INFORMATION TECHNOLOGY

Information solutions for evaluating the quality of software tests

I. V. Liutenko, A. O. Goloskokova, O. I. Kurasov, D. A. Lukinova

National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
Corresponding author. E-mail: liv@kpi.kharkov.ua, semit.department.16@gmail.com, kurasov.oleksii@gmail.com, dasha.lutenko@gmail.com
Paper received 11.12.19; Accepted for publication 28.12.19.

https://doi.org/10.31174/SEND-NT2019-215VII26-07

Abstract. An approach to evaluating the software tests quality using aggregated quality criteria is proposed. The subject of the study is the formation of a software tests quality evaluation system, which can be used in the software development process. The article gives the full classification of the testing types. The testing process was represented in several stages: testing requirements, test analysis, and the testing process. Tests quality evaluation will improve the testing process, which purpose is to ensure the specified quality of the software being developed.

Keywords: *tests, quality, evaluation, information solution, software development process.*

Introduction. Today, few doubt the feasibility of conducting the testing process of software products being developed. The goal of any testing project is to ensure the quality of the developing product. Quality is defined in ISO 9126 as the set of its characteristics related to the ability to satisfy the expressed or implied needs of all stakeholders.

Evaluation of the software tests quality will provide an opportunity to create such a complex of tests for various purposes, which will allow to control the quality of the software with the least expenses for testing.

Related publications. Software testing is a process to identify errors and determine the correspondence between real and expected behavior of software, which is performed on the basis of a set of tests selected in a specific way. More broadly, software testing is a software quality control technique that involves designing tests, performing tests, and analyzing the results obtained [1].

There are quite a number of different features in the literature that can be used to classify tests and software testing.

The main task of software testing is to obtain information about the readiness status of the declared functionality of the developing system or to obtain information about its quality (customer expectations). Software testing is performed by a team of qualified specialists, but it is not possible to do everything automatically because many stages are performed by experts.

The right approach to testing will allow you to supply the customer with a quality product, but this requires a responsible approach to the organization of testing, design and development of software tests.

The species diversity of tests (testing methods) is quite extensive. Tests can be classified according to various criteria. The general classification of testing methods is shown in Figure 1.

Static testing happens without running the code for execution. Within this testing approach, the following artifacts of the software system are checked:

1) documents (requirements, test cases, application architecture descriptions, database schematics, etc.);
2) graphical prototypes (eg, sketches of the user interface);
3) the code of the software application, which is often executed by the programmers themselves within the framework of code audit (the code of the application can also be checked using testing techniques based on code structures);
4) parameters of the program execution environment;
5) prepared test data.

Validation of these documents prevents serious structural errors and defects of both the application itself and its tests.

Dynamic testing involves running the code to execute to test its actual behavior. Can be run as the code of the whole program as a whole (system testing), and the code of several interrelated parts (integration testing), individual parts (unit or component testing), and even separate sections of code [2].

The classification of the degree of automation is rather superficial because of the individual features of testing by various techniques, so all testing methods can be divided into conventionally manual and automated, despite the fact that an automated test may require manual configuration.

Manual is a test in which the test cases are performed by a person manually without the use of automation tools. Although this sounds very simple, the tester at some point or another requires qualities such as patience, observation, creativity, the ability to perform non-standard experiments, as well as the ability to see and understand what is happening "inside the system", ie to see as external impacts on the application are transformed into its internal processes. The results of such testing are often dependent on the concentration of the tester and may be to some extent undetermined, that is, have different results over a series of experiments under the same conditions.

Automated testing is a set of techniques, approaches and tools that exclude a person from performing some of the tasks in the testing process. Test cases are partially or completely performed by a special tool, but the development of test cases, preparation of data, evaluation of performance, writing reports of defects and other work is performed by a person [3].

Functional testing is a type of testing aimed at verifying the correctness of the functionality of the application (the correct implementation of functional requirements). Often, functional testing is associated with black box testing, but using the white box method, it is quite possible to verify the correct implementation of the functionality.

Non-functional testing is a type of testing aimed at verifying the fulfillment of non-functional software requirements such as performance, reliability, security, portability and usability [4].

Unit testing (module testing) is intended to test individual small parts of a program, which can usually be investigated in isolation from other similar parts. When performing this testing, certain functions or methods of classes, the classes themselves, the interaction of classes, small libraries, and individual parts of the program can be checked. Often, this type of testing is implemented using specialized technologies

and testing automation tools that greatly simplify and accelerate the development of appropriate test cases.

Integration testing is intended to test the interaction between several parts of the application (each of which, in turn, is tested separately during the unit testing phase). Unfortunately, even if we work with very high quality individual components, there are often problems at the junction of their interaction.

System testing is intended to test the entire application as a single unit, made up of the parts tested in the previous two stages. It is possible to detect defects at the joints of components and fully interact with the application from the end-user perspective, using other types of testing [2].

Installation testing - testing that aims to identify defects that affect the progress of the installation (installation) phase of the application. In general, such testing tests many scenarios and aspects of the installer in situations such as:

1) a new runtime in which the application was not previously installed;
2) updating the existing version;
3) changing the current version to more;
4) reinstallation of the program in order to eliminate the problems;
5) restarting the installation after an error, which made it impossible to continue the installation;
6) uninstalling the program;
7) installation of a new application from the application family;
8) automatic installation without user participation.

Regression testing - testing aimed at verifying the fact that in the previously functional functionality there were no errors caused by changes in the application or its operating environment.

Acceptance testing - formalized testing aimed at validating an application from the end-user / customer's point of view and deciding whether the customer accepts the work of the contractor (project team). The following subspecies of acceptance testing can be distinguished.

Production acceptance testing - the design team examines the completeness and quality of the program implementation in terms of its readiness for delivery to the customer.

Operational Acceptance Testing - Operational testing performed in terms of installation performance, application resource consumption, compatibility with software and hardware platforms, etc.

Final acceptance testing - testing of the application by end users (customer representatives) in real-world conditions to determine whether the application needs modifications or can be commissioned in its current form.

Smoke testing is aimed at testing the most critical functionality, the inability of which makes it impossible to use the software system or part of it.

Smoke testing is performed after the release of a new version of the PS to determine the overall quality level of the application and decide on the (in) expediency of performing more detailed testing. Since there are very few test cases at the smoke testing level, and they are quite simple, but often repeated, they are good candidates for automation. Due to the high importance of test cases at this level, the threshold of the metric of their passage is often set equal to 100% or close to 100%.

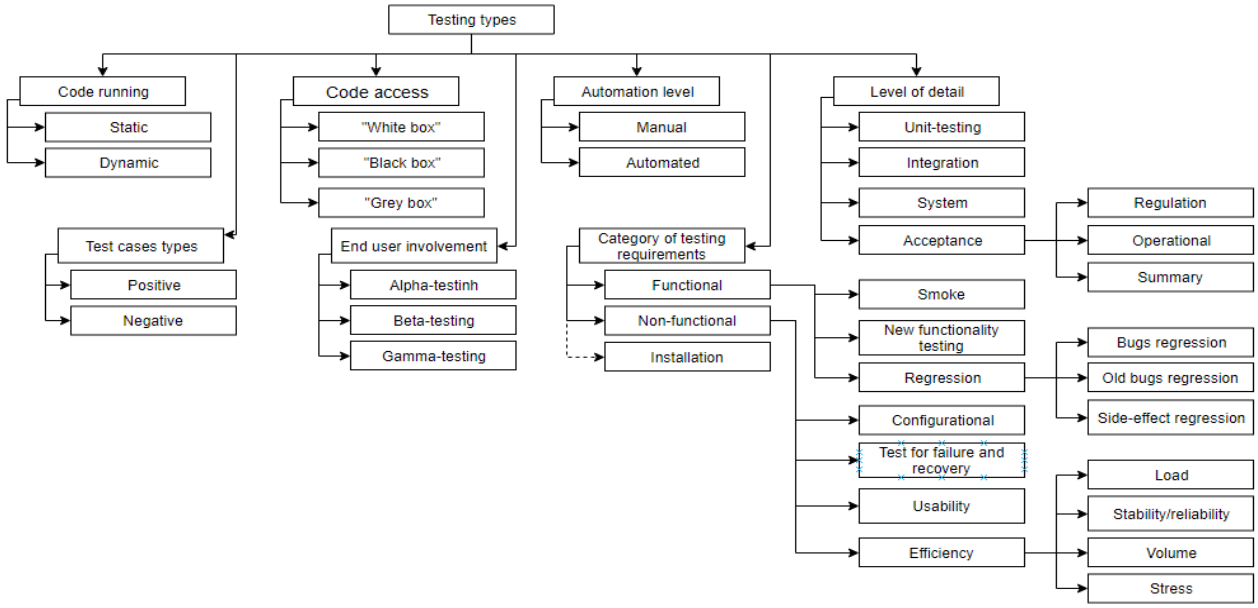

Figure 1 - Classification of testing types

Alpha testing is performed inside the developer organization with the potential partial involvement of end users. May be a form of internal acceptance testing. Some sources state that this testing should be conducted without the involvement of the development team, but other sources do not make such a requirement. The essence of this type is short: the product can already be periodically displayed to external users, but it is still quite "raw", so the basic testing is performed by the developer organization.

Beta testing is performed outside the developer organization with the active involvement of end users / customers. Runs when a product can already be shown openly to external users, it is already quite stable, but the problems can still be, and their detection requires feedback from real users.

Gamma testing is the final stage of pre-product testing aimed at correcting minor defects found in beta testing. As a rule, it is also performed with the maximum involvement of end users and customers. It can be a form of external acceptance testing [5].

The aim of the paper. The purpose of this work is to develop models and information solutions, using which software solutions can be created that will automate the procedure for evaluating the quality of software tests. To do this, you should also consider the testing procedure and build

models that reflect it. It is also superfluous to review the various types of testing that will help you choose the types that are most appropriate to conduct a quality evaluation that may affect the overall quality of the software.

Material and methods. In general, the testing process can be represented in several stages: testing requirements, test analysis, and, directly, the testing process.

The main stages are the test analysis and the process of direct testing (on the diagram - A2 and A3, respectively). In practice, stage A1 (requirements analysis) is not always carried out, however, it is very important, because allows you to find inaccuracies in the requirements. This, in turn, makes it possible to avoid system errors in development and saves time for subsequent testing. A schematic general model of the testing process is presented in Figure 2.

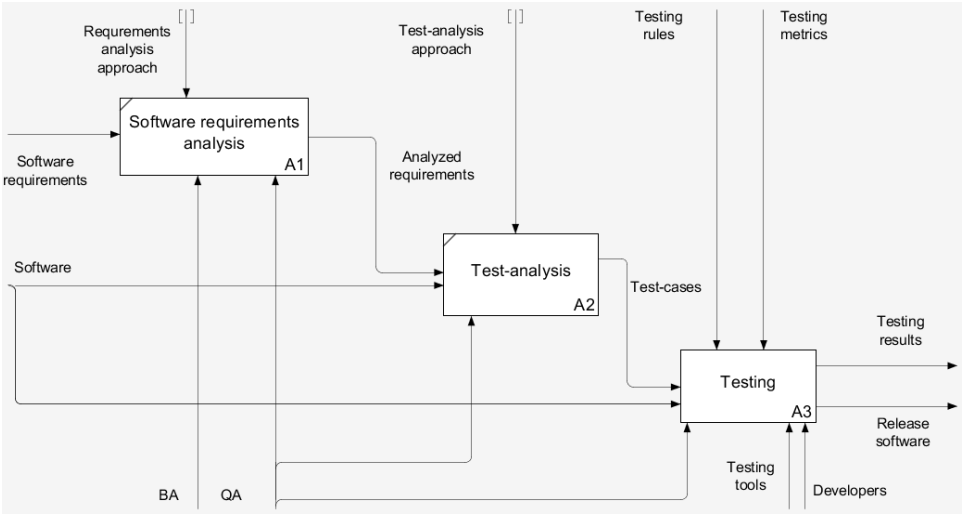

Figure 2 - General testing scheme

The testing process (A3) is illustrated in more detail by the model shown in Figure 3. The testing process includes smoke testing, testing of new functionality and regression testing.

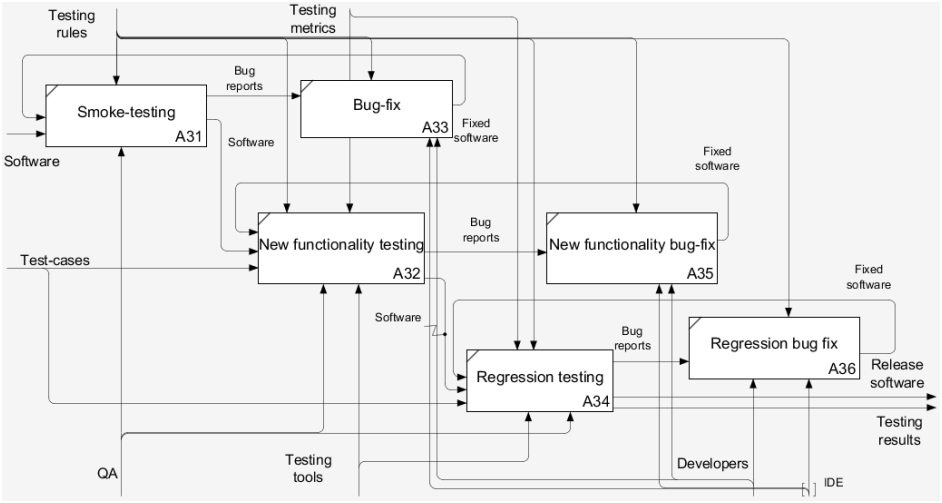

Figure 3 - Detailing the testing process

To assess the quality of tests, it is proposed to use aggregated criteria, this approach was considered in [6].

Results and their discussion. Using the theoretical foundations of this approach, software solutions (or an information system) can be developed that will automate the process of evaluating the quality of tests. For such an information system, business rules were formulated that served as the basis for creating a data model.

Business rules include the following key points:

1) as part of the project, it is possible to make many quality assessments of tests;

2) many employees who are not necessarily involved in this project can take part in the assessment process;

3) each employee has duties according to the specialty, different for each project in which he takes part;

4) a lot of employees with specific specialties can be involved in a project, moreover, many employees with a common specialty can work on one project;

5) evaluation criteria have a hierarchical relationship among themselves in the framework of the evaluation system;

6) each criterion must have at least two gradations;

7) gradations of aggregated criteria represent a set of gradations of hierarchically subordinate criteria;

8) assessment is the specific value of the quality criterion adopted as part of the examination;

9) the examination report contains many evaluations of criteria hierarchically subordinate to each other.

The data model is shown in Figure 4.

For the server side developing, it is proposed to use the Entity Framework Core, ASP.NET Core. The ASP.NET Core platform is a technology that is designed for creation of web applications of all kinds: from small websites to large web portals and web services. With ASP.NET Core, you can create cross-platform applications. Users have the ability to run web applications not only on Windows, but also on

Linux and Mac OS. Entity Framework Core (EF Core) is an object-oriented technology from Microsoft for accessing data. EF Core allows you to work with databases, but represents a higher level of abstraction. Entity Framework Core supports many different database systems, i.e. through EF Core, you can work with any DBMS if it has the right provider.

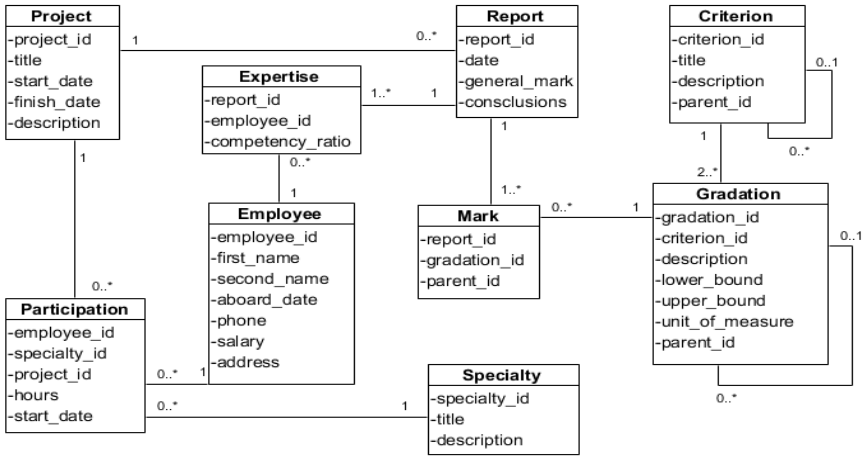

Figure 4 - Data Model

It seems appropriate to use the client-server architecture. Architectural solutions are presented in Figure 5.

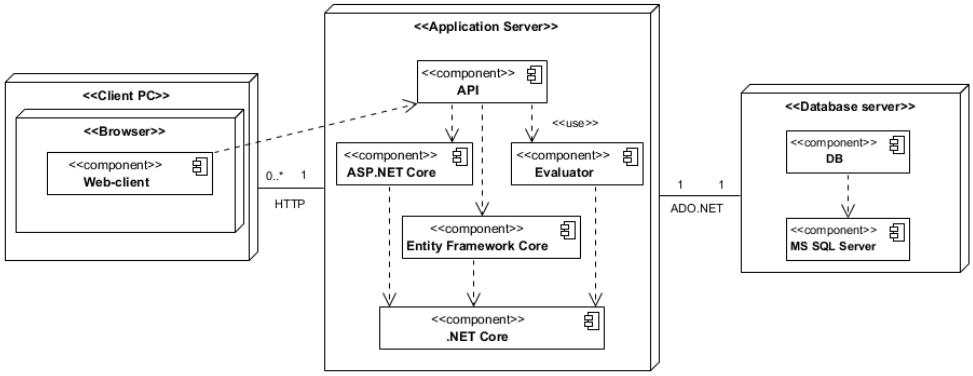

Figure 5 - The proposed system architecture for a system for assessing the quality of software tests

For the client side developing, you can use React.JS and Redux. Redux is one form of an application architecture building on React. Redux is a container for application state management. Redux is not directly tied to React.js and can also be used with other js libraries and frameworks. Also, for the client part creation, the JQuery framework can be used, which is a set of JavaScript functions that focuses on the interaction of JavaScript and HTML.

In this case, the development tools will be Microsoft Visual Studio, Visual Studio Code, SQL Server Management Studio.

Conclusions. Developing an approach for software tests quality evaluation can in the long term improve test results, reduce the time and other resources spent on finding defects in the software system and quickly eliminate the shortcomings of the current testing approach. The obtained results confirm the possibility to use the indicators that can be used to evaluate the overall quality of software tests. These include test performance metrics, test coverage of software capabilities and its software code, namely functions, as well as metrics that make it feasible to use the tests themselves. For these criteria, metrics were formed, the intervals of which were defined as qualitative indicators, which were used to create a hierarchical system of criteria that allows to obtain an integral quality index. Tests quality evaluation will improve the testing process, which purpose is to ensure the specified quality of the software being developed. The testing process was represented in several stages: testing requirements, test analysis, and, directly, the testing process, using the IDEF0 diagram and its decomposition. System architecture of the future software solution was represented in the article. These results can become the basis for the further development of the software solutions for evaluating the quality of the software tests.

REFERENCES

1. Baum T., Liskin O., Niklas K., Schneider K. Factors Influencing Code Review Processes in Industry / Springer, - 2016.
2. ISTQB Glossary - https://www.istqb.org/downloads/glossary.html, 03.02.2019.
3. Prause C., Werner J., Hornig K., Bosecker S., Kuhrmann M. (2017). Is 100% Test Coverage a Reasonable Requirement? Lessons Learned from a Space Software Project. In: PROFES. Springer, 2017.
4. Adams, K.M. Non-functional Requirements in Systems Analysis and Design – Springer, 2015.
5. Lancu L. QA Quality Assurance & Software Testing Fundamentals – Independently published (March 29, 2019) – 197 p.
6. Liutenko I., Kurasov O., Lukinova D., Yershova S., Semanik A. Using the Aggregated Criteria to Evaluate the Software Tests Quality // Bulletin of NTU "KhPI". Series: System analysis, control and information technology. – Kharkov : NTU "KhPI", 2019. – No. 2. – P. 70–75.