Why Teach Reverse Engineering?

Muhammad Raza Ali
raza5680@hotmail.com

Abstract

Software reverse engineering is a fascinating discipline of software engineering. But it has failed to attract attention from students. Largely due to the facts that many universities around the world do not offer relevant courses, developing new software has always been considered superior then to maintain existing systems. But owing to the arrival of the internet, and client-server technology. Many organizations wish to adapt their existing systems. Thus the trend has somewhat shifted towards software evolution and maintenance. And now, more than ever before we need software engineers who can work effectively with legacy systems. In this paper I wish to highlight importance of incorporating Reverse engineering concepts and techniques into software engineering curriculum. I will start with a brief overview of reverse engineering concepts, and then discuss advantages of teaching reverse engineering.

Keywords

Legacy systems, software engineering, software reverse engineering, reengineering, program understanding, software evolution and maintenance, obfuscate, Internet, and plagiarism.

Introduction

Reverse engineering can roughly be defined as taking a final product, dissecting it to understand its functionality, obtain design and other useful information. And Software Reverse Engineering is to analyze system's code, documentation and behavior to create system abstractions and design information. And understanding internal system complexities. While having access only to external behavior. And it is the first phase of reengineering process.
There are three different of approaches to the system analysis employing reverse engineering.

White Box analysis consists of analyzing and understanding program's code without running it. This 'static analysis' approach is used to produce not only the potential program but also to obtain information regarding characteristics of the code.
Black box analysis involves checking program behavior by inputs. It is usually done before white box analysis. It is often used to specify areas for white box analysis.
Third approach is Gray-Box Analysis. In which black and white box analysis is used together. Some parts of the system
treated as a black box while others looked at more closely using white box analysis.

Using above mentioned approaches software engineers usually use one of the following methods.

1. Using the tedious approach of stepping through the code from initial input. This method is called Input tracing.
2. Another interesting method to analyze system is to study it's different versions.
3. Code Coverage is a technique to help in constructing a code-pathway map.
4. Kernel access is used to run device drivers and insert addresses into kernel memory.
5. Leftover data in shared buffers can be used to extract important information regarding the system and expose its vulnerabilities.
6. Often APIs are studied to point functions that can cause problems.

Learning the Art

Main purpose of my paper is to convince academics who are involved in designing software engineering curriculum at universities especially at undergraduate level and also those who can take this course in university. Owing to ever changing nature of real world problems and merging of various diverse fields, engineering education as a whole is undergoing continuous evolution. And reverse engineering provides students a taste of industrial practice along with academic research.

The importance of Reverse engineering concepts has been realized by many universities and it is being used as a tool in many disciplines of science and engineering. Approach of Rowan University [3] for instance is to instill maximum confidence in its engineering students to work on various multi-disciplinary real world problems. And they are using Reverse Engineering projects to achieve their goal. And the goal is produce practicing engineers who have adaptability, flexible and innovative thinking and can take calculated risks.

Another interesting research was conducted at University of Missouri-Rolla [5], by introducing reverse engineering techniques and encouraging students to study actual real world products. And the results were very encouraging. 77 % students thought that by introducing reverse engineering methodology reinforced concepts taught during lectures. And further 82% wanted it to be incorporated in future courses especially design courses. And it made practical work more fruitful and fun.

Benefits to Software Engineering Students

Sometimes, in software industry when it comes to skill set there is a very thin line between a person with a software engineering degree and a computer science degree. This is due to the fact that many software engineers prefer to work in application development. However, as a software engineer I have always felt

that Software engineers should work on things that are more challenging and indeed software reverse engineering is an adventure not meant for faint hearted. . And studying reverse engineering will certainly develop necessary skills. In this section I will try to explain how software engineering students can benefit:

1. Better and deeper understanding is the first and foremost advantage of teaching reverse engineering concepts. Valuable knowledge is gained regarding software and hardware functionality. Along with, understanding of file systems, and protocols etc. These things beyond the 'pretty' IDE are very important. Hackers and crackers, for instance, have long bragged their better and deeper understanding of system level details. And in the last few years' software community has been forced taken notice after they unlocked popular commercial software license managers. And now some experts recommend software developers and engineers to attend hacker conventions [1]. As these conferences show their passionate pursuit of technical excellence and learning for the love of it, these characteristics should be inculcated through classroom. Since this 'software underworld' is adventurous and they are ready to think out of the box their skills are often at par with their qualified counterparts.

   Practice with reverse engineering techniques improves ability to understand a given system quickly and efficiently. This skill has serious economic effects when software engineers are called upon to solve problems pertaining to software evolution. As fifty to ninety percent of the time, depending upon the system under consideration is spent on program understanding [6]. And reverse engineering is used as a significant aid to program comprehension and from there, recovery of higher level abstractions [4, 7].

   Due to the shift towards Web technology companies are now adapting their systems to Web. And experts agree that reverse engineering has helped tremendously in solving legacy system issues. And reverse engineering is now ready to be taught in computer engineering and software engineering curricula [8].

   Reverse engineering no doubt is difficult, time consuming and painstaking but it is very informative as well. You will come across many people who have used reverse engineering for better understanding of various software technologies, and learned things as advanced as operating system programming [2, 8].

   It is worth mentioning here that software engineering education has benefited a lot form 'open source' movement. Many educators are using it to teach software engineering courses . Just as students in other disciplines including art, law learn a lot by studying works of others software engineering students can benefit a lot by looking code written by more experienced programmers.

2. Software industry is perhaps the most rapidly changing industry. Not only it is advancing itself with breakthroughs but consequently it has changed practices in other fields of science and engineering. Fascinating fields like Bio-informatics are attracting researchers and students alike. So, we need Software engineering graduates who have can adapt fast and cope with this intermingling of disciplines. Reverse engineering is one teaching tool which can provide confidence and skills to do so. Students should be encouraged to undertake projects involving of study of real life products of versatile nature.

   By employing reverse engineering in assignments and projects programming and design skills can be vastly improved. Teachers should encourage students to self analyze their program/design by reverse engineering to point out limitations and remove them. Or they can analyze assignments of their peers. This experience can be very useful as often companies analyze products of their competitors to check for potential infringement [2]. And also in a scenario where there is lack of fixed standard, and a particular company is trying make it's product compatible with another product i.e. Interoperability[2].

   Programs written by students are very small and straightforward compared to what they will do in the industry. Realizing this, a very unique and fruitful research was conducted [9]. It focused on using open source software as a classroom material. Instead of designing and implementing new systems students were asked to modify one of the given open source systems. This experience is valuable as often software engineers have to deal and maintain code written by other people. And it not only enhanced programming experience of students in terms of size of code and development methods. But also generates discussions on topics such software piracy and ethics, which seldom feature in class activity.

   One very critical question that is asked before choosing a major is its job scope especially when you are in a developing economy. Same applies for available optional courses; students tend towards courses with obvious benefit. Reverse engineering knowledge will be a valuable addition to one's skill set. As more companies realize the importance of this skill to cut cost of maintaining legacy systems and to minimize risks in software evolution [8]. Pressed by industrial requirement there is a lot of scope for research as researchers work on developing better reverse engineering methodologies. And focus on software maintenance and reengineering has evolved as a central part of software engineering research [20]. And it is believed that legacy system relevant issues will now be an essential part software literature [11, 12];

3. Since reverse engineering a system is a difficult, patient job it helps one's PROBLEM SOLVING SKILLS. And more often than not, the determining factor of one's career success. Strong analytical and problem solving skills are very important when dealing with legacy system and software integration issues. Not every time we are required to build a system from scratch. Existing systems are sometimes requiring maintenance and modification. Usually fresh graduates are unable to work on a such a project due to no experience in software reverse engineering and re-engineering. But many universities have realized it's importance and some educators are even using GNU software

to teach advanced courses in reverse engineering [10]. And teaching reverse engineering to graduates has opened a new and exciting avenue for their career. And skills they develop while working with legacy systems will give them an edge. Especially when we take into account the changing trend in software industry. Traditionally software practitioners have focused mainly on development of new software. But with scenarios like Y2K and migration towards the internet companies are devoting more to understanding of legacy systems. Even if the system is to be retired, it has to be understood as new system will be functionally similar to the existing one. Arrival and acceptance of internet as a universal interface and rise of client server technology has definitely increased the demand for understanding legacy systems [11]. As around 70% of world's source code is in COBOL and in scientific communities FORTRAN has been the obvious choice. [13]. And code is usually a mess, design is poor and quality documentation is often missing. This situation demands trained and skillful software engineers who can work effectively and efficiently with legacy systems. And to meet future requirements in software evolution and maintenance software reverse engineering courses should be introduced as an essential part of software engineering curricula.

4. Many universities around the world offer reverse engineering at an advanced level and they are conducting research. But why is it important to make reverse engineering an essential part of the curricula at undergraduate level at all universities.

   As we know that traditional software engineering syllabus is 'forward engineering' based. As a result developing new software is considered far superior. That's why new graduates are hardly interested in working with legacy systems and they lack the essential skills to do so. And as long as we don't software reverse engineering principles and students do not gain hands-on experience with small or medium scaled legacy system, this trend will continue. And we need people to do the 'dirty work'. And a lot of stress is now being laid on teaching reverse engineering concepts not only as a separate subject but also part various other courses e.g. database reverse engineering [14].

   Student at undergrad level is more receptive to things being taught And approach to problem solving developed during four years of undergrad study will remain more or less the same. Regardless which field of computing they choose later on. Analytical skills gained through learning reverse engineering principles will help software engineering students. As reverse engineering encourages thinking 'out of the box'.

   Software piracy and illegal reversing of the disassembly has been a major concern for the software manufacturers. Another wonderful use of reverse engineering is to make illegal reversing difficult. Because sometimes to 'defeat a crook you have to think like one'. Reversers rely heavily on certain powerful editors and debuggers namely SoftIce, HexEdit, W32DASM etc. These tools are very powerful and by using these tools you'll know their strengths and limitations. And having this in mind you will be able to develop software that is difficult to reverse. And develop sound protection schemes. Like Microsoft recommends to developers working with .NET technology to obfuscate their code making it's reversing hard and time consuming [17]. Software developers and security experts are being urged to master the art of reverse engineering because industries like antivirus industry survive on it's professionals ability to quickly understand binary code of viruses and Trojans [16]. And before long next generation of computing students, might be required to study unconventional topics like viruses and worms in classroom [19].

Conclusion

After the 1990s the demand by all business sectors to efficiently and cost effectively adapt their systems to web interfaces has increased enormously. And software reverse engineering has been heralded as the promising technology to combat legacy system problems[8]. With this changing trend every software engineer must be educated and trained in software reverse engineering. It is also suggested students should not be discouraged to use design of somebody as long as they acknowledge it. Because in software circles, taking anything, how small it might be. 'unacknowledged' is considered stealing. This is an advanced level technique and a great learning experience. This type of 'plagiarism' can be of great use [15]. And on the research side need will be there to develop better methodologies, tools etc. And if look at the publications of past few years Research in maintenance and reengineering has flourished [20]. Thus reverse engineering is opening a new and a fascinating field for computer scientists and software engineers in both industry and academia.

References

[1] Gregroy Conti (2005): Why Computer Scientists Should Attend Hacker Conferences, Communications of the ACM, March 2005.

[2] Cem Kaner (1998): Article 2B and Reverse Engineering, *(www.badsoftware.com), July 1998*.

3]Debbie Barsotti (2003): Giving back in a big way, *(www.progressiveengineer.com), March 2003*.

[4]Leon Moonen (2002). Exploring Software Systems, (PhD Thesis University of Amsterdam, Faculty of Natural Sciences, Mathematics and Computer Science), October 2002.

[5] Robert. B. Stone (Dept. of Basic Engineering), Daniel A. McAdams (Dept. of Mechanical Engineering), The Touchy –Feely Side of Engineering Education: Bringing hands-on Experience to Classroom, University of Missouri-Rolla.

[6] Hausi Muller, Kenny Wong, Scott Tilley. Understanding Software Systems Using Reverse Engineering Technology, Colloquium on Object Orientation in Databases and Reverse Engineering: *The 62nd Congress of "L'Association Canadiene Francaise pour l'Avancement des Sciences (AFCAS)";* (May 16-17; Montreal; Quebec; Canada).

[7] Scott R. Tilley, Hausi A. Muller, Margaret-Anne Storey, Kenny Wong: Programmable Reverse Engineering. *This work is supported by British Colombia advanced Systems Institute, IBM Software Solutions Toronto, and IRIS Center for Excellence, Natural Sciences and Engineering Council of Canada, Council of British Colombia, and the University of Victoria.*

[8] Scott R. Tilley, Hausi A. Muller, Margaret-Anne Storey, Kenny Wong, Jens H. Jhanke, Dennis B. Smith(2000): Reverse Engineering: A Roadmap, Taken from *"The Future of Software Engineering" , Anthony Finkelstein (Ed.)*, ACM Press 2000.

[9] David Carrington, Soon-Kyeong Kim (2003): Teaching Software Design with Open Source Software, *33rd ASEE/IEEE Frontiers in Education Conference*, 5-8 Nov 2003, and Boulder, CO.

[10] J.H Andrews and H.L Lutfiyya (2003): Experience with a Software Maintenance Project Course, *IEEE Trans. Education,* November 2003.

[11] Michael L. Nelson (1996): A Survey of Reverse Engineering and Program Comprehension,
ODU CS 551-Software Engineering Survey, April 19 1996.

[12] S. Rugaber, K. Stirewart, and L. Wills (1995): The Interleaving Problem in Program Understanding, *2nd Working Conf. on Reverse Engineering, Toronto, Ontario, Canada, July 14-16 1995*.

[13] Edward Yourdon (1989): *Structured Walkthroughs,* Yourdon Press 1989.

[14] Anthony Finkelstein (UCL), Jeff Kramer (Imperial College London). *Software Engineering: A Roadmap.*

[15] J. Paul Gibson (2003): Software Reuse in Final Year Projects: A Code of Practice, *Report NUIM-CS-2003-TR-12(National Uni. Of Ireland, Maynooth)*, November 2003.

[16] Cyrus Piekari, and Anton Chuvakin(2004): *Security Warrior: Know your Enemy,* O'REILLY, February 2004.

[17] Gabriel Torok, and Bill Leach (2003). *Obfuscate It Thwart Reverse Engineering of Your Visual Basic .NET or C# Code*, MSDN® Magazine the Microsoft Journal for Developers, November 2003.

[18] Brian Blong. *Reverse Engineer to Learn .NET Better,* (www.blong.com/Conferences/DCon2003/ ReverseEngineering/ReverseEngineering.htm.)

[19] George Ledin Jr(2003): *Not Teaching Viruses and Worms Is Harmful,* Communication of the ACM, January 2005.

[20]Ahmed E. Hassan, and Richard C. Holt: *The Small World of Reverse Engineering*, Software Architecture Group (SWAG), School of Computer Science, University of Waterloo, Canada.