TAMPERE UNIVERSITY OF TECHNOLOGY

VEERAKISHORE GODUGULURI

KommGame: A Reputation Environment for Teaching Open Source Software

Master of Science Thesis

Subject approved by Department Council on May 04, 2011

Supervisors: Adjunct Prof. Dr. Imed Hammouda (TUT)

Teaching Associate. Terhi Kilamo (TUT)

I. FOREWORD

This thesis work was done in 2010-2011 in the Department of Software Systems as a part of the OpenSE project - Open Educational Framework for Computer Science Software Engineering

I would like to thank Adjunct Prof. Dr. Imed Hammouda and Teaching Associate. Terhi Kilamo for considering me worth for this work and their continuous support and guiding during the thesis work.

I would also like to express my gratitude to the Department of Software Systems and my colleagues at the Department for providing the pleasant and joyful atmosphere which enabled the work on this thesis. I would also like to thank my friends who encouraged me to complete the thesis work.

Finally, I want to thank my family for their support during my seemingly endless studies.

Tampere, April 2010

Veerakishore Goduguluri
Orivedenkatu 8B 44
33720 TAMPERE
Tel: +358465956054
veerakishore.goduguluri@tut.fi

II. ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Faculty of Computing and Electrical Engineering
Department of Software Systems
Veerakishore Goduguluri: KommGame: A Reputation Environment for Teaching Open Source Software
Master of Science Thesis, 47 pages.
Examiner: Adjunct Prof. Dr. Imed Hammouda (TUT), Teaching Associate. Terhi Kilamo (TUT)
May 2011
Keywords: open source education, reputation systems, open source teaching, community model of education.

Over past several years the importance of teaching open source software in universities is increasing with the advent of open source as a development and business model. A novel, student centric approach of teaching open source was tried out at Tampere University of Technology where a new environment called KommGame was introduced to assist in teaching open source development. This environment includes a reputation system to motivate learners to participate. In this thesis, presents an approach of teaching open source and how the KommGame environment was employed to teach open source software.

The thesis is divided into three parts: background study, implementation and evaluation. In the background study part, issues related to open source software education, reputation systems and reputation model for teaching are discussed. Customization of reputation model for open source education is all considered. In the implementation part, different components of KommGame environment are investigated, architecture and implementation of each component is discussed. All possible use case scenarios are also covered in implementation part. In the evaluation part, details of KommGame environment, course setup are discussed and finally evaluation of KommGame environment is based on the open source course experience.

III. TABLE OF CONTENTS

I. Foreword ii
II. Abstract iii
III. Table of Contents iv
IV. Lists of Figures and tables vi
V. Terms and abbreviations vii
1 Introduction 1
2 Background 4
2.1 Open Source Software 4
2.2 Teaching Open Source Software 6
2.3 Reputation Systems 7
2.3.1 Models 8
2.4 Reputation Systems in Teaching 12
3 A Reputation Model for Teaching Open Source Software 13
3.1 Reputation Model 13
3.2 Karma Model 15
4 KommGame environment 17
4.1 Towards KommGame Environment 17
4.2 Features Required for KommGame 18
4.3 Architecture 20
4.4 Implementation 22
4.5 Using KommGame Environment 24
4.5.1 Registration to access the KommGame environment 25
4.5.2 Login to access the KommGame 25
4.5.3 Resetting login password 26
4.5.4 Reporting and viewing issues in bug tracking system 27
4.5.5 Commenting and Uploading Patch File 31
4.5.6 Closing or Updating an Issue in Bug Tracking System 32

4.5.7 Contribution Open Content to Wiki 32
4.5.8 Viewing the karma report 33
4.5.9 Viewing individual's profile 34
4.5.10 Writing content to personal blog 34
5 Case Study: Teaching OSS at TUT 36
5.1 Course setup 36
5.2 Course Activities 37
5.3 Feedback 38
5.4 Evaluation 40
6 Conclusions 43
References 45

IV. LISTS OF FIGURES AND TABLES

Figure 2-1 Hypothesized OSS development team structure. 6
Figure 2-2 An example of star rating model 9
Figure 2-3 Example of Favorites and Flags Model 9
Figure 2-4 Example of This-or-That voting model 10
Figure 2-5 Example of Points model 10
Figure 2-6 Example of Reviews model 11
Figure 2-7 Robust karma model [3] 11
Figure 4-1 An overview of KommGame structure 18
Figure 4-2 Learning environment Architecture. 21
Figure 4-3 Karma engine workflow 24
Figure 4-4 Registration page for KommGame 25
Figure 4-5 Login page for KommGame 26
Figure 4-6 Password reset 26
Figure 4-7 Interface to view issues. 28
Figure 4-8 Interface to report issues. 29
Figure 4-9: Interface to view issues. 29
Figure 4-10 Lifecycle of bug reported in bug tracking system 30
Figure 4-11 Commenting issue and Uploading patch file 31
Figure 4-12 Closing or updating an existing issue. 32
Figure 4-13 Add content to wiki and Favorites bookmark 33
Figure 4-14 Karma reporting interface. 33
Figure 4-15 Detailed user profile interface. 34
Figure 4-16 Writing personal blog 35

Table 2.1 List of reputation models 7
Table 3.1 List of activities that can contribute to participatory karma 14
Table 5.1 List of activities and corresponding weight functions 38
Table 5.2 Feedback questions and summary of response 39
Table 5.3 Statistics about the sample open source project 41

V. TERMS AND ABBREVIATIONS

OSS	Open Source Software a general term for software with source code publicly available and does not need any royalty for use.
SVN	A popular version control system Subversion. A system where source code is stored for different versions.
GUI	Graphical User Interface
COP	Community of Practice
SSO	Single sign on
OSI	Open Source Initiative

1 INTRODUCTION

> *"One of the questions I've always hated answering is how do people make money in open source. And I think that Caldera and Red Hat -- and there are a number of other Linux companies going public -- basically show that yes, you can actually make money in the open-source area." -Linus Torvalds*

With the advent of open source software (OSS) as a development and business model, the number of job vacancies valuing open source knowledge and experience has been rising on a regular basis. This in turn has motivated many universities and professional schools to introduce new courses and programmes related to teaching OSS principles and practices (e.g [1], [15]).

So far OSS teaching has mostly been organized in a traditional lecture course format, for example taking the form of a seminar where students present specific OSS related topics. Other attempts rely on sending students out into real open source projects and communities (e.g. [2]).

The approaches of teaching open source software mentioned above face two major challenges. First, classical teaching methods where most of the time the lecturer spends teaching the course and the students listen to the lectures. Furthermore students are made to work course assignments and they must work on the assignments in a deadline oriented way that is the students start working when deadline of assignment approaches. This method may not fully convey all the special aspects involved in OSS development, such as community collaboration, peer review, and co-creation.

Second, students may find it hard to participate in real OSS project as a first experience. This is because the development model of OSS is not as same as the commercial software, OSS projects typically have their own principles, practices, processes, and

tools; often students are not familiar with open source software development model principles, practices, processes, and tools.

The main research topics that this thesis would investigate are: teaching open source software in a controlled environment setting and using the reputation system for motivating open source education. This thesis defines a more attractive approach and building an environment which is similar to real OSS project infrastructure for teaching OSS. A more attractive approach is to provide a learning environment for OSS where students could collaborate collectively to achieve a common goal. This kind of approach is called constructive approach [21]. Constructivism is defined as philosophical position that views knowledge as the outcome of experience mediated by one's own prior knowledge and the experience of others. Constructivism seemingly fits in with, and supports, a range of multicultural, feminist and broadly reformist programmes in education. Although constructivism began as a theory of learning, it has progressively expanded its dominion, becoming a theory of teaching, a theory of education, a theory of the origin of ideas, and a theory of both personal knowledge and scientific knowledge. Indeed constructivism has become education's version of the 'grand unified theory' [25]. Such constructivist approach to learning allows students to generate new knowledge through the interaction of the group's past experience and new ideas. The community spirit and way of working could be maintained by organizing the learning tasks in a controlled collaborative setting that simulates real OSS projects.

A constructivist learning method however needs individual's active participation, which from the OSS perspective means student contribution to the community. An important question is, therefore, how to keep students' motivation high for the purpose of learning OSS concepts through active contribution. It has been argued that reputation systems could play an important role in maintaining student motivation [9]. Reputation systems are used to compute and publish members' contribution in a community.

This M.Sc thesis argues that reputation systems can be applied to build a learning environment for open source software. The approach is also inspired by the experiences of using reputation systems to reward and recognize developers in OSS communities such as Qt [10]. Towards this aim, we present an example reputation model and a concrete reputation environment known as KommGame that mimics real open source projects. The environment has successfully been tested at Tampere University of

Technology (TUT) to introduce OSS concepts and practices to software engineering students.

This M.Sc thesis is structured as follows. Chapter 2 discusses the theoretical bases of the thesis, including teaching open source and reputation systems. Chapter 3 studies the methods and concepts used for teaching open source software and provide brief overview of how reputation model can be used for teaching open source software. Chapter 4 presents the implemented solution KommGame environment. In Chapter 5 discussion is placed regarding how the implemented environment met the aims of the work. In Chapter 6 draws conclusions of the thesis.

2 BACKGROUND

The aim of this chapter is to discuss and familiarize the readers with the terms, concepts and techniques used in this thesis. This chapter provides a basis to proceed further in understanding the work.

2.1 Open Source Software

OSS [22] can be defined as computer software for which the source code is made freely available to anyone to view, modify and distribute under open source definition compliant license, as articulated under the Open Source Initiative [OSI]. Any software to be OSS its license should comply with certain criteria [23] along with access to source code.

First is the license of the software must not restrict of charging fee for giving away the software as a component of an aggregate software distribution containing programs from several different sources. The source code must be distributed with initial work and all other derived works.

Second is the license it must not restrict the derived work and must allow them to be distributed under the same terms as the license of the original software. The license should protect the integrity of author’s source code; the license may restrict the distribution of modified work only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time.

Third is the license that should not restrict any person or a group for using the software. License must not contaminate other software and must not be specific to product. The license must not restrict anyone from making use of the program in a specific field of endeavor.

Fourth is the rights that are attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Social Structure of Open Source Development Team

Structure of the open source development model is one of the important factor which makes the development model different from proprietary software development. Understanding the basic structure of the open source development model helps understanding team practices. Several authors have described OSS development teams as having a hierarchical or onion-like structure [26] as shown in Figure 2-1. The team at the center of the model is developers, they are the publishing entities of OSS project they take care of software infrastructure and the development process of OSS project. They contribute most of the code and oversee the design and evolution of the project. A study [27] on some OSS projects found that only about 15 developers contributed more than 80% of the code for new functionality. Since they contribute most of the code, they deiced about the licenses that apply to OSS project. From the study, we can see that core team of OSS project is small and they exhibit a high level of interaction, which would be difficult to maintain if the core group were large. The publishing entity defines the process to use infrastructure for granting access to the open source project including access to source code, contributing to the project (patches, wish lists, bug reports, bug fixes, documentation, and developer support) and reaching other community members. In the next level of hierarchy are the developers, Observers, and industrial partners. The team at this level mostly deals with the bug fixing in the project, however the fixed bugs should be reviewed by the rest core developer team before the bug accepted as fixed. For example, during much of the development of Linux, Linus Torvalds personally reviewed and decided on all code submissions. Usually this level of team is much larger compared to code developer team. Their level of interaction is also much lower.

Some open source projects are associated with industrial partners; those partners might get involved in the community process. Industrial partners are typically two types: enthusiastic and conservative. Enthusiastic partners participate with developers to contribute to the software and stay close to its evolution. On the other hand conservative partners are reluctant for the software being open source as this may change their mode

of operations and business. This kind of partners has their own members in the community to observe the evolution of the software and the outcome of the community.

The other important element in open source development team is the existing open source communities and other individuals. This group consists of mostly non-programmers. A subset of this group are active in the open source project, they use the latest release of software and contribute bugs or feature requests. However, the majority of this group are passive users; they just use the software, without making any contribution to the project.

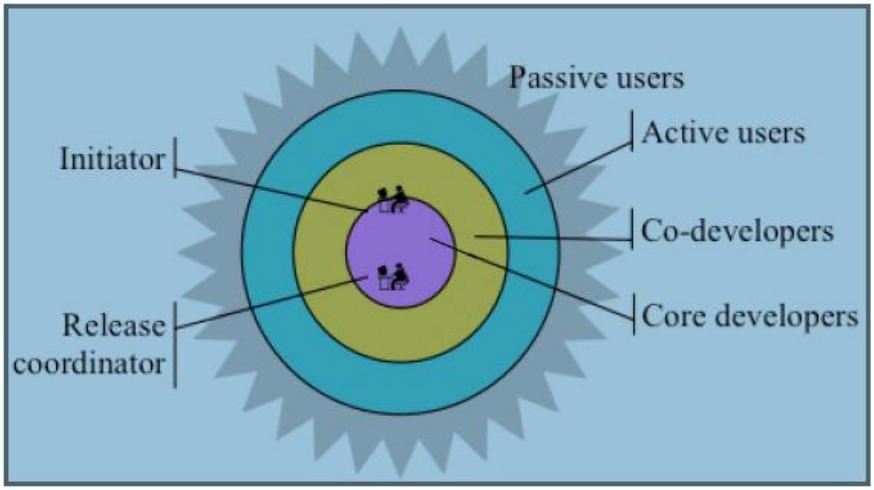

Figure 2-1 Hypothesized OSS development team structure [26].

2.2 Teaching Open Source Software

In modern pedagogical approaches, the learner plays a center role. The learner should continuously be motivated to build a new knowledge based on previous experience. The role of the instructor is to assist the learner through the learning process by applying the right teaching methods and by providing an appropriate learning environment. In the OSS setting, such learning environment is built around community dynamics. As a matter of fact, we can observe that motivation is the common driving force of modern pedagogical theories and OSS development.

The need for teaching OSS has been emphasized in the recent past [18], taking different forms. Some universities have tried out teaching software engineering using OSS [4] [15], which helped students learn basic and advanced software engineering topics. In

[13] open source evolution is taught in software engineering courses to give students a more realistic experience. Some universities have organized international masters programmes in open source software [1], devoted for providing students with an integral education on the different aspects of OSS. On a similar note Google and The Finnish Center for Open Source Solutions (COSS) have conducted summer code camps to encourage student's participation in real life open source projects. There are some communities like Teaching Open Source [7] and OpenSE [8] dedicated for promoting and researching open source teaching.

All of the above mentioned methods of learning OSS require participation in a real open source projects, which requires certain kinds of skills and practice to practices. Real OSS development involves different open source terminology like committer, contributor, bug report, feature request, version control, etc. A novice learner does not have prior knowledge of practical issues in OSS development. Open source education should convey these practical issues to individuals before encouraging them to participate in a real open source project. The learning environment should motivate students to make more contributions; this can for example be achieved by using a reputation system.

2.3 Reputation Systems

Reputation systems are used to measure the contribution of individuals in an online community. They are also applied in different fields such as e-commerce, search engines, and social news.

The reputation for any entity can be decided in several ways like writing feedback about the entity, rating the entity with some points, marking the entity as favorite and promoting the entity by voting. All the above mentioned models are simple. However, simple does not mean that they are not useful or easy to break. We can build a more robust reputation model by taking an aggregation or a different combination of such simple models of reputation. In [3] Farmer has explained about different reputation models, Table 2.1; shows the list of reputation models and description of their reputation values.

Table 2.1 List of reputation models

Model	Reputation value
Rating	The average of all rating that the entity has got.
Favorites and Flags	The number of bookmarks received.
This-or-That Voting	The number of votes to an item. Voting a particular item within a bounded set of possibilities.
Points	The sum of point for different actions user are engaged in.
Reviews	The number of normal ratings or freeform text comments.
Karma	The aggregate of all user activities. All the user activities have corresponding reputation.

2.3.1 Models

In this section all the reputation models are explained in detail along with examples.

Rating model

A Rating model is used when a participant has to rate the entity between a range of values. There are different types of rating models that are in use, examples are points scale rating, star rating, or "HotOrNot". This model is very easy for the users to provide the reputation feedback, for example Figure 2-2 shows an example of how uses this kind of a reputation model for rating movies.

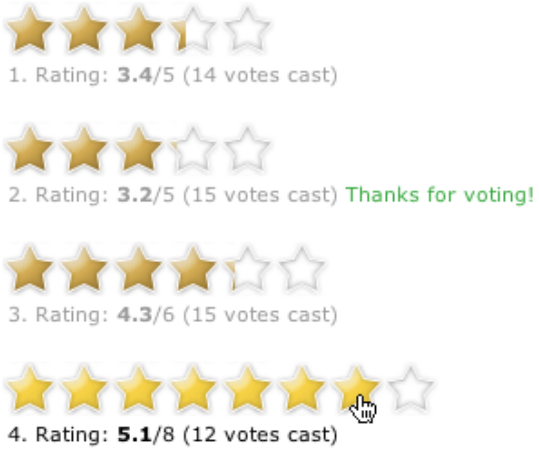

Figure 2-2 An example of star rating model

Favorites and Flags Model

Favorites and Flags model gives a control to the community for identifying a particular entity of exceptionally high or low quality. Favorites and Flags model has three variants Vote to promote, Favorite, and Report Abuse. Vote to promote model is used when user has to vote to a particular item to promote it from a pool of items on the poll. Figure 2-3 shows an example of this model. A Favorite model is used to keep track of the number of times a particular entity is bookmarked as favorite, thus increasing the reputation of the entity. Report Abuse is a negative reputation model. This model is used to avoid bad content in the community. In this model total reputation is the number of times particular item is flagged as abuse.

Figure 2-3 Example of Favorites and Flags Model

This-or-That Voting Model

This-or-That Voting model is used when the community members have to choose best option within a bounded set of options available. An example of this type of model is in communities when someone answers a question, any member of the community can vote the answer as “was the answer helpful?”. Figure 2-3 shows an example of this model.

Figure 2-4 Example of This-or-That voting model

Points Model

Point's model is used when we want a very specific value of user activity on something. When a user is engaged in various actions, these actions are recorded, weighted and summed to calculate total reputation. Figure 2-5 shows an example of this model.

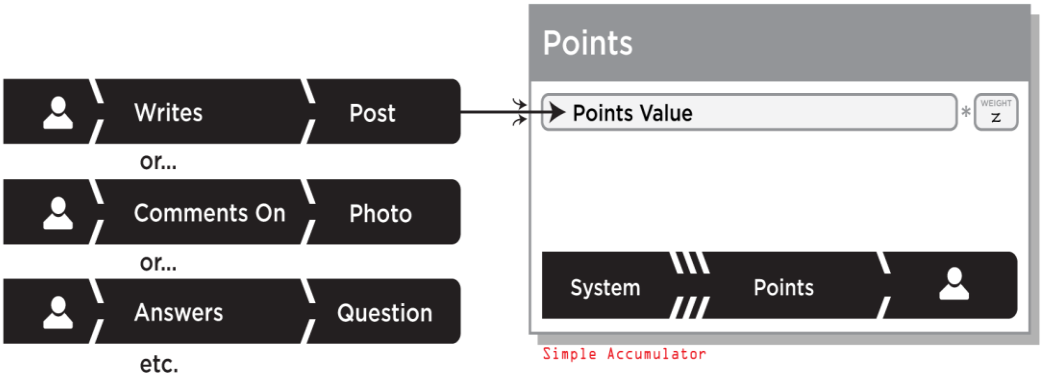

Figure 2-5 Example of Points model [3]

Reviews Model

In the reviews model the reputation is a combination of series of rating models or freeform text comment on a particular entity. For example in *www.amazon.com* users can write review comments about the product they bought. The reputation increases or decreases based on the review comments. Figure 2-6 shows an example of this model.

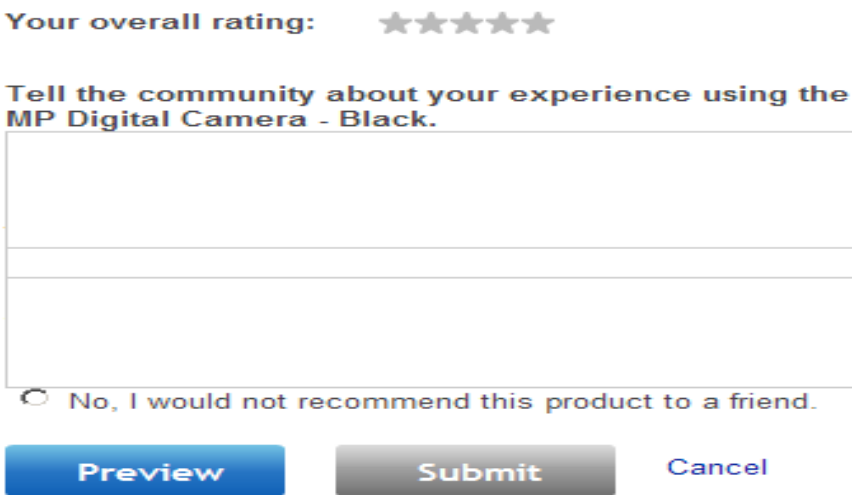

Figure 2-6 Example of Reviews model

Karma Model

The word Karma refers to the results humans get because of their past actions. A Karma model of reputation is particularly used when the entity to which is the reputation subjected is a human. Karma model is generated using combination of the above mentioned reputation models.

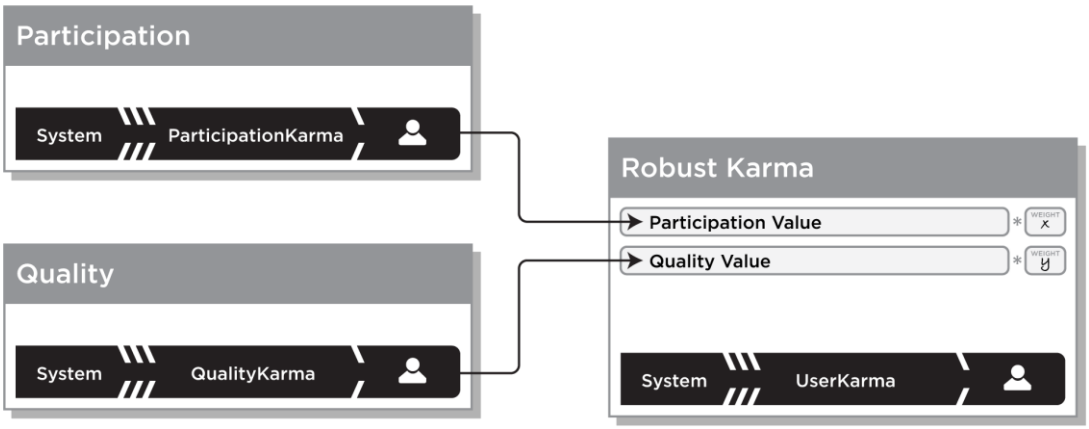

Figure 2-7 Robust karma model [3]

There are two primitive forms of karma models: participatory and quality. Participatory karma is used to represents the amount of user's contribution. Quality karma is used represents the quality of user contribution. An example of both participatory and quality karma is when karma is given to reviews written by a user then participatory karma is about the number of reviews that user has written and quality karma is about the content

of the reviews not the count of reviews. When a model contains both the participatory karma and quality karma then it is called robust karma model. The Figure 2-7 gives visual illustration of robust karma generated by combining participatory karma and quality karma models. Using a robust karma model gives highest score to the most active and best quality contributor.

2.4 Reputation Systems in Teaching

As reputation systems are applied for measuring online activities one can see that reputation systems can be applied for e-learning in the educational context where most of the activities happen online. In the web-2.0 era internet users are familiar with reputation and are used to do different kinds of online activities for instance writing reviews about something or bookmarking something as favorite which give them reputation. E-bay is the best e-commerce website where people do activities like writing reviews about the products that they bought and bookmarking some products as favorite.

Reputation systems have been a popular method of motivating the participants of online communities. As only a few open source communities have adopted the reputation systems for motivating developers, more research needs to be done regarding to using reputation systems in open source communities. But it is discussed in [9] that reputation systems suites a small group of young participants; they have high competitive sprit (spirit maybe) which makes learning more active and motivated.

A Reputation system can be applied to Community of Practice (CoP) [12]. CoP is similar to collaborative learning where a group of people with common interest who want to share and gain knowledge from other group members. In [16] it is showed that reputation system can be used in the context of e-learning. A web based reputation system called SocialX [17] was designed to support collaboration and social aspects of e-learning. SocialX lets the students to share and exchange of solutions, to discuss solutions of exercises through a forum, and to participate in project activities. The previous research results supports that the use of reputation systems in education.

3 A REPUTATION MODEL FOR TEACHING OPEN SOURCE SOFTWARE

In this chapter the reputation model that suites for open source education and Quantities of reputation model are discussed.

3.1 Reputation Model

In OSS development, most of the activities are done online. OSS development requires an infrastructure where the community can perform different project related activities. Example infrastructure components include a bug tracking system, a wiki system, a discussion forum, a version controlling system, an irc channel and a mailing list. The most common OSS community activities are based on bugs, features, improvements, wiki pages, a code repository, a forum, an irc channel and a mailing list.

A reputation model for open source education should be designed so that most common activities of open source communities such as writing new source code, testing of existing source code, reporting bugs in source code, documenting the source code, and adding content to wiki are taken into consideration to give the participants a feel of real open source communities. In OSS development, all kinds of contribution are treated as equally important and there is no good metric with which to compare or quantify different types of contribution with each other. This is the reason why most of the open source communities have not adopted a reputation system. In an educational context, however, the course moderator may decide which types of contribution should be emphasized. A reputation model can be designed accordingly.

This thesis argues that the karma reputation model fits well the activities and the nature of OSS communities, where the object subjected to reputation is human. Both forms of karma model (i.e. participatory and quality) can be used to measure both the amount and quality of contributions.

All kinds of online activities can contribute to the reputation of an entity. Activities of any OSS project that can contribute to participatory karma values are listed in Table 3.1.

Table 3.1 List of activities that can contribute to participatory karma

Category	Activity
Bug	Reporting new bug
	Commenting on bugs
	Closing bug
Feature	Request new feature
	Commenting on feature request
	Closing new feature
Improvement	Request Improvement
	Commenting on Improvement
	Closing Improvement
Wiki pages	Creating/Editing wiki pages
Code repository	Apply a patch
	Add a new code file
	Removing a code file
Forum	Starting new discussion
	Commenting discussion
IRC	Communicating through IRC
Mailing List	Send email to mailing list

In the process of contributing to OSS any community member may find a bug in the software. Then the bug has to be reported in the bug tracking system. Community members can comment on the bug and close the bug. The feature request and the improvement request are also handled in the same manner as bugs. Any community member can add or edit open content on the wiki pages, add code or apply patches to the

source code in the code repository, participate in forum discussion and communicate to other community members through IRC channel and mailing list. All these activities contribute to the participatory part of karma value. The contents of the wiki pages can be bookmarked as "like" by any community member, if they like the page contents. The number of bookmarks that any wiki page receives will contribute to the quality part of the karma value of the author of the wiki page. At regular intervals of time, best quality contributor will be selected by the community. The best quality contributor is given a hat, a hat is considered as token for best quality contribution. The number of hats any community member receives contributes to their quality karma.

3.2 Karma Model

The karma model we suggest is composed of different kinds of contributions. Care is taken that most common contributions are covered by the karma model. Each contribution is given a particular weight and the course moderator decides the weights for different contributions in the community. The final karma value of the participants is the sum of weight times of each contribution. The universal karma model can be written as in equation 3.1

$$Karma = \sum_{k=1}^{n} (f_k(contribution_k)) + f(Favorites) + g(Weekly\ Quality\ Tokens) \quad (3.1)$$

Here n corresponds to the total number of contributions. f_k is the weight function corresponding to contribution type. "Favorites" is the number of like bookmarks a content author gets. "Weekly Quality Tokens" corresponds to the number of time the particular participant was selected as the best quality contributor of the week by the rest of the members of community. The karma model in Eq. no. 3.1 is composed of two types of karma, participatory karma and quality karma. The sum of all contributions gives the participatory karma. The quality karma is composed of two parts, favorites, and weekly quality tokens.

For example, a sample robust karma model which covers activities related to bugs, features, improvements, wiki and quality contribution is given below. A similar model was used in Meamo community [11]. In the formula each activities is multiplied with its associated weight. Total karma is sum of all karmas from each individual activity.

Karma= 6√(number of bugs reported) + 3*√(number of bug comments) + 2*(number of bugs closed) + 4*√(number of feature requests) + 3*√(number of bug comments) + 2* (number of closed new features) + 4*√(number of request) + 3*√(number of improvement comments) + 2*√(number of closed improvements) + 4*√(number of edits) + 4*√(number of likes) + 4*√(number of weekly quality tokens)* (3.2)

There is mapping between equation no. 3.1 and equation no.3.2. The sum of all the contributions that belongs to bugs, features, improvements and wiki edits is the participatory part of total karma, this part of 3.2 corresponds to sum of contributions (sigma part) in equation 3.1. In equation 3.1 quality karma has two parts one is *f(Favorites)* and the other part is *g(weekly quality tokens*). In equation no. 3.2 *4*√(number of likes)* corresponds to *f(Favorites)* in equation no. 3.1 and the number of weekly quality tokens this to *g(weekly quality tokens).*

The weight function of each contribution is chosen based on the priority of contribution, in this example bug reports is given the highest priority with weight function *4*√*. And the lowest priority is given to closing the bugs, new feature, and improvement requests with weight function *2*√*.

4 KOMMGAME ENVIRONMENT

In this chapter there is discussion about approach towards KommGame environment, an overview of components in KommGame environment, how they are implemented and some example scenarios how they are used.

4.1 Towards KommGame Environment

With the environment the aim is to construct an infrastructure with which students can practice OSS projects. It is very common that people who participate in OSS projects are self-motivate to contribute to the project. But when it comes to OSS education all the students who participate in the course may not be self-motivated, so the environment should be able to create motivation among the students without any damage to the principles of OSS. To create motivation among the students who participate in the course we choose a reputation system integrated with infrastructure.

A brief overview how the KommGame environment looks like is shown in Figure 4-1. There two databases in the system one is store the details of the users who participate in KommGame and the other is to store user contribution data. The karma engine is the heart of the KommGame environment; it reads the contributions of students from contribution-collector and creates display-ready statistics of student karma from the fresh data came from contribution-collector. The arrows indicate the flowing of the user contribution data, except for in the case of the student, for student it shows that student contribution in OSS project infrastructure, student requests reputation statistics through GUI and gets the visual data back from the display chart. The contribution collector requests and receives the data from user database and contribution database and creates association between user data and contribution data so that every user gets his corresponding contribution details.

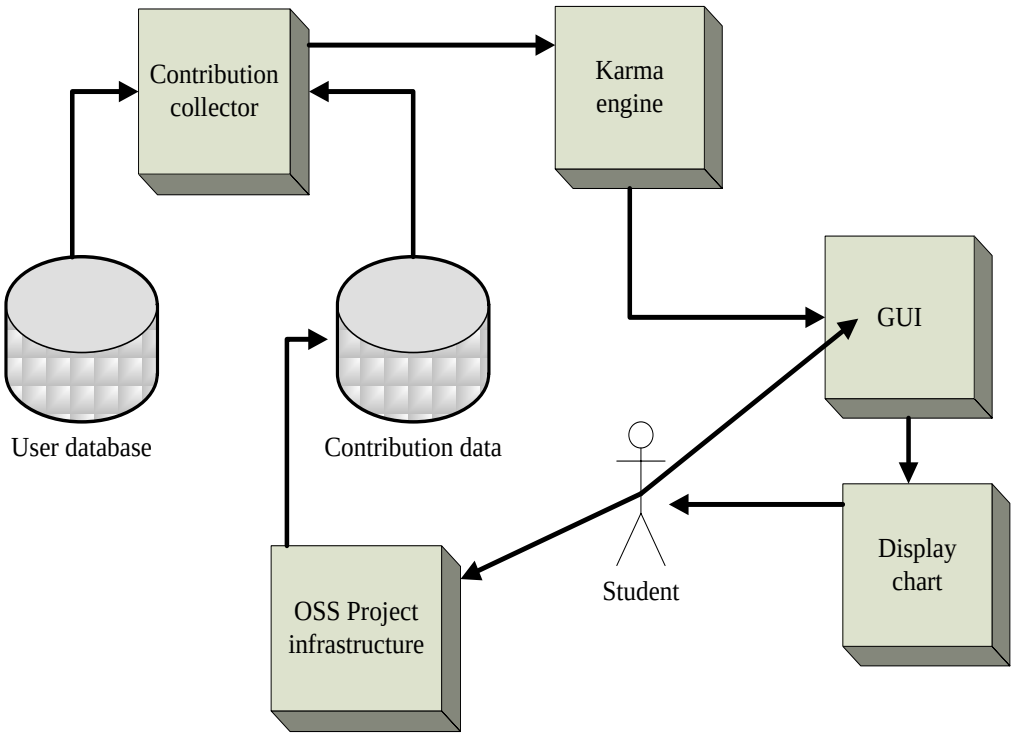

Figure 4-1 An overview of KommGame structure

4.2 Features Required for KommGame

We have developed an OSS learning environment based on the reputation model presented earlier. The learning environment, called KommGame [20], maintains karma values as a motivational factor for a community of learners. The KommGame environment forms an infrastructure required for collaborative and student centric learning.

The KommGame infrastructure has been developed to mimic the infrastructure of a real open source community. Real open source communities have following feature: Content management system, Bug tracker, User manager, IRC, Mailing list, Code repository, and Blogging system.

Content Management System

This is a place where the community members can contribute open content about the project. Wiki assists the users to edit text easier and encourage community members to contribute and annotate text content for a project.

Bug Tracker

Bug tracker is usually a web interface accessible to all the community members, where the users can report the bug they found, request new feature they wish to have in the project or any request improvement of any feature. Bug tracker is a important channel of communication between the community members. When a user submits any request lot discussion occurs between the users who are interested in the reported request.

User Manager

The name of the component is self explanatory. This component is responsible for performing create, read, update and delete operations on user profiles of the community.

Communication Channels

Communication channels like Internet Relay Chat and mailing list are required by KommGame environment. IRC is one of the popular methods of communication in open source projects. Open source projects often have designated people to moderate IRC. IRC is a great avenue for asynchronous communication. Usually conversation on IRC is not saved but some project log the conversation for future reference.Mailing lists are used in addition or replacement to IRC. Open source code hosting sites like GoogleCode, SourceForge and Lanchpad offer mailing list for the projects they host. Mailing list is a great tool for asynchronous communication.

Code repository

This component is the code base where the project is hosted. They act as revision control system for the project. GoogleCode, SourceForge and Lanchpad are some famous open source hosting.

Blogging system

Blogs are usually used by the community member to update their activity in the project to the rest of the community. Blogs act as a gateway for social interaction with community members.

Along with above mentioned components KommGame environment also needs a reputation system for to calculate the karma of each community member and an user interface to publish karma values.

4.3 Architecture

From the architectural point of view the learning environment can be divided into the following modules:

- Content management module
- Blogging module
- Bug reputation system
- Karma Engine
- User management module
- Version controller

Figure 4-2 shows the architectural design of the learning environment.

Content management module is responsible for assisting the community members to contribute open content with other community members. Content management system has a web interface; users can add content through the web interface. Content management keeps track of all the activities done on any page.

Blogging module is responsible for maintaining the blogs created by the community users. Blogs are more like personal web pages. Blogging module provides each corresponding user with their blog page where the user has permission to update the blog. And all the content that is updated in the blog page is publicly made available to all the community members.

Bug reporting system is responsible for managing the bug, new feature and improvement requests reported by community users. All the requests reported by any community member are publicly visible to every other community member. Once a request is reported in the bug reporting system this request is handled by any other member who is interested in it and a lot of discussion happen before any request is closed.

Karma engine is responsible to calculate the user contribution in the community. As explained in chapter 4 the contribution that should be considered for final karma based on the focus of activity is decided by the course moderator. All the karma values calculated are post on public web page accessible all the community members.

User management module is responsible for performing: create, read, update and delete operations on user profiles of the community. User management module provides all the details of the contribution done by any community member.

Version controlling system is responsible for revision management of the source code.

All the modules share the common learner database. The Karma module interacts with all other modules to collect the learner contribution information and calculates the user karma based on learner contribution and makes the karma value publicly visible to every other learner.

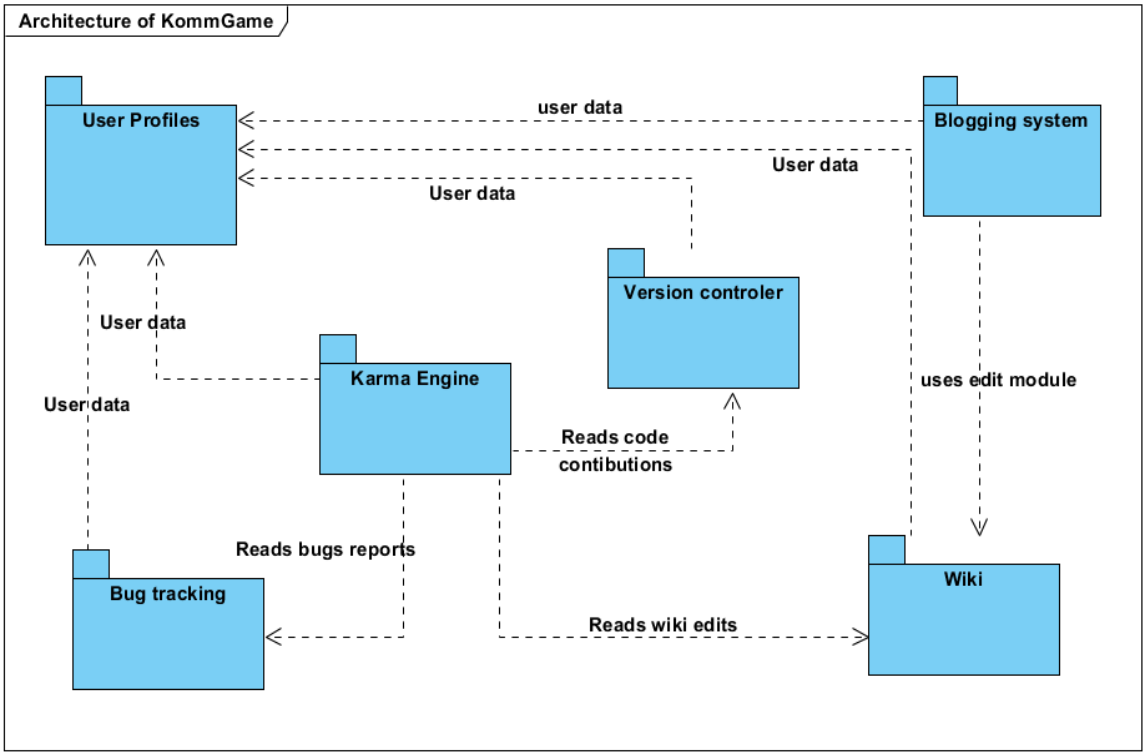

Figure 4-2 Learning environment Architecture.

The architectural diagram in Figure 4-2 shows how the each individual component in the infrastructure depends on other components. These dependencies are shown using dashed line with arrow head. The component on the arrow head side is the component on

which the component on the other side of the line depends on. The Karma engine is at the heart of the whole learning environment. It interacts with all other components to calculate the total karma of each participant

The dashed lines with arrow heads represent uses relationships between components. For example, the Karma engine depends on Bug tracking, wiki, version controller, and user profile to read the user contribution and calculate total karma. The blogging system uses the services of Wiki system.

4.4 Implementation

The environment is accessed through web browser; therefore it web application and is should be platform independent so we have chosen PHP for development of the whole environment and MySql as backend database since they are open source and I am more familiar with PHP and MySql.

Content management system is implanted using Dokuwiki [6]. Dokuwiki is a open source software aimed at small companies documentation needs. It is developed using PHP scripting language and licenced under GPL 2. It works on plain text files and does not require any database. Dokuwiki has its own user management system, but this user management is replaced with the user management of the KommGame environment to support single-sign-on(SSO) feature.

Blogging system is implemented using PHP. It reuses the editing module from Dokuwiki. Blogging system uses plain text files to store the user edited content on the blog. The aim of this module is to provide users a simple interface to maintain and track their activities on KommGame.

Bug tracking system is implemented using Mantis [5]. It is a free and open source web-based bug tracking system. It is developed using PHP scripting language and licensed under GPL 2. It can use MySql, MS SQL, and PostgreSQL databases to store bug details. We have used MySql for database since its open source and I am more familiar with it. Mantis has its own user management system, but this user management is replaced with the user management of the KommGame environment to support SSO feature.

Mantis and Dokuwiki because they can be integrated together and support SSO. The wiki system and the bug tracking system are deployed over the Apache web server. They share a common user management system.

The User profile component is the user management system of KommGame environment. It is responsible for User registration to access KommGame environment, Single sign-on to KommGame environment, maintaining the participant's data. User profile system is developed using PHP, it uses MySQL for database. The aim of this module is make have central user management and let users to use same credentials for sub modules of KommGame environment.

Version control system used is Subversion. Subversion is often abbreviated as SVN, after the command name svn. It is widely used by open source communities for example Apache Software Foundation, PHP, SourceForge and Python. Reasons for choosing subversion are, it is open source and it is easy to use.

Mantis and Dokuwiki have their own graphical user interfaces (GUI). A GUI module is designed for displaying statistics of karma. GUI module is implemented using HTML, CSS, and PHP. The statistics of karma are displayed as graph, the graph is drawn using pChart [24] open source library. The statistics are display as histogram with karma values on y-axis and user-ids on x-axis. The aim was to let users easily understand the graph and view karma values.

Karma engine is implemented using PHP. Implementation based on the workflow shown in Figure 4-3, whenever user request for karma value first karma engine is initialized; after karma engine is initialized it retrieves user contributions by interacting with wiki system, bug tracking system and version control system. After reading the contributions karma model is applied on the contributions to calculate the latest karma values of all users. GUI module receives all the karma values and it prepares a histogram graph to display karma statistics. |The aim was to make latest karma values whenever a user accesses the karma values.

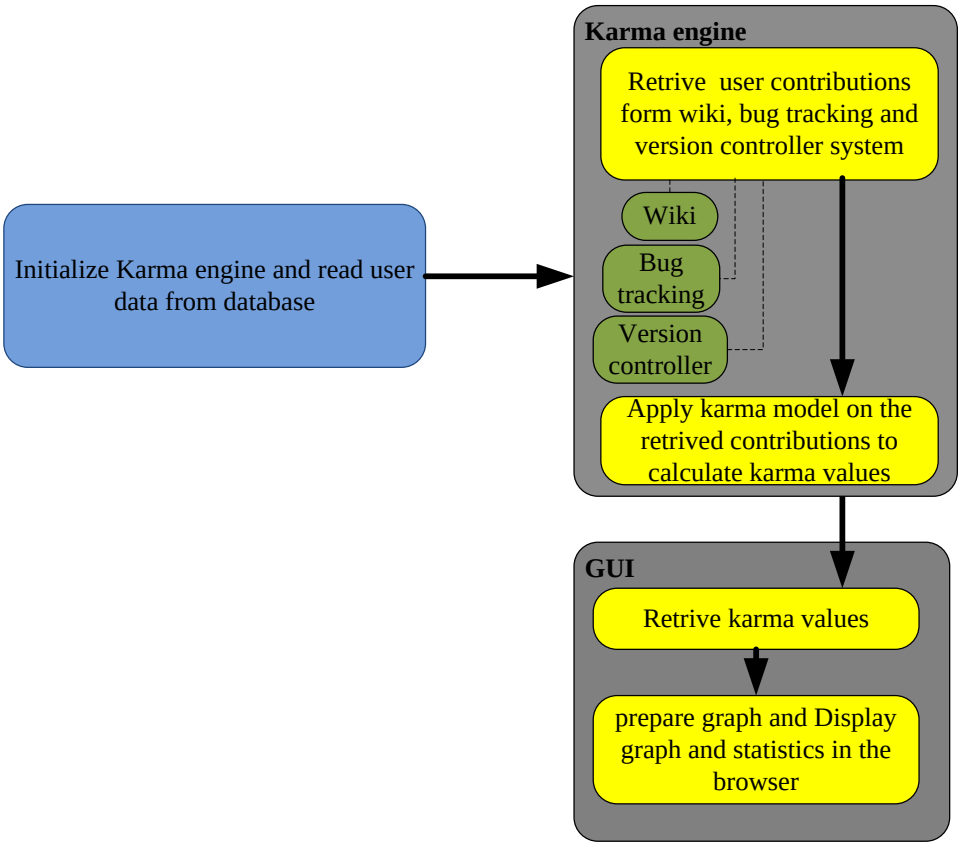

Figure 4-3 Karma engine workflow

4.5 Using KommGame Environment

In order to use the KommGame environment the learners and course moderators has to be familiar about how to use the environment. This section provides details about all the interfaces through which the users can interact with the system. Different use cases that learner can do in the KommGame environment are registration to access the KommGame environment, login to access the KommGame, resetting login password, reporting and viewing issues in bug tracking system, commenting on issues in bug tracking system, uploading a patch file for an issue, closing an open or unresolved issue on bug tracking system, contributing open content to wiki, writing content to personal blog, contributing code in version control system, viewing Karma report, viewing student's profile and giving quality token for their contribution.

4.5.1 Registration to access the KommGame environment

To make any contribution in KommGame environment the users have to log-in. To get the log-in credentials user have to be registered. For registration user have to access the internet address at [20] and click signup for a new account. User will be redirected to a new page shown in Figure 4-4. After completion of sign up, user will be sent a confirmation email to the email address user has specified. Using the confirmation email, users will be able to activate their account. If users fail to activate their account within seven days, it will be purged. User must specify a valid email address in order to receive the account confirmation email.

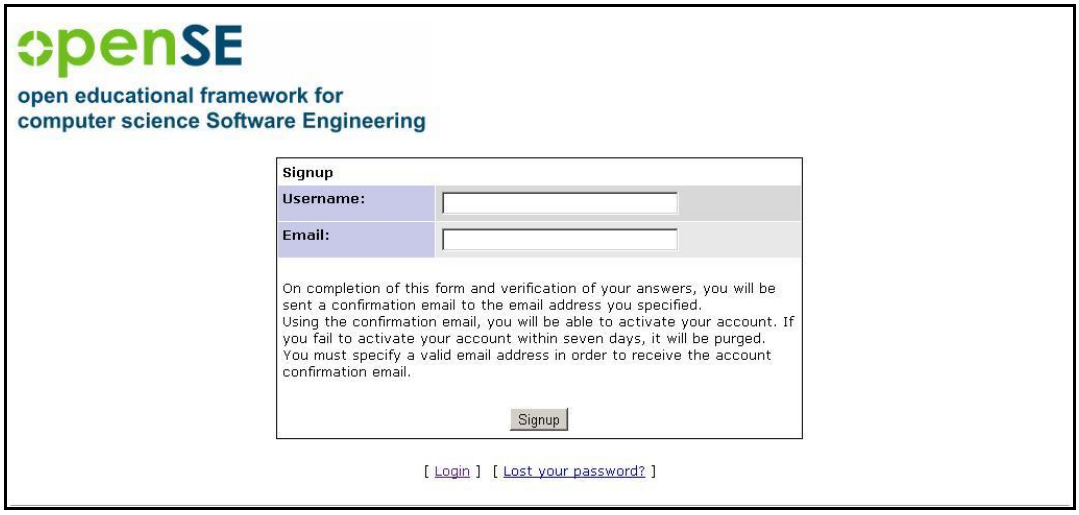

Figure 4-4 Registration page for KommGame

4.5.2 Login to access the KommGame

Once the user account is activated they can log in to KommaGame environment at [20]. The login page shown in Figure 4-5, user has to give username and password in corresponding fields and click login button to login.

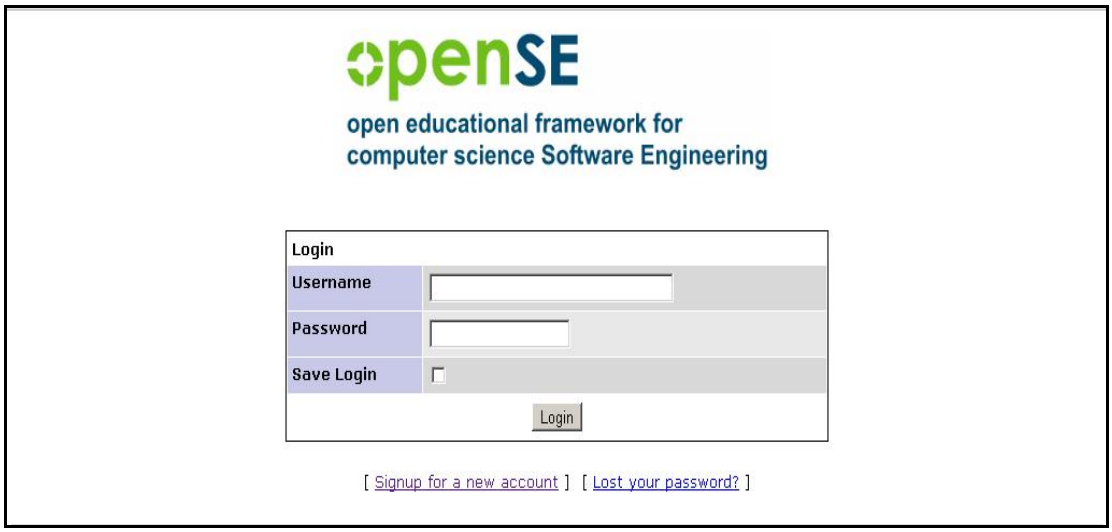

Figure 4-5 Login page for KommGame

4.5.3 Resetting login password

It is very common scenario in any kind login system, that the user forgets the password and wants to reset the password. On the login screen there is link to with title “Lost your password?” user has to click that link which will redirect the user to the page shown in Figure 4-6. On this page there are two fields: username and email. To reinstate your lost password, user has to supply the name and email address for the account and click submit button. If the data corresponds to a valid account, you will be sent a special URL via email that contains a validation code for your account. Please follow this link to change your password.

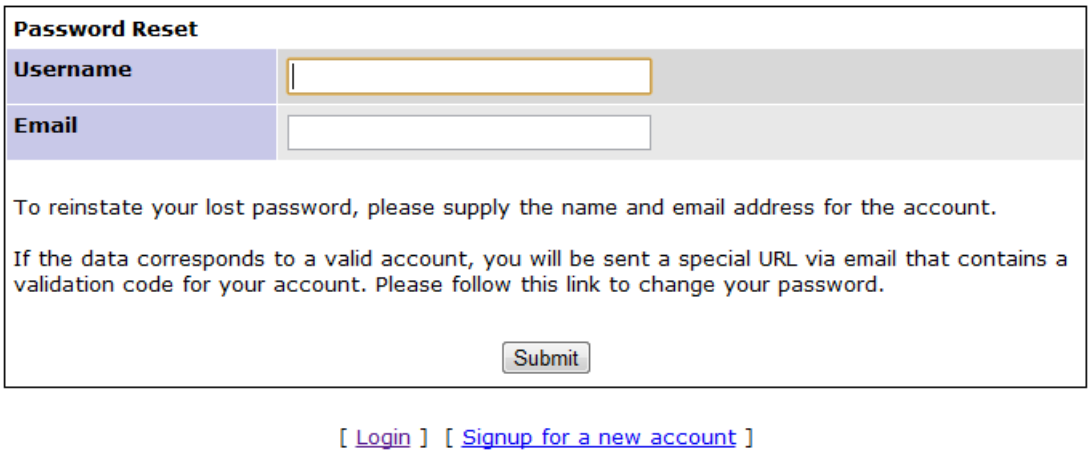

Figure 4-6 Password reset

4.5.4 Reporting and viewing issues in bug tracking system

Once the user is logged in the user interface looks like in Figure 4-7. Different parts of KommGame are accessible through links that are visible in Figure 4-7. “View issues” link is used to view all the issues that are reported. “Report Issue” link is used to report a new issue. “FLOSSpedia” link is used to access the open content generated by students. “Karma Ranks” link is used to view the karma value of all the students. “Project wiki” link is used to access the wiki of the project done during the learning process. The view provided by Figure 4-7 has color coding of issues in it. The issues that highlighted with ping color are the newly reported bug and needs attention. The issues which are highlighted with blue color are the once that are assigned to one of the developers, they are in assigned status. Once these assigned issues are fixed they are put to review status. The issues which are in grey color are the once which are review and marked as closed. This color coding makes it easy for the user to get a quick grasp of relative quantity of the number of issues under different status.

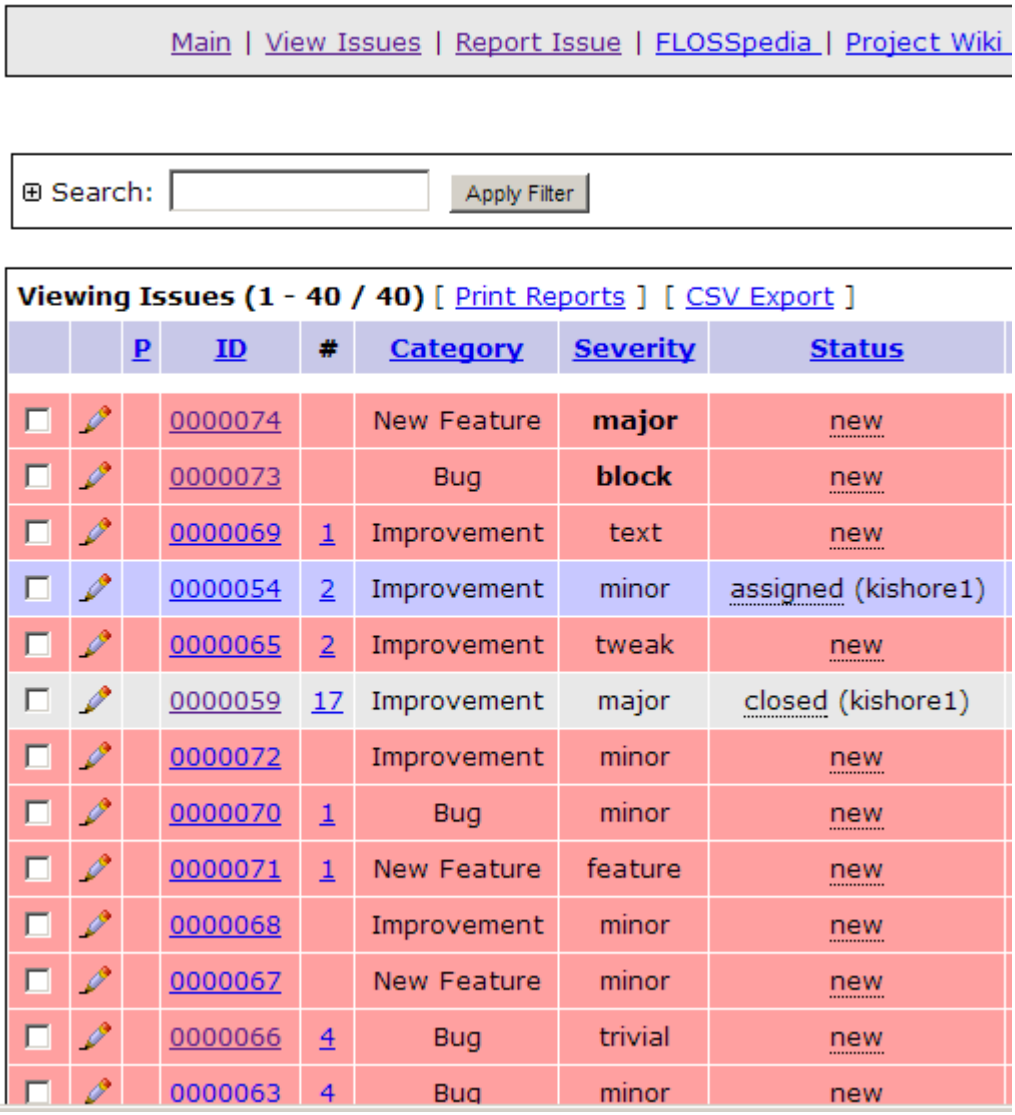

Figure 4-7 Interface to view issues.

When the user clicks “Report Issue” link the user is redirected to a page shown in Figure 4-8. User has to fill the fields with * symbol in order to save the bug. Under category field user has three options Bug, Improvement, New Feature. Corresponding option has to be selected based on the requirement. In reproducibility filed user says if this issue is a reproducible or not. Severity and priority fields tell the importance of the issue. User should write clear description of the issue in description field. Any other additional information is provided in additional information field. After filling all the fields user has to click “Submit Report” button to submit the issue.

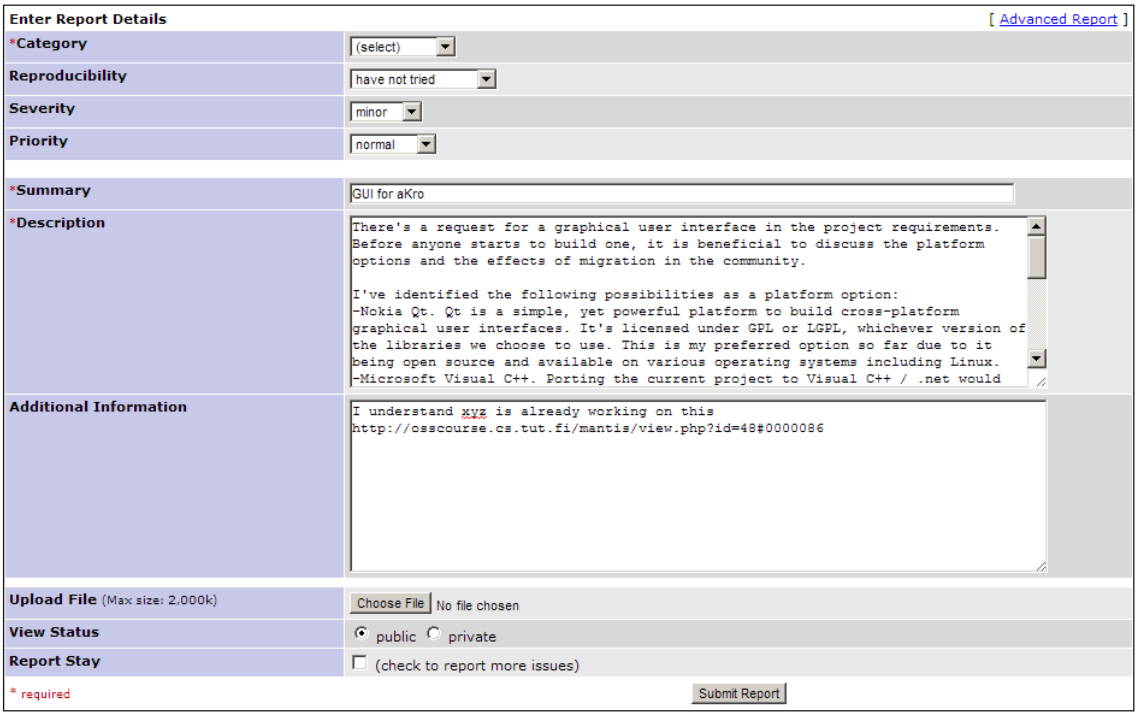

Figure 4-8 Interface to report issues.

When the user clicks “View Issue” link the user is redirected to a page shown in Figure 4-9. All the fields in the page are self explanatory. This view shows all the details about any issue that were given by the reporter. All the history associated with the issues can also be viewed in this page.

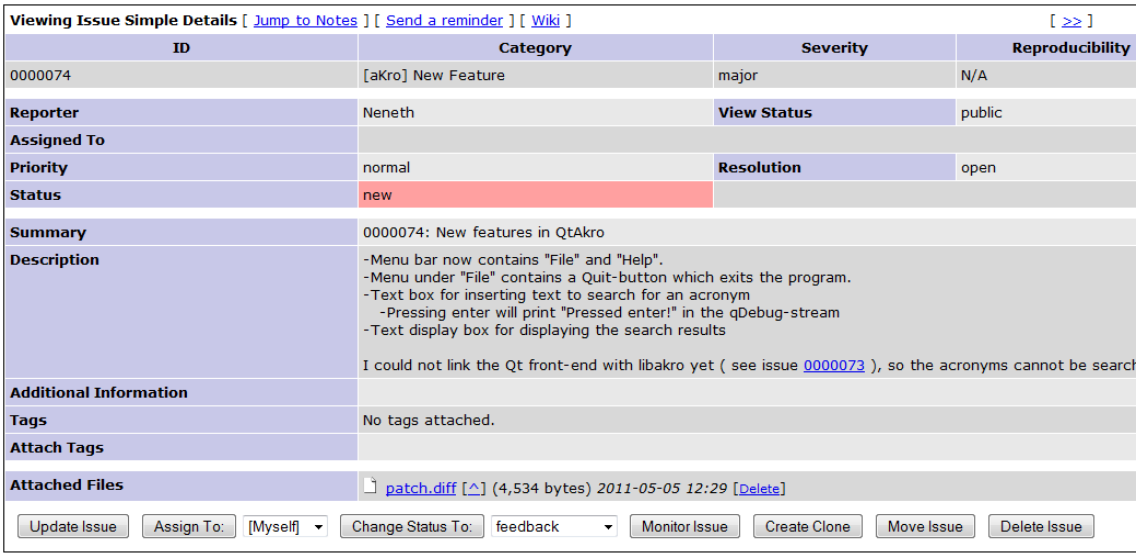

Figure 4-9 Interface to view issues.

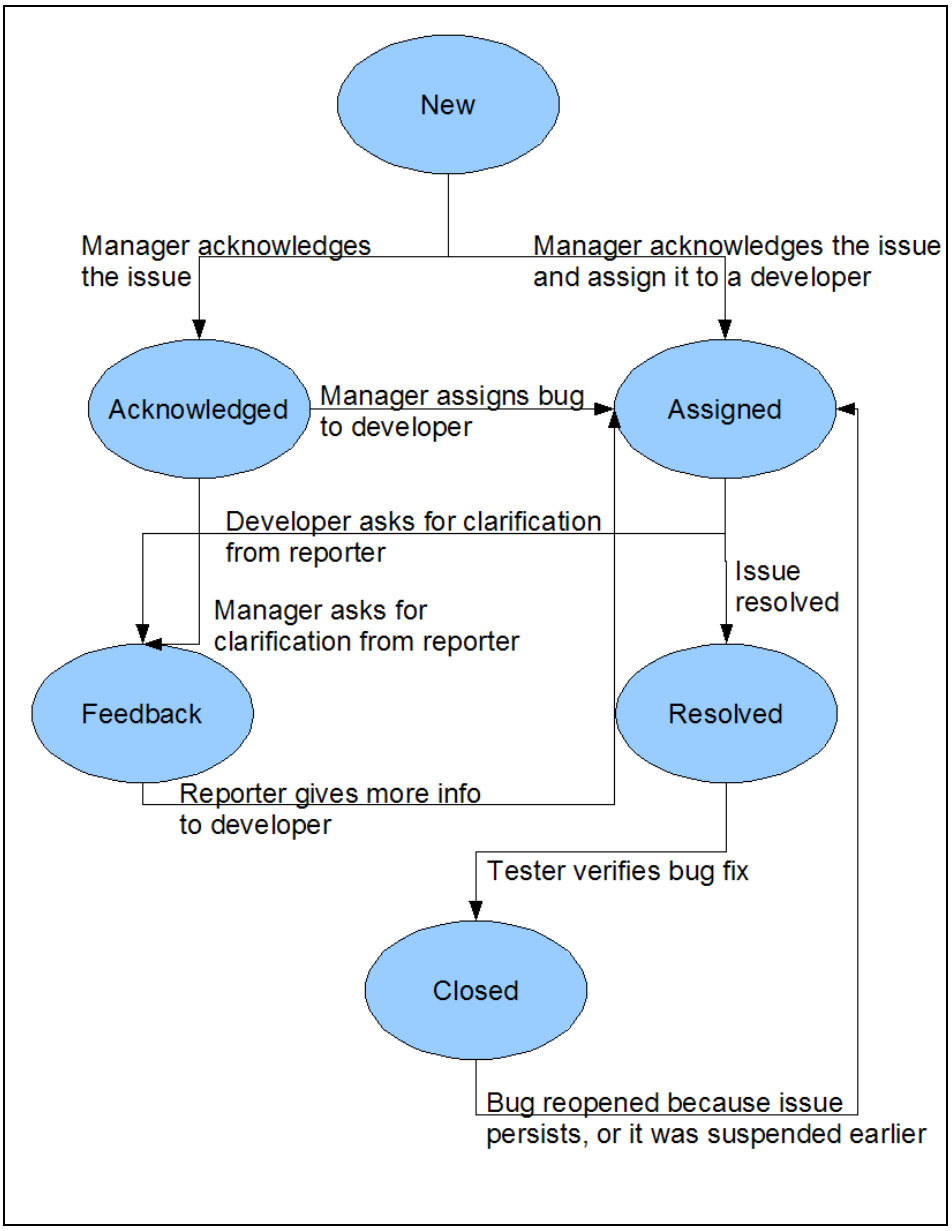

Figure 4-10 Lifecycle of bug reported in bug tracking system

The bugs reported in the bug reporting system has six stages, new, acknowledged, assigned, feedback, resolved and closed. These six stages form a lifecycle for the bug, lifecycle is shown in Figure 4-10. The first state of the bug in its lifecycle is new; this is when the bug is reported for the first time. Second state is acknowledged; this is when once the manager of the project sends an acknowledgment to the report about the bug. After the acknowledged state the bug goes to feedback state, if the manager needs more clarification on the bug from the reporter else it enters to assigned state where it is assigned to a developer. After assigned stat the developer can put the bug into feedback state if he needs more clarification on the bug from the reporter else the developer will

resolve the bug and puts the bug to resolved state. Then the tester tests the resolution submitted by the developer and puts the bug to close state else if the tester finds that the issue still persists then the tester puts back the bug to assigned state.

4.5.5 Commenting and Uploading Patch File

Once the issue is reported on bug tracking system any other member of the project can view the issue and take up the issue to resolve it. If the reported issue does not provide enough details about the issue, then the member who wants to take up the issue can write comments on the issue. Any member of the project can write comments on any issue. Once the issue is resolved by any member of the project who takes it up, then they can upload the corresponding patch files to the issue. Commenting on a bug or uploading patch file to bug can be done from "View Issue" link. Figure 4-11 shows the fields where users can upload patch file and comment on the issue.

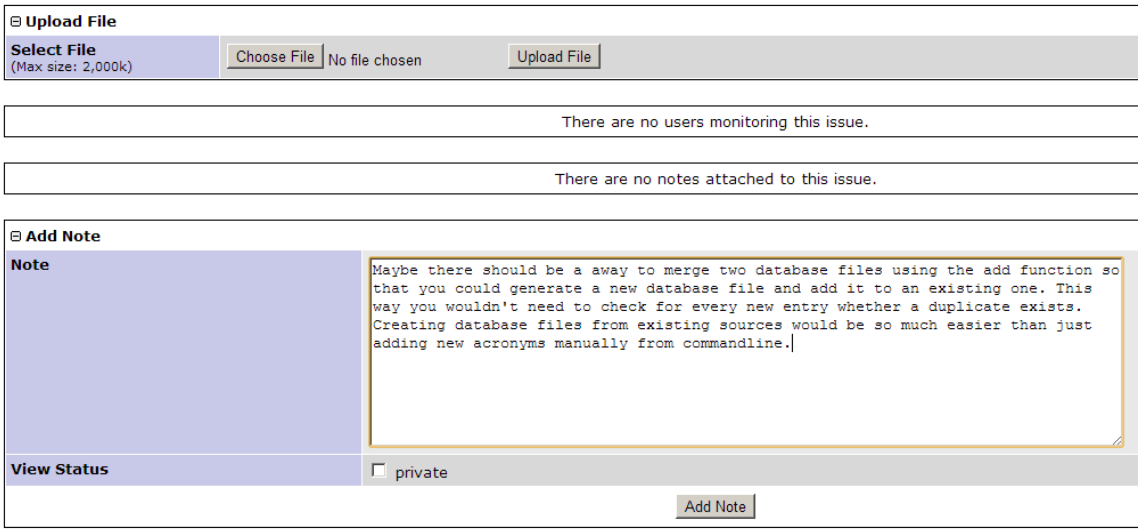

Figure 4-11 Commenting issue and Uploading patch file

In order to upload the patch file user has to click the browse button which is against select file field. Clicking on browse button opens a new window where user can browse through the file system and select the particular patch file. After selecting the patch file click on upload file will upload the file and attach the file to the issue. Any kind of files can be uploaded in same way for example if user want to upload an image which clearly depicts the issue, then the user can upload that image in the same way as patch file and this will be attached to the issue. "Add Note" section in Figure 4-11 is the section for writing comments on the issue. Text box that is against Note label is the field where

users can write the comments, after writing the comments user has to click Add Note button to submit the comments. Once the comments are submitted they appear in the history of the issue.

4.5.6 Closing or Updating an Issue in Bug Tracking System

Once any member of the project find a resolution to the issue any issue on bug tracking system. Then the project member who has found the bug will submit the resolution to the bug as an attachment and changes the state of the issue to "resolved". After the issue is in resolved state then the tester take up the resolution and verify if the issue is fixed. If the tester finds that the issue is fixed then he/she puts the issue to closed state from resolved state. Any modification to an existing issue can be done by clicking on "Update Issue" button in Figure 4-9. After clicking the button user will be redirected to the view shown in Figure 4-12. Status of the issue can be change using the dropdown list against the status label in Figure 4-12. The dropdown list is highlighted using a circle around it. Any field of the issue can be modifies in this view.

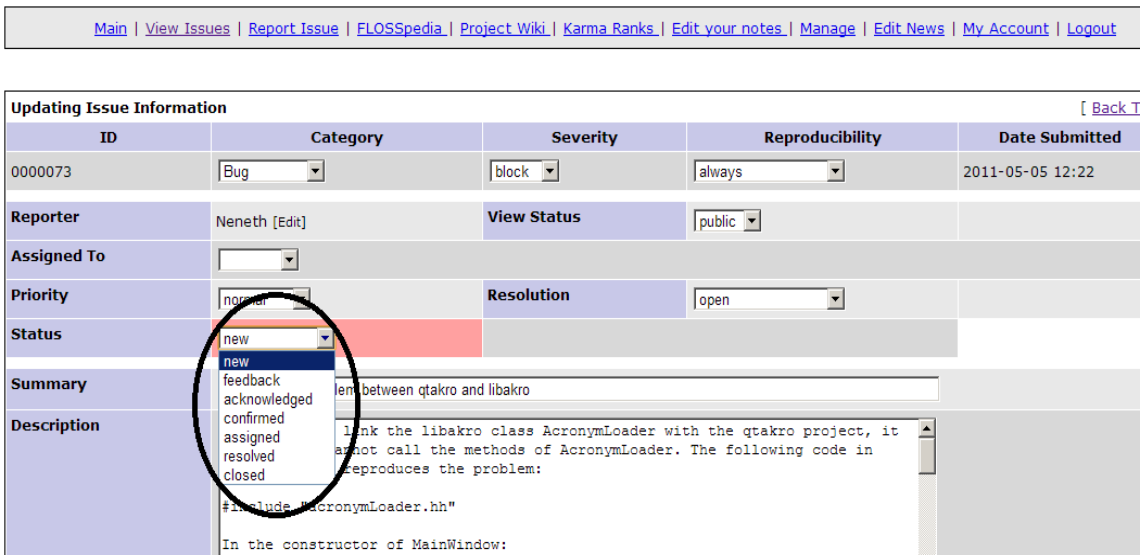

Figure 4-12 Closing or updating an existing issue.

4.5.7 Contribution Open Content to Wiki

The system used to share the open content with community is Wiki. Every participant can add open content to the wiki system by providing the authentication credentials. Figure 4-13 shows wiki edit feature where the user is adding content to the wiki. The user clicks the edit button to add content to the wiki. Once the content edited is saved, any other user can start editing the page by clicking the edit button again.

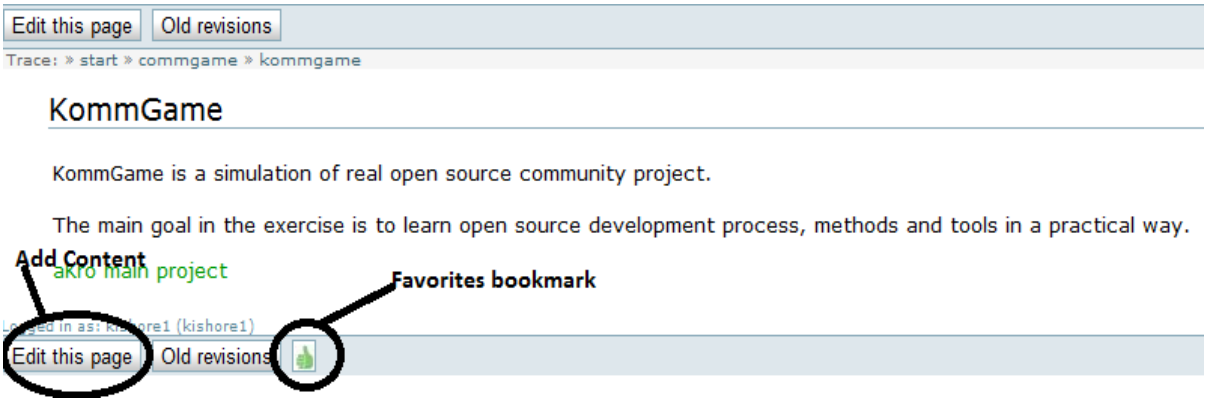

Figure 4-13 Add content to wiki and Favorites bookmark

If any community member likes the content of the wiki page then they can bookmark the page as a favorite. When some community member bookmarks a wiki page as a favorite then the author of the page gets karma.

4.5.8 Viewing the karma report

All participants can view the karma values of all the participants in a graph using the Karma reporting system. Figure 4-14 shows sample view of a karma graph. The graph shows different categories of users like committers and reported in different colors.

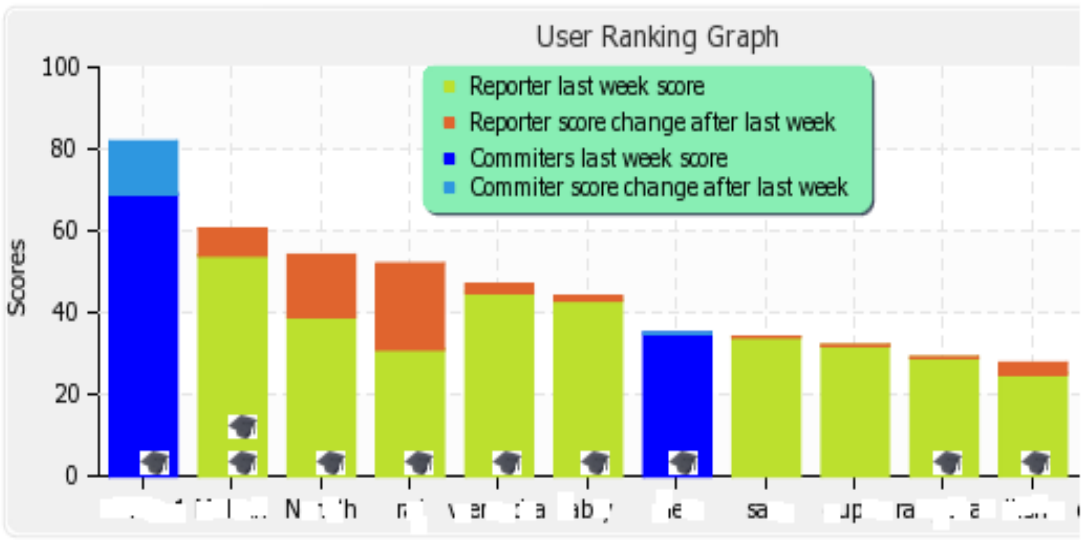

Figure 4-14 Karma reporting interface.

Each vertical bar in the graph represents the score of each user of the system. Each vertical bar has two parts with different colors; the bottom part indicates the score of the previous week. The upper part indicates the score of the current week. The hat like icon

in some bars indicates that those users were the best contributors for some week in the past. The names in Figure 4-14 are blurred out to protect user privacy. On karma report there are links to access user profile; the profile view provides details of user contribution. The profile view is discussed in next section. The legend in the Figure 4-14 describes the color codes of the chart.

4.5.9 Viewing individual's profile

All participants can have a detailed view of the activities done by a community member by using the user profile system. Figure 4-15, show the sample view of the user profile. The left side table shows numerical figures of the participant's contribution, on the right side we can see the blog that is maintained by the participant. This view is publicly available to anyone. When the course coordinator logs in they can see two more buttons "Add a hat" and "Remove a hat", here "hat" is a token for quality contribution. If the current user is the best quality contributor of the week then he will be given a hat. The remove button is used to eliminate human errors. These buttons are not visible when a learner is viewing the page.

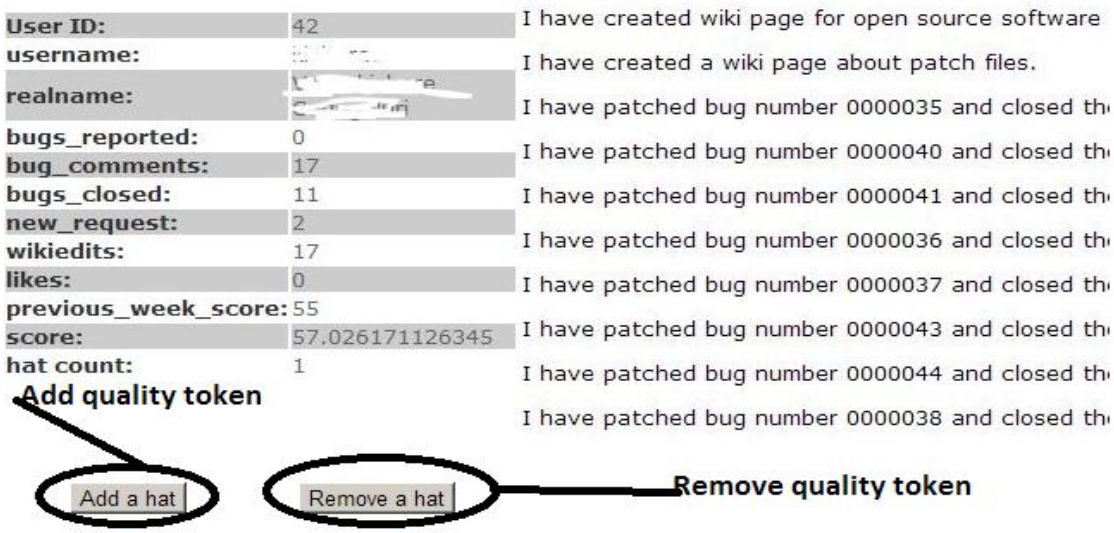

Figure 4-15 Detailed user profile interface.

4.5.10 Writing content to personal blog

Once the user is logged in the user interface looks like in Figure 4-7, in this link called "Edit notes". When user clicks the link he/she is redirected to page which looks as in Figure 4-16. This page has two buttons: edit this page and old revisions. User has to

click edit this page to write on their blog. The number on the top of the page is the unique user id given to the student while registration. User can go to previous version of the page using old revisions. This blog can be view by all other students by accessing the profile page of the particular student as shown in Figure 4-15.

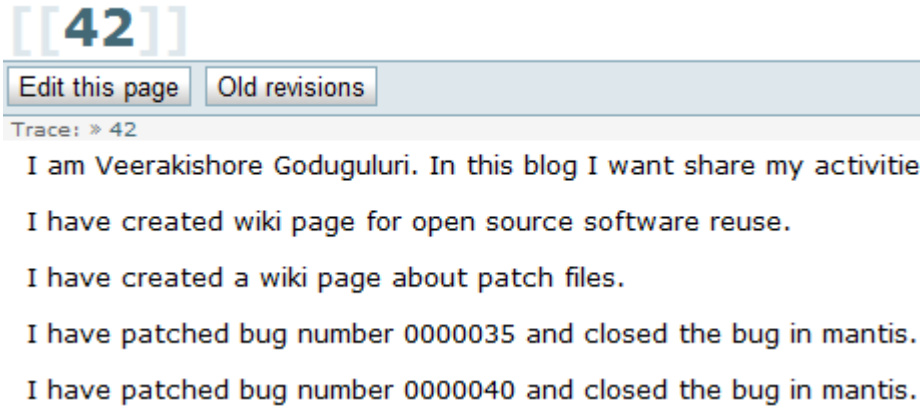

Figure 4-16 Writing personal blog

5 CASE STUDY: TEACHING OSS AT TUT

The purpose of this chapter is to validate the KommGame environment by applying in a real course at TUT. A course on OSS was organized at Tampere University of Technology. An open source infrastructure was setup for the course and a reputation system was constructed. The main goal of the course was to give a practical experience of OSS development in an actual open source infrastructure which in turn should give students the ability to participate in real OSS projects.

5.1 Course setup

The duration of the course is 16 weeks and its extent is 3 to 6 credits depending on the student performance. The course uses KommGame environment to help students how to work in open source projects. The course intention was to deliver both practical and theoretical knowledge about OSS. For practical knowledge the students were made to participate in sample open source project and for theoretical knowledge every students are made to give seminars on a given open source topic. Every week students participate in a two hour classroom session.

The first part of the session is dedicated to discussing a specific open source topic. The second part is to discuss student contributions to an example open source project. Both parts use the KommGame environment to record student contributions. In the second part of the project all the participant along with coordinator will decide the best quality contributor in the open source project for the current week. Attendance to all the course classes is not mandatory. However, participating in the sample open source project and contributing to OSS topics is an absolute requirement to pass the course. There are two coordinators for the course.

A total number of 24 students have registered to the course. Initially two students who had the prior knowledge of open source development and infrastructure were assigned the committer role and the rest of the students played the role of developers, testers, and

users. The KommGame environment was introduced to the students during the first class session. As the environment is accessible from internet, like in real open source projects, students were allowed to contribute to the project at any time they wanted to do so.

5.2 Course Activities

Course activities are divided in to three parts. In the first part every student has to select one topic from a given set of topics related to OSS. Then each student contributes with own part of content to the wiki so that information is accessed by everyone. Every week a number of students present their topic to the rest of the class. During the presentation, another group of students act as topic discussants providing peer review of the content.

The second part was to participate in the development of an example open source project called aKro. Initially aKro was developed with very basic features and made available in the version control system. In this exercise students were asked to add new features, fix existing bugs and maintain the project wiki. Whenever students observed a bug, improvement, or a new feature request they reported those issues on the bug tracking system. Any student interested in the reported issue could take up the report and submit a corresponding patch file. Once the patch file is submitted committers test the patched version and apply the patch. During the learning process students used IRC as a communication channel. Every student maintained their own blog page where they reported their activities in the project. Every week in the classroom the course coordinators inspected the contributions of the past week and rewarded the student with the best quality contribution.

The karma model used in the course covers the following activities:

1. Creating or editing Wiki pages.
2. Reporting bugs, new feature and improvement requests.
3. Comment on bugs, new feature and improvement requests.
4. Closing bugs, new feature and improvement requests.
5. Thumbs up to wiki content page.
6. Adding hats for best quality contributor of the week.

Table 5.1 List of activities and corresponding weight functions

Activity	**Weight function**
Editing Wiki pages	0.5*√(number of wiki edits)
Reporting bugs	4*√(number of bugs)
Reporting new feature requests	3*√(number of new feature requests)
Reporting improvement requests	3*√(number of improvement requests)
Comment on bugs	2*√(number of comments on bugs)
Comment on new feature requests	2*√(number of Comments on new feature requests)
Comment on improvement requests	2*√(number of Comments on improvement requests)
Closing bugs	5*√(number of bug requests closed)
Closing new feature	5*√(number of new feature requests closed)
Closing improvement requests	5*√(number of improvement requests closed)
Thumbs up to wiki	2*√(number of Thumbs up)
Getting a hat	3*(number of hats)

The third part of the course is to contribute to a real open source project after getting familiar with OSS principles and practices covered in the first two course parts. As example open source projects, students were introduced to the Apache Software Foundation programme [19] and the Demola open innovation platform [14].

5.3 Feedback

At the end of the course students were asked for feedback about the course. The feedback form contains four questions. The feedback questions and their corresponding reponses are in displayed table 5.2.

Table 5.2 Feedback questions and summary of response

Feedback Question	**Result**
Give an overall grade to the course (0-5)	Average value is 3.55
What was the best part?	1. Working in a open source environment. 2. Community game. 3. Presentation and discussions during presentation.
What would you improve?	1. Dedicated committer resource. 2. Better governance framework. 3. Distributed revision controller. 4. A more complex project. 5. Modify karma model for wiki edits.
Grade the community game project (0-5)	Average value is 3.65

The average of overall grade to the course is 3.55. The best of the course is answered as follows: The best part was to know the concepts of open source software development process and working as a member of the open source community. The course realized the difference of working in a open source environment and proprietary environment. Discussion in class were one of the best parts of the course, students liked when a presentation change to a debate with multiple participants. Some students felt that the course could have been more interesting if all the signed up students were active. Another important thing that contributes to the best part of course is Community game which gave the students Real time OSS development experience making the students to Work as a group on the project. Discussion about the quality of contribution after presentations and giving hats to the best quality contributors made course more interesting and motivated for students.

Students have suggested following improvements: Dedicated resources to act as a committer and take care of all the unanswered communications. In the current implementation committers were also participants of the course. Some students, the karma system resulted as competition factor rather than motivation factor. It would be

better if it significance of karma system was explained better in the beginning of course. Students wanted to modify the karma give to wiki edits. The current implementation of course used centralised revision control system, but students preferred to have a distributed revision control system. The decision making process was very slow during the community game. Students suggested to have a voting system that make decision make process easy. Students felt that that the project done during the course was too small. Not really much to do or problem solving. Simple project was chosen because the aim of the course is to practice open source development. The average grade of Community game was is 3.65

5.4 Evaluation

Out of 24 registered students 15 students have participated actively in the course. In the first part of the course where students have to create wiki pages about their topics, students were actively contributing and present the topics. Students were familiar with all the tools like wiki to contribute open content on OSS; this was in the first part. In the second part of the course, the students have to actually develop a project called aKro. Participating in project took some time for the students to understand the concepts like bug, features, new request and other concepts related to bug fixing. Students have learned how to handle the bugs posted in the bug tracking system, what a patch is and how to create a patch. These concepts were not taught to students but students learnt these concepts though participating in the project.

There was a slow start in the project initially as students were not used to OSS development, but later as the project progressed there were more contributions from all students. This trend can be observed from the statistics. There were totally 63 issues reported in the bug tracking system, on average each student posted 4 issues. In the first week there were only 3 issues reported and the second week 12 issues were reported. This count kept on increasing every week. When we observed all the contributions then we found that there were 6 students who were more active than rest of the community. This kind of scenario is also very common in real open source project where 90% of the code is created by very few member of the project. On an average each student contributed 40 wiki edits. Every week there was a session where the contributions to the open source project are discussed to find the best contributor and to motivate the best contributor using the reputation system. The discussing about the user contribution using

the reputation system show a good impact on most of the students, this evident from the fact that every week during the course all the students try to make their contributions just before one or two days of discussion about student contributions. But some students failed to understand the importance of reputation system setting in KommGame environment. Due to this, some students felt the reputation system as a competition system rather than a motivation system. It would be better to make sure all participate understand the importance of the reputation system so that would not lead to any confusion among the students.

Table 5.3 Statistics about the sample open source project

Statistics name	**Value**
Number of bugs reported	7
Number of new feature requests	10
Number of improvement requests	25
Number of bug comments	21
Number of comments on new feature requests	49
Number of comments on improvement requests	48
Number of lines of code	1000
Number of revisions on version control system	34
Number of code files	10
Number of wiki edits	300

Initially, it was planned that all the communication during the development process happens online, but in some situations there was verbal communication between groups of students. This verbal communication aroused when students have to take critical decision about the project. This kind of physical interaction is also common in real world projects for important decisions. As the project development was progressing, the amount of student's contribution also kept increasing. All the students played the role of user, developer, bug fixed and bug reporter. Most often the reported bugs are fixed by

the one who has reported, only in few cases bugs were fixed by others, when students were asked for “why is it so?”; they answered that coding is fun

Every week, after the seminar session the coordinator along with the student verified the contributions in the project to decide the best quality contributor and add a hat in the best quality contributor’s profile. The top contributor of the week is given a small gift to motivate others. This practice of verifying all users’ contribution and select the best quality and top contributors motivated other students to be the top or best quality contributor. Every week there was a new student who got a hat. This shows that everyone who participated in the project tried to get a hat or be a top contributor. During the initial phase of the project it was easy to get the top contributor or quality contributor. As the project progressed students became more and more motivated for the project, the student has to contribute a lot to become a top contributor.

Table 5.3 shows some statistics about the sample open source project done by students. The statistics shows that open source development can be taught in closed environment setting like KommGame. Most of the students have made similar level of contribution; the average karma value of all the students is 38.5. There are group of 5 students who were very active during the project they have made crucial decision regarding the road map of the project. It is very common scenario in open source projects to have a small group of people who are very active than the rest of the group. In nearly all open source projects over time 80% of the work is done by 20% of the people. As discussed earlier there were 15 active participants out of 24 registered and out of 15 active students, 5 students were more active than the rest of the students. From the statistical numbers discussed above and course experience we consider that controlled setting like KommGame environment along with reputation system give the students a valuable open source project experience.

6 CONCLUSIONS

With this thesis and the implemented KommGame environment the aim to make open source software education more natural and effective. The KommGame environment resembles the infrastructure of real open source community and has a mechanism to motivate students for making contributions.

OSS education is introduced in many universities where students have to present specific OSS topic and participate in real open source project, the goal was to create environment which helps students to learn the special aspects involved in OSS development process through participating in OSS project under the KommGame environment. After this kind of learning experience students find it easy and confident to start participating in real OSS projects.

The approach of KommGame for OSS education allows students to practice OSS project in safe and realistic OSS environment. The KommGame motivates the students to make more contribution to the OSS project and thus give them a valuable OSS project experience. The students are free to learn from their own and communities past and present experience. Based on the course experience teaching OSS development we consider that using KommGame setting for OSS education is a good practice. This kind of realistic setting gives the students a good starting point to work in real OSS development.

The current implementation scope of the KommGame environment the reputation model is hard coded in the program. The implementation done after designing the reputation model for OSS education and the environment is specifically implemented for OSS education. The environment does not allow the course moderator to change the reputation model. If the moderator wishes to use a different reputation model than the one in the KommGame has, this can only be done by modifying the code. One of the future development ideas is to create an interface where the course moderator can make their own reputation model; this feature will make the KommGame environment more

generic. After the course has completed we collected comments from students about improvements to KommGame environment. Most of the students suggested that to change the way of giving reputation for wiki edits. In current reputation model students are given reputation if they make any modification on wiki system. Students have proposed to limit the reputation given for wiki edits, maximum number of a student can get reputation from wiki edit per day is three. Also the current system does not take in to account about the files uploaded to bugs on bug tracking system, students also suggested to consider giving reputation for the files uploaded to bugs on bug tracking system.

The use of KommGame environment is not limited to open source software education. The concept of KommGame can also be applied in traditional programming courses where, students have to collaborate and participate in programming exercises. This kind of collaborative learning will improve the learning experience of programming from student's perspective.

So far there is no certification system to certify someone as a beginner OSS programmer. The future plans for KommGame are to research how to use KommGame as a standard system to issue the certificate of OSS beginner to any participant around the world who participate and contribute.

The course experience and feedback give by the students, concludes that KommGame is a good environment for students to learn and practice open source software development. The students also gave feedback about how make KommGame better, by considering all those improvements mentioned by students KommGame will be a better place to learn and practice open source software development

REFERENCES

1. D. Megías, J. Serra, and R. Macau: An International Master Programme in Free Software in the European Higher Education Space. *In Proceedings of the First International Conference on Open Source Systems,* pages 349–352, July 2005.

2. B. Lundell, A. Persson, and B. Lings: Learning Through Practical Involvement in the OSS Ecosystem: Experiences from a Masters Assignment. *In Open Source Development, Adoption and Innovation*, volume 234 of *IFIP International Federation for Information Processing*, pages 289–294. Springer, 2007.

3. R. Farmer and B. Glass: *Building web based reputation systems*, page 72. O'Reilly Media / Yahoo Press, 2010.

4. I. Stamelos: Teaching Software Engineering with Free/Libre Open Source Projects. *IJOSSP*, 1(1):72–90, 2009.

5. Mantis bug tracking system, *http://www.mantisbt.org*, last visited 20 March 2011.

6. Wiki system *http://www.dokuwiki.org/dokuwiki*, last visited 20 March 2011.

7. Open source teaching community *http://www.teachingopensource.org*, last visited March 2011.

8. Open software engineering, *http://opense.net*, last visited 20 March 2011.

9. M. Temperini and A. Sterbini: Learning from Peers: Motivating Students through Reputation Systems, *International Symposium on Applications and the Internet*, pages 305-308, 2008.
10. Qt developers network reputation system *http://developer.qt.nokia.com/ranks*, last visited on March, 2011.

11. Karma model of open source mobile operating system Meamo *http://wiki.maemo.org/Karma* , last visited on March, 2011.

12. C.C.P. Cruz, M.T.A. Gouvêa, C.L.R Motta, F.M. Santoro: Towards Reputation Systems Applied to Communities of Practice. *In Proceedings of 11th International Conference on Computer Supported Cooperative Work in Design.* Pages. 74-79, 2007.

13. J. Buchta, M. Petrenko, D. Poshyvanyk, V. Rajlich: Teaching Evolution of Open-Source Projects in Software Engineering Courses, *In Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM'06),* pages.136-144, 2006.

14. Demola: Open innovation platform for students and companies *http://www.demola.fi/what-demola-new-factory* last visited on March, 2011.

15. M. D. German: Experience teaching a graduate course in Open Source Software Engineering. *In Proceedings of the First International Conference on Open Source Systems (OSS 2005), Genova, Italy*, pages. 326-328, July 11-15, 2005.

16. S. Wang: Study on E-Learning System Reputation Service, *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, vol., no., pages.1-4, October 12-14. 2008.

17. A. Sterbini, and M. Temperini: Social exchange and collaboration in a reputation-based educational system, *Information Technology Based Higher Education and Training (ITHET), 2010 9th International Conference* pages. 201-207, April 29 2010-May 1, 2010.

18. G. Attwell: What is the significance of open source software for the education and training community? *In Proceedings of the First International Conference on Open Source Systems (OSS 2005), Genova, Italy*, pages. 353- 358, July 11-15, 2005.

19. The Apache Software Foundation *http://www.apache.org/foundation/* last visited on March, 2011.

20. OSS Learning environment at TUT. *http://osscourse.cs.tut.fi/mantis/login_page.php* last visited on March, 2011

21. J. Piaget: The Child's Conception of the World. Rowman and Allenheld, New York. 1960.

22. E. S. Raymond: The Cathedral and the Bazzar, O'Reilly & Associated, Inc, 2001

23. P. Bruce: Open Sources: Voices from the Open Source Revolution, O'Reilly & Associated, Inc, 1999.

24. pChart - a PHP class library to build charts *http://pchart.sourceforge.net/* last visited on March, 2011.

25. M. Ryder: *The Cyborg and the Noble Savage*, IGI Global, 2008.

26. K. Crowston, J. Howison: The social structure of free and open source software development. First Monday , pages. 1–100.

27. A. Mockus, R. T. Fielding, J. D. Herbsleb: Two Case Studies Of Open Source Software Development: Apache And Mozilla. *ACM Transactions on Software Engineering and Methodology*, pages 309–346.