UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Software Engineering Curriculum

Stanislav Kutasevitš

Communication-oriented Project Management Solution

Master's Thesis (30 ECTS)

Supervisor: Siim Karus

Tartu 2014

Communication-oriented Project Management Solution

Abstract:

Growth of popularity of distributed software development makes development process more adaptive and flexible in terms of human resources. But in order to sustain the process there is an additional burden put on the communication between customer, team members and project managers.

In the contemporary software development practice there exists a number of smart and handy tools, which help making the communication more fluent and convenient. However none of those tools tackle a problem of integrating multiple communicational sources into a single tool.

This paper intends to present a solution to this problem by introducing a collaboration tool for distributed software development. The collaboration tool will be oriented on integration of multiple communication sources and provide analytical information on software development project.

Keywords:

Data integration, analytics, project management, distributed software development

Kommunikatsioonile orienteeritud projektijuhtimise lahendus

Lühikokkuvõte:

Hajusa tarkvaraarenduse populaarsuse kiire kasv muudab tarkvaraloomeprotsessi kohanemisvõimelisemaks ja paindlikumaks inimressursside osas. Selleks, et hajusat loomeprotsessi toetada, tekib kliendi, meeskonna liikmete ja projektijuhi vahel lisakoorem kommunikatsiooni näol.

Kaasaegse tarkvaraarenduse praktikas eksisteerib hulk nutikaid ja mugavaid tööriistu, mis aitavad muuta kommunikatsiooni mugavamaks ja ladusamaks. Kahjuks need riistad ei tegele mitme kommunikatsioonivahendi integratsiooniga ühtseks töötavaks süsteemiks.

Töö eesmärgiks on pakkuda kirjeldatud probleemile lahendus. Töös kirjeldatakse loodud koostöötamise tarkvara, mis on mõeldud toetama hajusat tarkvaraarendust ning mille eesmärgid on erinevate kommunikatsioonivahendite andmete integratsioon ja tarkvaraarenduse projektiga seotud analüütilise informatsiooni pakkumine.

Võtmesõnad:

Andmete integratsioon, analüütika, projekti juhtimine, hajus tarkvaraarendus

Table of Contents

Acknowledgements

I am highly indebted to my supervisor Siim Karus for his guidance, patience, professionalism and the valuable advices that helped bringing this Master's thesis to its current level.

Next I would like to express my deepest regards to Bellcom and its unchallenged leader Jørn Skifter Andersen for suggesting the initial idea, giving enough freedom for its implementation and their kind co-operation.

Lastly I would like to take this opportunity to express my gratitude to HITSA and IT Academy (ITA) for the scholarship grant that supported me throughout the entire study period.

1 Introduction

Software development is a complex process that requires intensive collaboration between customer and software developers, and among the software developers themselves. It has been established, that during the development process programmers actually spend about 60% of their time on communication [1, 2]. This time can be spent on discussing new feature with customer, figuring details of bug report, planning the future work etc.

Taking into account that distributed software development (DSD) is becoming a widespread practice [3], it is not hard to predict that distributed software development (like distributed agile development [4]) requires even more communication between developers. It has been discovered that distributed software development generally takes 2.5 times more time [5].

Since the time spent on communication between regular and DSD has been increased it can be concluded that most of the discussions that previously took place in the real life now takes place online. This fact proves the importance of using the right tools for handling the communication.

1.1 Goal

The intention of this Master's thesis is to investigate the existing problem of distributed software development, study why the available collaboration tools are not sufficient at solving the problem, suggest an alternative solution and develop it.

The result of this Master's thesis will be a running application, that solves the mentioned problem, and the thesis report starting from defining the problem, analysing the current "state of the art", explaining the design decisions and ending with describing the contribution of this Master's thesis.

1.2 Customer

The customer of this project is a company called Bellcom Open Source ApS[1]. Bellcom is a Danish software development company, with headquarters located in Kolding (south of Denmark). It consists of around 20 employees, most of which are developers. Bellcom is a modern company, which uses distributed agile methodology to run its development process.

1.3 Outline

The further structure of the thesis is divided into 8 sections. Section 2 gives a basic introduction to the thesis by defining the problem itself and briefly presenting the idea and some of the key features of the suggested solution. Section 3 covers the related work by presenting the current state of the art in distributed software development and integration fields. Additionally the section gives a description and compares other collaboration tools, which currently exist on the market. Section 4 presents initial steps of the development process, which are requirements elicitation, design decisions and the choice of tools. Section 5 describes the integration implementation between the application and communicational channels – email and Skype, as well as gives an overview of the entire system integration design. Section 6 continues the previous section by explaining how the information piece is processed after it is received by the application: how the matching between re-

[1] http://bellcom.dk

ceived message and the existing content is done and how a message can change the existing content data. Section 7 looks into the analytical part of the application – how the generated and received metadata can be used for analytical purposes, which technologies and techniques were used and what values does this give the end-user. Section 8 covers quality assurance part by running a set of manual tests. Section 9 concludes the results of the thesis, identifies thesis contribution and speculates what future work can be done based on the thesis.

2 Problem

On one hand using distributed software development can give a company set of strong benefits, such as reduced cost and staff liquidity. On the other hand, it produces additional challenges, which can slow down the development process and reduce the quality of the developed product.

2.1 Problem Definition

Fragmented Documentation

Oftentimes the documentation in the agile software development project has the accessory role and is fragmented into multiple pieces. In general distributed software development project in agile methodology produces the following communication artefacts:

- initial project description
- a set of task breakdowns (many of which have an extensive dialog with customer clarifying the requirements)
- a collection of emails between customer and developer/project owner
- a collection of documents (official documents, development related files)
- instant messenger logs (as sometimes it is easier to get a hold of customer in IM application rather than emailing)

This kind of fragmentation of a DSD project specification often results in a lack of a structure, which makes it especially challenging to be followed by managers or to be introduced to a new person (e.g. new developer joining the team) [6, 7].

In order to get a complete overview of the project the manager (or other interested person) would need go though all the tasks (user stories) including the comments and ask the developers in regards to latest emails/instant messages they received from the customer.

The severity of the problem is growing proportionally to the complexity of the project and number of people it involves, not to mention the possibility of multiple customers, who in some cases might provide contradicting requirements.

As a result, there is a fair chance that the DSD project has its specifications in many forms and places (official documents with signatures, project wiki page, user story log etc.) and is constantly updating with the new requirements. There is quite some manual work needs be done in order to just keep everything consistent.

Multiple Communicational Channels

A continuation of previous problem is the diversity of the communication channels between customer and the development team. Ideally all the customers would have to use a single communicational channel, for example, use a common task-tracking tool, which has all the information about the project. Yet many of the customers tend to do things their own way, and instead of adding the comment under the specific task they can send an email, write a comment in version tracking system or send an instant message. Those situations might happen because of multiple reasons, e.g. customer is not well aware of the process, customer is trying to save his time, policy in the company forbids usage of external task-tracking tools etc.

While for the customer this kind of behaviour does not seem causing many troubles, for the development team the fact that requirements and changes are coming from multiple

sources creates a mess. Firstly, that is obligating the recipient of running the change by the project manager, secondly, sharing the new information with the rest of the team, and thirdly, including the updated/new requirements into the project documentation.

All in all this can lead to a troublesome development process and demoralized team.

Ignoring the Project Workflow Metadata

Every incoming information piece that is related to the project (regardless of its origin) has some metadata. The example of the metadata can be message author, recipient, date, attached files etc.

Interrelation of the metadata between different messages is a poorly discovered area with a great potential. The examination of the metadata can give an insight about multiple statistical and analytical grains of the project. It can also give an answer to a number of interesting questions, e.g. “During which phase is the customer participation most active?”, “Which tasks are the most discussed?” etc.

At the moment there is no available tool that would allow running such kind of analysis on the project, and doing it manually does not seem to be realistic due to an extra large number of variables.

2.2 Suggested Solution

The solution to overcome the problems mentioned above is introducing collaboration tool with integration functionality. Tool will be oftentimes referred to its working title – Collatool.

The tools should support storing documentation on the project in one single place, in order to avoid fragmentation of the documentation.

Additionally the tool should serve as some analogue of chronicler that would systematically store (aggregate) every incoming and outgoing information piece. This “chronicler” should be able to connect to potentially any source of information (be that email, IM-client or maybe even SMS) and be able to retrieve the information for future uses.

As the information comes in many forms and shapes, the aggregator should be able to interpret, translate and persist it in a uniform way. Also, as the information comes from different sources/platforms, the aggregator should support different methods of handling the information (meaning both pull and push mechanisms).

Besides presenting the timeline (the timely ordered combined list of messages received from different sources), the tool should also have a feature of automatic changing the information on either a project or task level.

Not to be neglected, the received information contains the metadata, which should be put in use in the collaboration tool. The tool should provide an analytical insight to the project flow in order to bring most value for the project managers.

The tool has to present the software project development information in a clean and understandable way - so that it is both approachable for user and brings the most value.

3 Related Work

As the suggested solution is tightly related to the concepts of distributed software development and data integration, the related works of those fields are going to be used and considered as the state of the art.

3.1 Distributed Software Development

Abbattista et al. raise an issue, that combining agility and distribution is not that simple, as distributed development goes against the very core principle of agile methodology – physical proximity (e.g. face-to-face communication) with both customer and development team [8]. Bearing this issue in mind, Abbattista et al. perform an analysis of commonly used social software types (blogs, wikis, social networking etc.) and the actual implementations of them, coming to the conclusion that social software is a valuable resource that can improve the communication in distributed software development, assist knowledge sharing, help building team and aid the development process in general.

Paasivaara et al. perform a case study, where the different agile practices (such as: daily scrum meeting, weekly scrum meeting, sprint planning, retrospective meetings etc.) were successfully used even with large projects with the teams distributed between location, culture and time zone (Norway/Malaysia) [6]. During the case study the following challenges were met: technical (internet connection is slow/unreliable), communicational (misunderstanding the requirements, cultural differences). In the end, however, Paasivaara concludes that using agile with distributed software development was a positive experience.

3.2 Data Integration

Liu talks about the integration of email into a portal using AJAX (for fetching) and SSO (for authentication) [9]. The portal is connected to the database using Telnet protocol. As a backbone the application uses JSP (JavaServer Pages) that is connected with the mail server.

Baccehelli et al. present a tool called Miller [10] that, according to the specification, supports the following functionality:

- import mailing lists
- manual interaction with emails (e.g. labelling)
- automating search for traceability links between email and code entities
- integrate email in IDE

For the email import (as it is the most related issue) Miler uses web crawler that is extracting the information from MarkMail[2] website, which performs a search in about 8800 mailing lists.

Another example of the related work has been found in the publication by Zürich Software Evolution and Architecture Lab work[3]. In the paper Fischer et al. raised an issue of miss-

[2] http://markmail.org
[3] http://www.ifi.uzh.ch/seal.html

ing connection between the subversion control system (CVS[4] and the bug tracking system (BugZilla tracking system[5]) on the example of Mozilla Web Browser[6] [11].

In order to extract the information (with a further intention to analyse it) authors have used the following approach - see Figure 3.2. While the actual results of the paper do not seem to be directly connected with this project, the approach of retrieving the information from different sources and storing it (so that it is most convenient to use for the future purposes) allows to consider this paper as the related work and use as a starting point.

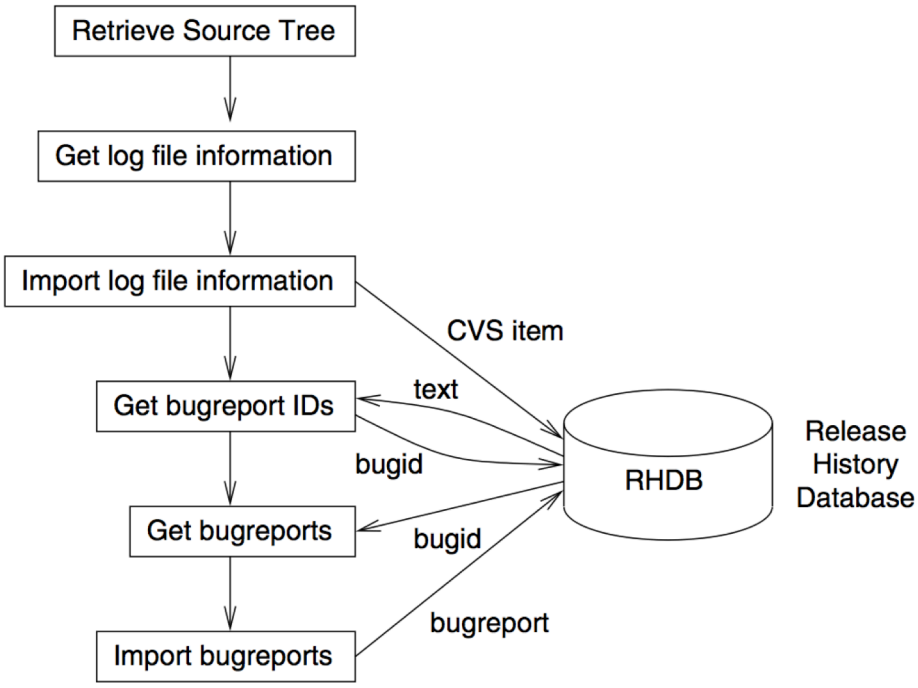


Figure 3.2 Approach of extracting the information from the subversion control system and bug report system [11]

Chung explains the integration style based on passing messages between the integrated applications [12]. This approach allows using centralized message server, which is responsible for communication. Because of centralized message server the whole flow is flexible, meaning that it is possible to easily add new applications to the flow as well as remove the ones not needed.

Brown talks about other advantages that are brought by messaging-passing, those advantages can be listed as follows: adaptability, scalability, simplicity, efficiency etc. [13].

Banner gives an introduction to the concept called unified messaging [14]. Unified messaging is aiming at removing the borders between different communication protocols (the paper is mentioning short message service (SMS)[7], multimedia message service (MMS)[8], email and instant messaging). Schreiner adds the voice exchange protocol [15] to the con-

[4] http://ximbiot.com/cvs/manual
[5] http://www.bugzilla.org
[6] http://www.mozilla.org
[7] http://en.wikipedia.org/wiki/Short_Message_Service
[8] http://en.wikipedia.org/wiki/Multimedia_Messaging_Service

cept of unified messaging, which can convert the textual message to the voice format and vice versa, to the communicational protocols. Using unified messaging approach communication is becoming homogenous to the end-user disregarding of which application the recipient is using.

Wams et al. provide a theoretical solution to the unified messaging problem by introducing a middleware layer [16]. The middleware layer is implementing multiple strategies (based on a number of connected protocols) to deliver the messages to the right channels.

3.3 Existing Solutions

While the range and variety of collaboration tools is quite extensive[9] [10], neither of those tools seems to decently support information aggregation from multiple sources. The only communicational channel some of the tools support is email. But even then the integration is limited to simply receiving the email (and posting it), routing of which is handled via different email addresses – each task has its own dedicated email, sending a message to which would publish the content. A short description of the most popular collaboration tools is provided below.

Assembla[11] is a collaboration tool supporting such features as task/issue management, code repositories, communication (messaging, file sharing, sharing code etc.), managing team (time tracking, activity statistics etc.) as well as customer (multiple permission sets). Assembla has an option of posting the message via email.

SourceForge[12] is an environment for creating projects supporting issue tracking, team communication, code repository, documentation and others with a focus on open source projects. No integration seems to be possible.

Teambox[13] is a collaboration tool with project management functionality (tasks assignment, time tracking, etc.). Additionally it supports integration with Google Drive[14], Dropbox[15], Box[16] and email (posting the message via different emails).

JIRA[17] is a collaboration tool with project management, code integration (embedded code repository supporting commenting, reviews etc.) as well as a number of applications from original Atlassian Marketplace[18], which primary purpose is to extend the JIRA functionality and improve usability. JIRA also supports communication via emails.

Basecamp[19] is a collaboration tool with rich project management functionality, supporting many add-ons[20], has integration with Google Drive.

LiveMinutes[21] is a tool, which allow integration with some communicational/collaboration applications: Evernote[22], Skype (is limited to interacting with locally installed Skype program by initiating audio calls).

[9] http://en.wikipedia.org/wiki/List_of_collaborative_software
[10] http://en.wikipedia.org/wiki/List_of_project_management_software
[11] https://www.assembla.com
[12] http://sourceforge.net
[13] https://teambox.com
[14] http://drive.google.com
[15] https://www.dropbox.com
[16] https://www.box.com
[17] https://www.atlassian.com/software/jira
[18] https://marketplace.atlassian.com
[19] https://basecamp.com
[20] https://basecamp.com/extras
[21] https://www.liveminutes.com

FogBugz[23] is a bug tracking system, which above else offers couple of interesting features with a focus on integration:

- **Customer email management**
  This represents storage for all customer emails with shared access, so that everyone can track the communication with a single customer and communicate with a customer from the shared email address.
- **Webservice integration**
  This is an XML API, that allows manipulating system content via webservice requests (e.g. create case, assign the person to a bug report etc.).
- **URL triggering**
  Allows triggering the specified URL(s) on a certain events in the system with a set of predefined parameters (depending on the event type). That represents another side of the integration – giving the possibility for any other system implementing webservice to listen to changes from FogBugz.

While FogBugz offers a few interesting decisions on integration level and is the closest tool found to what this thesis is trying to achieve, there exists couple of strong arguments, why FogBugz cannot solve the above-mentioned problems:

- FogBugz is bug tracking oriented, not task oriented, which sets number of limitations in the workflow (e.g. task discussion, appending files etc.)
- FogBugz does have any statistical data on the project, which can later be used for analytics
- FogBugz is an internal tool, which does not support involving the customer, which goes against agile development principles and hinders the development process

[22] https://evernote.com
[23] http://www.fogcreek.com/fogbugz

4 Design

Analysis of the existing solutions helps in seeing what is missing in those solutions and therefore come up with a proper list of requirements that will both satisfy the customer and bring the scientific value to the project.

4.1 Requirements

This is no doubt that the project can be implemented in numerous ways. However there have been given a couple of obligatory requirements from the customer that must be fulfilled. Those pre-given requirements are presented below to emphasize on them:

- The solution for this project must be based on the web-interface, more particularly Drupal content management system[24].
- The application must be able to integrate with email communicational channel.
- The application must be able to integrate with Skype.

The rest of the requirements were elicited during multiple discussions with the customer:

1. Application:
    a. Application should cover the notation of project, task and task comment. With a hierarchy of project containing tasks and task containing task comments.
    b. Project should have a list of administrators, list of participants and list of related files.
    c. Task should have a status, due date, estimation and an assignee.
    d. Comment should have an author and the published date.
2. Integration:
    a. Besides the pre-given list of sources (Skype, email) application should be potentially able to integrate with any information source (both static, for example, static webpage and dynamic, for example, another IM-application).
    b. Unobtrusive integration with different platforms (loose coupling).
    c. Supporting of both pull and push mechanism of retrieving the information.
    d. Network failure awareness.
3. Data extraction requirements:
    a. Tool should be able to match the extracted data with the existing content.
    b. Tool should be able to identify multiple people involved in the communication (e.g. multiple customers, multiple developers etc.).
    c. It should be possible to update the content of the project using the communicational channel (in an automatic or semi-automatic mode).
4. Analytics
    a. Tool should provide statistical information about project flow using chart notation.
    b. Tool should provide statistical information about user activity using chart notation.
    c. Tool should provide statistical information about comment sources using chart notation.

[24] https://drupal.org

4.2 Class Diagram

As the development process was tightly conjugated with the research, the class diagram was undergone multiple changes. While the final version of the class diagram can be seen on Figure 4.2, the explanation of some of its non-trivial fields is saved to the dedicated sections.

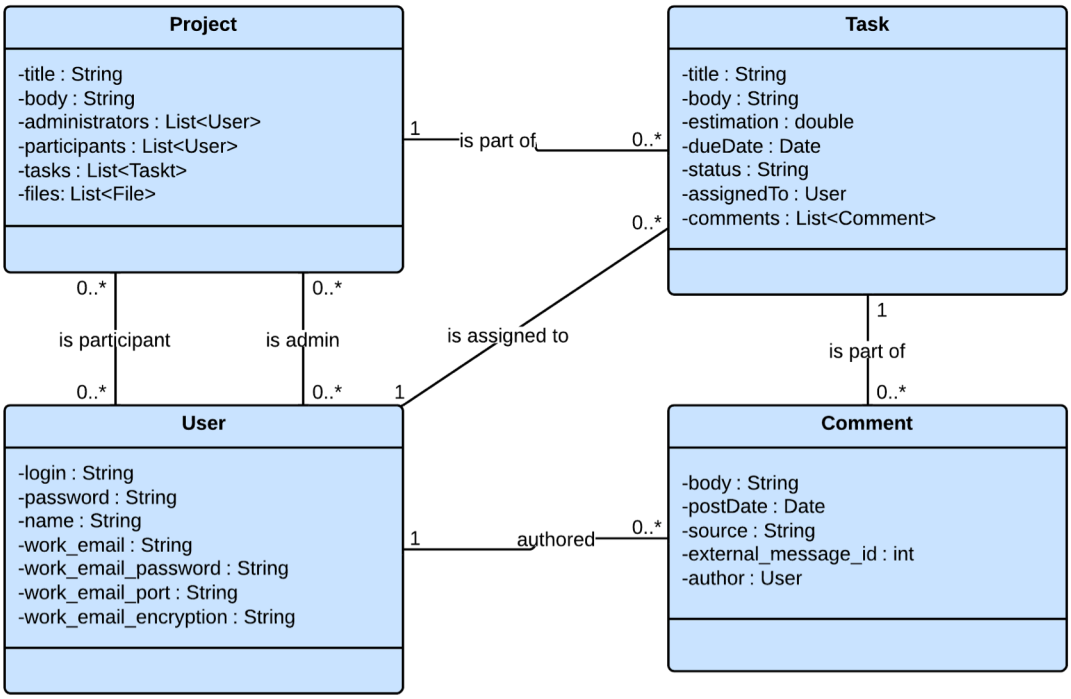


Figure 4.2 Application class diagram

4.3 Architecture Decisions

As the application must to be integrated with multiple communicational sources it is crucial to design a robust and flexible architecture that is capable of that.

There are many ways of how the integration part can be handled (shared databases, remote procedure invocation, file transfer etc.) [17], however not all of them are possible in this case mostly because of the inability to alter other application source code (for example, Skype).

Based on that the messaging integration [17] seemed the most optimal choice. Messaging integration is a concept of integrating different platforms by exchanging small packages, called messages. The architecture of typical messaging system is presented on figure 4.3.1[25].

[25] http://www.enterpriseintegrationpatterns.com/Chapter1.html

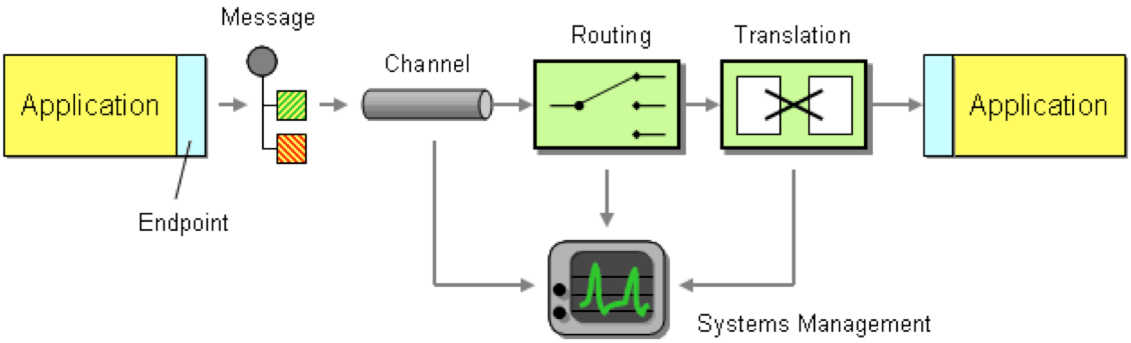


Figure 4.3.1. Typical message system architecture

The reasons for choosing messaging integration are the following:

- **Platform/language independent**
  Contemporary messaging systems allow integration with practically any platform/language.
- **Asynchronous communication**
  The sender does not have to wait for the recipient to receive and process the information, which increases the speed and removes the dependencies.
- **Flexibility of the message size**
  From very small to large data exchange, which allows sending only the information required and therefore reducing the traffic.
- **Reliability**
  Contemporary messaging systems allow ensuring that the message is delivered by sending the confirmation upon receiving.
- **Flexibility**
  The messaging architecture supports adding more communicational into the flow of message exchanging with reasonable effort.
- **Scalability**
  Messaging systems are capable of scaling – increasing the number of messages to be process in a period of time.

The integration with email and Skype using messaging system will be explained in details in the section 5 Integration.

4.4 Choice of Tools

Drupal

As part of the pre-dictated user requirements, the web application has to be developed using Drupal.

Drupal is an open source content management system (CSM) and is built using PHP. Drupal first release was in January 2001 by its original author Dries Buytaert. At the present moment Drupal has had 7 versions and serves as a back-end system for more than 1.9% of all the websites[26].

Drupal as one of the leaders in CMS world is famous for its community, which contributes a lot of efforts into making Drupal more and more efficient to use. These contributions are eventually formed into so-called modules – ready functionality, that can be used by anyone. There are a range variety of modules solving most of the generic problems. However, not all of the modules are properly maintained, and not all of them are mature enough to

[26] http://w3techs.com/technologies/overview/content_management/all

be used in production. So choosing a right modules is always a compromise between how much can it do, how mature is it and how well it is maintained.

As any mature CMS Drupal provides a set of default functionality that can be used right out of the box. This functionality is users (including authentication), creating and publishing content and other.

Drupal Cron

Drupal Cron[27] is part of the Drupal CMS, but since it is going to be referenced quite often, its description seems necessary. In its essence Drupal Cron is a tool that allows scheduling some processing job with different periodicity. Example of that could be sending email to every subscribed user every hour.

Messaging System

There exist a number of messaging systems[28], few of the most popular will be briefly analysed in this section.

RabbitMQ[29] is an open-source messaging system implementing AMQP[30] protocol. The key characteristic of RabbitMQ is, as the official website claims, simplicity. That characteristic is gained by the implementation of the broker pattern[31], which makes RabbitMQ easy to deploy and use. Broker pattern assumes queuing all the messages on central server prior to sending them to the consumers. That gives broker pattern a couple of strong advantages (e.g. routing, load balancing etc.), but results in reduced scalability and loss in speed of processing.

ZeroMQ[32] is an open-source messaging system positioning itself as a lightweight messaging system. It is designed for using in the situations where the high throughput is crucial, e.g. like stock exchange applications.

ZeroMQ possesses a high flexibility, but is notorious for steep learning curve. Additional drawback – is lack in build-in advanced messaging patterns[33]. However implementing them manually using system's high flexibility can solve that issue.

Apache ActiveMQ[34] is an open-source messaging system implementing numerous advanced messaging design patterns. The main advantages of ActiveMQ are high performance. It can be used in both server-client and peer-to-peer communication.

All of the mentioned messaging systems have clients with most common programming languages, support cross-platform message exchange, have extensive documentation and are actively maintained.

As there were no major functionality differences, which were crucial for bringing the project to life, the one that offered the most simplicity - RabbitMQ has been chosen. RabbitMQ offers neatly documented tutorials (available for multiple platforms), which allowed using it almost right "out of the box". Besides that there happens to exist a well-

[27] https://drupal.org/cron
[28] http://en.wikipedia.org/wiki/Category:Message-oriented_middleware
[29] http://www.rabbitmq.com
[30] http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol
[31] http://en.wikipedia.org/wiki/Broker_Pattern
[32] http://zeromq.org
[33] http://www.enterpriseintegrationpatterns.com/toc.html
[34] http://activemq.apache.org

maintained module[35] for Drupal, which support integration with messaging systems implementing AMQP standard.

Other Tools

During the development process, it has been discovered that there is a need for more tools for achieving one or another goal. To keep the report consistent, it has been decided to introduce those secondary tools when they are first required.

4.5 Graphical User Interface (GUI)

As the graphical design is not a primary focus of the project and therefore allocated little time for, it has been decided to reuse the existing GUI solutions as much as possible. Drupal framework allows changing the design of the application by using a concept of theme[36]. Among the range variety of available Drupal themes[37], the one called Bootstrap[38] has been chosen. Bootstrap theme is an adaptation of Bootstrap framework[39], both of which have the following features:

- Mobile first responsive design – takes no or little effort to make application responsive to different design layouts
- Rich collection of styles and elements[40]

The main goal is to make the GUI clean and simple, yet with awareness that none of that should affect the usability. While developing the design the Eight Golden Rules of Shneiderman [21, 22] were used as the guidelines, while some of the rules were followed by Drupal itself, following the others was a manual process:

- **Strive for consistency**
  Every element on the page was created with the same set of styles and colours.
  See Appendix I, Appendix II.
- **Enable frequent users to use shortcuts**
  Every user can add unlimited number of shortcuts using Drupal build-in shortcuts functionality. Figure 4.5.1 shows the shortcut functionality layout.

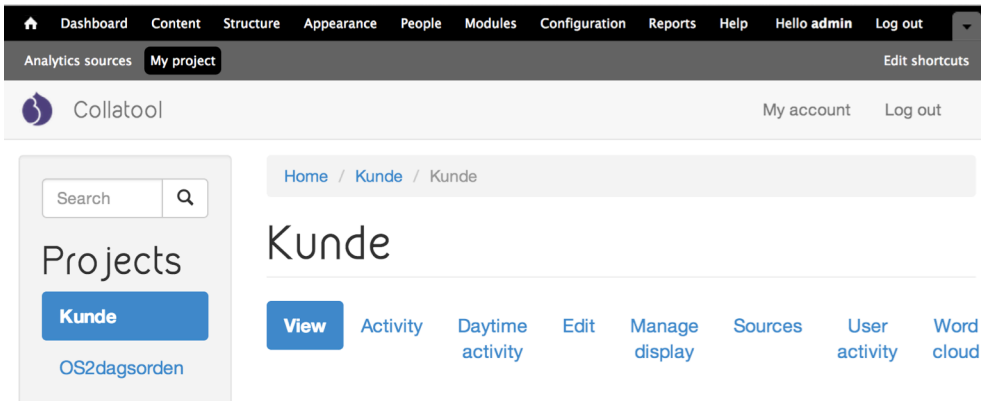


Figure 4.5.1 Shortcuts functionality

[35] https://drupal.org/project/message_broker
[36] https://drupal.org/documentation/theme
[37] https://drupal.org/search/site/?f[0]=ss_meta_type%3Atheme
[38] https://drupal.org/project/bootstrap
[39] http://getbootstrap.com
[40] http://getbootstrap.com/components

Offer informative feedback
Every action in Drupal is followed by an information message, presented on the top of the screen, as figure 4.5.2 shows.

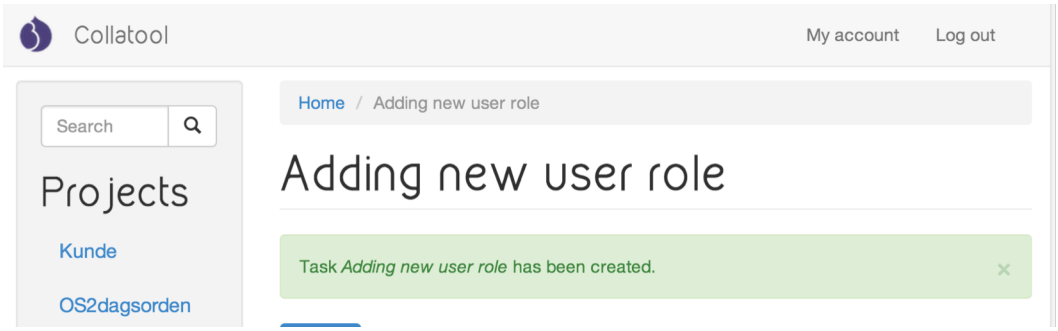


Figure 4.5.2 Information message after creating a task

- **Design dialog to yield closure**
  Not applicable, as application does not have any multi-stepped user actions.
- **Offer simple error handling**
  Drupal offers the internal field validation, on error an informative message is given to a user, as figures 4.5.3a and 4.5.3b show.

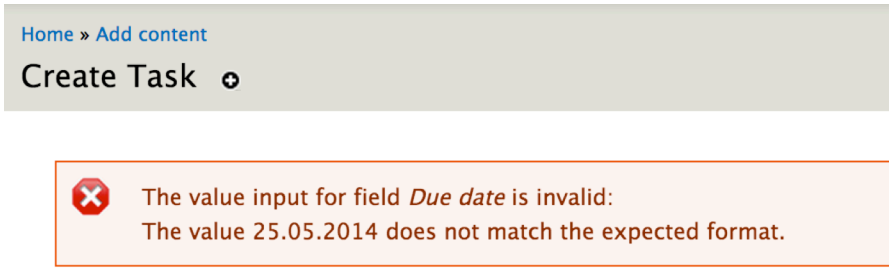


Figure 4.5.3a Validation error message on task creation page

Figure 4.5.3b Validation error field on task creation page

- **Permit easy reversal of actions**
  While adding content or posting the comment user has two ways of revising his or her actions: proactive or reactive. Following proactive way user can preview the content before publishing it, as figure 4.5.4a shows.
  Reactive way allows user to edit the content after it has been published, as figure 4.5.4b shows.

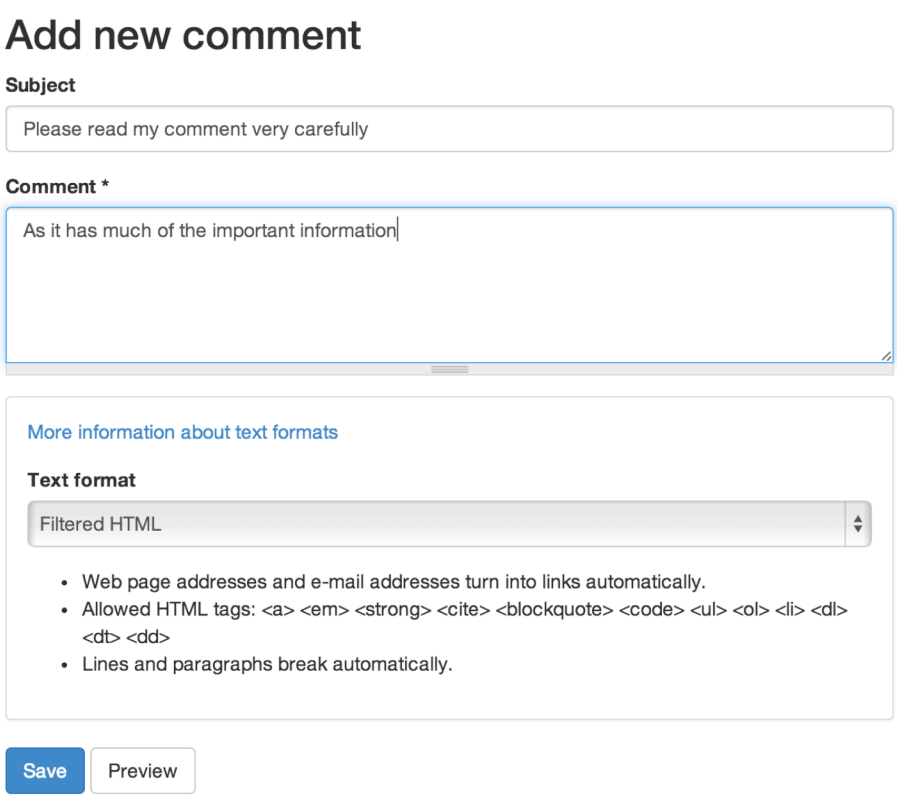


Figure 4.5.4a Preview the content before publishing

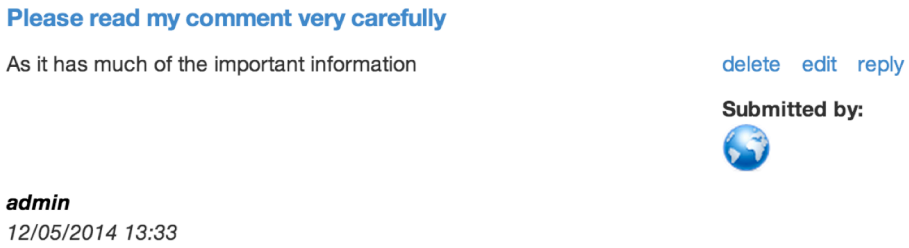


Figure 4.5.4b Editing the published content

- **Support internal locus of control**
  Every important (e.g. content deletion) asks for user confirmation, so user is never loses control of what the application is doing, as figure 4.5.5 presents.

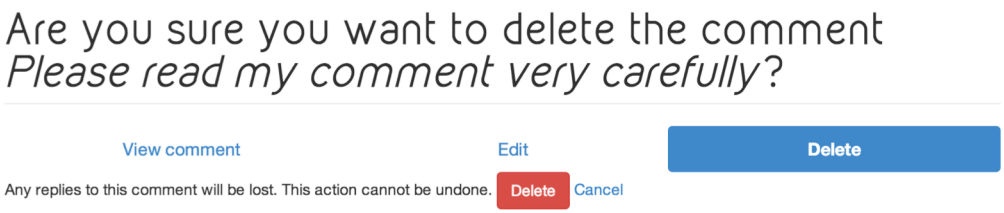


Figure 4.5.5 User confirmation prompt

- **Reduce short-term memory load**
  Not applicable, as application does not have any multi-stepped user actions.

5 Integration

While some of the requirements provided by the customer are rather common and do not involve any research, the description of their implementation is omitted from the thesis. Integration part of the application, being the core functionality, involves both the research and a number of important design decisions, and is explained in details.

5.1 Email Integration

Email is the first communication channel, which application has to integrate to.

Integration with email is possible in two ways:

- **Push mechanism**
  This integration expects user to set up his/her email application to forward the emails to a specific email address. The contemporary emailing systems (such as Gmail[41], Windows Live Hotmail[42] etc.) allow users configuring advanced forwarding rules by creating custom filters[43].
  The benefit of this integration is that user has a full control over what is going out from his/her mailbox. This can be seen useful when user is using his personal email for communication, or when policy of the company forbids forwarding some of the emails.
- **Pull mechanism**
  This integration expects user to provide his credentials so that application itself can access user's email and fetch the latest email messages.
  The benefit of this integration is a simple set-up for a user – there is nothing needed to be done from the user side besides providing the credentials. This can be used if user does not have much technical knowledge, and all of the communication is allowed for reading by an automatic tool.

As both of the methods have some advantages and disadvantages, it has been decided to let the user choose by implementing both of those methods.

The design of the email integration is presented on figure 5.1.

[41] http://gmail.com
[42] http://hotmail.com
[43] https://support.google.com/mail/answer/6579

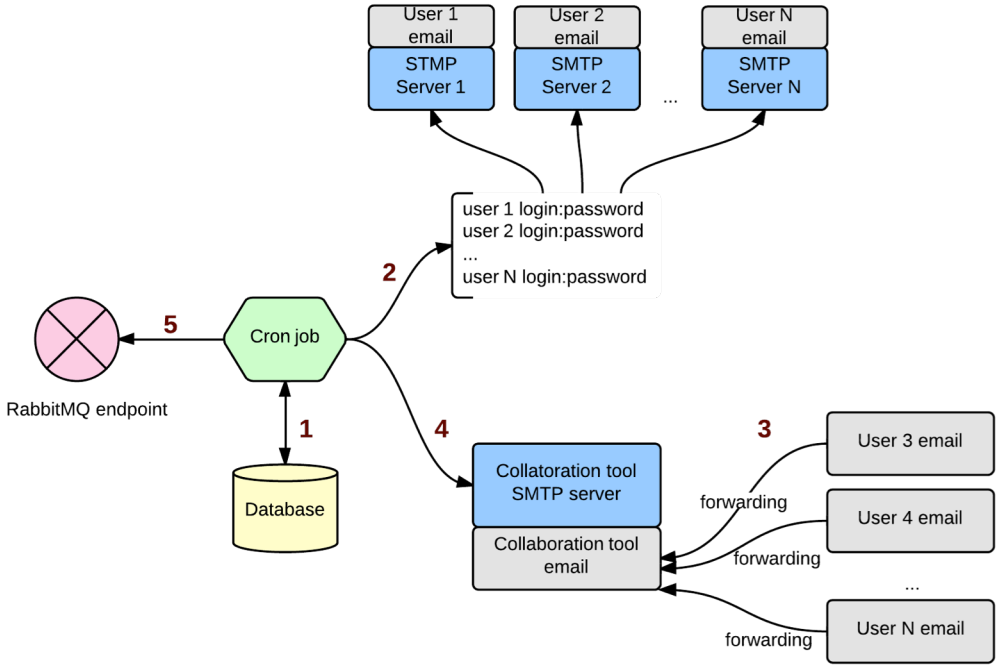


Figure 5.1. Email integration diagram

The descriptions of the email integration diagram are the following:

1. Cron job (scheduled to be triggered with some periodicity) is fetching the list of user email credentials, which are:
    a. Simple Mail Transfer Protocol[44] (SMTP) server host and port
    b. user email login
    c. user email password
    d. encryption method (if any)
    e. timestamp of last check of that particular email address
2. Using the received credentials cron job is making an Internet Message Access Protocol[45] (IMAP) request to each SMTP server for getting new emails. In order to get only those email that are dated after the last check timestamp, IMAP request is supplemented with the last check date. Unfortunately, the filtering in IMAP can be done only with a precision of a day (it does not support hours, minutes and seconds), so if last timestamp was, for example, 27.02.2014 16:37, with IMAP request all the email dated after 27.02.2014 00:00 will be received. The rest of the filtering – emails discarding is done using application logic.
3. Those user emails, which were set up by users, are forwarding messages to a common email address of the collaboration tool at their own pace.
4. Based on the same mechanism as was described in 2, cron job is checking the common email of collaboration tool.
5. All the messaged receiving from common of users' own emails are delivered to RabbitMQ server endpoint as an AMQP message.

[44] http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
[45] http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol

5.2 Skype Integration

Skype is the second communicational channel, which application has to integrate to.
In contrast to the email integration, Skype has a peculiar feature that fundamentally reduces the integration options: Skype does not store messages online. All messages of Skype are kept locally in a small file – SQLite database[46]. More over there is absolutely no API provided by Skype to ease a retrieval of the messages for the developers.

These limitations of Skype did not allow using pull mechanism for fetching the messages, which only left an option of using the push mechanism.

The pushing of the messaging happens transparently for the end user after he installs a developed plugin. As Skype is widely used application on many platforms[47], it has been decided to use cross-platform programming language that is supported on every popular desktop operating system (Windows[48], Linux[49], OS X[50]) - Java[51].

The intention of the plugin is to read the incoming/outgoing messages and send them to the application backend. The plugin could have been made as a service to be running in the background, but since it can be potentially classified as a spying program, it has been decided to make its running state completely obvious for the user. It has also been assumed that Skype account can only be used as a corporate account, which completely eliminates any personal communication. However by clicking "Start" and "Stop" button user can control when plugin is logging the user communication.

The application user interface is presented in figure 5.2.1.

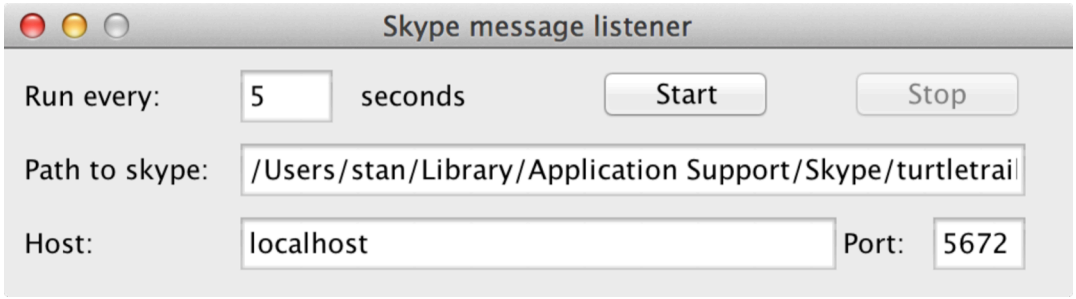


Figure 5.2.1 Skype message listener user interface

The architecture of the Skype message listener plugin is presented in the figure 5.2.2.

[46] http://www.sqlite.org
[47] http://www.telecompaper.com/news/skype-grows-fy-revenues-20-reaches-663-mln-users--790254
[48] http://windows.microsoft.com
[49] http://www.linux.org
[50] https://www.apple.com/osx
[51] http://www.java.com

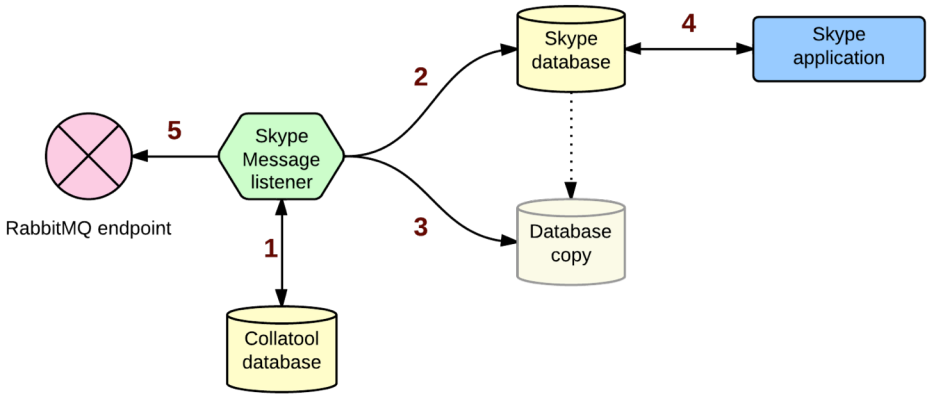


Figure 5.2.2. Skype integration diagram

The descriptions of the Skype integration diagram are the following:

1. With the scheduled periodicity (set by user in "Run every" field, see Figure 5.2.2) plugin connects to collaboration tool SQLite database. After that plugin receives a timestamp, which represents either the moment when user started the plugin (by clicking "Start" button) or timestamp of the latest message fetched from the Skype database – in case plugin is running second or subsequent iteration.
2. After that plugin connects to SQLite Skype database. It fetches the last 500 messages starting for the received timestamp. It has been assumed that the amount of messages per scheduled time does not increase amount of 500.
3. It has been noticed that if Skype application is running in parallel with the plugin, plugin cannot connect to Skype database because Skype puts an exclusive lock on it, which prevent from reading and writing the database.
   To solve this issue and avoid waiting Skype to release the lock (which happens only after Skype application is closed), the temporary copy of the database is created and plugin connects to it instead. The temporal copy has all the latest messages of the original database (by the state of when the copy has been created), is lock-free and exists only while reading is in progress. Moreover creation of it happens instantaneously due to very small size of the original database (couple of megabytes).
   Fetching of the messages is done as describe in step 2. After fetching is done, database copy is deleted.
4. Skype application is reading and writing to its original database without any awareness of the running plugin.
5. The fetched messages delivered to RabbitMQ server endpoint as an AMQP message.

5.3 Canonical Data Model

As application is receiving the messages from multiple channels (currently: Email and Skype), there is very little chance that the structure and the formatting of the received message will be the same. It means that every communication channel would require custom piece of code for message processing. As the message processing does not seem to be a trivial task (see 6 Message processing), the code for that is likely to be quite extensive and complex. Having the similar code existing in multiple variations (for each message

format) not only creates the code duplication, but also makes it much harder to do the changes and/or corrections.

In order to avoid such issue it has been decided to use a concept of canonical format [17]. The idea behind it is to convert each message into a common (canonical) format and then treat each message homogeneously, disregarding of its origin. Figure 5.3.1[52] demonstrates the approach of canonical format.

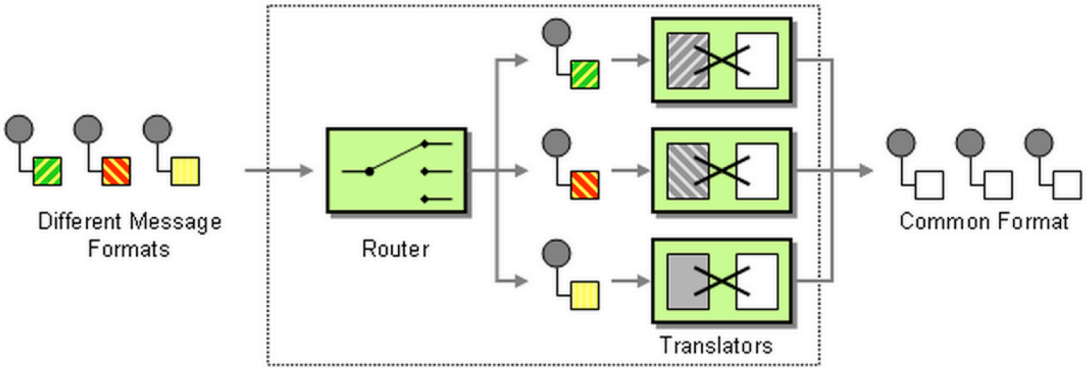


Figure 5.3.1 Canonical data format approach

This approach does not allow complete avoiding of writing the custom code, but amount of custom code is shortened to minimum – its solely purpose is converting the custom format into a canonical format. Therefore the advantages of the canonical data model can be listed as follows:

- custom code is simple and short (in comparison with fully functional code for each format)
- no need for the code change (all functional code changes take place on the higher level)

The canonical format chosen for the message is presented in table 5.3.2.

Table 5.3.2 Message canonical format

Field	Description
external_id	external id of the message, if any
sender	email address or Skype identifier of the sender
receiver	email address or Skype identifier of the recipient
date	Date in the following format: dd.mm.yyyy HH:mm:ii, e.g. 24.03.2014 15:36:55
body	Content of the message
source_type	Skype, Email etc.)
meta	meta information used for search purposes

[52] http://www.enterpriseintegrationpatterns.com/Normalizer.html

While purpose of the other fields is obvious, the purpose of meta field has to be emphasized explicitly: the purpose of the meta field is to provide some data in free form that can aid during search of the corresponding content (see 6.1 Classifying Message):

- For email message the meta data is extracted from previous messages, which come as a quotation below the real message. Those previous messages have no value for the application (assuming that content of those messages is already in the system), but are a great aid for finding the corresponding content types, as they could potentially contain many keywords.
- For IM messages there has not been found any meta data that can ease the search of the content types.
  However, in the future implementation some amount of previous IM messages can be used to compose a metadata for a single message.

5.4 Overall Architecture Overview

The overview of the entire integration architecture (including the AMQP design principles) is presented in figure 5.4.

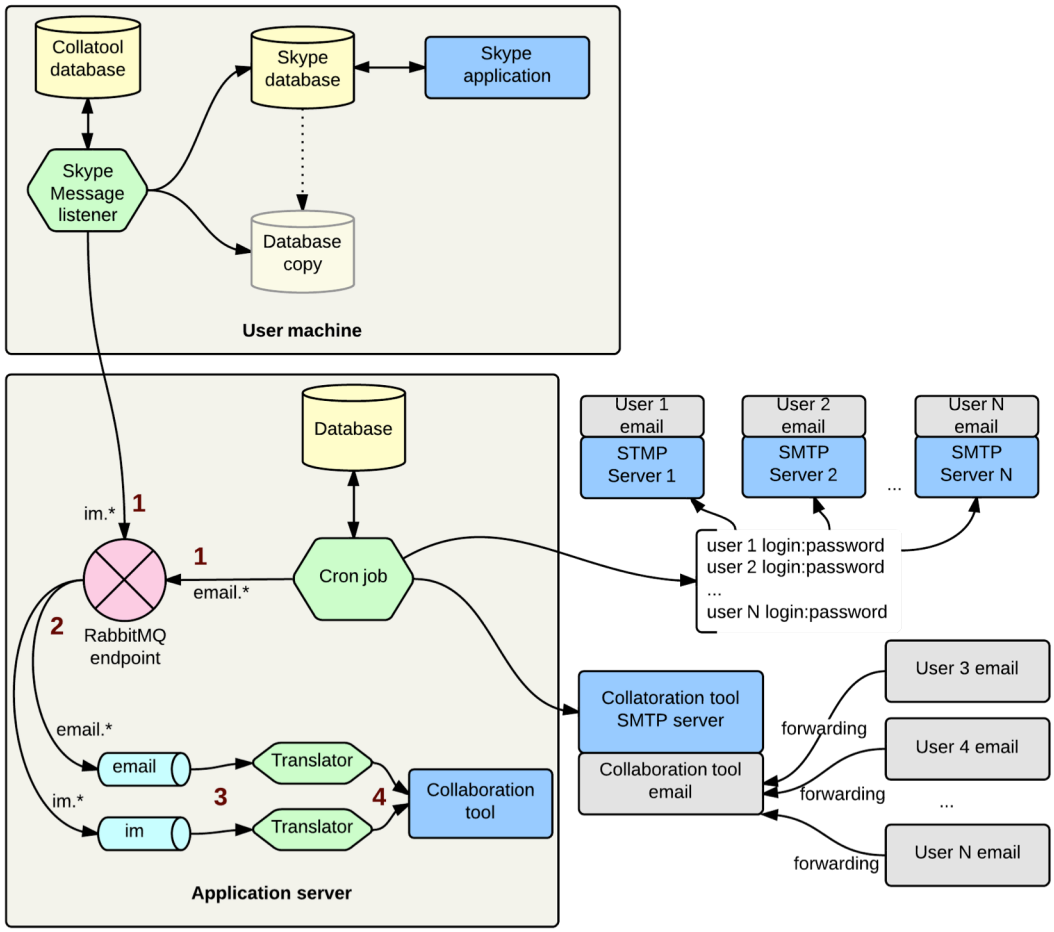


Figure 5.4. Full integration diagram

The descriptions of the full integration diagram are the following:

1. When sending the AMQP messages, each of them is supplied with a special tag – routing key, which lately allows distinguishing what is the message origin.
2. Based on the routing key, messages are forwarded to different queues.
3. Translator is fetching the message from the queue based in FIFO (first-in-first-out principle), and changes the message to respect the canonical format, presented in the paragraph 5.3 Canonical data model.
4. After translation all the messages are delivered to collaboration tool.

Besides robustness, the flexibility of the current architecture has to be mentioned. In order to add another communicational source the only change that has to be done on the application side is creating new queue and adding a custom translator.

6 Message Processing

Message processing is a continuation of integration part and is intended to explain how the received by the application piece of information (a message) is matched against the existing content.

6.1 Classifying Message

In order to know how to process the message, every message received by the application has to be matched with the corresponding content: either project or task. For doing this a concept of full text search[53] is used. The process of matching is presented in figure 6.1.

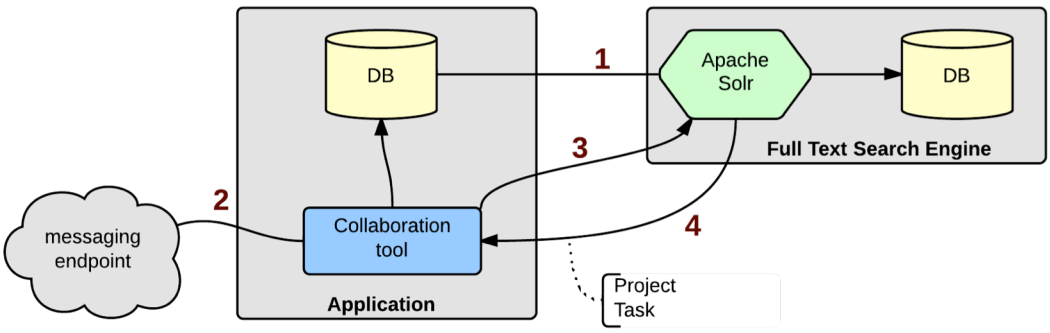


Figure 6.1 Matching content diagram

The description of the matching content diagram is the following:

1. On demand or with the scheduled periodicity full text search engine is indexing the database of the application, making it possible to be later on searched using a text. The engine used for indexing is Apache Solr[54], along with Apache Solr Search[55] for Drupal.
   There are created search indexes for project and tasks (task's comments are indexed as part of the task). Indexing process consists of splitting text into separate words, applying lowercase, removing plural form, omitting the stemming words etc.
   Apache Solr is indexing each field of the content with a possibility to set custom weight (from 0.0 to 21.0) on each field. The custom weights are set on both project and task fields:
   - Title (21.0)
   - Body (13.0)

   Rest of the fields is left with default (1.0) weight.
2. The message is received by the application in a regular way (through messaging endpoint).

[53] http://en.wikipedia.org/wiki/Full_text_search
[54] https://lucene.apache.org/solr
[55] https://drupal.org/project/apachesolr

3. The query based on the received message is sent to full text search engine. The search query consists of a concatenated message body and message meta fields (see 5.3 Canonical data format). Upon search query it is possible to specify which content types is expected as the result.
4. Search query, same as the indexing process, is put under number of transformations (lowering case, removing plural form etc.) and is compared against the indexed content. Most relevant results (1 project and 1 task) are returned by the query, which are then considered to be the found entities.

Comparing found project and task can lead to the following scenarios:

- Perfect match: found task is part of the found project
- Partial match: found task is not part of the found project, or project is not found
- No match: task is not found

In case of the perfect match message is added as a comment to the corresponding task. Partial match and no match ignore the received message.

Later on the concept of the partial match can be used to temporarily add the message to multiple places, so that application moderator can decide which place is correct. Upon confirming the right place for the particular comment – it will be deleted from the rest of the places.

6.2 Business Commands Engine

Part of the requirements was to be able to update the content via received messages in an automatic or semi-automatic mode. In order to achieve that it has been decided to use the concept of business rules engine [18] and natural language processing [19].

Business commands extraction process is shown on figure 6.2.0.1.

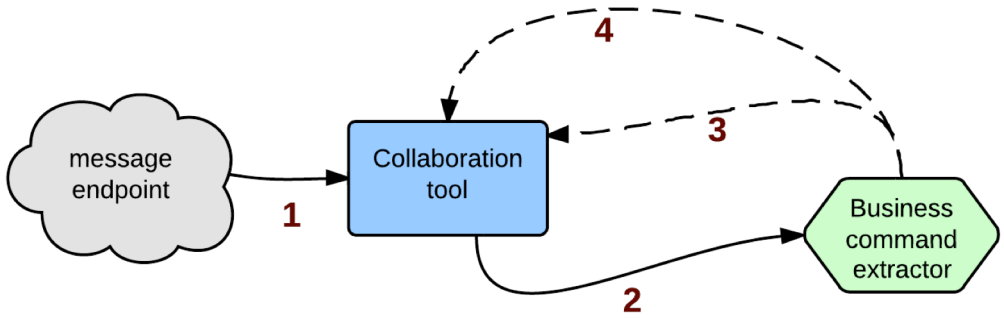


Figure 6.2.0.1 Business command extraction process

The diagram descriptions are the following:

1. The message is received by the application in a regular way (through messaging endpoint).
2. Received message is sent to business command extractor.
3. Business command extractor processes the message by comparing it against the number of regular expressions. The found commands are returned to the application.

4. If the commands are found, business command extractor cuts them from the message body (it can be useful for the full text search query for increasing the relevance precision, see 6.1 Classifying message).

Currently application supports two types of commands: create and update. In order to improve the user experience, and ease the control of the system the commands recognized by the application are written in simple English. The following formats currently supported presented in table 6.2.0.2.

Table 6.2.0.2 Commands supported formats

Format	Matching regular expression
set *context field_name* **as** *new_value***.**	\s?set (task|project) \w*(\w*)? as \w*(\w*)*\.
create task with *field_name* **as** *new_value***, with** *field_name* **as** *new_value***.**	\s?create task ((\,)?with \w*(\w*)? as \w*(\w*)*)+\.

Written in **bold** are the mandatory parts of the format (used for identifying and parsing the command) and written in *italic* are the variables:

- context – is either task or project, it defines the level, which the command should affect.
- field_name – is the name of the field, which should get updated.
- new_value – is the value, which the selected field should be set to. The value supports strings, number and dates.
  Date parsing has been also made in intelligent way – there is no dependency on the date format, the natural values such as today, tomorrow etc. are also supported.

The variables are extracted from the command with the various substring methods, which are dependent on the string indexes of the mandatory parts.

"Create" command deliberately supports task context only, as it has been decided that project creation requires manual actions. Additionally, create commands support unlimited number of fields to be set, which are separated using the following separator "**, with**".

Fields Synonyms

Any language has multiple ways to express the same things, for example, the title of the task can be called – name, subject, headline etc. Having that thought in mind, in order to lessen the strictness of the requirements and give end-users more flexibility in writing the commands it has been decided to use a list of aliases for each field name. That reduces the number of mistakes and increases the overall application intelligence. The synonyms definition is presented in figure 6.2.1.

The mapping is done in an .ini file, which makes it very easy for the application administrators to extend. Each field is places under the group:

- simple
- text
- date

Figure 6.2.1 Fields aliases

```
[text]
;Mapping for estimation
estimation = field_task_estimation
estimate = field_task_estimation
hours = field_task_estimation
;Mapping for body field
body = body
description = body
desc = body

[date]
;Mapping for due date
due date = field_task_due_date
due to = field_task_due_date
duedate = field_task_due_date
date = field_task_due_date
deadline = field_task_due_date
```

That decides how the field assigning should be handled by Drupal. Field can have unlimited number of aliases consisting of one or several words.

Business Commands Examples

This section gives the examples of the messages with business commands inline, syntax of the business commands is highlighted:

> "Hi, few clarifications about the conferences project. The hotel rooms list task is going to take a bit longer. **Set** *task estimate* **as** *5 hours***. Set** *task deadline* **as** *tomorrow***.**"

This message updates the corresponding task estimation to 5 hours, and sets the deadline to tomorrow.

> "Regarding the booking process in the conferences project. I think we are going to use another framework for that. **Set** *task name* **as** *Use Angularjs for the booking***.**"

This message updates the corresponding task title to "Use Angularjs for the booking".

> "Note of the conferences project. **Create task with** *title* **as** *Modify the reservation process***, with** *description* **as** *The modified process should now use AJAX to increase the load time***.**"

This message creates a new task with title "Modify the reservation process" and description "The modified process should not use AJAX to increase the load time".

7 Analytics

As the best way for presenting the information is by making a visual presentation of it [20], it has been decided to create a chart representation of the application statistical data.

The solution does not operate too much of the numerical information, which can be used for analytic purposes. Despite that, there is still some numerical data that can be used to build non-trivial charts, for example:

- Received message date
- Number of messages (per user)
- Number of messages (per information source)

7.1 Used Tools

There exist quite a number[56] of tools capable of drawing the informative charts. Here is the list of most possible choices:

- D3.js[57]
- Highcharts[58]
- Chart.js[59]
- JSCharts[60]

As feature-wise the tools are rather similar, the choice has been mostly done on the degree of integration with Drupal platform and the product license. Among everything Highcharts has been chosen for a number of reasons:

- Good selection of chart types[61]
- Issued under Creative Commons Attribution-NonCommercial 3.0 License[62]
- Has a Drupal integration[63]
- Is based on JavaScript[64], which allows it to be platform and device independent

Unfortunately, despite the good selection of charts Highcharts did not have a chart of type word cloud. For that chart a free library JQCloud[65] has been used.

7.2 Charts

Using the application statistical information six chart types (presented in the following sections) have been created, each of which is available for three levels of abstraction:

- Task – takes into account information regarding one particular task only.
- Project – takes into account information regarding all task of particular project.
- Global – takes into account information of all tasks of all projects.

[56] http://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_frameworks
[57] http://d3js.org
[58] http://www.highcharts.com
[59] http://www.chartjs.org
[60] http://www.jscharts.com
[61] http://www.highcharts.com/demo
[62] http://creativecommons.org/licenses/by-nc/3.0
[63] https://drupal.org/project/highcharts
[64] http://en.wikipedia.org/wiki/JavaScript
[65] https://github.com/lucaong/jQCloud

As there is the hierarchical structure in the solution, as shown on Figure 7.2.0.1

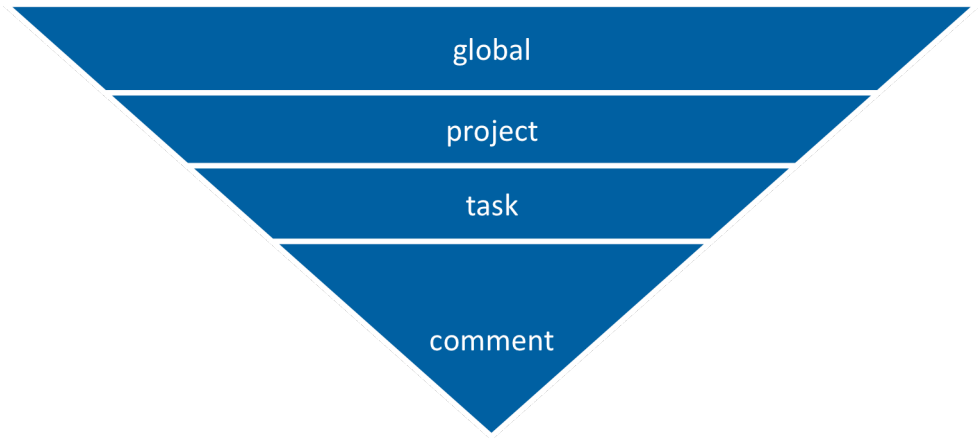


Figure 7.2.0.1 Solution hierarchical structure

it possible to use the same approach for calculating the statistical information. The approach consists of creating only one piece of functional code per chart type – for task level. It involves looping though all the comments of the task and doing the required calculation.

In this case, another level of abstraction (global or project) simply means merging the calculated result from multiple tasks. The approach can be formulated as presented on Figure 7.2.0.2.

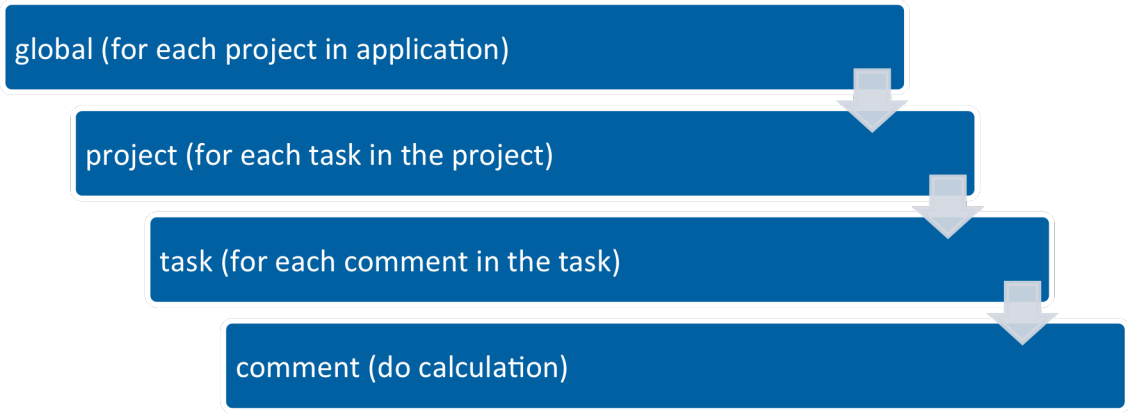


Figure 7.2.0.2 Calculation approach

Activity

Activity chart presents the total number of comments received, disregarding of the origin, grouped by date.

Example of the chart can be seen on Figure 7.2.1.

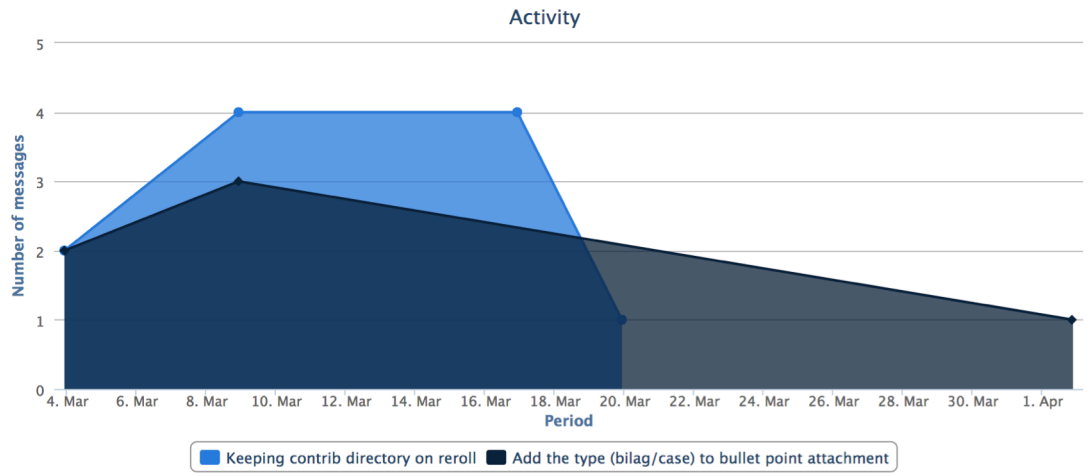


Figure 7.2.1 Activity chart, project level

Daytime Activity

Daytime activity chart presents the total number of comments received in each day (disregarding of comments origin) grouped by day of the week and hours of the day. Inspiration to this chart has been found on GitHub[66].

Example of the chart can be seen on Figure 7.2.2.

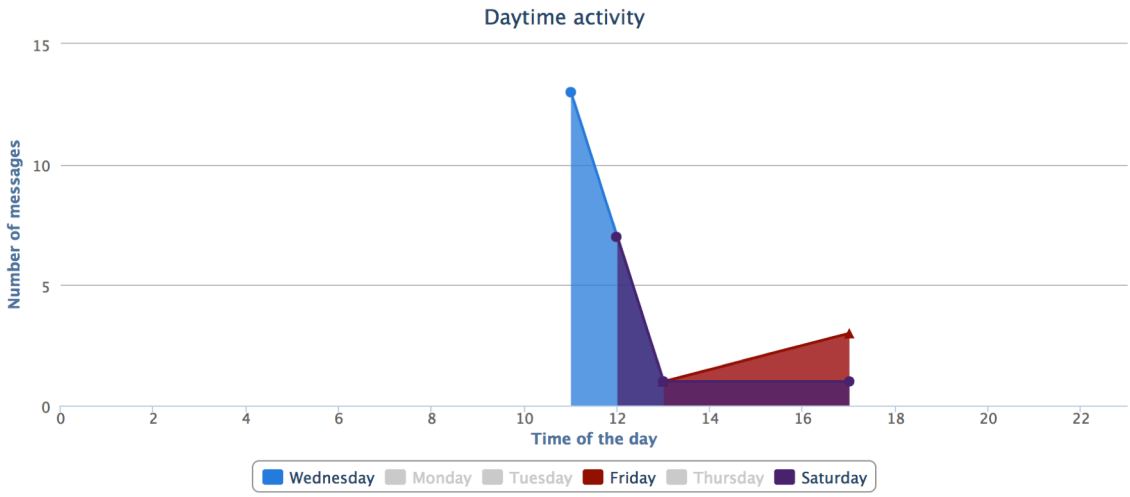


Figure 7.2.2 Daytime activity chart, global level

Sources

Sources chart presents the total number of comments grouped by their origin/source.

[66] https://help.github.com/articles/using-graphs#punchcard

Example of the chart can be seen on Figure 7.2.3.

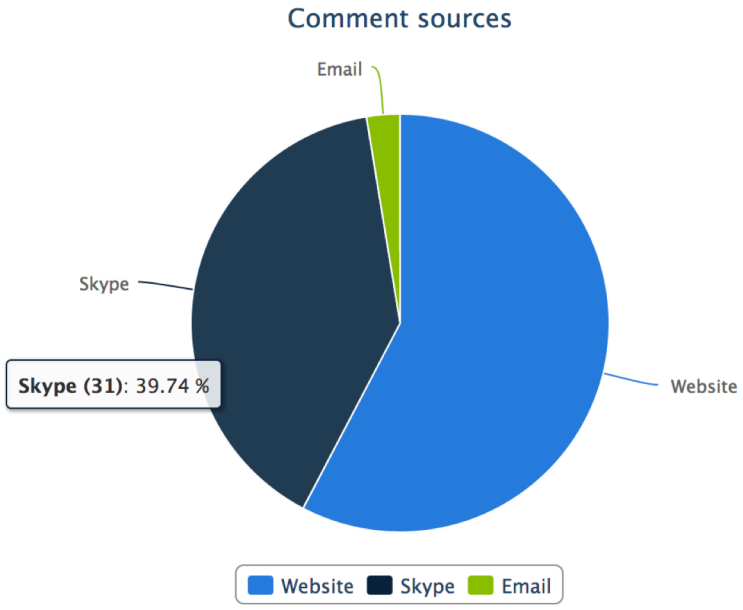


Figure 7.2.3 Source chart, global level

User Activity

User activity chart presents the total number of comments grouped by the user.

The example of the chart can be seen on Figure 7.2.4.

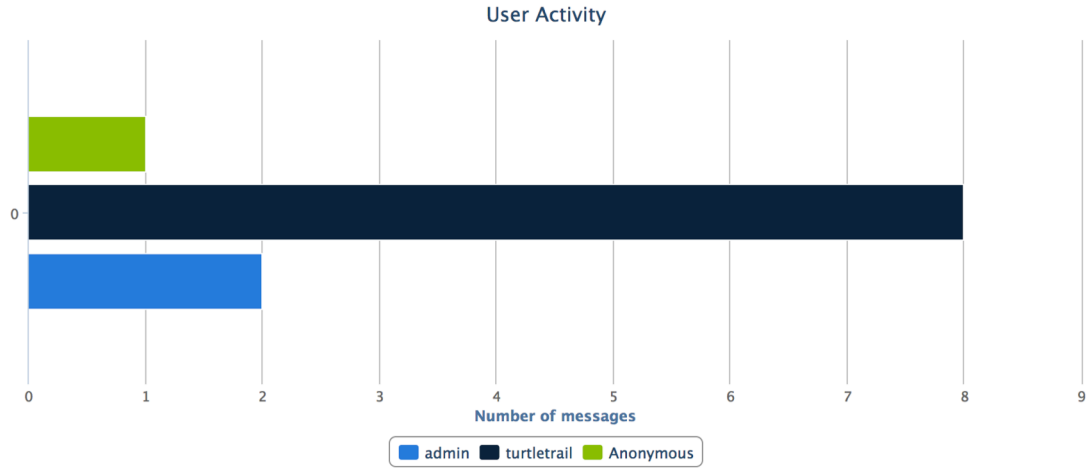


Figure 7.2.4 User activity chart, task level

Word Count

Word count chart presents the statistic on the used words – how many times each word has been used in the text of the comments. In order to improve the output of the result the text of the comments is put under few modifications:

1. Remove links using the regular expression filter.

2. Remove any non-text characters using the regular expression filter (e.g. dots, comas, slashes etc.).

Example of the chart can be seen on Figure 7.2.5.

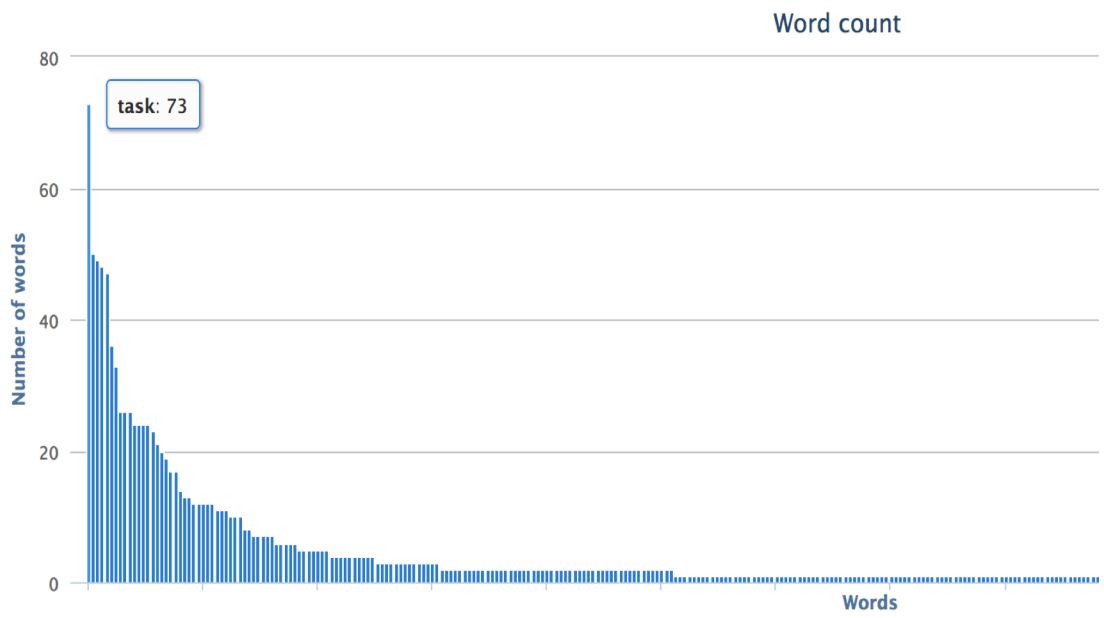


Figure 7.2.5 Word count, global level

Word Cloud

Word cloud chart is an extension of the Word count chart presented in paragraph 7.2.5. It uses the same information – amount of each word, but presents it as a cloud of words with different sizes, where size represents the weight/frequency of each word.

Example of the chart can be seen on Figure 7.2.6.

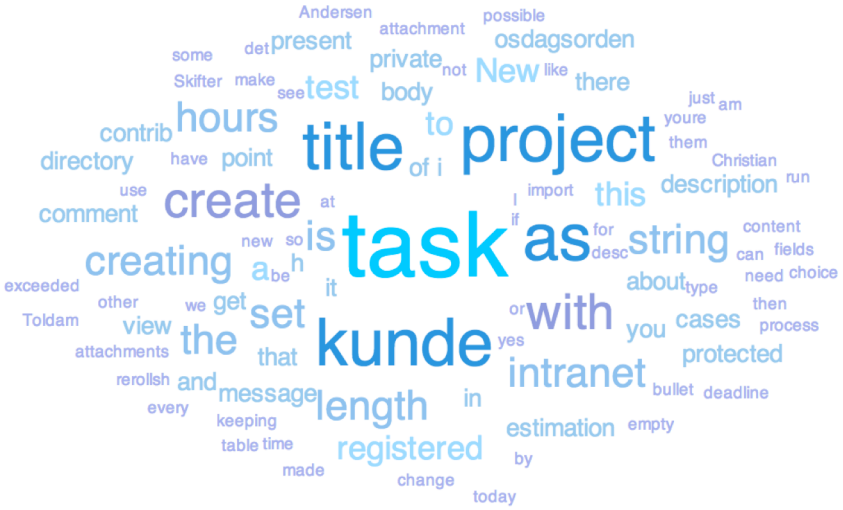


Figure 7.2.6 Word cloud, global level

7.3 Caching Mechanism

Calculating any statistical information is an expensive task (in terms of the resources), and doing this on visitor request means poor response time. Moreover, doing the very same procedure for many users within the same timeframe is clearly a waste of resources. In order to tackle these two problems a caching mechanism has been developed. Caching mechanism allows to:

- Calculate the statistical information prior to user request and therefore drastically reduce the page load time.
- Schedule the statistical information calculation and therefore manipulate caching expiration.
- Serve the same statistical information to all users, as the information is independent on the logged in user.

For storing the cached data, a database table has been created with the structure described in Table 7.3.

Table 7.3 Caching mechanism database table

Name	Description
nid	Stands for node ID – the internal ID of the context/abstraction – task or project. In case of global level 0 is set.
cached_data	String representation of the cached data. Clarified in the next section.
cached_data_type	activity | daytime_activity | sources | user_activity | word_count
last_update	Timestamp of the when the data has been gathered

7.4 Chart Building Sum-up

In order to sum up sections 7.2 Charts and 7.3 Caching Mechanism and give reader a clear understanding of how the chart building in combination with caching works, the workflow is presented on Figure 7.4.

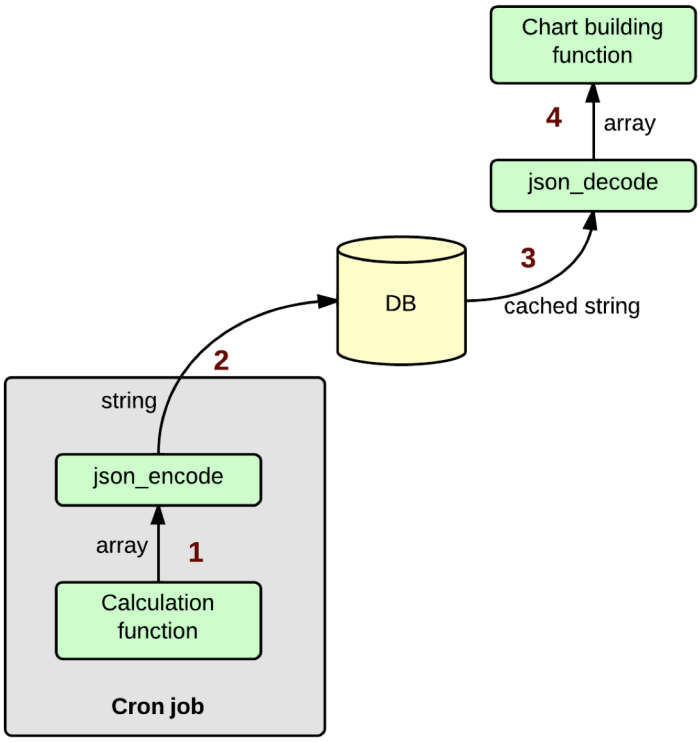


Figure 7.4 Chart building sum-up

The descriptions to the diagram are the following:

1. With the scheduled periodicity cron job starts calculate function (different for each type of chart):
   - Task level - For each single task
   - Project level - For all tasks of each single project
   - Global level - For all tasks of all projects

   It has to be noted specifically, that the higher levels of abstraction do not depend on the lower level, for example, when doing the calculation for project level, previously calculated task level statistical information is not reused and therefore the calculation is repeated. Justification of that approach is the following:
   - Calculation order is independent, which means that one can be run asynchronously from another.
   - Cron job runs seldom enough to neglect the extra resources load (it does not make sense to recalculate the statistical information more often than once an hour).

   Alternative approach could be caching the statistical information for task level only, and later - on page request merge the results of different tasks (e.g. for project level – all tasks from a particular project). Advantages of this approach are:
   - Normalized structure
   - Reduced time of cron job (due to lower number of calculation)

   Whereas the disadvantages are:
   - Increased time for each analytical page load (each request requires additional data manipulation and database read operations).
   - Complex code for post-merging the result from multiple tasks

The disadvantages outweigh the advantages of this approach mainly because reducing time on each page request is much more important that saving time on occasional cron job run, therefore this approach has not been chosen.

2. The received result (array) is converted into the string representation using json_encode function[67].
3. On page request the required cached data is extracted from the database as a string.
4. String is converted back to array using json_decode function[68].

It has been assumed that everyone involved in the project has the permissions to see the statistical project information.

7.5 End-user Value

Capturing the information and visually presenting it to the end-customer using the mentioned charts allows answering the following set of questions:

1. Which of the project/task has been more discussed and when?
    a. Can the discussion peaks and notches be justified by the requirements changes, customer involving etc.?
2. When does the team spend most of their time discussing the projects?
    a. Does the weakly SCRUM on Wednesday helps unload the online communication or on the contrary makes team spend more time on online communication?
3. What is the most preferred channel of communicating among the team members?
    a. Does it make sense to invest in the new channel, or no one cares about it?
4. Who is the most active user and where?
    a. Which ones are the most "chatty": old gurus or newbies of the projects?
5. Which words are the most used among the discussion?
    a. Is the overall mood of the team positive or negative – which words prevail?
    b. How often does cursing happen in the project?

Generally speaking the analytical information provides a deep insight of the project and people involved. Taking into account that the number and the variety of charts are practically unlimited, the analytical part of the application has a great potential to grow, evolve and provide even more valuable information, which can be used by managers to improve the personal/professional skills of their team members and the general workflow.

[67] http://www.php.net/manual/en/function.json-encode.php
[68] http://www.php.net/manual/en/function.json-decode.php

8 Testing

It has to be mentioned that some parts of the application (for example, user registration, content creation etc.) are provided by Drupal CMS and therefore testing them is out of the scope of this thesis. Testing the rest of the application feature, those that are directly related to the thesis goals, is going to be performed using an approach called exploratory testing [23] suggested by Whittaker [24].

Exploratory testing is a type of manual testing where testers interact with the application in any desired way with a purpose of discovering bugs. Compared to the traditional testing, exploratory testing, allows tester to get an understanding of the application flow. That is the reason this kind of testing has been chosen – make reader more familiar with application workflow and summarize the main features.

In order to do that Whittaker's approach called *The Landmark tour* [24] is going to be used. Landmark tour implies selecting one landmark, reaching it and after that selecting a new one, until all of the desired landmarks are visited. The metaphor of *landmark* will be presented by an application feature, and metaphor of *reaching a landmark* – using application to execute the feature's functionality. Table 8.0.1 gives an overview of the features to be tested and the criteria of successful tests.

Table 8.0.1 Application landmarks and testing success criteria.

Landmark	Action	Success criteria
Email integration	Adding a comment using Email channel.	Comment is added to the task. Task is identified correctly using comments context. Comment is marked to be from source "Email".
Skype integration	Adding comment using Skype channel.	Comment is added to the task. Task is identified correctly using comments context. Comment is marked to be from source "Skype".
Business rules engine	Creating task using Skype channel.	Task is created. Project is identified correctly using comment's context.
	Updating task information using Skype channel.	Task information is updated. Task is identified correctly using comment's context.
Analytics	Checking the Sources chart before and after adding the comment.	Statistical information of Sources chart is updated after analytics information is expired.

8.1 Testing Environment

The testing is going to be held on a clean installation prefilled with the data as shown in Appendix III and the testing environment as presented in Appendix IV.

8.2 Email Integration

Email to be tested upon consists of the following information, as shown in table 8.2.1. The message, as it can been seen from the context, is expected to be added to *Collatool project -> Remove stopwords from analytics words chart*, presented on figure 8.2.2a

Table 8.2.1 Email integration test message

Field	Value
To	collatool@blueflex.eu
From	stanislav.kutasevits@gmail.com
Subject	Collatool stopwords
Body	The list of the stopwords to be removed is the following: "an, a, as, the, is, are, on ,in, into" the list is not complete, more words will come in the future

Message is sent at 17:21, and the cron processed it at 18:00 (cron is set to be run every hour).

After processing the message, message is successfully added to the task, as presented on the figure 8.2.2b.

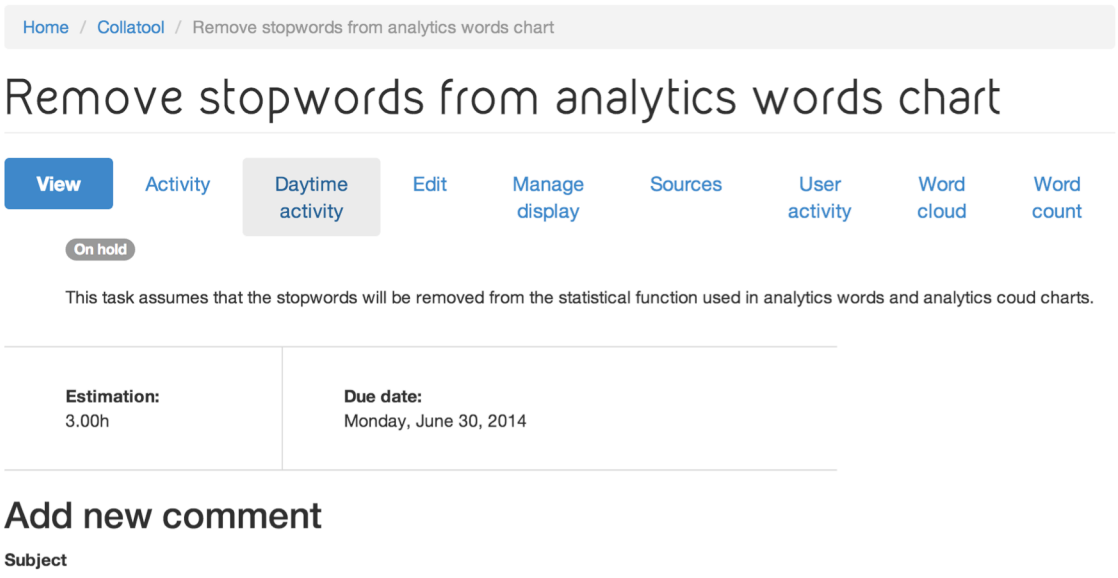


Figure 8.2.2a Task before adding the email message

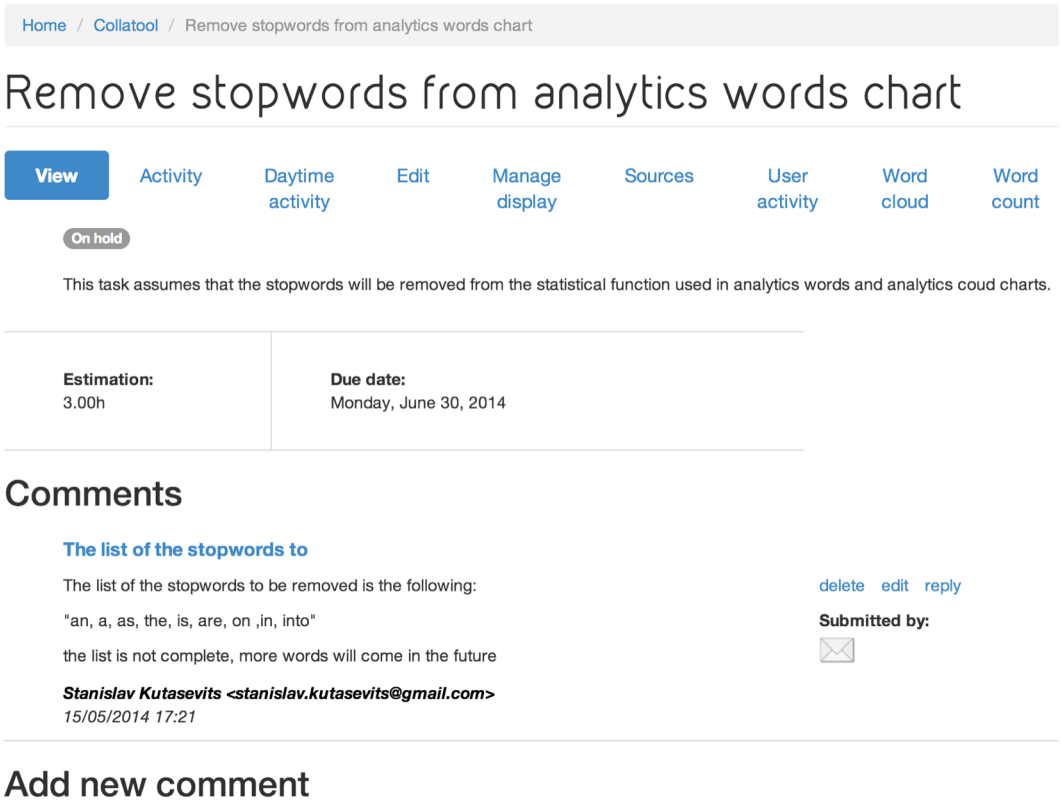


Figure 8.2.2b Task after adding the email message

8.3 Skype Integration

Skype message to be tested upon consists of the following information, as presented in table 8.3.1. Message is expected to be added to *Collatool project -> Remove stopwords from analytics words chart*.

Table 8.3.1 Skype integration test message

Field	Value
Message	here are a few additional stopwords that should be removed from collatool analytics words chart: "within, without, upon"

Message is sent at 18:53, and hence the is almost no delays in Skype message listener (default value to check for the messages is every 5 seconds), the message is added to the task, as shown on figure 8.3.2.

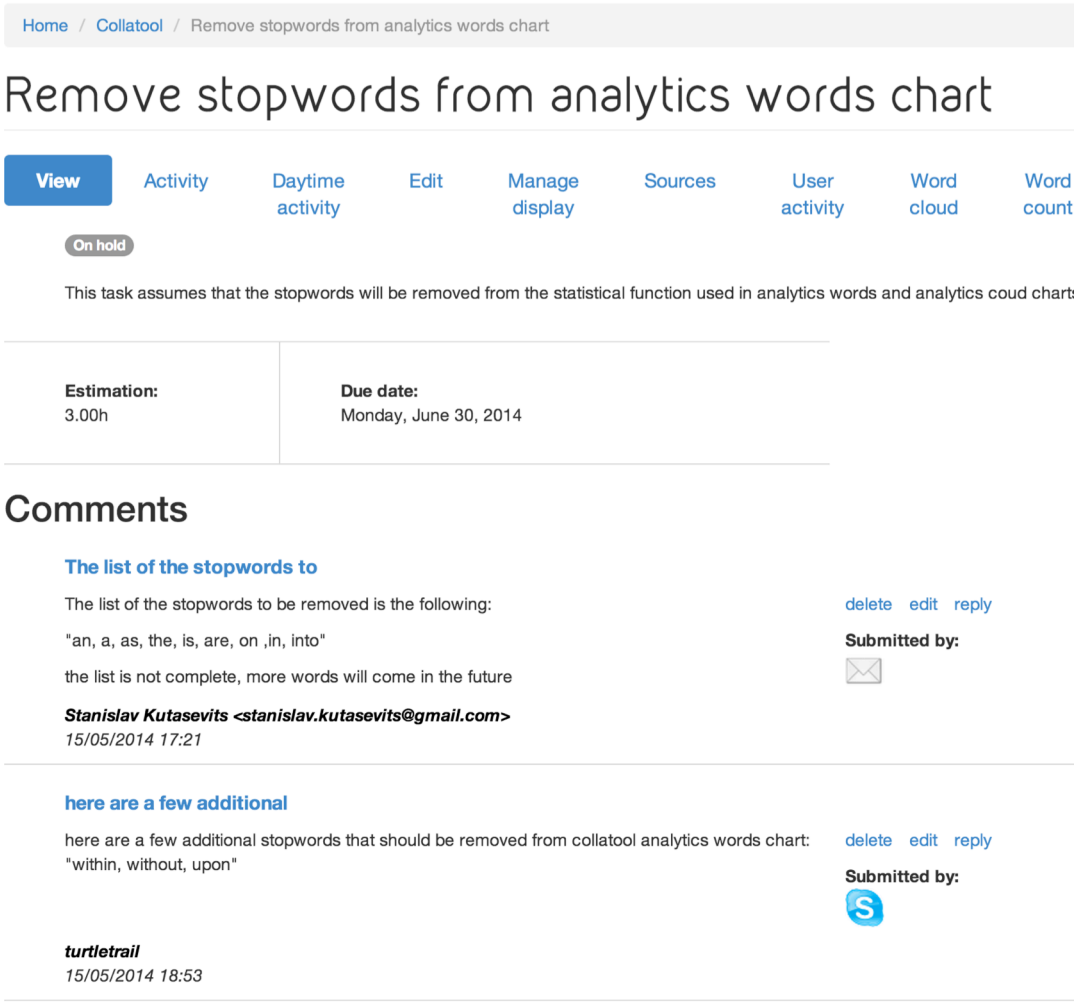


Figure 8.3.2 Task after adding the Skype message

8.4 Business Rules Engine

For testing the business rules engine Skype messages with the content as presented in table 8.4.1 are going to be sent.

Table 8.4.1 Business rules engine test messages.

Field	Value
Message	i suppose we should duplicate the task in collatool about removing stopwords from analytics word chart to word cloud chart as well. Create task with title as Remove stopwords from analytics cloud chart, with description as Check that the stopwords are removed from the analytics cloud chart, with estimate as 4 hours.
Message	I also suggest to immediately update the status of the collatool task about removing stopwords from analytics cloud chart. set task status as active.

First message added a new task to the project as shown on figure 8.4.2.

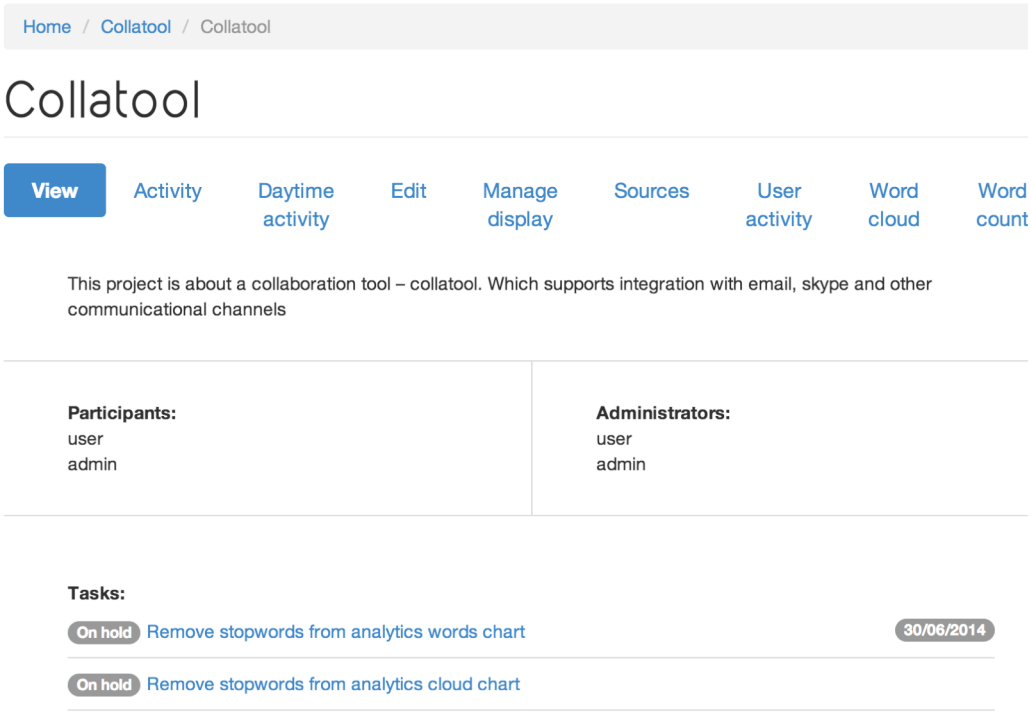


Figure 8.4.2 Collatool project after adding a task

Second message changes the status of the task to *active*, as shown on figure 8.4.3.

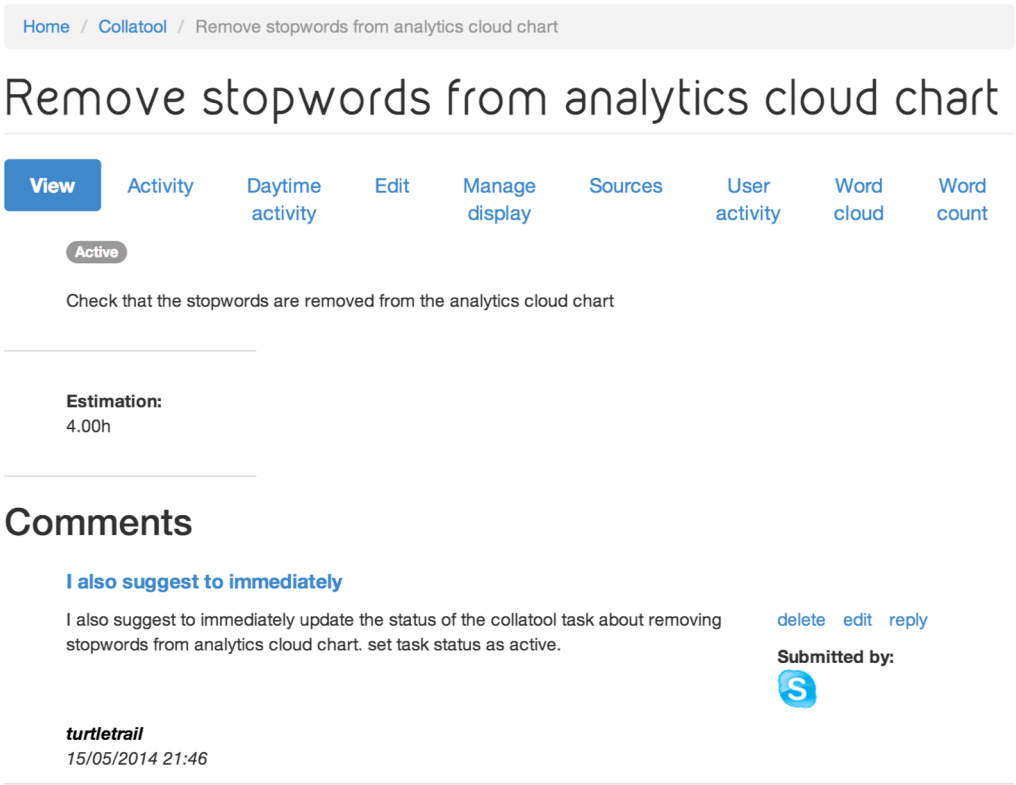


Figure 8.4.3 Task after adding a message

8.5 Analytics

Having the above-mentioned comments/messaged added through multiple channels the sources chart presents the following, as shown on figure 8.5.1.

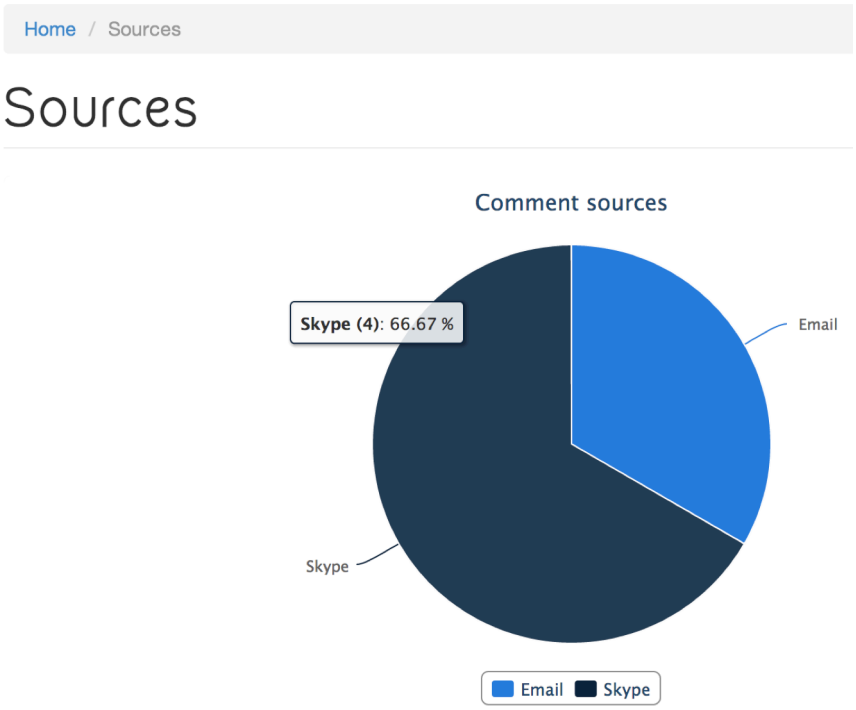


Figure 8.5.1 Sources chart, global level

For testing purposes a sample message is going to be added via Drupal comment form, as shown on figure 8.5.2.

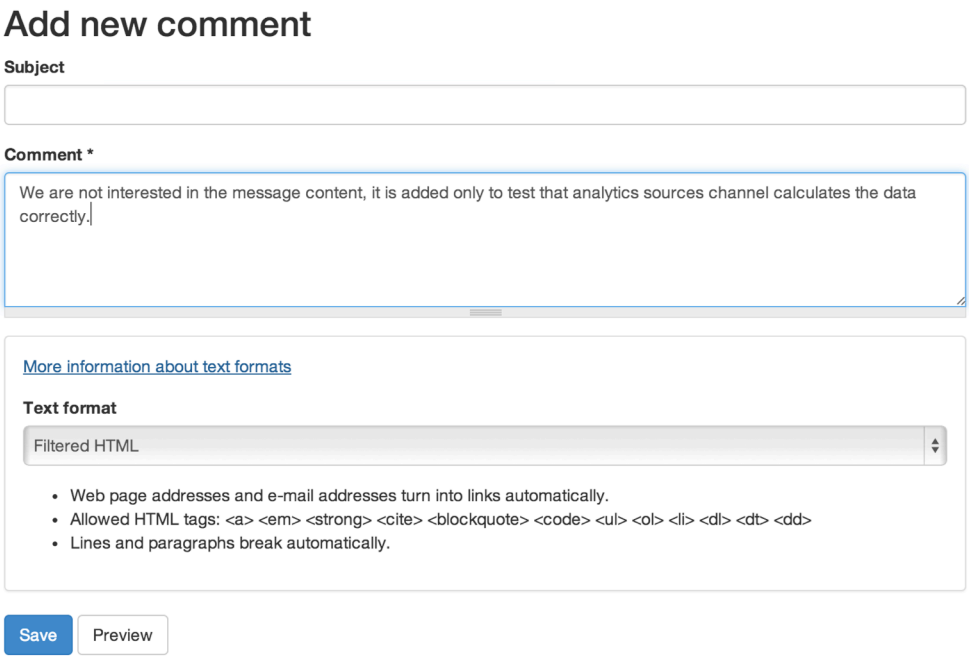


Figure 8.5.2 Sample message adding via Drupal comment form

After message is added and statistical information is recalculated via cron, the sources chart looks as presented on figure 8.5.3.

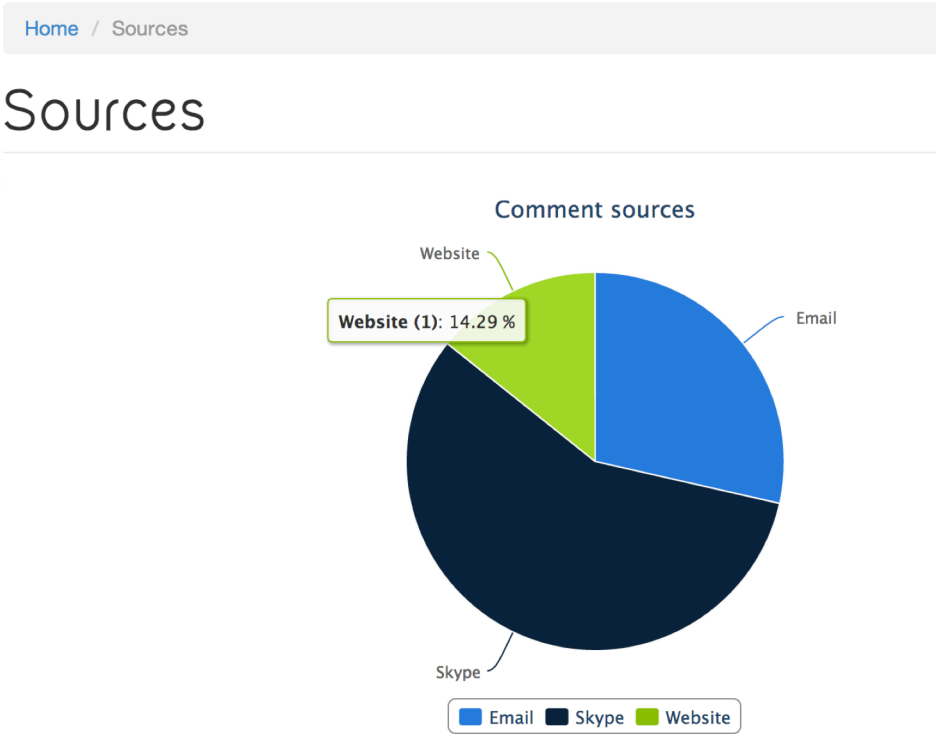


Figure 8.5.3 Sources chart after message adding, global level

8.6 Minimal System Requirements

The minimal system requirements are presented in the table 8.6 and are based on Drupal minimum system requirements[69], RabbitMQ system requirements[70] and Apache Solr system requirements[71].

Table 8.6 Minimal system requirements

Variable	Value
Disk space	500 MB
RAM	2 GB
Processor	1 GHz
PHP	PHP 5.2.5 or higher (5.3 recommended)
Database	MySQL 5.0.15 or higher with PDO, PostgreSQL 8.3 or higher with PDO, SQLite 3.3.7 or higher
Web server	Apache 2.0
Java	1.5

[69] https://drupal.org/requirements
[70] https://www.rabbitmq.com/configure.html
[71] http://wiki.apache.org/solr/FAQ#What_are_the_Requirements_for_running_a_Solr_server.3F

9 Conclusions

The main goal of the thesis was to examine distributed agile development, identify the core problems and suggest a solution to those problems. The problems of the distributed software development were defined as following: fragmented project documentation, multiple communication channels and limited analytics functionality. To solve this problem a new collaboration tool, Collatool, has been developed.

Developed tool tackles all of those three problems by keeping the project documentation (including the attached files) in a single place; allowing collecting the information from multiple communicational channels (Email and Skype) and thus keeping project documentation organised; providing non-trivial analytical data using the descriptive graphs.

As the application was developed as a prototype and by an effort of one-person team, it is not so much shaped to be used in the real life and might miss some functionality. Therefore it cannot compete with real-life collaboration tools, many of which were closely examined in section 3.3 Existing Solutions. Yet despite of the lack of functionality, the developed tool has a number of unique features, which make it standing out.

That, combined with the tackled and solved problems of the distributed agile development, allows considering the thesis together with the developed tool a success.

9.1 Thesis Contribution

Thesis makes a contribution to the field of system integration by presenting a real-life example of using messaging system to integrate multiple systems/platforms (Email, Skype). Besides that thesis justifies using the messaging system integration by giving and explanation of how other systems/platforms can be easily integration into such kind of architecture.

Additionally thesis makes a contribution to the field of business intelligence by showing how some trivial development process data can be used to answer some non-trivial questions, which give the project managers much more insight about the project flow, team coherence and mood.

Last but not the least thesis makes a general contribution to the software development field by demonstrating software development flow from the very beginning – business problem till the end – working software prototype, which solved that problem.

9.2 Further Work

Thesis touches several large scientific areas and there is a great potential for further work in each of them.

Software Development

Further work in software development field might involve enhancing the integration part by introducing more communication sources. That can be, for instance, issue register (GitHub issue reports), static web pages (project wiki pages), short message systems messages (SMS) etc.

In addition, the business rules engine might be made more intelligent by enhancing its flexibility (supporting more languages, less strictness of the command formatting etc.) and extending the variety of commands.

Analytics part can benefit by removing the stop words (“the”, “a”, “as”, etc.) from word count and word cloud features.

Business Intelligence

Further work in business intelligence might include gathering more data about the development team (e.g. developers age, gender, nationality etc.), project (complexity, duration, estimations etc.) and the rest, which would allow using more complex and sophisticated analytical algorithms to reveal the hidden information.

That information might potentially lead to a number of the significant conclusions, which not only make the developments process more efficient from business point of view but also consider the human factor to make the work as stress-free and comfortable for the people as possible.

9.3 Live Demo

A live demo of the application can be accessed by the following URL:

- http://193.84.27.159/collatool/public_html

using the following credentials:

- login: user
- password: collatool1234

The application supports the feature of integration to application general email (check is scheduled to run every hour):

- collatool.demo@gmail.com

Additionally application is listening to the incoming messages, which can be added using the enclosed CollatoolSkypeMessageListener application using the following information:

- Server: 193.84.27.159
- Port: 5672

10 References

[1] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, organizations, and process improvement," *Software, IEEE*, vol. 11, no. 4. pp. 36–45, 1994.

[2] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "Understanding and improving time usage in software development," *Trends Softw. Softw. Process*, vol. 5, pp. 1–25, 1995.

[3] D. Herbsleb, J.D. and Moitra, "Global Software Development," *Software, IEEE*, vol. 18, pp. 16–20, 2001.

[4] M. Korkala and P. Abrahamsson, "Communication in Distributed Agile Development: A Case Study," in *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, 2007, pp. 203–210.

[5] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Trans. Softw. Eng.*, vol. 29, 2003.

[6] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Distributed Agile Development: Using Scrum in a Large Project," *2008 IEEE Int. Conf. Glob. Softw. Eng.*, 2008.

[7] H. Holmstrom, E. O. Conchuir, P. J. Agerfalk, and B. Fitzgerald, "Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance," in *Global Software Engineering, 2006. ICGSE '06. International Conference on*, 2006, pp. 3–11.

[8] F. Abbattista, F. Calefato, D. Gendarmi, and F. Lanubile, "Incorporating social software into distributed agile development environments," *2008 23rd IEEE/ACM Int. Conf. Autom. Softw. Eng. - Work.*, 2008.

[9] X. L. X. Liu, L. L. L. Liao, Y. D. Y. Duan, and B. Y. Bin Yang, "Email information integration with SSO in portal service based on AJAX," *Comput. Appl. Syst. Model. (ICCASM), 2010 Int. Conf.*, vol. 12, 2010.

[10] A. Bacchelli, M. Lanza, and M. D'Ambros, "Miler: a toolset for exploring email data," *2011 33rd Int. Conf. Softw. Eng.*, pp. 1025–1027, 2011.

[11] M. Fischer, M. Pinzger, and H. Gall, "Populating a Release History Database from version control and bug tracking systems," *Int. Conf. Softw. Maintenance, 2003. ICSM 2003. Proceedings.*, 2003.

[12] C.-M. C. C.-M. Chung, Y.-H. W. Y.-H. Wang, G.-C. H. G.-C. Hsieh, W.-C. L. W.-C. Lin, and Y.-F. K. Y.-F. Kuo, "Tools cooperation in an integration environment by message-passing mechanism," *Proc. Eighteenth Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC 94)*, 1994.

[13] A. W. Brown, “Control integration through message-passing in a software development environment,” *Software Engineering Journal*, vol. 8, no. 3. pp. 121–131, 1993.

[14] C. M. Banner, “Understanding Unified Messaging,” *IT Prof.*, vol. 12, 2010.

[15] K. Schreiner, “Unified Messaging: Will It Finally Meet Its Promise?,” *IT Prof.*, vol. 9, 2007.

[16] J.-M. S. Wams and M. van Steen, “Unifying user-to-user messaging systems,” *IEEE Internet Comput.*, vol. 8, 2004.

[17] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. 2003, p. 736.

[18] F. Rosenberg and S. Dustdar, “Design and implementation of a service-oriented business rules broker,” *Seventh IEEE Int. Conf. E-Commerce Technol. Work.*, 2005.

[19] T. Patten and P. Jacobs, “Natural-language processing,” *IEEE Expert*, vol. 9, 1994.

[20] L. M. Fadel and M. C. Dyson, “Comparing a text- and visual-based interface presenting social information in an online environment,” *Vis. Lang. Human-Centric Comput.*, 2006.

[21] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, vol. 9. 1998, p. 639.

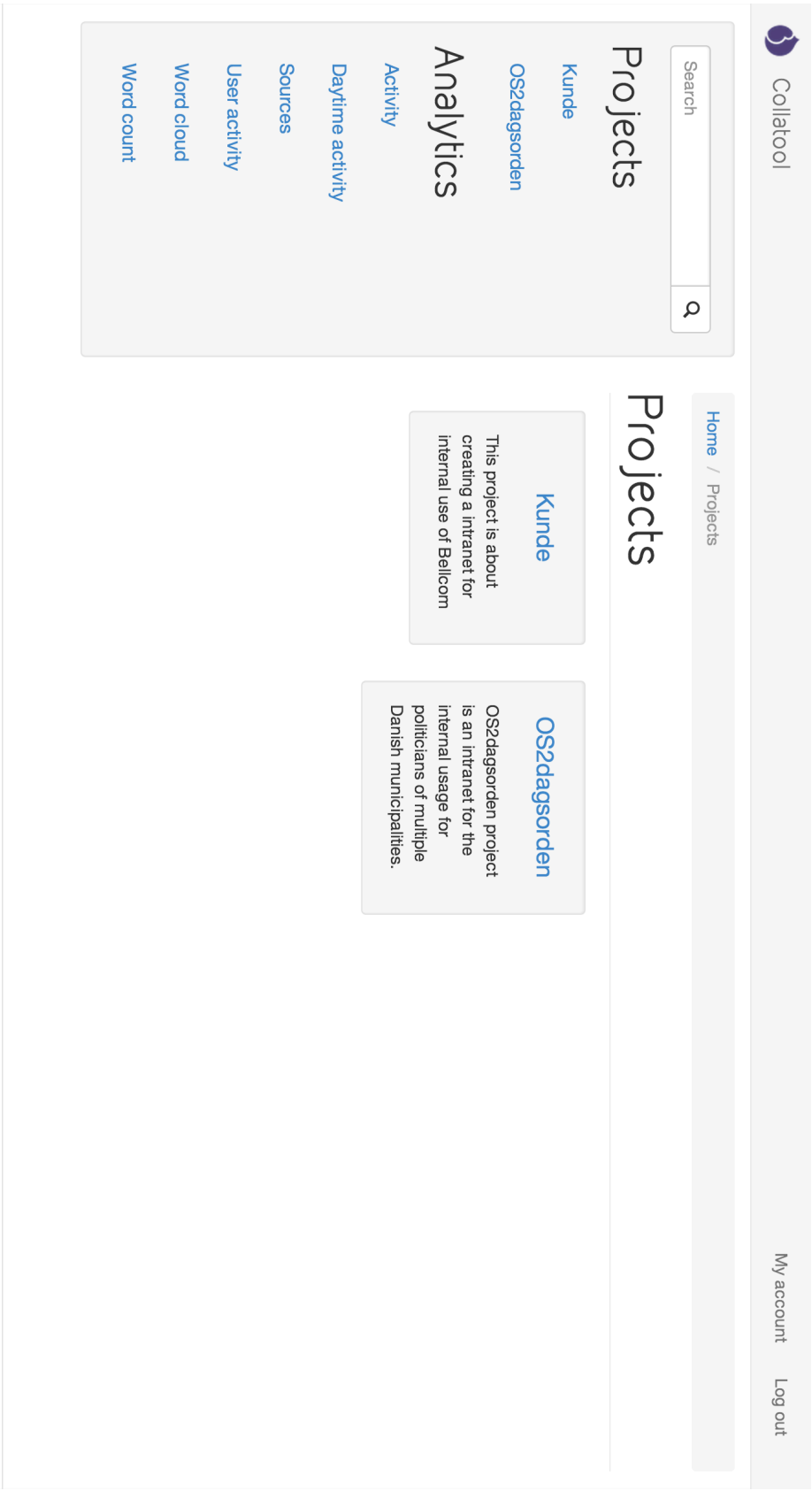
[22] B. Shneiderman, “Designing for Fun: How Can We Design User Interfaces to Be More Fun?,” *Interactions*, vol. 11, pp. 48–50, 2004.

[23] C. Kaner, J. Falk, and H. Q. Nguyen, *Testing Computer Software Second Edition*. Dreamtech Press, 2000.

[24] J. A. Whittaker, *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*. Pearson Education, 2009.

Appendix

I. Main Page Screenshot



II. Project Page Screenshot

III. Test Data

Field	Value
Project title	Collatool
Project description	This project is about a collaboration tool – collatool. Which supports integration with email, Skype and other communicational channels.
Task title	Remove stopwords from analytics words chart
Task description	This task assumes that the stopwords will be removed from the statistical function used in analytics words and analytics cloud charts.
Task Estimation	3h
Due date	30.06.2014
Status	On hold
Project title	CRM system
Project description	This system manages the relations with the customers allowing to quickly add, edit or change the information about the customer.
Task title	Add relation between customers
Task description	This task assumes creating the relation between customers, so that it can be seen which customers are related to one another.
Task Estimation	4h
Due date	29.06.2014
Status	Active

IV. Test Environment

Variable	Value
Operating system	OS X 10.9.3 2.7 GHz Intel Core i7 16 GB
PHP	5.2.24
Java	1.7.0.25
Apache	2.2.26
MySQL	5.6.13
Drupal	7.26
Apache Solr	3.6.2
RabbitMQ	3.1.5.

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Stanislav Kutasevitš** (date of birth: 14.07.1989),

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Communication-oriented Project Management Solution,

(title of thesis)

supervised by Siim Karus,

(supervisor's name)

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **26.05.2014**