Models and algorithms of testing software on the basis of the basic spesifications

G I Khassenova*, S T Amanzholova, N G Khaimuldin

International University of Information Technologies, Kazakhstan

**Corresponding author's e-mail: khassenova@rambl.ru*

Received 1 May 2015, www.cmnt.lv

Abstract

This article discusses the study of software reliability. Define the concept of software reliability. Examines existing software reliability models and their classification. The main stages of the software life cycle.

Keywords: method of proving the correctness of programs, test method for diagnosis, methods of structured programming, reliability of software, testing, verification, validations

1 Introduction

Software reliability issues typically involved only after the completion of software development. This leads to a chronic imbalance in the operation of applications. The survey revealed that most of the errors made during the design phase. Research in the field of software development have formed a set of methods, processes, technologies, models, the use of which allows to achieve specified performance reliability and quality of software. Research on how to improve the reliability of software, showed that the quality assurance and reliability should be given consideration at all stages of the development process. To improve the reliability of software research include methods such as - a method of proving the correctness of programs, methods of test for diagnosis, as well as methods of structured programming [1].

2 Overview of the study area

The main component of the quality of the software is its reliability. *Software reliability* is defined as the ability of the program to operate smoothly certain period of time under certain conditions. *Model of software reliability* - a detailed, formalized definition of reliability. Nowadays developed hundreds of models of software reliability, taking into account the different types of programs and their applications, but there is no common, universal model, applicable to any program. In general, software reliability models are divided into the following groups: **predicting, measuring and evaluation**. These differences are determined at what stage of the life cycle of the program they are used and for what purposes serve. In its turn predictive models include the following: [2]

1. model of Halstead,
2. model of Motley-Brooks.

Measuring models include the following:

1. model without calculation errors,
2. model with calculation errors.

Estimated models are:

1. Moussa model (selection of the data area),
2. model of error's seeding

Most of the developed models of software reliability are based on various assumptions, which seriously limit the scope of their application. In such a situation, model of based on a probabilistic approach more broadly applicable, but the practical value of their use is not great.

3 Adopting relevant technology

Based on the experience of technical diagnostics, developing a model of software reliability, consisting of the basic model, which is the result of the study of a software system, and the diagnostic model, which is constructed on the basis of the base and focused on the the process of ensuring software reliability.

The process of ensuring software reliability should be applied at all stages of the life cycle of programs. Depending on the stage of implementation such a process will vary.

Considering the stages of the software life cycle in terms of features ensure reliable software product produced by each of them.

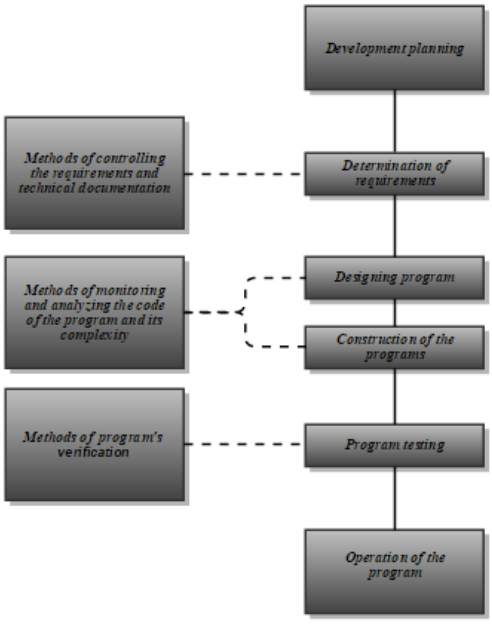

FIGURE 1 The main stages of the software life cycle

Software development process (Figure 1) consists of the following stages:

- planning program development;
- determination of requirements for developing program;
- designing architecture and interfaces;
- construction of (coding) program;
- program testing;
- operation of the program.

Each stage corresponds to its own specific methods and tools for monitoring and ensuring the reliability of the software, as well as the strata of the program description.

Core business of reliability software on the determining step is the *validation* requirements (certification) and *verification* requirements generated.

Core business of reliability software to the software design phase is to *analyze the quality* and *assessment of the program design*.

At the construction stage (coding) software to create a working software product through a combination of coding, verification, unit testing, integration testing, and debugging.

One of the methods of quantify the program code is a code block count code. *Block of code* - is an indivisible sequence of statements executed one after the other.

During the testing phase of software products is subject to various checks compliance behavior of the functions of the working copy of the program specifications as defined in the previous stages of the software life cycle.

4 Testing, verification and validation - differences in terms

Despite the apparent similarity of the terms "*test*", "*verification*" and "*validation*" means different levels of validation work of a software system. To avoid further confusion, clearly define these concepts. (Figure 2)

Software Testing - view of the design associated with the implementation of procedures aimed at the detection of (evidence of) errors (inconsistencies, incompleteness, ambiguity, etc.) in the current definition of the developed software system. The testing process is primarily to verify the correctness of a software implementation of the system, the implementation of compliance requirements, ie, testing - it manages the execution of the program in order to detect inconsistencies of her behavior and requirements.

Software verification - a more general concept than testing. The purpose of verification is to achieve assurance that a verification object (or code requirements) meets the requirements implemented without unintended funktsy and meets design specifications and standards. Verifikatsii process includes inspection, testing, code analysis of test results, the formation and analysis of reports about problems. Thus, it is assumed that the testing process is an integral part of the verification process.

Validation of a software system - a process whose goal is to prove that as a result of the development of the system, we have achieved the goals that were planning to reach thanks to its use. In other words, validation - is to check compliance of the customer's expectations.

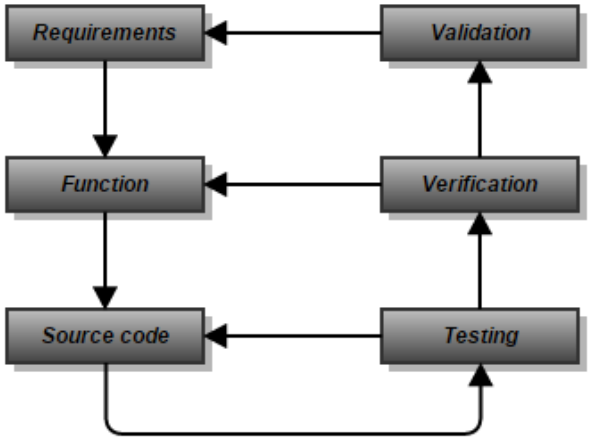

FIGURE 2 Testing, verification and validation

5 Verification

Verification - is the process of determining whether a software tools and components requirements imposed on them in the successive stages of the life cycle of a software system developed.

The main purpose of verification is to confirm that the software meets the requirements. An additional objective is the identification and registration of defects and errors that are made during the development or modification of the program.

Verification is an integral part of the work under the collective development of software systems. In this case, the verification task includes monitoring the results of some developers to transfer them as input to other developers.

To improve the utilization of human resources in the development, verification should be closely integrated with the design, development and maintenance of software systems.

In advance to distinguish between verification and debugging. Both of these processes are aimed at reducing errors in the final software product, but debugging - a process aimed at finding and eliminating errors in the system, and verification - a process aimed at demonstrating the presence of errors and the conditions of their occurrence.

Furthermore, verification - controlled and controllable process. Verification includes an analysis of the causes and consequences of errors that will cause them to fix, planning processes, find errors and their correction, evaluation of the results. All this suggests the verification as a process to ensure a predetermined level of quality in the delivery of a software system.

First, consider the purpose of verification. The main objective of the process - proof that the result meets the design requirements placed thereon. Typically, the verification process is carried out from top to bottom, starting from the general requirements specified in the terms of reference and / or specifications for an information system to all the detailed requirements on the software modules and their interactions. The structure of the tasks of the process includes consistent verification that a software:

- the general requirements for an information system designed for software implementation, correctly processed into high-level requirements specification for complex applications that match the original system requirements;

- high-level requirements correctly processed in software architecture and specification requirements to the functional components of a low level that meet the requirements of a high level;
- specification of requirements for software functional components located between the components of high and low level, meet the requirements of a higher level;
- Software architecture and requirements for low-level components correctly processed in satisfying their source software and information modules;
- source code and the corresponding executable code does not contain errors.

In addition, verification of compliance requirements specification for a specific project software tools are subject to the requirements for technological support lifecycle, as well as requirements for operational and technical documentation.

Purpose software verification are achieved by a combination of sequential execution of inspections of project documentation and analysis of their results, the development of test plans, test and test requirements, test cases and procedures, and follow these procedures. Test scenarios are used to verify internal consistency and completeness of the implementation of the requirements. Run the test procedures shall ensure compliance demonstration test program source requirements.

The choice of effective methods of verification and consistency of their application to the greatest extent influenced by the basic characteristics of the test objects:

- Class set of programs determined by the depth of his connection with the operation of real-time and random effects from the external environment, as well as requirements for the quality of information processing and reliability;
- the complexity or scale (size, dimensions) complex programs and its functional components, is the end result of development;
- prevailing in the program: calculates complex expressions and conversion of measured values or processing logic and character data for the preparation and display solutions.

Defining some concepts and definitions related to the testing process, as part of the verification. Myers gives the following definitions of key terms [3]:

- *Testing* - the process of implementation of the program in order to detect errors.
- *Test data* - inputs that are used to test the system.
- *Test situation (test case)* - Inputs to test the system and the expected outputs, depending on the inputs if the system operates in accordance with its requirements specification.
- *A good test situation* - a situation that has a high probability of detection is still undetected error.
- *A successful test* - a test that detects undetected errors.
- *Error* - programmer action at the design stage, which leads to the fact that the software contains internal defects in the process that the program can lead to incorrect results.
- *Denial* - the unpredictable behavior of the system, leading to unexpected results, which could be caused by defects contained in the system.

Thus, in the process of software testing, as a rule, checking the following:

- To verify that the software meets the requirements for it;
- Verifying that in situations that are not reflected in the requirements, the software behaves adequately, that is not the case of system failure;
- Checking software for common mistakes that make programmers.

6 The life cycle of software development

Collective development, unlike the individual requires careful planning of works and their distribution during the creation of a software system. One way to work is to break the process of developing into separate successive stages, after the passing of which the final product is obtained or a portion thereof. These stages are called software development life cycle of the system. As a rule, the life cycle begins with the formation of a common understanding of the system being developed, and their formalization in the form of a top-level requirements. Completed development life-cycle system start-up. However, it should be understood that the development - just one of the processes associated with the software system, which also has its own life cycle. In contrast to the system development life cycle, life cycle of the system itself ends with the conclusion of its operation and termination of its use.

Software lifecycle - a set of iterative procedures associated with consistent changes in the state of software from the formation of reference to it before the end of its use by the end user.

6.1 THE LIFE CYCLE MODELS

Any stage of the life cycle has a clearly defined start and end criteria. The composition of the phases of the life cycle, as well as the criteria ultimately determine the sequence of stages of the life cycle is determined by a team of developers and / or customer. Currently, there are a few basic life cycle models that can be adapted to real development.

Systems development life cycle (sometimes called a waterfall) is based on a gradual increase in the level of detail describing the whole system being developed. Each increase in the level of detail defines the transition to the next state of development (Figure 3).

V-model of life cycle – as a kind of "work on the bugs" classical cascade model has been applied life cycle model, containing two types of processes - the basic processes of development, similar to the cascade model and verification processes, representing a feedback loop with respect to the basic processes (Figure 4).

Spiral model development system is repeating steps - spirals. Each turn of the spiral - a cascade or V-shaped life cycle. At the end of each turn is obtained a complete version of the system that implements a set of functions. Presented to the user for the next round of transferred all documentation developed at the turn of the previous, and the process repeats.

Thus, the system is developed gradually through constant coordination with the customer. At each turn of the spiral system functionality expands gradually grow to the full.

Life cycle of extreme approach - the maximum shortening of the duration of one stage of the life cycle and close interaction with the customer. In fact, at each stage, the implementation and testing of a system function, which upon completion, the system immediately delivered to the

customer for checking or service.

The main problem with this approach - the interfaces between the modules that implement this feature. If all previous types of life cycle interface is clearly defined at the beginning of development, as are known in advance all the modules, then the extreme approach designed interfaces "on the fly" with the developed modules.

7 Conclusions

Developing basic pattern and diagnostic software has wider application as compared with existing models. This allows their use in the production cycle of any software, regardless of the model used by the software life cycle and software engineering.

Achieving this goal work the following tasks:

- analysis of existing models of software reliability for the possibility of their use in real production processes;
- development of stratified (base) model software;
- development of diagnostic model of software;
- creation of automated tests to verify the proper functioning of its program specifications.

References

[1] Khassenova G I, Amanzholova S T, Khaimuldin N G 2014 Work books of the Third Internatoinal scientific-practical conference "Status, problems and challenges of information in Kazakhstan", *Testing of the information system for software engineering quality. UDK 004-075.8*, 256

[2] Volkov V 2009 Automated system for monitoring and ensuring software reliability [Text] / VG Wolves // *Mathematical modeling. Optimal control: Bulletin of the Nizhny Novgorod University NI Lobachevsky* **5** 173-5

[3] Sinitsyin S V, Nalyutin N Yu 2008 *Verification of software - M: BINOM* 368

Authors

Gulbanu Khassenova, 1948, Almaty, Kazakhstan

Current position, grades: associate professor in International Information Technology University, Almaty
University studies: candidate of technical science at Kazakh National Technical University, Almaty in 1998.

Saule Amanzholova, 1969, Almaty Kazakhstan

Current position, grades: associate professor in Kazakh National Technical University, Almaty
University studies: candidate of technical science at Kazakh National Technical University, Almaty in 2009

Nursultan Khaimuldin, 1991, Almaty, Kazakhstan

Current position, grades: tutor in International Information Technology University, Almaty
University studies: bachelor, currently studying master degree at International Information Technology University, Almaty, 2015
Scientific interest: Bigdata, software engineering, cloud computing, information systems.
Publications: 1