

BERN UNIVERSITY OF APPLIED SCIENCES

PROJECT 1 — CLOCKALARM

Requirements (version 0.1.0)

Loïc Charrière, Samuel Gauthier

supervised by
Claude Fuhrer

March 20, 2017

Contents

1	Project Vision	3
2	Project Goals	4
2.1	Goals	4
2.2	Requirements	4
3	System and Context Boundaries	5
3.1	Stakeholder	5
3.2	Actors	5
3.3	System context	6
3.4	System boundary	6
3.5	Context boundary	6
4	System diagrams	7
4.1	Domain model	7
5	System description	8
5.1	User Stories	8
5.1.1	As a User	8
5.2	Use Cases	9
5.2.1	Import Configuration	9
5.2.2	Export Configuration	9
5.2.3	Edit Default Configuration	10
5.2.4	Launch ClockAlarm manager (no login)	11
5.2.5	Launch ClockAlarm manager (with login)	11
5.2.6	Setup a new alert: Simple alert	12
5.2.7	Setup a new alert: Periodic alert	13
5.2.8	Setup a new alert: E-mail sender	14
5.2.9	Edit an alert: Simple and Periodic alert	15
5.2.10	Edit an alert: E-mail sender	16
5.2.11	Import Alerts	16
5.2.12	Export Alerts	17
5.2.13	Snooze Alert	18
6	Development decisions	19
6.1	Word Processor	19
6.2	Version Control	19
6.3	Python	20
6.4	Unit Testing	20
6.5	Test Coverage	20
7	Minutes	21
7.1	Meeting 1: March 1, 2017	21
7.2	Meeting 2: March 17, 2017	22

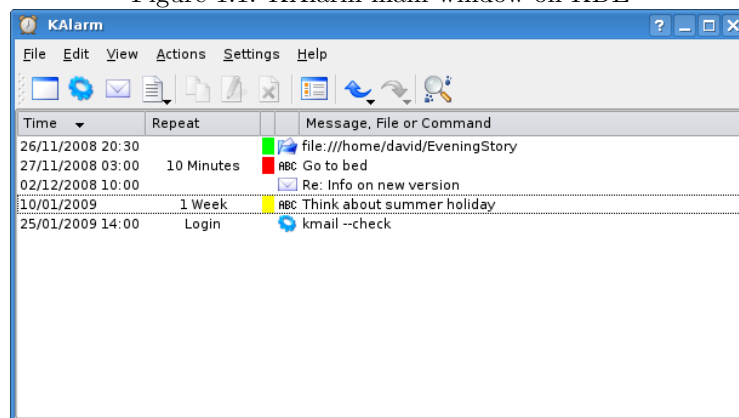
Glossary	24
Bibliography	25

Chapter 1

Project Vision

The ClockAlarm project is an application allowing the user to manage alarms which will alert him at specified times. The user can create new alarms and assign them to categories in order to stay organised. The project is thought to replace in the long run an existing application named KAlarm.

Figure 1.1: KAlarm main window on KDE



Chapter 2

Project Goals

2.1 Goals

- Improve the overall life organisation of the user
- No more missed events

2.2 Requirements

- The application has to be cross platform (Windows, Linux, macOS)
- The database must not be stored in a binary format
- The configuration has to be easily transferable to another computer
- Customisable alerts
- Classifiable alerts
- Recurrent alerts
- Alerts can be scheduled
- Delay alerts
- Snooze alerts

Chapter 3

System and Context Boundaries

3.1 Stakeholder

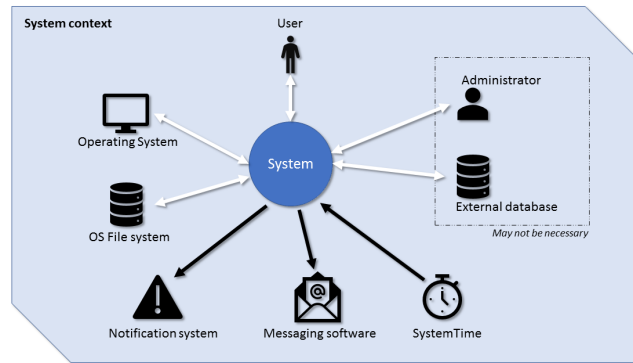
1. kAlarm users
Users who previously used kAlarm on linux, and wishing to find a similar but remastered and cross-platform software.
2. Computer users
Those who tend to forget many things.
3. Investors
Investors in the project. For example advertisers looking for visibility in the software.
4. The GNU Project
The GNU Project collective could be pleased to see that an updated version of kAlarm is proposed.

3.2 Actors

1. Software User (Primary Actor)
The peoples who will use the software. They are the first concerned by the product.
2. Administrator (Primary Actor)
Super users with special authorizations. May be able to manipulate the data and the other users.
It is likely that the software does not need an administrator.
3. System Time (Supporting Actor)
The service specific to the OS and providing the exact international time.
4. Messaging Software (Supporting Actor)
Allows ClockAlarm to send Emails.
5. OS Notification Services (Supporting Actor)
OS specific notification center or notification service.

3.3 System context

Figure 3.1: System context



3.4 System boundary

In addition to the elements above, special attention should be given to the following points.

- The must-have goal isn't to add functionality to the existing software kAlarm, but to create a similar and cross-platform product.
- The existence of a database and an administrator could be dropped and go out of context.
- The software isn't meant to be an alternative to an agenda. It behaves like a list of tasks and reminders.

3.5 Context boundary

In addition to the logical considerations and those stated above, the following points should be taken into account.

- The software is responsible for asking the messaging software to send to emails. This means that the way emails are sent isn't part of the environment.
- The software bases its alarms on the internal time. The accuracy of the time and of the time zone with respect to the geographical position of the machine is not part of the environment.

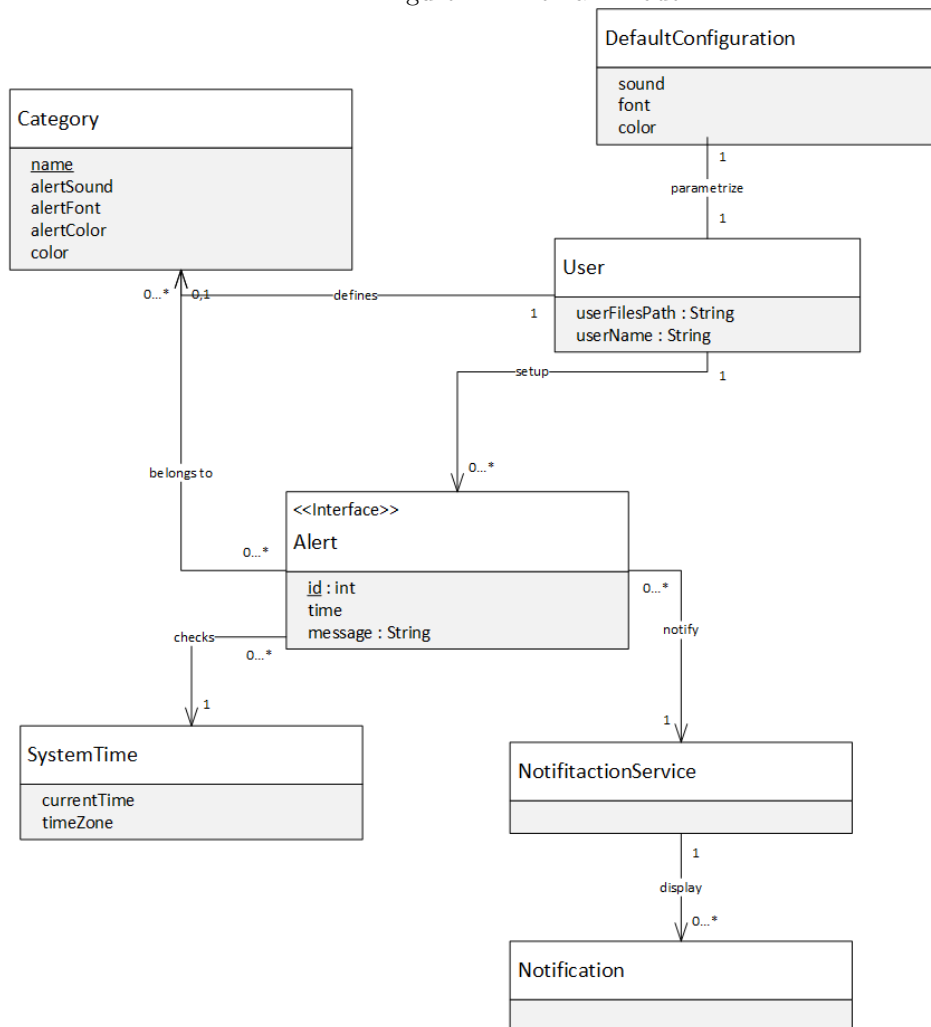
Chapter 4

System diagrams

4.1 Domain model

Conceptual model including system behavior and data.

Figure 4.1: Domain model



Chapter 5

System description

5.1 User Stories

As an [actor] I want [action] so that [achievement].

5.1.1 As a User

Alerts functionality

1. I want to registrate some alerts so that the software warns me when they occure
2. I want to be able to chose a different color, font and sound for every alert
3. I want to edit the existing alerts when needed
4. I want to delete the existing alerts when needed
5. I want to postpone an alarm so that i'm notified again later (snooze)
6. I want to mute an alert so that i'm not notified

Managing alerts

7. I want to create categories of alerts, so that I can sort my alerts
8. I want to edit my categories, in a way my alerts remain categorised
9. I want to delete categories of alerts, so that my list of categories stay concise
10. I want to assign color identity to my different categories, some that I can easily find them
11. I want to personalize the default color, font and sound of my alarms in the software configurations
12. I want to mute an alert catogory so that i'm not notified by any alert in this category

Persistence

13. I want to retrieve the correct state of my alerts after turning my computer down and back on
14. I want to be alerted at any times, as soon as I'm loged on my computer session

Exportability

15. I want to export my alerts so that I can't import them on an other computer with ClockAlarm installed
16. I want to import an alerts file, so that I can retrieve previously exported alerts
17. I want to load configuration files, so that I can use predefined color and sound themes

Pivacy

18. I want my alerts to remain private, so that nobody except me knows about them

5.2 Use Cases

5.2.1 Import Configuration

Import a ClockAlarm application configuration from an external file.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is replaced with the one in the given file.

Main success scenario

1. The User selects a file to be imported.
2. The application checks if the file is a valid configuration file.
3. The application replaces the configuration with the one specified in the file.
4. The application saves the new configuration.
5. The application notifies the User that the configuration has been imported successfully.

Extension

2. The application detects that the file is invalid.
 - i The application notifies the User that the file is invalid.
 - ii The User chooses if he wants to select another file.

5.2.2 Export Configuration

Export a ClockAlarm application configuration to a file.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is exported to a file whose path was chosen by the User.

Main success scenario

1. The User is asked where he wants to save the file.
2. The application exports the configuration to the file.
3. The application notifies the User that the configuration has been exported successfully.

Extension

1. The application detects that it has not enough privileges to write to the specified folder.
 - i The application notifies the User that it has not enough privileges to write to the specified folder.
 - ii The User chooses if he wants to select another folder.

5.2.3 Edit Default Configuration

Edit the default configuration of the ClockAlarm application.

Scope

ClockAlarm application configuration phase.

Primary actor

User

Precondition

The application has to be started.

Postcondition

The application configuration is edited.

Main success scenario

1. The User edits the configuration
2. The application checks if the User entered configuration is valid
3. The application saves the configuration.

Extension

2. The application detects that the User entered configuration is invalid
 - i The application notifies the User that the entered configuration is invalid.
 - ii The User chooses if he wants to edit the configuration.

5.2.4 Launch ClockAlarm manager (no login)

In the case of an application without shared database and user accounts.

Scope

A computer running any common operating system (Windows, Mac OS, Linux).

Primary actor

User or Administrator

Precondition

The computer is on and the user is logged on his computer user session. The ClockAlarm application is correctly installed.

Postcondition

ClockAlarm is running and the manager window is open. The user is logged on his personal ClockAlarm session.

Main success scenario

1. The user launches the ClockAlarm application.
2. The application starts and loads configurations and alarms from the user's personal files folder.
3. The manager window opens and displays the main window.

Extension

- 2 The application is already running in background.
 - i Goto point 3.
- 2 Configurations or alerts can not be loaded (e.g. first use of the program).
 - i The application creates a new default configuration file and an empty alert file.
 - ii Goto point 2.

5.2.5 Launch ClockAlarm manager (with login)

In the case of an application with a shared database and user accounts.

This solution is unlikely to be kept.

For this reason, not all scenarios, especially those related to the connection with the server, will be treated here.

Scope

A computer running any common operating system (Windows, Mac OS, Linux).

Primary actor

User or Administrator

Precondition

The computer is on and the user is logged on his computer user session. The ClockAlarm application is correctly installed.

The user has a registered user account on the server.

Postcondition

ClockAlarm is running and the manager window is open. The user is logged on his personal ClockAlarm session.

Main success scenario

1. The user launches the ClockAlarm application.
2. The application starts.
3. The user is asked to enter his credentials and the program tries to check these.
4. The application is connected to the server.
5. The application loads configurations and alarms from the server database.
6. The manager window opens and displays the main window.

Extension

- 2 The application is already running in background.
 - i Goto point 6.
- 3 The credentials are incorrect.
 - i The connection to the server is denied.
 - ii The user is asked to give his credentials again.
 - iii Goto point 3.
- 3 The user isn't registered.
 - i The connection to the server is denied.
 - ii The user is redirected to a page to an account creation page.
 - iii The user creates a new account on the server.
 - iv Goto point 3.
- 5 Configurations or alerts can not be loaded (e.g. first use of the account).
 - i The application creates and upload a new default configuration file and an empty alert file.
 - ii Goto point 2.

5.2.6 Setup a new alert: Simple alert

A simple alert displays a message to the user at the requested time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

A new simple alert is added to the alert list and is ready to warn the user at the right time.

Main success scenario

1. The user browse the menu and selects “ew alert > simple alert”. The simple alert creation window is displayed.
2. The user sets his alert. The message to be displayed as well as the time of display are mandatory.
The user can, if desired, assign a category to the alert, or set the color, sound and font of the alert.
3. The user validates his alert. It is redirected to the main window.

Extension

- 3 The user does not complete the category and settings.
 - i The default settings are used and the alert does not belong to any category.
- 3 The user complete the category, but not the settings.
 - i The parameters of the category are used.
- 3 One of the parameter is invalid (for example, the time entered is earlier than the current time).
 - i The user is asked to check his entries. Back to point 2.

5.2.7 Setup a new alert: Periodic alert

A periodic alert displays a message to the user at the requested periodic time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

A new periodic alert is added to the alert list and is ready to warn the user whenever it occurs.

Main success scenario

1. The user browse the menu and selects “new alert > periodic alert”. The periodic alert creation window is displayed.
2. The user sets his alert. The message to be displayed, the time of display as well as the periodicity are mandatory.
The user can, if desired, assign a category to the alert, or set the color, sound and font of the alert.
3. The user validates his alert. It is redirected to the main window.

Extension

- 3 The user does not complete the category and settings.
 - i The default settings are used and the alert does not belong to any category.
- 3 The user complete the category, but not the settings.
 - i The parameters of the category are used.
- 3 One of the parameter is invalid (for example, the time entered is earlier than the current time).
 - i The user is asked to check his entries. Back to point 2.

5.2.8 Setup a new alert: E-mail sender

An e-mail sender send an e-mail at the requested time.

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

Postcondition

An e-mail is configured and ready to be sent at the scheduled time.

Main success scenario

1. The user browse the menu and selects “new alert > e-mail sender”. The e-mail sender creation window is displayed.
2. The user sets his e-mail. He enters the recipient, subject, and body of the message. He also sets the time of sending.
The user can, if he wants to, enter the path to an attachment.
3. The user validates his e-mail. It is redirected to the main window.

Extension

- 3 One of the parameter is invalid (time entered earlier than current time, invalid recipient e-mail address, empty object).
 - i The user is asked to check his entries. Back to point 2.

5.2.9 Edit an alert: Simple and Periodic alert**Scope**

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

A simple(resp. periodic) alert is already set.

Postcondition

The selected alert is updated and ready to alert the user at the chosen time.

Main success scenario

1. The user selects the alert he wishes to modify.
2. The user browse the menu and selects"file >edit alert". The simple (resp. periodic) alert edition window is displayed.
3. The current settings are displayed. The user edits his alert. The message to be displayed as well as the time of display (and the periodicity for the periodic alert) are mandatory.
The user can, if desired, assign or reassign a category, color, sound and font to the alert.
4. The user validates his modifications. It is redirected to the main window.

Extension

- 2 No alert is selected.
 - i Nothing happens.
- 3 The user changes the alert category.
 - i The user is asked if he also wants to use the parameters of the category.
- 4 The user does not complete the category and settings.
 - i The default settings are used and the alert does not belong to any category.
- 4 The user complete the category, but not the settings.
 - i The parameters of the category are used.
- 4 One of the parameter is invalid (for example, the time entered is earlier than the current time).
 - i The user is asked to check his entries. Back to point 3.

5.2.10 Edit an alert: E-mail sender

Scope

The ClockAlarm manager window.

Primary actor

User

Precondition

ClockAlarm is running. The configurations and existing alerts are loaded. The user is on the main window.

An e-mail sender is already set.

Postcondition

The selected e-mail sender is updated and ready to send an e-mail at the scheduled time.

Main success scenario

1. The user selects the e-mail sender he wishes to modify.
2. The user browse the menu and selects "file >edit alert". The e-mail sender edition window is displayed.
3. The current settings are displayed. The user edits his e-mail. He can edit the recipient, subject, and body of the message. He can also edit the time of sending and the path to an attachment.
4. The user validates his modifications. It is redirected to the main window.

Extension

- 4 One of the parameters is invalid (time entered earlier than current time, invalid recipient e-mail address, empty object).
 - i The user is asked to check his entries. Back to point 3.

5.2.11 Import Alerts

Load ClockAlarm alerts from an external file.

Scope

ClockAlarm application user interface.

Primary actor

User

Precondition

The application has to be started.

Postcondition

New alerts from the given file are added to the alert list.

Main success scenario

1. The User selects a file to import.
2. The application checks if the file is a valid alert file.
3. The application asks the User which alerts he wants to import.
4. The application adds the alerts to the User alert list.
5. The application saves the alert list.
6. The application notifies the user that the alerts have been imported successfully.

Extension

2. The application detects that the file is invalid.
 - i The application notifies the user that the file is invalid.
 - ii The user chooses if he wants to select another file.
3. The application detects that the User has chosen no alerts.
 - i The application notifies the User that no alerts were chosen to import.
 - ii The User chooses if he wants to select alerts to be imported.

5.2.12 Export Alerts

Export ClockAlarm alerts to a file.

Scope

ClockAlarm application user interface.

Primary actor

User

Precondition

The application has to be started.

Postcondition

Alerts are saved to a file whose path was chosen by the User.

Main success scenario

1. The User is asked where he wants to save the file.
2. The User selects the alerts he wants to export.
3. The application exports the alerts to the file.
4. The application notifies the User that the alerts have been exported successfully.

Extension

1. The application detects that it has not enough privileges to write to the specified folder.
 - i The application notifies the User that it has not enough privileges to write to the specified folder.
 - ii The User chooses if he wants to select another folder.
2. The application detects that the User has chosen no alerts.
 - i The application notifies the User that no alerts were chosen to export.
 - ii The User chooses if he wants to select alerts to be exported.

5.2.13 Snooze Alert

Snooze the alert on the screen.

Scope

ClockAlarm application user interface

Primary actor

User

Precondition

The application has to be started. A default configuration file exists.

Postcondition

The alert is snoozed for a certain amount of time.

Main success scenario

1. The User snoozes the alert.
2. The application changes the alert time of the specific alert and adds an amount of time defined by the configuration.

Extension

2. The application can't find the configuration.
 - i The application notifies the User that the configuration has not been found.
 - ii The application uses the time defined in the default configuration.

Chapter 6

Development decisions

The structure of the development process of this project is based on the book ‘Requirements Engineering Fundamentals’[2].

6.1 Word Processor

In order to write this document, we had to choose between several word processors. We required that the file could be used with a version control system so that the documentation and code could live in the same place. Also, we didn’t want to loose time if there was a need to change the layout. After discussion, we came up with the following list:

1. Microsoft Word
2. Google Docs
3. LibreOffice
4. Scribus
5. T_EX

The main problem with number 1 and 3 is that the output of these programs are binary files. A built in versioning functionality exists in Microsoft Word but it adds another version control layer to the workflow. The .docx and .odt file types are in fact ZIP archives containing XML documents which can be uncompressed. They could be stored in the uncompressed state to be compatible with the chosen version control system [3]. Or they could be converted into another format. Clearly this is not convenient. Therefore we decided that they were not compatible with our first requirement stated above.

Using a Google Doc means that the document will be stored on Google’s servers and not within the folder containing the source code. I thus also violates our first requirement.

Scribus is a free alternative to InDesign and doesn’t produce binary files. It allows a fine-grained control of the layout, frames and styles [4].

Finally there is T_EX, a typesetting system created by Knuth. It provides a low-level language that is not directly used when writing documents. Instead there exist higher level formats such as L^AT_EX or Plain T_EX (much more lower level than L^AT_EX) which provide a large set of macros [5].

Our supervisor strongly recommended us the use of L^AT_EX, that is why we ended up using it.

6.2 Version Control

Out of the multitude of version control systems (VCSs) which exist, the three following are mostly used [6]:

1. Git
2. Mercurial
3. Apache Subversion

All the three offer equivalent features and are equally well suited for our project [1]. We decided to use Git with GitHub mainly because we were already familiar with them. GitHub allows us to easily manage issues and milestones (an equivalent service for mercurial is Bitbucket).

6.3 Python

As two teams are developing separately in parallel an alternative for KAlarm, we had the choice to use Java or Python. We took the latter because we wanted to learn a new programming language.

6.4 Unit Testing

6.5 Test Coverage

Chapter 7

Minutes

7.1 Meeting 1: March 1, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- Present the Use Cases and User stories
- Discussion about the usage of Python

Notes

Use Cases and User stories

- Vision&Goals: The vision and goals of the project have to be enhanced and not taken “as is”.
- If a choice is made during the project, we should write down why it has been made this way. Everything that hasn’t been documented is considered not been done.
- If we go for a specific product (such as DBMS, module, etc.) we should at least try out other alternatives (2–3) and explain why we made the choice of using this specific product.
- Split the user story “reate/delete categories” in two.
- Sort the uses cases by topic.
- Alarms could be snoozed. (for example, if the user is at a meeting snooze all alarms so that they don’t disturb the meeting)
- Alarms should be user specific, i.e., they should belong to the user connected on the computer and not be shared across all the users of the machine.
- Book recommendation (chapter about use cases) -i Applying UML and Patterns (Larman).

Usage of Python

- Strongly discouraged to share python code via e-mail (General life advice)
- Code coverage 70–80% is ok, 30% is not

Tasks

- Improve vision&goals
- Create use cases / user stories / actors correctly
- Read Python doc

Next Meeting

~~TBD~~ (edit) March 17, 2017

7.2 Meeting 2: March 17, 2017

Present at meeting

Samuel Gauthier, Loïc Charrière, Claude Fuhrer

Agenda

- System and Context Boundaries
- System diagrams
- System description

Notes

The Project Goals should not be written down as a list. We need to create full sentences in order to help the reader's comprehension.

Question: "Should this document contain snippets of code?"

Answer: It depends on the target audience. But here we should only show snippets of pseudo code for complicated parts of our code.

System and Context Boundaries

- Some of the Stakeholders will not influence our project. They are only there so that our documentation is complete.

System diagrams

- Good domain model.
- The domain model should contain types only if they are very specific and part of the DM.
- It is not useful to specify interfaces and abstract classes.

System description

- We need to link the User Stories and Use Cases.

Other

Question: "How should our workflow be? Should we write the documentation and the code at the same time?"

Answer: We shouldn't waste too much time on writing documentation. Basically we should use the Scrum method. All the Use Cases (or the most important ones) have to be written down. Then we can decide what has to be included in each sprints. For each sprint we must have a working version of our application. Start with a small version having the alerts date hard coded in the code, with no GUI. Then add a new feature with each sprints.

Tasks

- Rewrite the Project Goals
- Prepare sprints

Next Meeting

TBD

Glossary

L^AT_EX Is a mark up language specially suited for scientific documents. 19

T_EX Is a typesetting system developed by Donald Knuth. 19

Apache Subversion Is a software versioning and revision control system developed by the The Scribus Team. 20

Bitbucket Is a web-based Git and Mercurial repository and Internet hosting service owned by Atlassian. 20

Git Is a software versioning and revision control system originally created by Linus Torvalds and now developed by Junio Hamano. 20

GitHub Is a web-based Git repository and Internet hosting service. 20

Google Docs Is an online word processor created and developed by Google. 19

InDesign Is a desktop publishing software developed by Adobe Systems Incorporated. 19

KAlarm Is a personal alarm scheduler developed by David Jarvie. 3, 20

LibreOffice Is an open source office suite created and developed by The Document Foundation. 19

Mercurial Is a software versioning and revision control system developed by Matt Mackall. 20

Microsoft Word Is a word processor created and developed by Microsoft. 19

Scribus Is a is a desktop publishing application developed by The Scribus Team. 19

VCS version control system. 19

Bibliography

- [1] Patrick Thomson. *Git vs. Mercurial: Please Relax*. Aug. 2008. URL: <https://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/> (visited on 03/12/2017).
- [2] Klaus Pohl and Chris Rupp. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*. English. 1 edition. Santa Barbara, CA : Sebastopol, CA: Rocky Nook, Apr. 2011. ISBN: 978-1-933952-81-9.
- [3] mpm. *ZipdocExtension - Mercurial*. Nov. 2012. URL: <https://www.mercurial-scm.org/wiki/ZipdocExtension> (visited on 03/12/2017).
- [4] William von Hagen. *Open source desktop publishing with Scribus*. Apr. 2013. URL: <http://www.ibm.com/developerworks/library/os-scribus/index.html> (visited on 03/12/2017).
- [5] *Levels of TeX - TeX Users Group*. Jan. 2017. URL: <http://tug.org/levels.html> (visited on 03/12/2017).
- [6] *Version Control Systems Popularity in 2016*. URL: <https://rhodecode.com/insights/version-control-systems-2016> (visited on 03/12/2017).