TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Elina Muhhamedjanova 153069IAPM

Implementing non-functional quality control for the System of the Administration of Chess Tournaments (SACT) based on ISO/IEC 25010:2011 quality model.

Master’s thesis

Supervisor: Maili Markvardt
MSc

Tallinn 2017

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Elina Muhhamedjanova 153069IAPM

Maleturniiride haldussüsteemi mittefunktsionaasete testide läbiviimine ja kvaliteedi hindamine vastavalt ISO/IEC 25010:2011 kvaliteedimudelile.

Magistritöö

Juhendaja: Maili Markvardt MSc

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Signature: ______________________________

Date: ________________________________

Permission

Remarc Systems company and Sport Rating Technologies International DMCC aware and agree that Elina Muhhamedjanova is using and considering the System of the Administration of Chess Tournaments (SACT) in the scope of master’s thesis in the Tallinn University of Technology.

Signature: ______________________________________

Date: ____

Abstract

The goal of this thesis is to describe an opportunity to make the non-functional product quality control of the System of the Administration of Chess Tournaments (SACT). In the scope of the project were considered the non-functional requirements provided by the customer - Chess Rating Agency for FIDE - World Chess Federation. The quality assessment of the current state of the system based on ISO/IEC 25010:2011 quality model referring to the data obtained as a result of testing.

The first section of this work is devoted to establishing the notion of quality control and its components. The second part of the thesis focuses on collecting data for the evaluation purposes. In the second section, requirements are analysed and test cases are developed and executed. The third part of the research examines data that is received after testing and an evaluation of the quality is made based on the received data.

As a result of this case study research, the non-functional product quality control was made for the SACT system including all principles of quality control process. It could be concluded that the system has some weaknesses and it needed to be improved to completely correspond to the each quality characteristic in ISO/IEC 25010:2011 quality model.

The thesis is in English and contains 84 pages of text, 5 chapters, 26 figures, 54 tables.

Annotatsioon

Käesoleva magistritöö eesmärgiks on kirjeldada Maleturniiride haldussüsteemi (SACT – the System of the Administration of Chess Tournaments) mittefunktsionaalse kvaliteedi kontrolli loomise võimalust vastavalt kliendi (Rahvusvahelise maleliidu reitinguagentuur FIDE) antud mittefunktsionaalsetele nõuetele ning hinnata ISO/IEC 25010:2011 kvaliteedimudelil põhineva süsteemi hetkekvaliteeti, mis põhineb mittefunktsionaalsete testide tulemustel.

Magistritöö on jagatud kolmeks peatükiks. Käesoleva töö esimeses peatükis tuvastatakse kvaliteedi kontrolli tagamise põhimõtteid. Töö teises peatükis kogutakse andmeid, mida tuleb süsteemi juures hinnata. Lisaks analüüsiti nõudeid ning arendati ja viidi ellu testitsenaariume. Uurimuse kolmandas peatükis analüüsitakse testimise ja kvaliteedi hindamisega saadud andmeid.

Selle tulemusena loodi SACT süsteemile mittefunktsionaalse kvaliteedi kontroll kõik kvaliteedikontrolli protsessi põhimõtted. Käesoleva töö tulemuste põhjal saab väita, et süsteemil on puudusi ning seda peab täiustama, et see vastaks ISO/IEC 25010:2011 kvaliteedimudeli alusel ette antud nõuetele.

Magistritöö on inglise keeles ning koosneb 84 leheküljest, 5 peatükist, 26 joonisest ja 54 tabelist.

List of abbreviations and terms

CPU	Central Processing Unit
CSRF	Cross-Site Request Forgery
DAO	Data Access Object
EC2	Elastic Compute Cloud
EVS	Estonian Centre for Standardisation
FIDE	Fédération Internationale des Échecs (french); World Chess Federation(eng)
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	the International Organization for Standardization
OOP	Object-oriented programming
OWASP	Open Web Application Security Project
PCI DSS	Payment Card Industry Data Security Standard
PGN	Portable Game Notation
RDS	Relational Database Service
SACT	The System of the Administration of Chess Tournaments
SMART	Specific, Measurable, Attainable, Realisable, Traceable
SQuaRE	Systems and software Quality Requirements and Evaluation
SSH	Secure Shell
TCP	Transmission Control Protocol
TFD	Tape Format Requirements Document
TRF	Tournament Report File
UCI	Universal Chess Interface
UI	User Interface
UX	User Experience

Table of contents

List of figures

List of tables

1 INTRODUCTION

The modern world consists of social-technical systems, developed by human to make our lives easier. Since we are surrounded by automated systems, software should be secure, allowing us to rely on it and have no fear of being endangered. This is true both for large-scale systems and simple administrative systems or services.

Any system should ensure the personal data protection, be comfortable and easy to use, and optimal in terms of resource usage. This principle also applies to exposed software extensions and other modifications. During a process of system evolution, new problems may occur. System reconstruction, changing platforms, adding new features, fixing existing defects might lead to defects and bugs. Considering the aforesaid, system should be designed as easily maintainable and testable.

Consequently, to receive the user satisfaction, provision of the trustworthy service is crucial, while software should be exposed to the permanent quality control.

1.1 RESEARCH PROBLEM AND OBJECTIVES

The aim of this research is to discover the structure of the non-functional product quality control and implement it in the System of the administration of chess tournaments (SACT), taking into consideration the non-functional requirements provided by the customer – Chess Rating Agency for the World Chess Federation – FIDE. The main motivation points include ensuring product quality, minimizing expenses, improving the level of security for personal data protection, ensuring the rational use of resources, or, in other words, the main motivation is to provide the trustworthy and reliable service for users and gain their satisfaction. Based on the problematics, were formulated the following research questions: how to implement the non-functional product quality control in practice; what stages does the product quality control include?

1.2 RESEARCH SCOPE AND LIMITATIONS

This research covers the product quality, while the process quality is left out of the scope of the research. The quality control of product is based on the non-functional requirements provided by the customer. Each requirement from the provided list is analysed and, where necessary, it is redesigned in a more appropriate way according to the best principles of the requirement specification and formulation. Each requirement is related to a quality attribute from ISO/IEC 25010:2011 product quality model. According to the requirement and its relation to the quality factor, an appropriate non-functional aspect of the system is tested. The final quality evaluation is made as determination of compliance with the expected value defined in the requirement and its implementation in the system.

1.3 RESEARCH METHODOLOGY

Designing the research, the author opted for the Case Study method. As for the meaning of the case study concept, several definitions are available, one of which is presented below. It concerns the definition of the software engineering case study methodology. "Case study in software engineering is an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified." [1]

The case study strategy also features purpose classification. The main purpose of this research is descriptive.
"Descriptive – portraying the current status of a situation or phenomenon." [1]
The case study research includes the following stages: formulating the research questions, case selection, analysis of the selected techniques, collecting data, and, finally, evaluation, analysis and reporting of the received data.

1.4 THESIS STRUCTURE

The master's thesis is divided into six main chapters. The brief overview of each chapter is presented below.

In the **introduction,** the author introduces the research goals, objectives and the research problems. The main motivation points and research methodologies could be found here.

The background and general principles chapter gives a brief overview of the main theoretical aspects which characterise the background for achieving the assigned objectives and goals.

The case study section focuses on testing the system according to the specified non-functional requirements, related to the quality attributes, taken from the product quality assessment model. This chapter provides the system description and the non-functional requirements from the customer, which should be fulfilled. Next, the detailed test procedures, tests results, and suggestions for improvements – where test failed – are presented. After testing the system, the quality evaluation with reasoning is made.

The results and analyses chapter illustrates the achieved results. The evaluation of done work is made here. The answers to the research questions are given.

A separate section – **future work –** presents prospects for further research on this topic.

The last chapter – **conclusion** – sums up all results which the author produced during the research.

2 BACKGROUND AND GENERAL PRINCIPLES

In this chapter are presented the relevant concepts and theory which form the basis of master's thesis research related to product quality control topic. All concepts are considered as a part of the Software Engineering.

2.1 SOFTWARE PRODUCT QUALITY CONTROL CONCEPTS

2.1.1 Concept of quality

First of all, it is necessary to clarify what is software quality means and what components it consists of. Below are provided some of the software quality definitions suggested by IEEE (Institute of Electrical and Electronics Engineers). [2]

1. "The degree to which a system, component, or process meets specified requirements." [2]

2. "The ability of a product, service, system, component or process to meet customer or user needs, expectations or requirements" [2]

3. "The degree to which a set of inherent characteristics fulfils requirements" [2]

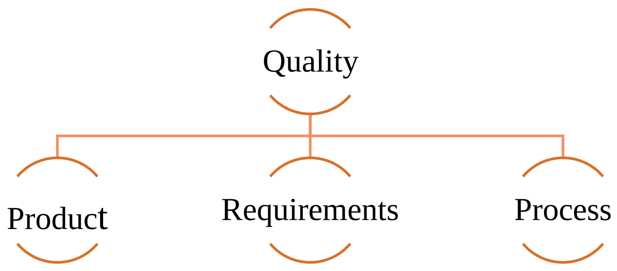


Figure 1 Quality components [3].

As it could be seen from the Figure 1, the quality involves three main components: product, process, and requirements. It means that to achieve quality software it should be specified set of rules, needs or expectations for the system, or program component and for the development process.

The unit of study of my research is based on the product quality. To receive the quality product, it should be made the analysis of expectations and needs, they should be translated into clear, measurable product attributes and as a result, these attributes should be successfully implemented in the product.

2.1.2 Software product quality control

“Quality control involves monitoring specific project results to determine if they comply with relevant quality standards, and identifying ways to eliminate causes of unsatisfactory results.” [4] To ensure the software assurance and reliable product, it should be exposed to continuous quality control.

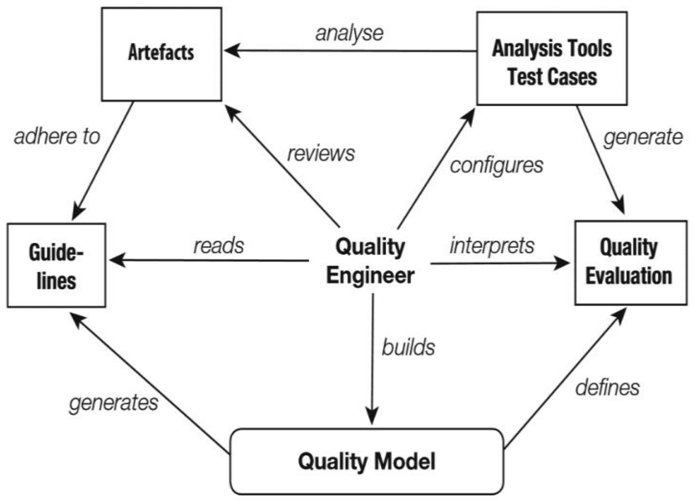


Figure 2 Quality control model [4].

Figure 2 illustrates the main components in software product quality control. Let us consider all stages from the process in more details. First of all, at the heart of the entire process lies quality model. Quality engineer could build the own quality model or he could refer to the existing quality models. There are a large variety of models, the use of which depends on the goals and preferences of the developers and quality engineers.

The requirements and guidelines for the product should be formulated based on a quality model. The developers should follow all requirements in the system implementation.

The quality analytic discovers product requirements, according to it configures test cases and selects appropriate test tools for each testing module.

The next stage is to execute the test cases, collect data, review and analyse it. At the end, the quality engineer should give the quality evaluation of the system. Further requirements are edited and the process is repeated.

2.1.3 SQuaRE

The good instrument for each quality engineer is the set of standards from SQuaRE project. “SQuaRE stands for Software Product Quality, Requirements and Evaluation. It is a long-running project at ISO to consolidate and replace the old ISO/IEC 9126 which used to be the main standard for software product quality.” [5]

All standards from SQuaRE project are approved by Estonian government and could be found c in EVS (Estonian Centre for Standardisation). [6] The standards are translated into Estonian, also it is possible to acquire an English version too.

The aim of the SQuaRE project is to have one set of standards to model, specify, measure and evaluate software product quality. SQuaRE has five series of standards. Each of them is presented and described in Figure 3.

Quality Management

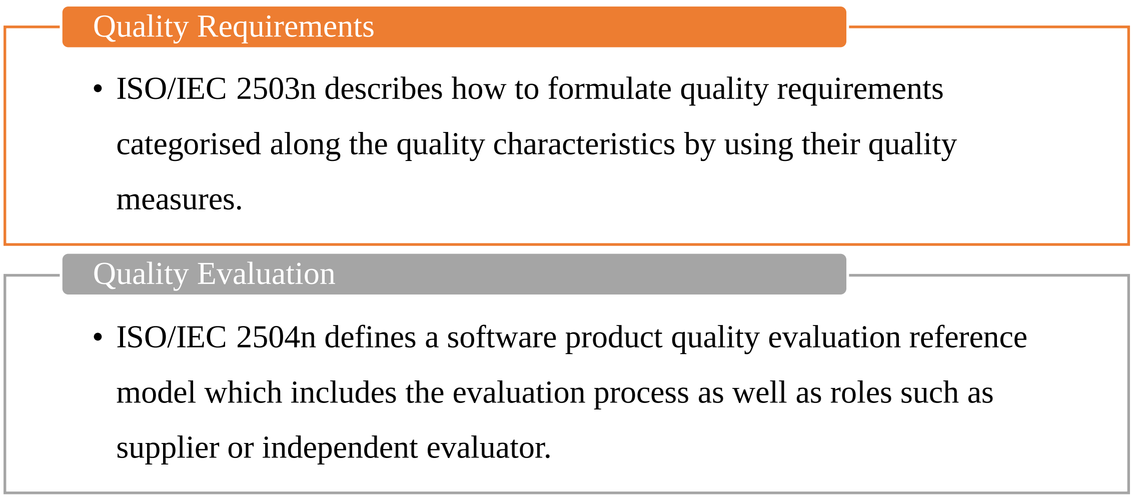
- ISO/IEC 2500n defines all common models, terms and definitions referred further by all other standards from SQuaRE series. Provides requirements and guidance for the planning and management of a project. It is good assistant in acquaintance with the rest of standards.

Quality Model

- ISO/IEC 25010 - describes a quality model framework. Contains the set of quality characteristics of the product. Quality model is the base of the product quality control. ISO/IEC 2010:2011 model is a definition quality model and acts like a check list for quality testing.

Quality Measurement

- ISO/IEC 2502n describes what is measurement in the context of quality evaluation. Illustrates in details the concept of quality measure elements. Specifys the metrics for each quality attribute from quality model. Very useful document in requirement specification process.



Quality Requirements

- ISO/IEC 2503n describes how to formulate quality requirements categorised along the quality characteristics by using their quality measures.

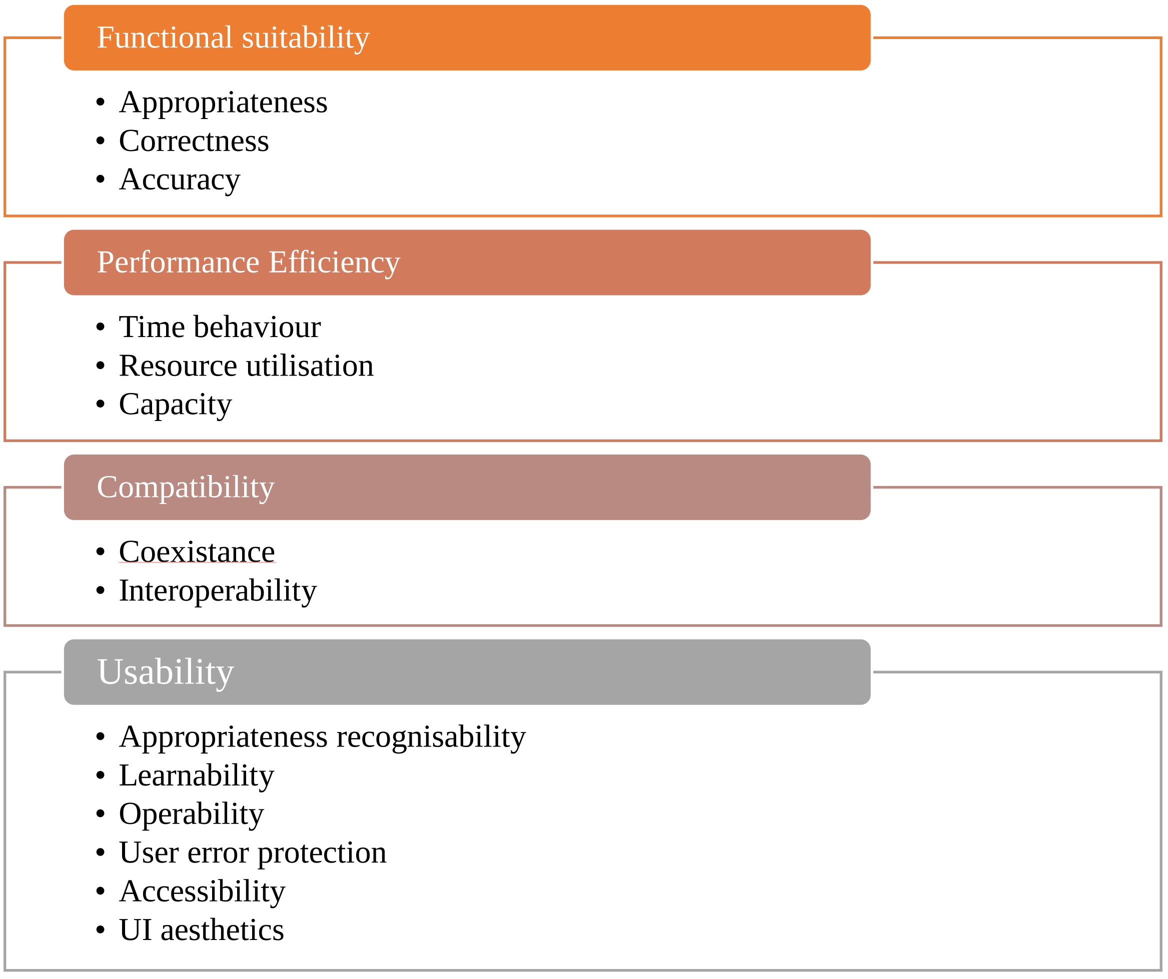
Quality Evaluation

- ISO/IEC 2504n defines a software product quality evaluation reference model which includes the evaluation process as well as roles such as supplier or independent evaluator.

Figure 3 SQuaRE series of standards.

2.1.4 Product Quality Model

This research is based on product quality model from SQuaRE project: ISO/IEC 25010:2011. Model is composed of eight characteristics, which are divided to sub-characteristics (Figure 4).



Functional suitability

- Appropriateness
- Correctness
- Accuracy

Performance Efficiency

- Time behaviour
- Resource utilisation
- Capacity

Compatibility

- Coexistance
- Interoperability

Usability

- Appropriateness recognisability
- Learnability
- Operability
- User error protection
- Accessibility
- UI aesthetics

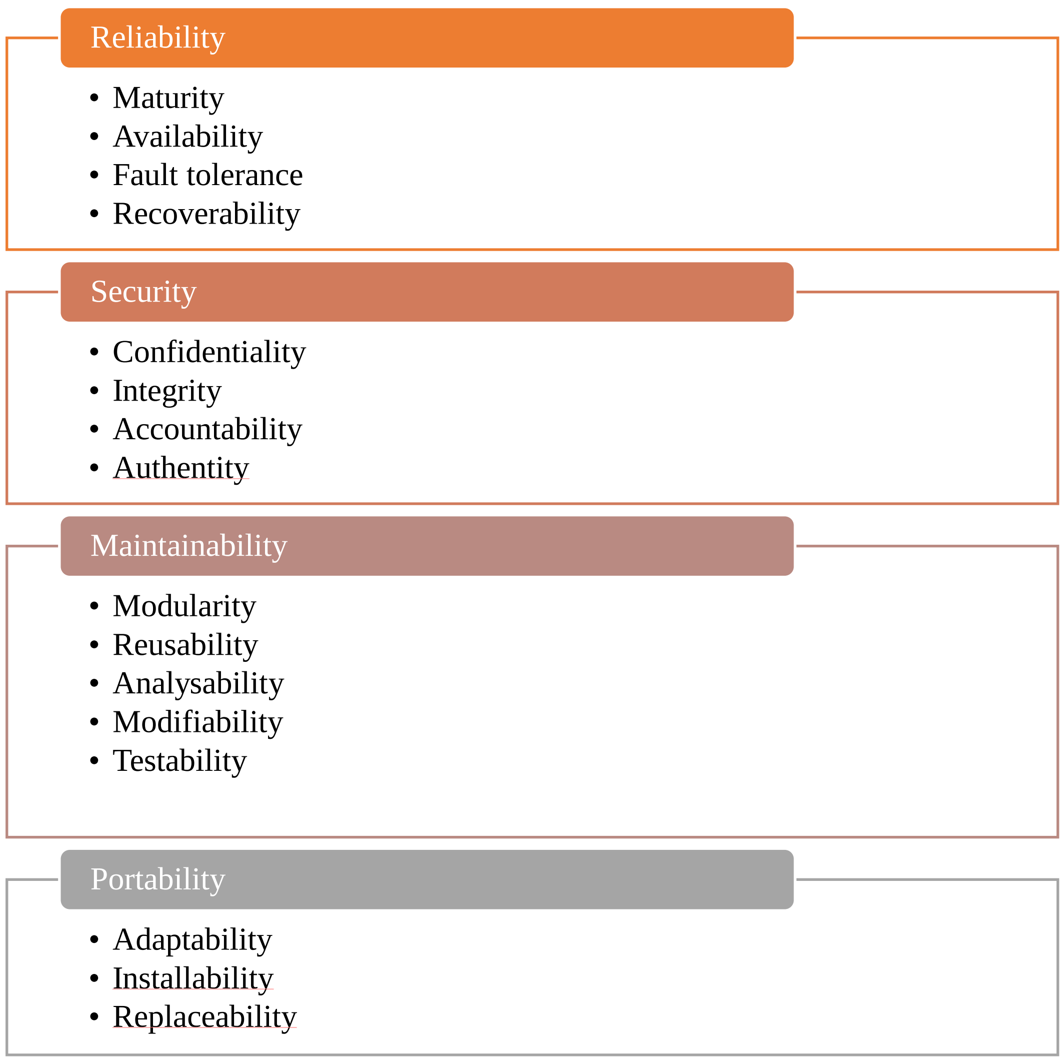


Figure 4 product quality model from SQuaRE project: ISO/IEC 25010:2011.

Each characteristic of this product quality model represents the quality factor, which in future could be measured and evaluated. It should be noticed, that model does not provide standards how to measure and evaluate quality attribute.

Product quality model represents the checklist of each system's quality components, which should be controlled and tested according to specific requirements for achieving quality assurance goals.

The research is concentrated only on non-functional quality aspects, so it will not be considered functional suitability attributes.

Below is presented the brief description for each quality attribute according to the official document of International Standard ISO/IEC 25010. [7]

Performance efficiency describes how well the product responds to user requests and how efficient it is in its execution.

Time behaviour - Responding and processing time for the appropriate request. In other words, the performance of the system.

Resource utilization - The amount of the recourses which are used by processing for an appropriate request. For instance, CPU (Central Processing Unit), memory storage, battery etc.

Capacity - The amount of maximum limits of the system parameter meet the request. For instance, the number of items that can be stored, the number of concurrent users, throughput of transactions, and size of the database.

Compatibility defines the quality that a product does not disturb or can even work together with other products.

Co-existence - Ability to interface and work together with other software or hardware products.

Interoperability – Co-working with another system or systems. Ability to exchange information between components from another system.

Usability subsumes the aspects of how easy the system can be used. This includes how fast using it can be learned but also if the interface is attractive and if challenged persons are able to use it.

Appropriateness recognisability – ability to recognize by user, if the system is meet their needs and expectations. For instance, availability of tutorials, documentation or representative information.

Learnability – Availability of learning how to use the system.

Operability – Availability of attributes that make it easy to operate and control the system.

User error protection – Does the system protect users against of making errors?

User interface aesthetics – Is the UI(User Interface) of the system is attractive and comfortable in use for users? Involves such aspect like UX(User eXperience).

Accessibility – How easily user could operate or access the system? This applies to such circumstances as people with disabilities, the age factor.

Reliability attribute specifies how software product reacts on failures or how the system behaves under specified conditions for a specified period of time.

Maturity – Statistical measure of how often and how long the system is in condition of non-responding and non-working.

Availability – The time and period when the system could be accessible by users.

Fault tolerance - How the system behaves when it meets unexpected environment.

Recoverability – Indicates how fast and how qualitatively the system can be restored after failures and problems. It includes data recovering, re-establishing the desired state of the system.

Security includes keeping data intact and a secret as well as that the product needs to make sure that its users are who they claim to be.

Confidentiality – The system ensures that data are accessible only to authorized users which have access or rights to see/modify a specific data.

Integrity – The system prevents unauthorized access, or modification of any specific data.

Non-repudiation – If some actions have been made, there will be a prove of it.

Accountability – Modifications are traced to the source.

Authentity– Confidence that information being what it is claimed to be.

Maintainability describes that the system should be designed and programmed in a way that it is easy to comprehend, change and test.

Modularity – Does system composed of components such that a change to one component has minimal impact on other system's components.

Reusability – If there any components in the system which could be reused in this system or in another one?

Analysability – How easily it could be found the place/ the component of the system for further modifying or testing procedure.

Modifiability – Availability of the system to be modified without introducing defects or degrading existing product quality.

Testability – How easily the system could be tested?

Portability means that the necessary changes can be easily done and it can be easily installed.

Adaptability – Availability of the system to be adapted to different hardware, software or other operational or usage environments.

Installability - Availability of the system to be successfully installed and/or uninstalled in a specified environment.

Replaceability – How easily a product can replace another specified software product for the same purpose in the same environment.

2.1.5 Requirements

Requirements represent the set of rules which should be fulfilled in the product to achieve the quality software as a result. Requirements could be functional and non-functional and related to product or process.

Requirement Engineering process involves three main stages: elicitation, analysis and specification. The unit of study in this paper focuses only on third stage – requirement specification. It would be considering the correctness of requirement specification to make it testable.

Figure 5 illustrates the way where requirements are getting from. The main role in requirement specification process play laws, ethic norms and customs of the country. The government decides which standards they will admit or not. According to standards are build best practices, principles and suggestions how to achieve the best implementation.

After all these components are formulated and accepted, the users of the product, stakeholders, purchasers, customers and other interested faces put forward their demands, expectations and requirements based on the remaining sets of rules.

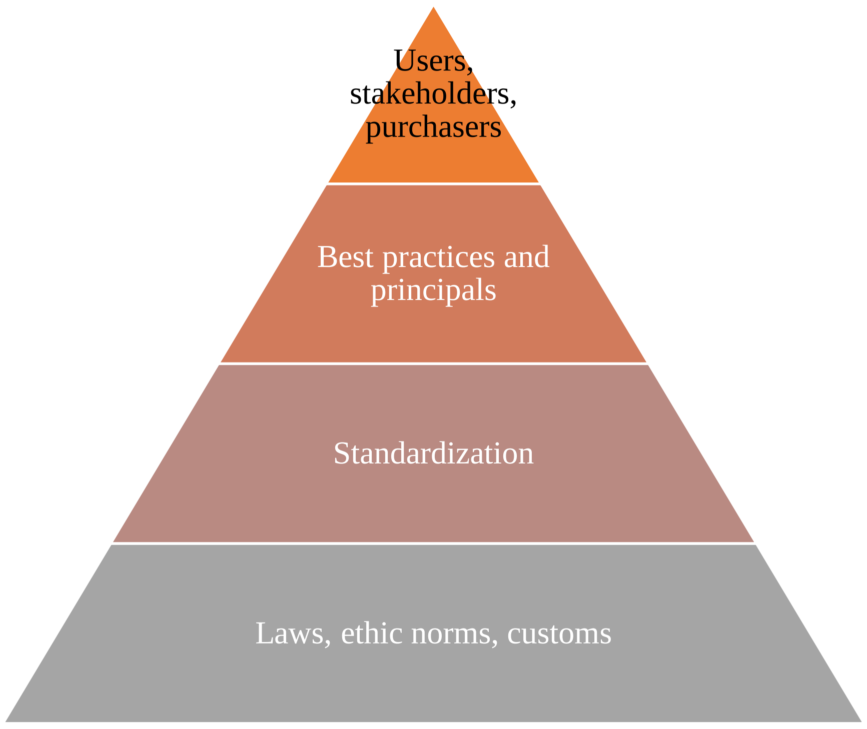

Users, stakeholders, purchasers
Best practices and principals
Standardization
Laws, ethic norms, customs

Figure 5 Origin of requirements [3].

In this research are considered non-functional product requirements, also they are called product quality requirements.

It is very important to formulate/ specify requirement in the correct way. It should be testable and measurable. The requirement should refer to quality model attribute. Quality model characteristics and sub-characteristics represent the checklist for requirements identifying. One of the techniques to ensure the correctness of requirement formulation is SMART requirement. The SMART technique identifies the main objectives which are needed to be achieved. A SMART determines five main rules which must meet the formulation of the requirement. These goals are: Specific, Measurable, Attainable, Realisable and Traceable.

Here would be explained each objective in details.

- Specific

Specific means that requirement should describe exactly what is required. It should be simple in its formulation, only one condition/aspect should consider in one requirement. It should be free of ambiguities.

- Measurable

In requirement should be met objective and simple way of measuring conformance. The requirement should contain the measured attribute, measurement units, and expected measurement value.

- Attainable

"By an attainable requirement, it means if it is possible physically for the system to exhibit that requirement under the given conditions." [8] In other words, it should be technically possible to implement and test.

- Realisable

When setting requirements, it is taken into account the availability of resources. The main question here is it possible/ realistic to achieve these requirements with specified constraints? Into account should be taken following resources as time, budget, staff skills and qualification, hardware and software for development and testing etc.

- Traceable

"Requirements Traceability is the ability to trace (forwards and backwards) a requirement from its conception through its specification to its subsequent design, implementation and test." [8] Other words, traceability shows the relation between business needs, implementation and testing. Usually, this kind of connection is used if there is a specific need to identify interconnection between components. For instance, when one requirement depends on another. Moreover, it helps to ensure the implementation of each requirement, allows to easily make modifications.

The correct requirement formulation should contain following components [9]:

- Measured attribute - *product quality factor. In other words: the attribute which should be tested/evaluated/measured*
- Test and conditions - *short description how the quality factor could be measured*
- Scale - *measurement units*
- Result - *what is expected to be achieved*

2.1.6 Testing procedure

To find out the correspondence the system to specified non-functional requirements and give the quality evaluation according to the fulfillment the requirements, the system should be tested. The data for comparison should be collected, after the analysis and evaluation could be made. Generally, this procedure called testing. The testing procedure structure is presented on Figure 6.

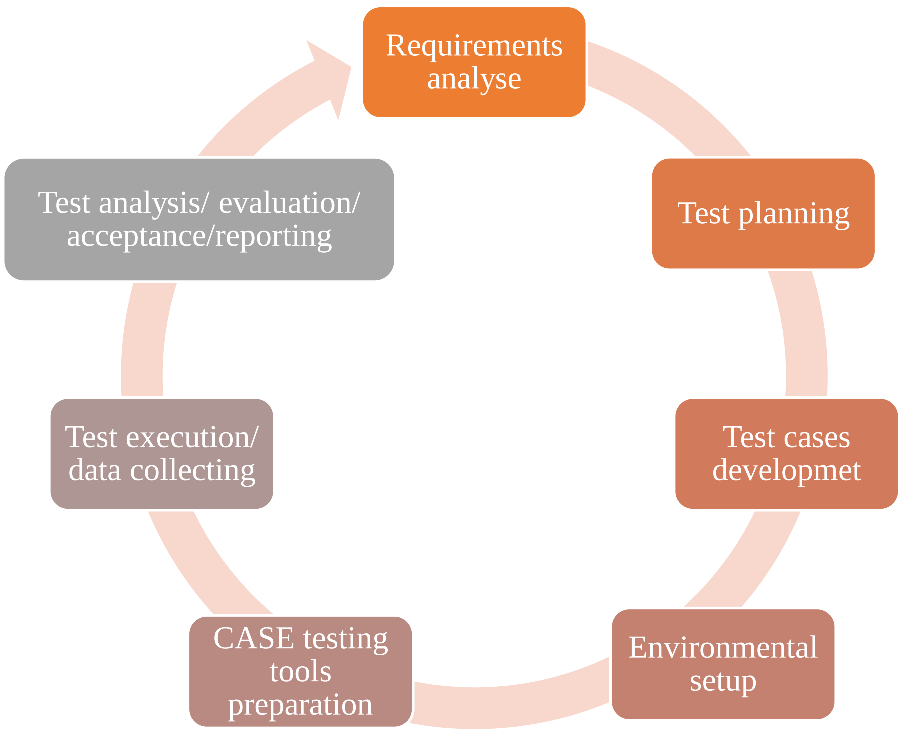

Requirements analyse
Test planning
Test cases developmet
Environmental setup
CASE testing tools preparation
Test execution/ data collecting
Test analysis/ evaluation/ acceptance/reporting

Figure 6 Testing life cycle [10].

When requirements are specified and analysed, the test plan should be identified. Testing plan should involve such components as: system description, testing objectives – requirements, if there are any constraints and limitations in testing – they should be indicated, testing strategies, environments and other testing details.

After requirements are analysed, test plan is formulated, the test cases should be designed for each requirement. The test case should describe the situation and detailed description of execution steps.

Environment setup stage includes the setting up the testing environmental: server, client, network and other necessary components.

The next section is the preparation of case tools. There are a big variety of different analyses tools, which will help to identify the weaknesses in the system according to an appropriate aspect. Each test case should be accompanied by appropriate testing tool for it.

After all of the tools are prepared the tests could be executed and output data collected. Above all collected data during the testing procedure, carried out all subsequent work: data is compared, analysed and reported. Reports also could involve the explanations where weaknesses have met and how they can be eliminated.

2.1.7 Quality evaluation

Definition of quality evaluation: “Systematic examination of the extent to which an entity is capable of fulfilling specified requirements.” [2]

For quality evaluation process is needed to adhere to the following plan:

1. Should be specified clear requirements for the product with measurable properties according to appropriate attribute in the assessment quality model.

2. Collecting data by checking system’s component for specified aspect in the requirement. In other words, the system should be tested.

3. Make comparison between received data and criteria specified in requirements.

4. Summarize the results if the system meets the specified requirements

3 THE CASE STUDY

The case study chapter is focused on non-functional product quality control on the real system - The System of the administration of chess tournaments (SACT). This chapter provides all stages of the process of assessing the quality of the system based on all theoretical aspects presented in the previous chapter.

3.1 SYSTEM's TARGETS

The System of the administration of chess tournaments (SACT) has been ordered by Chess Rating Agency (international) Ltd for FIDE. FIDE is a French organization and stands for - Fédération Internationale des Échecs. In English, it reads: World Chess Federation. Chess Rating Agency provides service of different rating calculations, information about chess tournaments and broadcasting chess games.

The purpose of the system is to ensure automation of the activity of the organizers, judges and members of chess tournaments, the order of which is determined by the rules of FIDE, by offering services of administration chess tournaments in the Internet environment.

3.2 CASE STUDY TOOLS AND TECHNIQUES

All test procedures described below imply that the test executor has an access to the source code of the system on the test server and has the local copy on his machine. On the local machine, should be installed the Java SE Development Kit 7, Apache Maven, Apache Tomcat (acceptable from 6 version), MySQL database, IDE (Integrated Development Environment). Below are provided the list of case study tools which are needed to be installed before testing. Before testing, all automated tests and files for testing should be downloaded by link from Appendix 1.

3.2.1 Apache JMeter – performance testing

“The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.” [11]

With this instrument was implemented the performance testing procedure – load tests and capacity tests. Moreover, for testing software reliability aspect were used stress tests. It was implemented one test plan, which involves different test cases. Before executing tests, it should be configured in test plan properties window the server name, port, tournament and round ids for testing purposes.

Load tests

“The purpose of loading tests is to verify application behavior under normal and peak load conditions.” [12]

In our case, it was imitated the situation of handling of specific request with assigned number of concurrent users. In thread properties window, in input field “number of threads (users)” should be set the value of concurrent users specified in requirement.

Capacity tests

“The purpose of capacity tests is to determine how many users and/or transactions a given system will support and still meet performance goals.” [12]

To check the capacity of the system was simulated a situation of handling the specific request with assigned a critic, but still relevant number of concurrent users. It was taken appropriate test case from load tests and updated the thread properties- the number of concurrent users.

Stress tests

“The purpose of the stress tests is to determine or validate an application’s behavior when it is pushed beyond normal or peak load conditions. Stress testing enables to identify application’s weak points, and shows how the application behaves under extreme load conditions.” [12]

Stress tests were used to find out how the system reacts and recovers after failures. It is understood that stress tests will give such a load on the system, which it will not be able to process. As a result, it would figure out how the system behaves in situations of extreme and unexpected input flow. In general, by exposing the system to stress tests, the reliability of the system will be tested.

3.2.2 Security testing

Nessus scanner

“Nessus is a proprietary vulnerability scanner developed by Tenable Network Security. Nessus allows scans for the following types of vulnerabilities:

- Vulnerabilities that allow a remote hacker to control or access sensitive data on a system.
- Misconfiguration (e.g. open mail relay, missing patches, etc.).
- Default passwords, a few common passwords, and blank/absent passwords on some system accounts. Nessus can also call Hydra (an external tool) to launch a dictionary attack.
- Denials of service against the TCP (Transmission Control Protocol)/IP (Internet Protocol) stack by using malformed packets
- Preparation for PCI DSS (Payment Card Industry Data Security Standard) audits” [13]

The Nessus scanner allows easily and quickly detect the common possible vulnerabilities of the system. It is very good solution for small enterprises and simple systems because the security audit could cost a lot of money.

SSL Server Test tool

SSL Server Test tool is free online server provided by Qualys SSL Labs company [14]. “Service performs a deep analysis of the configuration of any SSL web server on the public Internet.” [14]. This tool provides the total rating based on analysis of the SSL configuration. The final Assessment is based on following categories: certificate, protocol support, key exchange, cipher strength.

3.2.3 Selenium tests

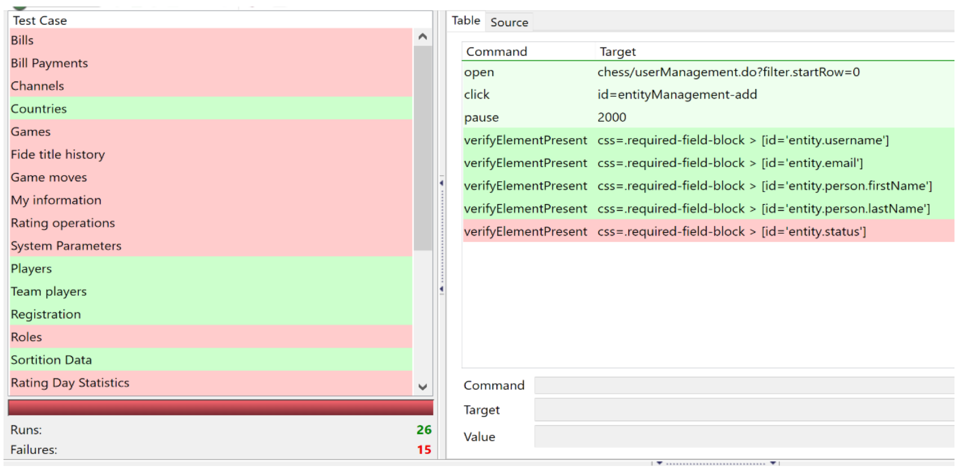
“Selenium is test automation framework for Web Applications.” [15]

To developed selenium tests were used Selenium IDE plugin for Firefox browser. It helps to develop test cases quickly and easily. After test cases have been developed and recorded, they can be generated to the code in the selected programming language.

Selenium tests were written to usability testing for user error protection aspect. It was developed two test cases.

- Required fields test suite

Test suite consists of twenty-six test cases and controls that all required fields of edit modes are marked and clearly identified that they are mandatory to fill. Each test case represents one edit page. Below is presented an example of the command and target value. The mandatory field should be wrapped in the required-field-block class. Moreover, after execution, it is clearly visible which fields were marked and which are not, which allows you to quickly find the error and fix it (Figure 7).



Test Case
Bills
Bill Payments
Channels
Countries
Games
Fide title history
Game moves
My information
Rating operations
System Parameters
Players
Team players
Registration
Roles
Sortition Data
Rating Day Statistics

Runs: 26
Failures: 15

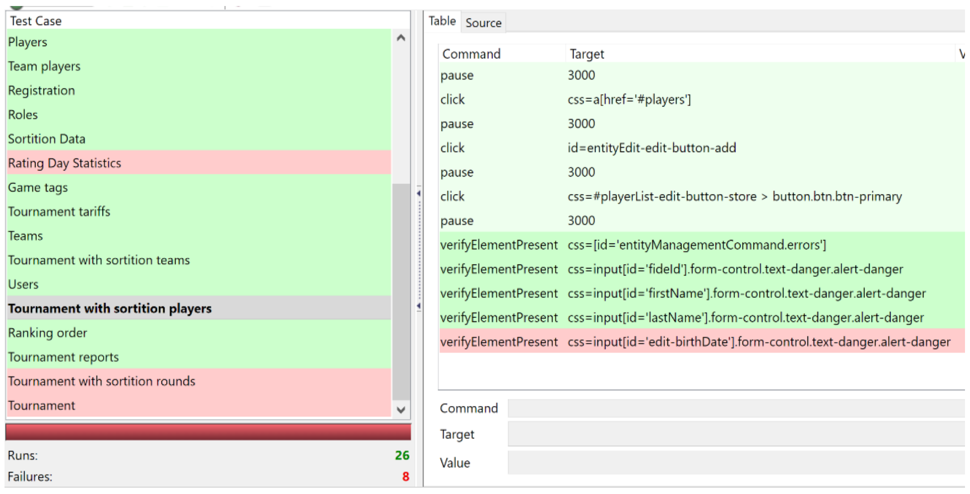
Table Source

Command	Target
open	chess/userManagement.do?filter.startRow=0
click	id=entityManagement-add
pause	2000
verifyElementPresent	css=.required-field-block > [id='entity.username']
verifyElementPresent	css=.required-field-block > [id='entity.email']
verifyElementPresent	css=.required-field-block > [id='entity.person.firstName']
verifyElementPresent	css=.required-field-block > [id='entity.person.lastName']
verifyElementPresent	css=.required-field-block > [id='entity.status']

Command
Target
Value

Figure 7 JMeter required fields test result example.

- Error handling test suite



Test Case
Players
Team players
Registration
Roles
Sortition Data
Rating Day Statistics
Game tags
Tournament tariffs
Teams
Tournament with sortition teams
Users
Tournament with sortition players
Ranking order
Tournament reports
Tournament with sortition rounds
Tournament

Runs: 26
Failures: 8

Table Source

Command	Target
pause	3000
click	css=a[href='#players']
pause	3000
click	id=entityEdit-edit-button-add
pause	3000
click	css=#playerList-edit-button-store > button.btn.btn-primary
pause	3000
verifyElementPresent	css=[id='entityManagementCommand.errors']
verifyElementPresent	css=input[id='fideId'].form-control.text-danger.alert-danger
verifyElementPresent	css=input[id='firstName'].form-control.text-danger.alert-danger
verifyElementPresent	css=input[id='lastName'].form-control.text-danger.alert-danger
verifyElementPresent	css=input[id='edit-birthDate'].form-control.text-danger.alert-danger

Command
Target
Value

Figure 8 JMeter error handling test result example.

Test suite consists of twenty-six test cases and controls that all required fields or fields which have some restrictions have been highlighted in case of invalid input and the message alert with error explanations occurs. Each test is built in such a way that in the event of a fail, it is clearly apparent on which field it is originated (Figure 8).

3.2.4 Source code analyses tools

EclEmma

“EclEmma is Java coverage tool for Eclipse IDE.” [16] It was used to analyse jUnit test coverage of the specific aspect in code. According to provided requirements, it should be analysed the instruction coverage by jUnit tests. It is possible to select another coverage counter, for instance, branch, line, methods etc. The tool is easy in use. It involves only two steps: run program, analyse results.

Unnecessary Code Detector

“Unnecessary Code Detector is plugin to find unnecessary, not used or dead public code in the Java code application.” [17]

Unnecessary/not used/ dead code is all synonyms and it involves public classes, methods or fields which have no references in the whole source code. It works very simply, the only thing you need is to select the part to be analyzed and run it. This tool could help in refactoring process in the big systems, where it is difficult to trace the dead code.

3.3 SYSTEM QUALITY CONTROL

In this chapter is presented the technical part of the work. Below could be found requirements in the form in which they were received from the customer, their analysis on SMART feature, developed test cases and tests results. The test case format represents the requirement id, requirement description, measured attribute and its scale, goal which is needed to be achieved to pass the test, analysis on SMART feature, test steps, the detailed explanation of expected results after executing test steps, to clarify what should be implemented to achieve the goal, result after test execution and if test was failed, the suggestion how to improve it is also presented. Requirements are divided by quality characteristics and sub-characteristics of the ISO/IEC 25010:2011 quality model.

3.3.1 Performance efficiency

Time behaviour

Table 1 Sortition algorithms test case.

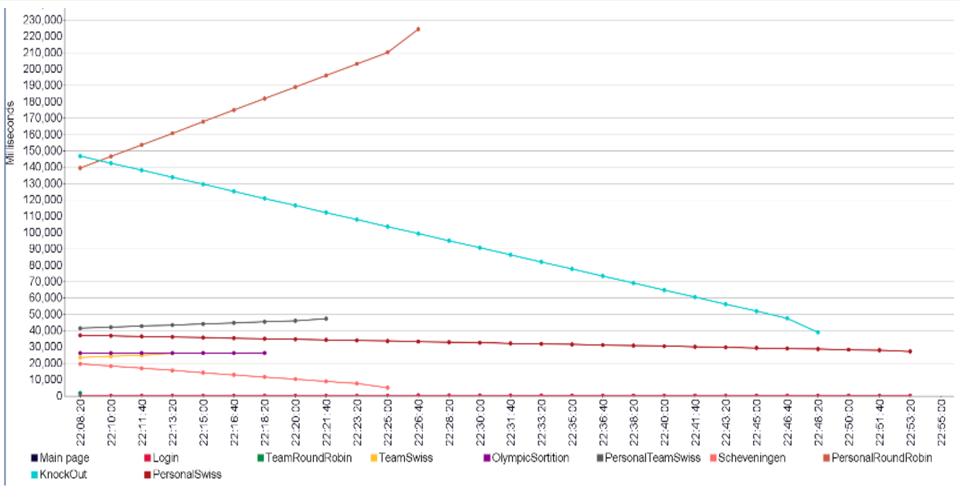
R. ID: R.1		
Description: Sortition algorithm up to 500 players should take up to 5 minutes with 5 concurrent users. It should be tested all sortition algorithms for following tournaments: personal round-robin, personal swiss-system, personal knock-out, team round-robin, team swiss-system, personal/team swiss-system, Scheveningen system and Olympic system.		
Measured attribute	**Scale**	**Goal.**
Response time	minutes	up to 5 minutes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Before running the test, select appropriate tournament according to testing parameters (tournament system, type) and delete all existence pairings. 2. Open Apache jMeter test plan `SACTPerfomance.jmx.` 3. Run thread: `SortitionAlgorithm.` 4. See result report (graph or report table).	
Expected results	All tests were successfully passed. Sample time of each test case is gained up to 10 minutes.	
Result	**OK.** Test was successfully passed.  Figure 9 Response time graph of sortition algorithms. The maximum response time was 225000 milliseconds, which is 3.75 minutes (Figure 9).	

Table 2 Entity lists performance test case.

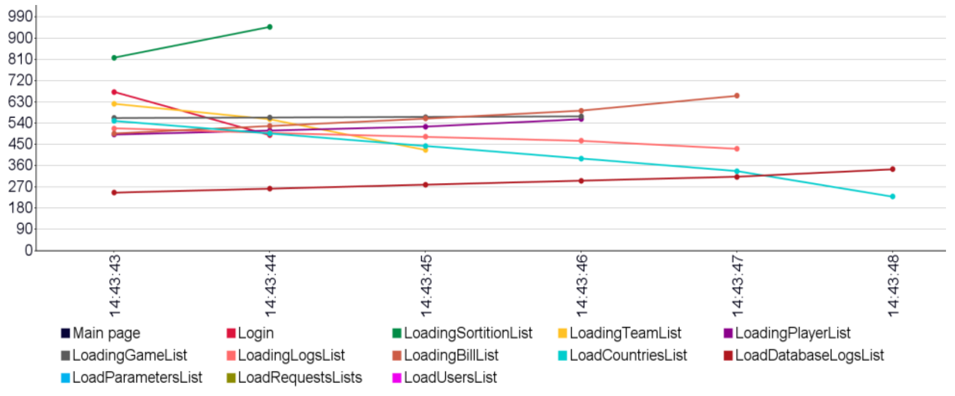
R. ID: R.2		
Description: The response time of the loading pages with lists of entities should be up to 1 second, with condition that it could be up to 100 row of records with the participation of 5 concurrent users.		
Measured attribute	**Scale**	**Goal.**
Response time	seconds	up 1 second
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Before starting, make sure that there are 100 records in each table of tested entity. 2. Open Apache jMeter test plan `SACTPerfomance.jmx.` *3.* Run thread: `LoadTestsLists.` 4. See result report (graph or report table).	
Expected results	All tests were successfully passed. Sample time of each test case is gained up to 1 second.	
Result	**OK.** Test was successfully passed.  Figure 10 Response time graph of lists loading. The maximum response time was taken – 980 milliseconds (Figure 10).	

Table 3 Tournament reports performance test case.

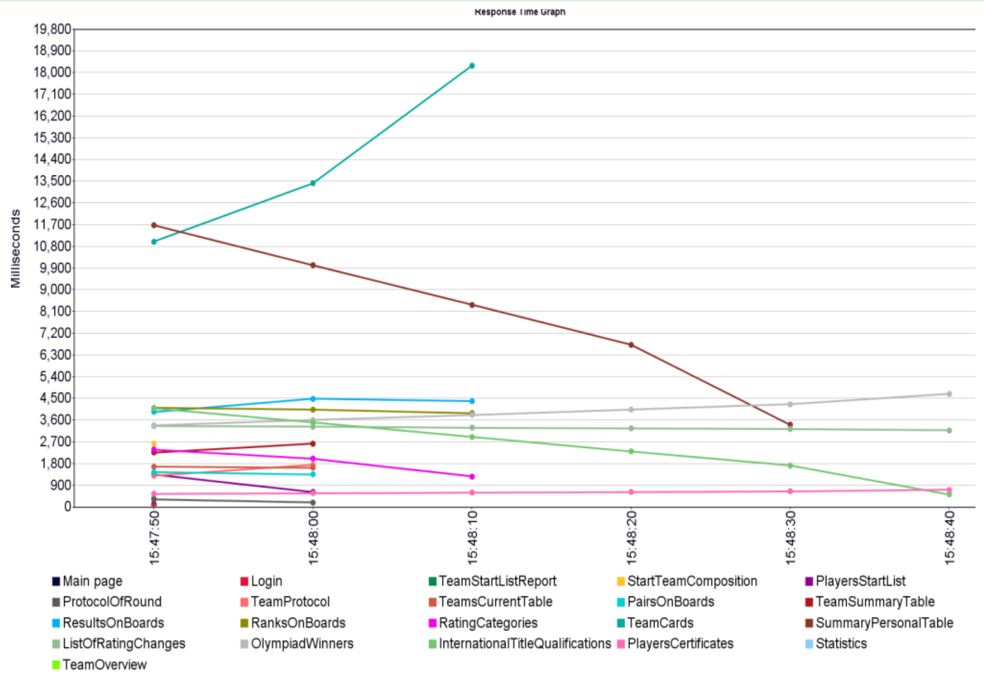
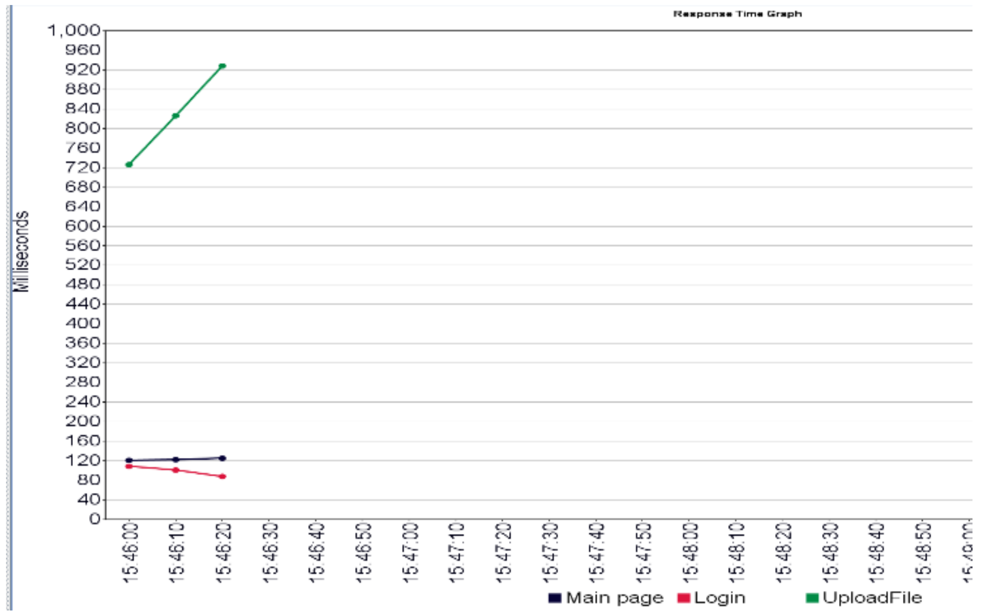
R. ID: R.3		
Description: Loading tournament reports with 150 teams per 5 players in each should take up to 20 seconds with the participation of 5 concurrent users.		
Measured attribute	**Scale**	**Goal.**
Response time	seconds	Up to 20 seconds
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Open Apache jMeter test plan `SACTPerfomance.jmx`. 2. Before starting, configure appropriate tournament with 150 teams and 5 players in each. 3. Run thread: `LoadTestsReports`. 4. See result report (graph or report table).	
Expected results	All tests were successfully passed. Sample time of each test case was gained up to 20 seconds.	
Result	**OK.** Test was successfully passed.  Figure 11Response time graph of loading tournament reports. The maximum response time was 18.5 seconds – Team cards report (Figure 11).	

Table 4 PGN file parsing performance test case.

R. ID: R.4		
Description: Parsing file with data of one chess game in PGN (Portable Game Notation) format should take up to 2 seconds with 5 concurrent users. File for testing purpose: `sample_(with_comments).pgn`		
Measured attribute	**Scale**	**Goal.**
Response time	seconds	up to 2 seconds
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Open Apache jMeter test plan `SACTPerfomance.jmx`. 2. Browse the file to the `PGNFileUploading` test case. 3. Run thread: `PGNFileUploading`. 4. See result report (graph or report table).	
Expected results	All tests were successfully passed. Sample time of each test case was gained up to 2 seconds.	
Result	**OK.** Test was successfully passed.  Figure 12 Response time graph of PGN file uploading. The maximum response time was 920 milliseconds (Figure 12).	

Capacity

Table 5 Concurrent users capacity test case.

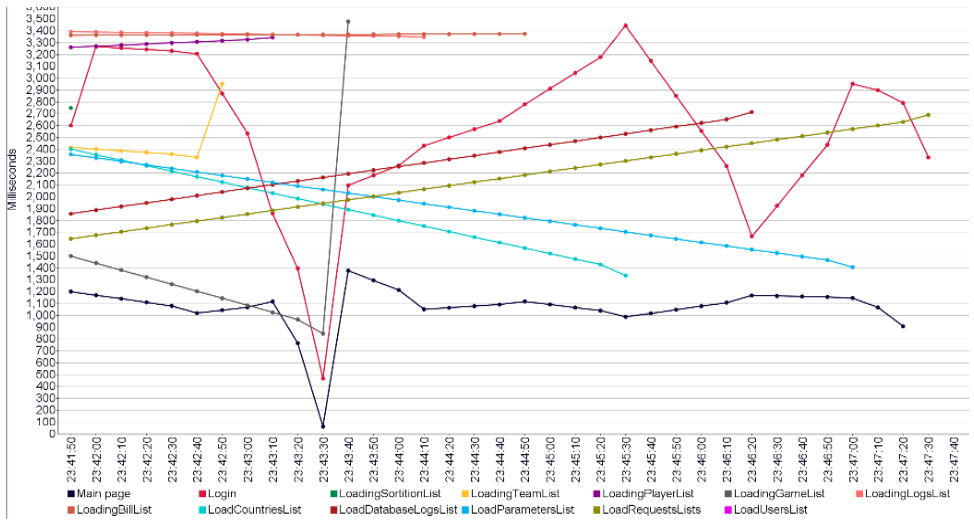
R. ID: R.5		
Description: The system should be able to process 150 concurrent users for simple requests, such as page visiting, or lists loading with the amount of 100 records.		
Measured attribute	**Scale**	**Goal.**
Transactions succeeded	%	100%
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Open Apache jMeter test plan `SACTPerfomance.jmx.` 2. Open `LoadTestLists` thread properties window and set the number of concurrent users to 150. 3. Run each test from `LoadTestsLists` one by one. 4. See the result report (graph or report table). 5. See logs.	
Expected results	The system was not failed. The system continued to support the stable work. It means that no errors and exception did not appear in log monitoring and the system was able to accept and process a further request.	
Result	**OK.** Test was successfully passed. Loading lists up to 100 records each under the load of 150 concurrent users.  Figure 13 Response time graph of lists loading with 150 concurrent users. The maximum response time was 3.5 seconds. If the system was able to process lists of entities, so it means that is able to process 150 concurrent users on simple page visiting process (Figure 13).	

Table 6 Database capacity test case.

R. ID: R.6		
Description: The system should be able to accommodate at least a 500 thousand player's records in one year. The same requirements see in Appendix 2.		
Measured attribute	**Scale**	**Goal.**
Database capacity – amount of records	records/per year	500
SMART analyse	Requirement is specific, measurable, traceable. However, it is difficult to test technically and realize it.	
Test steps	Please refer to MySQL documentation [18]. "The effective maximum table size for MySQL databases is usually determined by operating system constraints on file sizes, not by MySQL internal limits. Generally, partitioning of tables into multiple tablespace files is recommended for tables larger than 1TB in size." [18] Referring to this information, it could be concluded that it is possible to store 500 thousand player's records in MySQL table.	
Result	**OK.** Test was successfully passed. Referring to this information, it could be concluded that it is possible to store 500 thousand player's records in MySQL table.	

3.3.2 Compatibility

Coexistance

Table 7 Tomcat version 8 supporting test case.

R. ID: R.7		
Description: The system must be runnable on Tomcat server version 8.		
Measured attribute	**Scale**	**Goal.**
Available co-existence - Tomcat v.8 support	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Install tomcat v.8 on local machine and configure it. 2. Open IDE with source code and configure project to run on tomcat server. 3. Build project and run it on server. 4. Make regression testing through all system. Check that all functionality works as it should work. 5. Check errors, exceptions in console.	

Expected results	If the system was launched, no errors occurred in DevTool console, no exceptions were thrown, regression tests were passed, it means that system is runnable on Tomcat server version 8.
Result	**OK.** Test was successfully passed. The system is runnable on Tomcat v.8

Table 8 MySQL version 5 supporting test case.

R. ID: R.8		
Description: The system must support the MySQL v5. Database		
Measured attribute	**Scale**	**Goal.**
Available co-existence - MySQL support	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Make sure that local database it is configured. If it is not configured, MySQL database should be installed and scripts with initial data should be run. 2. Open file `dataSource.properties` in Properties folder. 3. Check such properties as url, username, password. They should correspond to local database properties. 4. Run application on server. 5. Make regression testing through all system. Check that all functionality works as it should work. 6. Check errors, exceptions in console.	
Expected results	If the system was launched, regression tests were passed, no exceptions and errors occurred, it means that application supports MySQL v.5 database.	
Result	**OK.** Test was successfully passed. The system supports the MySQL v5.	

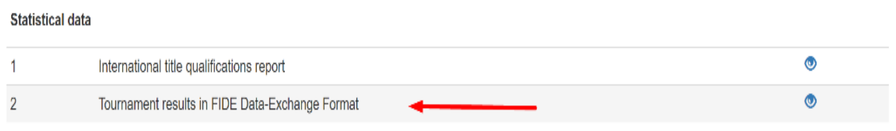
Table 9 Amazon EC2 co-existence test case.

R. ID: R.9		
Description: The system must be able to exists on Amazon EC2 (Elastic Compute Cloud)		
Measured attribute	**Scale**	**Goal.**
Available co-existence -Amazon EC2 exitance	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	

Test steps	1. Open installation guide `InstallationGuide_v[x]_[x].pdf.` 2. Install and configure all necessary components which are described in the document (Installation and configuration an application test server, installation and configuration database, installation and configuration main system's software and installation of chess application). 3. Make regression testing through all system. Check that all functionality works as it should work. 4. Check errors, exceptions in console.
Expected results	System was launched, no errors occurred and no exceptions were thrown, selenium tests were passed, it means that system exists on Amazon EC2 cloud.
Result	**OK.** Test was successfully passed. The system is able to exist on Amazon EC2.

Interoperability

Table 10 TRF format supporting test case.

R. ID: R.10		
Description: The system should support FIDE format of tournament representation (TRF - Tournament Report File) [19]. The system should be able to process and provide information about tournaments results in TRF format.		
Measured attribute	**Scale**	**Goal.**
Data exchangeability - TRF format support	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	**TRF file for appropriate tournament** 1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Go to Tournaments – tournaments with sortition. 3. Select tournament with id 136. 4. Go to the "Report" tab. 5. Download Tournament results in FIDE Data-Exchange Format report (Figure 14).  Statistical data 1 International title qualifications report 2 Tournament results in FIDE Data-Exchange Format Figure 14 Link to download Tournament results in FIDE Data-Exchange Format report. **TRF files for selected tournaments** 1. Sign in to the system by following link under ADMIN role:	

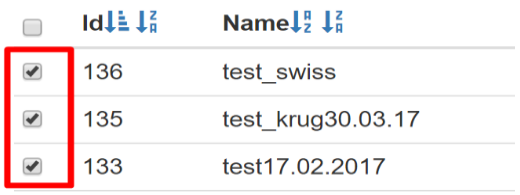
	`https://test.chessrating.agency/chess/login.do.` 2. Go to Tournaments – tournaments with sortition. 3. Select three tournaments with id: 135, 136, 133 (Figure 15).  Id · Name ☑ 136 · test_swiss ☑ 135 · test_krug30.03.17 ☑ 133 · test17.02.2017 Figure 15 Tournament selection. 4. Click “Fide Data-Exchange” button. 5. Download zip file.
Expected results	Downloaded file “Tournaments results in FIDE Data-Exchange Format report” content should correspond to format of TRF. Downloaded zip file should consist of three files in TRF format for selected tournaments. File content should correspond to format of TRF.
Result	**OK.** Test was successfully passed. The system supports TRF format.

Table 11 PGN format supporting test case.

R. ID: R.11		
Description: The system should be able to read and process game data for broadcasting from PGN (Portable Game Notation) format files [20].		
Measured attribute	**Scale**	**Goal.**
Data exchangeability - PGN format support	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Clean database tables and configure the system with the initial data. 2. Clear the directory `"/home/chess/in"`. 3. Copy file `"sample_(with comments).pgn"` from the test data in directory `"/home/chess/in"` on the application server using the FTP (File Transfer Protocol) client Win SCP. 4. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 5. Go to the System-> Channels. 6. Add a new data import channel: Channel Code: "PGN1";	

	Channel name: "PGN1"; Enable; Data Format: “PGN” Directory URL: “/home/chess/in”; Tournament: World Rapid Chess Championship 2012 File Mask: „*.pgn “; Cron string: “* * * * * ?”; Press save. 7. From the list of all channels, start the import process of two new channels. 8. Check for errors in the system: System -> Logs menu. 9. Go to Tournament-tournament. Select World Rapid Chess Championship 2012. Fill all necessary information. Enable translation. 10. Go to Broadcasting page and select World Rapid Chess Championship 2012.
Expected results	The game data from PGN file was successfully loaded and appeared on broadcasting page of the selected tournament.
Result	**OK.** Test was successfully passed. The system supports PGN format.

Table 12 TFD format supporting test case.

R. ID: R.12		
Description: The system should be able to read and process game data for broadcasting from TFD (Tape Format Requirements Document) format files.		
Measured attribute	**Scale**	**Goal.**
Data exchangeability - TFD format support	Yes/No	Yes
SMART analyse	The requirement is specific, measurable, realistic technically, realizable and traceable.	
Test steps	1. Clean database tables and configure the system with the initial data. 2. Clear the directory `"/home/chess/in".` 3. Copy file "tfd.zip" from the test data in directory `"/home/chess/in"` on the application server using the FTP client Win SCP. 4. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 5. Go to the System-> Channels. 6. Add new data import channel: Channel Code: " TFD1"; Channel name: " TFD1";	

	Enable; Data Format: “TFD” Tournament: World Rapid Chess Championship 2012 Directory URL: “/home/chess/in”; File Mask: „*.zip”; Cron string: “* * * * * ?”; Press save. 7. From the list of all channels start the import process of two new channels. 8. Check for errors in the system: System -> Logs menu. 9. Go to Tournament-tournament. Select World Rapid Chess Championship 2012. Fill all necessary information. Enable translation. 10. Go to Broadcasting page and select World Rapid Chess Championship 2012.
Expected results	The game data from TFD file was successfully loaded and appeared on broadcasting page of the selected tournament.
Result	**OK.** Test was successfully passed. The system supports TFD format.

Table 13 UCI format supporting test case.

R. ID: R.13			
Description: The system should support UCI (Universal Chess Interface) format of data exchange with chess engines like Houdini or Stockfish [21].			
Measured attribute		**Scale**	**Goal.**
Data exchangeability - UCI format support		Yes/No	Yes
SMART analyse	The requirement is specific, realistic technically, realizable and traceable.		
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Go to System -> System parameters. 3. View parameter values `«UCI_ENGINE_DIRECTORY», «UCI_ENGINE_PATH», «UCI_ENGINE_PARAMETER», «ENGINE_INIT_1»,«UCI_DEPTH_BASIC» , «UCI_DEPTH_FAST»,«UCI_MOVE_TIME_BASIC», «UCI_MOVE_TIME_FAST»,«UCI_VARIANT_COUNT», «UCI_ENGINE_CONCURRENCY», «UCI_ENGINE_PRIORITY».` 4. Go to “Broadcast matches”. 5. Choose the tournament "World Rapid Chess Championship 2012". 6. Choose any game.		

	7. Choose any move. 8. Look at the results of the analysis of the chess engine.
Expected results	Chess engine parameters should match the parameters which were set for the engine in the parameters management module window.
Result	**OK.** Test was successfully passed. The system supports UCI format.

3.3.3 Usability

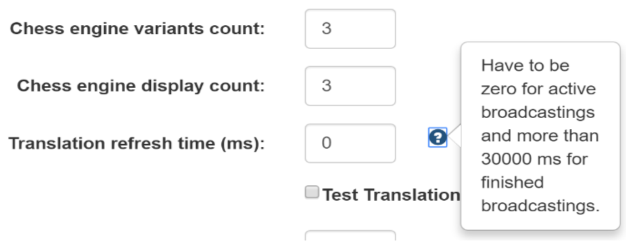
Appropriateness recognisability

No specific requirements were identified for the appropriateness recognisability quality attribute. The web application should provide main information about its main objectives for the end user. It is suggested to develop blocks on the index page, which would describe and explain the purpose of the system and system's features.

Learnability

No specific requirements were identified for the learnability quality attribute. To improve learnability of the system it could be implemented helping icons with question mark pressing on which the tooltip message appears. They could be placed near all major concepts, menu headers, and input fields.

It is implemented in some places of the system; however, tooltips could be placed near each concept, menu label, input label (Figure 16).



Chess engine variants count: 3

Chess engine display count: 3

Translation refresh time (ms): 0

Test Translation

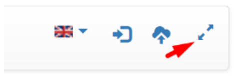
Have to be zero for active broadcastings and more than 30000 ms for finished broadcastings.

Figure 16 The tooltip example.

Moreover, it could be implemented the site map, which will illustrate all possible links and pages on the web-site in hierarchical fashion.

Operability

Table 14 Customizing possibility - container width on resize test case.

R. ID: R.14		
Description: The system should have opportunity to change the width of the main container. The width of the main container should be 100% on the resize.		
Measured attribute	**Scale**	**Goal.**
Customizing possibility -Container width on resize	width in %	100%
SMART analyse	Requirement is specific, measurable, realistic technically, realizable and traceable.	
Test steps	1. Go to `https://test.chessrating.agency/chess/login.do.` 2. On the top right corner on the main menu press resize button (Figure 17).  Figure 17 Resize button 3. Inspect "`mainContainer`" element. It should have width 100%. 4. Press one more time on the resize button. 5. Inspect "`mainContainer`" element.	
Expected results	On the resize - increase in width, the width of the container should be set to 100%, On resizing back, the width attribute should be removed and container should have its default width.	
Result	**OK.** Test was successfully passed. The system has opportunity to resize main container width.	

User error protection

Table 15 Error handling test case.

R. ID: R.15		
Description: The system should ensure correct handling of emergencies caused by all incorrect user actions: invalid format or invalid input values. In all these cases, the user should receive the appropriate messages with error explanation, all invalid input fields should be highlighted.		
Measured attribute	**Scale**	**Goal.**
Error handling in UI. Existence of alert with message, fields highlighting existence	Percent of successfully passed tests	100%
SMART analyse	Requirement is specific, measurable, realistic technically, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role:	

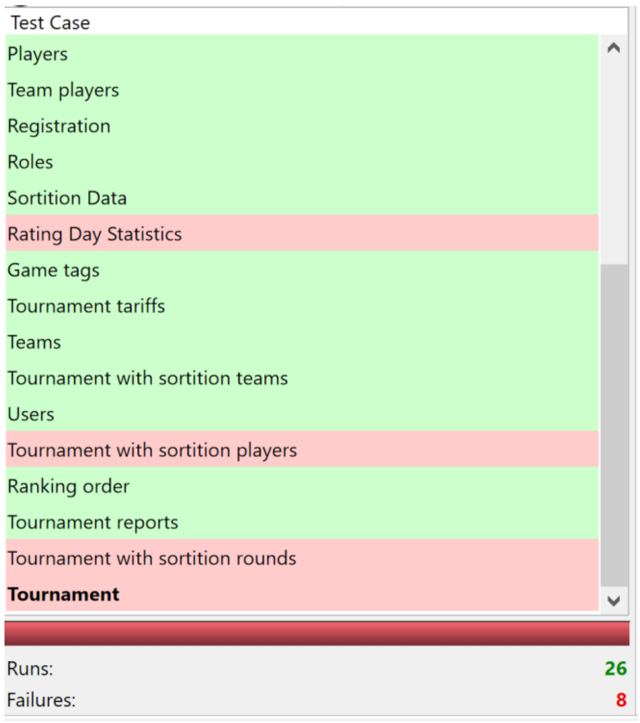
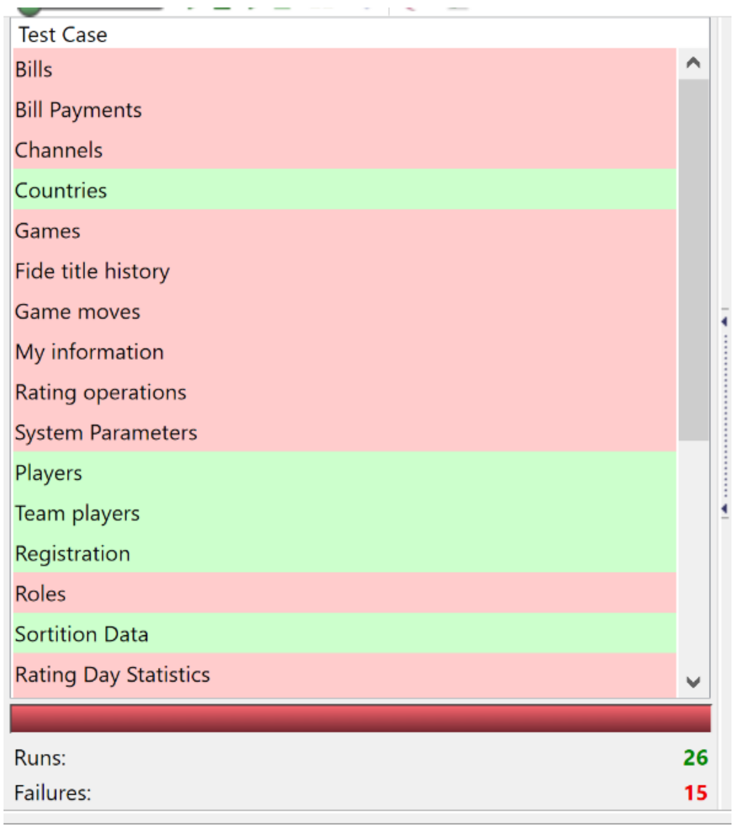
	`https://test.chessrating.agency/chess/login.do`. 2. Run Selenium tests for checking alert existence with fields highlighting (Selenium test suite: `ErrorHandling`). [Selenium tests] 3. Check if all tests were passed.
Expected results	All tests from test suite were successfully passed.
Results	**NOK.** Test was failed. It was failed 8/26 test cases (Figure 18).  Test Case Players Team players Registration Roles Sortition Data Rating Day Statistics Game tags Tournament tariffs Teams Tournament with sortition teams Users Tournament with sortition players Ranking order Tournament reports Tournament with sortition rounds **Tournament** Runs: 26 Failures: 8 Figure 18 Error handling selenium test cases.
Solutions	Go through each test case and see which field was not highlighted and where was not appear error alert. Add appropriate classes and error messages to fields.

Table 16 Required fields test case.

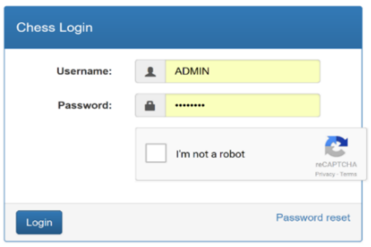
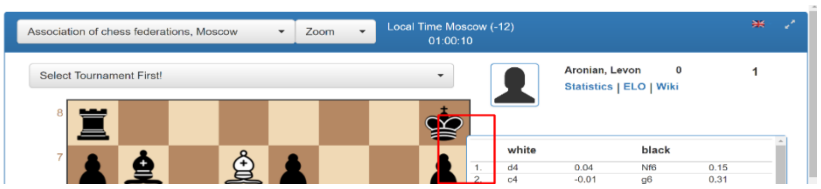
R. ID: R.16		
Description: All required input fields should have red star identifier.		
Measured attribute	**Scale**	**Goal.**
Identifier for required field existence	Percent of successfully passed tests	100%
SMART analyse	Requirement is specific, measurable, realistic technically, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do`.	

	2. Run Selenium tests for checking existence of red start identifier for required field (Selenium test suite: `RequiredFields`). [Selenium tests] 3. Check if all tests were passed.
Expected results	All tests from test suite were successfully passed.
Result	**NOK.** Test was failed. It was failed 15/26 test cases (Figure 19).  Test Case 🟥 Bills 🟥 Bill Payments 🟥 Channels 🟩 Countries 🟥 Games 🟥 Fide title history 🟥 Game moves 🟥 My information 🟥 Rating operations 🟥 System Parameters 🟩 Players 🟩 Team players 🟩 Registration 🟥 Roles 🟩 Sortition Data 🟥 Rating Day Statistics Runs: 26 Failures: 15 Figure 19 Required fields selenium test cases.
Suggestions	Go through each test case and see which field was not marked. Mark all required fields according to test cases. Try to pass all test cases in the suite.

Accessibility

Table 17 Browser scaling test case.

R. ID: R.17		
Description: Pages, forms, lists and other elements should be displayed correctly when scaling with standard browser tools (zoom) for people with problem with vision.		
Measured attribute	**Scale**	**Goal.**
Correctness view on scaling, element location, align	% of elements	100%
SMART analyse	Requirement is specific and measurable. It is hard to implement and test technically, because result is also depending on monitor resolution. In this reason, testing results could be different by following one test case. Requirement is realizable.	

Test steps	1. Open browser, find opportunity to change zoom of the page. 2. Put zoom into 125%. 3. Go through pages and check the correctness align of the elements. 4. Put zoom into 150%. 5. Go through pages and check the correctness align of the elements. 6. Put zoom into 80%. 7. Go through pages and check the correctness align of the elements.
Expected results	All elements of the UI align correctly, do not go out of bounds.
Result	**NOK.** Test was failed. On the big scaling like 125% and 150% on the login page, the ReCaptcha element goes beyond (Figure 20).  Chess Login Username: ADMIN Password: •••••••• I'm not a robot reCAPTCHA Privacy - Terms Login Password reset Figure 20 ReCaptcha element on the resizing. On the big scaling like 125% and 150%, in broadcasting mode with standard theme and zoom layout, the table with moves goes on top of the board (Figure 21).  Association of chess federations, Moscow Zoom Local Time Moscow (-12) 01:00:10 Select Tournament First! Aronian, Levon 0 1 Statistics | ELO | Wiki white black 1. d4 0.04 Nf6 0.15 2. c4 -0.01 g6 0.31 Figure 21 Move's table on the resize. On the scale, less than 100%, all elements are located correctly.
Suggestions	To fix the first error with ReCaptcha, it could be decreased the ReCapthca width or increased the login container width. To fix the second error with moves table, it should be put margin-left of the table element to 0px.

UI aesthetics

Table 18 Unity of graphical representation test case.

R. ID: R.18			
Description: The unity of the graphical representation must be observed.			
Measured attribute		**Scale**	**Goal.**
unity of the graphical representation		Yes/No	Yes
SMART analyse	Requirement is not specific enough. It should be mentioned for example, that it should be used bootstrap framework to provide the unity of graphical representation. Also, it should be mentioned, for instance, that each entity management module (lists, edit, adding new, filtering) should be unified. Requirement is technically realistic, realizable and traceable.		
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Go through each entity management module. 3. Check that all lists with data are presented in table in blue panel. 4. Check that on the top right corner of the panel it should be the filter button. 5. Check that on the left bottom corner should be list of buttons: add, remove or others (depends on appropriate rights on entity). 6. Press on filter button. In all cases, it should appear modal window with filtering. 7. Press add button, in all cases, except sortition data should appear modal window with opportunity to modify data. 8. If entity have references with other entities, in edit mode should be tabs for management sub-entities. 9. Check that all elements have the same style. For this purpose, should be used bootstrap framework and bootstrap framework elements. Go to the DevTool – Network. Refresh the page. Check that was loaded bootstrap.min.css file.		
Expected results	All elements are styled by using bootstrap classes. In system are used the same general principles of graphical representation and organization of information. Lists are in tables in blue panels, each list has filter button, and buttons for management in left bottom corner. Edit mode is made by using modal window. Switching between sub-entities is made by bootstrap tabs.		
Result	**OK.** Test was successfully passed. The unity of the graphical representation is observed.		

3.3.4 Reliability

Maturity

Table 19 Service window appearing test case.

R. ID: R.19		
Description: Service window should appear only in case of changing database characteristics. It should not take more than 15 minutes.		
Measured attribute	**Scale**	**Goal.**
Service window appearing	Reason /Time	Changes in database characteristics/15 minutes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Change database characteristics on the test server. 2. Launch application. 3. Record time how much service window appears.	
Expected results	The service window should appear after changes. It should be visible up to 15 minutes or less.	
Result	**OK.** Test was successfully passed. The service window was appeared and was visible 7 minutes.	

Availability

Table 20 Availability of the system test case.

R. ID: R.20		
Description: A system requires all of its components to be operational 24 hours a day 7 days a week, except the service window in case of changing the characteristics of the database.		
Measured attribute	**Scale**	**Goal.**
Time without failures	Hours per day	24/7
SMART requirement	Requirement is specific, measurable, realizable and traceable. However, it is not technically realistic to test this requirement, because there a lot of factors which could affect the system’s work, which cannot to be controlled by the human.	
Test steps	Daily analyses of logs and visiting the web page statistics during the week	
Expected results	At any time, the user could visit the page and use the service.	
Result	**OK.** Test was successfully passed. The system was available 24/7.	

Table 21 Life time test case.

R. ID: R.21		
Description: Average life time of the system should be at least five years.		
Measured attribute	**Scale**	**Goal.**
Life time	Years	5 years
SMART requirement	Requirement is specific, measurable, realizable and traceable. However, it is not possible to test it and give the exact result. For the lifetime of the system can affect many factors, including those which are could not be controlled by humans. Also, there are not enough time resources to check this requirement.	

Fault tolerance

Table 22 Ability to resumption to work after incorrect input test case.

R. ID: R.22		
Description: The system should keep working capacity under the improper actions of the end users. (invalid input)		
Measured attribute	**Scale**	**Goal.**
Ability to resumption to work after incorrect user's actions	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do`. 2. Open Selenium test suite: `ErrorHandling`. 3. Run Selenium tests one by one. 4. Check if all tests are executable.	
Expected results	All test cases are runnable. If the system would not be able to work after incorrect user input, the next test cases would not be able to execute. It means that if all test cases are runnable, the system keeps working capacity under incorrect input of end user.	
Result	**OK.** Test was successfully passed. The system keeps working after the improper actions of the end users.	

Recoverability

Table 23 Database backups test case.

R. ID: R.23		
Description: It should be done automatically daily database backups.		
Measured attribute	**Scale**	**Goal.**
Database backups frequency	Frequency/mode	Daily/automatically
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Login to Amazon Web Services account. 2. Go to Services – Database – RDS (Relational Database Service) in main menu. 3. Open Instances link in the sub menu. Press properties of the instance of database and check that automatic backups are enabled and frequency is daily. 4. Go to Snapshots link in the sub menu. Check the time of the backups.	
Expected results	In p.3 **-** In the settings of the instance automatic backups were enabled with frequency one per day. In p.4 each snapshot was made once time per day.	
Result	OK. Test was successfully passed. Database backups frequency is once per day and the process it automatic.	

Table 24 Dynamic allocation of tasks between server's instances test case.

R. ID: R.24		
Description: The system should provide an opportunity of the dynamic allocation of tasks between multiple running server's instances up to three pieces.		
Measured attribute	**Scale**	**Goal.**
Existence of the dynamic allocation of tasks between multiple running server's instances	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Find `Properties/ee/remarc/job.properties` file. 2. Check that `jobs.quartzDbInitializer.enabled` parameter in file is enabled. 3. Check that `org.quartz.jobStore.tablePrefix` parameter is provided. 4. Check the existence of the script:	

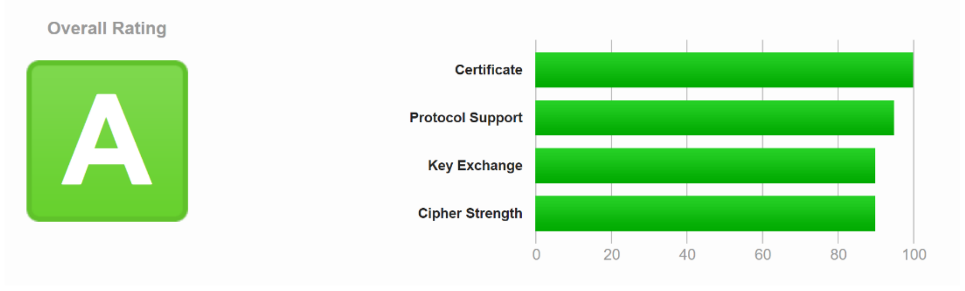
	`chess-impl\src\main\resources\quartz\mysql\create-quartz-tables.sql`. 5. Login to the database using MySQL query browser or MySQL workbench. 6. Check the existence of specific database tables with the same prefix which was configured in p.4.
Expected results	In p.1 file exists. In p.2 parameter is enabled. In p.3 parameter is provided. In p.4 script exists. In p.6 tables exist.
Result	**OK.** Test was successfully passed. Yes, there is an opportunity of the dynamic allocation of tasks.

3.3.5 Security

Integrity

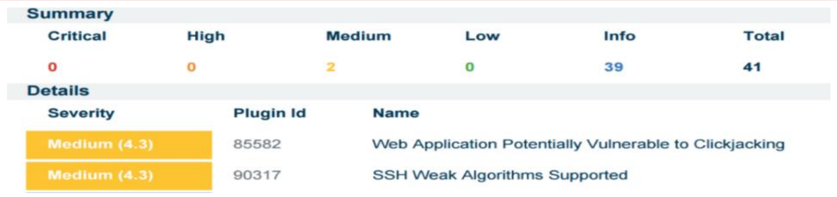
Table 25 SSL configuration test case.

R. ID: R.25		
Description: It should be done deep analysis of the configuration of SSL web server on the public Internet using SSL testing tool [14]. Overall rating should be at minimum A. Each factor which involves overall rating should be evaluated at least 90%.		
Measured attribute	**Scale**	**Goal.**
Configuration of SSL web server on the public Internet	Overal rating (A+,A,B,C,D,E,F)	Min: A.

SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.
Test steps	1. Open SSL server test tool [14]. 2. Fill hostname field with test`.chessrating.agency` and press submit button.
Expected results	Overall rating should be at minimum A. Each factor which involves overall rating should be evaluated at least 90%.
Result	**OK**. Test was successfully passed.  Figure 22 SSL server test overall report.

	Certificate and protocol support has the maximum assessment – 100% and 95%, other factors have assessment – 90% (Figure 22). Some of cipher suites have weak encryption/decryption algorithm and Public Key Pinning is partial. All trust paths are not pinned.
Suggestions	Consider encryption/decryption algorithm of cipher suites, to make them stronger. Make all Public Keys pinned.

Table 26 Vulnerabilities detection test case.

R. ID: R.26		
Description: Application should successfully pass all tests of detecting vulnerabilities of Nessus scanner. No vulnerabilities with low, medium and high marker should be detected.		
Measured attribute	**Scale**	**Goal.**
Number of detected vulnerabilities (low, medium, high)	Quantity	0
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Launch Nessus tool. 2. Add new scan. 3. Select basic scan. 4. Insert all information in basic tab. Put in targets field the system's host name. 5. In assessment tab select scanner type – "Scan for all web vulnerabilities".	
Expected results	No vulnerabilities with low, medium and high marker should be detected.	
Result	**NOK**. Test was failed.  Summary Critical: 0 · High: 0 · Medium: 2 · Low: 0 · Info: 39 · Total: 41 Details Severity · Plugin Id · Name Medium (4.3) · 85582 · Web Application Potentially Vulnerable to Clickjacking Medium (4.3) · 90317 · SSH Weak Algorithms Supported Figure 23 Nessus scanner results. It was detected two medium priority problems (Figure 23). 1. Web Application Potentially Vulnerable to Clickjacking "Nessus has detected that the remote SSH (Secure Shell) server is configured to use the Arcfour stream cipher or no cipher at all." [22] 2. SSH Weak Algorithms Supported "The remote web server does not set an X-Frame-Options response header or a Content-Security-Policy 'frame-ancestors' response header in all content responses. This could potentially expose the site to a clickjacking or UI redress attack, in which an attacker can trick a user	

	into clicking an area of the vulnerable page that is different than what the user perceives the page to be. This can result in a user performing fraudulent or malicious transactions.” [22]
Suggestions	Suggestions are provided by Nessus tool: For SSH Weak Algorithms problem: “Contact the vendor or consult product documentation to remove the weak ciphers.” [22] For Clickjacking problem: “Return the X-Frame-Options or Content-Security-Policy (with the 'frame-ancestors' directive) HTTP (Hypertext Transfer Protocol) header with the page's response. This prevents the page's content from being rendered by another site when using the frame or iframe HTML (Hypertext Markup Language) tags.” [22]

Confidentiality

Table 27 Administrator role test case.

R. ID: R.27		
Description: Information granting to users should consider user roles. For administrator, all menu pieces are available: broadcast matches, tournaments with sub-menus (tournaments for broadcasting, tournament with sortition, teams, players, games, entity logs), accounting with sub menus (bills, tournaments tariffs), system with sub-menus (roles, users, countries, database logs, system parameters, system jobs, channels, request statistic), my account (change password, my information, my comments). All tournaments are available. Addition: Federation administrator of the system has the same privileges as the administrator, expect that only tournaments and players of the same federation as the administrator is are available.		
Measured attribute	**Scale**	**Goal.**
Abidance of privileges for the admin and federation admin roles	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Go through all links of the main menu and compare them with the expected results. 3. Login to the database using MySQL query browser or MySQL workbench. 4. Go to the Tournaments-Tournaments with sortition and compare the amount of tournaments with amount of records in `sortition_data` table in the database. 5. Select appropriate tournament from the list, go to modification mode. For federation admin Point 1, 2 and 3 are the same. Start with point 4.	

	4. Run following script `SELECT * FROM users;` and get the `COUNTRY` value of the user under which you have logged in. 5. Suppose the country value is EST. Go to the Tournaments-Tournaments with sortition and analyse results.
Expected results	In p.2 should be available all links: broadcast matches, tournaments with sub-menus (tournaments for broadcasting, tournament with sortition, teams, players, games, entity logs), accounting with sub menus (bills, tournaments tariffs), system with sub-menus (roles, users, countries, database logs, system parameters, system jobs, channels, request statistic), my account (change password, my information, my comments). In p.3 should be available all tournaments. The number of tournaments in the system's UI for administrator should be the same as in database In p.4 tabs should be visible. Admin could modify any tournament data. In p.5 it should appear only those tournaments that are declared by country attribute in Estonia.
Result	**OK.** Test was successfully passed. Yes, the Admin and federation admin roles have those privileges which are described in the requirement description.

Table 28 Organizer role test case.

R. ID: R.28		
Description: Information granting to users should consider user roles. For organizer role are available following navigation aspects: broadcast matches, tournaments (tournaments with sortition), accounting (bills), my account (change password, my information, my comments). In menu tournaments with sortition are available only those tournaments, where organizer is involved in. In tournament management module for organizer are available only tournament data, players, and registration tabs. The organizer could create a and change the tournament general data.		
Measured attribute	**Scale**	**Goal.**
Abidance of privileges for the organizer role	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ORGANIZER role: `https://test.chessrating.agency/chess/login.do.` 2. Go through all links of the main menu and compare them with the expected results. 3. Login to the database using MySQL query browser or MySQL workbench. 4. Run following script `SELECT * FROM users;` and get the `USER_IR` value of the user under which you have logged in.	

	5. Suppose the user id is 2. Run following script with appropriate user id instead of "2" `SELECT * FROM sortition_data s where ORGANIZERS like "%2%";` 6. Go to the Tournaments-Tournaments with Sortition and compare the amount of tournaments in the system's UI with the amount of records received by running the script presented in p.5. 7. Select appropriate tournament from the list, go to modification mode. 8. Modify tournament data.
Expected results	In p.2 should be visible following links: broadcast matches, tournaments (tournaments with sortition), accounting (bills), my account (change password, my information, my comments). In p.6 tournaments in the system's UI should be the same as in the database. In p.7 it should be available only three tabs: tournament data, players and registration. It is impossible to add new players. In p.8 it should be available to modify tournament's data. The operation should be successfully done.
Result	**OK.** Test was successfully passed. Yes, the organizer role has those privileges which are described in the requirement description.

Table 29 Arbiter role test case.

R. ID: R.29			
Description: Information granting to users should consider user roles. For arbiter role are available: broadcast matches, tournaments (tournaments with sortition), my account (change password, my information, my comments). For arbiter are available only these tournaments were user belongs to arbiter list. All tabs in the tournament management module are available, except registration. Addition: Deputy arbiter has the same privileges as arbiter. For deputy arbiter are available only these tournaments were user belongs to deputy arbiter list.			
Measured attribute		**Scale**	**Goal.**
Abidance of privileges for the arbiter and deputy arbiter roles		Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.		
Test steps	1. Sign in to the system by following link under ARBITER role: `https://test.chessrating.agency/chess/login.do.` 2. Go through all links of the main menu and compare them with the expected results. 3. Login to the database using MySQL query browser or MySQL workbench.		

	4. Run following script
	`SELECT * FROM users;`
	and get the `USER_IR` value of the user under which you have logged in.
	5. Suppose the user id is 2. Run following script with appropriate user id instead of "2"
	`SELECT * FROM sortition_data s where CHIEF_ARBITERS like "%2%";`
	For deputy arbiter:
	`SELECT * FROM sortition_data s where DEPUTY_ARBITERS like "%n%";`
	6. Go to the Tournaments-Tournaments with Sortition and compare the amount of tournaments in the system's UI with the amount of records received by running the script presented in p.5.
	7. Select appropriate tournament from the list, go to modification mode.
	8. Modify tournament data.

Expected results	In p.2 should be available following links: broadcast matches, tournaments (tournaments with sortition), my account (change password, my information, my comments). In p.6 tournaments in the system's UI should be the same as in the database. In p.7 All tournaments tabs are available (teams, players, ranking order, rounds, forbidden pairs, absence rules, reports) In p.8 it should be available to modify tournament's data. The operation should be successfully done.
Result	**OK.** Test was successfully passed. Yes, the arbiter and deputy arbiter roles have those privileges which are described in the requirement description.

Table 30 Password complexity test case.

R. ID: R.30		
Description: Password must be at least 8 characters in length. Password must contain at least 1 digit character. Password must contain at least 1 alphabetical character. Password must contain at least 1 uppercase character.		
Measured attribute	**Scale**	**Goal.**
Password complexity	Length (amount of characters), amount of digit characters, amount of alphabetical character, amount of uppercase character	Length: at least 8 char; digit char: at least 1, alphabetical char.: at least 1, uppercase char.: at least 1
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.`	

	2. Go to the reset password page: `https://test.chessrating.agency/chess/passwordReset.do.` 3. Insert e-mail for ADMIN user. (It could be changed in database) and press submit button. 4. Insert “test” in password fields and press change button. 5. Insert “test12” in password fields and press change button. 6. Insert “Test12” in password fields and press change button. 7. Insert “Test1234” in password fields and press change button.
Expected results	In p.3 should appear error message, because password consists only of letters. In p.4 should appear error message, because password consists only of small letters and digits and length is less than 8. In p.5 should appear error message, because password’s length is less than 8. In p.6 password should be successfully changed.
Result	**NOK.** Test was failed. The password was successfully changed by execution p.4. It means that at the moment, the constraint is combination of letters and numbers.
Suggestions	Implement the missing password complexity constraints and rules.

Table 31 Private information access test case.

R. ID: R.30		
Description: Anonymous users cannot get access to private functionality or for functionality with specific permissions.		
Measured attribute	**Scale**	**Goal.**
Private information browsing requires to be authenticated	Yes/No	Yes

Test steps	1. Check if you are not logged in the system. 2. Try to connect URL which should be not visible for no-authenticated users. 3. Analyze the source code. All controllers and business logic methods which are process private information should have `@PreAuthorize("isFullyAuthenticated()")` annotation. 4. Analyse `spring-security.xml` file.
Expected results	In p.2 The system should prevent non-authorized access; it should redirect to the login page. In p.3 all methods in controllers and business logic which process private information should have `@PreAuthorize("isFullyAuthenticated()")` annotation In p.4 the enforcement mechanism should deny all access by default, it should be presented the list of exceptions for public access.
Result	**OK.** Test was successfully passed. Yes, the private information browsing requires to be authenticated.

Table 32 Password hashing test case.

R. ID: R.32		
Description: The password should be stored in hashed mode in the database.		
Measured attribute	**Scale**	**Goal.**
Existence of password hashing	Yes/No	Yes
Test steps	1. Login to the database using MySQL query browser or MySQL workbench. 2. Open all records from users table. `SELECT * FROM users u;` 3. Analyse `PASSWORD` column	
Expected results	The password should be hashed and should not be stored as a plain text.	
Result	**OK.** Test was successfully passed. Yes, the password is hashed.	

Table 33 Complexity of changing password test case.

R. ID: R.33		
Description: Changing password process should include the old password, the new password, and a password confirmation.		
Measured attribute	**Scale**	**Goal.**
Complexity of changing password process	Fields	Old password/new password/password confirmation
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Go to `https://test.chessrating.agency/chess/changePassword.do.`	
Expected results	Three fields are presented: old password, new password, password confirmation.	
Result	**NOK.** Test was failed. In changing password form is required to fill only two fields: new password and password confirmation.	
Suggestions	Add field for old password.	

Table 34 Session timeout test case.

R. ID: R.34		
Description: The session timeout should be configured to 600 minutes.		
Measured attribute	**Scale**	**Goal.**
Session timeout	Seconds	600
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Open source code and open `web.xml` file. 2. Found `<session-timeout>` field and analyse value.	
Expected results	The session timeout is set to 600 minutes.	
Result	**OK.** Test was passed successfully. The session timeout is configured to 600 minutes.	

Table 35 CSRF token test case.

R. ID: R.35		
Description: Application uses strong random anti-CSRF (Cross-Site Request Forgery) tokens. It should be unique per user sessions; it should be the large random value.		
Measured attribute	**Scale**	**Goal.**
CSRF tokens existence	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Open DevTools – Network session and check the POST request to the `/chess/j_spring_security_check,` analyse post data. 3. Modify any entity. For instance, go to the Tournament-Tournaments with sortition. Select tournament and modify it. Analyse post request data.	
Expected results	CSRF token is unique per user session. It represents the large random value. CSRF token is sent by each POST request. The server rejects the requested action if the CSRF token fails validation.	
Result	**OK.** Test was passed successfully. Yes, the system uses CSRF token.	

Accountability

Table 36 The presence of monitored fields in the table test case.

R. ID: R.36		
Description: Each table in the database should have fields like `CREATED_BY, CREATED_DATE, UPDATED_BY, UPDATED_DATE` to monitor creation and modifications in entities.		
Measured attribute	**Scale**	**Goal.**
The presence of monitored fields in the table (`CREATED_BY, CREATED_DATE, UPDATED_BY, UPDATED_DATE`)	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Login to the database using MySQL query browser or MySQL workbench. 2. Go through all the tables in the database. 3. Check that each table has columns `CREATED_BY, CREATED_DATE, UPDATED_BY, UPDATED_DATE.`	
Expected results	Each table in the database has four columns: `CREATED_BY, CREATED_DATE, UPDATED_BY, UPDATED_DATE`.	
Result	**OK.** Test was successfully passed. Yes, required fields are presented in each table.	

Table 37 Ability to track previous player entity state test case.

R. ID: R.37		
Description: For the Player entity it should be kept the previous entity state before modifications in separate `players_history` table.		
Measured attribute	**Scale**	**Goal.**
The presence of `players_history` table. Ability to track previous player entity state.	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Login to the database using MySQL query browser or MySQL workbench. 2. Check that player_hitory table exist: run script: `SHOW TABLES LIKE 'players_history'.` 3. If there is a row in the result set, it means that table exists in the database.	

	4. Login to the system under ADMIN role and go to Tournaments-Players. Select player to modification. 5. Modify player's tagName. 6. Open `players_history` table, find modified player record. 7. Check the `TAG_NAME` and `TO_DATE` value.
Expected results	`players_history` table should exist in the database. After player modification, this record in `players_history` should contain previous information about a player before modification procedure was made. `TAG_NAME` value should be equal to the initial. `TO_DATE` column should contain the modification date.
Result	**OK.** Test was successfully passed. The `players_history` table presences and traces player entity modifications.

Table 38 Presence of Logs monitoring test case.

R. ID: R.38		
Description: All logs should be saved in the database in logs table and be accessible for system's administrator in UI of the system.		
Measured attribute	**Scale**	**Goal.**
Presence of Logs monitoring	Yes/No	Yes

SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Login to the database using MySQL query browser or MySQL workbench. 3. Select all records from logs table ``SELECT * FROM `logs`;`` 4. Analyse the last records from the logs table sorted discerningly by created date. 5. In system UI go to System-Logs. 6. Analyse the first record sorted descending by created date.
Expected results	The record in logs table should represent the information about the last authentication. It should involve the date and time of the authentication, who was made it, what client was used. The log record in the system's UI should be the same as in database and illustrates the last authentication information.
Result	**OK.** Test was successfully passed. Yes, in the system is available logs monitoring.

Table 39 Presence of entity logs monitoring test case.

R. ID: R.39		
Description: All actions on entities must be written to the table `entity_logs` in database and be accessible for system's administrator in the UI of the system.		
Measured attribute	**Scale**	**Goal.**
Presence of entity logs monitoring	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Sign in to the system by following link under ADMIN role: `https://test.chessrating.agency/chess/login.do.` 2. Change team name. Go to Tournaments-Teams. 3. Open any team. Change tag name field. Save entity. 4. Login to the database using MySQL query browser or MySQL workbench. 5. Select all records from logs table `` SELECT * FROM `entity_logs`; `` 6. Analyse the first record from the logs table sorted discerningly by created date. 7. In system UI go to Tournaments-Logs. 8. Analyse the first record sorted descending by created date.	
Expected results	The record in the database should contain modified entity declaration in `ENTITY_NAME` column. It this test case it should be "Team". The record in the UI should match the record in the database.	
Result	**OK.** Test was successfully passed. Yes, in the system is available entity logs monitoring.	

Authentity

Table 40 Presence of ReCaptcha.

R. ID: R.40		
Description: The user should pass ReCaptcha during registration, authorization, and password reset process to confirm that user is a human, not a bot.		
Measured attribute	**Scale**	**Goal.**
Presence of ReCaptcha	Yes/No	Yes
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Go to registration page `https://test.chessrating.agency/chess/userRegister.do.` 2. Fill all necessary fields, except ReCaptcha. Analyse result.	

	3. Fill all necessary fields with ReCaptcha. Analyse result. 4. Go to login page `https://test.chessrating.agency/chess/login.do.` 5. Fill all necessary fields, except ReCaptcha. Analyse result. 6. Fill all necessary fields with ReCaptcha. Analyse result. 7. Go to password reset page `https://test.chessrating.agency/chess/passwordReset.do.` 8. Fill all necessary fields, except ReCaptcha. Analyse result. 9. Fill all necessary fields with ReCaptcha. Analyse result.
Expected results	On each page where ReCaptcha is required, in the case of a non-fulfilment of it, an error message will appear, and if ReCaptcha was entered correctly - the operation will be successfully passed (with condition that all other fields have been filled correctly).
Result	**OK.** Test was successfully passed. Yes, the application uses ReCaptcha technology.

3.3.6 Maintainability

Reusability

Table 41 Presence of interfaces test case.

R. ID: R.41		
Description: Each service with business logic and DAO (Data Access Object) class in the system should implement the appropriate interface to make system more reusable.		
Measured attribute	**Scale**	**Goal.**
Interfaces amount for services and DAO	%	100%
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Go to `chess-impl\src\main\java\ee\remarc` folder and check that each service and dao class implements appropriate interface.	
Expected results	Each service and dao class implements appropriate interface.	
Result	**OK.** Test was successfully passed. 100% of services and dao classes implement the interface.	

Table 42 Polymorphism existence test case.

R. ID: R.42		
Description: To ensure reusability of the system it should be used polymorphism OOP (Object-oriented programming) principle. It should be created base classes for services, for entity services, for models and DAO classes, controllers and public controllers, commands and they should be extended by their sub-types.		
Measured attribute	**Scale**	**Goal.**
Polymorphism existence	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Press “Open Type” button in menu bar (Ctrl + Shift + T). 2. Find `EntityDAO` class and check that dao classes extend it. 3. Find `EntityDAO` class and check that model classes extend it. 4. Find `EntityServiceImpl` class and check that entity management service classes extend it. 5. Find `AbstractServiceImpl` class and check that service classes extend it. 6. Find `EntityManagementCommand` class and check that management command classes extend it. 7. Find `EntityManagementController` class and check that management controller classes extend it. 8. Find `EntityListController` class and check that entity lists controller classes extend it. 9. Find `EntityListCommand` class and check that list command classes extend it. 10. Find `AbstractEntityManagementRestController` class and check that REST controller classes extend it. 11. Find `EntityEditCommand` class and check that edit command classes extend it. 12. Find `EntityEditController` class and check that edit controller classes extend it. 13. Find `ActionController` class and check that action controller classes extend it. 14. Find `SimpleActionController` class and check that action controller extends it.	
Expected results	All of the listed classes are present in the system and are extended by appropriate classes.	
Result	**OK.** Test was successfully passed. Yes, the polymorphism principle is used in the system.	

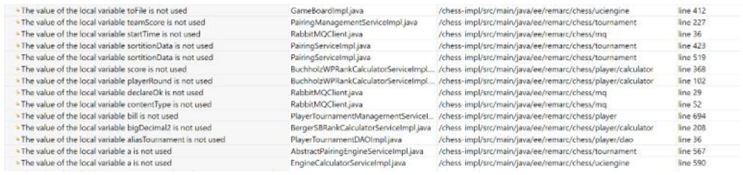
Analysability

Table 43 Comments coverage test case.

R. ID: R.43		
Description: Source code should be self-documented. All services and controllers should be documented/commented on 80%. It involves comments for class headers, which are represent the aim of the controller/service or contain a reference to the documentation and comments for methods which are described the method mission.		
Measured attribute	**Scale**	**Goal.**
Comments coverage	%	80%
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Go through all controllers in chess-web project. Check that each controller should have in the header the reference to the documentation. Check that methods in the controller should be also commented. 2. Go through all services with business logic and calculations. Classes should be commented. Each method should be commented.	
Expected results	90% of controllers and services are commented. They have an explanation about for what they are responsible, in comments are identified main objectives of the service/controller (if they realize appropriate point in documentation – it should be a reference to it). Each separate method is also should be documented.	
Result	**NOK.** Test was failed. Most of controllers and services have not any comments. Most of them have only auto-generated comments.	
Suggestions	Go through developing tickets and documentation and write comments for controllers and services. Detected controllers with missing comments: `BillManagementController, PayPalChargeController, RefundController, StripeChargeController, TransactProController, BroadcastinMatchesController, ChannelListActionsController, CheckGameController, GameListActionsController, TestGameCurrentMoveController, ImageController,ChessGameFileUploadController, ChangePasswordController, LoginController,PasswordResetController, PlayerApplicationPublicController, PlayerListActionsController, PlayerTournamentImportController, PlayerTournamentListFullSimple1Controller, PublicPlayerIDAssignmentRestController, PublicPlayerRestController,AbstractTournamentReportController, InternationalTitleQualificationReportController, KnockOutGridReportController, OlympiadWinnersReportController, PublicReferencesRestController, PublicTeamRestController,`	

	`TeamScoreRestController, DefaultResultManagementController, PairingController, PairingManagementController, TournamentAbsenceRuleManagementController`
	Detected services with missing comments:
	`GameChangeEventGeneratorImpl, RoundEventGenerator, PairingsStatisticsReportServiceImpl,`
	`RanksOnBoardsReportServiceImpl, RatingCategoriesReportServiceImpl, TitlesStatisticsReportServiceImpl`

Table 44 Unnecessary (dead) public java code detection test case

R. ID: R.44		
Description: The application should not involve unnecessary (dead) public java code. For example, public classes, methods or fields which have no references.		
Measured attribute	**Scale**	**Goal.**
Unnecessary (dead) public java code	Number of elements	0 elements
SMART analyse	Provided requirement is specific and detailed enough. It is measurable. It is realistic technically and realizable.	
Test steps	For the execution of this test case, the Unnecessary Code Detector plugin for Eclipse should be installed. 1. Go to preferences – UCDetector. In drop down menu select "Unused only". This setting will help to detect unused code. Press Apply and OK. 2. Press the right click on the main project folder `(chess-j2ee).` Select UCDetector – Detect unnecessary code. 3. See results after analyse in console or in the report file. By the default settings, the report file located in the folder `"ucdetector_reports"` in the current workspace.	
Expected results	It should be found 0 unnecessary, unused, dead elements in the analysed source code.	
Result	**NOK.** Test was failed. All public classes and methods have references and used in the application. However, it was found some local variables and fields which were not used in the method (Figure 24).  The value of the local variable toFile is not used GameBoardImpl.java /chess-impl/src/main/java/ee/remarc/chess/uciengine line 412 The value of the local variable teamScore is not used PairingManagementServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/tournament line 227 The value of the local variable startTime is not used RabbitMQClient.java /chess-impl/src/main/java/ee/remarc/chess/mq line 36 The value of the local variable sortitionData is not used PairingServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/tournament line 423 The value of the local variable sortitionData is not used PairingServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/tournament line 519 The value of the local variable score is not used BuchholzWPRankCalculatorServiceImpl.... /chess-impl/src/main/java/ee/remarc/chess/player/calculator line 368 The value of the local variable playerRound is not used BuchholzWPRankCalculatorServiceImpl.... /chess-impl/src/main/java/ee/remarc/chess/player/calculator line 102 The value of the local variable declareOk is not used RabbitMQClient.java /chess-impl/src/main/java/ee/remarc/chess/mq line 29 The value of the local variable contentType is not used RabbitMQClient.java /chess-impl/src/main/java/ee/remarc/chess/mq line 52 The value of the local variable bill is not used PlayerTournamentManagementServiceI... /chess-impl/src/main/java/ee/remarc/chess/player line 694 The value of the local variable bigDecimal2 is not used BergerSBRankCalculatorServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/player/calculator line 208 The value of the local variable aliasTournament is not used PlayerTournamentDAOImpl.java /chess-impl/src/main/java/ee/remarc/chess/player/dao line 36 The value of the local variable a is not used AbstractPairingEngineServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/tournament line 567 The value of the local variable a is not used EngineCalculatorServiceImpl.java /chess-impl/src/main/java/ee/remarc/chess/uciengine line 590 Figure 24 UCDetector analysis results.	
Suggestions	Install UCDetector plugin to the Eclipse and run this test case. Go through all unused local variables and make code refactoring.	

Modifiability

Table 45 Correctness of modification test case.

R. ID: R.45		
Description: The architectural solution should ensure the development and modification of the System's components throughout the entire period of exploitation. It should be provided the ability to add new pages, menu items, etc. without affecting the previous parts and modules.		
Measured attribute	**Scale**	**Goal.**
Correctness of modification	Number of failures after modification	0
SMART analyse	Requirement is not specific enough. It covers a wide range and does not have clear bounds. Provided requirement is measurable, it is correctness of modifications. Requirement is technically realistic, realizable and traceable.	
Test steps	1. Open source code in IDE and prepare it for modifications. 2. Create new entity management controller. Configure it in action config file. Add tile definition for it. (For any entity). 3. Create jsp page for configured view. 4. Add link to main menu.	
Expected results	Application work stable after adding new page and menu link.	
Result	**OK.** Test was successfully passed. No failures were found after adding new module.	

Testability

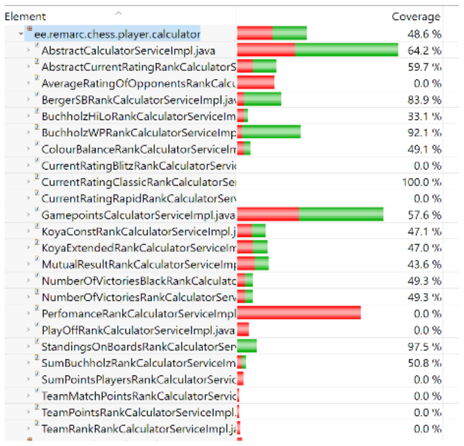
Table 46 Pairings algorithms test coverage test case.

R. ID: R.46		
Description: The delivered system should include unit tests that ensure 95% instructions coverage of pairings algorithms		
Measured attribute	**Scale**	**Goal.**
Test coverage	% of instructions	95%
SMART analyse	Requirement is specific, measurable, realizable, technically realistic and traceable.	
Test steps	For the execution of this test case, the EclEmma – Java Code Coverage plugin to Eclipse should be installed. 1. Go to `chess-impl\src\test\java\ee\remarc\chess\pairing\` folder. 2. Press right click on pairing folder – Coverage As – Junit Test. 3. Open coverage window. Open	

	`chess-impl/ee.remarc.chess.`tournament package. 4. See the coverage of Pairing engine services classes.
Expected results	The instruction coverage by tests should be 95%
Result	**NOK.** Test was failed. The test covered instructions of pairing engine services is 0%. The system has not any Junit tests for pairings algorithms.
Suggestions	Create for each pairing algorithm the test case.

Table 47 Calculation algorithms test coverage test case.

R. ID: R.47		
Description: The delivered system should include unit tests that ensure 75% instructions coverage of the parameters calculation algorithms**.**		
Measured attribute	**Scale**	**Goal.**
Test coverage	% of instructions	75%

SMART analyse	Requirement is specific, measurable, realizable and technically realistic.
Test steps	For the execution of this test case, the EclEmma – Java Code Coverage plugin to Eclipse should be installed. 1. Go to `chess-impl\src\test\java\ee\remarc\chess\player\calculator` folder. 2. Press right click on calculator folder – Coverage As – Junit Test 3. Open coverage window. Open chess-impl/ee.remarc.chess.player.calculator package 4. See the coverage percent of selected package
Expected results	The instruction coverage by tests should be 75%
Result	**NOK.** Test was failed.  Figure 25 Test instruction coverage of calculation algorithms.

Element	Coverage
ee.remarc.chess.player.calculator	48.6 %
AbstractCalculatorServiceImpl.java	64.2 %
AbstractCurrentRatingRankCalculatorS	59.7 %
AverageRatingOfOpponentsRankCalcu	0.0 %
BergerSBRankCalculatorServiceImpl.jav	83.9 %
BuchholzHiLoRankCalculatorServiceIm	33.1 %
BuchholzWPRankCalculatorServiceImp	92.1 %
ColourBalanceRankCalculatorServiceIm	49.1 %
CurrentRatingBlitzRankCalculatorServic	0.0 %
CurrentRatingClassicRankCalculatorSe	100.0 %
CurrentRatingRapidRankCalculatorServ	0.0 %
GamepointsCalculatorServiceImpl.java	57.6 %
KoyaConstRankCalculatorServiceImpl.j	47.1 %
KoyaExtendedRankCalculatorServiceIm	47.0 %
MutualResultRankCalculatorServiceImp	43.6 %
NumberOfVictoriesBlackRankCalculato	49.3 %
NumberOfVictoriesRankCalculatorServ	49.3 %
PerfomanceRankCalculatorServiceImpl	0.0 %
PlayOffRankCalculatorServiceImpl.java	0.0 %
StandingsOnBoardsRankCalculatorSer	97.5 %
SumBuchholzRankCalculatorServiceIm	50.8 %
SumPointsPlayersRankCalculatorServic	0.0 %
TeamMatchPointsRankCalculatorServic	0.0 %
TeamPointsRankCalculatorServiceImpl.	0.0 %
TeamRankRankCalculatorServiceImpl.ja	0.0 %

	The test instruction coverage percent of calculation algorithms is 48.6%. Some of algorithms have not any tests (Figure 25).
Suggestions	Write test cases for classes which have not them. See what instructions have not covered by existed test, complete existing test cases.

3.3.7 Portability

Adaptability

Table 48 Completeness of the translation test case.

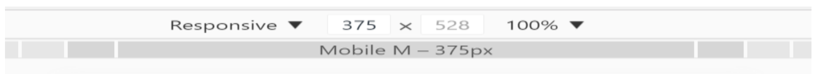
R. ID: R.48		
Description: All system should be translated into Russian and English language. It means that each message should be translated into two languages.		
Measured attribute	**Scale**	**Goal.**
Localization. Completeness of the translation	%	100%
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Go to messages files by following path*:* `Sources/AppServer/chess-web/src/main/resources.` 2. Open files: `messages_ru.properties` and `messages_en.properties.` 3. Check that each message code appears in all two files and is translated in accordance with the required language (Russian in `messages_ru.properties` and English in `messages_ru.properties`) Only specific terms could be translated only in English.	
Expected results	100% of messages have translation on Russian and English	
Result	**OK.** Test was successfully passed. 100% of messages are translated on both languages.	

Table 49 Browsers compatibility test case.

R. ID: R.49		
Description: The system should maintain the stable work on all the most popular browsers like Chrome: Version 32+, Firefox: Version 3.5+, IE Version 9+, Safari: Version 4.0+ and Edge 14.		
Measured attribute	**Scale**	**Goal.**
Browsers compatibility	Yes/No	Yes

SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.
Test steps	1. It should be downloaded all listed browsers, or it could be used the online tool which simulates the work of a lot of different browsers with all browser's features. 2. Go through application. Try to view/add/remove entities. 3. Check elements location. 4. Open browser developer tool and check errors in console.
Expected results	All functionality works correctly on all provided browsers. Elements has correctly located. No errors appear in console.
Result	**OK.** Test was successfully passed. The system is compatible with listed browsers.

Table 50 Responsiveness for mobile screens test case.

R. ID: R.50		
Description: System's UI should be adaptive to mobile phones and tabs screens based on resolution and viewport.		
Measured attribute	**Scale**	**Goal.**
Responsiveness for mobile screens	% of elements correctly and comfortable displayed	100%
SMART analyse	Requirement is not specific enough. It should be specified what screen resolutions or mobile models should be supported. Requirement is measurable, technically realistic, realizable and traceable.	
Test steps	1. Open Chrome browser (v. 49+) and launch `https://test.chessrating.agency/chess/login.do` 2. Open DevTools: Right-click on a page element and select Inspect. 3. Press on the left top corner the mobile devices button. ("toggle device toolbar") or press Ctrl + Shift + M). 4. On the top of the screen select mode "responsive". 5. Select one of the screen resolutions provided by Chrome DevTool: Mobile S (320px in width), Mobile M (375px in width), Mobile L (425px in width), Tablet (768px in width). 6. The main screen resolutions could be selected on the top of the page (Figure 26).  Figure 26 screen resolutions scale.	

Expected results	All elements are aligned properly, clickable areas are suitable, elements do not run into the edges of the screen, text is readable, the horizontal scroll bar is not appearing on all screen resolutions.
Result	**NOK.** Test was failed. The system part which is for registered users is not mobile responsive. For mobile screen resolutions, the main menu after authorization is not available at all. The horizontal scrolling appears.
Suggestions	Make administrative part of the system responsive using bootstrap responsive classes.

Installability

Table 51 Presentence of installation guide test case.

R. ID: R.51		
Description: The installation guide should be presented and consists of step by step detailed instruction. The screenshots should be attached.		
Measured attribute	**Scale**	**Goal.**
Documentation availability	Yes/No	Yes
SMART analyse	Requirement is specific, measurable, technically realistic, realizable and traceable.	
Test steps	1. Open the folder with source code. Go by following path: `Sources\Docs\Guides`. 2. In folder, should be found file `InstallationGuide_v[x]_[x].pdf`. 3. Go through the file. It should be the step by step installation instruction with screenshots.	
Expected results	File with detailed step by step installation instruction located in `Sources\Docs\Guides` folder. The file contains 23 pages and each step is following by the screenshot.	
Result	**OK.** Test was successfully passed. Yes, documentation for installation is available.	

Table 52 Ease of updating test case.

R. ID: R.52		
Description: When a new version of the main system is released, it shall be possible to upgrade to it from any previous version using the appropriate script for it by one action.		
Measured attribute	**Scale**	**Goal.**

Ease of updating		number of actions	1
SMART requirement	Requirement is specific, measurable, technically realistic, realizable and traceable.		
Test steps	1. Go to `home/chess` directory in the test server and find `install` file. 2. Run `install` file.		
Expected results	The install file is in home/chess directory. The file contains the set of commands which will update the system's state. After running the file, the system would be updated to the new version.		
Result	**OK.** Test was successfully passed. It took only one action to completely update the system - run the file.		

3.4 SYSTEM QUALITY EVALUATION

In this chapter presents an evaluation of the system's quality according to test results. The system's quality evaluation could not be optimal, in reason that the scope was limited by a number of requirements which were presented by the customer. To make the full quality assessment each quality factor should be tested deeply because provided list of requirements has not covered all quality aspects of the product. Despite this fact, this quality evaluation template could be used in further researches about quality control. Below are presented the summary tests report. The first report illustrates the total amount of test cases (Table 53). And the second report includes statistics of the test cases execution among quality characteristics. (Table 54).

Table 53 Total test cases statistics

Test cases planned	Test cases executed	Test cases passed	Test cases failed
52	**51**	**40**	**11**

Table 54 Total test cases statistics by characteristics

Characteristic	Total test cases executed	Test cases passed	Test failed failed
Performance Efficiency	**6**	**6**	**0**
Time behaviour	4	4	0
Capacity	2	2	0
Compatibility	**7**	**7**	**0**

Coexistance	3	3	0
Interoperability	4	4	0
Usability	**5**	**2**	**3**
Operability	1	1	0
User error protection	2	0	2
Accessibility	1	0	1
UI aesthetics	1	1	0
Reliability	**5**	**5**	**0**
Maturity	1	1	0
Availability	1	1	0
Fault tolerance	1	1	0
Recoverability	2	2	0
Security	**16**	**13**	**3**
Integrity	2	1	1
Confidentiality	9	7	2
Accountability	4	4	0
Authentity	1	1	0
Maintainability	**7**	**3**	**4**
Reusability	2	2	0
Analysability	2	0	2
Modifiability	1	1	0
Testability	2	0	2
Portability	**5**	**4**	**1**
Adaptability	3	2	1
Installability	2	2	0

A task of developing all test cases took, on average, one and a half months, of which two weeks were used for writing automated tests. Each test case could be executed approximately in ten minutes. (Working day – 8h, 5 days per week).

Based on the results, the author may claim that tests which failed could be explained by aspects of usability, security, maintainability and portability. On the other hand, the successful tests showed such quality aspects as performance efficiency, compatibility and reliability.

The usability aspect needs to be improved. The team of developers should pay more attention to handling of errors, the selenium tests might help fix them quickly. The test case on the browser scaling requirement has less impact and could be ambiguous as everything depends on the initial screen resolution. Moreover, except for two elements, everything is located correctly and does not disturb the work.

The security quality is very important in non-functional testing. Two tests in confidentiality module failed, however, those tests concerned the password complexity and complexity of a password changing process. Such requirements have a lower priority than others, but they should ensure passing the test cases as soon as possible. Also, the vulnerability scanner has detected two gaps in the system. These kinds of vulnerabilities are not so dangerous, but they still need removal. Developers should follow the suggestions which were provided by Nessus tool.

The maintainability quality failed more times than succeeded. The detected gaps do not lead to any heavy losses on the side of the end user and client. Despite this fact, the maintainability attribute is important for the developers' team. The correct implementation of this feature in the system might save a lot of time for developers.

In portability section, a problem occurred with the adaptivity module. The application is not responsive or adaptive for mobile and tablets devices. Without implementation of that requirement the system still works, however, the number of users and the number of positive reviews might be decreased.

In the conclusion, it could be said that the system passed the majority tests, and all tests which might have led to big problems successfully achieved the set goals. However, the software lacks in quality as it still has some weaknesses and gaps which must be improved. To get a quality product, all failed tests should be corrected, then the quality control should be repeated until 100% of tests will reach positive results. There are two ways for doing that. One way – to repeat the failed tests only and continue with regression testing to make sure that the modification has not affected the operation of the entire system. Another option is to repeat all tested cases, also followed with regression tests. The second option would be more reliable.

4 RESULTS AND ANALYSIS

In conclusion, it could be said that it was possible to obtain answers to the posed questions. The main research questions were as follows:

- How to implement the non-functional product quality control in practice?
- What stages does the product quality control include?

Below are presented the answers and the flow of work that helped to obtain those answers is described.

The research contains step by step analysis of the correct implementation of the software quality control. [Software product quality control] In this thesis, all stages of the process were examined in detail. It was clarified that product quality control consists of five main stages: setting the quality model, giving the requirements specification, planning the testing procedure, collecting and evaluating results. Further, all parts of the product quality control were implemented in practice within the real-life system.

As a case study, it was selected the System of the Administration of Chess Tournaments – SACT. A quality model was chosen in compliance with ISO/IEC 25010:2011 from the SQuaRE project. The requirements were collected from the customer – the Chess Rating Agency Ltd. The requirements were analysed for testability feature by using SMART technique and were assigned to each quality attribute from the quality model. In total, 54 requirements were received, of which 4 were untestable. The next stage included planning of testing procedure. Firstly, test cases for each requirement were developed using the experience-based test design technique; i.e. test cases were developed relying on tester's experience, motivation, knowledge and resources. Therefore, it was difficult to identify the one and only solution as a basis for the testing procedure. Each requirement was associated with its own testing flow and adequate supporting tools, selected at the tester's discretion as the most optimal and relevant. Prior to starting with data collecting, the environmental and all other necessary case study tools were set up. In the scope of the project the following tools were used: Apache JMeter, Selenium IDE for recording Selenium tests, SSL Server Test tool, Nessus vulnerability scanner, and code source analysing plugins for Eclipse IDE. The instruments were chosen according to the authors background knowledge and experience.

When the system got ready for testing, all test cases were executed and the results reported. The evaluation of quality was made on the grounds of the obtained data. According to the evaluation, on that stage, the system failed to comply with the requirements of quality product, and had to be improved. Anyway, such system still may be used in production since the detected weaknesses are not critical, moreover, they do not cause any destructive effects. It were put out the suggestions concerning improvement of the failed aspects of the system. Upon fixing all gaps and problems, the product quality control should be repeated. To sum up, all identified goals were achieved. The non-functional product quality control was successfully implemented in practice in the real case study. All parts of the control were observed and applied.

5 FUTURE WORK

The potential future research directions are related to investigation of non-functional test cases, based on requirements not provided by the client. For instance, there is an opportunity to go deeper into each quality characteristics and complement the current analysis with new tests.   Another example is related with the security aspect. The system could be checked for each requirement from Application Security Verification Standard 3.0.1 of the OWASP (Open Web Application Security Project) [23].

Moreover, the scope of the research might be extended if the non-functional process quality control is included. Besides, when non-functional control is fully implemented, it would be considered the functional part of the system and the similar research on the functional part of the system could be conducted. In addition, in future this research allows to do the similar work with another project using this thesis as a template since the formulated requirements and the test steps prescribed in the test cases have identical features relevant for any web application.

CONCLUSION

In every software development process, it is very important not to forget to pay sufficient attention to the quality of the product. In the scope of this project, the non-functional quality control of the System of the Administration of Chess Tournaments (SACT) was performed based on the customer's requirements.

The customer provided the non-functional requirements. Each requirement was analysed for SMART feature, was correlated with the quality attribute from ISO/IEC 25010:2011 model and was tested. For test cases, different testing techniques, as manual as automated, were developed by an appropriate case study tool. During the testing procedure, weaknesses and gaps were found. For this, cases were provided with appropriate suggestions to solve the problem.

Using collected data, after testing it was made the quality evaluation of the system according to non-functional product quality characteristics of the ISO/IEC 25010:2011 quality model. As it was analysed, the system has some weak spots, which need to be improved. However, the system has successfully passed all major tests witch have the big influence on the quality of the system, they are related to such attributes as security, reliability, performance efficiency and compatibility. Only usability, analysability and portability and some non-critical parts of security characteristics needed to be revised. These characteristics does not have a big influence on the system’s work. The system is able to maintain the stable and secure work with gaps in this area without significant losses.

As the result, the developers team accepted the non-functional quality control of the SACT and expressed their acknowledgement for the work done. It was stated that they would start correcting weaknesses immediately

KOKKUVÕTE

Tarkvara arendamisel on tähtis, et kunagi ei unustataks toote kvaliteedile piisavalt tähelepanu pööramist. Käesolevas projektis kontrolliti Maleturniiride haldussüsteemi (SACT – the System of the Administration of Chess Tournaments) mittefunktsionaalsete nõuete vastavust kliendi nõuetele.

Klient esitas mittefunktsionaalsed nõuded. Igat nõuet analüüsiti SMART raamistikku silmas pidades, seoti ISO/IEC 25010:2011 kvaliteedi mudeli omadustega ning kõigile nõuetele vastavust testiti. Testimisel kasutati erinevaid meetodeid, sealhulgas manuaalset ja automaatset tesitmist vastavalt töös kasutatud vahenditele. Katsetamisel avastati nõrkused ja puudused. Nende lahendamiseks esitati asjakohased soovitused.

Pärast katsetamist hinnati kogutud andmete põhjal süsteemi kvaliteeti. Nagu eelpool juba nimetati, siis analüüs tuvastas süsteemis mitmeid puudusi. Ent süsteem läbis edukalt kõik katsed, mis on seotud põhinõuetega, näiteks turvalisuse, vastupidavuse, jõudluse ja ühilduvuse katsed. Täiustamist vajab veel ainult kasutatavus, analüüsimisvõime, kaasaskantavus ja mõned väjemolulised osad turvalisuses. Süsteem isegi ilmnenud puudustega võimeline säilitama tõrgeteta ja turvalise töö ilma märkimisväärsete kahjudeta.

Uurimuse tulemusena kinnitas arendusmeeskond SACT mittefunktsionaalse kvaliteedi probleeme ning tänas töö autorit tehtud töö eest. Nad mainisid, et hakkavad kohe puudusi paranda

REFERENCES

[1] Runeson Per, Höst Martin, Rainer Austen, Regnell Björn, Case study research in software engineering. Guidelines and Example, John Wiley & Sons, Inc., 2012.

[2] ISO/IEC/IEEE, “ISO/IEC/IEEE 24765:2010: Systems and software engineering – vocabulary,” ISO/IEC/IEEE, 2010.

[3] Tepandi Jaak, Tšukrejeva Jekaterina, Vassiljev Stanislav, Haug Pille, “Software quality, processes, and standards Basic concepts,” in *Tallinn University of Technology, Department of Informatics*, 2016.

[4] Project Management Institute, A guide to the Project Management Body of Knowledge, Project Management Institute, 2000.

[5] Wagner Stefan, Software Product Quality Control, Springer-Verlag, Springer-Verlag.

[6] “Estonian Centre for Standardisation,” [Online]. Available: https://www.evs.ee/.

[7] ISO/IEC, “ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models ISO/IEC 2011,” ISO/IECl, 2011.

[8] Mannion Mike, Keepence Barry, “SMART Requirements,” Napier University, 1999.

[9] Markvardt Maili, “Basics of Requirements Elicitation. Focus on non-functional requirements,” in *Tallinn University of Technology*, 2016.

[10] Glenford J. Myers, The Art of Software Testing - Second Edition, John Wiley & Sons, Inc., 2004.

[11] “Apache JMeter,” [Online]. Available: http://jmeter.apache.org/.

[12] “Performance Testing Guidance for Web Applications - Chapter 2 – Types of Performance Testing,” 2007. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb924357.aspx.

[13] “Nessus (software),” [Online]. Available: https://en.wikipedia.org/wiki/Nessus_(software).

[14] “SSL testing tool,” [Online]. Available: https://www.ssllabs.com/ssltest/index.html.

[15] “Selenium framework,” [Online]. Available: http://www.seleniumhq.org/.

[16] “EclLemma,” [Online]. Available: http://www.eclemma.org/.

[17] “UCDetector,” [Online]. Available: http://www.ucdetector.org/.

[18] “MySQL 5.7 Reference Manual - C.10.3 Limits on Table Size,” [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/table-size-limit.html.

[19] “FIDE format of tournament representation,” [Online]. Available: https://ratings.fide.com/download/fidexchg.txt.

[20] “PGN format,” [Online]. Available: http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm.

[21] “UCI protocol,” [Online]. Available: http://wbec-ridderkerk.nl/html/UCIProtocol.html.

[22] “Nessus scanner,” [Online]. Available: https://cloud.tenable.com.

[23] “Application Security Verification Standard 3.0.1,” [Online]. Available: https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf.

[24] Galin Daniel, Software quality assurance. From theory to implementation, Pearson Education, 2004.

[25] Gerald D. Everett, Raymond McLeod, Jr., Software Testing. Testing Across the Entire Software Development Life Cycle, John Wiley & Sons, Inc., 2007.

[26] “International Software Testing Qualifications Board – ISTQB,” [Online]. Available: http://www.istqb.org/.

[27] ISO/IEC, “ISO/IEC 25023 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality,” ISO/IEC.

APPENDICES

Appendix 1

Automated tests

1. Selenium Required fields test suite
https://www.dropbox.com/s/d5oihysgxoczayn/required_fields.zip?dl=0
2. Selenium Error handling test suite
https://www.dropbox.com/s/zpvapatuy51xsik/errors_handling.zip?dl=0
3. Apache JMeter performance tests
https://www.dropbox.com/s/7ysnwovahagl0x8/SACTPerfomance.jmx?dl=0
4. PGN sample file
https://www.dropbox.com/s/5wx9lced0l3cy22/sample_%28with%20comments%29.pgn
?dl=0

Appendix 2

Additional (similar) requirements

R.53	The system should be able to accommodate at least a 10 thousand tournament's records in one year.
R.54	The system should be able to accommodate at least a 500 thousand player's records in one year.
R.55	The system should be able to accommodate at least 30 games records per player in one year.