https://orcid.org/0000-0002-5706-6778
https://orcid.org/0000-0002-7371-5651
https://orcid.org/0000-0001-6155-5913
https://orcid.org/0000-0002-0106-6410

UDC 004.54; 004.77

J.N. DAVIES[*], M.V. VEROVKO[**], A.S. POSADSKA[***], I.V. SOLOMAKHA[****]

USAGE OF SIMULATION FOR QA IN REAL-TIME NETWORK ENVIRONMENTS

[*]Creative and Applied Research for the Digital Society (CARDS), Glyndŵr University, Wrexham, UK
[**]AgileVision sp. z o.o., Krakow, Poland
[***]Andersen Ltd., Chernihiv, Ukraine
[****]Chernihiv National University of Technology, Chernihiv, Ukraine

Анотація. *Стандарти забезпечення якості проектування і розробки ISO 9000 широко застосовуються при роботі із програмним забезпеченням. Більшість сучасних компаній прагнуть отримати акредитацію за ISO 9001, щоб поліпшити свої показники. У той же час для деяких галузей дотримання зазначених стандартів є необхідним. Багато компаній у даний час покладаються на інформаційні системи для ведення бізнесу, що стало причиною того, що для отримання акредитації в цих областях IT-компаніями додаються величезні зусилля. Використання програмного забезпечення і комп'ютерних мереж практично в усіх сферах сучасного життя значно підвищує важливість їх стабільної роботи і продуктивності загалом. Процес забезпечення якості QA є потужним інструментом, що допомагає визначити й усунути можливі збої інформаційної системи на етапі її розробки. Однак забезпечення високого рівня якості вимагає урахування впливу всіх компонентів системи. Дане дослідження присвячене питанням тестування програмних продуктів у режимі реального часу з урахуванням різної продуктивності комп'ютерних мереж, що дозволяє емулювати умови, максимально близькі до роботи кінцевих користувачів. Також виконано аналіз доцільності економічних витрат на процеси забезпечення якості. Досліджено алгоритми взаємодії користувачів із діючими системами реального часу при різних показниках продуктивності мережі з метою визначення найбільш частих дій користувача, які викликають збої програмного забезпечення, і можливого запобігання наслідків цих дій під час процесу тестування. Авторами запропоновано загальні підходи до тестування продуктивності і зручності використання системи в разі постійного погіршення роботи мережі.*

Ключові слова: *забезпечення якості (QA), IT-системи, продуктивність комп'ютерних мереж, QA в комп'ютерних мережах, економіка тестування.*

Аннотация. *Стандарты обеспечения качества проектирования и разработки ISO 9000 широко применимы при работе с программным обеспечением. Большинство современных компаний стремятся получить аккредитацию по ISO 9001, чтобы улучшить свои показатели. А для некоторых отраслей придерживание указанным стандартам является необходимым. Многие компании в настоящее время полагаются на информационные системы для ведения бизнеса, поэтому для получения аккредитации в этих областях IT-компаниями прилагаются огромные усилия. Использование программного обеспечения и компьютерных сетей практически во всех сферах современной жизни значительно повышает важность их стабильности и производительности. Процесс обеспечения качества QA является мощным инструментом, помогающим определить и устранить возможные сбои системы на этапе их разработки. Однако обеспечение высокого уровня качества требует учета влияния всех компонентов системы. Данное исследование посвящено вопросам тестирования приложений в режиме реального времени с учётом различной производительности компьютерных сетей, что позволяет эмулировать условия, максимально близкие к работе конечных пользователей. Также выполнен анализ целесообразности экономических затрат на QA. Исследованы алгоритмы взаимодействия пользователей с действующими системами реального времени при различных показателях производительности сети с целью определения наиболее частых*

ISSN 1028-9763. Математичні машини і системи, 2019, № 4

действий пользователя, которые вызывают сбои программного обеспечения, и предотвращения их во время процесса тестирования. Авторами предложены общие подходы к тестированию производительности и удобства использования системы в случае постоянного ухудшения работы сети.
Ключевые слова: *обеспечение качества (QA), IT-системы, производительность компьютерных сетей, QA в компьютерных сетях, экономика тестирования.*

Abstact. *ISO 9000 design and development quality standards are widely used in software. Most companies seek to gain accreditation to ISO 9001 to improve their overall performance. At the same time, compliance with these standards is necessary for some industries. Many companies are now relying on information systems for doing business, that is why huge efforts are being made to accredit IT companies in these areas. Usage of software incorporating networks in almost all areas of modern life significantly increases the importance of the stability of their operation and hence the quality of the performance. Quality assurance process QA is a powerful instrument which helps to identify and eliminate potential failures of an information system. However providing a high quality level requires considering the impact of all system components. This issue is devoted to software testing in real time based on various performance computer networks, allowing emulate conditions as close as possible to the end-users. The cost-effectiveness analysis of quality assurance processes was also analyzed. Algorithms of users' interaction with existing real-time systems with different network performance indicators are investigated in order to determine the most frequent user actions that cause software failures and possible prevention of the consequences of these actions during the testing process. The authors propose general approaches to testing the performance and usability of the system in the event of a permanent deterioration of the network.*
Key words: *Quality Assurance (QA), IT Systems, computer networks performance, QA in network environment, testing economy.*

DOI: 10.34121/1028-9763-2019-4-117-125

1. Introduction

Every part of everyday life involves IT and most devices used rely on software and networks for their operation. This begs the question "How reliable is software and Networks?" Even standalone software applications are released to the users which contain bugs. An analysis of software problems found in systems before acceptance testing was carried out by Sommerville [1] see Fig 1. These findings indicate the major problems being associated with the Design and analysis as opposed to coding.

Engineering disciplines have addressed the elimination of errors by using Quality Assurance (QA) processes but this has been slow to occur in the area of software development and operation particularly when used in a networking environment. The quality of software ultimately depends on the quality of the testing that takes place. From a business point of view there is a dichotomy between the requirement to deliver software at the earliest point and the length of time it takes to test the operation.

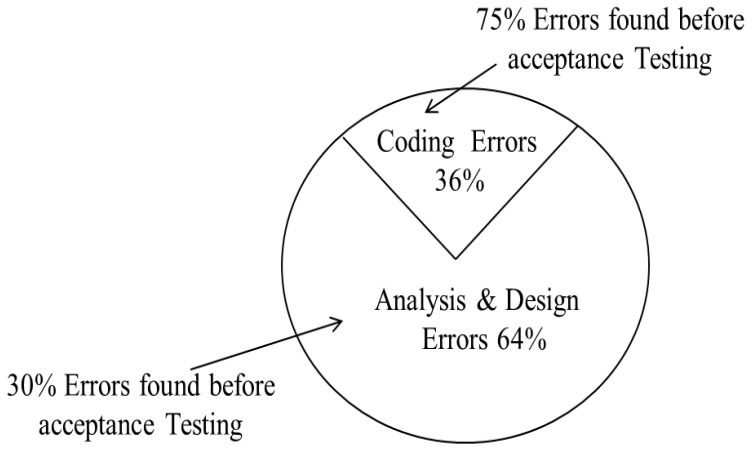

Figure 1 – Breakdown of the causes of Software Errors

ISO 9000 and the associated family of Quality Assurance standard specifications were originally published in 1987 by the International Organization for Standardization (ISO). They underwent major revision in 2000, 2008 and the current versions of ISO 9000 and ISO 9001 were published in September 2015 [2]. Most

companies these days are reliant on their IT systems, their customer IT system, supplier IT system and the network connecting these together. The QA standards are aimed at individual companies and the requirement is that they communicate with customers, suppliers, etc. externally to their QA system. Additionally there are no standards covering the infrastructure by which this is communication is achieved.

ISO 9000 and 9001 released in 2015 are based around seven principles namely: Customer focus, Leadership, Engagement of people, Process approach, Improvement, Evidence-based decision making, Relationship management. These are applicable to all disciplines of business. However ISO 9001 is the only standard within the ISO 9000 family that organizations can gain certification [3].

In the beginning of the 21st century it became obvious that the QA systems developed by companies to comply with ISO 9001 had problems dealing with IT Systems. These were becoming, more and more, a important to the business and so need special consideration. To this end a Software Engineering standard was developed in 2014 (ISO 90003).

2. QA Basics

Quality Assurance (QA) can be identified as the process, performed to ensure the corresponding quality level of a resulted product. Modern treads in software development demand that QA is a compulsory part of modern software development process, responsible for compliance with the requirements. QA includes Quality Management and Quality Control (QC). Ideally it covers all stages of software development e.g. planning and management, analysis and design, implementation and integration, criteria development and estimation. However, often only QC based around software verification using quality control technologies is performed. The main task of QA engineers is to search for defects and comparison of real and described behavior. It is a cyclic process repeated until the completion criteria is reached, i.e. result corresponds to requirements.

General scenario of QA is: software verification, defect identification, produces error (bug) report, sending bug report to developers, bug fixing and verification of modified software.

To perform a rigorous QA process there are several types of testing which should be implemented during the software development process. International Software Testing Qualifications Board (ISTQB) provides next determination: testing type is a tool for clear definition of the goal of a certain level for a program or project. Testing types can be classified according to the testing purpose (Fig. 2).

Another proposed classification is based on the testing levels and includes: Unit Testing, Integration Testing, System Testing, Release Testing and Acceptance Testing.

To perform QA different testing technologies can be use. There are five general testing technologies which correspond to different software development processes:

1. Formal testing i.e. testing according the documentation.

2. Exploratory testing, which implies simultaneous tests development and performance without strict requirements.

3. Ad hoc testing is similar to exploratory testing with only difference that such testing can be performed without background.

4. Scrum testing is agile technology, which covers all stages of software development and includes both manual and auto testing.

5. Testing for Extreme Programming (XP) consider testing of the most important elements, which obtains “extreme level” during software development process.

Generally software development process includes analysis of requirements, design, implementation and integration [6]. Two main approaches of QA can be specified: QA performed on each stage or QA as a separate stage, performed after implementation and before integration [7].

First approach is a consistent process, which includes verification of client requirements, checking of prototypes and mock-ups etc. at the design stage, module and integration testing at the implementation stage with functional testing after its completion and, in some cases, support testing after integration and release. Adopting such an approach guarantees a high quality level of developed products, but requires resources (time and money). Due to the economics of the process, in the most cases, QA is performed only after or as part of the implementation stage, which allows reduce costs, but provides a lower QA level [8].

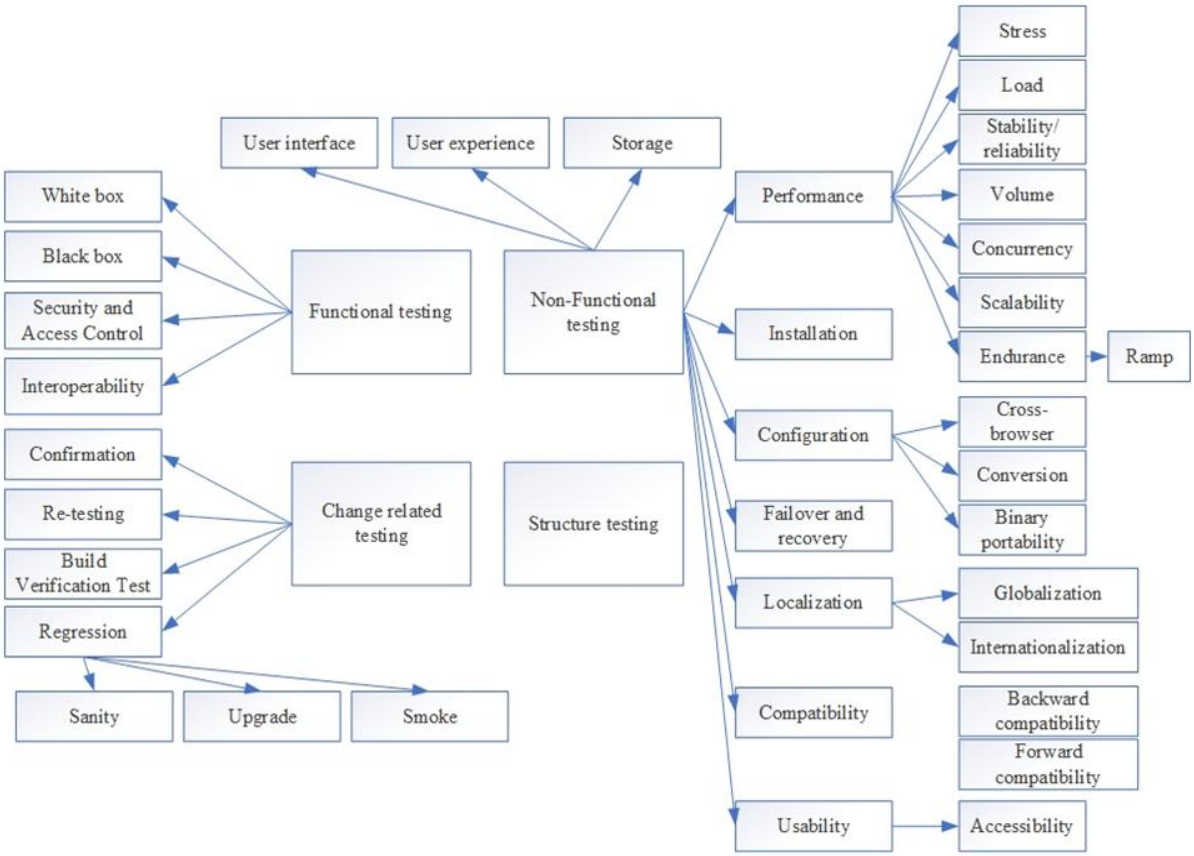

Figure 2 – Testing types

Real-time testing is a complex task, since it must cover not only functional compliance but also operation under strict timing constraints. Such testing is often performed at two levels. The first level includes QA performance with static data (stabs) to discover general errors in logic. Second level additionally covers timing parameters. Both levels include manual testing and auto testing. Usually testing is performed using dedicated QA servers or hosts, which can contain all the components of the system (in some cases some of them can be stabs). The availability of a large number of testing tools in the market is explained by the importance and high popularity of QA. The most popular with QA engineers are Selenium, Robotium, Linux Test Project, Junit for auto testing and Sprinter by Hewlett Packard Enterprise, Browserstack and Usersnap for manual testing [9].

3. Investigation of QA in Network Environments

3.1. Modern QA Problems

The availability of a huge number of testing tools and testing does not guarantee a high level of system quality. In most cases characteristics obtained on test servers are very far from the real results. Many problems appear because most of the testing is performed using black-box tech-

niques. This means that only general system behavior is covered without consideration of the possible problems that will be obtained after software integration into real network environments. This is the main reason why even well-tested systems on QA server software still have significant problems after release. This issue is partially covered by post-testing i.e. testing of the real deployed software. However such technique allows only the discovery of defects post-factum. A much better solution would be to prevent the appearance of problems related to real environment operation. Such an approach should contain not only standard ways of testing, but testing with changing settings of software architectural nodes characteristics that can affect software performance.

3.2. Investigation of Networks on QA

The most frequent problems that appear in a real environment are problems due to network operation. Pure network operation causes problems during resources loading, slow data processing and data loss. In turn slow system operation can be a reason for unexpected user actions (e.g. clicks), which can lead to system breaks, instability. A number of tests were performed to investigate user interaction with a live system.

Table 1 – The average No of clicks made by users in periods, with Pending Elements

Type of Content	0-4s	5-6s	7-10s	10-15s	>15s
Styles (css)	3%	36%	92%	94%	98%
Data blocks (text, news etc.)	4%	27%	93%	94%	97%
Forms	<1%	15%	72%	91%	98%
Graphs, videos	3%	8%	41%	52%	57%
Navigation	4%	25%	91%	95%	97%
User Profile	3%	12%	53%	61%	67%

Table 1 describes the number of redundant clicks made by the person until the resources are loaded. Experiments were also conducted Without Pending Elements and similar results were obtained as can be seen in Table 2.

Table 2 – The average No of clicks made by users in periods, with Pending Elements

Type of Content	0-4s	5-6s	7-10s	10-15s	>15s
Styles (css)	23%	56%	88%	95%	97%
Data blocks (text, news etc.)	13%	47%	78%	94%	96%
Forms	18%	78%	92%	95%	96%
Animation	<1%	<1%	<1%	<1%	<1%
Graphs, videos	12%	38%	43%	54%	56%
Navigation	20%	66%	91%	94%	96%
User Profile	18%	25%	33%	41%	42%

3.3. Recommendation of Investigation of Networks on QA

Considering the fact that each click or action can leads to unexpected software behavior, analysis of system behavior depending on different network settings is one of the tasks that should be performed during QA. Analyzing these aspects can help to find bottle necks and process them before the system is released as operational software. The results of such analysis should include:

– discovering UI elements that should be disabled before all data is obtained to prevent unexpected events;

– discovering network requests with the longest time to complete with further optimization of system operation with network resources (using asynchronous techniques, static data caching and preload, minimization of source data etc.);

– setting alternative scenarios of operation for different data loadings to prevent complete system failure;

– informing user about presented problems with recommendation how to improve system operation (i.e. deactivate the part which requires a lot of resources e.g. animation, graphs etc.).

The above proposed ways are not the complete solution however they can prevent unexpected system behavior and failures.

3.4. When Such Analysis is Necessary

Most modern software is implemented using client-server architecture with different levels of complexity. Complete performance of such applications depends on the performance of its individual parts. For QA this means that each of the system parts has an impact on the quality of system operation and requires attention [5].

Performance of the client part is limited by software and hardware characteristics of the client device and normally minimum specifications are used to control this. Network performance is guaranteed by Internet providers and is limited by network hardware and data transmission medium. The Server part can be represented by a huge number of devices connected in one network. Each of the nodes will have its own impact on total quality of the software. Dependency on system complexity and the amount of required testing is directly proportional, since the number of points that can lead to possible failure is higher.

Another popular deployment approach is using a number of different cloud services. In such a case the quality of system operation in a real environment is guaranteed by the cloud server provider according to the specified characteristics of the resources selected in the agreement.

Traditional QA seen in an engineering environment cannot cover all the possible issues that can occur because of problems in the components of the system operation. It is understandable that there is no existing way to predict all possible problems. However changes in software, hardware and network parameters can help to find a lot of the problems that never appeared on test environment. Despite the fact that it is not possible to test this type of systems under all possible conditions, clearly there are economic implications on the number and type of tests that can be performed.

3.5. Network Characteristics Without a Function in Testing Software

There is no universal tool which can solve the task of changing the conditions of the system operation. However there are some approaches that can be used during the QA process to produce possible errors due to low network or remote server performance. Approaches are based on the regulation of the network performance on the client side. This is particularly an issue when mobile devices using wireless networks are part of the requirement.

One approach is the usage of an external proxy-server to connect to remote servers with the testing system. There are a number of proxy-servers available at the moment with different quality of operation. These allow variable delays to be obtained in the range from 3s to 30-40s. Some can lead to a delay of more than 60s (mostly situated in Latin America or some countries in Asia). Usage of an external proxy-server allows the system to be observed in operation with different level of network performance. These approaches are limited by security policy issues.

The second way is by using a VPN connection. VPN connections with different settings are used as tunnels to control the speed of data exchange. VPNs have a good level of security but required additional resources to be established and configured for different tasks.

A similar variant is the usage of network devices on the client side to limit network performance by changing their parameters.

3.6. Network Characteristics Without a Function in Testing Software

There is a well-known axiom regarding QA: “While costs vary by project and environment, the costs to fix defects follow what has come to be known as the "1:10:100" rule” [11]. That means that the cost of each defect becomes 10 times higher at the next stage of product life-cycle.

The problems, discussed in the paper, are typically identical in a last production stage and their fixes will costs hundreds times detection at the initial stage. At the same time if a bug is detected and fixed in a traditional test phase it will be 10 times cheaper.

One of the most discussed questions is how much resources should be spent on QA. Figure 3 shows the Software Test Costs and Return on Investment (ROI) proportions [12].

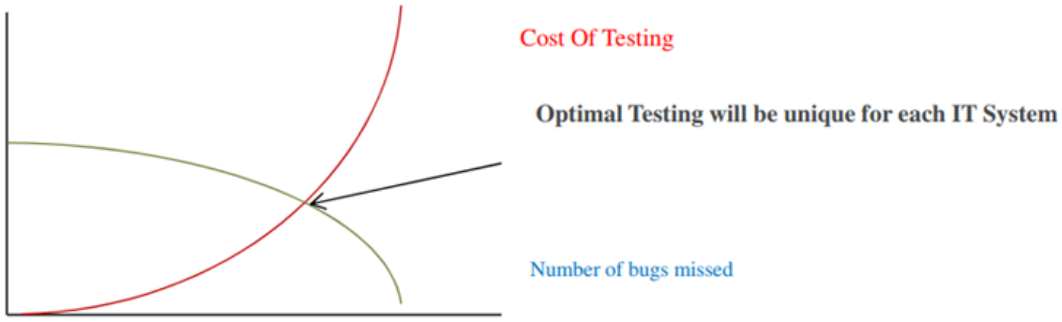

Figure 3 – Software Test Costs and Return on Investment (ROI)

The most common data provided in the books is that QA takes 30% of all product implementation process. At the same time most modern companies (e.g. Microsoft) prefer working using 1:1 ratio between QA and development (i.e. QA takes 50% of whole software development lifecycle). The results obtained from automated estimation models, available in the Internet, typically vary from 20 to 50% depending on project complexity. Table 3 contains result of survey based around the question “How much time spent on testing in comparison to the whole software development life cycle” [13].

Table 3 – Respondents to Questions

Question Response Option	% of Respondents
< 30%	28%
30% - 50%	48.6%
50% - 75%	14%
75% -100%	7%
Other (please specify)	3.75%

3.7. Conclusions from Experiments Conducted

Based on the users’ behavior from conducted experiments following general conclusions have been done:

1. Users don’t want to re-enter data.

2. In the most cases people didn’t experience the problems with animation (picture appeared without fade-In, pop- Ups hidden without fade-In and fade-Out). It was observed as design drawback but not as error on the website.

3. If the user is not focused on the specific content they will not pay attention to its absence (graphs, videos, user profile).

4. Problems with navigation are the only cases that cause an equal number of clicks and reloads, thus creating double load on system.

5. Current research is corporative and a large number of the respondents are involved in the IT industry (QA, developers etc.). This may suggest the reason is that they pay attention to problems which trivial user ignore.

6. It is better to display nothing, than broken styles.

7. Users don't like to click on pending elements – they prefer to initiate page reloading.

4. Conclusions

A significant saving can be made in the area of Software Maintenance by carrying out exhaustive testing before system release. This saving can be found not only in costs but also in terms of user annoyance. Fig. 4 shows the hidden areas that are not obvious when calculating the cost of failings that show up when the system has been delivered [14].

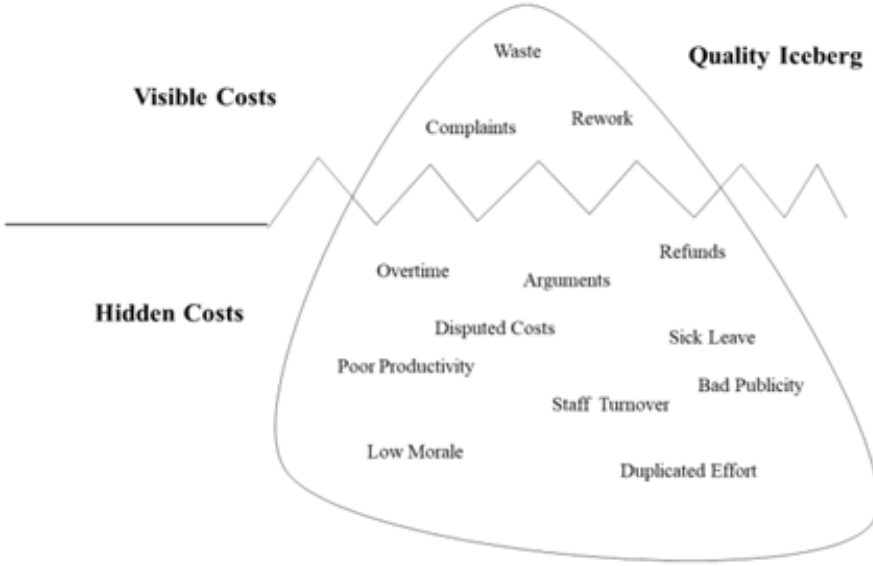

Figure 4 – Quality Iceberg

Having a tight QA procedure process can provide a significant improvement on the elimination of system failures. However this is dependent on considering all the components of the system.

This paper concentrates on the issue associated with testing in a real time environment under conditions that can be experienced by the end users.

Since most modern systems are dependent on external factors such as Cloud providers, internet operation, variety of client equipment this makes the testing process even more complex. Real time testing of the system is essential. It is also necessary to assume that the external providers do not meet their specification 100% of the time 24/7 and so testing assuming out of range conditions must be undertaken.

REFERENCES

1. Sommerville I. Software Engineering. Pearson, 2015. P. 121–140.

2. Quality management principles. 2019. URL: https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/pub100080.pdf.

3. SO standards. 2019. URL: https:// https://www.iso.org/standard/74348.html (Last accessed: 10.04.2019).
4. Potena P. et al. Creating a Framework for Quality Decisions in Software Projects. *ICCSA*. 2014. N 5. P. 434–448.
5. Selenium – Web Browser Automation. URL: https://www.seleniumhq.org (Last accessed: 10.04.2019).
6. Cinque M. et al. On the Impact of Debugging on Software Reliability Growth Analysis: A Case Study. *ICCSA*. 2014. N 5. P. 461–475.
7. Pressman R. Software Engineering. *A Practitioner's Approach*. McGraw Hill, 2014. P. 68–74.
8. Buthmann A. 2018. URL: https://www.isixsigma.com/implementation/financial-analysis/cost-quality-not-only-failure-costs/ (Last accessed: 10.04.2019).
9. Types of software testing. 2019. URL: https://geteasyqa.com/qa/software-testing-types (Last accessed: 18.04.2019).
10. Davies J.N., Verovko M, Posadska A. Simulation of network operation issues in QA process. *Mathematical modeling and simulation of systems*. MODS 2018: Thesis of reports of the Thirteenth International Scientific and Practical Conference (Chernihiv, June 25–29, 2018). Chernihiv: CNUT, 2018. P. 281–284.
11. Rice R. Achieving Software Quality Using Defect Filters. 2019. URL: https://www.riceconsulting.com/home/index.php/Defect-Management/achieving-software-quality-using-defect-filters.html (Last accessed: 18.04.2019).
12. Software Test Costs and Return on Investment (ROI) Issues Bob Hunt, Galorath Tony Abolfotouh, John Carpenter; Robbins Gioia. 2014. March. URL: http://www.iceaaonline.com/ready/wp-content/uploads/2014/03/Software-Test-Cost-and-ROI-Galorath-Feb-14-Hunt.pdf.
13. Dustin E., Garret T., Gauf D. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality, Addison-Wesley Professional; 1 edition. 2009. 368 p.
14. European Union, 2018, ICEBERG Report Summary, How to estimate costs of poor quality in a Software QA project: a novel approach to support management decisions, 2016 of Project ID: 324356 Funded under: FP7-PEOPLE. URL: http://www.iceberg-sqa.eu/ (Last accessed: 10.04.2019).

Стаття надійшла до редакції 04.07.2019