Method of dynamic selection of regression tests during developing multimodule information systems in conditions of CI/CD

I.B.Zarubin, A.D.Filinskikh
simarglz@yandex.ru | alexfil@yandex.ru
Nizhny Novgorod State Technical University n. a. R.E. Alexeev

The article considers the forming a pool of regression tests when using the CI/CD process in the development of information systems consisting of a significant number of interacting modules and using various database management systems. The reasons that do not allow using standard filters of testing management systems to account for possible interactions between modules of the developed information system are indicated. The method of selection of the tests to consider the interaction and potential mutual influence of modules on each other, which also minimizes the pool of selected tests, and rank tests for significance from the point of view of a decision on the correctness of the implementation of the functionality of the information system and the system's readiness for its transfer to the customer. The method of dynamic selection of tests that allow to quickly evaluate changes made to the components of the information system in terms of possible negative impact on the unaffected components and functionality is considered. The advantages and disadvantages of the considered methodology, the necessary conditions for its successful application, and ways to implement it both in new projects for the development of information systems and in existing projects in continuous development and without the possibility of organizing code freezing are given.

Key words: *regression testing, automated test cases selection, multicomponent information systems testing, quality assurance testing without code freeze.*

1. Introduction

High competition in the field of information systems (IS) development [1] forces companies that develop IS to reduce the development time and the number of employees involved in the process of creating IS, while trying to avoid reducing the quality of their product. Various software development models are used to reduce development time, such as Agile [2], Iterative model [3], V-model [4], and so on. Almost all IS development companies use methods for implementing the most popular Agile model at the moment. It is necessary to note the growing popularity of the methodology of "Extreme programming" (XP) [5] (one of the implementations of the Agile process) - "continuous integration/continuous delivery" (Continuous integration / Continuous delivery – CI/CD) [6], which allows for regular functional growth of the IS and its rapid updating. This methodology is especially convenient for use in the process of developing IS, which consist of several components (micro-services) [7].

An attempt to reduce the number of employees involved in the development of IS often results to reduce the number of staff in the testing Department (QA) [8], whose tasks, among other things, include checking the IS for bugs and usability [9]. However, in terms of the impact on the commercial success of IS, the role of testing is hard to overestimate – it is unlikely that a potential user will want to use a software product that is riddled with bugs and uncomfortable to use.

There are many different test methods that can be classified by the test object, the test subject, by the positivity of the scenarios, the degree of isolation of the tested component, the degree of automation, by the stage of testing and so on [10]. One of the most important types of testing that is used in almost every IS development project is Regression testing (RT) [11] – a specific activity of QA engineers aimed to detecting bugs in the modules or functionality of the IS that have not been changed, but could be affected by changes in related modules or functionality. Regression can occur both in the interaction between individual IS modules, and in the interaction between two different IS when transmitting any information [12].

Using the Agile development process in conjunction with the CI/CD methodology in conditions of insufficient resources for checking the quality of IS and extremely tight development deadlines may lead to a lack of sufficient time to run some extremely important types of testing. RT is often neglected, since the correct selection of the tests for this type of testing is quite complex and requires a deep understanding of the system architecture, and this type of testing reveals a small number (in absolute numbers) of is errors. In the case of developing a multi-component IS, the choice of scenarios for RT that would check the possible negative consequences of changes is an extremely difficult task. Unfortunately, very often the criterion for selecting tests for RT is the" importance " of the test or a failed test result during the previous RT – it is very easy to select tests based on these criteria, but they do not allow you to select the correct set of scenarios that can effectively detect errors in the operation of IS modules or functions that have not been modified.

The "code freeze" procedure is very important from the point of view of detecting errors in IS [13]. During this procedure, the IP code is not modified for some time – to check the entire IS, rather than any part of it, to detect and localize non-obvious bugs and bugs with complex playback scenarios that affect several components of the IS. It usually takes a significant amount of time to check the entire IS – from a week to a month (depending, of course, on the complexity of the IP and the number of test engineers involved in testing process), and the set of RT scenarios does not change. It is obvious that running a set of tests in parallel with the modification of the is code will lead to the loss of relevance of the test results after only one or two changes have been made. In other words, the results of tests performed at the beginning of a specific test procedure refer to an already outdated version of the IS and do not provide an answer about the quality of the current IS configuration. In practice, the "code freeze" procedure is often neglected during the IS development process due to planning errors and a lack of understanding

of the need to perform all the necessary quality control procedures before transferring the IS to the customer.

This situation in multicomponent systems is complicated by the presence of a significant number of paths and rules for interaction between components – in such cases, it is very difficult to take into account all possible influences on the information system in the development process and, as a result, avoid negative effects on components that are not directly affected by code changes.

Based on above, there is an urgent need to develop a methodology that would allow you to quickly create a list of scenarios for RT, as well as take into account the changes made to the IS components and an arbitrary number of criteria for selection.

2. Methods for selecting scenarios for regression testing using filters

In the context of continuous development of new functionality and new components in complex multicomponent IS, it is necessary to be able to select tests that would show regression (bugs occurrence) in the components of the system that have not been modified. The selection of tests should be made in a short time and, preferred, should not require special knowledge and a deep understanding of the system`s architecture. For some methods of selection, it is sufficient to filter tests using the tools built into the test management system. For example, it is possible to quickly select high-priority tests (for example, one of the most popular test and defect management systems - Jira) (Fig. 1).

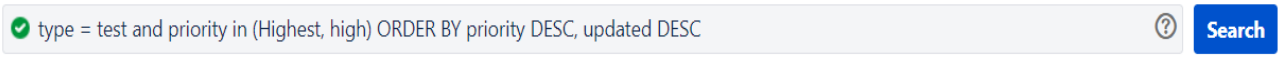

Fig. 1. High priority test cases selection

In the same way, it is possible to select test cases which are developed during a certain time period (Fig. 2) or test cases developed to verify one of the components of a multicomponent IS (Fig. 3).

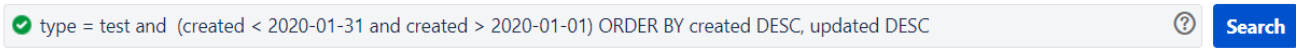

Fig. 2. Creation period test cases selection

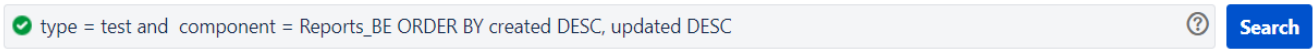

Fig. 3. Component test cases selection

Depending on the specifics of the project, it is possible to select test cases using one of the following parameters:

- by success of the previous test run;
- by the number of bugs found during the test execution;
- by priority of bugs found during the test execution;
- by duration of the test case;
- by the employee who completed the test;
- by test case developer;
- by finding the test in a specific test Suite;

and so on.

Obviously, it is possible to combine test selection parameters during filtering and even determine the order of these tests according to certain rules (Fig. 4).

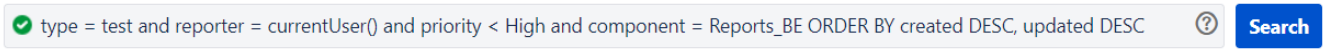

Fig. 4. Developer, priority and component test cases selection

Selection by filters is very convenient, but it has some very significant limitations:

1. It is not pfossible to select tests that would test functionality that indirectly depends on changes made to the IS, for example, cases of changing the format of a message transmitted from one module of a multicomponent system to another;
2. It is not Possible to organize the selection and ranking of test cases based on the weight coefficients of various selection methods;
3. It is not possible to quickly create a list of test cases that would track changes in the system over a certain period – sprint, week, or day.

3. Selection and ranking of test cases depending on the degree of potential impact of the changes

The IS development process is usually organized in such a way as to reduce the time and other resources spent on it. At the same time, there may be situations when savings are made by escaping some important procedures from the point of view of confirming the quality of IS. RT is a complex and time-consuming procedure – that is why in the process of developing complex multicomponent IS in a tight time frame, regression testing is performed only if resources are available and the selection of tests for RT is carried out using extremely simplified methods – based on the importance of the test or the results of the previous RT. This approach has a high risk of losing the quality of components that have not been explicitly changed and, consequently, have not passed the standard checks that has new or updated functions and modules. It is needed to develop the method that would allow you to take tests that would allow to consider the interaction of components and modules would not require comprehensive knowledge of the test IS and would use any number of methods of selection of tests for RT.

As an example, let's consider a simplified scheme of a multi-module information system (Fig. 5).

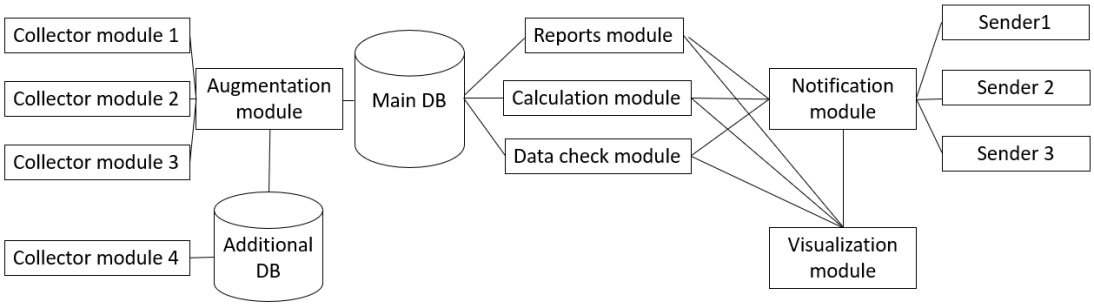

Fig. 5. IS components interactions scheme

Real-time data collection modules collect information from third-party systems or databases and transmit the collected information to the enrichment module. The augmentation module supplements the received data with additional information from the additional database (which, in turn, is supplemented by a separate data collection module) and transmits the enriched data to the main database.

Data from the main database is processed by various modules – for generating various types of reports, graphs and diagrams, and performing various transformations of data from the database. These modules can transmit processed data to the notification module for further conversion and transfer to the modules for sending data to third-party systems.

The visualization module can receive various types of information from all processing modules to display it to users in real time.

Let's imagine the interaction of modules of the presented is in the form of a graph [14] (Fig. 6).

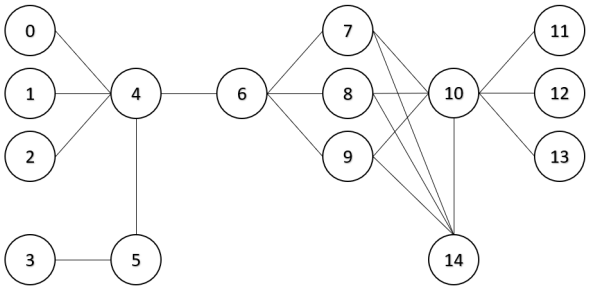

Fig. 6. Graf form IS components interactions scheme

Thus, it is obvious that the changes made to the calculation module (the vertex of graph 8) indirectly affect modules 6, 10, and 14 (the main database, notification module, and visualization module), and, accordingly, it makes sense to check the correctness of these modules in interaction with the updated module 8.

Moreover, it could not be exclude the indirect impact of changes – not on the adjacent module, but on the module that interacts with the adjacent module – changes made to the calculation module 8 may disrupt the correct operation of the sending modules 11, 12 and 13. At the level of unit and integration tests-selection and even the creation of the necessary test cases, can be done by automated systems that analyze the source code, but the selection of tests of high level, such as system and regression test cases, that constitute the sequence of steps in human language, is a very difficult task, which requires very deep knowledge in system architecture and the structure and content of the test scripts.

To solve this problem, it is suggested to entering information about adjacent modules in the description of the module verification test, changes in which may cause the need to run this test. This allows to select not only all the necessary tests for related modules, but also to check the correct operation of modules that are not directly affected by the change. Such effect can be secondary, tertiary, and so on, depending on the degree of distance from the module with the changes made. However, considering all levels of relationships will lead to the selection of all possible test cases, which clearly does not make sense in the context of the RT selection task.

It is obvious that if there is filtering of incoming and outgoing data from modules, the influence of changes made in any module on adjacent modules is on average more pronounced than the indirect influence of different levels [15]. Thus, it is possible to rank the importance of the RT of different modules (Fig.7).

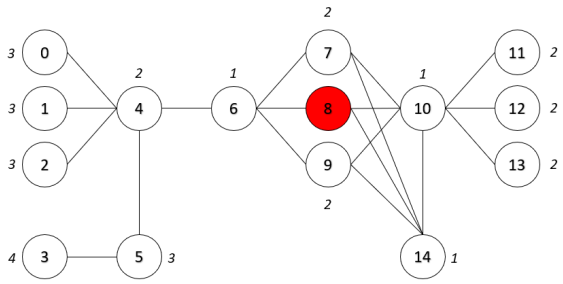

Fig. 7. The level of influence of changes made on related modules

Based on Fig. 7, it is obvious that it is necessary first to perform RT on modules 6, 10 and 14, and second of all-on modules 4, 7, 9, 11, 12, 13 etc.

When evaluating the possible impact of the changes on adjacent modules, it is also necessary to consider that in modern multi-module IS, not all interactions between modules are two-way. Let's look at an example of such an IS (Fig. 8).

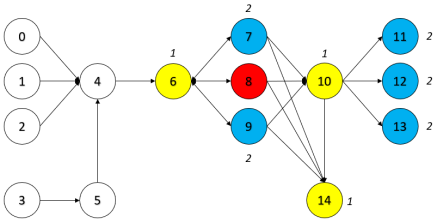

Fig. 8. The level of impact of changes made on adjacent modules, taking into account data flows

Based on the direction of data flows in Fig. 8, it is obvious that changes in module 8 may affect only some of the IP modules (highlighted in yellow and blue) – therefore, the number of RT to assess the impact of changes on adjacent modules will be significantly reduced.

For convenience in further research, we will introduce a numerical measure of the significance of the regression test of an adjacent component in terms of checking the correctness of the is operation – the significance of the test – ST. For simplicity of calculations, we assume that the ST of modules that directly interact with the updated module is equal to 100, tests of a module that interacts with the updated module through one intermediate module will have a ST of 50; through two modules - 25, and so on. In this way, we can rank tests and perform first those tests that are most likely to detect errors in the operation of IS modules introduced by the update.

In the process of developing modern multi-module IS, work on updating, correcting, refining the code is usually performed on several modules at once. Let us consider the situation of RT selection if two modules in the IS are changed simultaneously (Fig. 9).

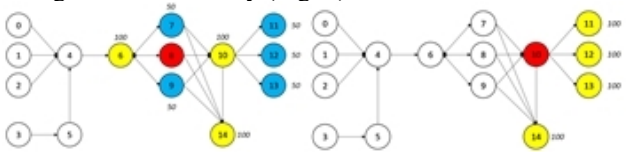

Fig. 9. Distribution of ST for changing two modules at the same time

The total ST value for changing two modules simultaneously is shown in fig. 10.

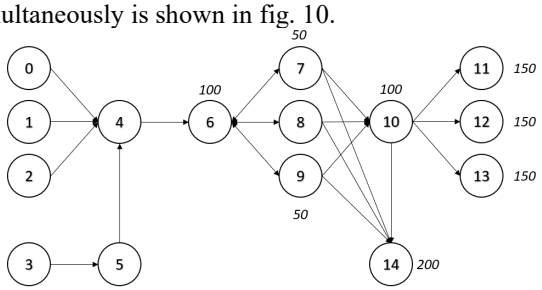

Fig. 10. The total value of the ST including changes to two modules at the same time

It is obvious that the changes made to modules 8 and 10 have the greatest potential impact on module 14. A ST value of 200 indicates the need to perform RT for module 14 with the highest priority. Next, you need to perform RT for modules 11, 12 and 13 (ST 150), then for modules 10 and 6 (ST 100), and finally for modules 7 and 9 (ST 50).

Thus, using this method, it is possible to determine the priorities of regression tests in a multi-module IS in a situation when changes were made to several modules simultaneously.

4. Selection and ranking of test scenarios using an arbitrary number of selection methods and the value of the ST

Currently, there are a significant number of methods for assessing the degree of is regression [16], RT selection, which are based on various principles for determining the need to run the test. The use of multiple methods of selection RT at the same time facilitates the identification of RT, which are able to detect bugs in the IS. Let's look at how to rank test scenarios selected using various methods together, taking into account the degree of potential impact of updated components. Let's assume that high-priority tests are selected, tests that were performed unsuccessfully in the previous RT, and tests that are able to detect bugs made to the system by updating an adjacent module.

$$I_a = \sum (I_p + I_f + I_c) \quad (1)$$

where: I_a – sum of ST; I_p – ST by priority; I_f – ST by success of the previous RT run; I_c – ST considering the potential impact of updated related modules.

For an arbitrary number of selection methods (1) is converted as follows:

$$I_a = \sum (I_1 + I_2 + \cdots + I_n) \quad (2)$$

where: I_a – sum of ST; I_1 – ST of the test cases selected by method 1; I_2 – ST of the test cases selected by method 2; I_n – ST of the test cases selected by method n.

Using (2), it is possible to get the significance of the test in terms of the possibility of detecting errors in the IS – the more importantly test case received higher value of the ST. Thus, we get a test queue ranked by the priority of test execution.

Separately, it should be noted that (2) is correct for the ST values of each selection method in a certain general range, for example, from 0 to 1 or from 0 to 100. If this condition cannot be met for various reasons, it is suggested to use additional rationing.

$$I_a = \frac{w_1 \times I_1 + w_2 \times I_2 + \cdots + w_n \times I_n}{w_1 + w_2 + \cdots + w_n} \quad (3)$$

где: I_a – sum of 3T; w_1 – weight coefficient for group of the test cases 1; I_1 – ST of the test cases selected by method 1; w_2 – weight coefficient for group of the test cases 2; I_2 – ST of the test cases selected by method 2; w_n – weight coefficient for group of the test cases n; I_n – ST of the test cases selected by method n;

5. Using the method of calculating the total significance of the test for the rapid creation of a list of RT

In case of use the practice of continuous development and issuance of new versions of IS modules (CI/CD), there is an urgent need to take into account the changes made to the IS when selecting tests as often as possible. The method of selecting regression tests presented above allows quickly select the most significant scenarios without a deep dive into the specifics of IS. For example, daily generating a list of tests for each tester that will allows to validate and detect bugs in the IS, given as the previous day's code changes, the results of the test execution for a certain period, etc. and thus reduce the negative impact of absence in the life cycle [17] development is "code freeze».

This method can be integrated into test management programs using the API and basic constructs of popular programming languages (Fig. 11).

Fig. 11. Form for setting parameters and weights for selecting regression tests

At the result of the calculating by application, we get a list of Regression tests sorted by total ST (table 1).

Table 1. List of the regression test cases

Test case ID	ST
SQM-1285	385
SQM-1452	385
SQM-1589	365
SQM-1698	365
SQM-1455	365
SQM-1889	265
SQM-2036	265
SQM-1845	245
SQM-1721	245
SQM-1805	225

Separately, it should be noted that the above method is also suitable for determining the priority of running automated module and integration tests when it is necessary to select autotests to run, for example, in the case of a high duration of test execution.

6. Conclusions

The presented method of selecting regression tests based on the sum of significance of tests has the following advantages:

1. Takes into account the potential impact of changes made to related modules;
2. Takes into account any number of selection methods;
3. Allows rapid selection of tests for RT;
4. Can be integrated into test management frameworks;
5. Can be used in automated testing systems.

Acknowledgments

This work was completed and published with financial support from the Russian Foundation for Basic Research, grant 19-07-00926.

References

[1] Kuznetsov S.D. Great Russian encyclopedia. Electronic resource: https://bigenc.ru/technology_and_technique/text/3444940.

[2] McHugh, Martin; McCaffery, Fergal; Coady, Garret (4 November 2014). Mitasiunas, Antanas; Rout, Terry; O'Connor, Rory V.; et al. (eds.). An Agile Implementation within a Medical Device Software Organization. Software Process Improvement and Capability Determination. Communications in Computer and Information Science. 477. pp. 190–201. doi:10.1007/978-3-319-13036-1_17. ISBN 978-3-319-13035-4.

[3] Larman C., Basili V.R. Iterative and Incremental Development: A Brief History // Computer. - 2003. – Vol.36, № 6. – P.47-56.

[4] Staroletov S. M. Fundamentals of software testing and verification. Textbook. Publishing house «Lan» 2018 p. 344. (pp. 16-18). ISBN 978-5-8114-3041-3.

[5] Kent Back Extreme programming. SPb.: Publishing house «Piter» 2017 p. 294 ISBN 978-5-496-02570-6.

[6] Pol M. Duval, Stiven M. Matias III, Andrey Glover. Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series). - Wiliams, 2008. p. 240 - ISBN 978-5-8459-1408-8.

[7] Balalaie, A.; Heydarnoori, A.; Jamshidi, P. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture (англ.) // IEEE Software: journal. - 2016. - 1 May (vol. 33, no. 3). - P. 42-52. - ISSN 0740-7459. - DOI: 10.1109/MS.2016.64

[8] Glenford Myers, Tom Budget, Cory Sandler. The art of software testing. Third edition. 2019. p. 272. (p. 20-21) ISBN 978-5-907144-37-8

[9] Filinskikh A.D., Guliaeva U.I. Analysis of opportunities for implementing cross-platform mobile app development. JSON markup language. KOGRAF-2019. Collection of materials of the 29th all-Russian scientific and practical conference on graphic information technologies and systems. 2019. Publishing house: Nizhny Novgorod state technical University named after R. E. Alekseev (Nizhny Novgorod).

[10] R. Saving. Testing dom com or Пособие по жестокому обращении с багами в интернет-стартапах. 2017. p. 312. ISBN 978-5-4485-4551-1.

[11] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: A survey. King`s College London - 2007. - c. 60.

[12] Filinskikh A.D., The information metric is the transmission and recovery of geometric models in professional software environments, thesis of candidate of technical Sciences, 05.13.17 - Theoretical foundations of computer science, protected 26.12.13, approved 23.06.14 – Nizhny Novgorod, 2013 – 180 p.

[13] Pete Goodliffe. chapter 22: The curious case of the frozen code // Becoming a Better Programmer: A Handbook for People Who Care About Code. - "O'Reilly Media, Inc.", 2014-10-03. - C. 195 - 203. - 362 c. - ISBN 9781491905586.

[14] Volchenskaya T.V., Kniazkov V.S. Computer mathematics: Part 2. Graph teothry / Textbook. Penza: Publishong house of Penza university, 2002. 101 p.
[15] Lipaev V. V. Quality of the software. – M.: Finance and statistics, 1983. – 263 p.
[16] Zarubin I.B., Filinskikh A.D. Method of estimation of completeness of regression testing with normalization by weight coefficients. // T 78 Proceedings of the NSTU named after R. E. Alekseev / NSTU named after R. E. Alekseev. - Nizhny Novgorod, 2019. №4 (127). - 204 p. (9-17).
[17] National standard of the Russian Federation GOST R 56136-2014 life cycle Management of military products. Terms and definitions. Reissue 11.2016 p. 20 (p. 5-6).

About the authors

Zarubin Ilia B. – senior lecturer Nizhny Novgorod State Technical University n.a. R.E. Alekseev, e-mail: simarglz@yandex.ru.

Filinskikh Aleksandr D., Ph.D. in Technology, Associate Professor, Nizhny Novgorod State Technical University n.a. R.E. Alekseev. E-mail: alexfil@yandex.ru