the whole software application. These business rules are implemented in the form of a line code whereas actual source code is hidden and only gets the binary form of the code. Technologies that used for reverse engineering are CVF, V7, CFC, 14D, RTR, B#. These instruments are used for a better understanding of the program algorithm, logic, and program specifics in windows API functions, programming assembler language, network interaction principle. The tools that are discussed will not disturb the code consistency and basic structure of software. Present research shows a comparative analysis of various tools to establish which reverse engineering tool is better based on what characteristics.

Keywords. Programming instrument assessment, Reverse engineering (RE), Code consistency, Comparative analysis of tools

1. Introduction

Due to increasing demand for reverse engineering, there are many tools introduced in the market by different companies. These tools will not disturb the basic structure of code. These tools will only open the code in a mannered way for programmers so they can easily modify the software according to the new structure. Software-based systems now take a central role in various domains. A combination for beneficial capacities into programming adds to the expanding unpredictability of programming-based reverse engineering frameworks, particularly in embedded frameworks.

Main cause of why programming projects stand unsuccessful is the barrier towards achieving their necessities because changes in requirements are hard before managed in a proper manner [1]. Software projects probably take negative effects without an exact way to cope with these requirements, specifically unnecessary progress, and management. However, new techniques to gather, analyses, file and sustain requests are increasing, their software supplies features are static generally printed in natural syntax.

[1] Manaqib Ahmad Zeeshan, Department of Mechanical Engineering, Capital University of Science and Technology, Email: Mazeeshan2006@gmail.com
[2] Salman Sagheer Waris, Department of Mechanical Engineering, Capital Uiversity of Science and Technology, Email: irfanmanarvi@cust.edu.pk

Module's properties remain the same whereas new software functions are being developed and utilized. It has been investigated that Reverse engineering (RE) was first used in equipment study, then its possibility of use in programming schemes was observed [2]. Afterwards, reverse engineering tools were made by the end of 80s, organizations then started to use these reverse engineering tools to speed the process of updating of different software [3]. RE can be stated as the method of investigating a previously executed programming system to reveal its design or extract knowledge from that software [2]. These tools can be used to correct (e.g. fix bugs, inform (e.g. alignment with updated user requirements), upgrade (e.g. add new capabilities), or even completely resequence the previous software [3]. RE delivers a new method for the condition of the properties for a structure [4]. RE instruments are effective, however not broadly castoff due to disintegration of this instrument plus the absence of several important types and skills [5]. This method is forever and workable if the foundation code for a structure last. The code group represents that a originator reads the puzzle or file linked to the realist ic model and create a high-level syntactical program (C, C + +, Java, Perl, Ruby, Python and HTML and so on) [6]. In this comparison, the different instruments based on capacities and properties are shown which takes the functions of the software that is to be tested to make it easy to understand. The instruments discussed are good in keeping the reliability of the project file.

2. Related Work

RE implications have been widely discussed in the previous years. Regarding RE method, many techniques have been introduced and the most related research works are discussed as:

- **Schema analysis** [7] generally intensive on noticing connections in charge domains and names. This method can assist to recognize absent concepts (e.g. foreign keys).
- **Data analysis** [8, 9] utilized substance excavated from a record. Foremost, this may be expended to observe the folder normalization and also to prove imaginary ideas recommended by different methods.
- **Screen analysis** [10] defined in a way that worker boundaries may also be suppliers of suitable information.
- **Static** [11, 12] **and Dynamic** [13, 14] **program analysis** may simply offer data related to ground creating and evocative tags, or to recognizing multipart restraint examining.

Other RE instruments create diagrams using informant cypher. It is impossible to analyse every device [15]. According [16], Raptor can be a software diagram-based graphical programming environment. Visual to flowchart (VFC) [17] code may be implemented to present its diagrams to MS Office applications including Visio. It was stated [18] that it is difficult for beginners to make flow of a program without the use of diagrams or flowcharts even with pseudo code. While flow charts for large and complicated programs are not feasible. Flowchart acts as a vital role in the preparing systems algorithm design. Without going deep into code programmer can easily understand the code with the help of flowchart [19]. It was stated [20], the usage of the instrument which makes flowcharts to implement visual solutions to correct the program

code. Having animation feature like flowchart helps the developer to understand the flow of code statements.

3. Reverse engineering

It has a long-term convention in various regions, starting from conventional engineering to information-based industries. The collective goal of reverse engineering in all of these areas is the abstraction of knowledge from human-made artefacts. The progression of RE normally initiate with lesser levels of data that define the artefact under study. Higher levels of knowledge and understanding are revealed or synthesized from this low-level information. Thus, RE may be shown as growing the conceptual level of the information which is accessible about the artefact under study. In the software domain, reverse engineering deals with artefacts that are created in the practice of constructing a software structure, most notably the method's codebase. It is best understood by looking at its context and role in software construction and assessment. While the classical software development process refines a system by going through the stages of requirements, design and implementation, reverse engineering starts with the implementation's artefacts and produces higher-level abstractions such as those found in the design and requirements stages as displays in below image.

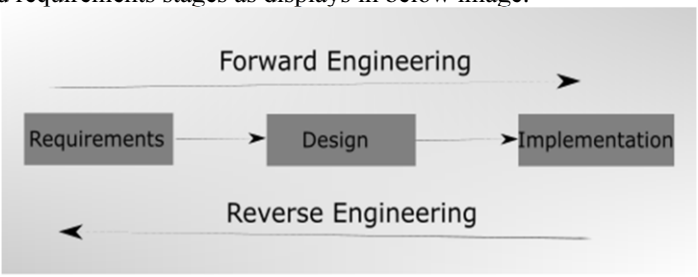

Figure 1: Relations of reverse and forward engineering [21]

3.1. Requirements for Reverse Engineering (RE)

Device arrangement in the RE area has specific characteristics, those can vary domain to domain. It is defined that every domain has its desires for obtainability [21], adjustability, presentation (Performance,) safety [22], or possibility of using (usability)'. Its essential analysis for the reason that it indicates that the specific desires [23] of a definite area can be good assumed by initializing from generic, domain-fair desires [24]. The five major requirements for RE instruments are the following [26]:

- *Scalability:* The instrument needs to be excellently and efficiently deal with a great amount of data. For Example, a display Visualizer should be proficient to render thousands of nodes and curves.
- *interoperability:* A instrument supposed to interact with other instruments. In other means, instruments that interoperate allow them to pass information and control among themselves. This permit to (opportunistically) assemble an

instrument arrangement that supports a specific reverse engineering task or process.

- *customizability:* A instrument has to be customizable. This is essential because Reverse engineering tasks are quite diverse and differ from many factors. As a result, reverse engineers should be continuously adapting their instruments to meet changing prerequisites.
- *usability:* An instrument should be usable. Usability provides more satisfied instrument users, escalates the reputation of the instrument and its developers, and reduces costly redevelopment done by user complaints.
- *adoptability:* A instrument supposed to be adopted by an (industrial) user community. One of the many reasons why instruments are not approved by industry is that many of these instruments do not support the right tasks. Conversely, an accepted instrument proves that it is useful in the eyes of its worker.

3.2. Purpose of Software Reverse Engineering

The program RE is important because there will always be "old" or else called legacy applications [21]. Nowadays, developers succeed in a huge legacy of current software. Old systems were difficult to understand and sustain because of their volume and complexity along with its past development [22]. To tackle the difficulty of the instructions, reverse engineering tools have been used [23]. Actually, legacy code was used to refer to programs written in COBOL, generally for large mainframe systems. However, current software developers principally use Object Oriented languages like C++ and Java, which means that future legacy code is being written today. Some of the purposes of the technology of reverse engineering software as:

- Reverse Designing programming offers a way to recoup lost data by conveying appropriate framework documentation. Recapturing lost data implies commonly the advancement of never open plan reports and recuperating data that has been missed during programming improvement or in any event, during long stretches of upkeep activities. In the vast majority of the cases, the real fashioners are no longer added to the improvement group in the time of a product framework, and visitors may need to do the safeguarding of the framework [21]. In this condition, recuperation of various types of data identified with the product framework becomes fundamental and figuring out strategies offer the methods for recouping lost data and creating different portrayals of a framework, for example, the age of structure outlines, dataflow charts, substance relationship graphs, and so on [22].
- Reverse Creating underpins the extenuation of another equipment/programming system or mix into a CASE situation. The progressions of those frameworks likewise need adjustments of the particular application, Since the improvement of uses is commonly not totally free of the hidden equipment/programming stage [21].
- The strategies for figuring out gave programming reuse through encouraging help for the definition, augmentation and ID of returnable segments inside current frameworks [21].

- Reverse creating gives remedial and versatile assurance by numerous strategies, for example, offering extra documentation and recreating. Figuring out of programming or equipment frameworks should be possible to help undocumented document designs or undocumented equipment peripherals subsequently offering help for interoperability and empowering the product to stumble into a few equipment stages [21].
- instructions Figuring out can be sent for security review, expulsion of duplicate insurance, avoidance of entrance limitations frequently existent in customer gadgets, adjustment of installed frameworks, internal fixes or rehabilitation, empowering of extra highlights on minimal effort equipment or should be possible for the insignificant fulfilment of interest [24]. This reason may seem corresponding to encroachment on proprietary rights; yet for scientists, it is perhaps a form for patent encroachment [25].
- Instructions Figuring out gives the procedures to moral hacking [a word for the training, by them with adequate aptitudes and along the development consent of the framework proprietors, of infringement into PC frameworks to show safety shortcomings. The thing, moral programmers, have a helpful undertone, is related with those utilizing their aptitudes for real reasons, for example PC safety specialists working framework exploration or weakness analysis to more readily protect in contradiction of assaults. This is rather than a dishonest programmer, having a negative meaning, signifies unapproved people who break into PC frameworks for ill-conceived purposes – in this manner being interchangeable with saltines [26].

3.3. Reverse Engineering Process

Once making a RE task, the RE surveys a special method. RE is a method of discovery that takes the scheme's artefacts as input and produces information about these artefacts as output. The definition does neither mandate the process's inputs nor its outputs. Reverse engineering has continually broadened and adapted its inputs as well as its outputs in its quest to provide more relevant information to engineers performing reverse engineering tasks.

Suppose the (basic) C code in image 2. In this situation, the extractor must represent the C functions within the source (i.e., main, f, and g) along with the calls within the functions. While the abstraction of such data may seem marginal at first glimpse, some cases make it problematic or impossible to gain full and accurate information. Calling a function employing a function pointer makes it hard to recognize the call and also doubtful the target of the call. C pre-processor directives may further intensify the call graph abstraction process since macro calls and function calls are lexically unintelligible and since the function names can be accumulated using the pre-processor concatenation operator.

```
/* file main.c */                          /* file utils.c */

void main() { ... f(); ... g(); ... }      void f() { ... g(); ... }
                                           void g() { ... g(); ... }
```

Figure 2: Sample C program (written program during practices) [14]

If one defines the thing assessment driven instrument creating to point out which instrument assessment should be an important portion of an instrument creating exertion. Assessment should be measured not for an addition but also while the complete instrument-creating project. “Sensalire et al” stated an instrument evaluation round that comprises 4 steps [27] (Image 3). This sequence describes that instrument creating is an iterative action that obtains response by directing evaluations. In the reverse engineering environment, there is a growing understanding that instrument estimations are desired to improve the field.

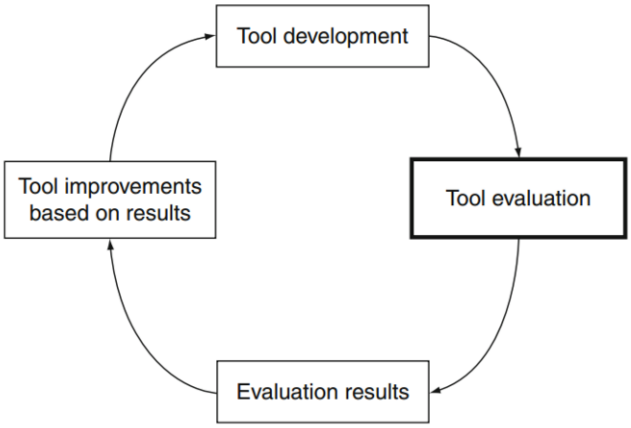

Figure 3: Instrument creating and instrument evaluation sequence [27]

A method to instrument creating that is estimations-driven and concept-grounded has to track a relatable procedure that comprises concepts, investigates, and instrument creating [28]. The procedure involves ‘some iterative stages of project, enlargement, and evaluation’ and has been used on the SHriMP instrument. Image 4 represents an interpretation of the method repetitions with SHriMP as the subject instrument. Assessment of the instrument’s strategy and execution can be proficient by experimental studies constructed by observation of instrument users [29]. This sequence is implanted in a higher repetitive sequence for recovering the original theory.

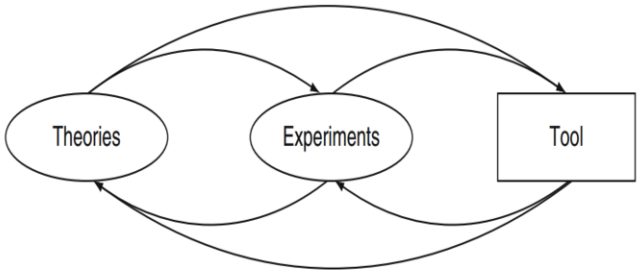

Figure 4: Sequences of the instrument creation, testing, and scheme creation [27]

4. Supporting Instruments

There are 6 RE instruments for the evaluation which create flow-charts considerably, everyone indicates a diverse group of RE instruments.

4.1. Software Code Visual to Flowchart (CVF)

CVF produces software which helps programmers comprehend and file source code. The flowchart vision auto upgrades indicating the database normally as every line passed keen on the CVF script chequer [30].

CVF [17] considered to imagine the practical encryption of some difficulty. This mechanism is finest with implementable code, then it can further spend a non-implementable affirmation. For the best outcomes, think to expel the nonprocedural measurements, as they bring about the messiness in the graph every now and then. You can envision a total instructional code archive, a solitary class, a solitary system or only a section of it. Code to Flowchart endeavors at the announcement level and couldn't care less about the centerpieces of different articulations, for example, the Booleans in an if-explanation. CVF arranges C, C++, VC++, VB, VBA, Qbasic, VBScript, ASP, Visual C#, VB.NET, Visual J# .NET, VC++,.NET, ASP.NET, Java, JSP, JavaScript, Delphi, Pascal, PowerBuilder, PHP, FoxPro.

4.2. Visustin v7

Visustin is a type of software which was used for making flowchart and images out your code and gets it to the stream graphs fundamentally. Visustin peruses the if and else explanations, circles and hops and constructs a graph completely unavoidably. Visustin flow charts ABAP, Action Script, Ada, ASP, numerous low level computing constructs, BASIC, .bat and others [31].

The graphs picture your code and alternatively the remarks also. Use stream charts in your undertaking documents. Spare documents in BMP, GIF, MHT,.EMF and other designs. It demonstrates both code and flowchart.

4.3. Imagix4D

Imagix4D instructs the designers to understand difficult or heritage C, C++ and Java source code [32]. You are able to accelerate your progress, upgrade, reuse, and examination with the use of this tool.

You can likewise for all intents and purposes locate a wide extent of points identified with your programming - control courses of action and information use. These are all centred around Imagix4D's for exact fixed assessment of source code. Learn unused code, oust errors due to broken comprehension, get recently recruited employees on-board quicker, dedicate a period creating and not perusing total code.

You can outwardly investigate an extensive scope of parts of your product control arrangements, information use, and legacy. All dependent on Imagix4D's exact static examination of code. For complex capacities that may comprise of several lines of code,

can help in more rapidly handle the interior rationale of the daily practice. A FlowChart outline presents total inner rationale, of a solitary capacity at once.

4.4. Code to Flowchart Converter

Code to Flowchart Converter can hastily deliver flowchart from the code of software [33]. This apparatus is likewise valuable for program originators and documentation scholars. Software engineers can without much of a stretch access get venture review by utilizing Code to FlowChart Converter. [34] and software engineers can screen their changed code associations consistently. Code to FlowChart Converter promptly shows the source code. It encourages report scholars with a visual flowchart. Those graphs make an archive author exceptionally clear consideration which is critical to their composition. Code to FlowChart Converter upkeep to completely extend the flowcharts, in any event, when there are different columns of code. Code can be shown in a tremendous flowchart. It chains punctuation featuring of code, it implies, when you check any coherent box on the stream graph, the related code will be featured, it encourages developers to discover blunders in an accommodating manner Code to FlowChart Converter supports to appropriate flowchart to Visio, MS Word, XML, SVG, BMP.

4.5. Raptor

The primary objective of Raptor is to refine the critical thinking aptitude of students just as evading punctuation blunders [28]. This device images out how to produce and complete highly contrasting flowcharts by moving entire structures onto the flowchart. This instrument supports the techniques and exhibits alongside recommended a library of inborn strategies to perform IO with text documents and characterize the data sorts of factors. In light of the centre region, the Raptor can change flowchart into code. The program isn't balanced, which could prompt linguistically off base code and not implementable without important manual adjustment in a fringe domain.

Besides, the fledgling developers can face extra issues generating flowcharts of codes. Additionally, it would not be able to keep up the improvement of program testing, troubleshooting just as discovering abilities. Its merits referencing, absence of some noteworthy attributes like code review and syntactic special cases would balance out the code age's depiction of Raptor [35].

4.6. B#

B# is an instrument dependent on Window applications. In B# the code would be produced in flowchart [36]. As articles are included, the software engineers decide pertinent boundaries that are linguistically tried, and any blunders are presented to the developers. This instrument additionally gives software engineers to see the association among code and flowchart which are on each other. Along these lines, synchronizing would be one of the undeniable highlights of B#. At the point when software engineers

pick the module of the flowchart, the related line of code will be featured. Additionally, this instrument gives an imagined program finishing by featuring each and every factor of the flowchart. Flowchart arrangement for this apparatus isn't coordinated with standard stream diagramming rules; therefore, B# isn't in accordance with the stream outlining portrayal regularly used.in reading material. Henceforth, B# won't be a movable program structure technique and toward the beginning, it should to be engaged with additional learning [36].

- Instrument Collection Criteria

Table 1 shows the highlights of the devices. The chose apparatus ought to likewise be either under dynamic current turn of events or be identified with logical distributions of programming support.

CVF: Code Visual to Flowchart
V7: Visustin
CFC: Code to Flowchart
I4D: Imagix 4D
RTR: Raptor
B#

Table 1: Instruments Comparison

Instrument Features	**CVF [17]**	**V7 [31]**	**CFC [34]**	**14D [32]**	**RTR [16,35]**	**B# [36]**
Flowchart automatically	.*Y*.	. *Y*.	. *Y*.	. *Y*	. *Y*	. *Y*.
Structural Rules Enforced	.*N*..	. *N*.	.Y.	. *N*.	. *N*.	. *N*.
Language-Independent	. *N*.	. *N*.	. *Y*.	. *Y*.	. *N*	. *Y*.
Reverse Engineering Concept	. *N*..	. *N*.	. *Y*..	. *N*..	. *Y*.	. *N*.
Highlight capability of Source Code	. *Y*..	. *N*.	. *N*.	. *Y*..	. *N*.	. *Y*.
OS Dependency	. *N*.	. *Y*.	. *Y*	. *Y*.	. *Y*.	. *Y*.
Code Review	. *Y*.	. *Y*.	. *N*.	. *Y*.	. *N*.	. *N*.
Bulk Charting	. *N*.	. *Y*.	. *N*.	. *N*.	. *N*.	. *N*.
Combine Data and Diagram	. *Y*.	. *N*.	. *N*.	*Y*.	. *N*.	. *N*.
Restructure	. *Y*.	. *Y*.	. *N*.	. *Y*.	. *N*.	. *N*.
Multi-page Print	. *Y*.	. *Y*.	. *N*.	. *Y*.	. *N*.	*N*.
Edit Flow Chart	. *Y*.	. *Y*.	. *N*.	. *Y*.	. *Y*.	. *Y*.
Error Feedback	. *N*.	. *Y*..	. *Y*.	. *Y*.	. *Y*.	. *N*.
Static View	. *Y*.	. *Y*.	. *Y*.	. *Y*..	. *Y*.	. *Y*..
Performance	*High*	*High*.	*High*	. *High*	*High*.	. *High*..
Clarity.	*High*	*High*.	*High*	*High*.	*High*	*High*

5. Conclusions

We have gathered some data of some features for each reverse engineering tool that was needed by programmers to update old software into new ones. According to our research the comparison of various instruments that were used for Reverse engineering by using Table 1 (Tools Comparison) on the basis of Flowchart, Structural Rules, Language-Independent, Reverse Engineering Concept, highlight capability of Code, Restructure, Edit Flow Chart, Performance and other features, the best reverse engineering tool is 14D best of all because the purity and the performance of this tool is high than the rest. Moreover, it also provides flowcharts for better understanding of developers. This tool is language independent which means it provides a common interface so there would be no difficulty in understanding and operating it. RE tool 14D shows the editing history to the developer in the form of a flow chart so he can easily see where changes were done.

References

[1] **J. M. E. Garcia**, Requirements Change Management based on Web Usage Mining. PhD thesis, UNIVERSITY OF PORTO, 2016

[2] **E. J. Chikofsky and J. H. Cross**, "Reverse engineering and design recovery: A taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13–17, 1990

[3] ***H. Bruneliere, J. Cabot, G. Dupe, and F. Madiot***, "Modisco: A model-driven reverse engineering framework*,"* Information and Software Technology*,* vol. 56, no. 8, pp. 1012–1032, 2014

[4] **E. J. Chikofsky,** Reverse engineering and design recovery: A taxonomy. IEEE Software pages 13–17, Jan. 1990

[5] **Muller H.A. Wong K., Tilley S.R.,** Reverse Engineering: A road map, proceedings of a conference on the future of software engineering, International conference on software engineering, 2000

[6] **D. Harel,** "Statecharts: A Visual Formalism For Complex Systems," Science of Computer Programming, 2007

[7] **W. J. Premerlani and M. R. Blaha**, "An approach for reverse engineering of relational databases," in *Reverse Engineering, 1993., Proceedings of Working Conference on*, pp. 151–160, IEEE, 1993

[8] **R. H. Chiang, T. M. Barron, and V. C. Storey,** "Reverse engineering of relational databases: Extraction of an EER model from a relational database," *Data & Knowledge Engineering*, vol. 12, no. 2, pp. 107–142, 1994

[9] **N. Pannurat, N. Kerdprasop, and K. Kerdprasop**, "Database reverse engineering based on association rule mining," *arXiv preprint arXiv:1004.3272*, 2010

[10] **R. Ramdoyal, A. Cleve, and J.-L. Hainaut,** "Reverse engineering user interfaces for interactive database conceptual analysis," in *International Conference on Advanced Information Systems Engineering*, pp. 332–347, Springer, 2010

[11] **J.-M. Petit, J. Kouloumdjian, J.-F. Boulicaut, and F. Toumani,** "Using queries to improve database reverse engineering," in *International Conference on Conceptual Modeling*, pp. 369–386, Springer, 1994

[12] **G. A. Di Lucca, A. R. Fasolino, and U. De Carlini**, "Recovering class diagrams from data-intensive legacy systems," in *Software Maintenance, 2000. Proceedings. International Conference on*, pp. 52–63, IEEE, 2000

[13] **A. Cleve, J.-R. Meurisse, and J.-L. Hainaut**, "Database semantics recovery through analysis of dynamic SQL statements," in *Journal on data semantics XV*, pp. 130–157, Springer, 2011

[14] **A. Cleve, N. Nought, and J.-L. Hainaut**, "Dynamic program analysis for database reverse engineering," in *International Summer School on Generative and Transformational Techniques in Software Engineering*, pp. 297–321, Springer, 2011

[15] **Rashmi Yadav, Ravindra Patel, Abhay Kothari**: Reverse Engineering Instrument Based on Unified Mapping Method (RETUM): Class Diagram Visualizations, Journal of Computer and Communications, Oct-2014, 2, 39-49

[16] **Carlisle M, Wilson T, Humphries, J and Hadfield,** M, 2004, RAPTOR: Introducing Programming to Non-Majors with Flowcharts, Journal of Computing Sciences in Colleges, *Consortium for Computing Sciences in Colleges,* University of Central Missouri, USA pp: 52 – 60

[17] FATESOFT (2009), Code Visual to Flowchart, FateSoft, Eden Prairie - NM - USA, [Accessed 29/ Auber, D., **Melancon, G., Munzner, T. And Weiskopf**, D. (2020) SolidSX: A Visual Analysis Instrument for

Software Maintenance. Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization. [05/2020] http://www.fatesoft.com/s2f/

[18] **Westphal B, Harris F and Fadali M**, 2003, Graphical Programming: A Vehicle for Teaching Computer Problem Solving, *33rd ASEE/IEEE Frontiers in Education Conference,* IEEE, Boulder, Colorado, and pp: 19-23

[19] **Denial Hooshyar, Maren T. Alrashdan, Masih Mikhak**: Flowchart-based Programming Environments Aimed at Novices. ISSN: 2232-1942 Vol. 13 No. 1 January – March 2013

[20] **Andrew Scott, Mike Watkins and Duncan McPhee.** , (2010), E-Learning For Novice Programmers, A Dynamic Visualization and Problem Solving Instrument, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=452 9966

[21] **Klosch, R.R.** (1996). Reverse Engineering: Why and how to reverse engineer software, Proceedings of the California Software Symposium (CSS ″96), Los Angeles, California

[22] **Rugaber, S.** (1994). "Program Comprehension for Reverse Engineering," http://www.cc.gatech.edu/ reverse/papers.html, College of Computing, Georgia Institute of Technology, March 9, 1994

[23] **Osuagwu, O. E., Oladipo, O. F. and Yinka-Banjo, C**. (2008). Deploying Reverse Software Engineering as an instrument for Converting Legacy Applications in critical-sensitive systems for Nigerian Industries. In Proceedings of the 22nd National Conference and AGM of the Nigeria Computer Society Conference (ENCTDEV 2008). 24- 27 June

[24] Reverse engineering from Wikipedia the free encyclopedia. http://en.wikipedia.org/wiki/ Reverse_engineering

[25] **Oladipo, O.F**. (2010a). Software Reverse Engineering of Legacy Applications. Unpublished Ph.D. Thesis. Computer Science Department, Nandi Azikiwe University, Awka Nigeria. External Assessment, November 2009

[26] **Main, A. and Van Oorschot, P.C.** (2003). Software Protection and Application Security: Understanding the Battleground. State of the Art and Assessment of Computer Security and Industrial Cryptography, June 2003, Heverlee, Belgium, Springer-Verlag LNCS

[27] **M. Sensalire, P. Ogao, A.** Telea, Evaluation of software visualization instruments: Lessons learned, in 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT'09), 2009

[28] **Magnusson, P. S., Christianson, M., Eskilson, J.** (2002). Simics: A full system simulation platform, IEEE Computer vol.35 no.2 (Feb.2002), pp.50-58

[29] **Kennell, R. and Jamieson, L.H.** (2003). Establishing the Gentry of Remote Computer Systems, Proceedings of the 12th USENIX Security Symposium (August 2003), pp.295-310

[30] **Xiang-Hu Wu, Ming-Cheng Qu, Zhi-Qiang Liu, Jian-Zhong Li**: Research and Application of Code Automatic Generation Algorithm Based on Structured Flowchart Journal of Software Engineering and Applications, 2011, 4, 534-545 DOI: 10.4236/jsea. 2011.49062

[31] **Aivosto**, (2003). Visustin Flow Chart Generator [online]. Aivosto.com. Available from: http://www.aivosto.com/visustin.html [Accessed 21-5-2020]

[32] http://www.Imagix.com/

[33] **Auber, D., Melancon, G., Munzner, T. And Weiskopf, D**. (2010) SolidSX: A Visual Analysis Instrument for Software Maintenance, Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization

[34] http://www.cocodex.com/

[35] **CARLISLE, M., WILSON, T., HUMPHRIES, J. & HADFIELD, S.,** (2005), RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving, ACM SIGCSE Bulletin, 37,1, ACM, New York - NY - USA, pp 176-180

[36] **GREYLING, J., CILLERS, C. & CALITZ, A**., (2006), B# The Development and Assessment of an Iconic Programming Instrument for Novice Programmers, 7th International Conference on Information Technology Based Higher Education and Training, Ultimo - NSW - Australia, IEEE, pp 376-375