The method for assessing security of embedded software based on the fuzzy logic

Andrei Iuganson and Danil Zakoldaev
ITMO University, 49 Kronverksky Pr., St. Petersburg, 197101, Russia

E-mail: a_yougusnon@itmo.ru

Abstract. Software occupies a key role in all information processes of society. Frequently, questions of software security are pushed into the background when designing and developing software for the quickest launch of a software product on the market. In this paper we have tried to explore some innovative measures and proposed a new approach for assessing software security. The subject of research is embedded software designed to control various devices and microcontrollers. The fuzzy logic was used to optimize the process of assessing in conditions of possible vagueness, inconsistency, incompleteness and qualitative nature of the initial information. This method will help to minimize economic risks on stages of exploitation and technical maintenance of embedded systems.

1. Introduction

Nowadays issues of software security become more relevant. The management of the development process is at a low level, which in turn leads to an increase in the number of errors in the program code and, as a consequence, to an increase in the cost of producing the final product. According to a study commissioned by the National Institute of Standards and Technology of the USA, losses arising from the weak development of the infrastructure for eliminating defects in software (vulnerabilities and programming errors) reach $60 billion a year [1]. The cost of eliminating a defect that was missed during the development and testing phases may increase many times after the delivery of the program.

Embedded software is computer software, written to control machines or devices that are not typically thought of as computers, e.g., cars and TV. According to recent surveys, approximately 90% of all processors are part of embedded systems, computing systems that continually and autonomously control and react to the environment. The embedded system itself is an information processing system that consists of hardware and software components. Nowadays, the number of embedded computing systems—in areas such as telecommunications, automotive, electronics, office automation, and military applications—is steadily growing [2].

Since software is a major component of embedded systems, it is very important to properly and adequately test the embedded software. Due to the complex system context of embedded-software applications, defects in these systems can cause life-threatening situations, and delays can lead to huge business losses [3].

As known security is a core requirement for any software system and embedded systems are no exception. This object has been achieved by adopting SDL (Security Development Life Cycle) addressing security issues at all levels [4].
Security is a quantitative characteristic which means probability of a software to maintain a required level of fitness under specific conditions for a specific time interval, where security defects and vulnerabilities are considered as a limitation on the level of fitness. A defect can also be defined as a product anomaly [5]. The probability is a function of the inputs to, and use of, the system as well as a function of the existence of faults in the software. The inputs to the system determine whether existing faults, if any, are encountered. Security is designed to quantify the likelihood of software failure. A failure is defined as the termination of the ability of a functional unit to perform its required function. In the other way, failure can be defined as an event in which a system or system component does not perform a required function within specified limits [6].

Modern standards for the development of industrial software imply a predominance of the problem of ensuring the security of software operation over the assessment of it. The paradox is that to properly ensure the level of security of the developed software, it is necessary to unambiguously determine this level. To date, the industry has not adopted a unified standard for assessing the security of embedded software.

The organization of the paper is as follow: Section 2 provides objective and details of fuzzy logic technique and model for evaluating security of the embedded software. Section 3 presents results of the model, implementation, and discussion. Section 4 is the conclusion of the paper.

2. Research objective
Despite the large number of publications reviewed [7,8], no study has proposed an effective method for assessing security of embedded software in conditions of possible vagueness, inconsistency, incompleteness and qualitative nature of the initial information.

Thus, the task of developing innovative measures and approaches for assessing security of embedded software is currently relevant. The main objective of the proposed research is to put forward a methodology for evaluating security of the embedded software. Here, a fuzzy logic approach is modelled to evaluate the qualitative metrics of embedded software. Fuzziness is beneficial in situations of vagueness and uncertainty, especially from evaluation by expert survey.

2.1. Fuzzy inference system
First of all, it is necessary to develop a fuzzy inference system for qualitative criteria, such as functional testing and vulnerability analysis.

In general, the development and application of fuzzy inference systems includes a number of steps, the implementation of which is based on the provisions of fuzzy logic [9,10]:

- determining a set of input variables;
- determining a set of fuzzy rules;
- fuzzification (representation of the physical meaning of the attribute in a linguistic form);
- combining the fuzzified inputs according to the fuzzy rules to establish a rule strength
- finding the consequence of the rule by combining the rule strength and the output membership function;
- combining the consequences to get an output distribution;
- defuzzifying the output distribution.

In this case, the fuzzy rule set is understood as a formal representation of the empirical knowledge of experts in the form of fuzzy production rules. In the Mamdani-type fuzzy inference systems A fuzzy production rule is an expression of the form:

$$R^i: \text{if } X_1 \text{ is } A^i_1 \text{ and } X_2 \text{ is } A^i_2 \text{ then } Y \text{ is } B^i$$ (1)
where \( i \) – number of rule, \( R_i \) – rule with number \( i \), \( X_i \) – inputs of the fuzzy inference systems, which are restricted by rule \( R_i \) in the form of the corresponding linguistic variable \( A_j \), \( Y \) – output of the fuzzy inference systems, which is accepting the values of the corresponding linguistic variable \( B_i \).

Mamdani-type fuzzy inference systems have higher linguistic interpretability that defines convenience of its use in man-machine systems.

The intersection (AND) and union (OR) operations that are used in fuzzy rules set are defined in the following way

\[
\mu_{A \cap B} = \min(\mu_A(x), \mu_B(x))
\]
\[
\mu_{A \cup B} = \max(\mu_A(x), \mu_B(x))
\]

Defuzzification is performed using method called Center Of Area (COA). It stipulates calculation of the resultant value \( res \) by way of

\[
res = \frac{\int_U x \mu(x) dx}{\int_U \mu(x) dx}
\]

3. Illustrative example
Let’s consider the assessment of the security of the embedded software at the stage of quality assurance testing. The systematized information on the existing methods for developing safe software in accordance with the stages of the product life cycle, were used as functional requirements for the security of the embedded software.

Functional testing and vulnerability analysis are considered as system’s input variables, whereas the quality assurance test is taken as an output variable.

Thus, the following linguistic variables \( L_i \) denoted as \( ATE_{FUN} \), \( AVA_{VAN} \) and QA_TEST and their values (labels of linguistic terms) are considered in table 1.

| \( T(ATE_{FUN}) \) | \( T(AVA_{VAN}) \) | \( T(QA) \) |
|---------------------|---------------------|-------------|
| value ‘unsatisfactory’ | value ‘unsatisfactory’ | value ‘very small’ |
| (1,1,5,20) | (1,1,5,20) | (1,1,5,20) |
| value ‘bad’ | value ‘bad’ | value ‘small’ |
| (5,15,30,40) | (5,15,30,40) | (5,15,30,40) |
| value ‘satisfactory’ | value ‘satisfactory’ | value ‘medium’ |
| (25,40,50,75) | (25,40,50,75) | (25,40,50,75) |
| value ‘good’ | value ‘good’ | value ‘large’ |
| (60,70,85,95) | (60,70,85,95) | (60,70,85,95) |
| value ‘excellent’ | value ‘excellent’ | value ‘very large’ |
| (80,95,100,100) | (80,95,100,100) | (80,95,100,100) |

The input and output fuzzy sets and membership functions are constructed based on the experts’ opinion. The fuzzy “IF-THEN” rules are framed to reflect the relationships between any possible relation of input variables and the output variable. The levels of the input parameters defined in above step are used in the antecedent of rules and the level of the output parameter is used as the consequent of rules.

Based on the proposed method, the fuzzy rules are obtained. The following table 2 gives a summary of these fuzzy rules.
Table 2. Fuzzy rules.

| Rule | Antecedent | Consequent |
|------|------------|------------|
| $R^1$ | ($\text{ATE}_\text{FUN} = \text{unsatisfactory}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{bad}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{unsatisfactory}) \land (\text{AVA}_\text{VAN} = \text{bad})$ | $\text{TEST}_{QA} = \text{very small}$ |
| $R^2$ | ($\text{ATE}_\text{FUN} = \text{unsatisfactory}) \land (\text{AVA}_\text{VAN} = \text{satisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{bad}) \land (\text{AVA}_\text{VAN} = \text{bad})$  
$\lor (\text{ATE}_\text{FUN} = \text{satisfactory}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{satisfactory}) \land (\text{AVA}_\text{VAN} = \text{bad})$ | $\text{TEST}_{QA} = \text{small}$ |
| $R^3$ | ($\text{ATE}_\text{FUN} = \text{unsatisfactory}) \land (\text{AVA}_\text{VAN} = \text{good})$  
$\lor (\text{ATE}_\text{FUN} = \text{satisfactory}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{bad}) \land (\text{AVA}_\text{VAN} = \text{good})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{bad})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{bad})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{good})$ | $\text{TEST}_{QA} = \text{medium}$ |
| $R^4$ | ($\text{ATE}_\text{FUN} = \text{satisfactory}) \land (\text{AVA}_\text{VAN} = \text{good})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{unsatisfactory})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{bad})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{good})$ | $\text{TEST}_{QA} = \text{large}$ |
| $R^5$ | ($\text{ATE}_\text{FUN} = \text{excellent}) \land (\text{AVA}_\text{VAN} = \text{excellent})$  
$\lor (\text{ATE}_\text{FUN} = \text{good}) \land (\text{AVA}_\text{VAN} = \text{excellent})$  
$\lor (\text{ATE}_\text{FUN} = \text{excellent}) \land (\text{AVA}_\text{VAN} = \text{good})$ | $\text{TEST}_{QA} = \text{very large}$ |

Considering that $\text{ATE}_\text{FUN}$ is 65% and $\text{AVA}_\text{VAN}$ got 38 points (out of 100), fuzzification can be performed as shown in figure 1.
Figure 1. Fuzzification of input parameters.

In this case, fuzzy rules $R_2$, $R_3$ and $R_4$ will give non-zero resultant value. As already stated above, Mamdani inference system (FIS) is used in the experiments – it allows to obtain an output in the form of fuzzy set. Rules $R_2$, $R_3$ and $R_4$ “fire”, thus ensuring non-zero results; in compliance with (2) and (3), we arrive at the following:

$R_2$: $\max(\min(0.67, 0.2)) = 0.2$ – membership degree that corresponds to the term ‘small’ (element of $T(\text{TEST}_Q)$).

$R_3$: $\max(\min(0.67, 0.87), \min(0.5, 0.2)) = 0.67$ – membership degree that corresponds to the term ‘medium’ (element of $T(\text{TEST}_Q)$).

$R_4$: $\max(\min(0.5, 0.87)) = 0.5$ – membership degree that corresponds to the term ‘good’ (element of $T(\text{TEST}_Q)$).

COA (Center Of Area) method (1) is applied to obtain crisp result. According to equation (4), the output value equals to approx. 53 story points as shown in figure 4.

Figure 2. Defuzzification of output parameter (COA, approx. 53 points).

The result of fuzzy modelling is a qualitative and quantitative assessment of the security of the embedded software at a given stage of the product life cycle, i.e. at the stage of quality assurance testing. This result is interpreted as an assessment of the possibility of neutralizing threats to the security of malware.

4. Conclusion and feature work
The embedded software should be so designed that the chances of its failure are minimum. The present software security assessing systems are based on binary system where there is either a safe or an unsafe state according to expert survey. The proposed approach has specifically tried to avert this situation by incorporating fuzzy logic to generate intermediate marks of expert survey. This new approach will take a step further towards more secure software system which is the need of the hour.
The advantages of the proposed method are that, firstly, the method allows to obtain numerical estimates of the security of the embedded software taking into account its qualitative metrics. Secondly, the expert opinion is based on the criteria of evaluating software security formed in accordance with the stages of the product life cycle. Thirdly, this method will allow to take into account both the quality of input information and reliability (degree of confidence) of information from experts. Fourthly, this approach has great potential and will allow to adapt it to the existing models of software quality assessment, as well as to modify it to suit various types of software.

References
[1] Tassey G 2002 The economic impacts of inadequate infrastructure for software testing. RTI Project Number 7007.011, National Institute of Standards and Technology.
[2] Schulz S, Buchenrieder K J and Rozenblit J W 2002 Multilevel testing for design verification of embedded systems. Design & Test of Computers, IEEE 19 60-9
[3] Ebert C, Jones C 2009 Embedded Software: Facts, Figures, and Future IEEE Computer 42 42-52
[4] Howard M, Lipner S 2006 The security development lifecycle (Redmond: Microsoft Press) p 352
[5] IEEE Guide for the use of IEEE standard dictionary of measures to produce reliable software IEEE, New York, IEEE Std. 982.2-1988
[6] IEEE Standard glossary of software engineering terminology IEEE, New York, IEEE Std. 61012-1990 pp 1-84
[7] Platenius M C, Shaker A, Becker M, Huellermeier E and Schaefer W 2017 Imprecise matching of requirements specifications for software services using fuzzy logic IEEE Transactions on Software Engineering 43 8 739-59
[8] Gandotra E, Bansal D and Sofat S 2017 Malware threat assessment using fuzzy logic paradigm. Cybernetics and Systems 48 1 29-48
[9] Zadeh L 1988 Fuzzy Logic Computer 21 4 83-93
[10] Zadeh L 1975 The Concept of a Linguistic Variable and Its Application to Approximate Reasoning Information Sciences 8 3 199-249