Design of testing data visualization from the robotic system for the YSoft SafeQ print management

Master’s Thesis

Bc. Barbora Šišková

Brno, Fall 2018
Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Barbora Šišková

Advisor: doc. Ing. RNDr. Barbora Bühnová, PhD.
Acknowledgements

I would like to acknowledge my advisor, doc. Ing. RNDr. Barbora Bůhmová, PhD., for her constructive advice and time dedicated to this thesis. My thanks also go to Ing. Jakub Pavláč for his consultations and inspiration and the whole RQA team for their willingness to help during the implementation of this thesis. Last but not least, I would like to express my gratitude to my family and friends who supported and encouraged me throughout my studies.
Abstract

The goal of this thesis is to adjust the testing environment of YSoft SafeQ to simulate the production environment and to design and implement the visualization of the test results from such customized testing environment. First, some aspects of the real-world use of YSoft SafeQ that may influence its performance are discussed, together with the tools enabling the simulation of these aspects in the testing environment. The database changes required to store data from such testing are identified, and a new database schema is designed and implemented. The last part of the thesis is dedicated to visualization of the test results from a simulated production environment with regards to the business needs of Y Soft Corporation.
Keywords

YSoft SafeQ, data visualization, PostgreSQL, Entity Framework, DotVVM, robotic testing, production environment simulation
## Contents

1 **Introduction** 1

2 **Types of software testing** 3
   2.1 *Software testing* .................................. 3
   2.2 *Levels of software testing* ..................... 3
   2.3 *Functional and nonfunctional testing* .......... 4
   2.4 *Ways of testing software* ........................ 4

3 **Data visualization** 7
   3.1 *Data visualization and its importance* ........ 7
   3.2 *Multivariate data visualization* ................ 8

4 **Robotic testing of YSoft SafeQ** 9
   4.1 *YSoft SafeQ* ...................................... 9
   4.2 *Robotic Arm Project* .............................. 10
   4.3 *Testing environment setup* ....................... 13
   4.4 *Test results visualization* ....................... 14

5 **Analysis of production environment** 17
   5.1 *Production environment of YSoft SafeQ* ........ 17
   5.2 *Testing environment customization implementation* 19
      5.2.1 *Latency and packet loss* .................... 20
      5.2.2 *Number of users* ............................. 22
      5.2.3 *Persistent load* ............................. 22

6 **Database for test results** 23
   6.1 *Current database schema* ........................ 23
   6.2 *Requirements* .................................... 25
   6.3 *New database schema* ............................. 25

7 **Schema and data migration implementation** 29
   7.1 *Tools* ............................................. 29
      7.1.1 *PostgreSQL* .................................. 29
      7.1.2 *Entity Framework* ......................... 29
   7.2 *Implementation* .................................. 31
      7.2.1 *Database schema migration* ................. 31
8 Test results visualization

8.1 Tools

8.1.1 DotVVM framework

8.2 Design and implementation

8.2.1 Business needs

8.2.2 Visualized data

8.2.3 Common components of implemented charts

8.2.4 Implemented charts and their use

8.3 Other visualization options

8.3.1 Not implemented proposals

8.3.2 Future possibilities

9 Conclusion

Bibliography

A Source code information

A.1 Database schema and migration

A.2 Data visualization implementation
1 Introduction

YSoft SafeQ is a print management suite developed by Y Soft Corporation. Its primary goal is to provide companies with an efficient and flexible printing solution. It is a platform for centralized print management installed in printers instead of the native software developed by the printer manufacturer.

Robotic Arm Project (also known as RQA project) deals with the black box testing of YSoft SafeQ using a robotic system to automate manual, repetitive tasks. This way, the user interface of YSoft SafeQ terminals is tested in the physical environment. The team working on this project takes care of all aspects of the testing process - they set up the robots, write testing scenarios, prepare the YSoft SafeQ installation and process the test results from the robotic system.

Production environment of YSoft SafeQ customers differs from the testing, laboratory environment set up in the RQA project. Various kinds of problems appear in the real-world use of this solution, be it hardware issues (network, CPU, etc.) or the actual load on YSoft SafeQ (especially in case of big companies). Therefore, a requirement to simulate such production environment in the testing process and evaluate what are the consequences of its problems on the end user arises.

Multiple goals are defined for this thesis. The first goal is to perform an analysis of the production environment with the aim of identifying some variables that might influence the performance of YSoft SafeQ and subsequently simulate these variables in the testing environment of the robotic system. Another objective is to design and implement a new database schema to enable storing information about the customized testing environment together with the test results. Finally, the last objective of the thesis comprises of the design and implementation of the visualization of test results from the customized environment.

All practical parts of this thesis are my work. A laboratory testing environment necessary for these parts of the thesis was set up by people from the RQA team. However, all customizations of the testing environment were integrated into this environment by me. Also, all implementation parts of this thesis were developed by me and con-
1. Introduction

sulted with the RQA team members (namely with those focusing on databases and web application).

The first two chapters are dedicated to the theoretical basis of the thesis. The importance of proper software testing and comprehensive data visualization is emphasized. Chapter 4 describes YSoft SafeQ, RQA project and the current state of testing and visualization of test results in more detail. Chapter 5 deals with the analysis of the production environment of YSoft SafeQ and its simulation in the testing environment. Subsequently, chapters 6 and 7 discuss the changes necessary for storing data from such testing environment and the implementation of these changes. Finally, chapter 8 deals with the design and implementation of the visualization of the obtained test results.
2 Types of software testing

This chapter provides an introduction to software testing. Various classifications of software testing are explained and the importance of testing is emphasized. The chapter also points out some advantages of different approaches to testing. Although the testing process used in the RQA project is described in latter parts of this thesis, its advantages are apparent based on the information provided in this chapter.

2.1 Software testing

The goal of the software testing process is to deliver a correctly working software. According to the ANSI/IEEE 1059 software testing is defined as the process of analysis of software with the objective of detecting the differences between existing and required conditions and evaluating the features of given software. In short, testing identifies to what extent the system meets its requirements [1].

2.2 Levels of software testing

Testing may be performed at various levels. Testing the whole software after each small change in the source code would be time-consuming and resource demanding which is the reason why multiple levels of testing exist. Usually, the process is as follows – first, testing at the lowest level is performed; once tests pass, bigger units are tested. This way the testing process continues until the whole software is tested. Table 2.1 gives an overview of testing levels.

<table>
<thead>
<tr>
<th>Level of testing</th>
<th>What is tested?</th>
<th>Who is testing?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit testing</td>
<td>Individual components</td>
<td>Developer</td>
</tr>
<tr>
<td>Integration testing</td>
<td>Component groups</td>
<td>Developer/tester</td>
</tr>
<tr>
<td>System testing</td>
<td>Integrated system</td>
<td>Tester</td>
</tr>
<tr>
<td>Acceptance testing</td>
<td>Final System</td>
<td>End user/client</td>
</tr>
</tbody>
</table>

Table 2.1: Testing levels [2]
2. Types of software testing

2.3 Functional and nonfunctional testing

Functional testing

Functional testing can be done at various testing levels. It verifies that the system under test works with conformance with functional requirements on the system. Functional requirements describe the desired behavior of the developed software. The requirements documentation should clearly state what the system should do. Functional testing checks “what the system does” and the results are compared with specified requirements.

When functional testing happens at the system level, end user’s environment is simulated very often. It is frequently done as black box testing which implies that no assumptions about implementation details need to be made [3].

Nonfunctional testing

Nonfunctional testing takes into consideration nonfunctional requirements on the system. These requirements are sometimes also called “quality attributes”. Basically, they describe “how well the system works”. Some examples of such qualities are performance, compatibility, security, reliability, robustness, etc.

The evaluation of test results in nonfunctional testing differs from functional testing. While for functional tests, we know the expected output and it is possible to check whether the real output is the same, nonfunctional tests check the readiness of a system as per nonfunctional parameters. An example would be as follows: “After logging in, the user is redirected to their profile in less than 2 seconds.”

2.4 Ways of testing software

Two major approaches exist when it comes to ways of testing software, manual and automated. Both these approaches have their advantages as well as disadvantages. Some tests can be performed both manually and automatically. An example of such a test would be some user interface testing scenario. Both a tester clicking in an application as well as some testing tool (for instance, Selenium framework) can perform such test.
2. Types of software testing

Manual testing
In general, manual testing requires a person who performs the testing. Tester tries different test scenarios to check that the way the product works is acceptable. The main advantage of manual testing is that the person working with the software might observe some defects that an automated testing tool would miss. Another significant benefit of human-performed testing is exploratory testing. This means that the person testing the software tries various scenarios, without having exact steps that need to be followed. Compared to some automation, more ways of using the software may be covered.

Although there are advantages to manual testing, negative aspects of this approach do not fit in well with today’s requirement for frequent delivery of new versions of the software that requires early feedback from testing. Some of the disadvantages are these:

- Big resources requirements - when the test needs to be performed multiple times, there always has to be someone to actually perform it.

- Human errors - when manual testers follow some test scenario, they can always make a mistake in some of its steps and this might lead to inaccurate results.

- Waste of potential - when manual testing is done based on some predefined test scenarios and tester always follows the same steps, and no creative work is required, automated tests can do the same job.

Automated testing
Automated testing is a process where software and hardware tools are used to automate the testing. Tests are written by testers and then run by testing tools. The main advantages of automated testing are the following:

- Tests run in a repeatable manner.

- Tests can run outside working hours and continuously.

- Results of testing are automatically recorded.
2. Types of software testing

- Tests can be triggered after each change to check for regressions.
- Testers can focus on more creative tasks.
- Usable for both repeatable functional tests as well as performance, regression, and load testing.

It is worth mentioning that despite all advantages of automated testing, some drawbacks exist, including:

- No exploratory testing is possible.
- Initial costs may be very high.
- Each change in the UI requires modification of tests.
- Human consideration is not involved in UI testing (for example, no user-friendliness is checked) [4].
3 Data visualization

This chapter is dedicated to a short introduction to the data visualization. A lot of data is produced during testing of YSoft SafeQ and stored in a database. The data is stored in the form of thousands of records and therefore, is not easily interpretable by a human. Thus, the need for data visualization arises.

To begin with, the first section explains what the data visualization is and why it is used. The second section defines the multivariate data and provides an overview of the techniques for visualization of such data.

3.1 Data visualization and its importance

Visualization is the communication of information using graphical representations. Its value resides in the fact that image interpretation is performed in parallel within the human perceptual system, while the analysis of text is slowed down by sequential processing of written information [5]. Thus, the data visualization helps to understand the information quickly, identify relationships and patterns in the data and communicate the insights to others [6].

As defined by Ward, Grinstein and Keim in [5], a successful visualization is such that accurately conveys the information towards the targeted audience with regards to the purpose of the visualization. For each set of data, many ways to visualize it exist. However, some of the general requirements on effective visualization include:

- comprehensible,
- contains intuitive controls to customize views,
- has appropriate information density, and
- provides labels and legends [5].

While the choice of visualization design depends on the data being processed, the final design should take into account the target audience and the purpose of the visualization.
3. Data visualization

3.2 Multivariate data visualization

Multivariate data is defined as the data in which the analysis is based on more than two variables per one observation [7]. Special techniques exist for visualization of such data. They can be divided into the following groups:

- point-based techniques (e.g. scatterplots),
- line-based techniques (e.g. line charts),
- region-based techniques (e.g. bar charts/histograms),
- hybrid techniques (combination of the techniques mentioned above) [5].

The obvious advantage of using multivariate data visualization is the possibility to plot several categories/groups into the same chart (multiple values of y-axis are assigned to one value of x-axis, one y-axis value for each category/group). The main concern in such approach is the separation of different groups in case there is many of them (for example, many groups with similar y-axis values in a line-chart). This can be solved by using an interactive chart where a subset of groups/categories can be shown [8].
4 Robotic testing of YSoft SafeQ

The beginning of this chapter focuses on the description of YSoft SafeQ, including the basic scenario of its real-world use. The following section is dedicated to the Robotic Arm Project (further referred to as RQA project), which deals with the black box testing of the YSoft SafeQ user interface in an automated way.

Building on that, the second half of the chapter is dedicated to the testing of YSoft SafeQ, specifically to the testing environment setup and to the visualization of the test results. First, the setup of the testing environment used for testing by the robotic system is discussed, mentioning the possible enhancements to this environment setup. Afterward, the visualization of test results currently present in the web application of the RQA project is discussed.

4.1 YSoft SafeQ

YSoft SafeQ is a platform for centralized print management. It is distributed in various versions (offering different sets of features) to fit the needs of companies of all sizes and all kinds of sectors. It is installed in the printers or MFDs\(^1\) instead of the native software distributed by the printer/MFD manufacturers.

The YSoft SafeQ solution uses a server for remote management of various functionalities related to MFDs. Each device that the user wants to use for printing via YSoft SafeQ has to be connected to a YSoft SafeQ server. When a user needs to print a job, they send it to their YSoft SafeQ server logged in with their credentials. The job is ready to be printed on any device that is connected to that specific server. When it comes to an actual printing of a job, the user has to utilize a terminal - either an embedded YSoft SafeQ terminal or a YSoft SafeQ Mobile Terminal App of the specific printer. Both these terminals have a similar user interface.

The first step the user has to do when printing is the authentication. One of the following authentication methods is usually required - PIN

---

1. Multi-function device - device incorporating various features (scanning, printing, copying)
4. Robotic testing of YSoft SafeQ

code in the user interface, username and password or an ID card. Once the user is authenticated, they can select some jobs. The terminal sends a request to the server and provided there are no errors, the selected jobs are executed. Please, note that this workflow is only a simplified description of YSoft SafeQ and the actual implementation is more complex.

Some of the benefits of using print management compared to native software in printers and MFDs are:

- significant reduction of costs thanks to optimizations of the print environment and using print policies,
- increased document security thanks to the required authentication of users,
- fact-based print audit for a better understanding of overall usage trends, and
- pull-printing from any device that allows people to print without print drivers or IT assistance in the company environment [9].

For a basic understanding of YSoft SafeQ, a simple workflow of one use case is presented in figure 4.1. Please, note that this is a simplified illustration of YSoft SafeQ usage from the end user’s point of view. For this thesis, the essential part of the workflow shown in this figure is the user’s interaction with the YSoft terminal, as it also illustrates how the robotic system interacts with the YSoft SafeQ.

4.2 Robotic Arm Project

The aim of the Robotic Arm Project (also known as RQA) is thorough testing of the user interface of YSoft SafeQ terminals in the physical environment. Thanks to RQA, testing is efficient, precise and it is an automated process used for continuous testing. Also, time spent on manual clicking and repetitive tasks is reduced.

Robot design

Robots comprise of a robotic arm itself and a camera. The robotic arm consists of three flexible joints, four small engines, a rubber tip, and is
4. Robotic testing of YSoft SafeQ

Figure 4.1: YSoft SafeQ workflow

1. Print job is created at workstation
2. Print job is sent to the server
3. User authenticates and selects a job to print
4. Request for selected job is sent
5. Print job is sent to the printer/MFD
6. Job is printed
4. **Robotic testing of YSoft SafeQ**

connected to an IoT platform YSoft SafeQube\(^2\). The role of the camera is to monitor the state of the touchscreen thanks to the computer vision.

The whole construction is placed above an embedded terminal or a smartphone. The camera watches over the touchscreen (either it is the touchscreen control panel of a printer or a smartphone touchscreen), and the robotic arm performs the required activity using a rubber tip. The robotic arm is not only capable of simple clicking but also drag and dropping and clicking on physical buttons on the printer. The arm follows testing scenarios that are written in the Robot Framework which is based on Python programming language.

Thanks to the camera, the robotic arm can distinguish between various items on the touchscreen. If some unexpected error appears, the system is able to recognize it and handle the situation correctly.

Figure 4.2: Robotic arm [10]

---

4. Robotic testing of YSoft SafeQ

Testing and test results

As already mentioned, Robot Framework is the test automation framework used in the RQA project. It is a generic framework for acceptance testing and it utilizes the keyword-driven testing approach [11].

Multiple robots are used for YSoft SafeQ testing and they can basically operate non-stop. This results in a lot of data that need to be stored and processed. All the data from robot configuration to actual test results are stored in a PostgreSQL database. All the important information from this database, including the activity of robots, number of test runs, and test results can be found in a web application developed by the RQA team for easier testing and analysis of test results.

To sum up, some of the main features of the Robotic Arm Project include:

- many test runs,
- precise measurements,
- recognition of error state,
- automated recording of test results,
- screenshots for test failures,
- testing in the physical environment.

It is necessary to mention one important aspect of robotic testing. The test scenarios check the functional requirements of the application. However, thanks to the camera placed above the touchscreen, the time needed for specific actions to be completed is measured with high precision. Such measurement means that also nonfunctional requirements are tested. This implies that RQA combines functional and nonfunctional testing without the necessity of manual testing.

4.3 Testing environment setup

One of the important aspects of testing is having a reliable, stable and consistent testing environment whenever the testing is performed.
The same setup is done for all tests performed by the robotic system, specifically:

- dedicated and isolated hardware,
- clean YSoft SafeQ installation,
- clean Windows server for each YSoft SafeQ installation,
- single server setup (all YSoft SafeQ component running on a single server), and
- default YSoft SafeQ server configuration.

Although the robotic system is quite precise, some limitations exist. The accuracy of measurements is approximately 30 ms.

The testing environment setup "as is" may be called "laboratory environment". While this is a good way to ensure a consistent testing environment for each round of testing, the real world usage of the YSoft SafeQ is quite different. The production environment has many differences compared to a clean installation of YSoft SafeQ used in the testing environment. An example of such would be the number of users of YSoft SafeQ or the latency between the server and the printer. Therefore, an analysis of the production environment is performed in the next chapter. The goal of this analysis is to identify some of these differences so that they may be subsequently simulated in the testing environment.

### 4.4 Test results visualization

The test results from the robotic system are visualized as measurements. Measurement is always a part of some test, whereas each test can contain several measurements. An example of a measurement is a duration of a logout from the application. Each measurement is repeated hundreds of times to ensure adequate accuracy of the measured duration, and then a median value for given measurement is calculated.

The existing visualization of measurements is based on two filters. The first one defines what kind of measurement should be visualized
(authentication, logout, etc.), the second one provides the user with a possibility of choosing multiple YSoft SafeQ versions that can be visualized for given measurement. Once these two filters are applied, the required data is read from the database and shown.

The visualization itself consists of three parts. The first part is a table showing for each filtered YSoft SafeQ version and each kind of device, the median value of the duration of measurement and the number of measurement repetitions for the selected measurement type. The next part is the bar chart showing the same data as the above-mentioned table. An example of such is the figure 4.3. The same kind of visualization is also available for each device separately. It facilitates observing the differences between the calculated values for different YSoft SafeQ versions on one device.

The visualization in figure 4.3 can be easily understood. The bar chart illustrates how fast the given YSoft SafeQ version works on a given device for the selected measurement. However, it is not possible to compare multiple measurement types. In other words, the fact that some kind of measurement, e.g. authentication time, works fast does not imply that also another type of measurement would be fast on given device with the given YSoft SafeQ version as well.
5 Analysis of production environment

Chapter 4 of this thesis provides a description of YSoft SafeQ testing using the robotic system. The testing is performed in laboratory conditions. While such approach grants interesting outcomes, the behavior of YSoft SafeQ in the production environment may differ from the clean installation in laboratory conditions.

The aim of this chapter is to look at the production environment and identify some interesting aspects of this environment. Such aspect is further referred to as "environment customization", "customization variable" or "environment variable" interchangeably. The second part of the chapter is dedicated to the tools for simulation of these customizations, together with an explanation of setup of these tools in the testing environment.

5.1 Production environment of YSoft SafeQ

The following analysis of the YSoft SafeQ production environment focuses on the user experience, more specifically, on the performance of YSoft SafeQ that the clients face when using the YSoft SafeQ terminals in production. The output of the analysis is an identification of environment variables that might impact the user experience.

Once the environment variables are simulated in the testing environment, the assumption that these variables influence the performance of the product can be either confirmed or rejected based on the results. The analysis of such test results is beyond the scope of this thesis. The goal is to prepare the tools for such analysis for the future. However, some initial testing in the customized environment is performed and visualized as part of the thesis.

The identification of the environment variables of the production environment is divided into two parts; the first one deals with the YSoft SafeQ architecture and the other one with the YSoft SafeQ server.

YSoft SafeQ architecture

YSoft SafeQ was described in previous chapters in a very simple setup. The real architecture of the print management solution differs from
5. Analysis of production environment

client to client depending on their needs. Typically, big companies with a few thousand employees have multiple offices, often located in different cities or countries. Thus, the architecture of the provided solution is more complex than for a small company with a few employees working in one office. Figure 5.1 is an example of an architecture for multiple offices. As illustrated in this figure, site servers communicate with a management server. These site servers are both local and remote.

![Figure 5.1: YSoft SafeQ architecture [12]](image_url)

Various kinds of network issues may occur between a remote office and a management server. Latency and packet loss are two common network issues having an impact on the application performance [13]. The simulation of these two problems might lead to finding a threshold where the user experience is not acceptable anymore. Once the threshold is known, the architecture of the solution can be adjusted to avoid reaching this limit. Also, a comparison between two versions of YSoft SafeQ on the same values of network customizations might discover potential issues in the implementation of the software.
5. Analysis of production environment

The two network issues mentioned in the previous paragraph are simulated as part of this thesis. Still, it is worth mentioning that other environment variables concerning YSoft SafeQ architecture exist. These include hardware parameters like CPU, hard drive speed, and RAM of the machines where the YSoft SafeQ components are deployed.

**YSoft SafeQ server**

Not only the hardware related issues may influence the performance of YSoft SafeQ. It is often the case that the performance of applications decreases with increasing usage of these applications. In relation to YSoft SafeQ, the following aspects should be mentioned:

- number of users,
- persistent load (jobs per minute),
- number of YSoft SafeQ servers for given installation\(^1\),
- number of devices connected to the YSoft SafeQ, and
- number of workstation clients.

As part of this thesis, two of the previously named variables were implemented in the testing environment, namely the number of users and the persistent load (in the form of print jobs). Their simulation in the testing environment is also load testing provided that the values set (the number of users/jobs per minute) are approaching the limits of the application. Please, note that "jobs" in this context refers to a print job processed by a YSoft SafeQ server.

### 5.2 Testing environment customization implementation

Multiple differences between testing and production environment were identified in the previous section of the thesis. This section

\(^1\) When there are multiple servers in a solution, they not only communicate with the printers and the workstation clients but also between each other which causes additional load on these servers.
5. **Analysis of production environment**

describes the tools and their integration into the testing environment to simulate some chosen variables, namely:

- latency,
- packet loss,
- persistent load (jobs per minute),
- number of users.

5.2.1 **Latency and packet loss**

Clumsy 0.2 is a tool for simulation of poor network condition on Windows. It is based on the WinDivert 1.4 package. WinDivert is a user-mode packet capture-and-divert package for various Windows distributions. In summary, the WinDivert features allow the users to:

- capture network packets,
- filter/drop network packets,
- sniff network packets,
- (re)inject network packets,
- modify network packets [14].

There are several reasons for choosing clumsy 0.2 for packet loss and latency simulation in the testing environment:

- no installation required,
- works for any application (system-wide, not application specific),
- works also offline,
- no license purchase required (issued under the MIT license) [15].
5. Analysis of production environment

Clumsy has a simple user interface for the setup of bad network condition simulation (see figure 5.2). The first step of this setup is the filtering option (based on a logical combination of criteria like inbound/outbound, TCP/UDP, etc.). Only the behavior of the packets that are captured by the specified filter is modified by this tool. One or more functions modifying the network conditions can be applied to captured packets (e.g., lag, drop, out of order). More information about the options of clumsy 0.2 is to be found in its documentation [15].

![Figure 5.2: Clumsy 0.2](image)

**Latency**

The function “Lag” is used to simulate the latency between a YSoft SafeQ server and a printer. The destination address in "Filtering" is set to the IP address of the printer that is used for testing. To apply the latency, the following steps are necessary:

- Tick the "Lag" checkbox.
- Tick the "Outbound" checkbox (only the outbound packets should be captured, as the latency from server to printer is simulated).
- Set the value of latency in the "Delay(ms)" input field.
5. Analysis of production environment

- Click "Start".

Packet loss

To simulate the packet loss, the "Drop" option is used. The configuration of clumsy 0.2 is similar to the one described in the previous paragraph. The destination address in "Filtering" is set to the IP address of the printer used for testing. The "Drop" option needs to be ticked, together with the "Outbound" option in the respective row. The value of the "Drop" function is set to the amount of packets dropped in %.

5.2.2 Number of users

Increased number of users in YSoft SafeQ may worsen the user experience, particularly during the authentication. Concerning YSoft SafeQ, an increased number of users means multiple thousands of users. For this purpose, an internal Y Soft tool was chosen. The tool generates the required number of users into a CSV file. This file is then imported into the YSoft SafeQ used for testing.

5.2.3 Persistent load

As multiple printers are usually connected to one YSoft SafeQ server, the server needs to process the requests from all of them simultaneously. To simulate this situation, another Y Soft internal tool was used. This simulation works as follows:

- Print jobs are sent from a load generator to the YSoft SafeQ server.
- YSoft SafeQ server processes these jobs.
- Print of jobs is requested by the load generator.
- The jobs are sent from the YSoft SafeQ server to a virtual printer.

This means that no actual printing is necessary for this simulation. While the above-mentioned process is running and multiple jobs per minute are being sent to the server, the measurements in tests are performed as usual.
6 Database for test results

All the test results from the robotic system are stored in a PostgreSQL database. First, this chapter describes the old database schema using a so-called entity-relationship diagram (further also referred to as "ERD"). This schema is not suitable for storing information about testing environment setup described in chapter 5.1 and therefore a new one is proposed. The second part of this chapter is dedicated to the design of the new schema. An ERD is provided for an easier understanding of the design together with an explanation for the essential parts of the schema. The new schema not only adds the required support for testing environment setup, but it also brings some enhancements to improve the old design.

6.1 Current database schema

Figure 6.1 is an ERD of the database schema that is currently used. The whole schema is, of course, much more complex and only the parts relevant for this thesis are illustrated. The most important tables are TestResult and Measurements. Tables Device and SafeQServer are shown for a better understanding of context as these are referred to as foreign keys in the TestResult and Measurements tables.

TestResult table
All test results are stored in the TestResult table (one row in this table is also called a "test" in the further parts of this thesis). Each record in the database stores information about testing itself (e.g., test result, duration, start time, etc.) as well as about testing environment. Testing environment information includes what YSoft SafeQ server, what device (printer) and what testing robot were used.

Measurements table
Each measurement runs as part of some test. For each measurement, Measurements table keeps information about:

- testing environment (device, YSoft SafeQ server, terminal type, robot, YSoft SafeQ version),
Figure 6.1: Old database schema
6. Database for test results

- measurement itself (measurement type, duration, start time, result, TestGuid\(^1\)).

### 6.2 Requirements

The old database schema does not allow storing information about environment customization. Environment customization is a setup of the testing environment that simulates the production environment as already discussed in chapter 5.

Another problem of the schema is the duplication of information. As all measurements are part of some test, information about the testing environment and YSoft SafeQ version does not need to be present in the Measurements table.

Therefore, two major requirements on the new schema arise:

- Reduce duplicated information.
- Allow storing information about testing environment customization.

### 6.3 New database schema

This section is dedicated to the new database schema. This schema was designed to satisfy the requirements mentioned in the previous paragraph. To begin with, an overview of the most important relationships of the schema is provided. It is followed by a more detailed explanation of some chosen entities.

The figure 6.2 is an entity relationship diagram of the newly designed schema. The outstanding relationships of the design are these:

- Each measurement runs as part of some test.
- Each test runs on some combination of YSoft SafeQ server, device, and robot.
- Each test has zero or one environment customization.

---

1. GUID is an acronym that stands for Globally Unique Identifier
6. DATABASE FOR TEST RESULTS

- Each test runs against some specific product.

All the relationships between entities can be read from the schema, and therefore only a few of them are discussed here.

EnvironmentCustomization and CustomizationVariable

The tables EnvironmentCustomization and CustomizationVariable serve for storing information about the changes made to the testing environment that simulate the production environment. Each test performed in a customized environment is linked to some environment customization. The attribute Name is a descriptive naming for given customization while Description is an optional value to provide some more information about the testing setup. While at the moment, each environment customization has only one customization (therefore it is related to one customization variable), the design of the schema is prepared for multiple customizations in one test in the future.

Product and Component

Previously, the information about YSoft SafeQ product version was stored in the Measurements table. First of all, the information was redundant as each measurement is part of some test and each measurement of the given test always runs on the same product. Taking this into account, this information is no longer related to a measurement but to a test. Moreover, the information about YSoft SafeQ versions was extracted into a new table called Product. This table is related to another one - Component. The intention is to provide a mechanism to test not only with a productized versions but also with some nightly builds. In other words, it allows testing a product with some components having custom versions.

TestResult

TestResult table contains records about tests. Each test stores information about:

- the product it is testing (ProductId),
6. Database for test results

![Figure 6.2: New database schema](image-url)

```sql
CREATE TABLE Device (Id int4, 
  DateCreated timestamp(6), 
  DateModified timestamp(6), 
  IpAddress IpAddressData bytea, 
  DeviceModelId int4, 
  Comment text); 

CREATE TABLE Measurements (Id int4, 
  MeasurementType text, 
  Duration int4, 
  StartTime timestamp(6), 
  Result int4, 
  Note text, 
  TestResultId int4); 

CREATE TABLE SafeQServer (Id int4, 
  Name text, 
  EnvironmentVersion_Major int4, 
  EnvironmentVersion_Minor int4, 
  EnvironmentVersion_Build int4, 
  EnvironmentVersion_Revision int4); 

CREATE TABLE EnvironmentCustomization (Id int4, 
  Name text, 
  Description text); 

CREATE TABLE CustomizationVariable (Id int4, 
  EnvironmentCustomizationId int4, 
  Name text, 
  Value int4); 

CREATE TABLE Component (Id int4, 
  ProductId int4, 
  Name text, 
  Version text, 
  Note text); 

CREATE TABLE Product (Id int4, 
  Name text, 
  EnvironmentVersion_Major text, 
  EnvironmentVersion_Minor text, 
  EnvironmentVersion_Build text, 
  EnvironmentVersion_Revision text); 

Figure 6.2: New database schema
```
6. Database for test results

- the testing environment setup (DeviceId, ServerId, RobotId, EnvironmentCustomization, TerminalType),

- the test itself (TestlinkId, TestName, StartTime, Duration, PaperCount, JobType, Result),

- TestGuid - an identifier used for example for naming screenshots from given test.

Note is an optional value that contains some additional information about the test.

Measurements

The Measurements table has the following columns:

- MeasurementType - optional information about the type of the measurement,

- Duration and StartTime to illustrate when the measurement started and how long it took to complete it,

- Result of the measurement, where two outcomes are possible (pass or fail),

- Note to provide an option to mark some additional information about the measurement, and

- TestResultId which is a foreign key to the TestResult table.

The migration from the old to the new schema is explained in the next chapter.
7 Schema and data migration implementation

Migration from an old to a new database schema of a database having more than a million of rows may be challenging. Schema migrations where no data in a database exists are straightforward and many tools for generating migration scripts exist. The challenge comes with the migration of data. It is often the case that some records in the production database are corrupted, missing, not correctly migrated in the past, etc. The data should be migrated not only in a way where no data loss occurs but also so that all relations between data rows are kept.

This chapter describes how the database schema and all the existing data where migrated. First, the database system and the framework for accessing data are shortly introduced. Then the whole migration is explained - how the scripts for schema migration were generated, how the data was migrated and what problems occurred during the data migration and how they were solved.

7.1 Tools

7.1.1 PostgreSQL

PostgreSQL is an open source object-relational database system. PostgreSQL (also often referred to as “Postgres”) is very popular thanks to its huge community of developers and users, liberal license (allowing it to be used, modified and distributed by anyone for free) and being the most advanced open-source database available. Also, the official documentation is quite extensive and comprehensible\(^1\). PostgreSQL supports many parts of the SQL standard and offers various modern features, such as complex queries, triggers, foreign keys, updatable views and many more [16].

7.1.2 Entity Framework

Entity Framework is an open-source object-relational mapping framework for .NET developers supported by Microsoft. It enables develop-

---

1. https://www.postgresql.org/docs/
7. Schema and data migration implementation

ers to work with relational data the same way as with non-relational data using .NET objects. Therefore, it is possible to work at a higher level of abstraction and eliminate most of the data-access code [17]. Three development approaches are available in Entity Framework:

- Code-First,
- Database-First,
- Model-First.

**Code-First approach**
This is a good option when creating a new database. The basic concept resides in focusing on the domain of the application. First, the C# classes representing domain entities are created. Additional configuration is available in the form of data annotation attributes on classes and properties or using a Fluent API. When all the classes and configuration are ready, the database can be created using the Entity Framework migration commands.

**Database-First approach**
Unlike the Code-First approach, the Database-First is used for existing databases. The entities and context are generated from an existing database using the Entity Framework Designer. It is a reverse engineering approach mostly convenient when working with stable schemas where updates to schemas are not frequent.

**Model-First approach**
Likewise the Code-First, the Model-First approach is suitable when starting without an existing database. The database model is created using the Entity Framework Designer and the database schema is generated from the previously created model. The classes and the configuration are automatically generated from the model [18].

The Code-First approach is used in the RQA project as it provides some advanced options for working with database schema and data migrations. This is explained in detail in the following section.
7. Implementation

The PostgreSQL database of the RQA project stores more than one million rows about tests and their measurements. Several migrations were applied to the database schema previously leaving some data inconsistent. The goal is to migrate all the data to the new schema while also fixing the inconsistencies to leave the database in a correct state. The first part of this section justifies what database schema migration approach was chosen. A short explanation of the migration process in Entity Framework is provided. The next part is dedicated to data migration and its challenges.

7.2.1 Database schema migration

Entity Framework enables to use either an automated migration or a code-based migration when using the Code-First approach. The automated migration updates the database schema whenever the domain classes are changed. It works well when domain classes are added or removed, but it is not suitable when existing domain classes are modified as it might result in data loss [19]. Another approach is the code-based migration which gives more control over the migration to developers and is a suitable option for our use case.

Entity Framework code-based migration

The code-based migration allows the developer to create a database schema migration based on the changes in the C# classes representing entities of the database model. To begin with, some initial setup to enable the migrations needs to be done. Once the setup is done, each migration requires two commands in the Package Manager Console in Visual Studio, Add-Migration and Update-Database. The former command creates a migration file with two methods, Up() and Down(). The Up() method contains the code that transforms the database schema into the desired state. The Down() method is used in the opposite case when the migration is removed as it returns the database into the state before the migration.

The migration code generated by Entity Framework can be modified according to the current needs (for example, a default value
7. Schema and data migration implementation

for some column can be easily set). After all the necessary modifications, the changes to the database are applied by running the `Update-Database` command. As already mentioned, these migrations are reversible.

**Generated migration code**

The previous chapter explained the changes that need to be applied to the old database schema. There are three kinds of changes to be done in the migration:

- Remove redundant columns.
- Extract columns from old tables.
- Add support for storing newly added information.

Updating the database using the migration code generated by the Entity Framework would result in a correct database schema. However, applying this migration would cause a big data loss and inconsistency. The next subsection is dedicated to the explanation of the necessary steps for preserving all the data and its relations.

### 7.2.2 Data migration

While the generated migration code works well for the schema migration, it needs to be customized for correct data migration. There are basically two issues that need to be fixed:

- the order of SQL queries/missing context, and
- inconsistent/missing data in the database.

In our case, these issues are fixed by changing the order of methods called on the Migration Builder\(^2\) and by adding approximately 40 SQL queries into the migration. Custom SQL queries can be easily added to the migration, an example of such is the following code:

```csharp
migrationBuilder.Sql("INSERT INTO \"EnvironmentCustomiz-
```

---

\(^2\) https://docs.microsoft.com/en-us/ef/core/api/microsoft.entityframeworkcore.migrations.migrationbuilder
Following paragraphs describe the above mentioned issues in more detail.

**The order of SQL queries/missing context**

An example of this issue is the extraction of the environment version columns from the `Measurements` table, see the following figure:

![Figure 7.1: Extracting rows](image)

As expected, Entity Framework does not recognize the context of changes. Thus, the relation between the environment version columns in the `Measurements` table and in the `Product` table is overlooked. In the generated migration, firstly, these columns are dropped from the `Measurements` table and later on, they are created in the `Product` table. Therefore, to avoid data loss, the following steps are applied:

1. Create the `Product` table.
2. Extract all distinct products from the `Measurements`.
3. Fill in the extracted data into the `Product` table.

---

3. This refers to four columns, namely `EnvironmentVersion_Major`, `EnvironmentVersion_Minor`, `EnvironmentVersion_Build`, and `EnvironmentVersion_Revision`.
Inconsistent/missing data in the database

Currently, each measurement runs as a part of a test, but not all tests do necessarily have measurements. When you look at the old database schema (figure 6.1), there is no direct relation between the TestResult and Measurements table (no relation in the form of a foreign key).

Both TestResult and Measurements have a TestGuid column. This suggests that for each measurement, it should be possible to trace back the test that the measurement was part of. The problem lies in hundreds of thousands of rows that have a TestGuid with value '00000000-0000-0000-0000-000000000000'. In such a case, the relation between TestResult and Measurements cannot be based on the TestGuid value. The solution to this issue consists of two parts:

1. TestResult table - unlike the Measurements, the TestResult table does not provide information about EnvironmentVersion and TerminalType; therefore all the rows with "all-zeros" TestGuid were linked to artificially created values (to meet the "not nullable" condition of the missing values in the new schema).

2. Measurements table - new database schema requires all measurements to be linked to a test; for this purpose, multiple artificial TestResult rows were necessary to keep all the information from Measurements, and all the missing values (e.g. JobType, Duration) were filled with default values.

This is a simplified description of the data migration process. The migration was done in order not to lose any relationship in the original data using some helper tables, for example, so that each record in the Measurements table is still linked to the same environment version as before.

The implementation details of the migration are skipped in this text. The source code of the migration is included in the archive in the Information System of the Masaryk University.
8 Test results visualization

The previous chapters of this thesis described how to set up the testing environment to simulate the production use of YSoft SafeQ and how to store data from such testing in a database. This chapter deals with the visualization of this data. In the first part, DotVVM framework, a tool for building web applications is introduced. The implementation of test results visualization together with the description of the implemented charts are discussed in the latter parts of this chapter. Moreover, multiple charts that might be implemented in the future are presented in the last part of the chapter.

8.1 Tools

Many tools for visualization of data from a PostgreSQL database exist. Some of them may be integrated into existing solutions. An example of such is Power BI. However, the problem with such approach is that integration of a new tool into an existing solution requires additional maintenance. Therefore, data visualization proposed as part of this thesis was implemented with help of tools currently used in the RQA project web application.

8.1.1 DotVVM framework

DotVVM is an open source framework for building web applications. The "Dot" in its name stands for the .NET Framework and VVM refers to the Model-View-ViewModel, design pattern used in this framework. All Dot-VVM pages consist of two files - a dothtml file to define the view of the page and a C# class that represents the view-model part of the application [20].

View

The view is defined by a file with a .dothtml extension. The .dothtml files use standard HTML syntax. Moreover, thanks to a built-in mechanism for managing resources, DotVVM supports JavaScript and CSS files. The view must reference its respective view-model using a
8. Test results visualization

@viewModel directive [21].

ViewModel

ViewModels in DotVVM are C# classes that are JSON-serializable. A viewModel has two important roles, it represents the state of the page (by public properties of the class) and defines the commands that can be invoked in the page (public methods of the class). The public methods of the viewModel can also modify the state of the viewModel (preferably by calling methods from the business layer as classes representing the viewModels are part of the presentation layer and should not contain any business logic) [22].

8.2 Design and implementation

The process of design and implementation of the test results visualization is discussed in the following part of this chapter.

First, the business needs for the test results visualization are mentioned. In the second subsection, the variables of data to be visualized are discussed in more detail. The rest of the chapter deals with the common components of the design of all charts, followed by the presentations of all implemented charts and their use cases.

8.2.1 Business needs

The test results data from the customized environment is useful mostly for three groups of people - higher management, sales representatives and developers/testers of Y Soft company. Especially when it comes to sales representatives, the data should be available in a form that is clear and comprehensible. Such kind of information is usually useful in presentations for customers and therefore the visualization of this data should be easily understood.

The information that is immediately evident in the visualization is also an important aspect. All of the above-mentioned groups concentrate on their own interests, thus the visualization of data should differ according to the needs of these groups. Therefore, multiple charts were implemented as part of this thesis and they are discussed in more detail in the latter parts of this section.
8. Test results visualization

8.2.2 Visualized data

It was already explained in the chapter 4 that the test results are visualized in the form of measurements. Each test eligible for visualization consist of multiple measurements. As shown in the entity-relationship diagram of the database storing test results (figure 6.2), a lot of information is stored for each measurement. However, only a part of the stored information is relevant for the visualization, namely:

- measurement type,
- measurement duration,
- device,
- product version,
- type of environment customization, and
- value of given environment customization.

As the purpose of this visualization is to determine the performance of YSoft SafeQ in a customized environment, the duration of measurement is always the dependent variable.

To keep the visualization simple, only a subset of the information available for each measurement is shown in the chart. Therefore, the implementation of charts provides the user with an option to filter the data and afterward, the data for given filter is visualized. This is explained in more detail in the following part of the thesis.

8.2.3 Common components of implemented charts

Three charts were implemented as part of this thesis (discussed in 8.2.4). While all of them visualize the same data, the information that is directly visible in each of them is different. However, the approach used for their implementation is the same for all of them and is as follows:

- There are three drop-down buttons.
- Each chart shows the data filtered by these drop-down buttons.
8. Test results visualization

- A table containing some additional information is placed below each chart.

Figure 8.1 is an example of drop-down buttons for filtering the test results data. Figure 8.2 is an example of a table placed below each chart. It contains all the information from the chart in a numeric form (median value in seconds for each combination of independent and dependent variables) and the number of repetitions used for calculation of each of these medians. This table has several tooltips for better understanding of the data it contains.

![Figure 8.1: Drop-down buttons for filtering of data](image1)

<table>
<thead>
<tr>
<th>Device model</th>
<th>Type of environment customization</th>
<th>YSoft SafeQ version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Model 1</td>
<td>Latency</td>
<td>Version 1</td>
</tr>
</tbody>
</table>

![Figure 8.2: Table providing additional information](image2)

<table>
<thead>
<tr>
<th>Packet loss [%]</th>
<th>3</th>
<th>6</th>
<th>9</th>
<th>12</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication Measurements</td>
<td>1.923</td>
<td>2.312</td>
<td>3.290</td>
<td>4.851</td>
<td>7.300</td>
</tr>
<tr>
<td>Logout Measurements</td>
<td>3.937</td>
<td>3.979</td>
<td>4.510</td>
<td>6.492</td>
<td>6.950</td>
</tr>
<tr>
<td>Menu to SQPrint</td>
<td>2.766</td>
<td>3.191</td>
<td>4.239</td>
<td>4.602</td>
<td>5.519</td>
</tr>
</tbody>
</table>

8.2.4 Implemented charts and their use

The following subsection is dedicated to the description of charts implemented as part of this thesis. The purpose of each of them is explained, and a sample screenshot is provided. The source code of implementation is available in the archive of this thesis in the Information System of Masaryk University. Appendix 9 provides the description of the content available in the above-mentioned archive.
First chart

This chart is preceded by three drop-down buttons, where the user needs to choose:

- a device,
- a type of customization, and
- a YSoft SafeQ version.

The x-axis of this chart shows the growing values of the selected type of customization, while the measurement time values appear on the y-axis. A line chart was chosen so that the trend of decreasing performance of YSoft SafeQ on growing values of environment customization may be easily spotted. Each line in this chart represents one measurement type.

This chart is of use mainly for developers and testers. Its main advantage is the presence of multiple types of measurements in one view. The extent to which growing values of specific customization affect the performance of YSoft SafeQ can be easily compared between the measurement types and subsequently analyzed. For example, it could be used to investigate whether increasing number of users influences all measurement types equally or the impact on authentication measurements is more significant than on other types).

![Figure 8.3: First chart](image)
8. Test results visualization

Second chart

In this chart, the user first needs to filter the data by choosing

- a device,
- a customization type, and
- a measurement type

The x-axis of this chart is the same as in the previous one; it displays the increasing values of the selected customization type. As for all the implemented charts, the measurement time for given combination of independent variables appears on the y-axis. Each line in the chart represents the values calculated for one YSoft SafeQ version.

This chart is helpful in various use cases. It can be used to compare the overall trend of the performance of various versions of YSoft SafeQ for growing values of given environment customization in presentations to customers or higher management (for example, to show that the version 2 generally performs slightly better than version 1 on all values of environment customization for authentication measurements). Also, developers and testers may use this chart to review whether a decrease in performance does not appear for newer versions of YSoft SafeQ and eventually, analyze the root cause of such decrease.
Third chart

The values that appear on the x-axis of this chart are different versions of YSoft SafeQ, therefore, a bar chart was chosen. In this case, the drop-down buttons for filtering the visualized data contain these variables:

- a device,
- a customization type, and
- a value of the selected customization type.

This chart shows the comparison of the performance of various YSoft SafeQ versions on given customization type and its value. As for its advantages, this chart displays all types of measurements simultaneously (one color in the chart stand for one measurement type). While it may seem that visualization of YSoft SafeQ performance on a specific value of an environment customization type may be of no use, some specific situations where this chart may be beneficial for sales representatives exist. An example of such case is the following - if there is a big customer with tens of thousands users in YSoft SafeQ and they complain that they are not happy about the speed of login and logout operations, this chart may be used to show them that the performance in an environment with a huge number of users has improved between an older and a newer version of YSoft SafeQ.

![Figure 8.5: Third chart](image)
8. Test results visualization

All of the implemented charts are interactive and they offer the possibility to show a subset of its data by clicking the label of the part of data that should be hidden.

8.3 Other visualization options

This section is divided into two parts. First, the proposals that were not implemented as part of this thesis are discussed, be it in terms of combinations of visualized variables or types of charts. In the second part, future possibilities of other visualization options following up on this thesis are described.

8.3.1 Not implemented proposals

Five charts were proposed for visualization of test results, three of them were implemented. This subsection deals with the other two proposals and the reasoning behind not implementing them. Moreover, another type of chart that was considered to be used in the visualization is mentioned and its disadvantages compared to the selected charts are pointed out.

Proposed charts

Figures 8.6 and 8.7 were also proposed as visualization options. For both these charts, the user would first need to select some values in drop-down buttons to filter the data to be visualized (the approach with drop-down buttons is described in 8.2.3); namely measurement type, customization type, and customization value for the chart in figure 8.6 and measurement type, customization type, and YSoft SafeQ version for the chart in figure 8.7.

The reason why the two above-mentioned charts were not implemented is the same for both of them. They both compare the measurement time of various devices. It is known that some device models perform better or worse compared to others. The development of YSoft SafeQ is not focused on achieving the same performance for all devices but on improving the performance of YSoft SafeQ in general. Most YSoft SafeQ customers already have their device vendor and therefore, they care about the improvement of performance of YSoft SafeQ on
their devices (thus, the comparison of devices has no significant value neither for customers, nor for developers/testers).

![Figure 8.6: Bar chart comparing performance of devices](image1)

![Figure 8.7: Line chart comparing performance of devices](image2)
8. Test results visualization

Other chart types

Two chart types were implemented as part of this thesis. Line charts were chosen for situations, where the variable on the x-axis is continuous. Although it is possible to use bar charts in such a case, it is easier to spot trends in line charts. A bar chart was selected for visualization 8.5 as the variable on the x-axis is not continuous. All of the implemented charts have only two axes, x-axis and y-axis and no 3D effects are used. They are simple and easy to read.

One other chart type was considered as an option for test results visualization. It is a bar chart with three axes and a 3D effect (see figure 8.8). This chart was not implemented because of the following reasons:

- It may not be easy to spot trends in this chart.
- Even a small number of values on the y-axis might make the orientation in the chart difficult.
- Some bars might not be sufficiently visible because of other bars in front of them.

Despite the fact that the disadvantages of this chart are not significant, no obvious advantages were discovered, and therefore it was not a preferred option for the implementation.

\[\text{Figure 8.8: 3D bar chart}\]
8. Test results visualization

8.3.2 Future possibilities

The database schema designed in chapter 6 offers various options related to testing. As it was already explained in previous parts of this thesis, testing environment can have multiple customizations and testing with a product having a custom component can be performed.

This subsection takes into consideration these options and proposes some visualizations that can be implemented in the future. Please, note that the charts described in the following part of the thesis visualize only a subset of test results data and therefore, the implementation of these charts would include a filtering option (on the same principle as the drop-down buttons described in section 8.2.3).

Combination of two customization types

The chart proposed for visualization of test results from a testing environment with two customization types is shown in figure 8.9. A chart with three axes was chosen to point out the presence of two customization types. They appear on the two horizontal axes of the chart. The duration of the selected measurement on a combination of x-axis and y-axis values is shown on the vertical axis of the chart.

![Figure 8.9: Two customization types](image)

Provided that all measurements are performed on the same version of the tested product and that the bars with greater customization
value are placed behind bars with smaller customization value, no bar should be hidden by another one.

The proposed chart may be used for any combination of customization types as long as the combination makes sense. The example of such a combination shown in figure 8.9 may be used for analysis of the production environment of customers with many users and bad network condition.

**Testing with custom components**

Sometimes, important customers report bugs that need to be fixed immediately and therefore, software patches are necessary. If the requested patch is supposed to fix the performance of the product, the performance with and without the patch should be tested and compared.

The database schema in figure 6.2 allows storing information about a custom component of the tested product. Therefore, testing can be performed on two different versions of a product - a released version of the product and a version with a custom component (a component with an applied patch). The performance of these can be compared and the patch may be verified/declined. An example of such situation is a requirement to fix the performance of authentication of users with 20 000 users in YSoft SafeQ.

![Graph](image-url)

**Figure 8.10: Testing of product with custom components**
Failures of measurements

All measurements in the database are marked as passed or failed. There are some failures even if the testing is performed in a laboratory environment. Multiple causes of these failures exist (related both to YSoft SafeQ and to robotic system). However, with increasing values of customization applied to the testing environment, the rate of failed tests increases.

Figure 8.11 is a chart showing the percentage of failed measurements (out of all measurements performed), the customization type is the independent variable on the x-axis. At this moment, only the measurements marked as passed\(^1\) are used in all visualizations in the RQA web application. Increasing extent of customization of testing environment causes incorrect behavior of YSoft SafeQ, and this chart might be used to analyze it. The chart in figure 8.11 shows the comparison of two versions of YSoft SafeQ regarding their tolerance to worsening testing environment conditions.

![Figure 8.11: Rate of measurement failures](image)

\(^{1}\) Measurement is marked as "pass" if the robot is able to get from the screen A to the screen B.
8. Test results visualization

**Duration of all measurements combined**

In all charts mentioned in this thesis up until now, the performance of YSoft SafeQ on a specific measurement was observed. However, the overall performance of all types of measurements combined may provide interesting data for analysis. The chart in figure 8.12 shows measurements time for all release versions on given customization value (x-axis). In this case, measurements time represents the sum of durations of all measurement types for a given version. It is expected that a newer version of the tested software should perform better and this chart allows to easily check the comparison of the performance of various versions of YSoft SafeQ for all measurements in one chart. However, this chart can only be used if same measurement types are available for each release version and each customization value.

![Figure 8.12: Measurements combined](image)

**Figure 8.12: Measurements combined**


9 Conclusion

In the first part of the thesis, the theoretical basis was introduced to emphasize the importance of software testing and data visualization. Chapter 4, was dedicated to YSoft SafeQ and the robotic system designed for its testing. Also, the setup of the testing environment and the visualization of test results before the implementation of this thesis were described.

Chapter 5 dealt with the analysis of the YSoft SafeQ production environment. Multiple variables that may affect the performance of YSoft SafeQ for end users were identified and four of them were simulated in the testing environment. For all of these variables, hundreds of test runs were performed to test the customized testing environment setup and obtain some initial data to process.

Chapters 6 and 7 were both dedicated to the database for test results. Chapter 6 discussed the original database schema and its limitations. Based on that, a new schema was designed in this chapter. The new schema allows storing the information about the customized testing environment and fixes some drawbacks of the original schema. The following chapter described how the migration of the old database schema together with the migration of its data was performed and what were the problems encountered.

Finally, the design of the visualization of test results from the customized testing environment was proposed in chapter 8. Three charts were implemented into the web application of the RQA project, their use cases were also mentioned in this chapter. Other charts that might be implemented in the future were described as well.

Initial testing in the environment customized by variables identified in the analysis of the production environment was performed. Also, the database migration was tested and is ready to be used in the RQA project. The charts implemented for the test results visualization are fully functional. Also, the changes in the RQA web application that were necessary due to the new database schema were implemented as part of this thesis. Once the RQA team adapts the UI application used for starting tests, all the implementation done as part of this thesis may be used in testing of YSoft SafeQ.
9. Conclusion

Various opportunities for a followup work on this thesis exist. The database schema was designed to allow testing with multiple customizations and for testing the product with a custom component. Also, it is worth mentioning that now, only the results of tests that passed are visualized. For the future, it might be interesting to analyze how the extent of environment customization influences the number of fails in the test results. Most importantly, the implementation done in this thesis is a tool that can be immediately used to analyze the impact of the identified environment variables on the YSoft SafeQ performance.
Bibliography


A Source code information

In this appendix, the information about the source code files is presented. The code implemented in this thesis is part of a bigger solution. However, the whole solution may not be published and only the source code exclusively related to this thesis is available in the Information System of Masaryk University.

A.1 Database schema and migration

The folder "Database" contains the C# classes representing the tables of the new database schema described in chapter 6. The folder also contains the code for migration of the database schema and data discussed in chapter 7.

A.2 Data visualization implementation

The folder "Visualization" has two subfolders. The subfolder "View" contains the "CustomizedEnvironment.dothtml" and "CustomizedEnvironment.js" files that represent the "view" part of the implemented visualization. The subfolder "ViewModel" contains C# classes implementing the view-model part and its respective business logic.