NEW METHODS OF MINIMIZING THE ERRORS IN THE SOFTWARE

Dorenskaya E.A., Semenov Y.A.

NRC “Kurchatov Institute” – ITEP, 25 Bolshaya Cheremushkinskaya str., Moscow, 117218, Russia

E-mail: dorenskaya@itep.ru, semenov@itep.ru

The report focuses on the idea of creating a language for the description of the problem, not an algorithm. It also talks on the bank of algorithms. A method for modifying the program code with given input parameters is proposed. The method of task interpreting with a help of special language is described (implemented to simplify regular expression generation). The dialogue method between the computer and the programmer is described as a method of code generation.

Keywords: minimizing the errors, language for the description of the problem, the principle of “many eyes”, meta-language, method of analyzing the context of words and documents, the dialogue method, a method for modifying the program code by the given input parameters, the method of task interpreting with a help of special language, dialog programming, bank of algorithms

© 2018 Elizaveta A. Dorenskaya, Yuri A. Semenov.
1. Introduction

In principal it is just a task for AI, but who can answer the question, – Can AI, created in the image and likeness of human (neural model), provide less bugs than usual programmer [4,5]?

Today the software quality defines almost all sides of our life, including health [3]. That is why it is so important to minimize software errors. Almost all modern programming languages are algorithmic. There is no language to describe a problem. The best result now is 0.5 errors per 1000 code lines for OS Windows. But there are a lot of problems, where such a level of errors is too bad.

2. Main part

Our purpose is to minimize software errors. There is no ways to find and remove all program errors entirely. There is no even effective tool to estimate a real error number in the code [7]. One of the most reliable ways to mitigate error's number is to apply routines by many users with subsequent detection and correction of found errors (the principle of “many eyes”).

An optimal solution of the problem is seemed to be a problem description on natural language (e.g., English or Russian) with subsequent conversion it to the high level programming language routine (Figure 1).

![Figure 1. The proposed scheme for evolution of problem description to executive routine](image)

Therefore, in the process of searching for opportunities to minimize software errors and approaches to creating a meta-language description of the problem (ML), we have developed a method of analyzing the context of words and documents. It was conducted a check of the reliability of context calculation by using the Chebyshev inequality and the simulations using the Monte Carlo method. These studies have shown, that developed method for estimation contextual meanings of words and documents can not be considered universal. However, this algorithm is quite easy to implement and, in most cases, it gives a correct context value. Therefore, it may become an important part of the problem description metalanguage (ML) in the future [2].

As a possible alternative we can consider the special Meta language (ML) for problem description or dialog programming as it is shown on Figure 2.

![Figure 2. Dialog programming](image)

Meta-language ML determines a system behavior at a phase of problem description (phase 1, Figure 2), at this phase we may fix some global variable, e.g. IP-addresses, data exchange protocols, periodicity of requests or measurements and so on), at the phase 2 it is generated an executive routine itself. Besides at the first phase we have to set programming language. At the phase 2 we can start looking for program modules in the bank of algorithms. Here we fix, what modules can be built by computer and what should be written by the programmer himself [1]. Computer from these modules generates an executive routine (see Figure 3).
Figure 3. Structure of ML in case of dialog solution

Figure 4 shows part of the dialog graph for regular expressions generation method. The method has been tested on a routine of regular expression generation (Figure 4).

One more example of using dialog is the routine to prepare and configure the security system for WEB-server (Figure 5).
Figure 5. Diagram of security code generation for WEB-server

Notations on Figure 5:

C0 – case of usage \textbf{put} or \textbf{delete} methods (HTTP-protocol).
C1 – using methods \textbf{post} and so on.
F1 – path traversal attack (in the file access_log there are lines 

\texttt{..//}, \texttt{../../../../}, \%2e\%2e\%2f, \%.\%.0\%.af or \%2e\%2e\%2f/).
G0 - SQL-attack;
G1 – the same, but with no SQL-DB on the computer.
H1 – there is a request of the potentially dangerous file

A method of dialogue has one significant drawback. All versions of dialogue it is difficult to
describe and impossible to predict, because their number tends to infinity. But, nevertheless, this
method can be successfully used to solve simple problems with a limited number of branches of the
dialog graph.

To minimize program errors, you can use the method of modification the appropriate program
code by the specified input parameters. This method is implemented in the charting program
“Gnuplot”. Program chooses which type of chart to build and accordingly modifies the code based on
the given input data.

If in the past any task was solved by combining processor instructions, now we may use
programming modules instead. These modules’s number is one or even more orders greater than
number of processor instructions. The functions of the modules may be modified with the help of
special calculated modificators.

As examples of such modules may be programs for text analyses (e.g., log-files), data to
graphics conversion, statistical analyses routines and so on. These modules should become the main
structure elements from which the final program is built.

The text of problem description on macro language is analyzed at first to find out the problem
realm (dialog computer-programmer). Further the task is divided into subtasks. After that an attempt is
made to find a necessary algorithm in the bank, corresponding to the particular subtask. For each of
found algorithm modules the lists of input and output parameters are determ

ined. For each function, if
its description is not found in the algorithm bank, the programmer develops the code, written
according to Holzmann rules [6].

The closest method to the realization of our purpose is the method of task interpreting with a
help of special language (implemented to simplify the generation of regular expressions code). It is
implemented with the help of Russian words and symbols that describe part of the code of a regular
expression. This simplifies the formation of a regular expression and reduces the number of possible
errors in it. Therefore, this method is still far from ideal. Ideal, as mentioned above, we consider a
method, when a person writes a description text in a natural language, and the program converts it into
a high-level programming language code. This method is applicable for simplified generation of
program code. The described method is intended for use in highly specialized areas. However, it can
be a part of the global macro-language description of the problem.
3. Conclusion

It is clear that AI will make programming more effective, but we are sure - programming error problem is forever. Our developments are small step in proper direction, as people even in AI-era will find the possibility to make errors in their code.

References


