Reverse engineering

Reverse engineering is the process of discovering the technological principles of a human (or non-human) made 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 to be used in maintenance, or to try to make a new device or program that does the same thing without using or simply duplicating (without understanding) the original.

Reverse engineering has its origins in the analysis of hardware for commercial or military advantage.[1] The purpose is to deduce design decisions from end products with little or no additional knowledge about the procedures involved in the original production. The same techniques are subsequently being researched for application to legacy software systems, not for industrial or defence ends, but rather to replace incorrect, incomplete, or otherwise unavailable documentation.[2]

Motivation

Reasons for reverse engineering:

- Interoperability.
- Lost documentation: Reverse engineering often is done because the documentation of a particular device has been lost (or was never written), and the person who built it is no longer available. Integrated circuits often seem to have been designed on obsolete, proprietary systems, which means that the only way to incorporate the functionality into new technology is to reverse-engineer the existing chip and then re-design it.
- Product analysis. To examine how a product works, what components it consists of, estimate costs, and identify potential patent infringement.
- Digital update/correction. To update the digital version (e.g. CAD model) of an object to match an "as-built" condition.
- Security auditing.
- Acquiring sensitive data by disassembling and analysing the design of a system component.[3]
- Military or commercial espionage. Learning about an enemy's or competitor's latest research by stealing or capturing a prototype and dismantling it.
- Removal of copy protection, circumvention of access restrictions.
- Creation of unlicensed/unapproved duplicates.
- Materials harvesting, sorting, or scrapping.[4]
- Academic/learning purposes.
- Curiosity.
- Competitive technical intelligence (understand what your competitor is actually doing, versus what they say they are doing).
- Learning: learn from others' mistakes. Do not make the same mistakes that others have already made and subsequently corrected.

Reverse engineering of machines

As computer-aided design (CAD) has become more popular, reverse engineering has become a viable method to create a 3D virtual model of an existing physical part for use in 3D CAD, CAM, CAE or other software.[5] The reverse-engineering process involves measuring an object and then reconstructing it as a 3D model. The physical object can be measured using 3D scanning technologies like CMMs, laser scanners, structured light digitizers or Industrial CT Scanning (computed tomography). The measured data alone, usually represented as a point cloud, lacks topological information and is therefore often processed and modeled into a more usable format such as a
triangular-faced mesh, a set of NURBS surfaces or a CAD model.

Reverse engineering is also used by businesses to bring existing physical geometry into digital product development environments, to make a digital 3D record of their own products or to assess competitors’ products. It is used to analyse, for instance, how a product works, what it does, and what components it consists of, estimate costs, and identify potential patent infringement, etc.

Value engineering is a related activity also used by businesses. It involves de-constructing and analysing products, but the objective is to find opportunities for cost cutting.

**Reverse engineering of software**

The term *reverse engineering* as applied to software means different things to different people, prompting Chikofsky and Cross to write a paper researching the various uses and defining a taxonomy. From their paper, they state, "Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction."[6] It can also be seen as "going backwards through the development cycle".[7] In this model, the output of the implementation phase (in source code form) is reverse-engineered back to the analysis phase, in an inversion of the traditional waterfall model. Reverse engineering is a process of examination only: the software system under consideration is not modified (which would make it re-engineering). Software anti-tamper technology is used to deter both reverse engineering and re-engineering of proprietary software and software-powered systems.

In practice, two main types of reverse engineering emerge. In the first case, source code is already available for the software, but higher-level aspects of the program, perhaps poorly documented or documented but no longer valid, are discovered. In the second case, there is no source code available for the software, and any efforts towards discovering one possible source code for the software are regarded as reverse engineering. This second usage of the term is the one most people are familiar with. Reverse engineering of software can make use of the clean room design technique to avoid copyright infringement.

On a related note, black box testing in software engineering has a lot in common with reverse engineering. The tester usually has the API, but their goals are to find bugs and undocumented features by bashing the product from outside.

Other purposes of reverse engineering include security auditing, removal of copy protection ("cracking"), circumvention of access restrictions often present in consumer electronics, customization of embedded systems (such as engine management systems), in-house repairs or retrofits, enabling of additional features on low-cost "crippled" hardware (such as some graphics card chip-sets), or even mere satisfaction of curiosity.

**Binary software**

This process is sometimes termed *Reverse Code Engineering*, or RCE.[8] As an example, decompilation of binaries for the Java platform can be accomplished using Jad. One famous case of reverse engineering was the first non-IBM implementation of the PC BIOS which launched the historic IBM PC compatible industry that has been the overwhelmingly dominant computer hardware platform for many years. An example of a group that reverse-engineers software for enjoyment (and to distribute registration cracks) is CORE which stands for "Challenge Of Reverse Engineering". Reverse engineering of software is protected in the U.S. by the fair use exception in copyright law.[9] The Samba software, which allows systems that are not running Microsoft Windows systems to share files with systems that are, is a classic example of software reverse engineering,[10] since the Samba project had to reverse-engineer unpublished information about how Windows file sharing worked, so that non-Windows computers could emulate it. The Wine project does the same thing for the Windows API, and OpenOffice.org is one party doing this for the Microsoft Office file formats. The ReactOS project is even more ambitious in its goals, as it strives to provide binary (ABI and API) compatibility with the current Windows OSes of the NT branch, allowing software and drivers written for Windows to run on a clean-room reverse-engineered GPL free software or open-source counterpart.
Reverse engineering techniques

Reverse engineering of software can be accomplished by various methods. The three main groups of software reverse engineering are

1. Analysis through observation of information exchange, most prevalent in protocol reverse engineering, which involves using bus analyzers and packet sniffers, for example, for accessing a computer bus or computer network connection and revealing the traffic data thereon. Bus or network behavior can then be analyzed to produce a stand-alone implementation that mimics that behavior. This is especially useful for reverse engineering device drivers. Sometimes, reverse engineering on embedded systems is greatly assisted by tools deliberately introduced by the manufacturer, such as JTAG ports or other debugging means. In Microsoft Windows, low-level debuggers such as SoftICE are popular.

2. Disassembly using a disassembler, meaning the raw machine language of the program is read and understood in its own terms, only with the aid of machine-language mnemonics. This works on any computer program but can take quite some time, especially for someone not used to machine code. The Interactive Disassembler is a particularly popular tool.

3. Decompilation using a decompiler, a process that tries, with varying results, to recreate the source code in some high-level language for a program only available in machine code or bytecode.

Source code

A number of UML tools refer to the process of importing and analysing source code to generate UML diagrams as "reverse engineering". See List of UML tools.

Reverse engineering of protocols

Protocols are sets of rules that describe message formats and how messages are exchanged (i.e., the protocol state-machine). Accordingly, the problem of protocol reverse-engineering can be partitioned into two subproblems; message format and state-machine reverse-engineering.

The message formats have traditionally been reverse-engineered through a tedious manual process, which involved analysis of how protocol implementations process messages, but recent research proposed a number of automatic solutions. Typically, these automatic approaches either group observed messages into clusters using various clustering analyses, or emulate the protocol implementation tracing the message processing.

There has been less work on reverse-engineering of state-machines of protocols. In general, the protocol state-machines can be learned either through a process of offline learning, which passively observes communication and attempts to build the most general state-machine accepting all observed sequences of messages, and online learning, which allows interactive generation of probing sequences of messages and listening to responses to those probing sequences. In general, offline learning of small state-machines is known to be NP-complete, while online learning can be done in polynomial time. An automatic offline approach has been demonstrated by Comparetti et al. and an online approach very recently by Cho et al.

Other components of typical protocols, like encryption and hash functions, can be reverse-engineered automatically as well. Typically, the automatic approaches trace the execution of protocol implementations and try to detect buffers in memory holding unencrypted packets.
Reverse engineering of integrated circuits/smart cards

Reverse engineering is an invasive and destructive form of analyzing a smart card. The attacker grinds away layer by layer of the smart card and takes pictures with an electron microscope. With this technique, it is possible to reveal the complete hardware and software part of the smart card. The major problem for the attacker is to bring everything into the right order to find out how everything works. Engineers try to hide keys and operations by mixing up memory positions, for example, busscrambling.[18][19] In some cases, it is even possible to attach a probe to measure voltages while the smart card is still operational. Engineers employ sensors to detect and prevent this attack.[20] This attack is not very common because it requires a large investment in effort and special equipment that is generally only available to large chip manufacturers. Furthermore, the payoff from this attack is low since other security techniques are often employed such as shadow accounts.

Reverse engineering for military applications

Reverse engineering is often used by militaries in order to copy other nations' technologies, devices or information that have been obtained by regular troops in the fields or by intelligence operations. It was often used during the Second World War and the Cold War. Well-known examples from WWII and later include

- Jerry can: British and American forces noticed that the Germans had gasoline cans with an excellent design. They reverse-engineered copies of those cans. The cans were popularly known as "Jerry cans".
- Tupolev Tu-4: Three American B-29 bombers on missions over Japan were forced to land in the USSR. The Soviets, who did not have a similar strategic bomber, decided to copy the B-29. Within a few years, they had developed the Tu-4, a near-perfect copy.
- V2 Rocket: Technical documents for the V2 and related technologies were captured by the Western Allies at the end of the war. Soviet, and captured German engineers, had to reproduce technical documents and plans, working from captured hardware, in order to make their clone of the rocket, the R-1, which began the postwar Soviet rocket program that led to the R-7 and the beginning of the space race.
- K-13/R-3S missile (NATO reporting name AA-2 Atoll), a Soviet reverse-engineered copy of the AIM-9 Sidewinder, made possible after a Taiwanese AIM-9B hit a Chinese MiG-17 without exploding; amazingly, the missile became lodged within the airframe, the pilot returning to base with what Russian scientists would describe as a university course in missile development.
- BGM-71 TOW Missile: In May 1975, negotiations between Iran and Hughes Missile Systems on co-production of the TOW and Maverick missiles stalled over disagreements in the pricing structure, the subsequent 1979 revolution ending all plans for such co-production. Iran was later successful in reverse-engineering the missile and are currently producing their own copy: the Toophan.
- China has reversed engineered many examples of Western and Russian hardware, from fighter aircraft to missiles and HMMWV cars.
- During the Second World War, British military intelligence at the Bletchley Park centre, studied captured German "Enigma" message encryption machines, their operation was then simulated on electro-mechanical devices called "Bombes", that tried all the possible scrambler settings of the "Enigma" machines, to help break the coded messages sent by the Germans.
Legality

United States

In the United States even if an artifact or process is protected by trade secrets, reverse-engineering the artifact or process is often lawful as long as it is obtained legitimately. Patents, on the other hand, need a public disclosure of an invention, and therefore, patented items do not necessarily have to be reverse-engineered to be studied. (However, an item produced under one or more patents could also include other technology that is not patented and not disclosed.) One common motivation of reverse engineers is to determine whether a competitor’s product contains patent infringements or copyright infringements.

The reverse engineering of software in the US is generally illegal because most EULA prohibit it, and courts have found such contractual prohibitions to override the copyright law; see Bowers v. Baystate Technologies. Sec. 103(f) of the DMCA (17 U.S.C. § 1201 (f)) says that if you legally obtain a program that is protected, you are allowed to reverse-engineer and circumvent the protection to achieve the ability the interoperability of computer programs (i.e., the ability to exchange and make use of information). The section states:

(f) Reverse Engineering.—

1. Notwithstanding the provisions of subsection (a)(1)(A), a person who has lawfully obtained the right to use a copy of a computer program may circumvent a technological measure that effectively controls access to a particular portion of that program for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs, and that have not previously been readily available to the person engaging in the circumvention, to the extent any such acts of identification and analysis do not constitute infringement under this title.

2. Notwithstanding the provisions of subsections (a)(2) and (b), a person may develop and employ technological means to circumvent a technological measure, or to circumvent protection afforded by a technological measure, in order to enable the identification and analysis under paragraph (1), or for the purpose of enabling interoperability of an independently created computer program with other programs, if such means are necessary to achieve such interoperability, to the extent that doing so does not constitute infringement under this title.

3. The information acquired through the acts permitted under paragraph (1), and the means permitted under paragraph (2), may be made available to others if the person referred to in paragraph (1) or (2), as the case may be, provides such information or means solely for the purpose of enabling interoperability of an independently created computer program with other programs, and to the extent that doing so does not constitute infringement under this title or violate applicable law other than this section.

4. For purposes of this subsection, the term ‘interoperability’ means
the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged.

**European Union**

Article 6 of the 1991 EU Computer Programs Directive allows reverse engineering for the purposes of interoperability, but prohibits it for the purposes of creating a competing product, and also prohibits the public release of information obtained through reverse engineering of software.\[25\] \[26\] \[27\]

In 2009, the EU Computer Program Directive was superseded and the directive now states:\[28\]

(15) The unauthorised reproduction, translation, adaptation or transformation of the form of the code in which a copy of a computer program has been made available constitutes an infringement of the exclusive rights of the author. Nevertheless, circumstances may exist when such a reproduction of the code and translation of its form are indispensable to obtain the necessary information to achieve the interoperability of an independently created program with other programs. It has therefore to be considered that, in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of a person having a right to use a copy of the program is legitimate and compatible with fair practice and must therefore be deemed not to require the authorisation of the rightholder. An objective of this exception is to make it possible to connect all components of a computer system, including those of different manufacturers, so that they can work together. Such an exception to the author's exclusive rights may not be used in a way which prejudices the legitimate interests of the rightholder or which conflicts with a normal exploitation of the program.

**References**


Reverse engineering


Further reading


External links

- Java Call Trace to UML Sequence Diagram (https://sourceforge.net/projects/javacalltracer/) A reverse engineering tool for Java. This tool helps you to reverse engineer UML Sequence Diagram for your java program at runtime. It works well with both complex java programs (that have multiple threads) and J2EE applications deployed on Application Servers.
- Additional tutorials, bibliographies and related sites (http://www.imagix.com/links/reverse_engineering.html) for reverse engineering of available source code.