Classical system approach to reverse engineering

A V Ivonov, E V Volkova and I M Daudov

1 V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russia
2 Irkutsk National Research Technical University, 83, Lermontov street, Irkutsk, 664074, Russia
3 Chechen State University, 17a, Dudaev Boulevard, Groznyy, 364015, Russia

E-mail: kshakalov@mail.ru

Abstract. The paper presents one of the possible solutions of the reverse engineering problem from the point of view of the classical system approach, as well as highlights the direction for the development of a complex software product. This approach will make it possible to change the modern practice of reverse engineering, which is applied everywhere. This practice requires expert evaluation by operator teams on non-acceptance basis. The approach proposed by the authors will significantly minimize human participation in this process.

1. Introduction
Modern practical reverse engineering mainly relies on the inverse conversion of the executable code into texts in assembler language and even high-level languages. This is followed by the expert evaluation of actions of commands and operators that this code performs. The tools thus used include debuggers, disassemblers and decompilators, which are designed for direct development and adjustment of a software product. Besides, there are techniques that make it difficult for program developers to analyze them. This can be solved by a classical system approach, which involves not only modeling the executing code, but also the state of a computer or a computer system as a whole.

2. Theory
One of the founders of the mathematical theory of systems is Professor Rudolf Kalman of Stanford University. He formulated the most general and strict mathematical definition of the system at the level of the set theory: “The system is presented in the form of a group of sets and mapping of a set by a set”.

The dependence of the system on time is given by mapping its parameters to an ordered set of time points, which is typical for describing systems from the point of view of system analysis and neurophenomenological approach [1].

The states of the system are described in the form of a plurality of permissible states of the system, from which a plurality of instantaneous states of the system are distinguished by means of a special display, referred to as a narrowing (section). Similarly, multiple inputs and outputs are specified. It also considers specific approaches to the system description of practical engineering problems. The most common linear model is the discrete time model. This model perfectly describes a clock-driven computer, which states change from one cycle to another. The mathematical model of a computer can be defined as a linear dynamic system with discrete time [2]:
\[
\bar{X}_{k+1} = \Phi(t_{k+1} | t_k) \bar{X}_k + B_k (\bar{V}_k + \bar{\xi}_k),
\]

where \(\bar{X}_k\) – state vector at the moment \(t_k\), \(\Phi(t_{k+1} | t_k)\) – transition matrix of a state vector from the moment \(t_{k+1}\). \(\bar{V}_k\) – vector of documented impacts, \(\bar{\xi}_k\) – vector of unaccounted (undocumented) impacts on a computer condition. \(B_k\) – matrix describing the change in the state of a computer due to control effects. The observer over a condition of the computer cannot have complete information on vector \(\bar{X}_c\). Only some parameters derived from this state vector are subject to observation. The observation equations will be as follows:

\[
\bar{Y}_k = C_k \bar{X}_k
\]

\(C_k\) – observation matrix. It describes the relations between the parameters that a user can extract from a computer and the state of the computer. Vector \(\bar{V}_k\) and matrix \(B_k\) may be set in a way that \(\bar{V}_k\) can be considered as a computer program, and \(\bar{\xi}_k\) – as a strange, undescribed code.

If all matrixes of the model are set and the vector of a system state \(\bar{X}_c\) is fully observed [3] in terms of a group of measurements \(\bar{Y}_k\), then a problem of identifying \(\bar{\xi}_k\) is reduced to modeling of the sequence of states at the absence of \(\bar{\xi}_k\) and to receiving real estimates \(\bar{X}_k\) according to measurements \(\bar{Y}_k\). From the differences in the parameters of the system states, a system of linear equations can be obtained:

\[
(X_k - \bar{X}_k) = B_k \bar{\xi}_k
\]

If the matrix \(B_k\) has no inverse form, then equations with different \(k\) can be combined, but at least part of the components of the sought vector \(\bar{\xi}_k\) will not change depending on number \(k\). However, this is not needed since those components \(\bar{\xi}_k\), which cannot be defined from this equation are not of interest and the lines of the matrix \(B_k\) for these components can be excluded.

Similarly, it is possible to consider the problem of detecting program execution errors as a result of errors of command interpretation and command actions, which are not provided in the technical documentation [6].

For example, in practice, such an algorithm may be performed in oil by-product recovery to automate the process [2, 5, 8]. It is also used to control underground mining machines in order to prevent the creation of emergency situations (failure of the machine, sharp change of mining and geological conditions) [4, 7, 9, 10].

3. Actual practice

However, in real practice, the dimensions of vectors and matrices are quite high, and the connections between them are very complex. A detailed description of modern computer systems alone represents a scientific problem called Systems Structural Analysis.

The system structural analysis is a set of methodologies that allow presenting a description of a system in a convenient hierarchical way. Text descriptions and diagrams are applied. The best known methodology for system structural analysis is a combination of data flow diagrams, entity-relationship diagrams, and state transition diagrams. The so-called unified UML methodology is used for the object-oriented approach.

The automation of such descriptions and diagrams is called CASE Technologies. These methodologies are designed for engineering and generation of a software code, i.e. for forward
engineering.

Equation (2) describes the monitoring of computer parameters, the loaded system, and user programs. There are many programs and utilities that allow capturing the status settings of a computer and networks. These programs are mainly designed for system administrators who perform tasks of software and hardware health control, detect conflicts between programs, contradictions between user actions, etc. Quite often these are similar, standard actions and even a unified approach is offered. Nikolai Likhachev, known under the pseudonym Chris Kasperski, describes the monitoring of the state of computers and networks in terms of information protection.

Currently, there is an empirical approach in the field of reverse engineering. The executable program, its disassembled, or even decompiled code, is interpreted empirically by an experienced expert. At the same time, both the assembler and the high-level language are designed to directly develop the program and it is not a fact that these languages are the most effective for reverse engineering. Reverse engineering is supported in parallel by the execution of the program code by a special debugger, which allows analyzing the current state of the processor and RAM.

In case of open source programs, the source code of the program is directly reversed. This is the most effective way to detect unauthorized inserts and undocumented features. Besides, after compiling the open code, the executable code is almost free of viruses, which, if implemented, only through a bot-infected computer.

In all the approaches used, the executable code is translated into an easy-to-interpret language, and then an attempt is made to highlight the characteristic elements of the code and determine its meaning. To some extent, operators and their sequences reflect the meaning of the executable sequence of actions – an algorithm.

The algorithm diagram can be rich enough to effectively describe the algorithm, and can be poor and implement the algorithm in small and obscure steps. An example of such a poor algorithm diagram is a Turing machine. It is able to describe any algorithm in almost five actions. Reverse engineering of the program written for the emulator of the Turing machine can be considered the most difficult for an expert, who is used to determine the meaning of an algorithm by numerous program language structures. On the other hand, the expert used to pass the algorithm step-by-step and analyze the result of actions will be in a better position. His actions will resemble those of an expert who passes the code through a debugger and analyzes the response of a computer.

We are almost between two extremes – an algorithm diagram poor in operations with complex logic and another extreme – abuse of semantics. The choice of too abstract queries and commands leads to complication of their implementation, congestion with flags, etc. Such schemes lead to incorrect interpretation of the executable code. This is the case in practice – modern viruses, Trojan horses and other harmful programs seek to deceive de-assemblers and decompilers. The multiple-version of modern machine languages makes it possible to represent the same action by different commands, to place part of the code in non-traditional areas of memory and hard disk, to store the program code in encrypted form and to decrypt it only at the moment of execution.

Another problem is to determine a harmful code and what computer states correspond to it.

In fact, we reach the computer model described by formulas (1) and (2). It allows strictly distinguishing harmful code elements on the basis of harmful changes in the state of a computer, but as we indicated, this model is very difficult to interpret. In the simplest case of state transition diagrams the use of CASE technologies will streamline the process of analyzing the state of a computer, simplify its analysis and identify the elements of a foreign code according to the impact not described in the technical documentation. However, the language of these descriptions itself is designed for forward engineering.

4. Conclusion
Effective reverse engineering should use not only the studied code, but also the model of behavior of a computer under its influence, for example, in the form of a dynamic system with discrete time. The complexity of analyzing a computer’s behavior pattern can be overcome by a specialized software
product based on CASE technologies.

The issue of creating specialized languages to effectively interpret a computer’s behavior pattern in order to detect a foreign code should be studied.

Since simple detection of the abnormal code is not sufficient for evidence-based computer expertise, there is a need to develop languages, descriptions, diagrams to capture and document the fact of its presence and therefore this may be the direction for creating new specialized software packages for reverse engineering.

Besides, it is necessary to study the system by means of the system analysis, mathematical modeling and programming, which will allow significantly minimizing human participation in the studied process [3, 11].

References

[1] Shutaleva A V 2019 The fundamental problem of subjectivity in Francisco Valera’s neurophenomenology Tomsk State Univer. J. of Philos. Sociol. and Political Sci. 48 84–90

[2] Ivanov A V and Strizhenok A V 2019 Ecological and economic justification of the possibility of utilization of weathering gases from gas condensate enterprises on the basis of heat generation ARPJ J. of Engineer. and Appl. Sci. 14(22) 3877–3885

[3] Yegorova E V, Klyuev R V, Bosikov I I and Tsidaev B. S 2018 Evaluation of use of effective technologies for increasing sustainable development of natural and technical system of oil and gas complex Sustainable Development of Mountain Territories 10(3) 392–403

[4] Shishlyannikov D I, Romanov V A, Zvonarev I E 2019 Determination of the operating time and residual life of self-propelled mine cars of potassium mines on the basis of integrated monitoring data Journal of Mining Institute 237 336-343

[5] Koptev V Yu, Kopteva A V, Poddubniy D A 2019 Increase in Energy Efficiency of Oil and Gas Companies by Perfecting of Management Systems 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 1 pp 548-552 Doi: 10.1109/EIConRus.2019.8657041

[6] Zhukovskiy Y, Koteleva N 2018 Development of augmented reality system for servicing electromechanical equipment Journal of Physics: Conference Series 1015 (4) 042068

[7] Buevich V V, Gabov V V, Zadkov D A, Vasileva P A 2015 Adaptation of the mechanized roof support to changeable rock pressure Eurasian mining 2 pp 11 - 14 doi: 10.17580/em.2015.02.03

[8] Fetisov V.G., Nikolaev A.K., Lykov Y.V. 2017. Experimental Studies for Determining Gas Flow Rate Accidental Release on Linear Part of Pipeline. IOP Conference Series: Earth and Environmental Science 87(6) 062003

[9] Korshak A A, Gaisin M T, Pshenin V V 2019 Method of structural minimization of the average risk for identification of mass transfer of evaporating oil at tanker loading. Neftyanye Khozyaystvo - Oil Industry 10 108-111

[10] Kopin S V 2016 Desing improvement of the nozzle fringe of the final branch pipe in local ventilation system by computer simulation. Bulletin of Civil Engineers 1(54) 108-112

[11] Urazbakhtin R Yu, Yungmeyster D A 2019 The results of studies of the tunneling rescue complex for coal mines IOP Conf. Ser.: Mater. Sci. Eng. 560 012130