Degree project

Maintenance of the Quality Monitor Web-Application

Author: Maksym Ponomarenko
Supervisor: Rüdiger Lincke
Date: 2013-09-11
Course Code: 4DV00E, 15 credits
Level: Master
Department of Computer Science
Abstract

Applied Research in System Analysis (ARiSA) is a company specialized in the development of the customer-specific quality models and applied research work. In order to improve the quality of the projects and to reduce maintenance costs, ARiSA developed Quality Monitor (QM) – a web application for quality analysis.

QM application has been originally developed as a basic program to enable customers to evaluate the quality of the sources. Therefore, the business logic of the application was simplified and certain limitations were imposed on it, which in its turn leads to a number of issues related to user experience, performance and architecture design. These aspects are important for both application as a product, and for its future promotion. Moreover, this is important for customers, as end users.

Main application issues, which were added to the maintenance list are: manual data upload, insufficient server resources to handle long-running and resource consuming operations, no background processing and status reporting, simplistic presentation of analysis results and known usability issues, weak integration between analysis back-ends and front-end.

In order to address known issues and to make improvements of the existing limitations, a maintenance phase of QM application is initiated. First of all, it is intended to stabilize current version and improve user experience. It also needed for refactoring and implementation of more efficient data uploads processing in the background. In addition, extended functionality of QM would fulfill customer needs and transform application from the project into a product. Extended functionality includes: automated data upload from different build processes, new data visualizations, and improvement of the current functionality according to customer comments.

Maintenance phase of QM application has been successfully completed and master thesis goals are met. Current version is more stable and more responsive from user experience perspective. Data processing is more efficient, and now it is implemented as background analysis with automatic data import. User interface has been updated with visualizations for client-side interaction and progress reporting. The solution has been evaluated and tested in close cooperation with QM application customers.

This thesis describes requirements analysis, technology stack with choice rationale and implementation to show maintenance results.

Key-words: software quality, metrics, maintenance, analysis, visualization, Quality Monitor, ARiSA, optimization, architecture, data-driven documents.
Abbreviations and Definitions

ARiSA – Applied Research in System Analysis, an IT consulting company [1];
QM – Quality Monitor web application;
CPU – a Central Processing Unit, hardware in a computer that executes processing instructions;
RAM – Random-Access Memory, type of memory in computers;
XML – Extensible Markup Language, text markup language for representing data in a human- and computer-readable format [2];
GML – Geography Markup Language, an XML-based format for storing geographic information [3];
PNG – Portable Network Graphics, a raster graphics format for storing data in portable, lossless representation with definite compression level [4];
SVG – Scalable Vector Graphics, an XML-based vector image format for graphics with interaction and animation elements [5];
BIRT – Business Intelligence and Reporting Tools [6];
DOM – Document Object Model, a platform- and language-independent document model based on objects and their interaction [7];
D3 – Data-Driven Documents, a JavaScript library for documents manipulation based on data [8];
HTML – HyperText Markup Language, the markup language to create web-browser pages [9];
CSS – Cascading Style Sheets [10];
JSON – JavaScript Object Notation, a text-based data representation format [12];
CSV – Comma-Separated Values, a plain-text format for storing data [13];
NFR – Non-Function Requirement;
I18N (i18n) – Internationalization;
L10N (l10n) – Localization;
N/A – Not Applicable;
SSH – Secure Shell;
UML – Unified Modeling Language;
UC – Use Case;
Contents

1 INTRODUCTION .................................................................................................................. 1
  1.1 PROBLEM AND MOTIVATION ............................................................................... 1
  1.2 GOALS AND CRITERIA ......................................................................................... 1
  1.3 THESIS OUTLINE ................................................................................................. 2

2 BACKGROUND .................................................................................................................. 3
  2.1 SOFTWARE QUALITY ......................................................................................... 3
  2.2 QUALITY MONITOR APPLICATION .................................................................. 3

3 FEATURES AND REQUIREMENTS ANALYSIS ................................................................... 5
  3.1 USERS .................................................................................................................. 5
  3.2 FUNCTIONAL REQUIREMENTS ........................................................................... 5
  3.3 NON-FUNCTIONAL REQUIREMENTS .................................................................. 6

4 TECHNOLOGY RATIONALE ............................................................................................ 9
  4.1 WEB TOOLKIT ....................................................................................................... 9
    4.1.1 Choice Rationale .......................................................................................... 9
    4.1.2 Google Web Toolkit ..................................................................................... 10
  4.2 WEB SERVER ......................................................................................................... 10
    4.2.1 Choice Rationale .......................................................................................... 10
    4.2.2 Apache Tomcat ............................................................................................ 11
  4.3 DATABASE SERVER ............................................................................................... 11
    4.3.1 Choice Rationale .......................................................................................... 11
    4.3.2 MySQL Overview ........................................................................................ 12
  4.4 DATA VISUALIZATION TOOLKIT ......................................................................... 12
    4.4.1 Choice Rationale .......................................................................................... 12
    4.4.2 Data-Driven Documents .............................................................................. 13
  4.5 TECHNOLOGY CHOICE SUMMARY ....................................................................... 14

5 ARCHITECTURE DESIGN .................................................................................................. 15
  5.1 ARCHITECTURE STRATEGY .................................................................................. 15
    5.1.1 Modularity .................................................................................................... 15
    5.1.2 Localization .................................................................................................. 15
    5.1.3 Errors Handling ............................................................................................ 15
    5.1.3.1 Critical Errors .......................................................................................... 15
    5.1.3.2 Operational Errors ................................................................................... 15
  5.2 ARCHITECTURE VIEWS ......................................................................................... 16
    5.2.1 Logical View .................................................................................................. 16
      5.2.1.1 Quality Monitor – Daemon ................................................................... 17
      5.2.1.2 Quality Monitor – Web ......................................................................... 17
      5.2.1.3 Quality Monitor – Common ................................................................. 17
      5.2.1.4 Quality Monitor – Database ............................................................... 17
      5.2.1.5 Quality Monitor – Mail ........................................................................ 17
      5.2.1.6 Quality Monitor – External Dependencies ........................................... 17
      5.2.1.7 Quality Monitor – File System ............................................................. 17
      5.2.1.8 Quality Monitor – Analysis Tools ......................................................... 18
    5.2.2 Deployment View ............................................................................................. 18
    5.2.3 Security View .................................................................................................. 19

6 DATA DESIGN .................................................................................................................. 20

7 MEDIUM LEVEL DESIGN .............................................................................................. 22
  7.1 VIEW MANAGEMENT ............................................................................................. 22
  7.2 DATA UPLOADING FOR ANALYSIS .................................................................... 22
  7.3 ANALYSIS IN EXTERNAL PROCESS .................................................................... 23
  7.4 REMOTE DATA UPLOADING .............................................................................. 25
  7.5 IMPORT DIRECTORY MONITORING ................................................................. 27
  7.6 EMAIL NOTIFICATIONS ......................................................................................... 28
8 USER INTERFACE DESIGN ........................................................................................................30
  8.1 COMMON DESIGN GUIDELINES ......................................................................................30
  8.2 NAVIGATION HIERARCHY ..............................................................................................30
  8.3 CHANGE LOG PAGE .......................................................................................................31
  8.4 INDEX PAGE VIEWS ........................................................................................................32
    8.4.1 Security ..................................................................................................................32
    8.4.2 User Profile ...........................................................................................................33
    8.4.3 User Management .................................................................................................33
    8.4.4 Upload Management ............................................................................................34
    8.4.5 Analysis Results Review ......................................................................................35
  8.5 EMAIL NOTIFICATIONS ...............................................................................................38
9 CONCLUSIONS ..................................................................................................................40
10 FUTURE CHALLENGES ......................................................................................................42
  10.1 PERFORMANCE .............................................................................................................42
  10.2 DESIGN AND STYLING ...............................................................................................42
  10.3 ARCHITECTURE ............................................................................................................42
    10.3.1 Plugins and Modules ..............................................................................................42
    10.3.2 Customer-Specific Data in Separate Databases .......................................................43
    10.3.3 Multi-Queue Data Processing .................................................................................43
  10.4 FUNCTIONALITY EXTENSIONS ...................................................................................43
    10.4.1 Analysis Report and Real-Time Processing Status ..................................................43
    10.4.2 Alternative Analysis Results Visualization .............................................................43
    10.4.3 Data Export ..........................................................................................................44
    10.4.4 Automatic Data Import Configuration .................................................................44
    10.4.5 Projects and Project Families Management ...........................................................44
    10.4.6 Company Teams Management .............................................................................44
    10.4.7 System Monitoring ...............................................................................................44
REFERENCES ..........................................................................................................................45
APPENDIX A: FUNCTIONAL REQUIREMENTS ........................................................................48
  A. SECURITY ......................................................................................................................48
  B. USER PROFILE .............................................................................................................48
  C. USER MANAGEMENT ....................................................................................................49
  D. UPLOAD MANAGEMENT ...............................................................................................50
  E. ANALYSIS RESULTS REVIEW .....................................................................................52
APPENDIX B: QUALITY MONITOR DEPLOYMENT ................................................................54
  A. INITIAL DEPLOYMENT AND CONFIGURATION .............................................................54
  B. MAINTENANCE DEPLOYMENT .......................................................................................55
  C. SERVICE DEPLOYMENT ...............................................................................................55
  D. WEB APPLICATION DEPLOYMENT ..............................................................................56
APPENDIX C: QUALITY MONITOR CONFIGURATION ................................................................57
  A. SERVICE CONFIGURATION ............................................................................................57
    i. Stop and Uninstall Existing Service ............................................................................57
    ii. Prepare Service Workspace ....................................................................................57
    iii. Install New Service ................................................................................................58
    iv. Acceptance Testing ..................................................................................................58
  B. SFTP DIRECTORY CONFIGURATION ............................................................................58
    i. User System Account ................................................................................................58
    ii. SFTP Subsystem .......................................................................................................59
    iii. Import Directory .....................................................................................................60
    iv. Acceptance Testing ..................................................................................................60
List of Tables

Table 3.1 Use case attributes ........................................................................................................5
Table 3.2 Non-functional requirements grouped by categories ....................................................6
Table 4.1 Web toolkits comparison ..............................................................................................9
Table 4.2 Web servers comparison ..............................................................................................11
Table 4.3 Data visualization tools used in QM ..........................................................................12
Table 9.1 Performance testing results .........................................................................................40
Table A.1 UC1 Use case description ............................................................................................48
Table A.2 UC1 Functional requirements .......................................................................................48
Table A.3 UC2 Use Case description ............................................................................................48
Table A.4 UC2 Functional requirements .......................................................................................48
Table A.5 UC3 Use Case description ............................................................................................49
Table A.6 UC3 Functional requirements .......................................................................................49
Table A.7 UC4 Use Case description ............................................................................................49
Table A.8 UC4 Functional requirements .......................................................................................49
Table A.9 UC5 Use Case description ............................................................................................50
Table A.10 UC5 Functional requirements .....................................................................................51
Table A.11 UC6 Use Case description ............................................................................................51
Table A.12 UC6 Functional requirements .....................................................................................52
Table A.13 UC7 Use Case description ............................................................................................52
Table A.14 UC7 Functional requirements .....................................................................................53
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1</td>
<td>Use Case relationships between system users</td>
<td>5</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>GWT compilation process [22]</td>
<td>10</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Interactive visualizations built with D3</td>
<td>13</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Primary logical components of QM application</td>
<td>16</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Deployment view diagram</td>
<td>18</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>Database diagram – Entity Relationship model</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7.1</td>
<td>MVP and MVC patterns in GWT [36]</td>
<td>22</td>
</tr>
<tr>
<td>Figure 7.2</td>
<td>Sequence diagram – Data uploading for Analysis</td>
<td>23</td>
</tr>
<tr>
<td>Figure 7.3</td>
<td>Activity diagram – Background analysis</td>
<td>24</td>
</tr>
<tr>
<td>Figure 7.4</td>
<td>Activity diagram – Remote data uploading procedure</td>
<td>26</td>
</tr>
<tr>
<td>Figure 7.5</td>
<td>XSD Schema for content.xml upload descriptor</td>
<td>26</td>
</tr>
<tr>
<td>Figure 7.6</td>
<td>Activity diagram – Import directory monitoring</td>
<td>27</td>
</tr>
<tr>
<td>Figure 7.7</td>
<td>Class diagram – Infrastructure for email notifications</td>
<td>28</td>
</tr>
<tr>
<td>Figure 8.1</td>
<td>Page decomposition style</td>
<td>30</td>
</tr>
<tr>
<td>Figure 8.2</td>
<td>Sitemap for the Quality Monitor web application</td>
<td>31</td>
</tr>
<tr>
<td>Figure 8.3</td>
<td>Change Log page</td>
<td>31</td>
</tr>
<tr>
<td>Figure 8.4</td>
<td>Login view</td>
<td>32</td>
</tr>
<tr>
<td>Figure 8.5</td>
<td>Login panel view</td>
<td>32</td>
</tr>
<tr>
<td>Figure 8.6</td>
<td>User details view</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8.7</td>
<td>Users management view</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8.8</td>
<td>Uploads management view</td>
<td>34</td>
</tr>
<tr>
<td>Figure 8.9</td>
<td>Analysis over all uploads</td>
<td>35</td>
</tr>
<tr>
<td>Figure 8.10</td>
<td>Upload analysis details</td>
<td>36</td>
</tr>
<tr>
<td>Figure 8.11</td>
<td>Previous Architecture Analysis graph implementation</td>
<td>36</td>
</tr>
<tr>
<td>Figure 8.12</td>
<td>Architecture Analysis of projects – Case 1</td>
<td>37</td>
</tr>
<tr>
<td>Figure 8.13</td>
<td>Architecture Analysis of projects – Case 2</td>
<td>37</td>
</tr>
<tr>
<td>Figure 8.14</td>
<td>Email template in plain text format</td>
<td>38</td>
</tr>
<tr>
<td>Figure 8.15</td>
<td>Email notification in html Format</td>
<td>38</td>
</tr>
<tr>
<td>Figure 8.16</td>
<td>Email notifications with account activation info</td>
<td>39</td>
</tr>
</tbody>
</table>
1 Introduction

Software quality management plays a big role in successful construction of software systems development. Customers need to satisfy their quality standards and software development processes in more efficient way to save costs and assure customer satisfaction in the long run.

Applied Research in System Analysis (ARiSA) is an IT consulting company [1]. They create customer-specific quality models in customers' IT systems that measure and validate various success factors. To help their customers prevent future problems and to reduce maintenance costs in their software projects, ARiSA developed a "Quality Monitor" (QM) that automatically analyzes all projects and shows if they meet quality standards. QM can be customized and installed according to the customer needs.

1.1 Problem and Motivation

The current implementation of QM is web-based and allows only manual uploads. Furthermore, complex analyses are performed, but the results are displayed in a rather simplistic way. Finally, some analyses are rather time and resource consuming and a user does not want to wait until the web-frontend reports success.

To make improvements of the existing limitations, a number of maintenance procedures should be planned. The problem addressed by this thesis is to improve and extend the existing functionality, identify issues and select suitable solutions. In particular, to allow automated upload from different build processes, allow batch processing of uploads (in background) and to improve reporting.

1.2 Goals and Criteria

This section describes the goals pursued by this thesis in order to solve the problem and the criteria used for validating the goal:

- The first goal is to allow uploading to QM from build scripts, e.g., batch files, ant scripts or make files. The goal is achieved if it is possible to upload a project version efficiently to the QM for processing. The users are informed about the status of the process (e.g., upload received, processing, succeed/failed).

- The second goal is to allow efficient processing of several uploads. In the current version files from several builds can be submitted at the same time, where processing of each file could take up to 1 hour or more, and would require sufficient resources (CPU and RAM). This goal is achieved if only one file is processed in a proper way at a time; other files wait in a processing queue and are processed one by one. Processing scheduler ensures that upload import is possible (the user is authenticated and authorized to make uploads, archived file integrity is checked, read and write permissions are set, etc.). Scheduler sets failed status if upload does not fulfill predefined import requirements. Processing scheduler skips failed uploads and gets next upload from the queue.

- The third goal is to inform users about the progress and aid debugging. This goal is achieved if logs and other context data are collected, thus making debugging of the failed analysis possible. The users of the system are informed in the web-interface and by email about processing status.

- The fourth goal is to improve user experience by avoiding waiting times when viewing analysis details. This goal is achieved, if a user does not have to wait more
than 1-2 seconds to view analysis details. If it takes more time, a feedback should be provided in the form of progress bars. Graphs and other static content should be pre-calculated and updated as part of the analysis process.

- The fifth goal is to improve reporting of the results. This goal is reached if current charts present a total number for root metrics; if a history chart is present, and if existing charts can be navigated and provide context information.
- The final goal is to fix any bugs in the new or existing functionality uncovered during the time of the thesis. E.g., provide a progress bar in manual uploads.

Meeting all these goals individually and in combination is difficult because all goals are dependent on each other. On one hand, to meet a chosen goal, one needs to understand and at least partially achieve other goals. On the other hand, to meet goals in combination, each goal should be met individually, taking into account all other goals. Moreover, there are a few maintenance-specific difficulties:

- Application complexity is growing. Maintenance requires both good algorithmic base and good application domain knowledge;
- Application is already in production. Application system with real customer’s data shall be upgraded after maintenance phase;
- Application has specific domain, Software Quality. To make analysis of software, system itself must meet Software Quality requirements.

1.3 Thesis Outline

The thesis is organized into 10 primary chapters. This chapter presents an overview of the problem and main goals that are met in the thesis. The next chapter presents general information about used tools and technologies. Also chapter includes information about existing quality analysis solutions and their comparison. Chapter 3 contains structured set of technical requirements and features used for development. Chapter 4 describes the technologies and tools for QM application development, choice rationale. Chapter 5 and 6 describes the application architecture and design for QM application. Chapter 7 describes data model, and chapter 8 shows detailed screenshots of what the application user interface looks like. In final chapter, Chapter 9, we discuss results, the problems we have faced and conclusions. This chapter also covers future work directions and challenges.
2 Background

This chapter provides background information about both software quality and implementation of the Quality Monitor as application for software quality analysis.

2.1 Software Quality

In general, Software Quality is a set of goals and sub-goals (characteristics and sub-characteristics) of the software product to reflect conformance with functional requirements, design or any other stated or implied needs. Software Quality “…is the entirety of properties and attributes of a product or a process relating to their fitness to fulfil certain requirements” [14] [15].

ARiSA leverages quality of software development with different services [1]. To provide high-quality services and software for quality analysis, company works according to software standards defined in the industry.

The international standard ISO/IEC 9126 “Software engineering – Product Quality” provides general means for evaluation of software quality. The standard consists of four parts and defines the following [14]: quality model, internal and external metrics, and quality in use metrics.

We will assume that software metric (in simple way) is a combination of system characteristics and measured numerical value. According to the ISO/IEC 9126 standard software quality may be evaluated by the following main goals (system characteristics):

- Functionality – a set of metrics to verify that functions and their properties are defined and satisfy stated or implied needs;
- Reliability – a set of metrics to verify the ability of software to operate under defined conditions for a defined period of time;
- Usability – set of metrics to verify how much efforts required to use software by system end-users;
- Efficiency – a set of metrics to verify how efficient software is, e.g. performance, resources, etc.;
- Maintainability – a set of metrics to verify the amount of efforts required to maintain software;
- Portability – a set of metrics to verify the ability of software to be transferred from one operational environment to another one.

Each quality characteristic can be divided into several sub-characteristics (metrics), which can be measured by means of analysis and aggregated for the future high-level analysis.

2.2 Quality Monitor Application

Quality Monitor is a web-based application for uploading, analysis and a future review of analysis results. Current implementation supports the following features:

- Basic user management;
- Two types of user roles: Administrator and Members;
- Web based authentication;
- User profile and settings;
- Manual data uploading using web interface (data analysis is done immediately after uploading);
- Review of analysis results for current upload: calculated metrics with custom metric factors, dependency graph with filtering;
- Review of previous uploads and related analysis results;
- Static charts and graphs visualization;

The following set of metrics is analyzed in the core implementation of QM application:

1. Maintainability
   a. Lack of Testing (LOT)
   b. Complexity
   c. Coupling
   d. Global Variables (GVC)
   e. Lack of Documentation (DOC)

2. Size
   a. Lines of Code (LOC)
   b. Number of Classes (NOC)
   c. Number of Methods (NOM)
   d. Number of Interface Methods (NIM)
   e. Number of Interface Declarations (NID)

Pascal, Delphi dialect, is default language in external analysis engine.

As any other information systems, the first version of QM application has weak areas and potentialities for improvements and functionality extension. The most critical issues are described in previous sections (Section 1.1 and 1.2).

The following aspects should be taken into account for QM application maintenance and functionality extension:
- transform the application from the project into a **product**, competitive on the market;
- integration: the environment and the system shall be updated to communicate with external systems and applications, e.g., remote data uploading;
- optimization: improve data processing workflows and algorithms;
- bugs fixing and stabilization: the system should be quite stable and fit software quality requirements in the industry;
- Improve user experience: the system becomes more complex and it requires more work on design, usability, accessibility and system performance.
3 Features and Requirements Analysis

This chapter describes features and requirements to be implemented or redesigned in QM application. Each section gives basic theoretical information about purpose of requirement groups, possible use cases and scenarios.

3.1 Users

The following is the list of users considered:

- Basic User – a user that has limited access to the system: all main views, except management dashboards for users with administrative privileges only;
- Build Manager – a user that has the same level of privileges as Basic User; additional privilege is to enable remote uploads using SFTP protocol;
- Admin – a user with full web access to the system.

Figure 3.1 below shows relationships between different users in the system.

![Figure 3.1 Use Case relationships between system users](image)

3.2 Functional Requirements

Functional requirements describe behavior of the system, what the system should do. The following main rules and structure were selected to describe functional requirements:

- All requirements are mapped to corresponding use cases (UC);
- Each use case is represented by attributes provided in the Table 3.1 below;
- Neither use case describes user interface in details;
- Each use case provides basic flow and sub-flows.

Information and recommendations on how to create effective use cases are taken from resources [14], [15] and [16].

Table 3.1 Use case attributes

<table>
<thead>
<tr>
<th>№</th>
<th>Attribute name</th>
<th>Attribute definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name</td>
<td>A use case name.</td>
</tr>
<tr>
<td>2</td>
<td>Primary actors</td>
<td>Users who perform an action in one or more use case steps.</td>
</tr>
<tr>
<td>3</td>
<td>Preconditions</td>
<td>Conditions that must be always true prior to execution of use case steps.</td>
</tr>
<tr>
<td>4</td>
<td>Triggers</td>
<td>Actions or events that initiate a use case.</td>
</tr>
<tr>
<td>5</td>
<td>Basic flow</td>
<td>A main flow of events that covers normal UC execution behavior.</td>
</tr>
<tr>
<td>6</td>
<td>Sub-flows</td>
<td>Flows of events that cover a specific or a common part of a basic or an alternative flow.</td>
</tr>
</tbody>
</table>
7 Alternative flows Flows of events that cover optional and/or exceptional execution behavior in connection to a basic flow.

The following list gives an overview of all UCs applicable to maintenance of QM application:
- All users shall be authenticated in the system before they are able to access any application resources;
- All users can close current session and leave the system;
- All users can view and change their own user settings in the online profile;
- Administrators can manage users registered in the system: add, delete, update, disable, etc.;
- All users can manage uploads registered in the system: make new uploads or delete already analysed ones;
- Build managers can upload data for analysis remotely as part of continuous integration process established in a company. The most applicable procedure is the usage of known tools and builds scripts;
- All users can review end analysis results: analysed sources structure, metrics values, architecture analysis of projects, etc.

A detailed description of each use case can be found in the Appendix A.

3.3 Non-functional Requirements
It is important to consider non-functional requirements (NFRs) in the information systems as well. NFRs describe how software will/will not behave according to functional requirements. They are also known as software qualities (Section 2.1). Table 3.2 below describes NFR categories and concrete requirements applicable to QM application. Information about NFRs is taken from the following documentation: [17], [18] and [19].

Table 3.2 Non-functional requirements grouped by categories

<table>
<thead>
<tr>
<th>Requirement category</th>
<th>Requirement description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td></td>
</tr>
<tr>
<td>Hours of operation</td>
<td>NFR1. The system shall be available 24x7, except the maintenance time. Note: this does not include Internet connectivity issues.</td>
</tr>
<tr>
<td>Capacity and scalability</td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>NFR2. The system shall be able to process several subsequent uploads (processing shall be done in the background). NFR3. Up to 100 users in total can use the system.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>NFR4. Up to 5 users may be using the system at any given period of time.</td>
</tr>
<tr>
<td>Storage</td>
<td>Not applicable (N/A) for QM version 1.1.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Year-on-year growth</td>
<td>NFR5. Average uploads number is 1-2 uploads per week for 1 client.</td>
</tr>
<tr>
<td><strong>Maintainability</strong></td>
<td></td>
</tr>
<tr>
<td>Architecture standards</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| Coding standards        | - Code conventions for the Java language [20]  
                          - Coding conventions for Google Web Toolkit (GWT) developers [21]                   |
| Design standards        | N/A                                                                                                                                        |
| **Performance**         |                                                                                                                                             |
| Response times          | NFR6. Acceptable response time is 1-2 seconds. If it takes more time to respond, progress feedback shall be provided.                        |
| Processing times        | N/A                                                                                                                                        |
| Resource constrains     | NFR7. Hardware minimal requirements:  
                          - Multi-core processor (at least 2 cores);  
                          - 1.5 GB RAM;  
                          - Disk space: 5 GB.                                                                 |
| **Recovery**            |                                                                                                                                             |
| Restore time            | N/A                                                                                                                                        |
| Backup time             | N/A                                                                                                                                        |
| **Security**            |                                                                                                                                             |
| Login                   | NFR8. The system shall have root administrator account (access to user management subsystem).                                               |
| Password                | NFR9. All information shall be protected from unauthorized access without password or public key.  
                          NFR10. User password shall be mandatory.  
                          NFR11. Shall be no expiration date for user password.                                     |
| User session            | NFR12. The system shall uniquely identify all users who upload data to the system.  
                          NFR13. Not identified users shall not be permitted to the system.                         |
| Data encryption         | NFR14. Any data transferred to and from the system via public networks shall be encrypted.                                                 |
| **Compatibility**       |                                                                                                                                             |
| Shared applications     | N/A                                                                                                                                        |
| Internet browsers       | NFR15. The following browsers shall be supported:  
                          - Firefox 4.0 or higher;  
                          - Internet Explorer 7 or higher;  
                          - Safari 5 or higher;  
                          - Chromium and Google Chrome;  
                          - Opera 10 or higher;                                                                 |
| Standards               | NFR16. The system shall be compliant with XHTML 1.0 standard.  
                          NFR17. The system shall be compliant with CSS3 standard.                                    |
| Operation systems       | NFR18. The following operation systems shall be supported:  
                          - Linux (Ubuntu v.10 or higher) – prod. environment;                                       |
<table>
<thead>
<tr>
<th>Platforms</th>
<th>NFR19. X86 and x64 architectures shall be supported.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows (Win7/Win8) – dev. environment;</td>
<td></td>
</tr>
<tr>
<td>Mac OS (v.10 or higher) – dev. environment.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usability</th>
<th>NFR20. The graphical design shall be consistent, the same in the whole application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look and feel standards</td>
<td>NFR21. User interface shall be in English.</td>
</tr>
<tr>
<td>I18n, l10n</td>
<td>NFR22. Uploaded source code shall be removed after analysis. Only file structure and analysis results are allowed to store on the server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error-handling</th>
<th>NFR23. All system and analysis errors shall be recorded to the event log.</th>
</tr>
</thead>
</table>
4 Technology Rationale
This chapter describes technologies and tools that are used for QM web application implementation. Also the chapter describes choice rationale for each selected technology.

Some technologies and tools have already been used for the core version implementation, though the motives for the chosen selection of technologies is not enough documented. Therefore, the main purpose of this chapter is to understand existing technology alternatives and select the most suitable. If needed, maintenance of QM application will be done on top of new technology stack.

4.1 Web Toolkit
This section describes web framework selected for QM application development.

4.1.1 Choice Rationale
The main technology requirements for web application development are:
- Java or C# programming language;
- Ready for production use (non-experimental, released);
- Cross-compilation to JavaScript (JS) and possibility to use native JS code – to reduce a number of used technologies and speed-up application development;
- Cross-platform development (Windows, Linux and Mac operation systems);
- Reusable components and feature-complete standard library;
- Browsers compatibility: cross-browsers issues, modern web-standards, etc.;
- Client-side development to minimize web server loading;
- Full-featured debugging;
- Internalization and localization support;
- Third-party and open-source libraries.

We have analysed several toolkits that are available on the market and fulfil requirements described above. For comparison the following attributes were selected: author, licensing, number of fulfilled requirements, supported languages, full-cycle development (FCD) (+) or just code generation (–), and generated code quality (CQ – High, Medium and Low). We will rate each technology on a scale of 1 to 5, with 5 being the highest. Here goes the list of technologies and their comparison:

<table>
<thead>
<tr>
<th>Technology</th>
<th>Author</th>
<th>License</th>
<th>Languages</th>
<th>CQ</th>
<th>FCD</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>GWT</td>
<td>Google</td>
<td>Apache 2.0</td>
<td>Java</td>
<td>H</td>
<td>+</td>
<td>5</td>
</tr>
<tr>
<td>JSIL</td>
<td>K. Gadd</td>
<td>MIT/X11</td>
<td>C# 4.0</td>
<td>L</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>Script#</td>
<td>N. Kothari</td>
<td>Apache 2.0</td>
<td>C# 4.0</td>
<td>H</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>SharpKit</td>
<td>SharpKit Software Ltd.</td>
<td>GPL3</td>
<td>C# 4.0</td>
<td>H</td>
<td>+</td>
<td>3</td>
</tr>
</tbody>
</table>

As result we selected 5 toolkits that can translate code written in Java or in one of the .NET family languages. The most advanced toolkit is GWT implemented by Google. Target programming language is Java. It provides full-cycle development process and fulfills the requirements described above. There are no released solutions from Microsoft for .NET platform. Third-party toolkits are represented either just by code generation compiler or by a set of separate tools that do not provide competitive response to Google’s GWT.
GWT completely fulfills the following requirements: NFR7, NFR15, NFR16, NFR17, NFR18 and NFR19.

4.1.2 Google Web Toolkit

Google Web Toolkit (GWT) is a web framework that allows building high-performance AJAX-enabled web applications written in Java. AJAX (Asynchronous JavaScript and XML) is a web-development technique for asynchronous client-server communication in web applications. The main idea is to write back-end and front-end functionality completely in Java. Front-end transformation to JavaScript is performed automatically by GWT compiler with full optimization and obfuscation cycle, including high-compatibility with most of the modern web-browsers.

GWT consists of several components:
- GWT compiler – translates source code written in Java into pure optimized JS (Figure 4.1);
- JRE emulation library – provides a big subset of the standard Java class library implemented in JavaScript;
- Web UI class library – provides a predefined class library for building web interface using widgets;
- GWT Development Mode – provides standard tools and instruments to debug GWT applications as native Java code.

![GWT compilation process](image)

Design and styling of pages written in plain HTML are still the responsibility of frontend developers. GWT responsibility is to unify development of the client and server sides, where communication is implemented via remote procedure calls (RPC). This is a standard way of client-server communication in GWT.

Information about GWT toolkit is available on the official website [23].

4.2 Web Server

This section describes web server selected for QM application development.

4.2.1 Choice Rationale

As a consequence of the selected technology for web development, the main requirements for a web application server are:
- Possibility to serve Java servlets;
- Good integration with development environments;
- Easy deployment procedure and configuration;
- Open-source implementation and possibility to use in commercial projects;
- Cross-platform implementation (Windows, Linux, Mac);
- Compatibility with existing software infrastructure defined in ARiSA.

We have analyzed several web servers that are available on the market. For comparison the following attributes were selected: licensing, features completeness, open-source vs. commercial, and popularity. The table of technologies and their comparison is provided below:

Table 4.2 Web servers comparison

<table>
<thead>
<tr>
<th>Server</th>
<th>License</th>
<th>Adv. Features</th>
<th>Open-source</th>
<th>Popularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Tomcat</td>
<td>Apache 2.0</td>
<td>-</td>
<td>+</td>
<td>High</td>
</tr>
<tr>
<td>Jetty</td>
<td>Apache 2.0, EPL</td>
<td>-</td>
<td>+</td>
<td>Medium</td>
</tr>
<tr>
<td>GlassFish Server</td>
<td>CDDL, GNU GPL</td>
<td>+</td>
<td>-</td>
<td>Medium</td>
</tr>
<tr>
<td>RedHat JBoss</td>
<td>GNU GPL</td>
<td>+</td>
<td>-</td>
<td>Medium</td>
</tr>
<tr>
<td>IBM WebSphere</td>
<td>Commercial</td>
<td>+</td>
<td>-</td>
<td>Medium</td>
</tr>
</tbody>
</table>

The most suitable web servers which fulfill defined requirements are Apache Tomcat and Jetty. Both servers are lightweight implementations, easy to configure and have good integration with Eclipse development environment. Apache Tomcat server is popular as container for Java servlets, and Jetty is good for static content serving. Considering those two alternatives, we selected Apache Tomcat, since ARiSA already has several projects hosted on this server, as well as maintenance and support experience.

4.2.2 Apache Tomcat

Apache Tomcat is an open-source web server for web applications. It is developed by Apache Software Foundation (ASF). Mainly Apache Tomcat is used for hosting of Java servlets (program classes implemented in Java programming language that are able to process incoming web requests and return result in the form of a web response) and Java server pages (JSP). Configuration is done via management tools and configuration files written in XML.

The main components of the Apache Tomcat are:
- Catalina – container that implements servlet and JSP pages specification defined by Sun Microsystems;
- Coyote – HTTP Connector which implements HTTP 1.1 network protocol;
- Jasper – JSP engine which implements JSP pages specification.

4.3 Database Server

This section describes a database server selected for QM application development.

4.3.1 Choice Rationale

Taking into account technologies described in the previous sections, we selected MySQL database, since at the moment it is the most popular open-source solution on the market. ARiSA already has several projects which store data in MySQL database, as well as maintenance and support experience. No changes are required in scope of maintenance.
4.3.2 MySQL Overview
MySQL is an open-source relational database management system (RDBMS) developed by Sun Microsystems (now Oracle). It is available for the most of the widely used operation systems and currently is the most used open-source database server.

MySQL as a database platform offers the following features:
- Support of ANSI SQL 99 standard;
- Stored procedures support;
- Triggers and cursors;
- Transaction management;
- Data replication support;
- Database level caching;
- Full-text indexing and search;

4.4 Data Visualization Toolkit
This section describes data visualization toolkit selected for QM application development.

4.4.1 Choice Rationale
Current implementation of QM uses static images for charts and graphs visualization. Image rendering is done on the server using different tools. See the complete list of tools together with descriptions of both encountered issues and lessons learned below.

Table 4.3 Data visualization tools used in QM

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Description and encountered issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>gml2pic</td>
<td>“Gml2pic is a command-line utility which converts graph layout stored as GML file into a graphics file. Supported graphics formats are the bitmap formats PNG, JPEG, TIFF and the vector graphics formats SVG, PDF, and EPS.” [31].</td>
</tr>
<tr>
<td></td>
<td><strong>Encountered issues:</strong></td>
</tr>
<tr>
<td></td>
<td>1) Generated images for some graphs are big in size, and thus it takes time to generate an image, download it to the client and then show it on the client area in the browser. As a result, we got long response time on the client side;</td>
</tr>
<tr>
<td></td>
<td>2) Big images which do not fit into the drawing area in the browser are very difficult to understand. As a result, we have bad readability, hence bad understandability;</td>
</tr>
<tr>
<td></td>
<td>3) Additional physical space on the server is required to store cached images;</td>
</tr>
<tr>
<td></td>
<td>4) Low interaction level, not possible to make client-side manipulations: adjustments, data filtering, etc.</td>
</tr>
</tbody>
</table>
|           | **Lessons learned:** Server-side graph generation in the form of static images is not suitable for client-side data analysis, especially for big
graphs, or requires advanced algorithms to transfer and present big images.

| BIRT | Business Intelligence and Reporting Tools (BIRT) - “is an Eclipse-based open source reporting system for web applications, especially those based on Java and Java EE. BIRT has two main components: a report designer based on Eclipse, and a runtime component that you can add to your app server. BIRT also offers a charting engine that lets you add charts to your own application.” [6] |
| Use case: generate static chart images in PNG and SVG formats directly from the database using capabilities of BIRT engine. |
| **Encountered issues:** |
| 1) Several requests to the server to display similar charts represented by images. As a result server is highly-loaded; |
| 2) Additional physical space on the server is required to store cached images; |
| 3) Low interaction level, impossible to make client-side manipulations: adjustments, data filtering, etc. |
| **Lessons learned:** BIRT is good for reporting, though it is not suitable for high-load client-side applications. |

To make data visualization more interactive and less “expensive” for the server, we made a decision to replace static charts and graphs on client-side visualization using D3. There are no advanced alternatives for data-driven documents manipulation. The main advantages and the high-level comparison with other frameworks are described in the next section with technology overview (Section 4.4.2).

### 4.4.2 Data-Driven Documents

Data-Driven Documents (D3) [24] is an easy-to-use JS library for manipulating documents based on data. Powered by known web standards (HTML [9], SVG [5] and CSS [10]) it provides advanced toolset for data manipulation and visualization: standard document object model (DOM) [7] manipulation, interaction and animation of dynamic behaviors.

![Interactive visualizations built with D3](image)

**Figure 4.2 Interactive visualizations built with D3**

D3 provides a standard predefined library of functions to make elements manipulation, creating SVG objects, styling and subsequent animation.
Main features [8]:
- A declarative way for documents modification – operating on selections, arbitrary node sets;
- Dynamic properties similar to jQuery [25] and Prototype [26] javascript frameworks;
- Unlike Processing [27], Raphaël [28], or Protovis [29] data visualization frameworks, D3 uses vocabulary marks from W3C [11] web standards. That means that the main goal is transformations, not representation;
- D3 supports several data formats for input data: JSON [12], CSV [13] or geoJSON [30]. Custom data readers represented by JS function are acceptable as well;
- Well-defined transitions to make animation of different complexity;
- Simple debugging using a browser built-in debugger.

4.5 Technology Choice Summary
There are four main technologies selected for QM application maintenance. Three of them (GWT, Apache Tomcat and MySQL) completely fulfill requirements for QM application. The only change made in the technology stack was that libraries and tools for static content generation were replaced with more advanced interactive alternative – D3 JS library. It allows us to addresses known issues found in QM application of version 1.0.
5 Architecture Design

This section describes the whole application architecture and design.

5.1 Architecture Strategy

This section introduces general strategy and approaches that shall be used for development of QM application. Most of the sub-sections below are linked to functional and non-functional requirements that are fully or partially fulfilled by each section.

5.1.1 Modularity

QM application shall be module-based. Two of modules are Web and Daemon packages. They separate executable applications connected to each other only by a file system and a database. This allows maintaining modules separately, without shutting down the system completely. Such approach partially fulfills the non-functional requirement NFR1.

5.1.2 Localization

There are no specific requirements for application localization in the first releases. The main language of the system is English. This fulfills the non-functional requirement NFR21. Potentially, next releases will include translation of the web part into several languages. To make this possible all constant literals should be translated and localization mechanisms should be incorporated. Localization is out of scope in current master thesis work.

5.1.3 Errors Handling

There are two types of errors considered: critical and operational errors. Such error handling strategy completely fulfills the non-functional requirement NFR23.

5.1.3.1 Critical Errors

Critical errors are a set of low-level errors and exceptional cases that occur unexpectedly. They can occur due to software bugs and defects, functional misbehavior, hardware errors or any other type of errors which makes application execution either impossible or prevent the application from being fully functional.

   Examples of critical errors:
     - Index out of range exception;
     - Null reference exception;
     - Division by zero exception;
     - File system access errors.

   Errors handling strategy in QM application:
     - Exception handling routines catch errors and create a corresponding record in the event log with context information;
     - Not caught unexpected errors are also written to the event log. For debug purpose client-side logging is supported.

5.1.3.2 Operational Errors

Operational errors are a set of errors which are expected by design and treated as errors of normal application execution.

   Examples of operational errors:
     - A user enters data in invalid format and/or data does not pass validation;
- Database or internet connectivity errors;
- Configuration errors.

Errors handling strategy in QM application:
- Errors shall be handled in an acceptable for application way;
- The system informs a user about occurred errors and provides instructions on how to fix an issue.

5.2 Architecture Views
We considered 5 main architecture views: logical, development, deployment, security and scenarios. This section describes only logical, deployment and security views. Development view is represented by medium level design and described in Section 7. Scenarios is a small set of use cases, which are described in Section 3.2.

5.2.1 Logical View
This section introduces primary logical components of QM application (Figure 5.1).

QM application is logically divided into two parts: front-end and back-end. Back-end is responsible mainly for data analysis (different languages, different outputs, etc.). Front-end is responsible for system composition, handling and analyzing data using back-end, presenting analysis results to the end user.

There five colors were used in the figure above:
1) grey – third-party libraries and frameworks;
2) orange – packages that represent logic layers and common functionality; they are used by ‘application’ packages;
3) blue – packages that represent separate features; next releases of QM application will include decomposition of functionality on feature base;
4) olive – mainly ‘application’ packages;
5) aqua – logical packages that represent functionality or resources used by QM application.

5.2.1.1 Quality Monitor – Daemon
This package represents the daemon service, a process continuously running in the background. Daemon is responsible for remote uploads import and data analysis. At the base of the process is a scheduler which executes several tasks in parallel. Each task is a queue of actions, at any time only one action is executed. One of the main concepts introduced in the daemon is analysis in the external process. This helps to control resources allocation in a more efficient way and decouples analysis infrastructure and analysis itself.

5.2.1.2 Quality Monitor – Web
This package represents a web application responsible for client user interface (UI). The main requirement for this package is to present analysis results; it should not make analysis itself. Analysis logic was moved completely to the daemon service.

5.2.1.3 Quality Monitor – Common
This package provides commonly used functionality in QM application packages. The following features are implemented in the current version: file utilities, string utilities, path utilities, process management, common configuration management, encryption utilities, and daemon commons.

5.2.1.4 Quality Monitor – Database
This package provides a data access layer for QM application. Current version is based on Hibernate, an open source object-relational mapping framework. Also this package contains logic related to data serialization to different formats, for example, XML.

5.2.1.5 Quality Monitor – Mail
This package represents ‘email notifications’ feature, decoupled into a separate package (most features in next releases should be represented by separate packages). Current version contains functionality for: building email templates, building and sending email notifications, templates themselves in HTML and plain text formats.

5.2.1.6 Quality Monitor – External Dependencies
This package is logically represented by a set of third-party libraries and frameworks used for implementation of the common functionality in QM application. Most of them are directly referenced by other packages.

5.2.1.7 Quality Monitor – File System
This package is a logical representation of the file system in QM application. It is used to store application resources and also as a temporary storage for remotely uploaded data to the import directory.
5.2.1.8 Quality Monitor – Analysis Tools
This package is a logical representation of a set of external tools used for data analysis. QM application does not implement analysis itself. The main purpose of QM application is to store analysed data, visualize data, provide infrastructure for data analysis. All analysis logic is represented by pluggable modules and/or integrated tools which are developed internally by ARiSA and/or third-party community.

5.2.2 Deployment View
This section introduces the topology of the physical components that are deployed and their associations. The topology is represented by a top-level deployment diagram in Figure 5.2 below. Each physical component contains one or more system packages deployed.

![Deployment view diagram](image)

Figure 5.2 Deployment view diagram

Figure 5.2 is divided into two parts: client and server sides. Each component actually represents node type, therefore, each node can be duplicated or represented by multiple instances. Nodes redundancy for better reliability is one of the potential use cases. Opposed to it is a use case where system instantiation shall be compact and not distributed over several physical servers. In this case all nodes can be divided just logically and merged into a single organization unit where one physical server contains all nodes: web, daemon, database, static content storage and administration of users. Users could be managed by the hosting operation system itself.
There are a few node types:
1) Application server – hosts the web application and daemon for background data processing, all of the packages are deployed to this server;
2) Database server – hosts database with all analysed and application-specific data;
3) File server – stores static content: temporary files, logs etc.;
4) User Management Server – a server which handles access to the server; on Windows it could be Active Directory;
5) Continuous Integration Server – an external organization unit in connection with QM application; makes remote data uploading for analysis;
6) Maintenance Unit – an external organization unit responsible for QM application infrastructure maintenance;
7) Client Unit – actual clients of QM application; these units should have web-browser preinstalled to have access to the system via web interface;
QM application deployment and configuration processes are described in Appendix B and Appendix C respectively.

5.2.3 Security View
The diagram in Figure 5.2 is divided into two parts: a private and a public network. Splitting into security zones provides a better understanding of how secure data transferring between nodes is and how it is implemented.
Communication between private and public security zones is done using the following network protocols:
1) HTTP – web access via public networks;
2) SSH + SFTP – secure data uploading via public networks;
3) SSH and/or SFTP – servers maintenance via public networks;
4) TCP/IP and other not secure protocols – internal operation via private network.
Secured data transfer protocols partially fulfil the non-functional requirement NFR14.
6 Data Design

This section describes an existing database model in QM application for a better understanding of the system.

Figure 6.1 Database diagram – Entity Relationship model

Figure 6.1 shows the unchanged database relational model. Minor changes were made to implement background data analysis: added one-to-many foreign key reference User(id) –
Upload(user_id) to keep track of who uploaded data, extended User table to include flags related to email notifications.

Here goes a description for each table defined in the database:

1) User Tables
   The “User” table contains a list of users registered in the system. Each user can specify their email, phone number, enable or disable email notifications (“notifyByEmail” field). Administrators can enable or disable user account (‘active’ field).

2) Session Table
   The “Session” table contains a list of user sessions. Upon user login the system saves a session identifier to this table. This allows to authenticate user and to track active sessions.

3) Upload Table
   The “Upload” table contains a list of uploads with attributes (file name, upload date, version, description, current analysis status, user who made an upload). The UploadUUID field contains a randomly generated identifier created during data import phase before saving data to the database.

4) Program Table
   The “Program” table contains a list of programs collected during source code analysis.

5) File Table
   The “File” table contains a list of files collected during source code analysis.

6) Program_File Table
   The “Program_File” table represents a junction table which is a result of many-to-many relationship between “Program” and “File” tables. The table links a collection of files to some program organization unit.

7) Parameters Table
   The “Parameters” table contains the following parameters: median, deviation and count. These parameters are used during data analysis for a software metrics calculation. The ‘entryType’ field represents a type of source code organization unit: Upload, Program or File.

8) Measurement Table
   The “Measurement” table contains calculated software metrics and a hint for each metric.
7 Medium Level Design

This section describes medium-level design for the whole QM application. It is represented by several UML diagrams and algorithm descriptions.

7.1 View Management

QM web application architecture uses recommended GWT approaches:
- Remote procedure calls (RPC) [33] – client-server communication;
- Activities and places framework (MVP pattern) [34] – view management and view transitions;

Most of the listed approaches are used in the simplified version now, because they were incorporated only during maintenance phase in scope of the thesis. The following diagram shows raw implementation of approaches:

Figure 7.1 MVP and MVC patterns in GWT [36]

7.2 Data Uploading for Analysis

QM application was refactored to enable analysis run asynchronously in the background. For that particular reason QM Service with scheduler and two other tasks were implemented.
Each task is a separate thread with queue which processes uploads one by one. Two upload data sources are possible: using web interface and via SFTP protocol. A complete sequence of actions for background analysis is described on the diagram below.

Figure 7.2 Sequence diagram – Data uploading for Analysis

The import procedure and the analysis process are described in the next sections.

7.3 Analysis in External Process
The diagram below demonstrates the analysis process introduced in QM application:
To provide more controlled resources allocation, QM Service runs analysis in external process. This has several advantages:

Figure 7.3 Activity diagram – Background analysis
- When the process execution is finished, it completely releases all resources (native handlers, allocated memory, etc.);
- Handles possible errors in the third-party components, for example memory leaks;
- Termination of long-running or deadlocked operations by timeout.

Analysis part, which is intended to run in the external process, is designed to catch all analysis errors and mark upload as Failed. Upload analysis failure reasons are logged in the event log. When the process execution is finished, the main thread checks exit code and upload analysis status. If analysis status is not equal to Processed or Failed, an upload will be automatically marked as Failed.

### 7.4 Remote Data Uploading

Remote data uploading to the import directory was introduced in QM application. This makes it possible for continuous integration servers to upload data remotely as a part of the standard build process.

Here goes the list of requirements for remote data uploading:
- Simple solution based on existing tools and environment capabilities;
- Possibility to upload data using provided credentials, using secured protocols (HTTPS, SSH, SFTP, etc.);
- Status reporting;
- Quick and easy integration using build scripts;
- Remote operation system is Windows (customer-specific requirement).

Based on the requirements, we selected three alternatives:
1. Uploading using Ant or Batch scripts to the remote directory on Linux server using SSH credentials. Access should be granted only for the specified directory;
2. Uploading via HTTPS using manually developed command-line tool and web services as server front-end;
3. Uploading using Batch scripts via SFTP secured protocol to the rooted import directory.

The first data uploading alternative was rejected, because by default all users on Linux (production environment) have read-only access. This makes server configuration complicated. The second alternative was rejected as well. In order to implement proposed approach, additional effort is required for the tool and web services development. Another problem with the second alternative is that it requires future support and maintenance in case of any issues.

The final decision was to use the third alternative. It completely fulfills security and simplicity requirements defined above. From the configuration perspective, it requires some effort to make initial server configuration, but in future it can be automated by means of shell script and minor changes in QM application. From a customer point of view, remote data uploading is represented by batch script in a few lines of code and a few support executable files. Thus, it is acceptable as a solution for remote data uploading.

To make data uploading possible using SFTP protocol to the server secured by SSH protocol we selected PuTTY SSH Client [37]. PuTTY is the most popular and stable tool for SSH connection on Windows operation system. The server shall be configured to accept and authenticate SSH connections using login and private/public RSA keys. Server-side configuration is described in the Appendix B, Section b.
The diagram below demonstrates the sequence of operations for uploading zipped archive with source code to QM server via SSH and SFTP protocols:

![Diagram of the sequence of operations for uploading zipped archive with source code to QM server via SSH and SFTP protocols.](image)

Upload archive should contain an optional upload descriptor file – content.xml with upload attributes. XSD Schema which describes XML format is showed in the figure below.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="contentInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="version" type="xs:decimal" minOccurs="0" maxOccurs="1" />
        <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 7.5 XSD Schema for content.xml upload descriptor

Server-side processing of the remotely uploaded data is represented in Figure 7.6.
7.5 Import Directory Monitoring
The diagram in Figure 7.6 below demonstrates the sequence of operations performed to move uploaded data from the import directory to the processing queue directory.

![Diagram of import directory monitoring](image)

There are two possible main operational errors:
- Submitted data does not fulfil predefined requirements for data import. In this case data importer marks upload as Failed and deletes all uploaded data;
File system monitoring errors. For example, an archive file was locked during import phase and cannot be moved or deleted after successful import. In this case a file path will be added to the temporary ‘black-list’ to skip it during the next monitoring cycle. Upon QM service restart the temporary list with blocked uploads will be cleared.

7.6 Email notifications
To inform users about analysis progress, QM application was extended with e-mail notifications for each analysis phase (pending, in progress, failed, completed, etc.). This feature was developed in a generic way to support any type of notifications needed in QM application.

As shown in Figure 7.7 below, current implementation consists of 4 main class types: notification manager, template reader, mail notification and binding info.

![Class diagram – Infrastructure for email notifications](image)

MailNotificationManager static class is responsible for sending email notifications, manages SMTP server connection and any routines for emails sending.

28
MailTemplateReader class is responsible for reading predefined email templates in HTML and plain-text formats. Reader returns instance of the StringTemplate class from StringTemplate library. Mail templates examples are presented in Section 8.5.

StringTemplate is “...a java template engine (with ports for C#, Python) for generating source code, web pages, emails, or any other formatted text output....” [41]. This allows automated emails generation from template using attributes (binding objects) and attribute expressions written directly in template.

MailNotification abstract class is responsible for email generation (sender, recipients, header, body, etc.) using MailTemplateReader class and binding object classes (EmailStBinding, UploadStBinding, SiteStBinding, etc.) which keep information for template engine attributes. Direct realizations are UploadStatusEmail and AccountActivationEmail classes.
8 User Interface Design

This chapter describes user interface of QM Application. Also it includes common design guidelines. UI implementation shall fulfill the requirement NFR20 (The graphical design shall be consistent, the same in the whole application.).

8.1 Common Design Guidelines

All pages in QM web application have common design:

1) Decomposition into three logical parts (Figure 8.1):
   a. Header (1) with company logo and login panel;
   b. Body (2) with currently loaded view;
   c. Footer (3) with copyrights, current version and supported standards (XHTML and CSS).

![Figure 8.1 Page decomposition style](image)

2) Common color schema:
   a. Black color for most of the labels;
   b. Black color for border of Level 1 organizational blocks;
   c. Silver color for border of Level 2 organizational blocks;
   d. Grey color for read-only fields and labels;
   e. Standard HTML styling for navigation links;
   f. Colored icons for ‘edit’ buttons;

3) Navigation history:
   a. The user shall be able to use navigation buttons of internet browser;
   b. History navigation shall be possible at least for main views;
   c. Each view shall accept parameters.

8.2 Navigation Hierarchy

User interface consists of two physical pages: index and change log, where each page has a few separate views. Also the sitemap diagram (Figure 8.2) describes possible dialog boxes,
shows navigation connections and has optional descriptions for diagram organization elements.

The following colors are used to describe web application security zones: light blue (anonymous users), light green (authenticated user) and light orange (administrators).

Figure 8.3 Change Log page

8.3 Change Log Page
Change log page represents the history of all implemented features or fixed issues.
8.4 Index Page Views
GWT application is represented mainly by one single page with several views. Such single page in QM application is Index page. Different views will be explained by screenshots and corresponding description below in sub-sections.

8.4.1 Security
The security view is represented by login dialog that allows a user to log into the system, and login panel that shows who is currently logged into the system and provides possibility to log out (Figure 8.4 and Figure 8.5).

Figure 8.4 Login view

The login view was completely redesigned, according to the product styling defined by Softwerk. This improves user experience and brings QM application to a product stage. After login a user will be redirected to index view shown on Figure 8.5 below.

Figure 8.5 Login panel view

No changes are made for Login panel (see blue selection on the screenshot above), just minor refactoring to keep common coding style in QM application.
8.4.2 User Profile
The user profile view (Figure 8.6) allows user to edit their online profile and private user settings.

![User's details view](image)

**Figure 8.6 User details view**

The user details view allows users to view (see blue selection in the left part of the screenshot above) and edit (see blue selection with arrow in the right part of the screenshot above) user settings.

Only one change was made during application maintenance – optional setting “Email notifications” was added. Users will receive email notifications with current analysis status for uploaded data if this option is enabled.

8.4.3 User Management
User management view is represented by a table with a list of users. Only administrators are allowed to manage users registered in the system.

![Users management view](image)

**Figure 8.7 Users management view**

The feature ‘Temporary password’ with email notification was added in the latest release. When administrator creates a new user, the system will automatically generate temporary password, otherwise, administrator shall provide user password manually. If email address
is provided, the system will automatically send email notification with temporary password to a new user and instructions on how to change password and adjust user settings.

Also this table was extended with one more column which allows to enable or disable email notifications for a selected user.

8.4.4 Upload Management

The upload management view (Figure 8.8) is represented by a table with uploads list and expandable upload panel.

![Figure 8.8 Uploads management view](image)

The following features and improvements were implemented:

- (1) Asynchronous data uploading with progress reporting;
- (2) Real-time analysis status reporting: current version of QM application processes uploads in the background one by one; for uploads that are ready for analysis or analysis is in the progress, QM UI requests latest status and indicates all changes on the server;
- (3) Event logs downloading: analysis and server logs;
- (4) Fixed sorting of uploads by version: old algorithm sorts ‘version’ column in alphabetical order, new sorting algorithm supports software version in the format {major.minor.revision.build}. Major version number is required. All other version numbers are optional.

By clicking on the ‘View Analysis Result’ icon, a user will be redirected to the next view.
8.4.5 Analysis Results Review

There are three different views for page: analysis over all uploads (Figure 8.9), upload analysis details (Figure 8.10) and architecture analysis projects (Figure 8.12).

The following features and improvements were implemented:
- Updated color schema, hence now it is more easy to link metrics to colors visually;
- Sorting is updated in the same way as for Uploads (Section 8.4.4) to make it possible to sort by version;
- Added a history bar chart with two modes: clustered column bar chart and stacked column bar chart;
- Inline row charts were completely replaced for client-side rendering, absolute value is added for each row chart;
- Table rendering was updated – replaced with simpler implementation that does not support ‘edit’ mode. This speeds up table generation and data binding;
- Changed behaviour of table loading: old implementation loads all rows and then displays completely loaded table, while new implementation loads rows one by one; this improves status reporting: a user is able to view the first rows while last rows are not yet loaded;
- Data access layer was updated for more efficient data retrieving;
- Presentation layer was updated to be more generic and to reuse already existing functionality.
The next screenshot (Figure 8.10) shows analysis results for selected upload. It consists of three parts:
- (1) Upload attributes (upload date, upload version, description, file name, size, number of files, etc.);
- (2) Software metrics and factors selection panel, the same as for ‘Analysis over All Uploads’ view;
- (3) Analysis details for projects and files; display mode can be changed using the tool box above the table with projects/files.

![Figure 8.10 Upload analysis details](image)

For each project a user can view architecture analysis graph. Previous implementation of graph is showed on Figure 8.11 below. Analysis of such graph is complicated because labels and graph edges are not visible if architecture graph contains too many nodes and edges between them.

![Figure 8.11 Previous Architecture Analysis graph implementation](image)

New implementation of the graph is presented in the Figure 8.12 below.
The screenshot shows force-directed graph with one hovered node. On hover, only hovered node and linked nodes are shown, all other are transparent. Complete visible graph is shown on the next screenshot below (Figure 8.13).

The following features and improvements were implemented:
- Static image in PNG format was completely replaced with interactive client-side version implemented using D3 graphics library;
- Display node attributes on hover;
- Fix node position on dragging event;
- Release position of node on double click event;
- Distinct different file types, for example, ‘.dpr’ files are with additional border.
8.5 Email Notifications

All subscribed users receive notifications in HTML (Figure 8.14) and plain text (Figure 8.15) formats for all new uploads. To enable or disable email notification a user should change corresponding option in the user profile.

Moreover, account activation emails were implemented (Figure 8.16) to extend user management. When administrator creates new users, the system sets randomly generated password and sends notification to email specified by administrators. Administrators have
possibility to reset the password and type custom password. After the first login a user could navigate the user profile and change password.

Figure 8.16 Email notifications with account activation info


9 Conclusions

The aim of this thesis was to improve and extend the implemented version of QM web application developed in ARiSA. The following goals have been reached.

The **first goal** was to allow uploading to QM from build scripts. In order to achieve the first goal, we have developed QM daemon service which works in the background and imports data for analysis from the import directory (Sections 7.2, 7.4 and 7.5). Uploading to the import directory is secured using SSH and SFTP protocols (Section 5.2.3). As soon as upload is imported and scheduled, subscribed users receive an email notification. Moreover, manual uploading procedure uses the same import directory, thus, as a result, we have a generic solution with self-testing. No need to test import functionality because it is already tested by simple manual uploading via web interface.

The **second goal** was to allow efficient processing of several uploads. In the previous version it was possible to submit files from several builds at the same time, where it would take up to 1 hour or more, and would require sufficient resources (CPU and RAM). In order to achieve the second goal, we extended QM daemon service with processing queue which makes analysis of uploads one by one (Section 7.3). For each analysis phase subscribed users receive an email notification with current status information.

The **third goal** was to inform users about progress and aid debugging. This goal has been already partially fulfilled by the first and the second goals, where email notifications were implemented (Section 7.6 and Section 8.5). In addition, we updated the user interface to report current status for uploads that are either scheduled or in the progress of analysis. The possibility to download server logs via web interface for each upload has been implemented as well (Section 8.4.4).

The **fourth goal** was to improve user experience by avoiding waiting times when viewing analysis details. This goal has been partially achieved by the second goal where background data analysis, using processing queue, was implemented. Also we refactored data access layer and replaced server-side static content generation with client-side data visualization using D3 library (Section 4.4 and Section 0). Moreover, we have implemented progress bars and loading status bars for every operation that takes more than 1-2 seconds to respond and present results (Section0). Loading of the rows in the tables is dynamic now: row by row, so that user is able to view partially loaded data (tables shown in sections 8.4.3, 8.4.4 and 0). As shown in the Table 9.1 below (the most critical measurements were done), server traffic size and time for data loading has been improved. This fulfils the goal to show data in less than 1 second.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>v.0.8</th>
<th>v.1.1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph view – download big graph data, size</td>
<td>57.1Mb</td>
<td>3.8 KB</td>
</tr>
<tr>
<td>Stacked bar chart row – display time</td>
<td>200ms</td>
<td>15 ms</td>
</tr>
<tr>
<td>Stacked bar chart row – download data, size</td>
<td>42.4 KB</td>
<td>0 KB</td>
</tr>
<tr>
<td>Analysis results – list loading (10 items), waiting time</td>
<td>1.52 s</td>
<td>273 ms</td>
</tr>
</tbody>
</table>

The **fifth goal** was to improve reporting of the results. This goal has been partially achieved by the fourth goal where client-side data visualization is replaced by interactive JS version. New charts implementation is friendlier and provides content information on demand. In addition, we have implemented a history bar chart to show upload results in comparison with other uploads (Section 0).
The **final goal** was to fix any bugs in the new or existing functionality uncovered during the time of the thesis. We successfully fixed all found bugs internally by ARiSA and fixed reported issues by the customers of QM application in production. Moreover, we updated layout and styles to be compliant with XHTML 1.0 and CSS3 standards.

Thus, all of the goals specified for thesis work have been archived. ARiSA and the customers of QM application are satisfied by maintenance results.

Finally, except archived goals the following **conclusions** have been made:

- Maintenance of existing application is not an easy task for third-party developers involved into product support. It requires a good knowledge of technology stack, ability to learn quickly and some initial project pre-study;
- Cross-platform development requires additional effort while supporting the product;
- Close communication with real or potential customers guarantees that product development goes in a proper way;
- Incremental feature development or working by Scrum methodology is a good choice for applications development; As a result, after each milestone customer receives ready-to-use application with certain subset of features and bug fixes;
- Testing of the medium size and big size projects is mandatory; it requires automated or at least manual regression testing. This guarantees that developed software fulfils Software Quality standards and application requirements;
- Thesis work in cooperation with software companies is more efficient in comparison with standard university thesis topics. The main reason is that students have opportunity to develop commercial projects and receive professional experience. Students start their careers in cooperation with industry-leading companies.
10 Future Challenges

In the future work, there are lots of possible system improvements and functionality extensions. Most of the important features from ‘QM as product’ perspective are described below in the next sub-sections. All features are divided into four categories: performance, design and styling, architecture and functionality extensions.

10.1 Performance

QM is designed to analyze source code and to present analysis results visually for end users, in particular for subsequent human-analysis by end users. In the scope of master thesis all system workload was separated between front-end and back-end. Server-side static content generation was replaced with client-side interactive visualization implemented in pure JavaScript. Such approach increased system interactivity and usability and decreased server load. At the same time such approach requires good algorithms for data preprocessing on the client side. Results of the system testing based on data provided by the first customers and testing on slow client-side machines (with processor Intel Core i3 and older) showed that the system needed additional efforts to optimize visualization of the big amount of data. Especially that refers to the graph with a big number of edges between nodes, as force-weighted graph calculation is slow; it takes some time to render all edges for several calculate iterations. Thus maintenance actions should be planned.

10.2 Design and Styling

Design and styling of the current QM version is very simple, based on standard GWT components library and standard HTML functionality. It is working well for the system in prototyping or alpha stage, though not acceptable for web applications ‘in production’ where end users design and usability are of the same importance as functionality. Moreover, complex applications require good design and system navigation.

As a quick solution, this goal could be achieved, if we built user interface on top of the existing front-end frameworks. For example, nowadays one of the most popular frameworks is Twitter Bootstrap [38] licensed under Apache License v2.0, which provides a free collection of styles, tools, plugins and components for faster and easy web development. For GWT platform GWT-Bootstrap [39] is available. It provides API based on Twitter Bootstrap for GWT.

On the other hand, with growth of the mobile industry, additional use case surfaced. Users involved into software development (Product Owners, Project Managers, Team Leads, etc.) would use QM web application on mobile devices like tablets and tablet PCs. QM was designed mainly for desktop browsers, not taken into account mobile devices. For that reason mobile market should be analyzed, customization of QM should be planned to support more web platforms and increase compatibility.

10.3 Architecture

This section describes QM application feature proposals in scope of architecture improvements and extension. Some of the features below ARiSA have already added to product backlog for implementation.

10.3.1 Plugins and Modules

QM web application allows analysis of certain source code types for definite programming languages. Analysis functionality is represented by backend which consists of third-party
tools and implemented by ARiSA. To incorporate a new analysis back-end tool for a new type of source code, manual changes in QM code are required. That is an additional effort for application maintenance and thus additional costs to support QM. That is why architecture shall be improved by refactoring interface between front-end and back-end to incorporate generic entry point for plugins or modules. Each plugin is a new type of analysis which returns analysis results in the format defined by QM application, which can be automatically saved in the database and visualized by existing front-end functionality.

10.3.2 Customer-Specific Data in Separate Databases
These are two possible types of QM application deployment:

1) Single QM instantiation for several customers (planned to be implemented);
2) QM instance per customer deployment (current version).

The first type of deployment means that analysis results for several customers will be stored in the same database. At the same time for some commercial reasons customer could request to store data in a separate database which is not shared with other customers. This is partially solved by the second type of deployment, but this also leads to a use case where several QM instantiations need to be supported. A possible solution is to build mixed distributed implementation where QM would consist of one web front-end, one master database for system users’ management and separate slave databases for each customer. Each slave database will contain user profiles linked to user accounts in the master database.

10.3.3 Multi-Queue Data Processing
According to the second thesis goal (allow efficient processing of several uploads) processing of uploads is done one by one using processing queue. This completely satisfies current environment established in ARiSA. As soon as the system becomes more complex, the amount of data and the number of customers grows, QM will be redeployed to a more powerful web server. A new server potentially will have a multi-core processor where the number of cores is more than two. It means that in some cases environment usage will be not that efficient: processing can be distributed over several cores, or several queues can be started simultaneously to process data in parallel.

10.4 Functionality Extensions
This section describes several possible functionality extensions of QM application.

10.4.1 Analysis Report and Real-Time Processing Status
According to the third thesis goal (inform users about progress and aid debugging) QM was extended with progress dialogs and email notifications about current analysis status. Possibility to download logs via web interface is implemented as well. In future QM could be extended with more detailed information about analysis status showed on the Analysis Report page, where user is be able to view a table with analysis steps and messages (of information, warning or error types) related to them. Depending on the implementation some information can be collected and attached to the email notification in PDF or other suitable document format.

10.4.2 Alternative Analysis Results Visualization
QM Application can be extended with more charts and alternative data representations. This will improve user experience and provide additional means for quality analysis.
10.4.3 Data Export
From customer point of view QM application is a system-as-service provided by external company, ARiSA. For some customers it is not enough to make analysis and review analysis using a third-party system. They want to receive analysed data for reporting and a future post-analysis depending on their internal needs.

To satisfy customer needs, QM application should be extended with API for accessing data in a raw or aggregated form. An obvious solution is to develop web services for retrieving data in different formats, e.g. XML, JSON or CSV. As an additional option, we could customize widgets which can be used by a customer for integration of QM data into their own applications.

This feature was requested by the customers during the planning meeting in ARiSA.

10.4.4 Automatic Data Import Configuration
According to the first thesis goal (allow uploading to QM from build scripts) QM application was extended with the possibility to upload data to the import directory using SFTP protocol. To grant access to the server for external company, QM server administrators should create a system user and manually configure access according to the instructions (SFTP directory, login access key, etc.). See Appendix C for more details. In future, the configuration procedure could be automated.

10.4.5 Projects and Project Families Management
According to the proposed deployment type, Single QM instantiation for several customers (Section 10.3.2), QM will be able to store uploads for several customers. Taking into account this feature, upload groups should be implemented. The most common group types according to software development processes are projects and project families.

10.4.6 Company Teams Management
In future when QM application will be able to handle data for analysis from several customers, team management should be implemented. QM application will be used by different customer’s company departments, for example, Development and Quality Audit departments.

10.4.7 System Monitoring
With the system growth, effective tools for support and system monitoring is a must have in the modern information systems. QM is not an exception and it will also need automated tools to ensure that server’s and system’s health is Ok and all subsystems are working as expected without errors. Partially that should be solved on the implementation level. For example, currently QM is not handling errors which are related to the file system, available RAM on the server, etc. The system simply writes errors to the event log without any notification to the end users via web interface or by email to the system administrators.
References


Appendix A: Functional Requirements

This section provides a complete overview of function requirements.

a. Security

This section provides security-related functional requirements.

UC1: User login via web interface

<table>
<thead>
<tr>
<th>Use case description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary actors: All users</td>
</tr>
<tr>
<td>Preconditions:</td>
</tr>
<tr>
<td>1. The user is registered and active in the system.</td>
</tr>
<tr>
<td>2. The user is logged out of the system.</td>
</tr>
<tr>
<td>Triggers:</td>
</tr>
<tr>
<td>1. The user navigates to the web application in web browser.</td>
</tr>
<tr>
<td>2. The system indicates that user shall be authenticated in the system.</td>
</tr>
<tr>
<td>Basic flow:</td>
</tr>
<tr>
<td>1. User provides authentication details (user name and password).</td>
</tr>
<tr>
<td>2. Authentication succeeded. (E1-E2)</td>
</tr>
<tr>
<td>Sub-flows:</td>
</tr>
<tr>
<td>Alternative flows:</td>
</tr>
<tr>
<td>E1: Invalid user credentials:</td>
</tr>
<tr>
<td>- Authentication failed due to invalid user credentials;</td>
</tr>
<tr>
<td>- Error message is shown.</td>
</tr>
<tr>
<td>E2: The user is disabled:</td>
</tr>
<tr>
<td>- Authentication is failed because the user was disabled.</td>
</tr>
<tr>
<td>- Error message is shown.</td>
</tr>
<tr>
<td>Post-conditions:</td>
</tr>
<tr>
<td>1. The user is redirected to requested page.</td>
</tr>
<tr>
<td>2. Login panel displays currently logged in user (user name and a complete name).</td>
</tr>
</tbody>
</table>

Table A.2 UC1 Functional requirements

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNR1: Shall be possible to log into the system using password.</td>
</tr>
</tbody>
</table>

UC2: User logout via web interface

<table>
<thead>
<tr>
<th>Use case description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary actors: All users</td>
</tr>
<tr>
<td>Preconditions:</td>
</tr>
<tr>
<td>1. The user is logged into the system.</td>
</tr>
<tr>
<td>Triggers:</td>
</tr>
<tr>
<td>1. The user makes logout from current session using login widgets.</td>
</tr>
<tr>
<td>Basic flow:</td>
</tr>
<tr>
<td>Sub-flows:</td>
</tr>
<tr>
<td>Alternative flows:</td>
</tr>
<tr>
<td>Post-conditions:</td>
</tr>
<tr>
<td>1. User session is invalidated.</td>
</tr>
<tr>
<td>2. The user is redirected to the login page.</td>
</tr>
</tbody>
</table>

Table A.4 UC2 Functional requirements

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNR2: Shall be possible to logout the system.</td>
</tr>
<tr>
<td>FNR3: The system should invalidate current session on logout and discard all made changes in the current view.</td>
</tr>
</tbody>
</table>

b. User Profile

This section provides functional requirements related to user profile.
UC3: Change settings in user profile

Table A.5 UC3 Use Case description

<table>
<thead>
<tr>
<th>Primary actors</th>
<th>All users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>1. The user is logged into the system.</td>
</tr>
<tr>
<td>Triggers</td>
<td>1. The user requests to open user profile using navigation widgets.</td>
</tr>
<tr>
<td>Basic flow</td>
<td>-</td>
</tr>
<tr>
<td>Sub-flows</td>
<td>-</td>
</tr>
<tr>
<td>Alternative flows</td>
<td>-</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>1. The user is able to view and edit user profile.</td>
</tr>
</tbody>
</table>

Table A.6 UC3 Functional requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>FNR4: Shall be possible to view and edit user profile.</th>
</tr>
</thead>
</table>

c. User Management

This section provides functional requirements related to users management.

UC4: Manage registered users

Table A.7 UC4 Use Case description

<table>
<thead>
<tr>
<th>Primary actors</th>
<th>Administrators</th>
</tr>
</thead>
</table>
| Preconditions  | 1. The user is logged into the system.  
2. The user has administrative privileges. |
| Triggers       | 1. The user requests to open user management page using navigation widgets. |
| Basic flow     | 1. The user lists all users registered in the system.  
2. The user requests new user to be created.  
a. An empty row in edit mode is shown.  
b. The user fills in information about a new user.  
c. The user is able to change generated temporary password.  
d. The user commits all data related to a new user.  
e. New user appears in the list of registered users.  
f. New user receives email notification with temporary password on specified email address.  
3. The user requests to delete selected user in the list.  
a. The user confirms selected user deletion.  
b. Selected user disappears from the list. |
| Sub-flows      | - |
| Alternative flows | - |
| Post-conditions | 1. All user actions are successfully completed. |

Table A.8 UC4 Functional requirements

| Requirements | FNR5: Shall be possible to view and edit users registered in the system.  
FNR6: Shall be possible to create new users and delete existing.  
FNR7: The system shall generate random temporary password for newly created users. |
|--------------|---------------------------------------------------------------------|

49
FNR8: New users shall receive email notifications with temporary password and information how to change it.

d. Upload Management
This section provides functional requirements related to upload management. Upload is an archive with source code for analysis and context attributes attached to archive.

UC5: Manage data uploads

Table A.9 UC5 Use Case description

<table>
<thead>
<tr>
<th>Primary actors</th>
<th>All users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>1. The user is logged into the system.</td>
</tr>
<tr>
<td>Triggers</td>
<td>1. The user requests to open uploads management page using navigation widgets.</td>
</tr>
<tr>
<td>Basic flow</td>
<td>1. The user lists all uploads registered in the system.</td>
</tr>
<tr>
<td></td>
<td>2. The user makes a new upload manually via web interface. (S1)</td>
</tr>
<tr>
<td></td>
<td>3. The system refreshes a list of uploads to show a new upload.</td>
</tr>
<tr>
<td></td>
<td>4. The system processes a new upload in the background. (S2)</td>
</tr>
<tr>
<td></td>
<td>5. The user requests to delete a selected upload in the list.</td>
</tr>
<tr>
<td></td>
<td>a. The user confirms an upload deletion.</td>
</tr>
<tr>
<td></td>
<td>b. A selected upload disappears from the list.</td>
</tr>
<tr>
<td></td>
<td>6. The user requests to view analysis results for a selected upload.</td>
</tr>
<tr>
<td></td>
<td>a. The user is redirected to a page with analysis results.</td>
</tr>
</tbody>
</table>

| Sub-flows      | S1. The user makes a new upload: |
|                | a. The user fills in upload-related information. |
|                | b. The user selects file archive to upload. |
|                | c. The user initiates data uploading. |
|                | i. The system starts uploading data chunk (a fragment of data in terms of software) by chunk. |
|                | ii. The system shows upload progress percentage. |
|                | iii. The system informs the user about data uploading completion. (E2) |
|                | iv. All subscribed users receive an email that a new upload is scheduled for analysis. |
|                | S2. Analysis of new upload asynchronously: (E1) |
|                | a. All subscribed users receive an email about current analysis status. |
|                | b. The user is able to see a current processing status on UI. |
|                | c. The user is not able to navigate to analysis results or delete an upload which has analysis in progress. |
|                | d. The system reports about analysis completion. |
|                | a. Upload status is changed. |
|                | b. All subscribed users receive an email with current status and information how to view analysis results. |

| Alternative flows | E1. Concurrent uploads: |
|                  | a. Analysis of some upload is already in progress. |
|                  | b. The system postpones new upload analysis and keeps new upload in pending state. |
|                  | E2. Upload failure: |
### UC6: Remote data uploading via import directory

**Table A.11 UC6 Use Case description**

<table>
<thead>
<tr>
<th><strong>Primary actors</strong></th>
<th>Build Manager</th>
</tr>
</thead>
</table>
| **Preconditions**  | 1. The user is logged into the system.  
                          2. The system is configured for remote data uploading using the import directory. |
| **Triggers**       | 1. The user initiates remote data uploading. |
| **Basic flow**     | 1. The user creates an archive with data intended for analysis.  
                          2. The user includes context data related to the upload in a format defined by the data uploading procedure (Section 7.4).  
                          3. The user uploads archive and context information to the server.  
                          4. The system informs the user about data uploading status. (E1-E3)  
                          5. The user logs via web interface into the system.  
                          6. System shows a recently imported data upload to the system. |
| **Sub-flows**      | - |
| **Alternative flows** | E1. Data uploading to a different remote directory:  
                          a. The system cannot save data due to insufficient user privileges.  
                          b. The system informs the user that no access is granted to the specified directory.  
                          E2. Executing remote commands not allowed by the system:  
                          a. The user executes a command that is not allowed by the remote data uploading procedure.  
                          b. The system rejects remote command.  
                          E3. Upload failure:  
                          a. The system cannot save data due to internet connectivity or other upload errors. |
b. The system informs the user about data uploading failure.

<table>
<thead>
<tr>
<th>Post-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The system successfully imports remotely uploaded data.</td>
</tr>
<tr>
<td>2. The system starts data analysis in the same way as it does for manual uploads using web interface.</td>
</tr>
</tbody>
</table>

Table A.12 UC6 Functional requirements

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNR16: The system shall authenticate user for remote uploads.</td>
</tr>
<tr>
<td>FNR17: The user shall be able to upload data for analysis to the remote import directory.</td>
</tr>
<tr>
<td>FNR18: The system shall inform the user about uploading status (failure, success, etc.)</td>
</tr>
<tr>
<td>FNR19: The system shall validate all user actions executed remotely (invalid remote directory, invalid commands, etc.)</td>
</tr>
</tbody>
</table>

**e. Analysis Results Review**

This section provides functional requirements related to analysis review.

**UC7: Review analysis results**

**Table A.13 UC7 Use Case description**

<table>
<thead>
<tr>
<th>Primary actors</th>
<th>All users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>1. The user is logged into the system.</td>
</tr>
<tr>
<td>Triggers</td>
<td>1. The user requests to open page with analysis over all uploads using navigation widgets.</td>
</tr>
</tbody>
</table>
| Basic flow     | 1. The user lists all **analyzed** uploads registered in the system.  
|                | a. The user sorts listed uploads by name, version or software metrics value. |
|                | b. The system displays data according to provided parameters: sorting and filtering. |
|                | c. The system displays metric charts. |
|                | d. The system display history bar chart with last *n* uploads.  
|                | i. The user is able to view a history bar chart in two modes: clustered and stacked column view. |
|                | e. The user is able to select metrics to display and update factors of metrics. |
|                | 2. The user selects an upload and requests analysis results for selected upload.  
|                | a. The system lists files that belong to selected uploads. |
|                | b. The system shows upload-related attributes (upload date, version, description, archive file name, archive size and numbers of files in archive). |
|                | c. The user is able to group all files by projects inside upload. |
|                | d. The system displays metric charts for each file and projects in upload. (S1)  
|                | e. The user is able to select metrics to display and update factors of metrics. |
|                | 3. The user requests to show architecture analysis details for certain upload project. |
| Sub-flows | S1. The system displays metric charts:  
|           | a. The system displays charts with selected software metrics for each file in a selected upload;  
|           | b. The system displays absolute value for calculated metrics.  
| Alternative flows | -  
| Post-conditions | 1. All user actions are successfully completed.  

| Requirements | FNR20: The user shall be able view analysis results over all uploads.  
|             | FNR21: The system shall show only **analyzed** uploads.  
|             | FNR22: The user shall be able to view analysis results for certain upload in two modes: all files and files filtered by projects.  
|             | FNR23: The system shall show upload-related attributes (upload date, version, description, archive file name, archive size and a number of files in archive).  
|             | FNR24: The user shall be able to sort uploads by name, version and software metrics.  
|             | FNR25: The user shall be able to view metrics charts.  
|             | FNR26: The user shall be able to select display metrics and update factors of metrics.  
|             | FNR27: The user shall be able to view absolute values for each chart.  
|             | FNR28: The user shall be able to view history bar chart for currently loaded uploads.  
|             | FNR29: The user shall be able view a history bar chart in two modes: clustered and stacked column view.  
|             | FNR30: The user shall be able to view architecture analysis of projects by means of graph.  
|             | FNR31: The user shall be able to filter graph nodes (common and context units).  
|             | FNR32: The user shall be able to highlight certain nodes + connected nodes.  
|             | FNR33: The user shall be able to view node attributes on node highlighting.  
|             | FNR34: Shall be able to distinct different types of graph nodes.  

Table A.14 UC7 Functional requirements
Appendix B: Quality Monitor Deployment

This section describes Quality Monitor application deployment. In scope of maintenance Quality Monitor project were migrated to Apache Maven, “…software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation from a central piece of information…” [44]

Quality Monitor deployment is partially automated. Most of the work for automated deployment is done by Maven. By means of Maven building of the package archives .jar (service application) or .war (web application) with all dependencies and configuration files is one command with a few parameters.

Quality Monitor installation or deployment represented by two phases:
- Initial Deployment and Configuration;
- Maintenance Deployment;

Quality Monitor application has the following Maven profiles:
- test-unix – test environment with Linux installed;
- demo-unix – test environment with Linux installed;
- prod-unix – production environment with Linux installed;
- win-dev – development environment with Windows installed;

a. Initial Deployment and Configuration

Initial deployment (environment setup and the first deployment) includes several steps:
1) Install and configure Ubuntu Linux Server;
2) Create maintenance user and disable authentication by password; use authentication by private RSA key instead;
3) Install MySQL database server and create a database account ‘qmonitor’, use settings from configuration files; required credentials should be changed and all configuration files should be updated;
4) Install Apache Tomcat 7 web server into /opt/tomcat7/apache-tomcat-#.#.## where #.## is software version; create symbolic link named current for the directory with tomcat. A directory structure should look as shown below:

```
opt
-- tomcat7
---- apache-tomcat-7.0.37
-------- ...
---- current
-------- ...
```

5) Configure application workspace. Ensure that the directory owner and the group are set to tomcat, a default user under which tomcat web server works. The directory structure should look as shown below:

```
opt
-- tomcat7
---- current
------- webapps
```
6) Continue with steps described in the next section.

b. Maintenance Deployment

Maintenance deployment does not require actions 1-5 described in the section above. See corresponding instructions for redeployment in the sections below.

c. Service Deployment

Service deployment includes several steps:

1) Build package for ‘qmonitor.daemon’ project:

```
 cd $proj-dir # --- Navigate to project directory
 mvn -Denv=prod-unix package
```

2) Copy compiled package to the server into /opt/ directory and unzip the package. The directory structure should look as shown below:

```
 opt
   -- qmonitor
     ---- daemon
     ------ conf
     ------ ...
     ------ lib
     ------ ...
     ------ logs
     ------ jsvc
     ------ qmonitor.daemon.jar
     ------ qmonitor.daemon
     ------ qmonitor.daemon.sh
```

3) qmonitor-daemon.sh requires ACL to be installed:

```
 apt-get install acl
```

4) Run installation shell script or continue with manual configuration described in Appendix B, Section a.
5) Ensure that service is up and running:

```
sh -e qmonitor-daemon.sh install
```

```
service qmonitor-daemon status
```

Quality Monitor Service uses Jsvc [45] to run java application as daemon on the UNIX servers. Current package of QM Service includes precompiled jsvc. In case of incompatibility with the operation system version, navigate to the official site and download sources and then recompile on the target machine.

**d. Web Application Deployment**

Web application deployment includes several steps:

1) Review `{project-dir}/pom.xml` configuration settings, deployment servers, etc.

2) Deploy web project using Maven commands or navigate to `{project-dir}/launch` and run Eclipse .launch file for desired build configuration.

3) Make SFTP configuration for remote data uploading. See configuration instructions in Appendix B, Section b.

4) Perform acceptance testing.
Appendix C: Quality Monitor Configuration

a. Service Configuration
This section describes how to configure service daemon as part of QM application.

i. Stop and Uninstall Existing Service
Service can be already installed in the operation system, so it should be uninstalled first:

```
sh /etc/init.d/$INIT_FILE stop
sleep 1
update-rc.d -f $INIT_FILE remove
rm -f "/etc/init.d/$INIT_FILE"
```

Where $INIT_FILE is the name of service file.

ii. Prepare Service Workspace
Workspace represented by the installation directory and the directory with data needs to be updated to have correct access permissions:

```
# --- Directory for logs
mkdir -p $HOME_DIR/logs
chown -R $USERNAME $HOME_DIR/logs
chgrp -Rv $GROUP $HOME_DIR/logs

# --- Ensure that $USERNAME is owner of the $HOME_DIR
chown -R $USERNAME $HOME_DIR

# --- Ensure that $HOME is group-owned and group-writable
chgrp -R $GROUP $HOME_DIR
chmod -R u+rwx,g+rwx,o+rx $HOME_DIR

# --- Ensure that jsvc is executable
chmod +x $HOME_DIR/jsvc

# -- Ensure that $DATA_DIR
mkdir -p $DATA_DIR

# --- Ensure that $DATA_DIR is group-writable
chgrp -R $GROUP $DATA_DIR
chmod -R u+rwx,g+rwx,o+rx $DATA_DIR
setfacl -Rm d::u::rwx,d:g::rwx,d:o::rx $DATA_DIR

# -- Ensure that workspace is group-writable
chgrp -R $USERNAME $IMPORT_DIR
chmod -R u+rwx,g+rwx,o+rw $IMPORT_DIR
setfacl -Rm d::u::rwx,d:g::rwx,d:o::rw $IMPORT_DIR
```
Where $USERNAME$ and $GROUP$ – by default is ‘tomcat’, user under which Tomcat server is running; $HOME_DIR$ – a current directory where QM service is installed; $DATA_DIR$ – a directory where all static content is stored; $IMPORT_DIR$ – the import directory inside $DATA_DIR$.

ACL – is external command that can be installed on Linux, using the following command:

```
apt-get install acl
```

### iii. Install New Service

Install startup script and make actual start of the service:

```
cp -p $HOME_DIR/$INIT_FILE /etc/init.d/$INIT_FILE
update-rc.d $INIT_FILE defaults
sh /etc/init.d/$INIT_FILE start
```

### iv. Acceptance Testing

This section is intended to ensure that configuration is done successfully:
- Make new data upload via SFTP directory or using web interface;
- Check that the system picked up the data and started analysis.

### b. SFTP Directory Configuration

This section describes how to configure QM application for remote data uploading using SSH and SFTP secure protocols. Target operation system is Ubuntu Linux (as production environment specified by the requirement Error! Reference source not found.). Configuration consists of several steps which are described below.

#### i. User System Account

The user should be able to login to the system and access a file system for data remote data uploading. This requires a few actions/commands to be executed:

1) Create a user in the operation system:

```
useradd -m {user-name}
```

where {user-name} is a name of a user who will be granted to access the server remotely. This user will receive ‘Build Manager’ role in QM application.

2) Set some randomly-generated password, so that the user is allowed to login (a requirement of the most operation systems – a user is not allowed to login with empty password); It does not matter which password is set, because later access would be granted using public key only:

```
passwd {user-name}
```

3) Generate public and private keys, install the public key into a user home directory to enable authentication by key for user:
mkdir /home/{user-name}/.ssh/
ssh-keygen -C "{user-name}@qm.arisa.se"
  # directory: /home/{user-name}/.ssh/rsa_key
  cat /home/{user-name}/.ssh/rsa_key.pub >> /home/{user-name}/.ssh/authorized_keys2

4) Update .ssh directory permissions. SSH requires that home directory shall be writable only by the user, ~/.ssh permissions shall be 700, and authorized_keys - shall be 600:

  chown -R {user-name} /home/{user-name}/.ssh/
  chgrp -R {user-name} /home/{user-name}/.ssh/
  chmod g-w /home/{user-name}/
  chmod 700 /home/{user-name}/.ssh
  chmod 600 /home/{user-name}/.ssh/authorized_keys2

ii. SFTP Subsystem
Access to the file system shall be restricted, so user is able to read and write only to the allowed directory. This requires a few actions/commands to be executed (shall be done only once during installation/first configuration):

1) Restrict a user account and assign common group:

  groupadd sftp-internal
  usermod -g sftp-internal {user-name}
  usermod -s /bin/false {user-name}

2) Configure OpenSSH to use internal SFTP subsystem (/etc/ssh/sshd_config):

  # Change sftp subsystem in the /etc/ssh/sshd_config:
  Subsystem sftp internal-sftp
  # Add the following in the end of /etc/ssh/sshd_config:
  Match Group sftp-internal
  ChrootDirectory %h
  ForceCommand internal-sftp
  X11Forwarding no
  AllowTcpForwarding no

3) Set umask for chrooted sftp users (/etc/pam.d/sshd):

  # Add the following line:
  session optional pam_umask.so umask=0002

4) Set umask for chrooted sftp users (/etc/profile):

  # Add the following line:
  umask 022

5) Reload SSH service to apply all made changes:
iii. Import Directory
This section describes how provide access for QM application to data uploaded by users via SFTP protocol. This requires a few actions COMMANDS to be executed:

1) Create a user import directory and update permissions:

```bash
mkdir /home/{user-name}/_qm_import
chown -R {user-name} /home/{user-name}/_qm_import
chgrp -R tomcat /home/{user-name}/_qm_import
chmod -R g+rw /home/{user-name}/_qm_import
chmod -R g+s /home/{user-name}/_qm_import
```

2) Create symbol link in a user import directory in QM workspace:

```bash
cd /opt/tomcat7/current/webapps/qmonitor-data/uploads/_import
ln -s /home/{user-name}/_qm_import / {user-name}
```

QM workspace import directory contains another directories named by user name or user id. These users shall be registered and active in QM application, otherwise they are not able to make data import.

iv. Acceptance Testing
This section is intended to ensure that configuration is done successfully:

1) Login to the system with generated public key and user name;
2) Upload temporary file to a user import directory;
3) Try to execute any SFTP command, for example – rename uploaded file.