Kreogist Mail
Development Documentation

Software Quality Assurance Plans

February 19, 2016

Reference Number: KMKOT04.
This document is capable with IEEE Std 730-2002.

First Edition (Jan 2016)

This edition applies to Version 0.1 of Kreogist Mail.

Kreogist Dev Team

Kreogist Mail
Development Documentation
Software Quality Assurance Plans

February 19, 2016

KMKOT04

Revision History

Revision	**Version**	**Description**	**Date**
KMKOT04	-001	Initial commit	Jan. 21th, 2016
KMKOT04	-002	Internally accepted	Jan. 23th, 2016
KMKOT04	-003	Revision for correcting grammar	Jan. 25th, 2016
KMKOT04	-004	Correct grammar error by input mistakes	Feb. 19th, 2016

Preface

This document is an update to the specifications contained in the "Affected Documents" table below. This document is a part of product (project) Kreogist Mail.

This document may also contain information that was not previously published.

Affected Documents

Document Title	Document Number
Kreogist Mail Software Verification and Validation Plan	KMKOT05

Related Documents

Document Title	Document Number
Kreogist Mail Software Requirement Specification	KMKOT01
Kreogist Mail Software Project Management Plans	KMKOT02
Kreogist Mail Software Design Specification	KMKOT03

Contents

1 Introduction

This document describes procedures and control methods to obtain desired quality level of end products and process by which these end products are created. This document serves as a guide for managers and developers of Kreogist Mail project (the project). All team members must read this document and apply procedures stated in it. Document applies to all phases of software development as defined in Software Project Management Plan. Detailed information about the software quality assurance activities for these phases will be added in appendices during the project.

All activities directly related to purpose are considered to be in scope. All activities not directly related to purposes are considered to be out of scope. For example, network availability is not within the scope of this project.

2 Reference documents

[1]. Kreogist Dev Team. "Kreogist Mail Software Requirements Specification", January, 2016.

[2]. Kreogist Dev Team. "Kreogist Mail Software Project Management Plans", January, 2016.

[3]. Free Software Foundation, "GNU Free Documentation License", See http://www.gnu.org/licenses/fdl.html (last checked January 13th, 2016), November 3, 2008.

[4]. Wikipedia, "Qt (software) - Wikipedia, the free encyclopedia", See https://en.wikipedia.org/wiki/Qt_%28software%29)last checked January 16th, 2016), January 4, 2016.

[5]. IEEE Software Engineering Standards Committee. "IEEE Std 1063-2001, IEEE Standard for Software User Documentation", December 20, 2001.

[6]. IEEE Software Engineering Standards Committee. "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.

[7]. IEEE Software Engineering Standards Committee. "IEEE Std 1016-1998, IEEE Standard for Information Technology–Systems Design–Software Design Descriptions", July 20, 2009.

[8]. IEEE Software Engineering Standards Committee. "IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans", December 22, 1998.

[9]. IEEE Software Engineering Standards Committee. "IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes", 1997.

[10]. IEEE Software Engineering Standards Committee. "IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation", 1998.

[11]. IEEE Software Engineering Standards Committee. "IEEE Std 828-2005, IEEE Standard for Software Configuration Management Plans", August 12, 2005.

3 Management

This chapter details the structure and tasks of Software Quality Assurance team.

3.1 Organization

For the organization within the project and responsibilities of individual members of teams see Software Project Management Plan. Quality Assurance Manager leads Software Quality Assurance team. He is assisted by team leader. Furthermore Quality Assurance Manager is responsible for Software Quality Assurance and performance of Software Quality Assurance team. When software quality is endangered Quality Assurance Manager will contact team leader. They will also decide whether or not one or more of the following parties have to be informed: customers and project manager.

3.2 Tasks

Main task of Software Quality Assurance team is to check whether procedures are followed properly and that standards are handled correctly as defined in Software Quality Assurance and Software Verification and Validation Plans. Additionally Software Quality Assurance team inspects whether all group members fulfill their tasks as defined in Software Project Management Plan according to the parts of Software Quality Assurance applying to their specific tasks. If a problem is detected, appropriate procedure as defined in section 8, i.e. *Problem reporting and corrective action* will be followed. Besides described main task, Software Quality Assurance team has the following additional tasks:

1. Software Quality Assurance team has to check the consistency and coherence between documents.
2. Organize internal reviews.
3. Attend internal reviews.
4. Check whether all group members take their responsibilities as defined in Software Project Management Plan by talking to team members.

Specific tasks arising during different phases of the project will be added in the corresponding appendices.

3.3 Responsibilities

Main responsibility for Software Quality Assurance tasks, as described in section 3.2, i.e. *Tasks*, lies with Quality Assurance Manager. Quality Assurance Manager can delegate the tasks within Software Quality Assurance team. Minor problems can be solved by each member of Software Quality Assurance team, whereas major problems are matter of Quality Assurance Manager and are also reported to team leader. Every problem found by a team member has to be reported to Quality Assurance Manager (section 8, i.e. *Problem reporting and corrective action*). Reporting to project members outside Software Quality Assurance team is done by Quality Assurance Manager. In case Quality Assurance Manager will be unavailable for a short period of time, team leader will assume his tasks. If Quality Assurance Manager will be unavailable for a longer period of time, Software Quality Assurance team must be expanded and the tasks reorganized. Project Manager will be responsible for this.

4 Documentation

The documents to be delivered in specific phases of the project are listed and outlined in section *Project Deliverables* of Software Project Management Plan. Document standards are described in section 5, i.e. *Standards, practices, conventions, and metrics.*

Software Quality Assurance team must see to it that the following documents are properly reviewed internally before they are submitted for an external review.

Software Requirement Specification

Software Quality Assurance team checks before the internal review whether the SRS:

1. Contains a general description of the software that has to be developed.
2. Contains requirements on the software to be developed as stated by the customer.
3. Contains constraints on the software to be developed.
4. Contains a priority list of the requirements.

Furthermore it has to be checked that every user requirement complies with the requirements defined in Software Verification and Validation Plans.

Software Project Management Plans

Software Quality Assurance team must check whether the goals of the project are clearly described. A life cycle approach for the project must be defined. Software Quality Assurance team must ensure that Software Project Management Plans is realistic by checking:

1. The assumptions made during the planning of the project (by comparing the actual time spent with the reserved time in the planning).
2. Restrictions with respect to planning (e.g. availability of members).
3. External problems (e.g. room availability).

Software Quality Assurance Plans

With respect to Software Quality Assurance Plans, Software Quality Assurance team checks before the internal review whether the Software Quality Assurance Plans contains:

1. Project standards.
2. Problem reporting procedures.
3. Responsibilities of the project members with respect to quality assurance.

Software Verification and Validation Plans

With respect to Software Verification and Validation Plans, Software Quality Assurance team checks before the internal review whether the Software Verification and Validation Plans contains:

1. Reviewing and audits.
2. Testing.
3. Tracing.

During internal reviews Software Verification and Validation Plans, Software Quality Assurance team checks these documents and in case of problems, author(s) and team leader are informed. After the corrective action has been taken, Software Quality Assurance team reviews the document again.

5 Standards, practices, conventions, and metrics

5.1 Documentation standards

During this project many different documents will be made. Software Quality Assurance team checks that documents adhere to house style. Software Quality Assurance team checks that documents adhere to house style and documents are made following IEEE standard guidelines, this is done during random checks (section 6, i.e. *Review*) held by Software Quality Assurance team. Every document has to be approved by:

- Author(s)
- Team leader
- Quality Assurance Manager

In case that these three people happen to be one and the same, a second member of responsible team has to give his approval as well. Only approved documents affect the project. Documentation standards involve the following:

1. All documents must adhere to IEEE standard;
2. All documents use the template provided by Kreogist Dev Team.
3. Requirements on review and approval as described in section 6, i.e. *Review.*
4. Procedures involving change of documents.

These standards apply to all documents, to electronic versions as well as printed ones. However the layout requirements do not apply to documents other than the project and product documents. All documents are made available through Document repository. In case of unavailability of document repository, Project Manager sees to it that there are three copies available of every document (latest version with the highest status of approval) in groups workspace. Three copies consist of one copy on paper and two digital copies on two different geographical locations.

5.2 Design standards

Design standards in Architectural Design and Detailed Design phase will be defined or referenced in Architectural Design Document respectively Detailed Design Document. Software design paradigm that will be used for Mail is Object Oriented Programming.

5.3 Coding standards

This section show you the formatting rules of editing codes from Kreogist Dev Team.

Some existing code may not follow these conventions, but they must be used for new code. If changing existing code to follow these conventions, it is best to send changes to follow the conventions separately from any other changes to the code.

Indentation

- **4 spaces** are used for indentation.
- Using **spaces** instead of using tabs.

Declaring variables

- Suggest declaring each variable on a separate line.

```
//Wrong
int a=0,b=0,c=0;

//Correct
int a=0;
int b=0;
int c=0;
```

- Avoid meaningless names (e.g. a, rbarr, nughdeget) and abbreviations(e.g. "as-ap").

```
//Correct
int m_timeoutCounter;
```

- Single character variable names are only okay for counters and temporaries, where the purpose of the variable is obvious.

```
//Correct
for(int i=0; i<m_timeoutCounter; ++i);
```

- Wait when declaring a variable until it is needed.
- Variables and functions start with a lower-case letter. Each consecutive word in a variables name starts with an upper-case letter.

```
//Correct
int m_isPressed; //for member variables.
int isPressed;   //for local variables.
```

- Classes always start with two upper-case letters. Public classes start with a K (KNGlobal) followed by an upper case letter. Public functions most often start with a k (kSort).

```
//Correct
class KNGlobal : public QOjbect
{
}
```

- Acronyms are camel-cased (e.g. QXmlStreamReader, not QXMLStreamReader).

Whitespace

- Use blank lines to group statements together where suited.
- Always use only one blank line.

- For pointers or references, always use a single space between the type and * or &, but no space between the * or & and the variable name.

  ```
  //Wrong
  KNGlobal* m_global;

  //Correct
  KNGlobal *m_global;

  //Wrong
  void removeContent(QByteArray& data);

  //Correct
  void removeContent(QByteArray &data);
  ```

- No space after a cast.
- Avoid C-style casts when possible.

  ```
  // Wrong
  char* blockOfMemory = (char* ) malloc(data.size());

  // Correct
  char *blockOfMemory =
          reinterpret_cast<char *>(malloc(data.size()));
  ```

- Do not put multiple statements on one line. By extension, use a new line for the body of a control flow statement.

Parentheses

- Use parentheses to group expressions:

  ```
  // Wrong
  if (a && b || c)

  // Correct
  if ((a && b) || c)

  // Wrong
  a + b & c

  // Correct
  (a + b) & c
  ```

Switch statements

- The case labels are in the same column as the switch.
- Every case must have a break (or return) statement at the end or a comment to indicate that theres intentionally no break, unless another case follows immediately.

```
// Correct
switch (myEnum)
{
case Value1:
    doSomething();
    break;
case Value2:
case Value3:
    doSomethingElse();
    // fall through
default:
    defaultHandling();
    break;
}
```

Jump statements (break, continue, return, and goto)

- Do not put else after jump statements:

```
// Wrong
if (thisOrThat)
    return;
else
    somethingElse();

// Correct
if (thisOrThat)
    return;
somethingElse();
```

Line breaks

- Keep lines shorter than 100 characters; wrap if necessary.
- Operators go at the end of wrapped lines.

```
if (longExpression +
    otherLongExpression +
    otherOtherLongExpression)
```

```
{
}
```

5.4 Comment standards

Comment standards form a part of Coding standards and will thus be described in section 5.3, i.e. *Coding standards.*

5.5 Testing standards

Testing standards to be used are described in Software Verification and Validation Plans.

5.6 Metrics

Members of Software Quality Assurance team will measure the quality of delivered software, during random checks, by means of metrics. Some examples of metrics are:

- Length of function (should not be more than 100 lines).
- Total length of (useful) commentary divided by the total length of code (should exceed 1/5).
- Number of parameters divided by number of functions (should not exceed 5. Functions should have no more than 7 parameters).
- Maximum depth of nested if-statements (should be less than 4).
- Maximum depth of loops (should be less than 3).

Software Quality Assurance team makes a small report containing results of described tests in its log and delivers report to author(s) of checked document. If a violation of these metrics is detected, it has to be resolved, unless team leader and Quality Assurance Manager grant a permit.

5.7 Compliance monitoring

Software Quality Assurance team will monitor compliance to proposed conventions by way of random checking of codes during which references to other documents are checked. During reviews Software Quality Assurance team member present checks with authors of reviewed codes whether it has references in it. He will also check whether authors checked the referenced codes. Discovered problems are reported to team leader.

6 Review

Standards and procedures for Reviews and Audits are described in Software Verification and Validation Plans. In addition to Reviews, Software Quality Assurance team carries out checks as described below.

Persistent checks

All the codes will be checked as soon as it complete. Each function won't be commit before it has been checked completely.

Random checks

Software Quality Assurance team randomly checks all project and product documents to ensure that all products adhere to document standards and that all group members do their job properly. Management and product documents are tested for adherence to IEEE standards and if their layout and style adheres to house style. Furthermore the references and tracing to other documents are investigated. It is observed that program code adheres to the coding standards. Random checks are an addition to reviews. Every document undergoes a random check at least once. To save time, Software Quality Assurance team does not have to write a report. It just reports results to author and team leader. If problems are discovered a date is set when problem must be solved and then document is checked again.

7 Test

Methods and functions for testing are detailed in Software Verification and Validation Plans. In random tests of section 6, i.e. *Review* and in weekly interviews of team leaders, Software Quality Assurance team observes that these functions are followed and that team had their functions tested undertakes possible necessary actions. When it is detected that testing procedures are not followed, Software Quality Assurance team informs team leader.

8 Problem reporting and corrective action

When a problem is detected, it has to be solved. There are several kinds of problems:

Document problems

- Non compliance with other project documents.
- Non compliance with IEEE standards.
- Non compliance with house style.

- Incompleteness.
- Errors.

Code problems

- Lack of functionality.
- Wrong functionality.
- Non compliance with coding or commentary standards.

These are procedures to be followed when a problem is detected:

Problem reporting procedure When a problem is detected, person who discovered the error is responsible for reporting it to team leader and Quality Assurance Manager. When a problem is discovered during a review, a member of Software Quality Assurance team present is responsible.

Problem solving procedure

- Software Quality Assurance team appoints team leader to solve the reported error. Team leader will allocate the problem to one of team members. He is then responsible for solving the problem.
- When the problem is solved Software Quality Assurance team is notified to check whether changes could solve the problem.
- When the problem cannot be solved, or cannot be solved within a reasonable amount of time a meeting is set up with team leader, and Quality Assurance Manager. During this meeting a decision will be made about further dealing with the problem.

If the problem to be solved was discovered after internal or external acceptance, team leader first decides whether the problem is important enough to solve, if so, a Change Request (CR) has to be filled out. CR will be treated as a problem. This CR has to be approved by:

- In case of previous internal acceptance: team leader, author(s) of the document and Quality Assurance Manager.
- In case of previous external acceptance: team leader, author(s) of the document, Quality Assurance Manager, and Project Manager.

Change in requirements of the customer

It is also possible that requirements of customer change. In this case, requested change is matched to SRS. If the change conforms to SRS, it is accepted. If it does not conform to SRS, the team decides wether it will discard the changes or not.

9 Tools, techniques, and methodologies

Software Quality Assurance team has to make sure that appropriate tools, techniques and methods are used. These are described in Software Project Management Plan and Software Verification and Validation Plans. Software Quality Assurance team checks their use by means of random checks.

With respect to tool used during this project, special interest is paid to:

- Availability of tools.
- Knowledge. Group members working with the tools must have necessary skills to work with the tools.
- Tools must work properly.

Every used tool will be checked at least once before use and once during use. When problems appear, Software Quality Assurance decides together with team leader if the problem can be solved, or if the tool must be replaced by an alternative.

10 Code control

It is team leader's responsibility to assure correct handling of the code due to the standards described in Software Project Management Plan.

The following has to be valid:

1. Documents are available to all people who are authorized to access them and to no one else.
2. Latest version of a document is always available.
3. No file is unnecessarily locked.
4. Name conventions are consequentially used.

Software Quality Assurance team checks if procedures and standards as described in Software Quality Assurance are handled properly. This is done by reviews and checks defined in section 6, i.e. *Review.* Problems are reported to team leader and project manager.

11 Media control

Software Quality Assurance team checks if procedures and standards as described in Software Quality Assurance are handled properly. This is done in reviews and checks defined in section 6, i.e. *Review.* Problems are reported to team leader and project manager.

12 Supplier control

All external software components in codes, that have an unreliable source, will be tested according to IEEE standards. Software components that have reliable sources will undergo some quick tests. These tests will be focused on the parts of this software that are of importance to the project. Whether an external software component is reliable or not is to be decided by programmer that wants to use the specific component and the team leader who is from the team that wants to use the component.

Software tools which will be used for development of the source code are available to all project members.

13 Records collection, maintenance and retention

Minutes of meetings and notes of external reviews are added to the project library as described in Software Project Management Plan. Minutes of meetings are added after members of meeting have approved them. Minutes are delivered 3 workdays after the meeting at the latest. These documents will be kept throughout the duration of this project. Notes of reviews are reworked into a new version of the document.

14 Training

During the project there will arise tasks that require special skills. Due to the fact that all group members reached an acceptable level of knowledge in the area of computer science, special training in this area will probably be unnecessary. However, each member of the project should make sure he has sufficient skill in of the following:

1. Developing languages: C++.
2. Version control tools: git.
3. Framework: Qt.

If tasks arise that require special skills, team leader and project manager will assess the level of knowledge for the task in group and then they decide if special actions have to be taken. In that case, detailed information will be added in the appendices of this document.

15 Risk management

In Software Project Management Plan, risks of the project are described. During progress meetings the occurrence of any of risks described must be discussed and project manager must see to it that the necessary course of action is taken. Quality Assurance Manager will assist him in this task.