International Science Group

ISG-KONF.COM

V
INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE "TRENDS IN SCIENCE AND PRACTICE OF TODAY"

Ankara, Turkey
October 19-22, 2021

ISBN 978-1-63972-065-1

DOI 10.46299/ISG.2021.II.V

TRENDS IN SCIENCE AND PRACTICE OF TODAY

Abstracts of V International Scientific and Practical Conference

Ankara, Turkey
October 19 – 22, 2021

Library of Congress Cataloging-in-Publication Data

UDC 01.1
The V International Science Conference «Trends in science and practice of today», October 19 – 22, 2021, Ankara, Turkey. 478 p.

ISBN - 978-1-63972-065-1
DOI - 10.46299/ISG.2021.II.V

EDITORIAL BOARD

RESEARCH RESULTS OF FUNCTIONAL, WHITE BOX AND SMOKE TESTING METHODS FOR MOBILE APPLICATIONS

Tvoroshenko Iryna,
Ph.D., Associate Professor
Kharkiv National University of Radio Electronics

Kuznetsov Maksym,
Master in informatics
Kharkiv National University of Radio Electronics

Information technologies had evolved in almost all areas of human activity. Due to automation, the speed of information processing increases, which consequently increases productivity. The process of modeling of any information system allows to automate its management, increases the speed of information transfer, and improves quality control. The simulated system has a self-tuning character, i.e. there is an unambiguous algorithm of actions, where each byte of information is sent to the appropriate data center. Changing the environment will cause a chain of changes in the system of business organization. Technically, modeling consumes as much time as creating a business plan. In the work course, it is necessary to return again and again to already registered moments, to make changes and to build everything anew, and then to test and improve the received IT product.

The requirements are the starting point for determining what the project team will model, implement, and test. If there is uncertainty in the requirements [1-7], then not what was expected will be realized, i.e. people's work will be done in vain.

Mobile applications are especially in the need in-depth modeling of testing processes, because one toolkit can be installed on different devices with different operating systems, software shells, diagonals, and so on.

All types of software testing, depending on the goals, can be divided into the following groups:

– Functional;
– Non-functional;
– Related to changes.

Functional tests are based on functions and features, as well as interactions with other systems, and can be present at all levels of testing [8]: Component or Unit Testing, Integration Testing, System Testing and Acceptance Testing. Typically, these features are described in requirements, Functional Specifications, or Use Cases.

Functional types of testing consider the external behavior of the system and are based on an analysis of the specifications of the functionality of the component or the system as a whole. Functionality Testing can be performed in the two aspects:

– Requirements;
– Business processes.

Testing in “requirement” perspective uses the specification of functional requirements for the system as a basis for the design of Test Cases.

In this case, it is necessary to make a list of what will be tested and what is not, to prioritize requirements based on risks (if this is not done in the document with requirements), based on this to prioritize test scenarios (Test Cases). This will allow focusing and not missing the most important functionality when testing [9].

Business Process Testing uses the knowledge of these business processes, which describe the scenarios of daily use of the system, Test Scripts, as a rule, are based on cases of use of the system (Use Cases).

The advantages of Functional Testing include “simulation of the actual use of the system”, the disadvantages – the possibility of omitting logical errors in the software and the likelihood of over-testing. Consider the program code of an example of Functional Testing: it is necessary to check that the user can create a profile, give it a name, and see the created profile, according to the requirements of the product [10].

Listing1. Functional Testing script using program code:

```
LoginActions.login(self, creds)
HomeScreenActions(self).tap_settings_menu_button()
settings_menu = SettingsMenu(self)
settings_menu.tap_change_profile_button()
# Validate profiles page
self.validate_manage_profiles_page()
self.tap_add_profile_button()
edit_profile = EditProfilePage(self)
edit_profile.enter_nickname(self.NEW_PROFILE)
edit_profile.tap_save_changes_button()
# Validate that profile is added
self.wait_for_load()
self.validate_profiles_after_adding(self, profiles_before, self. NEW_PROFILE)
```

White Box Testing is a test of the internal structure, design, and coding of a software solution. In this type of testing, the code is visible to the tester. The main attention is paid to checking the flow of input and output data through the application, improving the design, ease of use, enhanced security. White Box Testing is also known as Clear Box Testing, Open Box Testing, Structural Testing, Transparent Box Testing, Code-Based Testing, and Glass Box Testing [11].

White Box Testing is one of the two parts of Box Testing's approach to software testing. Black Box Testing involves testing from an external or end user perspective. White Box Testing is based on the internal workings of the program and revolves around internal testing. The term “White Box” was used because of the concept of a transparent box, which symbolizes the ability to see through the outer shell of the software (or box) its internal work. The “Black Box” symbolizes the inability to see the internal workings of the software, so only the end-user experience can be tested.

White Box Testing includes testing software code for the following:

– Internal safety defects;

– Broken or poorly structured paths in coding processes;
– Functionality of conditional cycles;
– Testing of each operator, object, and function on an individual basis.

Testing can be performed at the system, integration, and modular levels of software development. One of the main purposes of White Box Testing is to verify the workflow for the application. White Box Testing involves testing several predetermined inputs against expected or desired outputs. If a particular data entry does not lead to the expected output signal, it indicates a system defect [12].

The first thing a tester often does is learn and delve into the source code of the program. Because White Box Testing involves testing the internal operation of the program, the tester must be very familiar with the programming languages used in the test applications. In addition, the tester must be well versed in secure coding techniques. Security is often one of the main tasks of software testing. The tester must be able to detect security issues, prevent attacks by hackers and naive users who may enter malicious code into the application both consciously and unconsciously.

The second step of testing the "White Box" involves testing the source code of the program for proper functioning and structure. One way is to write more code to verify the source code of the program. The tester will develop small tests for each process or series of processes in the application [13].

The concept of "smoke" testing came from the results of engineering experiments: "When commissioning new equipment ("iron"), it was considered that the test was successful if the installation did not leave the smoke".

In the software field, Smoke Testing is considered as a short cycle of tests performed to confirm that after assembling the code (new or edited); the application starts and performs its basic functions [14].

The conclusion about the efficiency of the main functions is made on the basis of the results of surface testing of the most important modules of the program for the ability to perform the necessary tasks and the presence of critical and blocking defects. In the absence of such defects, the smoke test is declared passed, and the application is transmitted for a full test cycle, otherwise, the smoke test is declared failed, and the application is finalized.

Analogs of Smoke Testing are Build Verification Testing and Acceptance Testing, performed at the functional level by a team of testers, based on the results of which it is concluded whether or not this version of the software is accepted for testing, operation or transfer to the customer [15]. Consider the program code of the example of Smoke Testing: it is required to check the functionality of the login to the application. This is one of the most important functions, its errors can lead to large losses for the correction of defects, if it becomes known too late [16, 17].

Listing 2. Smoke Testing script using software code:

```
home_page = HomePage(self)
home_page.tap_login_button()
subscription_page = SubscriptionPage(self)
subscription_page.wait_for_load()
subscription_page.type_email(self.creds.login)
```

subscription_page.type_password(self.creds.password)
subscription_page.tap_login_button()
assertFalse(home_page.is_login_button_displayed())

Modern development methodologies practice the Continuous Integration approach, which is part of the software product assembly [18].

Assembly is not always having a proper quality; it may contain defects in functionality that is critical to the business, so the inspection should be carried out immediately after the assembly and before submission for testing. This reduces the time spent on testing the assembly that contains blocking defects.

Key benefits of Smoke Testing:

– Detection of critical errors in the first few hours (minutes) after installation;
– Reduction of risks of removal of a low-quality product;
– Minimization of risks in system integration;
– Reduction of defect correction costs;
– Acceleration of verification due to automation.

Therefore, testing plays an important role in the development of quality software.

References:

1. Daradkeh Y.I., Tvoroshenko I., Gorokhovatskyi V., Latiff L.A., and Ahmad N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.

2. Кучеренко Є.І., Творошенко І.С. (2011) Оперативне оцінювання простору станів складних розподілених об’єктів з використанням нечіткої інтервальної логіки. *Штучний інтелект*, 3, С. 382-387.

3. Кучеренко Е.И., Корниловский А.В., Творошенко И.С. (2010) О методах настройки функций принадлежности в нечетких системах. *Системы управления, навигации и связи*, 1(13), С. 94-98.

4. Кучеренко Е.И., Творошенко И.С. (2003) Процессы принятия решений в сложных системах на основе нечетких интервальных представлений. *Вісник Національного технічного університету «ХПІ». Тематичний випуск: Системний аналіз, управління та інформаційні технології.* Х.: НТУ «ХПІ», 1(7), С. 79-86.

5. Творошенко И.С., Дехтярь А.П. (2005) Информационные технологии в задачах компьютерной диагностики с использованием интеллектуальных систем, *Клиническая информатика и Телемедицина. Компьютерная Медицина-2005*, С. 138.

6. Кучеренко Е.И., Творошенко И.С. (2010) Прикладные аспекты моделирования нечетких процессов в сложных системах, *Збірник наукових праць Харківського університету Повітряних сил*, 1, С. 127-131.

7. Бодянский Е.В., Кучеренко Е.И., Творошенко И.С. (2004) О синтезе нечетких алгоритмов на основе композиции фрагментов правил и моделей, *АСУ и приборы автоматики*, 128, С. 19-28.

8. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2021) Methods of Classification of Images on the Basis of the Values

of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

9. M. Ayaz Ahmad, Volodymyr Gorokhovatskyi, Iryna Tvoroshenko, Nataliia Vlasenko, Syed Khalid Mustafa (2021) The Research of Image Classification Methods Based on the Introducing Cluster Representation Parameters for the Structural Description, *International Journal of Engineering Trends and Technology*, 69(10), pp. 186-192.

10. Gorokhovatskyi V., Rusakova N., and Tvoroshenko I. (2020) The application of image analysis methods and predicate logic in applied problems of magnetic monitoring, *Telecommunications and Radio Engineering*, 79(20), pp. 1801-1811.

11. Творошенко І.С. (2021) Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ, 120 с.

12. M. Ayaz Ahmad, Irina Tvoroshenko, Jalal Hasan Baker, Liubov Kochura, and Vyacheslav Lyashenko (2020) Interactive Geoinformation Three-Dimensional Model of a Landscape Park Using Geoinformatics Tools, *International Journal on Advanced Science, Engineering and Information Technology*, 10(5), pp. 2005-2013.

13. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.

14. Кобилін О.А., Творошенко І.С. (2021) Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 124 с.

15. Творошенко І.С., Шевченко А.Р. (2018) Удосконалення просторової мережі навчальних закладів міста Сєвєродонецька на основі геоінформаційного аналізу, *Системи обробки інформації*, 1, С. 46-52.

16. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, *Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice»* (October 12-16, 2020). Stockholm, Sweden, pp. 443-448.

17. Tvoroshenko I., and Dziubenko M. (2020) Modern methods of analysis of the movement scheme using video detection of vehicles, *Abstracts of V International Scientific and Practical Conference «Study of modern problems of civilization»* (October 19-23, 2020). Oslo, Norway, pp. 422-428.

18. Gadetska S.V., and Gorokhovatsky V.A. (2018) Statistical Measures for Computation of the Image Relevance of Visual Objects in the Structural Image Classification Methods, *Telecommunications and Radio Engineering*, 77(12), pp. 1041-1053.