NTNU

Norwegian University of
Science and Technology

hACME game

Administrative Interface

Christian Grønnet Berrum
Morten Weel Johnsen

Master of Science in Computer Science

Submission date: May 2011
Supervisor: Lillian Røstad, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

NTNU
Norwegian University of Science and Technology

hACME game
Administrative Interface

Christian Grønnet Berrum
Morten Weel Johnsen

Master of Science in Computer Science
Submission date: May 2011
Supervisor: Lillian Røstad, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In the spring of 2008 a master student developed hACME Game. The game was extended by two students in 2008/09.The game consists of a set of hacking challenges and levels. The game is intended to be used for training in the course TDT4237 software security. A key feature of the game is how it measures the participant's progress and use of hints etc.

This master thesis focuses on implementing an administrative interface for hACME Game enabling easier access to statistics, to do modifications to the game and the use of questionnaires.

The master thesis extends a project thesis written autumn 2010 by Christian Berrum and Morten Weel Johnsen.

Assignment given: January 2011
Supervisor: Lillian Røstad, IDI

Preface

This master thesis is the result of a project assigned by the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The master thesis was executed during the spring semester of 2011. The project team consisted of two students from NTNU.

The group's supervisor were Lillian Røstad, Adjunct Associate Professor at NTNU. We would like to thank Røstad for good guidance and feedback throughout the project. We would also like to thank IDI for providing office space and a server.

May - 2011

Christian Berrum   Morten Weel Johnsen

Abstract

hACME game is a game based learning tool for teaching software security. The game is intended to help raising awareness and interest in the subject of software security. The purpose of the game is to make future software developers aware of how important security is. A key feature of the game is how it measures the participant's progress and the use of hints.

This thesis focus on implementing an administrative interface for hACME game enabling easier access to statistics and results by admin users. The foundation for the thesis is a project written in the autumn semester of 2010 by Christian Berrum and Morten Weel Johnsen. The project thesis contained a conceptual design for the administrative interface.

The result is a fully functional implementation of the administrative interface. The administrative interface consists of functionality for doing management tasks and getting statistics about the game and user progress. The thesis also focuses on the implementation of questionnaires in order to get information about the player's learning process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The purpose of this project is to implement an administrative interface to hACME game. Currently, the game does not offer any possibility to view the statistics that are collected, other than through SQL queries. Furthermore, there is not any easy way of changing the game setup. This makes it hard and more time consuming to get the results an administrator want. This master thesis will focus on the implementation of an administrative interface. The administrative functionality will be divided into two separate parts where the first part is to view game statistics, while the second is to perform changes in order to improve the game. A final part of the thesis is the implementation of questionnaires. These questionnaires will help in order to get solid information about the player's learning process.

1.1 Background

The project is sponsored by the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). Furthermore the project is related to the course TDT4237 Software Security, which focuses on software security and how to develop more secure software systems. hACME game is a part of a master project started by Øyvind Nerbråten in the spring 2008 as described in Nerbråten (2008). His task was to develop a site similar to Try2Hack, which is a 'game' for developers trying to test their hacking skills. The game consists of a series of challenges which gradually increases in difficulty.

1.2 Motivation

The Open Web Application Security Project (OWASP) states in Williams (2009), which is their 2009 annual report, that the software market fails to produce secure software. OWASP argue that the reasons for this are divided. First; the increased reliance on software applications. Software applications now control our health care information, finances, military, and other important infrastructure information. Secondly, the increasing complexity and interconnection of software applications. Applications become larger and more complex, as do the interconnectivity of the applications. While there has been made progress in the area of software security the last decade, OWASP writes that the software market still struggle to eliminate simple and well understood problems such as cross-site scripting and SQL injections. Although these problems are simple and easy to understand, the market fails to eradicate them.

The motivation for creating hACME game was to raise the awareness among software developers regarding software security. It was also intended as a tool for teaching software security in the course TDT4237 Software Security at NTNU. By developing a game based on software security, the course staff wanted to teach the students, who are the future software development engineers, about the aspect of security in computer systems. By teaching the students to hack software applications and giving them the perspective of a hacker, the course staff intended to give the students deeper insight of software security.

As hACME is a game based learning tool, which provides challenges and competition, it's objective is to make the learning of software security easier and more fun. This is an important factor that hopefully give students motivation to work with the other course material and make the learning experience of software security better.

The ability to extract information about player patterns and behavior trends will help the further development of the game and might raise the learning outcomes for students. With an administrative interface that both permits the coordinator to view statistics about the game and change it accordingly will make this process a lot easier.

1.3 Problem Description

The problem description given in the master thesis task is as follows:

In the spring of 2008 a master student developed hACME Game. The game was extended by two students in 2008/09.The game consists of a set of hacking challenges and levels. The game is intended to be used for training in the course TDT4237 software security. A key feature of the game is how it measures the participant's progress and use of hints etc.

This master thesis focuses on implementing an administrative interface for *hACME Game enabling easier access to statistics, to do modifications to the game and the use of questionnaires. The master thesis extends a project thesis written autumn 2010 by Christian Berrum and Morten Weel Johnsen.*

1.4 Project Goals

The following is a list of objectives defined by the project group and the supervisor. It is intended that the group reaches all of these goals during the project periods duration. The goals are not prioritized.

- **G1 - Implement the Administrative Statistics interface**
  The administrative statistics interface offers statistical results from hACME game. The results are visualized to the administrator to help interpret the data. The statistics panel is divided into statistics for comments, challenges, attempts and users.
- **G2 - Implement the Administrative Tools interface**
  The administrative tool interface offer managements tools such as news publishing and level managements. This will help the administrator change the game setup in an easy manner. The tool panel is divided into management tools for news, level setup, users and site statistics.
- **G3 - Implement the usage of surveys**
  The usage of surveys will give the administrator feedback on the students theoretical understanding of software security.
- **G4 - Run a test phase**
  The test period will give feedback on how a small sample of students will perform with the proposed changes added to hACME game. Administrative users will also be able to test the implemented system and give feedback.
- **G5 - Use the test period results to suggest improvements**
  The test period will serve as a basis for suggestions on further improvements on the game.

1.5 Approach

In the following section the approach for reaching each subgoal is explained. A more in depth description can be found in later chapters.

- **G1 - Implement the Administrative Statistics interface**
  To reach this goal, the conceptual design described in chapter 5.3 of

Berrum and Johnsen (2010) should be implemented. This entails implementing panels for statistics on comments, challenges, attempts and users.

- **G2 - Implement the Administrative Tools interface**
  To reach this goal, the conceptual design described in chapter 5.4 of Berrum and Johnsen (2010) should be implemented. This entails implementing panels for management of users, news, levels and site statistics.

- **G3 - Implement the usage of surveys**
  To reach this goal, the proposed changes described chapter 5.6 of Berrum and Johnsen (2010) should be implemented. This entails implementing panels for surveys and reflections before and after a level is played.

- **G4 - Run a test phase**
  To reach this goal, a test period of two weeks should be devoted to testing the implementation of *G1*, *G2* and *G3* based on a set of evaluation criteria.

- **G5 - Use the test period results to suggest improvements**
  To reach this goal, the results from *G4* should be interpreted and serve as a basis for suggestions for further improvements.

1.6 Deliverables

The following is a list of deliverable content supplied to IDI at the end of the semester.

- **D1** - This report containing all research done by the group.
- **D2** - Source code for the implementation.
- **D3** - A working copy of implementation running on *www.hacmegame.org.*

1.7 Outline

The following section contains an overview of the report's outline and a brief description of each chapter. The report consist of 8 chapters and 4 appendices.

Introduction

This chapter consists of a brief discussion on the project's background and the motivation for executing the project. It also contains a definition of the project and a list of project goals. Finally, the approach for implementing the project and a report outline is given. The purpose of the introduction chapter is to give the reader a brief understanding of the current situation and an overview of the information in the later chapters.

Preliminary Study

This chapter gives a brief overview of the background information on game based learning, hACME game and information visualization. The purpose of this chapter is to give the reader all the necessary background information in order to understand the later chapters.

Requirements

In this chapter it is intended to give a complete description of the system developed. It includes use cases and requirements for the administrative view in hACME game. The chapter is loosely based on the IEEE830-1998 standard, IEEE (1998b).

Design

This chapter is intended to give an overview of the architecture of the administrative view. It describes each of the components in detail. The chapter is loosely based on the IEEE1016-1998 standard, IEEE (1998a).

Security Analysis

This chapter is intended to give a security analysis of the hACME game's administrative interface. The purpose of this chapter is to supply the reader with information about all security aspects of adding an administrative interface to hACME game.

Evaluation

In this chapter it is intended to give an extensive evaluation of the implemented application. The chapter first defines a set of evaluation criteria, and then the final results from the test phase is presented.

Results and Discussion

This chapter provides a discussion on the result of the hACME game administrative interface implementation and the evaluation phase.

Conclusion and Further Work

In this chapter it is intended to give suggestions for further improvements in the game as well as a conclusion of the master thesis.

Chapter 2

Preliminary Study

This chapter is intended as an introduction to the field of game based learning and information visualization. It also gives an overview on previous efforts on developing hACME game. Most of this chapter is a recurrence of the theoretical groundwork in Berrum and Johnsen (2010). However, in order to fully understand the later chapters, this introduction is necessary.

2.1 Game Based Learning

This section will give an introduction to game based learning and its applications. Game based learning is found useful in a wide range of contexts, such as military training, medicine, and aspects of legal and science education.

The process of learning require effort. The effort put into learning often reflect certain motivations. These motivations can be high-stakes testing, like exams, where rewards and consequences are attached to the results of a test, as seen in Amrein and Berliner (2003). Other factors for motivation can be fear or the need to please others as well as personal growth and achievement. It can be hard to create good learning efforts when the motives are long range goals or rewards, such as a good test result 5 months from now. This is where computer games can play a role in education. Computer games have increased in popularity among children and young adults in the last decades and have become a part of our cultural environment, as seen in Oblinger (2004). Studies on young students in UK and Greece showed that most of them were regular game players and that the main reason for their Internet usage was online game playing, (McFarlane et al., 2002) and (Papastergiou and Solomonidou, 2005). Games are an important part of young peoples lives outside school and creates engagement in them, (Papastergiou, 2009), and may result in a new learning culture that corresponds better with students' habits and interests, (Prensky, 2004).

Norman (1993) emphasizes that games satisfy the basic requirements for engaging learning environments. Generally, fun and divertisement is considered some of the main characteristics of a computer game. In an educational game, the instructional content is blurred with game characteristics such as fun, (Pivec et al., 2003). Thomas W. Malone in his paper *Toward a Theory of Intrinsically Motivating Instruction*, (Malone, 1981), generated a set of guidelines for what makes video games fun and engaging. His findings was a combination of fantasy, curiosity and challenge. Malone later used these three concepts to outline a set of guidelines for what enjoyable educational games should contain. Bowman (1982) studied Pac-Man players. He gives a very similar framework to Malone. Malone argues that educational games should employ:

- Personally meaningful goals
- Variable difficulty levels
- Multiple level goals
- Hidden information
- Randomness
- Appealing fantasy

However, Kiili (2005) writes that the fun factor is not the magic bullet in educational game design. The promise of educational games is to engage and motivate players through direct experience with the game world. The game should be motivating so that the player repeats the games learning cycles within a context. Learning is the process for students of acquiring new knowledge. It is generally accepted that students should be involved in practical exercises and activities and not just receiving the results of somebody else's activities summarized in text or audio, (Bransford et al., 2002). Examples of this is experiential learning where students learn through reflection on doing (Goossens et al., 1984).

Garris et al. (2002) states that there exists a learning model that is inherent in most educational games. This model, shown in figure 2.1, describes how educational games are created and how the learning cycle should trigger certain outcomes. First, instructional content is mixed with game characteristics. Second, this triggers a game cycle containing user judgment, user behavior and system feedback. This is the cycle that creates self-motivated game play and finally, learning.

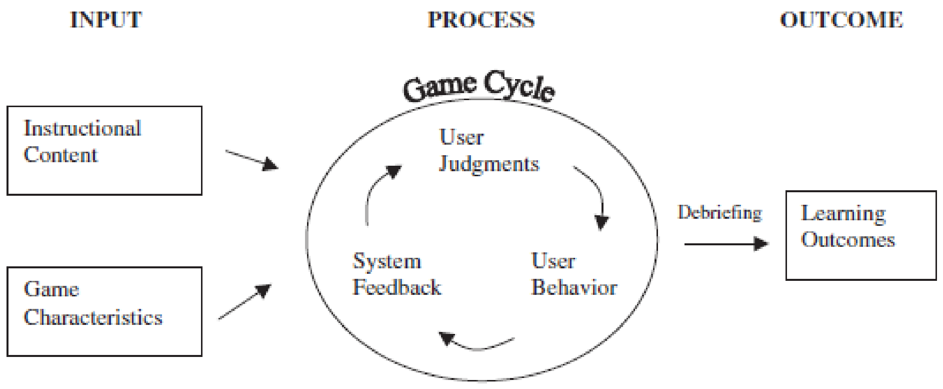


Figure 2.1: Input-Process-Outcome Game Model

This can be related to hACME game by first looking at the instructional content. hACME game are intended to support the course software security and to do so they can adapt much of the traditionally classroom teaching. Garris et al. (2002) mentions 7 different game dimensions. This thesis will only be concerned with those that can be related to hACME game. The first dimension are rules and goals, as described by Garris et al. (2002). This is described by clear rules, goals, and feedback on progress toward goals. hACME game gives the user clear goals by describing these as challenges. The second dimension that occur in hACME game are challenge. In the two earlier master theses, Nerbråten (2008) and Hagen and Taraldset (2009), the challenge dimension have been very visible. They have focused on making the difficulty of challenges at an optimal level, thus trying to gradually increase the difficulty. The different dimensions concerned can be further studied in Elliot and Harackiewicz (1994) and Driskell and Dwyer (1984).

The game cycle in hACME game consist of how the user conceives and performs the different challenges. The different challenges challenge the users and make them have to interpret and make user judgments. There will be different user behaviors for the different users and then the system gives feedback when users try to complete the challenges. There have been discussed the use of introducing repeatable challenges in Hagen and Taraldset (2009). This would increase the game cycle process on the different challenges and people learn better by repeating.

When the users have completed each challenge, and also when the users have completed all challenges they have the possibility of giving feedback to the system. Based on this feedback and also the statistics about the progress of each user it is possible to define the learning outcomes of the game.

2.2 hACME Game

This section contains a brief description of the game and the gameplay, then it gives an overview of the first two versions created in the previous master theses. The section is intended to serve as a basis for later chapters where hACME game is discussed.

2.2.1 The Game

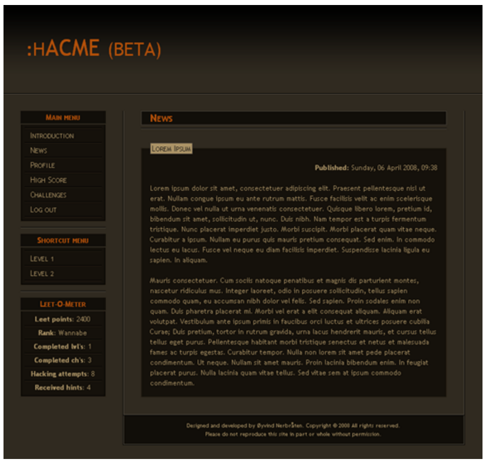
hACME game is a web application where the user assumes the role as a hacker, a person who breaks into computer systems which they are not intended to have access (Aschehoug and Gyldendal, 2010). hACME game consists of a series of challenges divided into several levels with increasing degree of difficulty. All challenges within a level must be completed in order to advance to the next level (except for optional challenges). hACME game uses a point system to rank users who play the game and these are mainly based on completed challenges. Each failed hacking attempt will yield a slight decrease of points for the given challenge, so will each received hint. Bonus points will be given for each completed level.

Currently there are over 1100 registered users and over 51.000 registered hacking attempts. More details about the database can be found in Appendix C.

2.2.2 First Version

The first version of hACME game was implemented by Øyvind Nerbråten in the spring semester of 2008. The result of his master thesis was a functional prototype of a learning game described in the master thesis Nerbråten (2008). The game consisted of the most common vulnerabilities and attacks described in Andrews and Whittaker (2006). The beta version has 9 challenges distributed over four levels, with a total of 16 available hints. A screen capture of the first version of the game can be seen in figure 2.2.

The beta version consisted of a web page where users could log in and do challenges and other tasks such as displaying user profile, news and change profile settings. A use case diagram can be seen in Appendix A. The overview of challenges can be seen in table 2.1. In addition to user management and challenges, the hACME game beta version also contained functionality for data collection. Data collection is divided in two parts; *user details* and *user progress.*



Leet points: 2400
Rank: Wannabe
Completed lvl's: 1
Completed ch's: 3
Hacking attempts: 8
Received hints: 4

Published: Sunday, 06 April 2008, 09:38

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent pellentesque nisl ut erat. Nullam congue ipsum eu ante rutrum mattis. Fusce facilisis velit ac enim scelerisque mollis. Donec vel nulla ut urna venenatis consectetuer. Quisque libero lorem, pretium id, bibendum sit amet, sollicitudin ut, nunc. Duis nibh. Nam tempor est a turpis fermentum tristique. Nunc placerat imperdiet justo. Morbi suscipit. Morbi placerat quam vitae neque. Curabitur a ipsum. Nullam eu purus quis mauris pretium consequat. Sed enim. In commodo lectus eu lacus. Fusce vel neque eu diam facilisis imperdiet. Suspendisse lacinia ligula eu sapien. In aliquam.

Mauris consectetuer. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer laoreet, odio in posuere sollicitudin, tellus sapien commodo quam, eu accumsan nibh dolor vel felis. Sed sapien. Proin sodales enim non quam. Duis pharetra placerat mi. Morbi vel erat a elit consequat aliquam. Aliquam erat volutpat. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis pretium, tortor in rutrum gravida, urna lacus hendrerit mauris, et cursus tellus tellus eget purus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur tempor. Nulla non lorem sit amet pede placerat condimentum. Ut neque. Nullam sit amet mauris. Proin lacinia bibendum enim. In feugiat placerat purus. Nulla lacinia quam vitae tellus. Sed vitae sem at ipsum commodo condimentum.

Designed and developed by Øyvind Nerbråten. Copyright © 2008 All rights reserved.
Please do not reproduce this site in part or whole without permission.

Figure 2.2: Screen capture of the beta version of hACME game.

User details contains data collection on study program, grade and knowledge level. These details are intended to give valuable background information on the user. *User progress* contains data collection on challenge progression and outcome of hacking attempts. It gives information on time spent on each challenge and hints used. This information can be used for various purposes. It can be used to determine which challenge is particularly easy or hard or to find how much time each users spends on each challenge or the whole game.

Level	Challenge	Name
1	1.1	The idiot test
	1.2	A slightly more interesting challenge
2	2.1	VIP
	2.2	Confirm order
	2.3	Random number puzzle
3	3.1	The garden of Eden
	3.2	Breaking the barrier
	3.3	The break in
4	4.1	All your database are belong to us

Table 2.1: Challenges in hACME game beta version.

The game was deployed on *www.hacmegame.org* where students taking the course TDT4237 were invited to play. In the duration of one week the game had a total of 51 registered users, 1177 hacking attempts where 297 where successful. The statistic showed that 47% of the users completed all challenges. The average playing time for users who completed all challenges was 1 hour and 40 minutes.

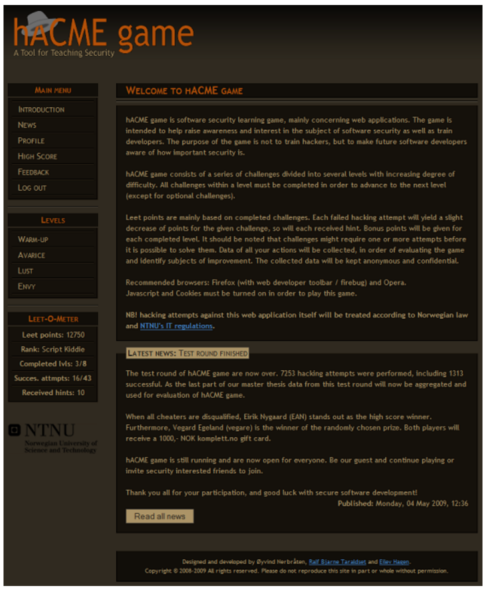
2.2.3 Second Version

The second version of hACME game was extended by Eilev Hagen and Ralf Bjarne Taraldset during autumn 2008 and spring 2009. Based on requirements and design derived in a preliminary project, the game was extended to contain more challenges, improved game-based learning aspects, improved motivational factors and extended data collection mechanisms. Also a security analysis was done on the game.

Hagen and Taraldset (2009) defined thirteen functional requirements and six non-functional requirements. All of them can be found in Appendix B. Twelve of the thirteen functional requirements were implemented and consisted of extending the rating system, collect data and add surveys. The non-functional requirements consisted of extending the game with more challenges, help functionality and to be more self motivating. All of them were fulfilled. The complete list of challenges can be seen in table 2.2. The challenges were also grouped according to attack type. This distribution can be found in Appendix B. Since the game itself is heavily exposed to attacks, as discussed in Hagen and Taraldset (2009), best practice security was implemented. The complete list of security requirements can be seen in Appendix B.

Again, the game was evaluated using an empirical student test. The game was deployed on *www.hacmegame.org* where students taking the course TDT4237 Software Security were invited to play. The test period was a total of 15 days. The game had a total of 74 registered players with 5179 hacking attempts and 1041 requested hints. The feedback from the students playing the game was positive and it seemed like they enjoyed playing the game. The results also showed that players ability to solve the various challenges increased through the game.

The thesis suggests a set of improvements that could be implemented at a later time. This includes an administrative interface that contains functionality for editing news, player challenges, hints and surveys. Data collection is also suggested extended. This is mainly focused around the attempt log. The last suggestion is cheat mitigation in order to avoid students cheating. Hagen and Taraldset (2009) proposes using randomization and cheat detection. A screen capture of the second version of hACME game can be seen in figure 2.3.



Leet points: 12750
Rank: Script Kiddie
Completed lvls: 3/8
Succes. attmpts: 16/43
Received hints: 10

Welcome to hACME game

hACME game is software security learning game, mainly concerning web applications. The game is intended to help raise awareness and interest in the subject of software security as well as train developers. The purpose of the game is not to train hackers, but to make future software developers aware of how important security is.

hACME game consists of a series of challenges divided into several levels with increasing degree of difficulty. All challenges within a level must be completed in order to advance to the next level (except for optional challenges).

Leet points are mainly based on completed challenges. Each failed hacking attempt will yield a slight decrease of points for the given challenge, so will each received hint. Bonus points will be given for each completed level. It should be noted that challenges might require one or more attempts before it is possible to solve them. Data of all your actions will be collected, in order of evaluating the game and identify subjects of improvement. The collected data will be kept anonymous and confidential.

Recommended browsers: Firefox (with web developer toolbar / firebug) and Opera.
Javascript and Cookies must be turned on in order to play this game.

NB! hacking attempts against this web application itself will be treated according to Norwegian law and NTNU's IT regulations.

Latest news: Test round finished

The test round of hACME game are now over. 7253 hacking attempts were performed, including 1313 successful. As the last part of our master thesis data from this test round will now be aggregated and used for evaluation of hACME game.

When all cheaters are disqualified, Eirik Nygaard (EAN) stands out as the high score winner. Furthermore, Vegard Egeland (vegare) is the winner of the randomly chosen prize. Both players will receive a 1000,- NOK komplett.no gift card.

hACME game is still running and are now open for everyone. Be our guest and continue playing or invite security interested friends to join.

Thank you all for your participation, and good luck with secure software development!
Published: Monday, 04 May 2009, 12:36

Designed and developed by Øyvind Nerbråten, Ralf Bjarne Taraldset and Ellev Hagen.
Copyright © 2008-2009 All rights reserved. Please do not reproduce this site in part or whole without permission.

Figure 2.3: Screen capture of the second version of hACME game.

2.2.4 Preliminary Project

During the autumn of 2010 two students wrote a preliminary project about hACME game in Berrum and Johnsen (2010). The result was a conceptual design based on a preliminary study on learning methods and visualization of statistical data. A design was derived with use cases, user stories and preliminary user interface mock ups. The design describes the statistical tool and a tool for changing the setup of the games challenges according to the statistics. The report also suggests how to improve the game further in the future with the use of surveys.

2.3 Information Visualization

In hACME game user data is collected throughout the whole game. This data is intended to give the administrative users valuable feedback on the learning process of students playing the game. Currently data is stored as records in a database system and is not presented visually to the administrative users trough the web interface. There is currently no possibility of adequately exploring the large amounts of data collected, making the data more or less useless.

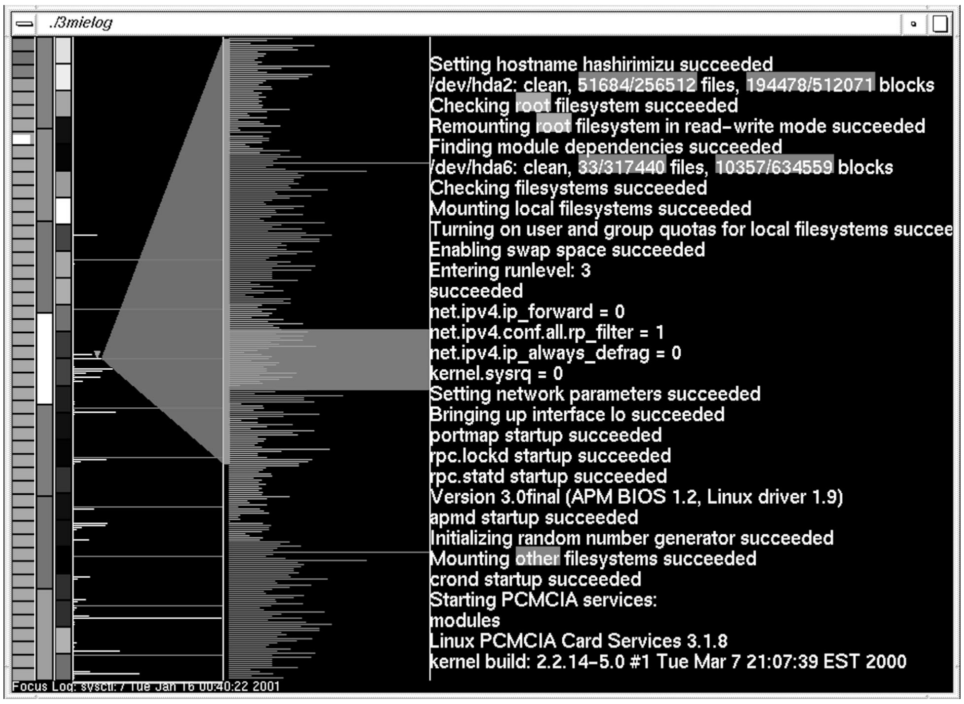
The hACME game database consist of 20 tables with over 90.000 records, almost 300 textual comments and other feedback. More details about the database can be found in Appendix C. The data varies from series of numerical data, such as time used on completing a challenge, to multidimensional tables such as the hacking attempt records. In hACME game, the only possible way of extracting information is through the database directly. This means that the system administrator must browse for records or write SQL queries in order to access collected data. One of the main tasks of this report is to implement an administrative interface for extracting statistics on the learning process of the students playing hACME game.

Keim et al. (2002) says when the amount of data becomes large, highly nonhomogenous and noisy, finding the valuable information becomes increasingly difficult. This is why the field of information visualization can be of great importance when designing an administrative interface aimed at giving administrative users an effective and efficient interpretation of hACME game data.

Visual representation of statistics has been used for a long time (Friendly and Denis, 2001). The earliest efforts include cartography and maps in aid of navigation, analytic geometry, demographic and social statistics. Diagrams and other graphics forms were used to create visual representations of statistics. These traditional methods are not always useful, especially when it comes to data embodied in lines of text (Eick, 1994). The textual lines may be logical entries such as sentences, database records or log entries, but can also be lines of code in a program or other textual statistics. One example of a visual representation of text is MieLog (Takada and Koike, 2002), as seen in figure 2.4. MieLog is an example of an interactive visual log browser and provides another interpretation technique of textual data besides just reading.

When data sets becomes multidimensional, such as a relational database, visualization techniques becomes helpful (Siirtola, 2007). Visual data exploration, as described in Keim (2002), aims at helping the human mind in the exploration process of large data sets in computer systems. It is increasingly being used in large data systems, digital libraries, data mining, financial systems and manufacturing production control systems (Bederson and Shneiderman, 2003).

Keim et al. (2002) proposes that a graphical encoding system has to be expressive, effective and appropriate. Mackinlay (1986) explains expressiveness as displaying relevant data and only the relevant data. This entails filtering out all uninteresting data from the visualization. The effectiveness is explained as the ability of the viewer to interpret the displayed data efficiently and correctly. The displayed data also has to be appropriate in the specific context. Keim et al. (2002) refers to this as general rules for information visualization.



```
Setting hostname hashirimizu succeeded
/dev/hda2: clean, 51684/256512 files, 194478/512071 blocks
Checking root filesystem succeeded
Remounting root filesystem in read-write mode succeeded
Finding module dependencies succeeded
/dev/hda6: clean, 33/317440 files, 10357/634559 blocks
Checking filesystems succeeded
Mounting local filesystems succeeded
Turning on user and group quotas for local filesystems succee
Enabling swap space succeeded
Entering runlevel: 3
succeeded
net.ipv4.ip_forward = 0
net.ipv4.conf.all.rp_filter = 1
net.ipv4.ip_always_defrag = 0
kernel.sysrq = 0
Setting network parameters succeeded
Bringing up interface lo succeeded
portmap startup succeeded
rpc.lockd startup succeeded
rpc.statd startup succeeded
Version 3.0final (APM BIOS 1.2, Linux driver 1.9)
apmd startup succeeded
Initializing random number generator succeeded
Mounting other filesystems succeeded
crond startup succeeded
Starting PCMCIA services:
modules
Linux PCMCIA Card Services 3.1.8
kernel build: 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000
Focus Log: sysctl:7 Tue Jan 16 00:40:22 2001
```

Figure 2.4: MieLog

Keim (2002) explains the idea of visual data exploration of presenting data in a visual form to help the human mind to draw conclusions from, and interact with the data. Keim (2002) and Shneiderman (1996) describes the process of visual data exploration as a three step process of *gaining overview*, *zooming and filtering* and finally, *viewing details on demand*. In the first step, the user gets an overview over all available data. Next, the user can then decide on which areas of information that is interesting and then access details on that data by focusing or zooming in. MieLog is an example of this and can be seen in figure 2.4. On the left side is the overview. When selecting a section in the overview, the right side shows the details on that section. Figure 2.5 shows the SolidSX giving a radial view of source code elements and the relationships between these elements in what is referred to as a linked view (Reniers et al., 2010). This enables users to create complex analysis of correlations between different data.

Koutsofios et al. (1999) describe the SWIFT-3D system, a data visualization and exploration system created at AT&T Labs for large scale network analysis. Figure 2.6 shows a network traffic animation, where network usage is shown in relation to time. Koutsofios et al. (2002) describes this in more detail. This is an example of how multidimensional time series can be dis-

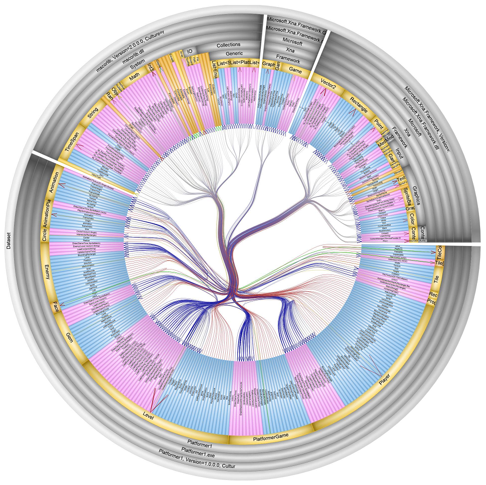


Figure 2.5: SolidSX

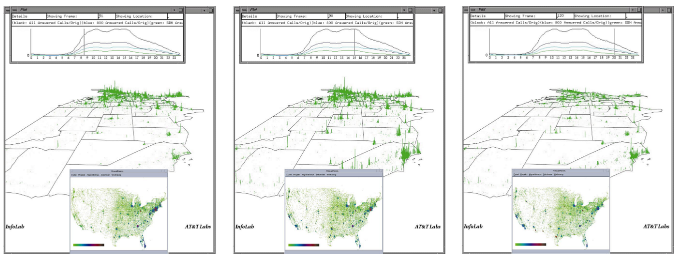


Figure 2.6: SWIFT 3D

played. Data shown in the view is a combination of time, place and volume.

All of these are examples of visual data browsing applications with overview, focus and details on demand. Baudisch et al. (2001) describe another similar concept, focus plus context screens, which combines the overview with details view, using what they call low-res and high-res regions. The visualized content preserves it's scaling, but the resolution changes. This can be seen in figure 2.7. The overview (low-res) region gives a bird's eye view of all data, while the focus (low-res) region gives a detailed view of a section of data.

Gutwin (2002) gives an example of an interactive fish eye view for browsing web pages. In this example the focus point is connected to the mouse pointer. When the mouse hovers a node, the focus point expands and the details on that node can be viewed in the focus area. This can be seen in 2.8. Again, the user gets an overview of all data, corresponding to the context area. When the users hovers a web page element, details on that element is

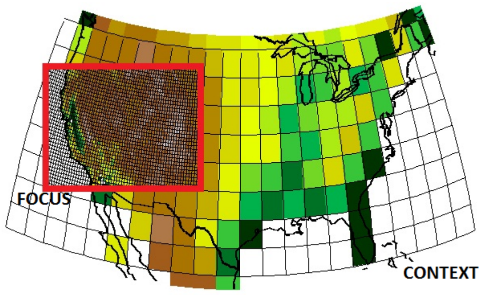


Figure 2.7: Focus plus Context View

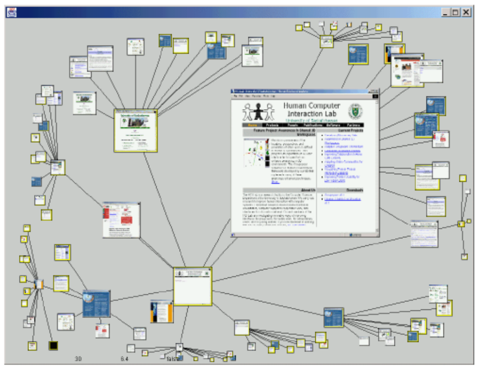


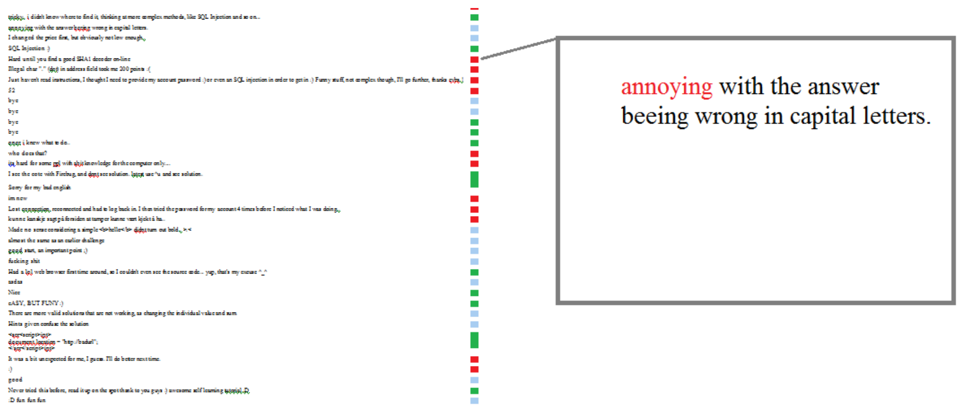
Figure 2.8: Web page site map with focus area

expanded, corresponding to the focus area.

The distinction between overview (or context) area and details (focus) area is important in order for the human mind to process and interpret the visualized data. Eick (1994) explains this with a concept of browser windows and high information density. Browser windows gives a transparent link between the birds-eye view (overview) and the detailed data, while high information density gives the ability to display large amount of data on a standard screen.

Keim (2002) classifies visual data mining techniques based on three criteria; *data type to be visualized*, *visualization technique* and *interaction and distortion technique*. Keim writes that the three criteria are orthogonal, meaning that they can be used in any combination.

hACME game uses a database with tables, varying number of variables and dimensions. All of the data corresponds to results from playing the game, such as challenge playtime, hints used and number of challenges played. The number of attributes differs from the various types of data.



tricky.. i didn't know where to find it, thinking at more complex methods, like SQL Injection and so on..
annoying with the answer beeing wrong in capital letters.
I changed the price first, but obviously not low enough..
SQL Injection :)
Hard until you find a good SHA1 decoder on-line
Illegal char "." (dot) in address field took me 200 points :(
Just haven't read instructions, I thought I need to provide my account password :) or even an SQL injection in order to get in :) Funny stuff, not complex though, I'll go further, thanks cybu.)
52
bye
bye
bye
bye
once i knew what to do..
who does that?
its hard for some ppl with shit knowledge for the computer only....
I see the code with Firebug, and dont see solution. latest use ^u and see solution.
Sorry for my bad english
im new
Lost connection, reconnected and had to log back in. I then tried the password for my account 4 times before I noticed what I was doing..
kunne kanskje sagt på forsiden at tamper kunne vært kjekt å ha..
Made no sense considering a simple <b>hello</b> didnt turn out bold.. >.<
almost the same as an earlier challenge
good start, an important point ;)
fucking shit
Had a lol web browser first time around, so I couldn't even see the source code... yup, that's my excuse ^_^
asdas
Nice
eASY, BUT FUNY :)
There are more valid solutions that are not working, as changing the individual value and sum
Hints given confuse the solution
<img><script></img>
document.location = "http://badurl";
</img></script></img>
It was a bit unexpected for me, I guess. I'll do better next time.
:)
good
Never tried this before, read it up on the spot thank to you guys :) awesome self learning tutorial :D
:D fun fun fun

annoying with the answer
beeing wrong in capital letters.

Figure 2.9: Example of a graphical display of text

Some data are one-dimensional, such as the number of hints used, while other are multi-dimensional. Some data are non-numerical, such as textual comments and feedback from students. The attributes or variables of a data set is referred to as dimensionality of the data (Camastra, 2003).

There is a wide variety of visualization techniques. Keim (2002) mentions standard 2D/3D displays, geometrically transformed displays, icon based displays, dense pixel displays and stacked displays. These are just a few examples.

According to Keim (2002) it is necessary to use an interaction and distortion technique in order to get an efficient data exploration. With interaction technique the analyst is able to change the data dynamically and interact with the visualization. With distortion technique the analyst is able to change the focus of the view.

In line charts, interactive zooming have been used in Stolte et al. (2003) and Eick (2000). Zooming should be used when it is important to present the data in a highly compressed form to provide an overview, but to also give a variable display over data on different resolution (Keim, 2002). An example of this technique is shown in figure 2.10. Here the left view is an overview (or context view) and the right view is a detailed view (or focus view). When the mouse pointer (the magnifying glass) hovers a part of the graph in the overview, the focus view gets updated with a more detailed view on that section. Another example of interactive zooming can be seen in figure 2.9. In this view, the textual comments are enlarged in the right view.

When dealing with textual data, interactive filtering can be applied to decrease the size of the focused data. This could be applied to the example

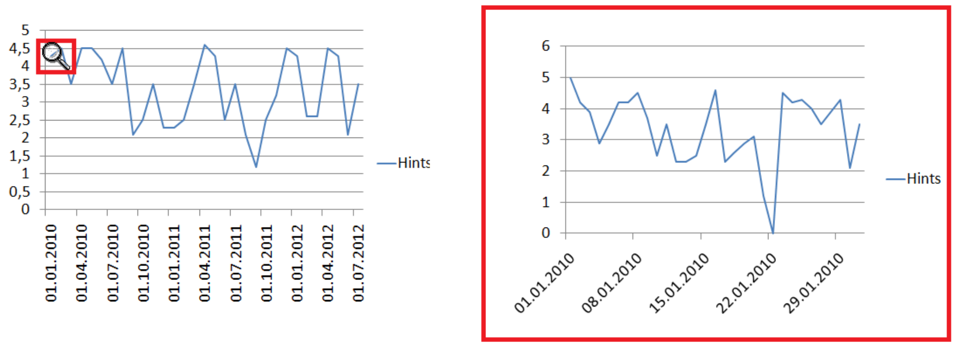


Figure 2.10: Zoom and filter technique used in line chart

shown in figure 2.9 by adding a text filter where administrative users could exclude all texts that does not contain a specific word. Another example of interactive filtering would be to only show positive comments (green comments in figure 2.9).

Level	Challenge	Name
1 - Warm Up	1.1	The novice test
	1.2	A slightly more interesting challenge
	1.3	Confirm order
	1.4	Random number puzzle
	1.5	Digging journalism
	1.6	The Garden of Eden
2 - Lust	2.1	Are you still with us?
	2.2	Accessing secret information
	2.3	We Make Web pages Inc
	2.4	Free pets
	2.5	Vladimir Vladimirovitsj Putin's guest book
3 - Avarice	3.1	Lottery Lothario
	3.2	VIP
	3.3	Still hanging in there?
	3.4	Text analysis was your favorite subject at school
	3.5	George Bush's guest book
4 - Envy	4.1	Ship some pets far out
	4.2	Doctor Online for humans only
	4.3	Access the exam
	4.4	The break-in
	4.5	My Sick Book Face
	4.6	Industrial espionage
5 - Sloth	5.1	iBay - free toilet paper
	5.2	Breaking the barrier
	5.3	SMS tool
	5.4	FaceSpace - forum spammers
	5.5	FaceSpace - a space for your face
6 - Gluttony	6.1	Terminate the register of taxpayers
	6.2	iBay - free bazooka
	6.3	Swedian sends you high up in the sky
	6.4	Breaking FaceSpace
	6.5	FaceSpace - write on my floor
7 - Pride	7.1	Industrial espionage - continued
	7.2	Creating the mail of death
	7.3	Photo Knutson
	7.4	Illuminati's secret web
	7.5	Farm Automagical Systems INC
8 - Wrath	8.1	The poet
	8.2	Ultra quick index search
	8.3	All your emails are belong to us
	8.4	Extraterrestrial contact
	8.5	Fresh Fish Online

Table 2.2: Challenges in second version of hACME game.

Chapter 3

Requirements

This chapter is intended to give a complete description of the system developed. It includes use cases and requirements for the administrative view in hACME game. The chapter is loosely based on the IEEE830-1998 standard, IEEE (1998b).

3.1 Introduction

This section contains an overview and a scope of the requirement specification for the administrative interface.

3.1.1 Purpose

The purpose of this software requirement specification for hACME game administrative interface is to give a complete description of the system.

3.1.2 Scope

The scope of the specification is the hACME game administrative statistics panel, tool panel and questionnaires.

3.1.3 Reference

References used:

- Berrum and Johnsen (2010)

3.1.4 Overview

The chapter starts with an overall description of the system with use cases and system characteristics. Later, specific requirements are presented.

3.2 Overall Description

This section of the requirement specification describes the general factors that affect the administrative interface and its requirements. The intention is to make the requirements more easy to understand.

3.2.1 Product Perspective

The administrative interface is a part of hACME game and is not self contained. It is an administrative module to the existing game. The module is also dependent on the database and the application server.

3.2.2 Detailed Requirements

This section provides a summary of the functions that the administrative interface will perform. It will discuss major functional areas without mentioning any of the large amount of detail that goes with those areas. First, the *statistics panel* is presented, then the *tool panel* and finally, the *questionnaires*. This section is a brief account of chapter 5 of Berrum and Johnsen (2010).

Administrative Statistics Panel

The administrative statistics panel permitts the administrative user obtain feedback on the users progression in hACME game. The administrative user will be able to see which challenge is particularly difficult and which is not. This makes it easy to identify which challenge should be early in the game and which should be late. The statistics panel also lets the administrative user browse through comments, hacking attempts and user info.

The system overview, or home view, is the first page the administrative user will see in the statistics view. Here, the administrative user can choose to view statistics about users, challenges, attempts or comments. Figure 3.1 shows all options for the administrative user in the statistics home view.

Figure 3.2 shows a use case for the challenge view. This view will be opened when the user clicks on challenges from the home view.

Here, the administrative user will be able to see an overview of challenge progression or quick navigate to both attempts and comments view. Quick navigation is to open a different view with the challenge preselected. So, by quick navigating to comments from challenge view, the comments on the selected challenge will be shown by default when entering the comments view.

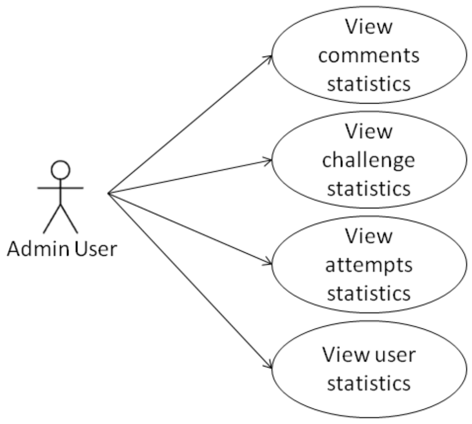


Figure 3.1: Use case showing options in statistics home view.

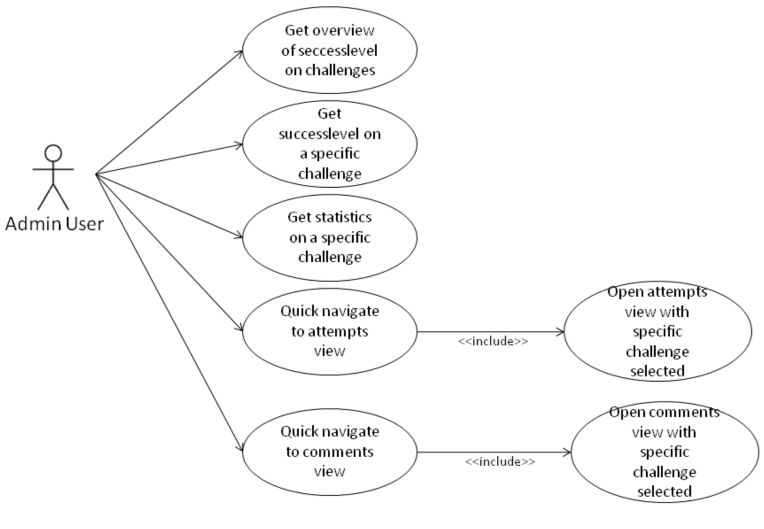

Admin User
Get overview of seccesslevel on challenges
Get successlevel on a specific challenge
Get statistics on a specific challenge
Quick navigate to attempts view
<<include>>
Open attempts view with specific challenge selected
Quick navigate to comments view
<<include>>
Open comments view with specific challenge selected

Figure 3.2: Use case showing the challenges view

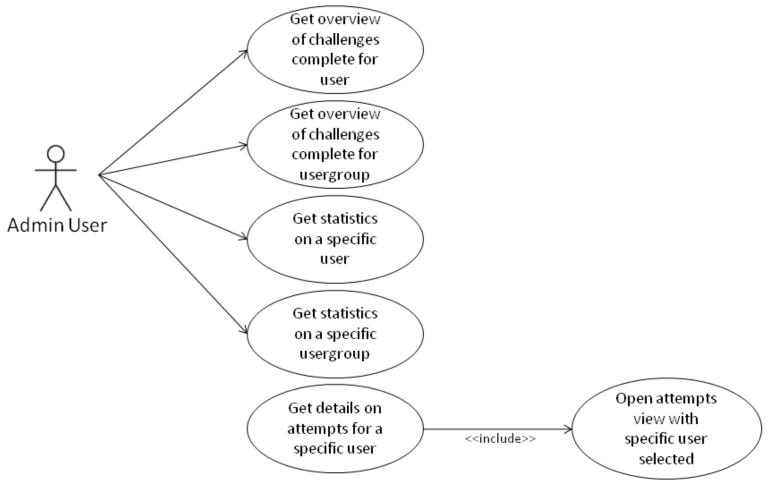

Admin User
Get overview of challenges complete for user
Get overview of challenges complete for usergroup
Get statistics on a specific user
Get statistics on a specific usergroup
Get details on attempts for a specific user
<<include>>
Open attempts view with specific user selected

Figure 3.3: Use case showing the users view.

Figure 3.3 shows a use case for the user view. The administrator can navigate to this view through the home view. The user panel gives the administrator an overview of completed challenges for a specific user or user group. The view can also be used to give information on attempts submitted by a user or a group of users.

Figure 3.4 shows a use case for the comments view. The administrator can navigate to the comments view through the home view, or quick navigate through the challenge view.

The comments view gives the administrator an overview of comments submitted by users. Here, all comments on a specific challenge is shown. The administrator will be able to filter comments according to predefined rules created in the filter dialog. The filter dialog can be opened and edited from this view. When using filters, comments will be marked with a color according to the rules defined in the filter dialog. The administrator can step through the comments one by one or view several at once.

Figure 3.5 shows a use case for the attempts view.

The attempts view gives the administrator statistics on submitted attempts on challenges. Each challenge can be selected and the administrator can then view details on that challenge. The details consist of the distribution of successful and unsuccessful attempts.

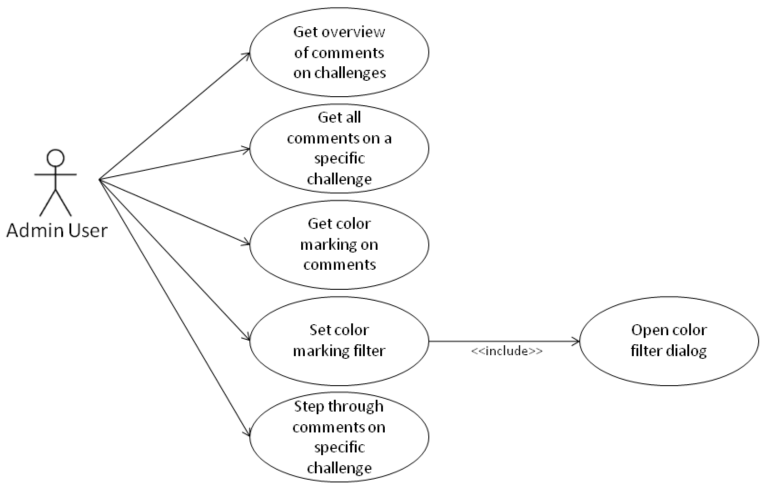

Admin User
Get overview of comments on challenges
Get all comments on a specific challenge
Get color marking on comments
Set color marking filter
<<include>>
Open color filter dialog
Step through comments on specific challenge

Figure 3.4: Use case showing the comments view.

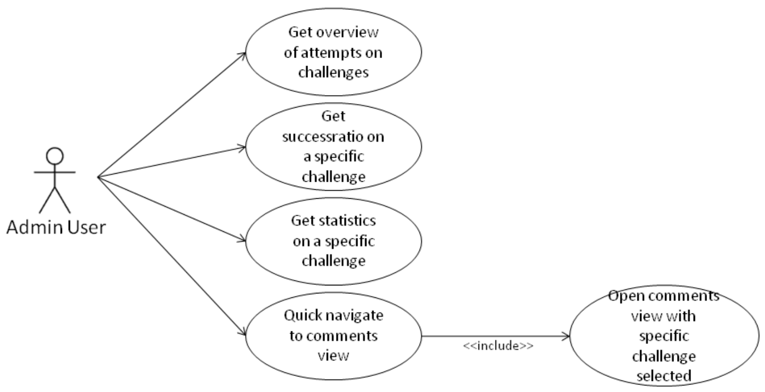

Admin User
Get overview of attempts on challenges
Get successratio on a specific challenge
Get statistics on a specific challenge
Quick navigate to comments view
<<include>>
Open comments view with specific challenge selected

Figure 3.5: Use case showing the attempts view.

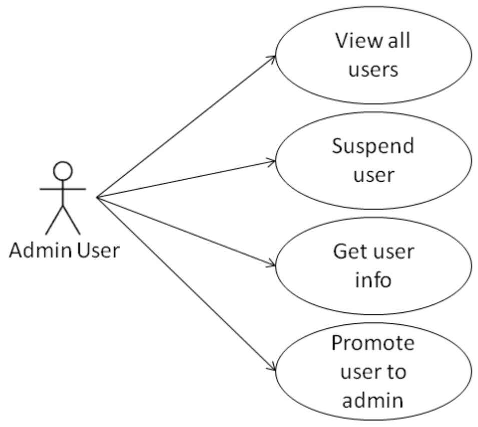


Figure 3.6: Use case showing the tools user view.

Administrative Tool Panel

The administrative tool panel will be a helpful asset for the administrator to make minor changes to the game in an easy manner. This will be also be a practical tool to add/remove and change news, but the most important part are to be able to change different challenges based on statistics. This means that the administrator can first look at the statistics and if it is found that some challenges should be placed earlier or later in the game because of difficulty, the administrator can change this. It should also be possible to make challenges optional or mandatory.

The tools user panel gives functionality for administering users. This includes suspending users from the game. This can be done if a user is cheating or using the game in a manner that is not intended (e.g conflict with NTNU's IT-regulations). When a user is suspended from the game, his or hers account will be disabled, but not deleted from the database. The user will not be able to log in and continue to play. Other actions that can be performed from the tools user panel is to view a list of all users, and get details about a specific user (e.g date of account creation, levels completed etc.). From the tools user panel there is also a possibility to promote users to administrators. A use case for the tools user panel is shown in figure 3.6.

The tools level panel is a tool for changing the sequence of challenges in hACME game. The panel gives functionality for viewing the current challenge setup and to do changes. This includes removing or making a challenge optional, and change the sequence of challenges.

It is possible to remove challenges from a level or add a challenge that has previously been removed to a level. The use case for the tools level panel

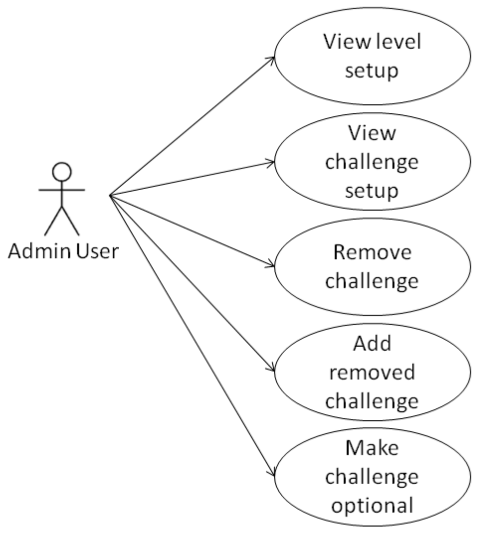


Figure 3.7: Use case showing the tools level view.

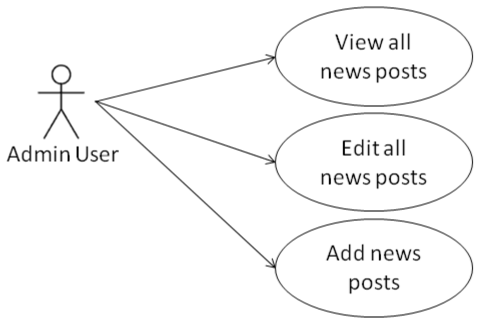


Figure 3.8: Use case showing the tools news view.

can be seen in figure 3.7.

The tools news panel is a tool for managing the news posts in hACME game. When a user is logged in to hACME game, the welcome page shows latest news. The administrator can use the news tool for adding or editing news posts. The use case for the tools news panel is shown in figure 3.8.

The tools site statistics panel is used for viewing miscellaneous statistics about the hACME game web site. This includes the visitor statistics, the count of registered users and latest changes to setup. This use case can be seen in figure 3.9. The difference between this statistics view and the statistics tool is the nature of the statistics shown. This view is for showing non-learning related statistics, such as the count of visitors, while the statistics tool is centered around the student learning aspect of the game.

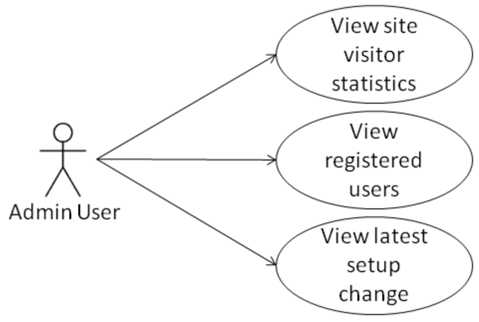


Figure 3.9: Use case showing the tools site statistics view.

Questionnaires

There is possible to add tests on the theoretical knowledge of the users. By adding this, it is possible to not just test how well the users know the basic theory of what they are trying to apply, but also be able to separate different user groups based on results from these tests. This will be an addition to what is already collected about user information, namely knowledge level and grade. This information could serve as a basis for the statistics tool. An example of this is the *successlevel* metric. Currently, this is just the number of successful attempts divided by total attempts for a given challenge. If there was a possibility to know the knowledge level for a given user, it would be possible to take this into account when deciding the *successlevel* for a given challenge. By obtaining information about the knowledge level, it would also be possible to get statistics on how the correlation between knowledge level and challenge progression is.

The questions should be treated as part of a level. When a user completes a survey, he or she should be rewarded with points such as any other challenge. This gives the user the impression that the questions are part of the level and he or she can see a progression in their game play. The user is also rewarded with points, which gives the user motivation to continue playing, while the administrators gets valuable knowledge of the user.

3.2.3 User Characteristics

This subsection describes the general characteristics of the users that will influence the requirement specification. There are two types of users in hACME game, *users* and *admins*.

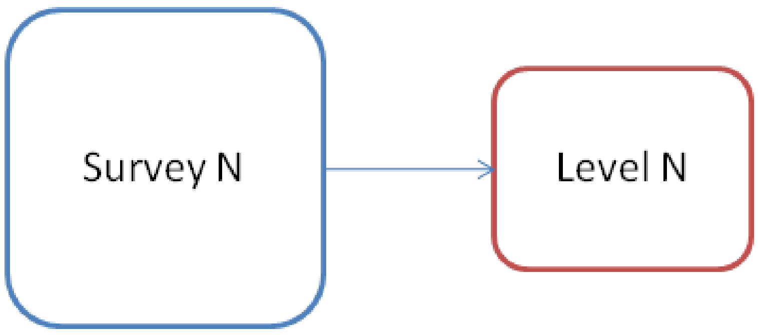


Figure 3.10: Surveys as part of hACME game.

User

This is the regular user who plays hACME game. This user may or may not have experience with software security, but wants to learn or test his or hers skills. The frequency and time spent playing the game will vary from user to user.

Admin

This is the administrative user. This user is concerned with administrating and maintaining the game so that the regular users can continue to play. This user has all rights in the system.

3.2.4 Constraints

hACME game is based on free software and all libraries and other software must have sufficiently flexible licenses to allow free usage of the application.

3.2.5 Assumptions and Dependencies

hACME game is dependent on a database and a application server to run properly.

3.3 Specific Requirements

This section contains the specific of the hACME game administrative interface. The requirements for the administrative statistics panel is first presented, then the administrative tool panel and finally, the requirements for the questionnaires.

3.3.1 External Interfaces

This section contains description of interfaces used by hACME game's administrative interface. First, the user interface requirements are explored, then the software interfaces.

User interfaces

The user interface is important to create a pleasant user experience when using the administrative interface. The interface needs to be consistent with the existing hACME game interface and use the same layout and color scheme. There can be no need for extra training or documentation needed in order to use the interface. All unnecessary interactions should be avoided.

Software interfaces

This section contains a brief summary of all software application frameworks that are required to run the hACME game and the administrative interface. Common for all of the software are that they are free of charge, open source and publicly available.

- **Hibernate**
  This is a storage and retrieval framework of Object/Relational Mapping for Java. Hibernate's primary task is to map Java classes to relational tables.
- **Spring Framework**
  This is a platform for building and running Java applications in a more efficient manner.
- **Spring Security**
  This is a system for providing authentication, authorization and other security features for Java applications.
- **Site Mesh**
  This is a web page layout and decoration framework used to aid in creating large web sites consisting of many pages, creating a consistent look/feel, navigation and layout scheme.

3.3.2 Functions

This section describes, in final detail, all functional requirements of the hACME game administrative interface.

Administrative Statistics Panel

This section presents the requirements for the administrative statistics panel. First, the attempt statistics requirements is presented in table 3.1, then the challenge statistics in table 3.2, the comment statistics in table 3.3 and finally, the user statistics in table 3.4.

Name	**FR-01 Attempt Statistics**
Summary	It must be possible to get an overview of which challenge students are successfully completing and at which distribution the attempts are successful (*successratio*).
Rationale	An administrative user will often want to get an overview of which challenge is the most difficult or easy. If a challenge is particularly easy, the administrator would perhaps want to put that challenge early in the game, and vice versa for difficult challenges.
Details	- It must be possible to get an overview of attempts on each challenge. - It must be possible to view the *successratio* on a specific challenge. - It must be possible to view comments on a challenge from the attempts view. - It must be possible to get statistics such as failed and successful attempts.
References	Figure 3.5

Table 3.1: FR-01 Attempt Statistics

Administrative Tool Panel

This section presents the requirements for the administrative tool panel. First, the user management requirements is presented in table 3.5, then the news management in table 3.6, the level setup in table 3.7 and finally, the site statistics in table 3.8.

Questionnaires

This section presents the requirements for the questionnaires. The questionnaire requirements is presented in table 3.9 and 3.10.

3.3.3 Performance Requirements

The following is a list of response requirements.

Name	**FR-02 Challenge Statistics**
Summary	It must be possible to get an overview of which challenge students are successfully completing and which they are not.
Rationale	An administrative user will often want to get an overview of which challenge is the most difficult or easy. If a challenge is particularly easy, the administrator would perhaps want to put that challenge early in the game, and vice versa for difficult challenges.
Details	- It must be possible to get an overview of all challenges. - It must be possible to view how students perform on challenges(successlevel). - It must be possible to view comments on a challenge from the challenge view. - It must be possible to view attempts on a challenge from the challenge view. - It must be possible to get statistics such as failed and successful attempts.
References	Figure 3.2

Table 3.2: FR-02 Challenge Statistics

- PR-01 The response times for any administrative task should be within 3 seconds 90 % of the time.
- PR-02 Up to three administrative users should be able to be simoultaniously logged in and do work.

3.3.4 Logical Database Requirements

All data will be saved in the hACME game database. The database allows concurrent access and will be kept consistent at all times. A diagram of the database can be found in Appendix C.

3.3.5 Software System Attributes

The following section contains a list of software system attributes that the administrative interface implementation must comply with.

Reliability

- The system should be running properly 99
- Data that has been saved in the system should be stored in persistent storage.

Name	**FR-03 Comment Statistics**
Summary	It must be possible to get an overview of which challenges are getting a lot of comments and to browse through those comments.
Rationale	An administrative user will often want to get feedback from students playing hACME game. A way of getting this type of feedback is through comments. If a challenge receive a lot of similar comments, the administrative user should use this feedback to improve the game.
Details	- It must be possible to get an overview of comments on each challenge. - It must be possible to view the comments on a specific challenge. - It must be possible to filter comments based on words. - It must be possible to to step through comments one by one.
References	Figure 3.4

Table 3.3: FR-03 Comment Statistics

Availability

- The system should be able to handle all types of input without malfunctioning.
- The web site should run in all common browsers.

Security

- The authorization should be restricted with role permissions.
- The system should use prepared statements and not plain SQL code.
- All user input should be validated.
- The system should not display error messages directly from java or SQL to end user.

Maintainability

- The system should log all administrative tasks performed in the system.
- All source codes should be well documented.

Name	**FR-04 User Statistics**
Summary	It must be possible to get an overview of which challenges users are successfully completing and which cause drop outs.
Rationale	An administrative user will often want to get input on which challenge students are dropping out. If a particular challenge is very difficult and students are quitting before completing the challenge, an administrator might want to change that challenge or put it in a later level.
Details	- It must be possible to get an overview of which challenge student are completing. - It must be possible to view average hints given on a challenge. - It must be possible to view average attempts on a challenge. - It must be possible to view how many students starts a challenge. - It must be possible to view how many students successfully completes a challenge.
References	Figure 3.3

Table 3.4: FR-04 User Statistics

Portability

- All code should be written in Java
- The system should have no operating system dependencies.

Name	**FR-05 User Management**
Summary	It must be possible for and administrator to view user details.
Rationale	An administrative user will often want to get info about an existing user. This might be to get the email address or a user name, but also to promote administrators or revoke administrator privileges.
Details	- It must be possible to view a list of all users. - It must be possible to get details on a specific user, such as user name, date registered, etc. - It must be possible to give a user administrator privileges. - It must be possible to suspend a user. - It must be possible to revoke administrative privileges. - It be possible to get the count of all registered users.
References	Figure 3.6

Table 3.5: FR-05 User Management

Name	**FR-06 News Management**
Summary	It must be possible for and administrator to get an overview of all news posts.
Rationale	An administrative user will often want to change or add a new news post in the system. This might be to inform users about changes in the system or to add a question/answer in the FAQ.
Details	- It must be possible to view a list of all news posts. - It must be possible to get details(data posted, text, etc.) on a specific news post. - It must be possible to hide a news post. - It must be possible to edit an existing news post. - It must be possible to add a news post to the system.
References	Figure 3.8

Table 3.6: FR-06 News Management

Name	**FR-07 Level Setup**
Summary	It must be possible for and administrator to get an overview of all levels and challenges.
Rationale	An administrative user will often want to change the level setup. A easy challenge might be moved from a later level to earlier in the game. An administrator might also want to make a level optional.
Details	- It must be possible to view a list of all challenges in a particular level. - It must be possible to view a list of all removed challenges. - It must be possible to reinstate a removed challenge to a level. - It must be possible to edit the sequence of challenges in a level. - It must be possible to make a challenge optional.
References	Figure 3.7

Table 3.7: FR-07 Level Setup

Name	**FR-08 Site Statistics**
Summary	It must be possible for and administrator to get statistics on visitors and latest administrative changes to the hACME game.
Rationale	An administrative user will often want to get information about visitors to the site, latest administrative changes and latest registered users.
Details	- It must be possible to view a count of registered users. - It must be possible to view a latest change on the site. - It must be possible to view which admin changed the site recently. - It must be possible to information on number of visitors.
References	Figure 3.9

Table 3.8: FR-08 Site Statistics

Name	**FR-09 Questionnaire**
Summary	It must be possible to test students knowledge level before a level is played.
Rationale	For an administrator to gain insight on the knowledge level of a student or a group of students, the use of surveys can be helpful. A score on a survey before a level is played gives valuable information on this.
Details	- It must be possible to get the score on surveys for an administrator.
References	Figure 3.10

Table 3.9: FR-09 Questionnaire

Name	**FR-10 Questionnaire Administration**
Summary	It must be possible for an administrator to maintain the surveys.
Rationale	An administrator must be able to change an survey.
Details	- It must be possible to add questions to a survey. - It must be possible to remove a question from a survey. - It must be possible to edit a question on a survey.
References	Figure 3.10

Table 3.10: FR-10 Questionnaire administration

Chapter 4

Design

This chapter is intended to give an overview of the architecture of the administrative interface for hACME game. It describe each of the components in detail. The chapter is loosely based on the IEEE1016-1998 standard, IEEE (1998a).

4.1 Introduction

This section contains an overview and a scope of the software design specification for the administrative interface.

4.1.1 Purpose

The purpose of this software design specification for the hACME game administrative interface is to give a complete description of the system and its components.

4.1.2 Scope

The scope of the specification is the hACME game administrative statistics panel, tool panel and questionnaires.

4.1.3 Reference

References used:

- Berrum and Johnsen (2010)
- Nerbråten (2008)
- Appendix A

4.1.4 Overview

The chapter starts with an overall design overview. Later, each component is presented with detailed descriptions.

4.2 Design Overview

hACME game administrative interface is an add-on module to hACME game. It's purpose is to permit the administrator detailed insight in player progress and to perform maintenance work.

4.2.1 Design Consideration

The following is a list of design considerations applied when designing hACME game's administrative interface.

- **Compatibility** - The administrative interface should be compatible and interoperative with hACME game. The administrative interface should use the underlying architecture of hACME game and it's components.
- **Extensibility** - The administrative interface shold be added to the exiting hACME game product. The administrative interface should easily be extended by adding other modules to it.
- **Modularity** - The administrative interface should consist of well defined, independent components. This ensures maintainability.
- **Usability** - The administrative interface's user interface should be easy to use and self explanatory. No training or user guides should be needed.

4.3 System Architectural Design

This section describe the architectural design of hACME game and the administrative interface. First, a brief account of the architecture of hACME game is presented. Finally, the architectural decisions of the administrative interface is described in detail.

4.3.1 hACME Game

hACME game is organized as a client-server architecture. An illustration of this can be found in Appendix A. The high level architecture has three tiers. The client tier which represents the client's web browser is the first. The client makes requests to the web server. The web server is represented by the logic tier. It is responsible for serving the client requests and communicating with the database. The data tier represents the database which is responsible for the persistent storage.

The application itself runs on the logic tier. The logic tier is divided into three layers. The layers is organized as a top down structure where the layers only can request services from the layer below. This makes the application more maintainable and extensible, according to Nerbråten (2008).

The layers are:

- Web Layer
- Business Logic Layer
- Data Access Layer

The *web layer* is the presentation layer, and handles data presentation to the users. It handles the communication from the user, including validation and population of business objects from user input, to the preferred controller. The response from the controller is sent back to the view.

The *business logic layer* is the service layer, and handles the requests from the web layer and provides business service. It is responsible for handling communication between the web layer and the data access layer.

The *data access layer* is responsible for storing and retrieving data to the business logic layer. This is mainly focused on communicating and executing transactions to the database.

A more in depth description of hACME game architecture can be found in Nerbråten (2008).

4.3.2 Administrative Interface

The administrative interface should use the same layered structure as hACME game itself. Generally, the web layer should be used for presenting data to the user, the business logic layer for handling requests and the data access layer for persistent storage. A chart of this can be seen in figure 4.1

Data Access Layer

In the data access layer (DA) all the data access object interfaces and implementations resides. This layer communicates with the hibernate system

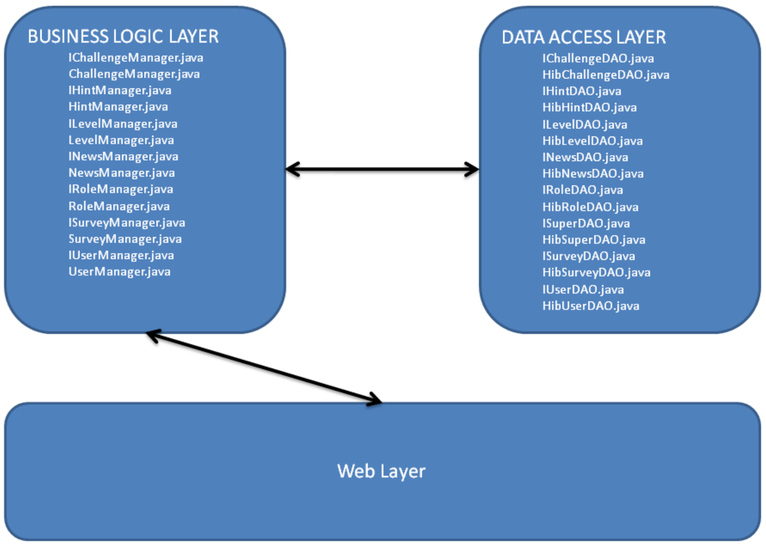


Figure 4.1: The layers of hACME game and it's components.

to get data objects from the database.

The administrative interface users the currently available classes, but should extend the DA with the class *HibCommentDAO.java* and the interface *ICommentDAO.java* in order to retrieve comments from the database.

Business Logic Layer

The business logic layer (BL) consists of a set of specialized data managers. These data mangers will help the web layer retrieving information presented to the users.

The BL should be extended with a comment manager, consisting of the interface *ICommentManager.java* and the class *CommentManager.java*, in order to give data access to comments in the web layer,

Web Layer

The web layer consists of different packages for presenting data to the users. A structure of the web layer can be seen in figure 4.2. The web layer consists of a collection of *Java* and *JSP* files. The java classes supply the jsp-files with data in order to create graphs and other visual elements.

In the administrative statistics view, the data objects should be presented as a Google Virtualization. The Google Virtualization API should be used in order to get an interactive chart system, where the user can click

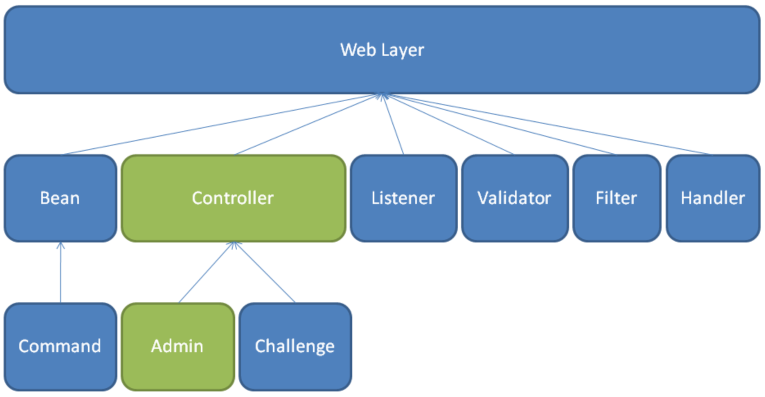


Figure 4.2: The web layer in hACME game.

and hoover charts in order to obtain detailed information.

The administrative tools view should use the data objects in combination with standard HTML controls.

4.4 Detailed Description

This section aims at presenting the general design of each of the components in the administrative interface. Some of the material is a recurrence of Berrum and Johnsen (2010), Chapter 5.

4.4.1 Administrative Statistics Interface

This section describes the administrative statistics interface and its components.

Statistics Home View

The *statistics home* is the first page the user will see after pressing statistics in the navigation bar to the left. Figure 4.3 shows all navigation paths in the administrative interface regarding statistics. The user will view *statistics home* as default and can navigate to either of the four other views from here.

The *statistics home* page is the default administrative page regarding statistics. Here the user can navigate to *user statistics*, *challenge statistics*, *attempt statistics* or *comment statistics*. Each hyperlink to the view is shown in each own grid in figure 4.4.

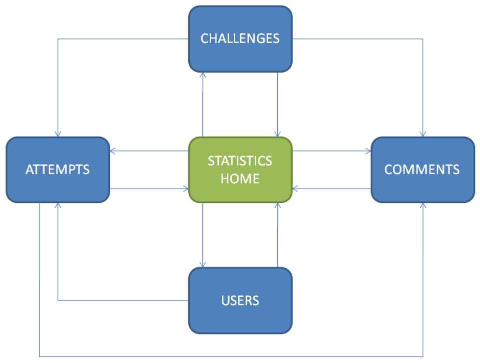


Figure 4.3: Overview of the administrative statistics navigation.

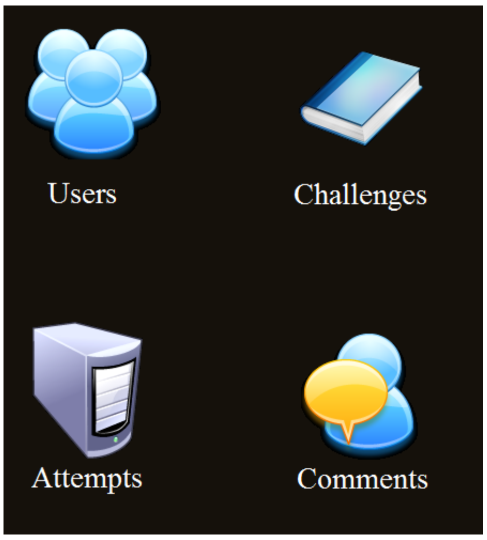


Figure 4.4: Mock up of the statistics home view.

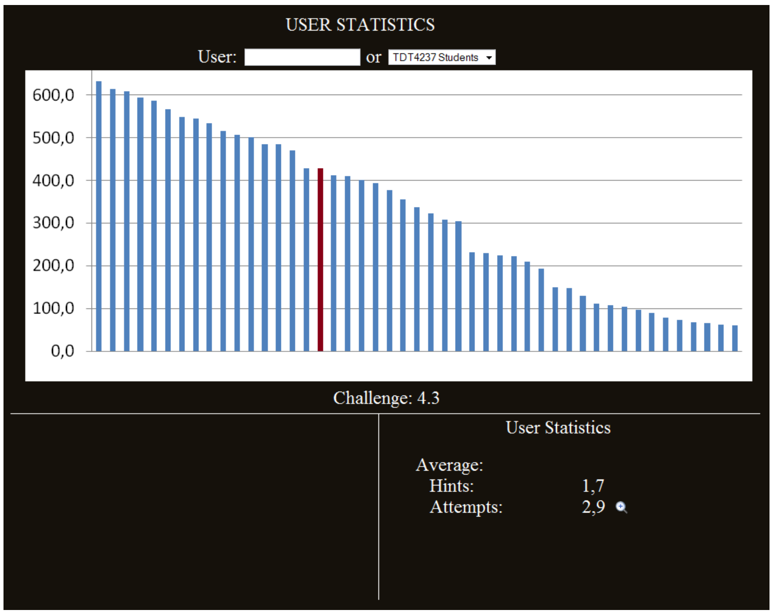


Figure 4.5: Mock up of the statistics user view.

Users View

The *statistics user* view is shown in figure 4.5. This view gives the administrative user the possibility to see how a group of users or a single users perform in the game. It is possible to view trends on where users drop out and how many hints and attempts are used.

The top view gives an overview on how many users complete each challenge. It is possible to select a predefined group of users or a single user. By selecting a challenge, it is possible to see the average hints and the attempts used.

Challenges View

The *statistics challenge* view gives the administrative user an overview of challenge *successlevel* as described earlier in the report. The panel gives the opportunity to select a detailed view from the bar chart by clicking on the desired challenge in the chart. The chart itself shows the *successlevel* of each challenge. By clicking on a challenge, the lower detail view will get updated with statistics on that specific challenge. On the left side of the details view is the *successlevel*. The *successlevel* is a metric defined as the percentage of players that completes the challenge. On the right side, more details on that specific challenge, such as comments and attempts are shown. There are also hyperlinks to navigate directly to the comments and attempts view.

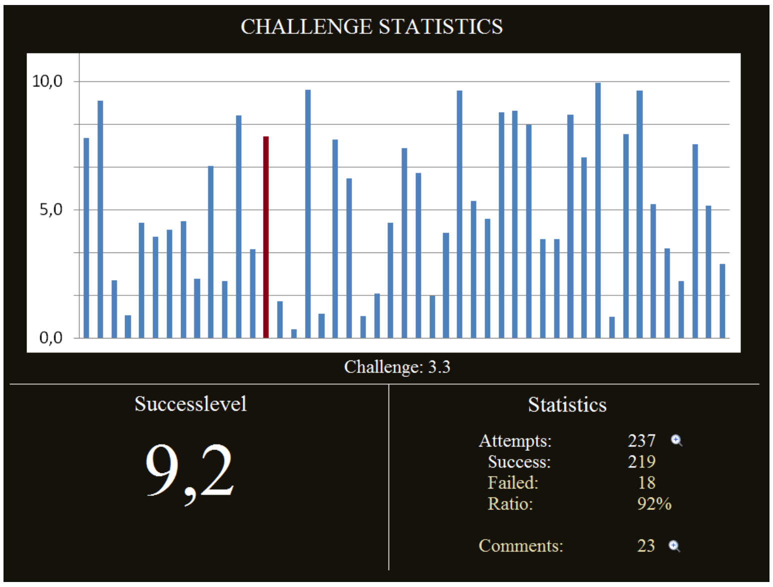


Figure 4.6: Mock up of the statistics challenge view.

By clicking on either on them the desired view is opened, pre configured with the selected challenge in view.

The *statistics challenge* view gives the administrative user an opportunity to quickly get an overview of the *successlevel* of each challenge. This is useful information when an administrator wants to change the challenge progression for a user group or see where users drop out of the game.

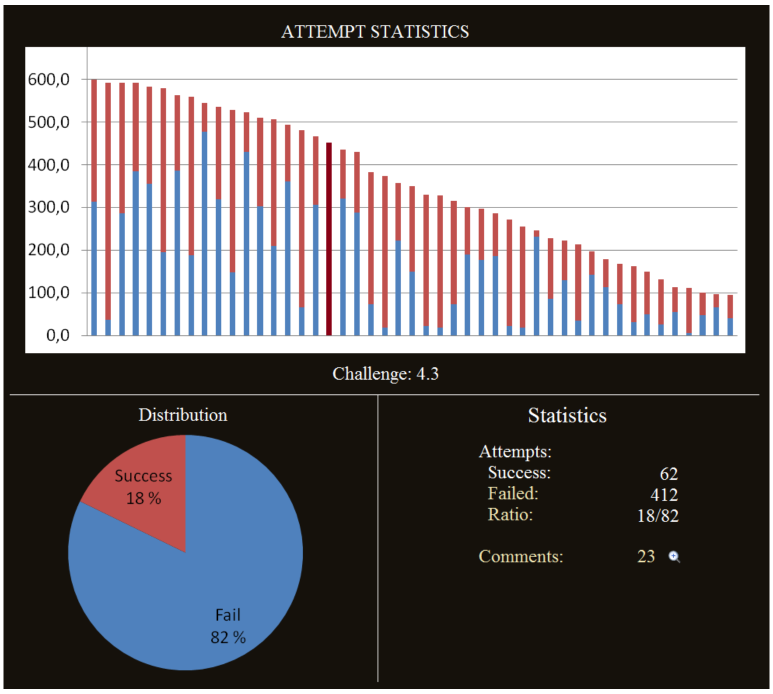
Attempts View

The *statistics attempts* view gives the administrative user an overview of how many attempts is used for each challenge. It is also possible to see success/fail ratio for each completed challenge.

The top view gives an overview of all attempts on each challenge, while the bottom detail view gives details on success-fail distribution and the number of comments. It is possible to navigate directly to comments and attempts view for more details.

Comments View

The *statistics comment* view gives the administrative user functionality for viewing comments in the game. In the overview on the top, the administrator can choose which challenge to view comments from. When a challenge is chosen, it is possible to filter the comments or step through one by one



Attempts:
Success: 62
Failed: 412
Ratio: 18/82

Comments: 23

Figure 4.7: Mock up of the statistics attempts view.

in the details view on the bottom of the page. All comments on a selected challenge is shown in the left side, and the selected comment is shown in the right side. This view is shown in figure 4.8.

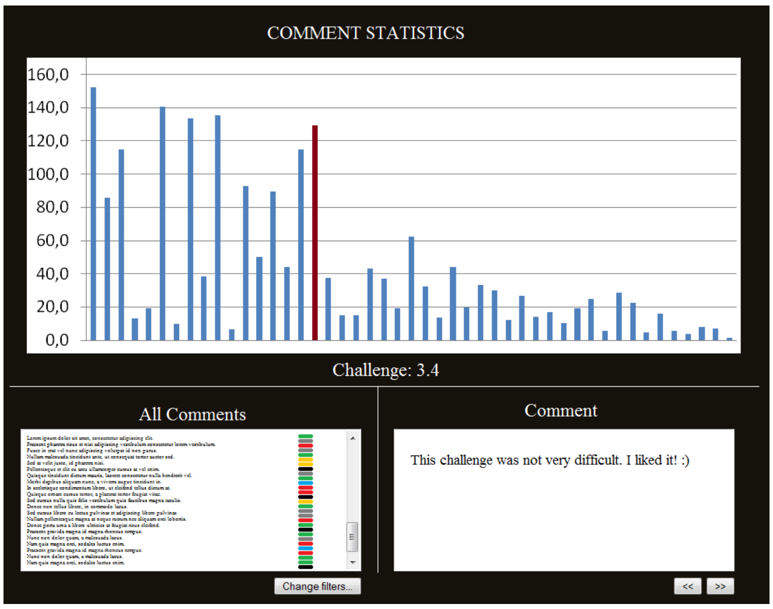
The filtering operates by color marking comments by contents. It is possible to give a class of words color marking, and thereby making it easier to sort comments. By pressing the 'change filters...' button the user gets redirected to a filter setup page, where predefined filters can be applied or new ones can be added.

4.4.2 Administrative Tools Interface

This section describe the administrative tools interface and its components.

Tools Home View

The *tool home* is the first page the user will see after pressing tools in the navigation bar to the left in hACME game's user interface. Figure 4.9 shows all navigation paths in the administrative interface regarding setup tools. The user will view *tools home* as default and can navigate to either of the four other views from here.



This challenge was not very difficult. I liked it! :)

Figure 4.8: Mock up of the statistics comment view.

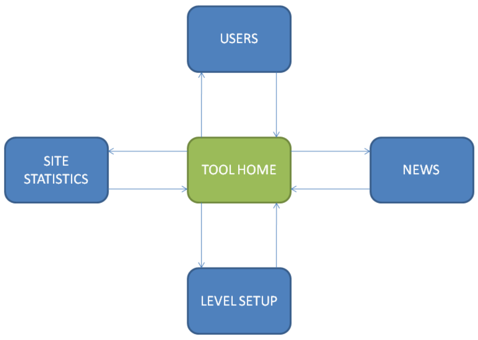


Figure 4.9: Overview of the administrative tool navigation.

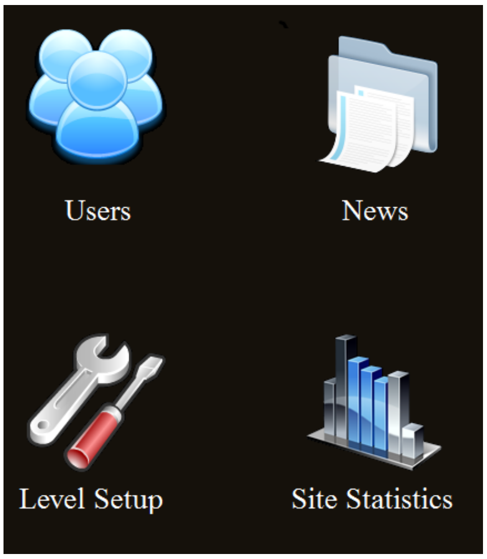
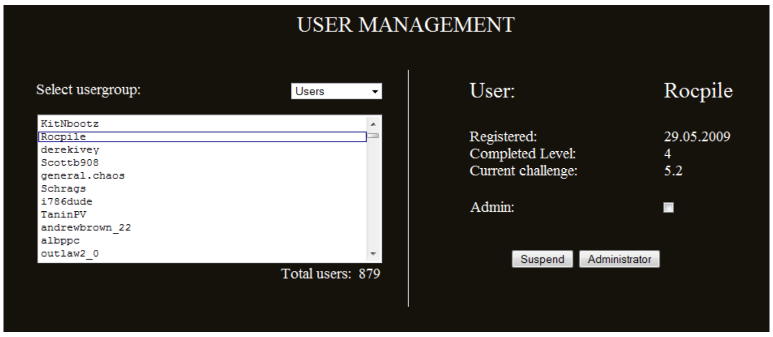


Figure 4.10: Mock up of the statistics home view.



```
KitNbootz
Rocpile
derekivey
Scottb908
general.chaos
Schrags
i786dude
TaninPV
andrewbrown_22
albppc
outlaw2_0
```

Total users: 879

User: Rocpile

Registered: 29.05.2009
Completed Level: 4
Current challenge: 5.2

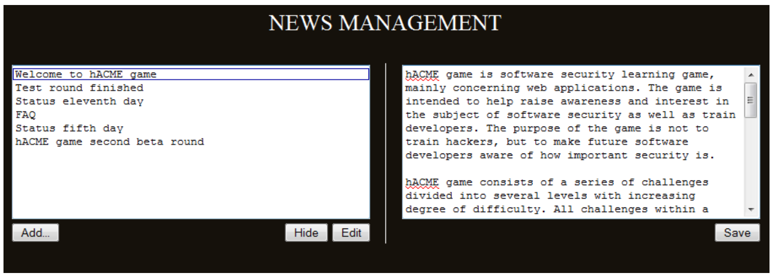
Admin: ☐

Figure 4.11: Mock up of the tools user view.

The *tools home* page is the default administrative page regarding setup tools. Here, the user can navigate to *users*, *news*, *level setup* or *site statistics*. Each hyperlink to the view is shown in each own grid in figure 4.10.

Users View

The user tool allows the administrative user to suspend a regular user from the system. By suspending a user, the user will still be present in the database, but not able to log in an use his or hers account. The user will also be left out of all statistics. The user tool also gives the administrative user the possibility to change administrative rights on other users. This makes it easy to promote a regular user to an administrator, or to get an overview of all administrators.



```
Welcome to hACME game
Test round finished
Status eleventh day
FAQ
Status fifth day
hACME game second beta round
```

```
hACME game is software security learning game,
mainly concerning web applications. The game is
intended to help raise awareness and interest in
the subject of software security as well as train
developers. The purpose of the game is not to
train hackers, but to make future software
developers aware of how important security is.

hACME game consists of a series of challenges
divided into several levels with increasing
degree of difficulty. All challenges within a
```

Figure 4.12: Mock up of the tools news view.

Figure 4.11 displays a mock up of the tools user view. In the left view is the list of users and selection of user group. When a user is selected, details on that user is shown in the rights view. The information includes date registered, last completed level, current challenge and whether the user is an administrator or not. If the usergroup 'Administrators' is selected, the user view lists all administrators.

News View

The news tool permitts the administrative user to add, edit or remove news on the front page of hACME game for logged in users. If an administrative user add a news post, this post will be on top of the front page when a user has logged in. Other pages such as the front page for non-logged in users can also be changed, but not removed.

Figure 4.12 shows a mock up of the tools news view. In the left view is a list of all news posts. Here it is possible to select a news post and edit it in the right view. It is also possible to add or hide a news post.

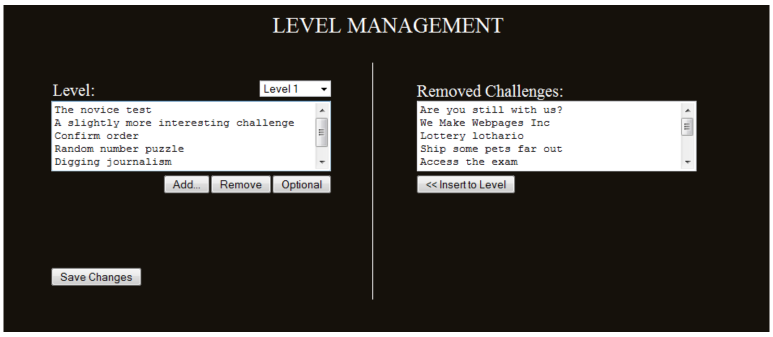
Level Setup

The level tools purpose is to let the administrative user change the course of the game challenges. The administrator can remove challenges from a level or add removed challenges. Challenges can also be made optional.

Figure 4.13 shows the level management tool. In the right view is a list of all challenges in a corresponding level. When a user selects a new level in the drop down, the list updates with challenges. The right view shows all removed challenges. It is possible to add a removed challenge to any level.

Web Statistics

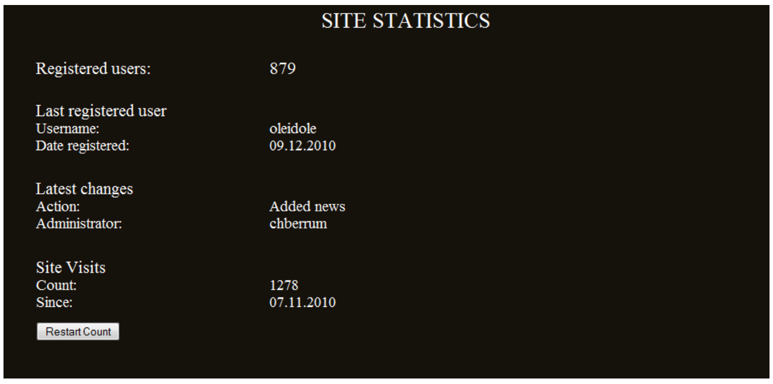
The site statistics tool lets the administrative user view statistics on the visits to hacmegame.org, latest changes to setup and latest registered users.



```
The novice test
A slightly more interesting challenge
Confirm order
Random number puzzle
Digging journalism
```

```
Are you still with us?
We Make Webpages Inc
Lottery lothario
Ship some pets far out
Access the exam
```

Figure 4.13: Mock up of the tools challenge view.



Registered users:	879
Last registered user	
Username:	oleidole
Date registered:	09.12.2010
Latest changes	
Action:	Added news
Administrator:	chberrum
Site Visits	
Count:	1278
Since:	07.11.2010

Figure 4.14: Mock up of the tools statistics view.

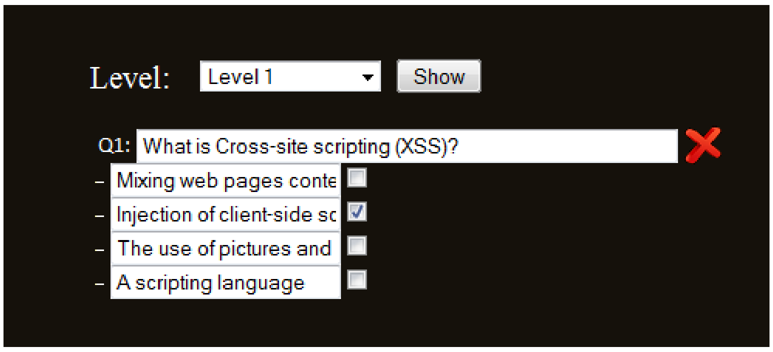

Level:
Level 1
Show
Q1: What is Cross-site scripting (XSS)?
– Mixing web pages conte
– Injection of client-side sc
– The use of pictures and
– A scripting language

Figure 4.15: Mock up of the administrative survey view.

Figure 4.14 shows the tools statistics view. This view contains miscellaneous statistics about the site. From here it is possible to reset a visit counter, view latest change on the site and latest registered user.

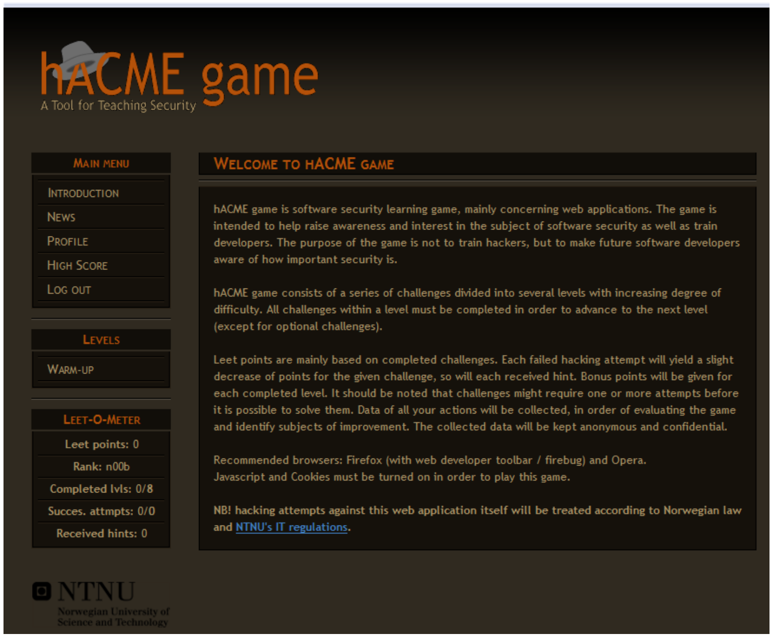
4.4.3 Administrative Queries

Before a student starts on a level, he or she has to answer a questionnaire. This results from the questionnaire will provide background information on the knowledge level of the student. The questions will be related to the contents of the level. If the level contains challenges on cross site scripting, the questionnaire will be about cross site scripting as well. The results from completing a survey will be idetical from the points given from a challenge. However, the data collected from an answered survey, should be used as a basis for the knowledge level of the student. This metric could be used when deciding how well a studen should perform in the game.

In figure 4.15 a mock up of the administrative survey view is shown. Here, an administrator can edit, delete and save questions and alternatives. The adminsitrator first choose a level, then press the show button to show the survey from that level. Then he or she can change the current questions or add new questions. The administrator can use the checkboxes to mark the correct answer. Some questions might have more than one correct answer, this is why checkboxes are preferred, and not radio buttons. When the administrator is done, the survey is saved and updated for that specific level.

4.5 User Interface Design

The user interface of hACME game administrative interface should follow the same conventions as in Nerbråten (2008). The user interface color theme should be a combination of black, dark brown and orange. This creates the



Leet points: 0
Rank: n00b
Completed lvls: 0/8
Succes. attmpts: 0/0
Received hints: 0

Welcome to hACME game

hACME game is software security learning game, mainly concerning web applications. The game is intended to help raise awareness and interest in the subject of software security as well as train developers. The purpose of the game is not to train hackers, but to make future software developers aware of how important security is.

hACME game consists of a series of challenges divided into several levels with increasing degree of difficulty. All challenges within a level must be completed in order to advance to the next level (except for optional challenges).

Leet points are mainly based on completed challenges. Each failed hacking attempt will yield a slight decrease of points for the given challenge, so will each received hint. Bonus points will be given for each completed level. It should be noted that challenges might require one or more attempts before it is possible to solve them. Data of all your actions will be collected, in order of evaluating the game and identify subjects of improvement. The collected data will be kept anonymous and confidential.

Recommended browsers: Firefox (with web developer toolbar / firebug) and Opera.
Javascript and Cookies must be turned on in order to play this game.

NB! hacking attempts against this web application itself will be treated according to Norwegian law and NTNU's IT regulations.

Figure 4.16: Screen capture of hACME game.

desired dark and mysterious look and feel, accoridng to Nerbråten (2008).

An important factor of the user interface is that users should not need any training or user guides. The interface should be intuitive and easy to understand. A screen shot of hACME game can be seen in figure 4.16 A more detailed description of the general user interface can be found in Nerbråten (2008).

The administrative interface should inherit most of the color conventions of the hACME game user interface. The background and text colors should be identical to hACME game's. An exception is the colored graphs, which should use red, green and blue as main colors. Red is used to emphasize failure or negative outcome, while green is success or positive outcome. Blue is used as a neutral color.

The layout of the administrative statistics panel is shown in figure 4.17. The main container is used for overview statistics. Here, the administrator can see all challenges and choose which to get details on. The detailed views(R and L) are used for detailed descriptions of the selected challenge in the main container.

The administrative tools view should use the same color conventions as hACME game, but not the layout as the administrative statistics interface.

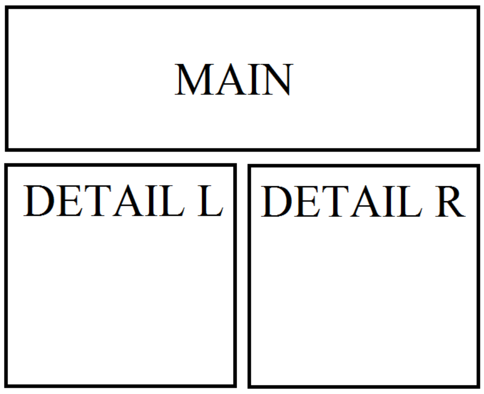


Figure 4.17: Administrative statistics panel layout.

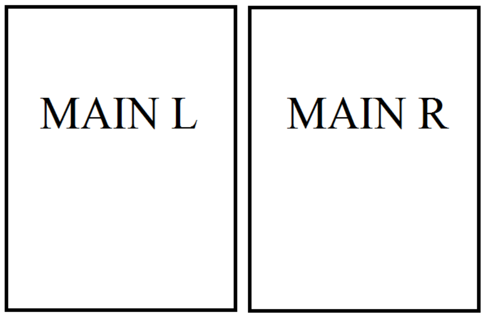


Figure 4.18: Administrative tools panel layout.

The reason for this is the diversity of the maintenance tasks. The different tools interfaces has different layout, depending on the task at hand. Most of the layouts follows the arrangement of figure 4.18. What the containers are used for varies with the different panels.

Chapter 5

Security Analysis

This chapter is intended to give a security analysis of the hACME game's administrative interface. Some of the material is a recurrence of Berrum and Johnsen (2010), Chapter 6.

5.1 Adding an Administrative Layer

No application is more secure than it's weakest level. hACME game consists of several modules and all of them needs to be secure in order for the application to be secure. A module such as the administrative tool will inherit some security aspects of the application itself. However, some security issues needs to be sorted out. This section identifies possible security risks and mitigation strategies for the administrative tool.

Currently in hACME game, there are not any possibilities to do extensive changes in the game through the web interface. It is not possible to change the setup of levels, or do any administrative tasks. These tasks must all be done directly in the database. By adding an administrative interface the administrative user gets the ability to change the workings of the game rapidly and easy. This functionality poses some threats in the means of a regular user obtaining administrative privileges. Figure 5.1 shows the changes in privileges.

On the left side of figure 5.1 is the current hACME game without the administrative interface. In this configuration, the write operations performed from the web view is registration of new users and player progress. In the right side of the figure, the new hACME game, where administrative users can perform miscellaneous changes and maintenance tasks.

By introducing an administrative interface, the administrative privileges actually get separated into two levels. The administrators with access to

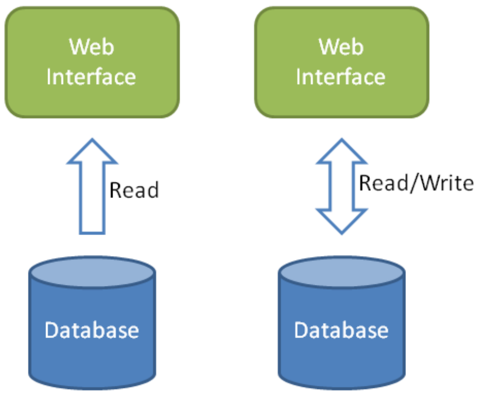


Figure 5.1: Before and after adding the administrative interface

the administrative interface, and the administrator with full database access. The administrator with access to the administrative interface, will get some access to the game setup and maintenance, while the administrator with access to the database will get full administrative privileges. This user can both delete, edit and add data. The administrator with access to the administrative web view will only be able to do changes according to the functionality offered by the administrative interface.

5.2 General Risks

This section contains general risks with web software and their mitigation strategies.

GR-01 Provoking exceptions

This is a logging and auditing vulnerability. Users can gain information about the site by provoking exceptions and fault messages. The information learned by exceptions and fault messages can then be used by an adversary to further attack the website.

Mitigation: Only display fault messages that does not provide any useful information to an attacker. All exceptions should be removed form the user interface and only be logged so that the administrators of the site can see them.

GR-02 Vulnerabilities in APIs

In order to implement the administrative tool, a variety of other APIs and libraries will be used. These APIs and libraries can be insecure and pose a threat to security if they contain vulnerabilities. Another important aspect of API abuse is the correct usage of a APIs. If an API is not used correctly, unintended errors and vulnerabilities might occur. More on this can be found in section 5.4.

Mitigation: When implementing the administrative interface the latest version of APIs is used and updated frequently. Also, the programmers should have sufficient knowledge about the usage of the APIs used.

GR-03 Forced Browsing

A user might try to jump to any page he or she wants by typing a Universal Resource Locator (URL) in the browser's address field. This include regular users trying to open the administrative interface when they are not authenticated as administrative users.

Mitigation: Perform user privilege validation on each page. Also ensure that focus on access control is a part of the whole development life cycle.

GR-04 Code Injection

Code injection or cross-site scripting is a common method for exploiting a web site by injecting unintended web content. This can be used to bypass access controls or injecting malicious scripts.

Mitigation: Perform input validation on all fields with user supplied input. All fields in the administrative interface intended for string values should use a white list of escape and encoding characters.

GR-05 SQL Injection

SQL injection is a code injection technique that exploits the database layer of an application. An attacker injects unintended SQL-code into the database through incorrectly filtered input fields.

Mitigation: Perform input validation on all fields with user supplied input and only use prepared statements and strongly typed user input.

GR-06 Directory Traversal

Directory traversal is a technique for exploiting missing validation of user input, such that sequences representing ”traverse to parent directory” are sent to the API. The goal of a directory traversal attack is to gain access to a file that is not intended to be accessible.

Mitigation: Perform input validation on all fields with user supplied input and enforce validation of user privileges on each site.

GR-07 Input Validation Attacks

Input validation attacks are attacks that focuses on exploiting missing input validation on input fields on a site. This might be in form of buffer overflows, injecting format strings, cause string termination errors and null string injections. An example of this is supplying <*%00script*> instead of <*script*> in an input field in order to inject the <*script*>-tag.

Mitigation: Perform input validation on all fields with user supplied input. The input validations should use a white list of all allowed characters.

GR-08 Command Injection

This attack exploits underlying functionality offered to the web site by other applications such as the operating system the web server is running. By directly supplying the web server environment with input from the web site, an attacker can run commands directly on the web server itself.

Mitigation: Perform input validation on all fields with user supplied input. Also never use user supplied input in any server based resource.

GR-09 Code Quality

Poor code might lead to unexpected behavior and vulnerabilities in the software. An attacker might use this as an opportunity to stress the system in unexpected ways.

Mitigation: Make code quality a priority when implementing the administrative interface.

5.3 Administrative Layer Risks

This section contains risks and mitigation strategies specific to the administrative layer. The risk analysis first discuss how a regular user can gain access to the administrative layer and then how the administrative layer can be used for harmful actions.

5.3.1 Access to the Administrative Layer

This section contains a discussion on how an attacker can get administrative privileges in hACME game. First, consider the attack tree in figure 5.2.

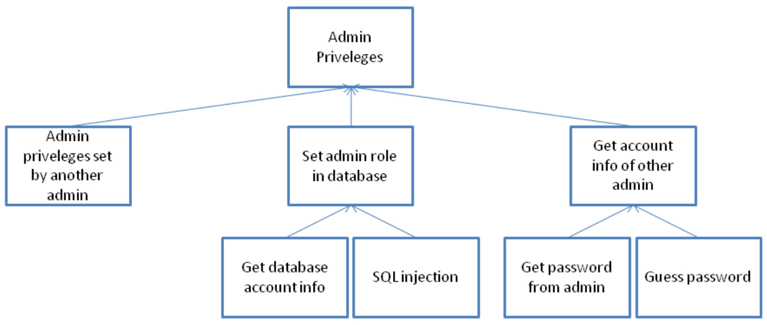

Admin Priveleges
Admin priveleges set by another admin
Set admin role in database
Get account info of other admin
Get database account info
SQL injection
Get password from admin
Guess password

Figure 5.2: Attack tree of getting administrative privileges

AR-01 Admin Privileges by Another Admin

In the tools user panel an administrator can promote another user to administrator and thereby giving the other user all administrative privileges. This is the the way it is intended to give a user administrative privileges. It is done by having an existing administrator promoting another administrator.

Mitigation: To promote another user to administrator is not a threat by itself. By promoting the wrong user, is however a threat to the system. All administrators should be well aware of this potential threat.

AR-02 Admin Role in Database

It is possible to add a role as an administrator in the database. This can either be done by logging in to the database directly or with help from an SQL injection. If a user gets the account information about the database it is simple to log in directly an add administrative privileges on a user. The other way of doing this is by a SQL injection. By successfully inserting the queries in Listing 5.1 a user will get administrative privileges.

```
SELECT id FROM `user` WHERE nickname = 'olanordmann';

INSERT INTO `g_hacmegame`.`user_role` (`user_id`, `
    role_id`) VALUES ('10', '1');
```

Listing 5.1: SQL Query for admin privileges

The first query will return a user id from the database. This is used for identifying the users when the second query is performed. Here, the user id and user role must be known. The role id is an integer for identifying the role. The user role id for an administrator is 1.

Mitigation: The application should use input validation on all user supplied input. This will ensure that SQL injection is difficult. The existing administrators must be well aware of the threat of sharing the username and the password to the database. The credentials should also be strong enough to not be guessed or fuzzed, Sutton et al. (2007).

AR-03 Account Information of Other Admin

It is possible to get administrative rights without adding the role as an administrator in the database or have another administrator promote you. The third way is by getting the account information of an existing administrator and log in as him or her. This can be done by getting the account information directly from the admin or a brute force attack on his or her account.

Mitigation: The administrators must be well aware of the potential threat of not having strong login credentials and revealing these credentials to others.

5.3.2 Administrative Layer Threats

The previous section discussed how an attacker can get administrative privileges in hACME game. This section explores what kind of harmful behavior a regular user or attacker with administrative privileges can do. We start by listing a variety of harmful behavior that is possible to do from the administrative layer and their mitigation strategies. The analysis is mainly centered around the administrative tool panel. This is because the administrative statistics panel is only reading from the database and is more vulnerable from the attacks mentioned in section 5.2. The administrative tool panel is also vulnerable from these attacks, but additionally the risks mentioned below.

Tools User Panel

This section describes potential harmful actions an administrator can execute in the *tools user panel* and corresponding mitigation strategies.

TR-01 Suspend Users

An administrator can suspend all regular users (not administrators) in hACME game. If an administrator wants to harm the game, he or she can suspend all regular users. In this way, they can not log in and continue playing.

Mitigation: Only let an administrator suspend regular users from the web interface, and not administrators. Also, only suspend the users, and not delete them, so that they can be restored if an unintended suspension occurs.

TR-02 Get User Information

An administrator can get personal information from all users in hACME game. This include full names and email addresses. The list of users can be used for unintended purposes such as advertisements, spam and id theft.

Mitigation: Only store the necessary personal information. It should not be possible to easily download all email addresses or other personal information about users from the administrative interface.

TR-03 Promote Administrators

An administrator can promote other administrators and give them the same privileges. If an administrator wants to harm the game, he or she can promote all users to administrators.

Mitigation: It should be possible to revoke administrative privileges from administrators. It should also not be possible to mass promote administrators.

Tools Level Panel

This section describes the potential harmful actions an administrator can execute in the *tools level panel* and corresponding mitigation strategies.

TR-04 Remove Challenges

An administrator can remove challenges from the game. If he or she wants to, it is possible to remove all challenges from the game, and make it useless.

Mitigation: Only let administrator remove challenges from levels, not delete them from the database. In this way, they can be restored back in the level.

TR-05 Change Level Setup

An administrator can change the level setup. It is possible to put all the difficult challenges at level 1 and thereby making the game a lot harder to play for the users.

Mitigation: Make it possible to revert the level setup.

TR-06 Make Challenge Optional

An administrator can make challenges optional. If an administrator make all challenges optional, then the game is not the way it was intended to be played.

Mitigation: Make it possible to revert all optional challenges to mandatory challenges.

Tools News Panel

This section describes potential harmful actions an administrator can execute in the *tools news panel* and corresponding mitigation strategies.

TR-07 Add News Post

An administrator can add new news posts to the game. This news post can be spam, advertisements or phishing attacks.

Mitigation: A news post can be set to 'hidden' in the tools panel. It should also be possible to edit an existing news post and remove harmful content.

TR-08 Edit Existing News Post

An administrator can edit an existing news post. This news post can be spam, advertisements or phishing attacks.

Mitigation: It should also be possible to edit an existing news post and remove harmful content.

Tools Statistics Panel

This section describes potential harmful actions an administrator can execute in the *tools statistics panel* and corresponding mitigation strategies.

TR-09 Clear Visitor Count

An administrator can clear the the counter of visited users and reset the data to the current day.

Mitigation: The visitor counter should reside in persistent storage so it can be restored from database.

5.4 Application Programming Interfaces

This sections contains information about the use of APIs in hACME game's administrative interface. The discussion is mainly centered around the use of the Google Chart API. OWASP (2011), describes an API as a contract between a caller and a callee and argues that the most common problem with the use of APIs is the caller not honoring it's end of this contract.

5.4.1 Google Chart Tools

The Google Chart Tools is an API used for adding visualization to a web page. In hACME game, it is used as the primary visualization tool for presenting statistical data to the administrative user. Listing 5.2 gives an example of how the API is used.

```
//Load the Visualization API and the ready-made Google
    table visualization
google.load('visualization', '1', {'packages':['
   corechart']});

// Set a callback to run when the API is loaded.
google.setOnLoadCallback(init);

function init() {

// Get data
var successAttempts = <c:forEach var = "a" items = "${
   successAttempts}" varStatus = "s">${a}${s.last ? ""
    : ","}</c:forEach>;

// Creating datatable
dataTable = new google.visualization.DataTable();
dataTable.addColumn('string', 'Challenge');
dataTable.addColumn('number', 'Successfull attempts');

// Adding data
for (i=0 ; i<=successAttempts.length ; i++){
        var att = successAttempts[i];
        dataTable.addRow(['Challenge x',att]);
}

// Draw the visualization.
var chart = new google.visualization.ColumnChart(
   document.getElementById('chart_div'));
chart.draw(dataTable, {width: 600, height: 400, is3D:
   true, isStacked:true, title: 'Users completed per
   challenge'});
}
```

Listing 5.2: An example of Google Chart Tools usage in hACME game

5.4.2 API Abuse Vulnerabilities

This section gives a brief overview of possible vulnerabilities related to API Abuse according to OWASP (2011). Each of the vulnerabilities are mentioned with the corresponding mitigation activities.

- **Dangerous Functions**
  Certain functions behave in dangerous ways. These functions were often implemented without the concern of security.

  Mitigation: Functions that can not be used safely should not be used. All functions should be used according to the API specification.

- **Directory Restriction Error**
  Improper use of directory restrictions may allow attackers to gain access to unintended files and functions. This may include password or configuration files.

  Mitigation: Do proper access control when directories, files or relative paths are used.

- **Failure to follow guideline/specification**
  This is a collection of vulnerabilities where the API specification is not followed. This can cause a variety of errors and dangerous situations. This include functions not being called in correct order, unintended variable usage or calls for unsafe functions.

  Mitigation: Follow the specification or guidelines for the API.

- **Heap Inspection**
  Sensitive data stored in memory may be leaked if it is stored as a managed String object. The garbage collector can relocate these objects and leave copies in memory.

  Mitigation: Do not store sensitive data (such as passwords) as String objects in memory.

- **Ignored function return value**
  Some functions use return values as indicators of occurring errors. If a function fails, it may return a false value. If this indicator is not checked the function could have failed without any warning.

  Mitigation: When using functions, if possible, use the return value as an indicator of whether the function executed properly or not.

5.4.3 Google Chart Tools Policy

When using a chart tool API some data collection by the chart tool provider may happen. Google states the following in their chart tool policy, Google (2011).

"The chart data included in the HTTP request is saved in temporary logs for no longer than two weeks for internal testing and debugging purposes. Of course you should understand that if your chart appears in an image tag on a public web page, it could be crawled."

"You may not store any personal information in the gadget or visualization. You may not collect sensitive personal information such as credit card numbers and social security numbers through a gadget or visualization."

Two main approaches can be used in order to minimize insight in hACME game data from Google. First, the data sent to Google should mainly be arrays of numbers. The numbers correspond to the data in the chart tool. Even though Google log these data for 2 weeks, no meta-data can be read from the arrays. Secondly, no personal information should be shown in the chart tool. This ensures that even if Google has a leakage of data, no sensitive or meaningful data can be obtained from the hACME game administrative tool.

5.5 Risk Assessment

This section contains two risk maps. The risk maps contains assessments of all identified risks in the previous sections.

There are a lot of risks involved in adding an administrative web interface. Not only is the interface vulnerable to the general risks listed in section 5.2, but there are also a lot of additional risks when users can modify the database from a web interface. The most important risk mitigation is to be able to revert changes and not delete anything completely from the database. Even though it is possible to suspend users or remove challenges from levels in the tool panel, complete deletion from the database must still

	Likelihood			Impact			Risk		
	L	M	H	L	M	H	L	M	H
GR-01	X			X			X		
GR-02	X				X		X		
GR-03		X			X			X	
GR-04		X				X		X	
GR-05			X			X			X
GR-06		X			X			X	
GR-07		X			X			X	
GR-08	X					X		X	
GR-09	X					X		X	

Table 5.1: A risk map of the general risks in the administrative tool.

	Likelihood			Impact			Risk		
	L	M	H	L	M	H	L	M	H
TR-01		X				X			X
TR-02	X			X			X		
TR-03	X				X			X	
TR-04		X				X			X
TR-05		X				X			X
TR-06		X			X				X
TR-07	X			X			X		
TR-08	X			X			X		
TR-09	X			X			X		

Table 5.2: A risk map of the risks in the tools administrative panel.

be done by SQL or the mySQLAdmin interface. This ensures that even if a user gets unintended access to the administrative interface, it is not possible to drop all database info in one click.

The general risks that threatens the administrative interface is related to code injection. Both cross-site scripting and SQL injections are common vulnerabilities in web sites. Symantec Internet Security Threat Report of December 2007, Symantec (2008), writes that cross-site scripting on web sites account for 80% of all documented voulnerabilities on web sites. The use of proper input validation on user supplied input should be a priority when implementing the administrative interface.

Chapter 6

Evaluation

This chapter is intended to give an overview of the evaluation phase of the hACME game administrative interface and survey interface implementation.

6.1 Motivation

There is a variety of evaluation and testing schemes for analysing that a product meets the business and technical requirements that helped guided its implementation. As this was a three-person project; supervisor and two students, the project group decided to a more informal testing process. An important aspect of an administrative interface is to be useful for the administrator. An administrator might have insights on which aspects of the prototype for the administrative interface that might be useful and what is not. In order to improve the prototype and create suggestions for further improvement, the administrative interface for hACME game was presented to the current administrator of hACME game, Lillian Røstad. Røstad has been involved in the previous master thesis on hACME game, Nerbråten (2008) and Hagen and Taraldset (2009). During these projects, she has gained extensive knowledge of hACME game and what functions might be useful for and administrator in the administrative interface.

Another way of evaluating the administrative interface against the design and requirements is to do acceptance testing. This is a test conducted to validate if the contract between the customer and the developer is met. The user stories developed in Berrum and Johnsen (2010) could be used in addition to user scenarios created by the customer to test whether or not a user story has been correctly implemented. However, all user stories are centered around the administrative user. The group therefore decided to present the prototype of the administrative interface to the current administrator of hACME game and document the responses. This chapter contains the feedback from the administrative user, Lillian Røstad, during the evaluation meeting, 26.04.2011.

6.2 Survey Evaluation

This section describes the test phase on the survey interface.

6.2.1 Method

The administrative survey interface was presented to Lillian Røstad during the evaluation meeting. Røstad then gave feedback to the project team on suggestions for further changes. All changes suggested is documented and discussed in this chapter.

6.2.2 Result

This section contains all changes suggested by the administrator and the project team during the evaluation meeting.

Alternatives

In the current prototype of the administrative survey interface, each question on a survey has four alternatives, where one or several alternatives are correct. Some questions might be suited with more or fewer alternatives. It was therefore suggested to change this, so that each question on a survey could contain an undefined number of alternatives.

Further studies

The survey interface added to hACME game provides the possibility to add surveys for each level. There has not been any studies on how this can be used to enhance the gameplay of the game other than the suggestions made in chapter 7 and chapter 8. The technical implementation has however been made and is ready to use. It were therefore suggested to exclude the survey package from the deployment of hACME game until a study on survey usage in hACME game has been done.

6.3 Administrative Interface Evaluation

This section describes the test phase of the administrative interface.

6.3.1 Method

The administrative interface was presented to Røstad, one panel at the time. Røstad then gave feedback to the project team on suggestions for further changes. All changes suggested are documented and discussed in this chapter.

6.3.2 Result

This section contains all changes suggested by the administrator and the project team during the evaluation meeting.

Level Management

All challenges in hACME game has an identifier, ranging from S1 to S45. The project group had used these identifiers in the level management list. These identifiers are used by the programmers of each challenge. They are however not used in hACME game administrative view. It was therefore suggested to use challenge titles instead of identifiers in the challenge list.

News Management

The previous master thesis implemented functionality for adding news to hACME game. The add news button was placed under the admin menu on the left. As the new administrative interface has a news panel, it was suggestions to move this button to the management news panel instead. There was also a suggestion to change the color of the news links from blue to white.

Tools Statistics

The tools statistics panel was intended to provide an overview of all statistics on the site in an easy manner. If an administrator for instance wanted a quick overview of the number of users registered, the tools statistics panel should be used. It was however suggested that the name *tools statistics* was misleading.

Challenge Statistics

The challenge statistics panel was intended to give an indicator of how many attempts a student needs in order to complete a challenge. The current administrator argued that it would be more beneficial to present a percentage indication of the drop out level of a challenge instead. Some challenges are designed in a way that they needs more attempts than others. If a student needs to test different inputs in a field, this would be registered as attempts. These attempts would give a low successlevel in challenge statistics, even though the challenge might be easy to solve. It was therefore suggested to change the successlevel metric to the number of students that starts a challenge divided by the number of students that completes the challenge. It would give a percentage indication of the drop out level of a challenge.

Comment Statistics

The comment statistics provides statistics and a list of comments on a challenge. These comments are provided by the users who have completed that challenge. The query a user is provided with when he or she completes a challenge contains a comment field and two questions: 'How much did you enjoy solving this challenge?' and 'How hard was this challenge to solve?'. There was a suggestion that both these questions should be added as graphs in order to provide a more composite representation of the student feedback.

General Improvements

All graphs in hACME game administrative view uses challenges as x-axis. The graphs presents all challenges, including optional challenges and surveys. There were therefore suggestions of dividing these into groups of non-optional challenges, challenges and surveys. This would result in a view where the administrator could choose which challenges were presented; all challenges, non optional challenges, or surveys.

Chapter 7

Results and Discussion

This chapter provides a discussion on the result of the hACME game administrative interface implementation and the evaluation phase. The chapter starts by exploring the results of the implementation phase, then a discussion of the evaluation phase.

7.1 Implementation

This section explores the results from the implementation of hACME game's administrative interface. For each of the interfaces; statistics, tools and questionnaires, it defines what functionality implemented.

7.1.1 Administrative Statistics Interface

This section describes the functionality implemented on the statistics interface. Table 7.1 gives an overview of the implemented features. The table contains the requirements from Chapter 3. Each requirement is either fully implemented, partially implemented or not implemented. For each partially implemented requirement, there is a comment describing the missing feature.

Requirement	F	P	N	Comment
FR-01 Attempt Statistics	X			
FR-02 Challenge Statistics	X			
FR-03 Comment Statistics		X		Add a comment filter.
FR-04 User Statistics	X			

Table 7.1: Administrative Statistics Interface

The administrative statistics interface has been implemented according to the requirement specification presented in Chapter 3. Most of the features has been fully implemented, but due to time restrictions, FR-03 Comment

Statistics, has not been fully implemented. The statistics interface is currently missing a comment filter.

A comment filter would make it possible for an administrator browse a large amount of comments based on preselected words. According to Berrum and Johnsen (2010), this would make the comment-reading process more efficient. However, due to time restrictions, this task was not prioritized. A comment filter is not a necessity for reading comments on challenges. The rest of FR-03 Comment Statistics, is however, vital for an administrator to obtain feedback from students in the web view.

7.1.2 Administrative Tools Interface

This section describes the functionality implemented on the statistics interface. Table 7.2 gives an overview of the implemented features. The table contains the requirements from Chapter 3. Each requirement is either fully implemented, partially implemented or not implemented. For each partially implemented requirement, there is a comment describing the missing feature.

Requirement	F	P	N	Comment
FR-05 User Management	X			
FR-06 News Management	X			
FR-07 Level Setup	X			
FR-08 Site Statistics		X		Latest change on site.

Table 7.2: Administrative Tools Interface

The administrative tools interface has been implemented according to the requirement specification presented in Chapter 3. Most of the features has been fully implemented, but not FR-08 Site Statistics. This requirement is only partially implemented. The requirement specification states that it should be possible to view details on latest administrative changes on site. This requirement has not been implemented, mainly due to time restrictions and the current absence of administrative logging functionality.

There is currently no pre-implemented functionality for logging administrative tasks. This makes the task of implementing the change log more time consuming. The change log is not vital to the administrative tools interface. It was therefore not prioritized.

7.1.3 Administrative Questionnaire Interface

This section describes the functionality implemented on the questionnaire interface. Table 7.3 gives an overview of the implemented features. The table contains the requirements from Chapter 3. Each requirement is either fully implemented, partially implemented or not implemented. For each partially implemented requirement, there is a comment describing the missing feature.

Requirement	F	P	N	Comment
FR-09 Questionnaire	X			
FR-10 Questionnaire Administration	X			

Table 7.3: Administrative Questionnaire Interface

The administrative questionnaire interface has been implemented according to the requirement specification presented in Chapter 3. All features have been fully implemented.

7.1.4 Requirement Coverage

Some of the requirements has not been fully implemented. This is mainly because of time constraints. The most important work was to make three functional interfaces for changing the game setup and maintenance, acquiring statistics about the player progress and the use of surveys. The unimplemented requirements provides functionality which are not vital for the administrative interface. Table 7.4 contains percentage coverage for each requirement. The coverage percentage is defined as the number of completed sub-requirements in a requirement. This process has been done by the project group. The intention is to give a brief overview of the implementation phase results.

Requirement	Coverage
FR-01 Attempt Statistics	100 %
FR-02 Challenge Statistics	100 %
FR-03 Comment Statistics	75 %
FR-04 User Statistics	100 %
FR-05 User Management	100 %
FR-06 News Management	100 %
FR-07 Level Setup	100 %
FR-08 Site Statistics	50 %
FR-09 Questionnaire	100 %
FR-10 Questionnaire Administration	100 %

Table 7.4: Requirement coverage of the administrative interface

7.2 Development Process

hACME game's administrative interface has been developed by using agile methods, well documented APIs and focus on code quality. The system has mainly been developed by two programmers, namely the authors of this thesis. This section describes the development process of hACME game.

7.2.1 Development

hACME game has been developed with emphasis on frequent *releases* and short development cycles. This has been done in order to improve productivity and code quality. The programming tasks have mainly been implemented with the use of pair programming with focus of simplicity and clarity in code.

Cockburn and Williams (2001) argue that pair programming results in better design, which in turn results in simpler and more extendable code. Another strong argument for using pair programming in the administrative interface development is that both developers understand each part of the system.

Each part of the administrative interfaces has been treated as *releases* with short development cycles. A *release* usually corresponded to a specific requirement in Chapter 3. When the programming pair finished a *release*, the corresponding documentation were created or updated. These short programming tasks varied from 4-16 hours. After the programming tasks, each *release* was tested.

7.2.2 Testing

Unit testing determines whether or not a given product works as intended. Unit tests either test the whole system or an individual part of a source code. Due to the time restrictions in the development of the administrative interface, not all of the source code has been tested as thoroughly as the developers intended. However, the most important functions in all administrative interfaces have been tested. The results of the testing phase have been the findings of several errors that have been corrected.

7.2.3 Code Quality

As mentioned earlier the administrative interface has been developed with emphasis on simplicity and clarity in code. The development phase of each *release* has started by implementing the simplest solution. All extra functionality have then been added later. This approach gives a simple design and simple code. In this way, it is easy to extend the software by other developers.

When implementing the administrative interface the *Code Conventions for the Java Programming Language* document written by Sun Microsystems (1999) was used. Following conventions should make it easier to both maintain the code and extend it further.

7.2.4 Application Programming Interfaces

Several application programming interfaces have been used in order to implement the administrative interface. A list of all APIs used can be found in table 7.2.4.

- Hibernate
- Google Chart Tools
- Spring Framework
- Site Mesh
- Acegi Security

Table 7.5: All APIs used in hACME game administrative interface

All of the APIs, except Google Chart Tools, have been inherited from the hACME game application. This section will therefore emphasize on Google Chart Tools. The reason for using the other APIs can be found in Hagen and Taraldset (2009) and Nerbråten (2008).

Google Chart Tools were introduced with the administrative interface. There are several reasons for choosing Google Chart Tools as chart library in hACME game. First, the support for bar charts, which is used extensively in the administrative interface. In order to get a efficient overview of challenges and success/failure rates, divided bar charts are extensively used. An example of this is the challenge drop-out rate. Here, each completed challenge is shown as a bar chart, with the top of the bar as the drop-out (students quitting hACME game at this challenge). This can be seen in figure 7.1.

Secondly, Google chart tools offer the possibility for interactive charts. When a user hoovers a bar chart, information about the specific data can be shown to the user without reloading the current page. This feature supports details on demand as discussed in chapter 2. Details on demand is an important aspect of obtaining an overview of large data sets. In the administrative interface, all charts offers the ability to gain details on a challenge

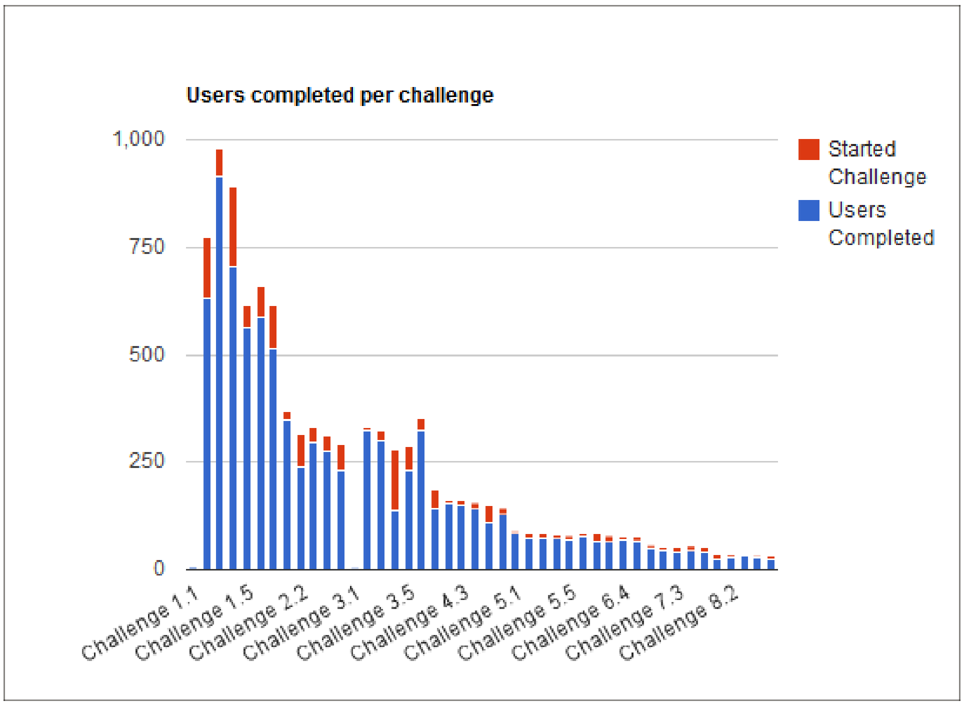


Figure 7.1: Statistics user interface.

in an easy manner. This can be seen in figure 7.2.

As mentioned earlier, hACME game administrative tools has been developed with emphasis on simplicity and clarity in code. Google Chart Tools offer this by providing a simplistic way of creating bar charts. By being easy to use and simplistic in design, most of the mitigation strategies in chapter 5 will be fulfilled. An example usage of Google chart tools code, can be found in chapter 5.

When selecting a graphing library for hACME game, the project group emphasized the aspects mentioned earlier; good support for bar charts, possibility for interactive details on demand and simplistic design. Google chart tools offered this and was therefore chosen as graphing library for hACME game's administrative view.

7.3 Goal Fulfillment

This section contains a discussion on the fulfillment of the goals defined in Chapter 1. In section 1.4 a list of objectives where defined by the project group and the supervisor. It was intended that all of these goals were fulfilled.

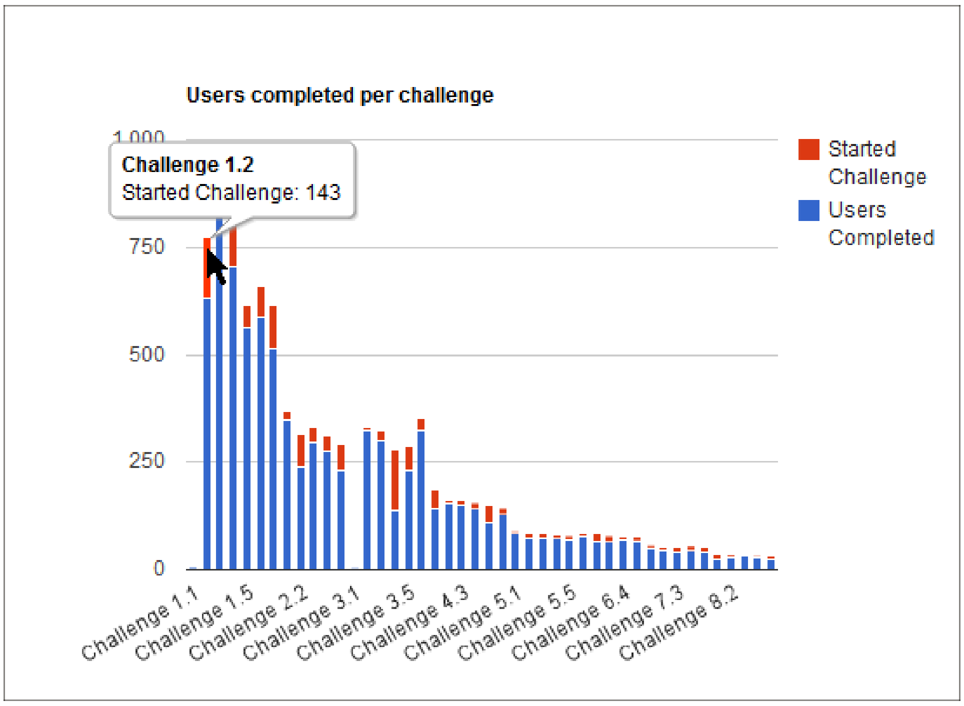


Figure 7.2: Mouse pointer hoovering statistics user interface.

7.3.1 G1 - Implement the Administrative Statistics interface

G1: *The administrative statistics interface offers statistical results from hACME game. The results are visualized to the administrator to help interpret the data. The statistics panel is divided into statistics for comments, challenges, attempts and users.*

This goal was reached by defining the requirements in section 3.3.2 and implementing them as described in section 7.1.1.

The administrative statistics interface design, as described in chapter 4, uses the preliminary study in chapter 2 as basis. The intentional usage of the administrative statistics interface is for the administrator to measure how students perform in the game and to measure their learning.

The statistics on which challenge users complete, and details about the user's progress is presented to the administrator in the statistical view. By presenting these data to the administrator in a suitable visual form, as discussed in chapter 4, the administrator can more easily obtain an overview of how students perform. How students perform on a challenge is a measure of their previous and recently gained knowledge (learning) about that particular challenge. The student completing the challenge can either already know how to complete it, learn how to complete it by gaining new knowledge or cheat (obtain the answer by another student or web site). For an administrator it is hard to see the distinction between the three from a completed challenge in the statistics view. However, there is safe to assume that there will be elements of current knowledge, learning and possible cheating when a student complete a challenge in hACME game.

By providing the administrator with the ability to view data on students performance, an administrator can monitor the students knowledge and learning. The administrator can also get perspective on which challenge is particularly hard or easy to complete. All of these data can help the administrator further improve the game and get insight on learning in hACME game. The implementation of hACME game's administrative statistics view, offers this functionality, as specified in chapter 3. The goal can therefore be considered as completed.

7.3.2 G2 - Implement the Administrative Tools interface

G2: *The administrative tool interface offer managements tools such as news publishing and level managements. This will help the administrator change the game setup in an easy manner. The tool panel is divided into management tools for news, level setup, users and site statistics.*

This goal was reached by defining the requirements in section 3.3.2 and implementing them as described in section 7.1.2.

The goal, *G2*, also includes a security analysis. As discussed in chapter 5, enabling database changes directly from the web view, involves some security aspects. A study on security aspects was therefore executed before the implementation of the administrative interface. The intention was to expose important security issues. This included how an administrator could obtain administrative privileges, what changes could be done and how they affect hACME game.

An important result from the security analysis is the decision to not let administrators delete from the database. Instead, an administrator can suspend (disabling an entry in database). If an administrator want to suspend, for instance, a user from hACME game, the user entry in database will not be deleted, but an enable bit will be set to false. The user supended will not be able to log in to hACME game, but not be deleted from the hACME game database. This mitigates the risk for an administrator to make non reversible changes through the administrative view. The changes an administrator performs in the hACME game administrative view, can easily be reverted directly from the database. This is another result from the security analysis; the discussion of a two layered administrative structure. A consequence of not giving administrators the privileges to delete from the database are two layers of administrators. One type of administrators, database administrators, can do non reversible changes, while administrative view administrators can not do non reversible changes. This sub/super administrator relationship can be used by current database administrators to promote administrators to maintain the game, with a low risk of them being able to corrupt game data.

In Berrum and Johnsen (2010), Chapter 5, user stories for hACME game administrative interface is defined. A lot of these user stories relies on a common work flow pattern for the administrator; first check the statistics interface, then do desired changes in the tools interface. If an administrator see in the statistics interface that an early-game challenge is particularly hard for a lot of users, the desired action will be to move the challenge to later in the game. This is what the tools interface offers.

The intention behind the administrative tools interface, is for the administrator to easily maintain and do changes to the game. This entails adding news, managing users and reorganize levels. The implementation of the administrative tools interface offers functionality for this, as specified in chapter 3. The goal can therefore be considered as completed.

7.3.3 G3 - Implement the usage of surveys

G3: *The usage of surveys will give the administrator feedback on the students theoretical understanding of software security.*

This goal was reached by defining the requirements in section 3.3.2 and implementing them as described in section 7.1.3.

The administrative survey interface design is described in chapter 4 and Berrum and Johnsen (2010). The intention is to get background information about the student's knowledge level. This knowledge level can then be used to modify the level and challenge setup accordingly. A student with solid theoretical knowledge can get a different challenge layout than a student with less theoretical knowledge.

As discussed in chapter 8, the implementation of surveys in hACME game has potential for further studies. As the survey interface incorporates a new metric, *student knowledge* into hACME game, a study on how this metric can be used to improve the game should be executed. *Student knowledge* can be used for dividing users into groups; beginner, intermediate and expert and change the challenges and levels thereafter. It can also be used for substitute challenges, so that if a student is not able to complete a challenge, a survey can be completed as a substitute for the challenge. In this way the drop out level may decline, as students who are stuck on a challenge can move on by completing a survey instead. A third way of using the surveys is to use them as learning indicators. By presenting an equal survey as the first challenge on a level and a survey as the last challenge on a level. If a student performs better on the last survey, learning should have occurred during the level completion.

These are some of the proposed usages of surveys in chapter 8. However, other suggestions for usages may exist. This topic could be the basis for further studies on hACME game.

All hACME game levels now begin with a survey. The administrative user can add, edit or remove questions to this survey. All requirements as specified in chapter 3 have been implemented. The goal can therefore be considered as completed.

7.3.4 G4 - Run a test phase

G4: *The test period will give feedback on how a small sample of students will perform with the proposed changes added to hACME game. Administrative users will also be able to test the implemented system and give feedback.*

The test phase of the administrative interface was intended to let the current administrators give feedback on the implemented system, as discussed in chapter 6. The administrators are the main users of the administrative interface and therefore a valuable source of information. Another intention behind this goal was to test the survey interface on a small sample of students. By testing the survey interface on a sample of students, it would be possible to obtain feedback from the students on what thoughts they would have on the survey interface.

As the implementation phase evolved and the survey interface prototype was implemented, it became clear that it might not be ready for testing yet. First, the authors have explored a lot of suggestions for further work on the subject. This can be seen in chapter 8. Without implementing any of these suggestions, the survey challenges are just multiple choice challenges. As argued in the preliminary study and the further work section of conclusion, in chapter 2 and chapter 8, the data collected form surveys should be used as a valuable source of information on student knowledge. Without a study on how this should be done, the survey interface should not be enabled yet. Secondly, the process of creating suitable questions for each level is an important and time consuming task. The project group did not have sufficient time at hand to create these questions.

The administrator test phase was an opportunity for the programmers to obtain suggestions for further improvements on the system. A list of suggested improvements can be found in chapter 6. All suggestions have been thoroughly explored by the project group.

The administrator feedback part of the goal was reached by giving a preview of the prototype of the administrative interface to the current administrator, Lillian Røstad on 27.04.2011. The project group and Røstad explored the possibilities for further improvements and changes. All suggested improvements have been documented and some of them implemented in the current system. The survey test phase was not initiated. The goal can therefore be considered as partly completed.

7.3.5 G5 - Use the test period results to suggest improvements

G5: *The test period will serve as a basis for suggestions on further improvements on the game.*

This goal was reached by exploring the possibilities for further improvements on the administrative interface and hACME game in general.

By developing the administrative interface, the project group has obtained extensive knowledge on hACME game. This knowledge should be transferred to future project groups working on hACME game or similar projects. The limited time available for implementing and designing hACME game administrative interface has also resulted in some unimplemented tasks, such as comment filtering and overview features. These tasks and several others have been documented in chapter 8.

The reason for writing the further work section of chapter 8 is to provide a starting point for deriving new development projects on hACME game. Both the supervisor, Lillian Røstad, and the project group have gained new insight on hACME game and the administrative interface during the project which has been documented in chapter 8.

The further work section has been divided into different categories; *statistics*, *tools*, *survey* and *overall improvements*. This has been done in order to provide a systematic overview of further improvements.

The *statistics* section is mainly focused on comment filtering and survey statistics. Comment filtering would be a helpful addition to the statistics comment interface because of the ability to sort and filter comments based on certain criteria. As discussed in Berrum and Johnsen (2010), it is important to be able to do interactive zooming and decrease the size of focused data. Survey statistics would provide the ability to obtain statistics on surveys. This is highly correlated with the different usages of surveys. If surveys are implemented as challenges, the administrator should be able to view how students perform equally to how the administrator can view how students perform on challenges. If surveys are implemented as a measure of student learning (two equal surveys at the start and end of a level) then the administrator should be able to monitor the difference in those surveys.

The *tools* section is mainly focused on discussing an active user tool. A problem that might occur in hACME game and other membership web sites is the issue with active and inactive users. The hACME game database contains personal information on hACME game users (such as email, password, study program and name). When a user is inactive, the user could be notified that he or she either has to log in to hACME game, or the user will get deleted from the game. It might be unnecessary to store personal information on users who will not log in to hACME game again. However, their game statistics might be useful for administrative statistics purposes. It might also be in conflict with the hACME game user agreement if it states that no unnecessary personal information will be stored. As of now, the only way to monitor user activity is by stepping through each user in the tools user interface and view the last login date. The *tools* section explores the possibility for other ways of doing this.

The *survey* section is mainly focused on how to use the survey interface. The reason for documenting this is to create a starting point for further studies on hACME game questionnaire interface. The possibilities are many. It might be desirable to use it to measure student learning or classifying players of different knowledge. It might also be useful to provide it as a means to keeping users who otherwise might be dropping out if they can not complete a challenge.

The project group and the supervisor, Lillian Røstad, have identified several further improvements on hACME game during regular meetings and the test phase meeting. Some suggestions concern the administrative interface, while others suggestions are conceptual changes to the game itself. Chapter 8 contains an extensive discussion of further work on the system. The goal can therefore be considered as completed.

7.4 Method

This master thesis has used a combination of qualitative and quantitative methods. The main goal of Berrum and Johnsen (2010) was the creation of a conceptual design for an administrative interface where the administrators could get an easy impression on student learning in hACME game. This master thesis is mainly focused on implementing this conceptual design. The statistics panel in the administrative interface provides quantitative measures on how many student complete a challenge, how many attempts they use and where the drop out. These measures combined can be used by an administrator to obtain a qualitative measure of student learning. Pivec and Dziabenko (2004) state that if a game becomes too hard or too easy to complete, users will loose interest to play, which in turn affect the users learning. By monitoring user progression and adjusting the difficulty level in the administrative tools interface, administrators can make a positive impact on student learning.

One of the main concerns in the preliminary study of Berrum and Johnsen (2010) was to define student learning in hACME game. The results from this preliminary study was used to create a conceptual design for an administrative interface. In this master thesis this interface has been implemented according to the requirement specification in chapter 3. In order to verify that it actually is possible to measure the student learning in hACME game, a thorough study on the qualitative metrics in the administrative interface is necessary.

Collins (2011) defines learning as the acquiring of new knowledge or modifying existing knowledge. In figure 7.3 a graph of completed users per challenge is shown. This graph is similar to the ones in the administrative interface. As mentioned earlier in this chapter, there are three ways of completing a challenge. Either to know how to solve it before the challenge starts, learn how to solve the challenge or to cheat. The first one is to perform a repetition of previous experience, or to modify previous knowledge. The second is to gain new knowledge. Both of these are examples of learning. The third one, cheating, is not. However, hACME game is a not mandatory to complete, and there are no rewards (other than learning) for completing the game, so it might be safe to assume that cheating is kept to a minimal. With this assumption, when a user completes a challenge, he or she has either modified previous knowledge or gained new knowledge. Both of these are examples of learning.

Figure 7.3 shows how many students completed a challenge. The blue bar is the number of users completed a specific challenge, while the red bar is the number of drop outs (users starting the challenge, but not completing).

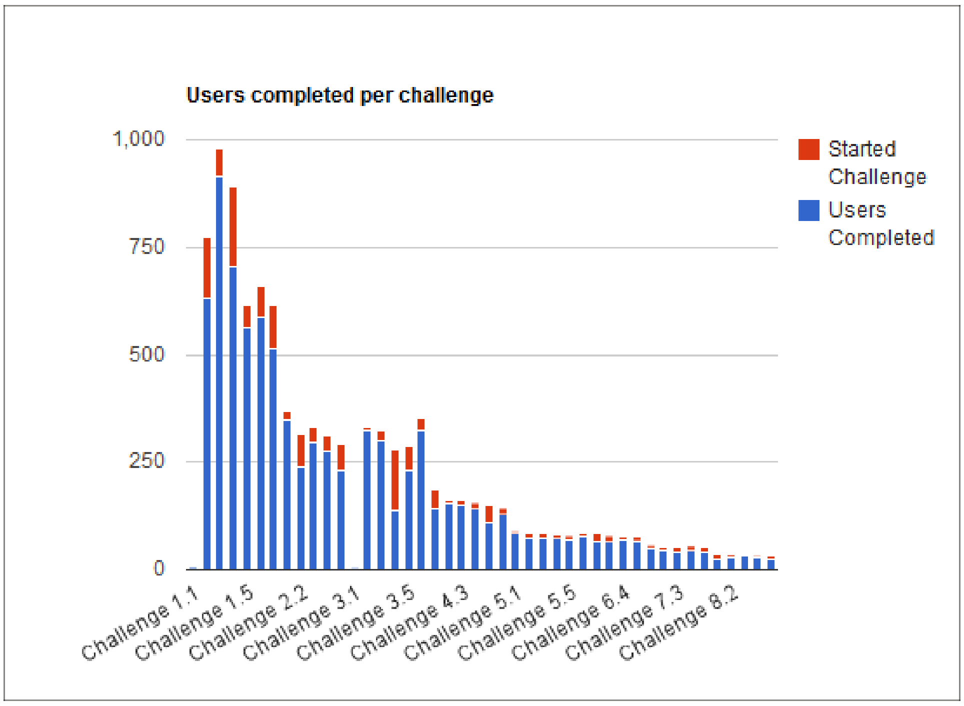


Figure 7.3: Statistics user interface.

By giving the administrator a graph representation of completed challenges for hACME game users, he or she can see which challenge users complete, and where they drop out. By the definition above, this is a measure of student learning.

However, the definition above does not take cheating in to account. Cheating would be to obtain an answer on a challenge from a friend playing the game, from the Internet or to guess the correct solution on a challenge. The first two are hard to differentiate from knowing the answer beforehand. It is hard to know how a user obtained the answer to a challenge. It could be previous knowledge and it could be cheating. To get an indication on the third, guessing, figure 7.4 could be useful. It is easy to see that challenge 4.3 and challenge 2.4 have a lot of incorrect answers. It could mean that users try out different solutions in order to obtain the correct one, which is not cheating. It could also mean that they are guessing. Both will yield a large number of incorrect answers. What figure 7.4 does supply, is an indicator of which challenge that might not be sufficiently explained to the students. This would explain the first two. Users does not know what to answer, and starts trying out different possibilities for solutions or guessing.

Figure 7.5 presents the successlevel metric. This metric indicates how

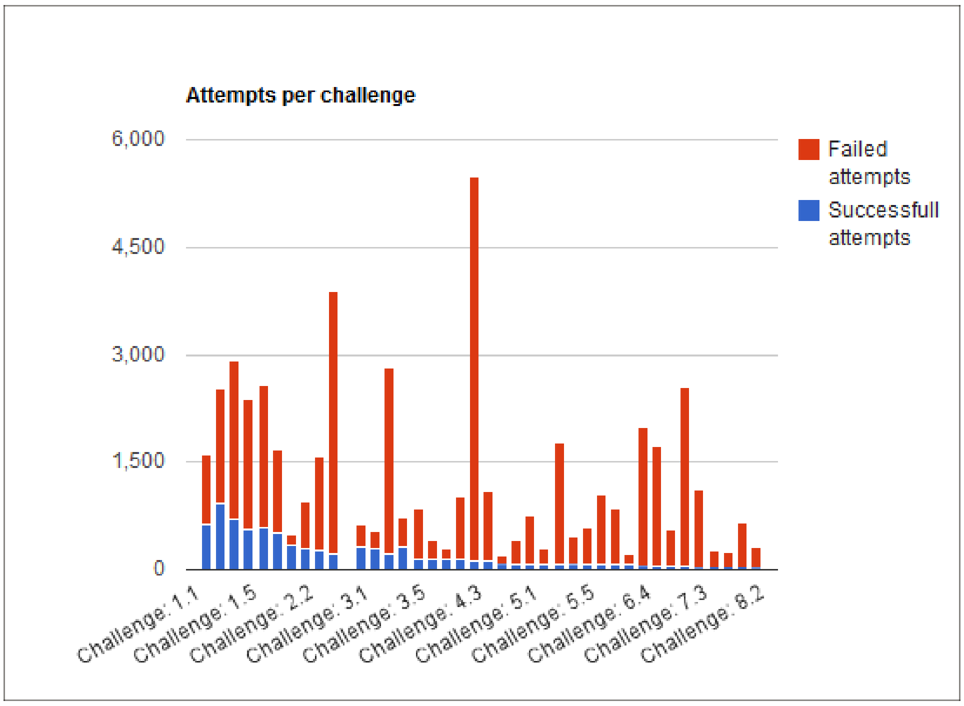


Figure 7.4: Statistics attempts interface.

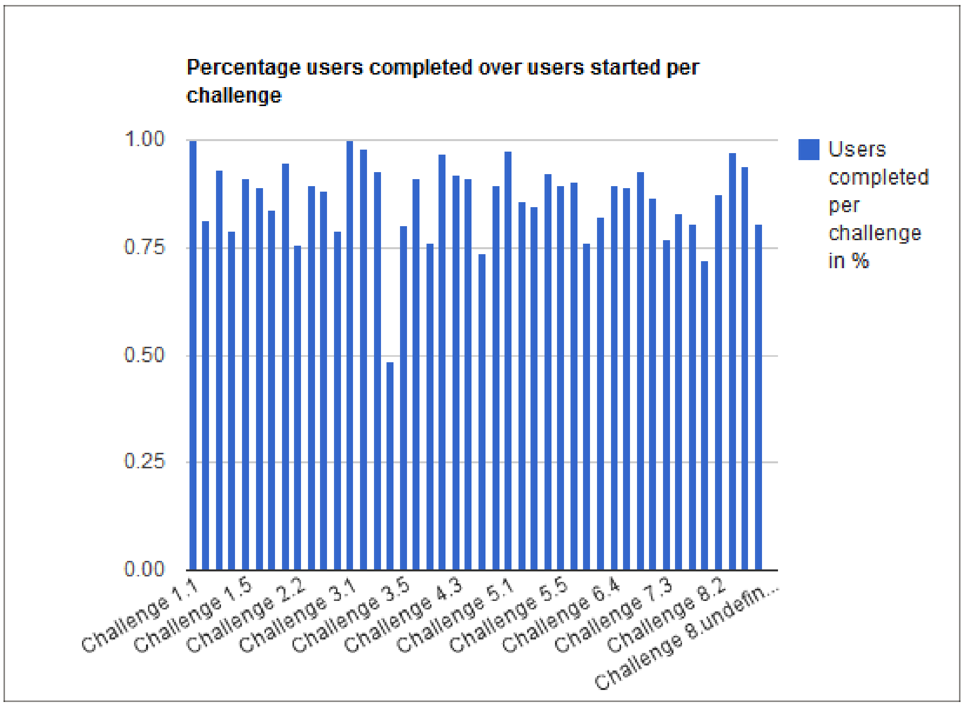


Figure 7.5: Statistics challenge interface.

many students that start a challenge, completes it. It is defined as the number of students that starts a challenge divided by the number of students that completes the challenge. It gives a percentage indication of the drop out level of a challenge.

Although none of the above figures can give a simple and conclusive measure of student learning, the combination of the figures can give a qualitative understanding of student learning in hACME game. The main focus from Berrum and Johnsen (2010) was to understand how student learning in hACME game occurs, and how a administrative interface for measuring this could be created. This master thesis focuses on implementing the administrative interface. This method of study first, then applying knowledge gained into a specific implementation has led to a prototype of the administrative interface.

Chapter 8

Conclusion and Further Work

This chapter is intended to give suggestions for further improvements in the game as well as a conclusion of the master thesis.

8.1 Conclusion

The purpose of this master thesis was to implement an conceptual design for an administrative interface described in Berrum and Johnsen (2010). The design was based on a preliminary study on student learning, data visualization and previous versions of hACME game.

The master thesis has been divided into several parts. First, a requirement specification was created based on the preliminary study in chapter 2 and the conceptual design in chapter 4. The requirement specification includes use cases and detailed description of the administrative interface.

After the requirement specification in chapter 3 was written, a security analysis of hACME game's administrative interface was created. This security analysis covers all aspects of security related issues for the interface.

A central and time consuming part of this master is the implementation phase described in chapter 3 and chapter 7. A lot of effort were put into developing the prototype of the administrative interface. Most of the requirements from the requirement specification was implemented.

The evaluation phase was intended to give the authors valuable information about the implemented prototype from the current administrator. The survey interface was not tested, as described in chapter 7. However, the administrative interface was presented to the administrator, Lillian Røstad, which gave valuable feedback. This feedback is part of the foundation for chapter 8.

The next section offers suggestions for further improvement of hACME game and is the final part of this master thesis. The suggestions has been derived from the evaluation phase and experience gained throughout the project.

The result of this master thesis is a fully functional prototype of an administrative interface for hACME game. An administrative user can now do a variety of maintenance work, obtain feedback from students and game statistics directly from the web view. Based on the discussion in chapter 7 and evaluation in chapter 6 this project has succeeded in implementing the proposed administrative interface for hACME game in Berrum and Johnsen (2010).

8.2 Further Work

This section describes possible future improvements of hACME game and the administrative interface. The suggested improvements has been derived from the evaluation phase of this master thesis and experience gained throughout the project.

8.2.1 Administrative Statistics Interface

The statistics interface provides the administrator with valuable information on player patters. This section suggests further improvements on the administrative statistics interface.

Comments Filtering

In chapter 2 textual filtering is explained. By adding a text filter in the comments view, an administrator could exclude all text not containing a specific word or phrase. This would decrease the size of focused data and may help the administrator interpret the feedback from comments more efficient.

Survey Statistics

A part of this master thesis was the implementation of a survey interface. The interface enables the administrative user to add questions and answers to surveys. Each level has it's own survey. The surveys are viewed as multiple choice challenges for hACME game users. A further improvement on this addition to hACME game is a survey statistics panel. This panel should provide statistics about each survey and the survey results. As an administrator it might be valuable to obtain information on how students perform on surveys. This information can include which part of hACME game's curriculum (theory from challenges in hACME game) is particularly hard or easy for students and the overall performance on surveys.

8.2.2 Administrative Tools Interface

The tools interface provides the administrator with maintenance functionality directly from the web view. This functionality can be improved with new functionality outside the scope of the tools design presented in chapter 4. This section provides some suggestion on further improvements.

Active User Tool

The user table in the hACME game database contains over 1100 users, as seen in appendix C. Some users may have been inactive for over a year, and there may not be any reason for storing their user information in the database, as they may not log in to hACME game in the future. An extension to the hACME game tool interface may be a panel where the administrator can send an email to inactive users (users not logged in for a predefined time limit) and inform them that their account will be deleted if they not log in to hACME game in the near future. With this extension, the hACME game database will only contain data about active hACME game users.

8.2.3 Administrative Questionnaire Interface

The questionnaire interface provides valuable information about hACME game players to the administrator. The data collected could serve as a basis for a variety of improvements in hACME game. This section proposes a couple of ways hACME game could be extended with this data collection.

Survey as Challenge

This thesis extended hACME game with surveys. The usage of surveys provides the administrator with background information about students knowledge level on software security. As of now, a survey is the first challenge on each level and is mandatory. An other way of using these surveys is to make them as substitutes for challenges. If a student reaches a challenge he or she can not complete, an option may be to take a survey, and then continue the level. In this way, when a student reaches an obstacle (a challenge which he or she can not complete) it is still possible to continue playing.

Player Classification

As mentioned, the surveys give the administrator valuable background information about student knowledge level. This information may be used to classify students in *beginner*, *intermediate* and *expert* levels. If a student scores well on a challenge, he or she is classified as an *expert* and otherwise, *beginner* or *intermediate*. This classification may be used for level setup. An *expert* user may be provided with other challenges than a *beginner*.

8.2.4 General Improvements

This section provides some suggestions for improvements outside the scope of this master thesis. These suggestions does not involve the statistics, tools or questionnaire interface, but hACME game itself.

Focus on Feedback

In hACME game the administrator obtain feedback from students by reading comments from challenges. These comments are not mandatory for the students to give. Feedback from students give the administrator valuable information about the game. A further improvement on hACME game may be to extend the way feedback is supplied to the administrator. In the current feedback system, the users who completes a level answers a two questions; 'how did you enjoy this challenge?' and 'how hard was this challenge to solve?'. Both questions by selecting from a scale ranging from 1 to 6. Additionally, a text field called comments is supplied. Here, the user can write additional comments as feedback. As further work, a study on how feedback from students to the administrator can be improved should be done.

Chapter 9

Bibliography

Bibliography

Audrey L. Amrein and David C. Berliner. The effects of high-stakes testing on student motivation and learning. *Educational Leadership*, 60(5):32–38, 2003.

Mike Andrews and James A. Whittaker. *How to Break Web Software: Functional and Security Testing of Web Applications and Web Services.* Addison-Wesley Professional, 2006. ISBN 0321369440.

Aschehoug and Gyldendal. Aschehoug og gyldendals store norske leksikon. `http://www.snl.no/.search?query=hacker`, September 2010.

Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: combining display technology with visualization techniques. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 31–40, New York, NY, USA, 2001. ACM. ISBN 1-58113-438-X. doi: http://doi.acm.org/10.1145/502348.502354.

Benjamin B. Bederson and Ben Shneiderman. *The Craft of Information Visualization: Readings and Reflections.* Morgan Kaufmann, 2003. ISBN 1558609156.

Christian Berrum and Morten Weel Johnsen. hacme game - administrative interface. Master's thesis, Norwegian University of Science and Technology, 2010.

R.F Bowman. A pac-man theory of motivation: Tactical implications for classroom instruction. *Educational Technology*, 22(9):14–17, 1982.

John D. Bransford, Ann L. Brown, , and Rodney R. Cocking. *How people learn: Brain, mind, experience, and school.* National Academy Press, Washington DC, 2002. ISBN 0309065577.

F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, 2003.

A. Cockburn and L. Williams. The costs and benefits of pair programming. *Extreme programming examined*, pages 223–248, 2001.

Dictionary Collins. Collins english dictionary - complete and unabridged 10th edition. *Dictionary Collins*, May 2011. URL `http://dictionary.reference.com/browse/learn`.

J.E. Driskell and D.J. Dwyer. Microcomputer Videogame Based Training. *Educational Technology*, 24(2):11–17, 1984.

S.G. Eick. Visual discovery and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):44–58, 2000.

Stephen G. Eick. Graphically displaying text. *Journal of Computational and Graphical Statistics*, 3(2):pp. 127–142, 1994. ISSN 10618600. URL `http://www.jstor.org/stable/1390665`.

A.J. Elliot and J.M. Harackiewicz. Goal setting, achievement orientation, and intrinsic motivation: A mediational analysis. *Journal of Personality and Social Psychology*, 66:968–968, 1994.

M. Friendly and D.J. Denis. Milestones in the history of thematic cartography, statistical graphics, and data visualization. *web document, available at http://www. math. yorku. ca/SCS/Gallery/milestone*, 174, 2001.

R. Garris, R. Ahlers, and J.E. Driskell. Games, motivation, and learning: A research and practice model. *Simulation and Gaming*, 33(4):441–469, 2002.

Google. Google chart tools, March 2011. URL `http://code.google.com/intl/no/apis/visualization/interactive_charts.html`.

Michel Goossens, Frank Mittlebach, and Alexander Samarin. *Experiential learning: Experience as the source of learning and development.* Prentice Hall Press, London, 1984. ISBN 0132952610.

Carl Gutwin. Improving focus targeting in interactive fisheye views. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 267–274, New York, NY, USA, 2002. ACM. ISBN 1-58113-453-3. doi: http://doi.acm.org/10.1145/503376.503424.

Eilev Hagen and Ralf Bjarne Taraldset. hacme game - a tool for teaching security. Master's thesis, Norwegian University of Science and Technology, 2009.

IEEE. *Recommended Practice for Software Design Descriptions.* IEEE Computer Society, 1998a.

IEEE. *Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998b.

Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/2945.981847.

Daniel A. Keim, Wolfgang Muller, and Heidrun Schumann. Visual data mining, 2002.

Kristian Kiili. Towards an experiental gaming model. *Internet and Higher Education*, 8(1):13–24, 2005.

E.E. Koutsofios, S.C. North, and D.A. Keim. Visualizing large telecommunication data sets. *Computer Graphics and Applications, IEEE*, 19(3): 16–19, 2002.

Eleftherios E. Koutsofios, Stephen C. North, and Daniel A. Keim. Visualizing large telecommunication data sets. *IEEE Comput. Graph. Appl.*, 19(3):16–19, 1999. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.761543.

Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, 1986. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/22949.22950.

Thomas W. Malone. Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4):333–369, October 1981.

A. McFarlane, A. Sparrowhawk, and Y. Heald. Report on the educational use of games. *TEEM: Teachers Evaluating Educational Multimedia*, 2002.

Øyvind Nerbråten. A tool for teaching security. Master's thesis, Norwegian University of Science and Technology, 2008.

D. A. Norman. *Things that make us smart : defending human attributes in the age of the machine.* Addison-Wesley Pub. Co., 1993.

Diana G. Oblinger. The next generation of educational engagement. *Journal of Interactive Media in Education*, May 2004. ISSN 1365-893X.

OWASP. Api abuse, March 2011. URL `http://www.owasp.org/index.php/Category:API_Abuse`.

Marina Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Comput. Educ.*, 52(1):1–12, 2009. ISSN 0360-1315. doi: http://dx.doi.org/10.1016/j.compedu.2008.06.004.

Marina Papastergiou and Christina Solomonidou. Gender issues in internet access and favourite internet activities among greek high school pupils inside and outside school. *Comput. Educ.*, 44(4):377–393, 2005. ISSN 0360-1315. doi: http://dx.doi.org/10.1016/j.compedu.2004.04.002.

M. Pivec and O. Dziabenko. Game-based learning in universities and lifelong learning:" unigame: Social skills and knowledge training" game concept. *Journal of universal Computer science*, 10(1):14–26, 2004.

Maja Pivec, Olga Dziabenko, and Irmgard Schinnerl. Aspects of game-based learning. In *Game Scenario "UniGame: Social Skills and Knowledge Training*, July 2003. Proceedings of I-KNOW 2003: Graz, Austria.

Marc Prensky. *Digital Game-Based Learning.* McGraw-Hill Pub. Co., 2004. ISBN 0071454004.

Dennie Reniers, Lucian Voinea, and Alexandru Telea. http://www.cs.rug.nl/ alext/papers/eurovis10/poster.pdf, 2010. URL `http://www.cs.rug.nl/~alext/PAPERS/EuroVis10/poster.pdf`.

Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, IEEE Symposium on*, 0: 336, 1996. ISSN 1049-2615. doi: http://doi.ieeecomputersociety.org/10.1109/VL.1996.545307.

Harri Siirtola. Interactive visualization of multidimensional data. Technical report, Tampere Universisty - Faculty of Information Sciences, 2007.

C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *IEEE Transactions on Visualization and Computer Graphics*, pages 176–187, 2003.

Sun Microsystems. Code conventions for the java programming language, April 1999. URL `http://www.oracle.com/technetwork/java/codeconv-138413.html`.

M. Sutton, A. Greene, and P. Amini. *Fuzzing: Brute Force Vulnerability Discovery.* Addison-Wesley Professional, 2007. ISBN 0321446119.

Symantec. Symantec internet security threat report. *Symantec Enterprise Security*, 8, 2008.

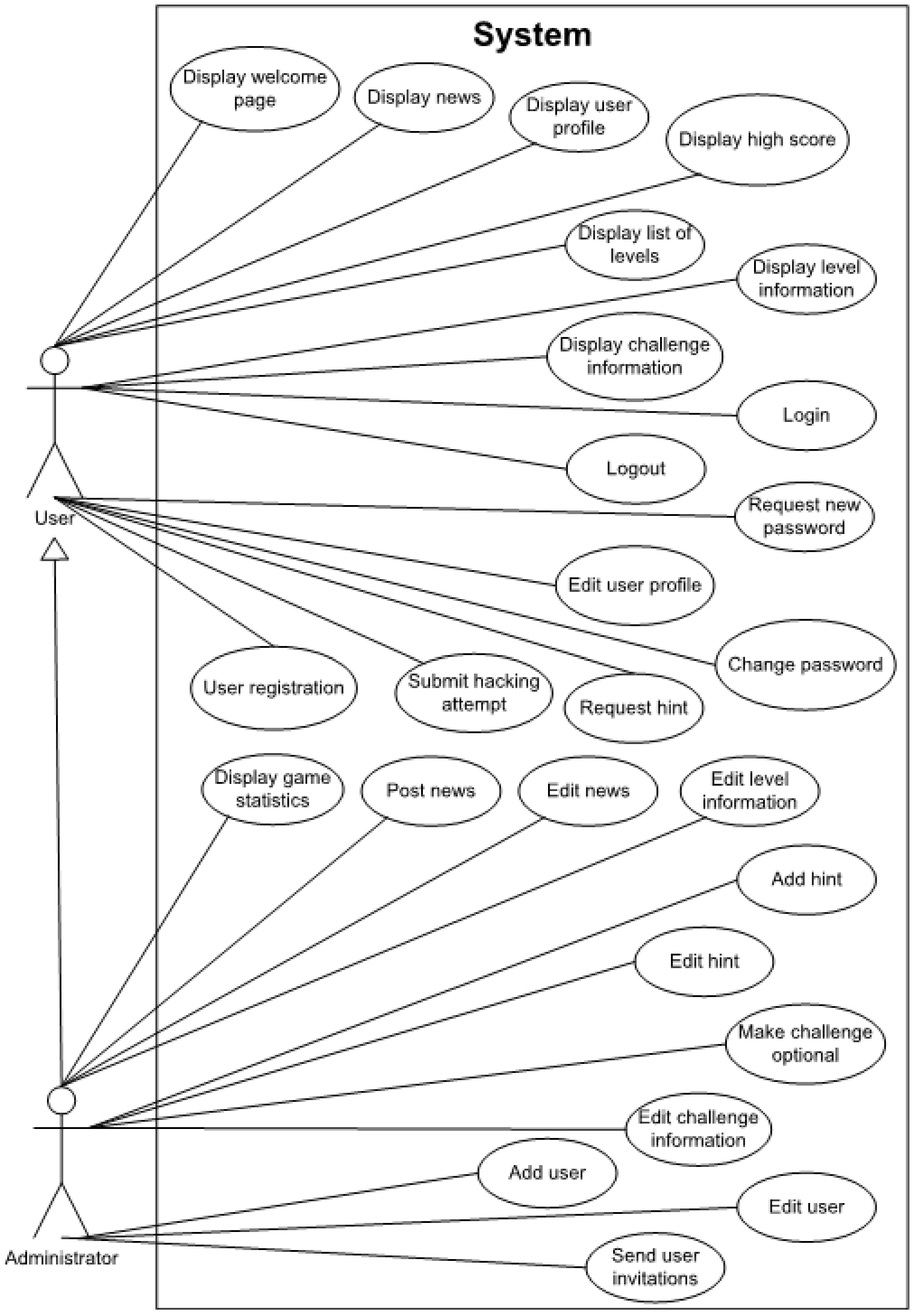
Tetsuji Takada and Hideki Koike. Mielog: A highly interactive visual log browser using information visualization and statistical analysis. In *LISA '02: Proceedings of the 16th USENIX conference on System administration*, pages 133–144, Berkeley, CA, USA, 2002. USENIX Association.

Jeff Williams. Owasp annual report 2009. In *OWASP AppSec DC 2009 Conference.* OWASP, 2009.

Appendix A

First Master Thesis

This appendix contains an overview use case of the first version of hACME game from Øyvind Nerbråten's master thesis from 2008. Nerbråten (2008)


System
Display welcome page
Display news
Display user profile
Display high score
Display list of levels
Display level information
Display challenge information
Login
Logout
Request new password
Edit user profile
Change password
User registration
Submit hacking attempt
Request hint
Display game statistics
Post news
Edit news
Edit level information
Add hint
Edit hint
Make challenge optional
Edit challenge information
Add user
Edit user
Send user invitations
User
Administrator

Appendix B

Second Master Thesis

This appendix contains functionial requirements, non-functional requirements, distribution of attacks and security requirements from the second version of hACME game from Eilev Hagen's and Ralf Bjarne Taraldset's master thesis from 2009, Hagen and Taraldset (2009).

3.3.1 Functional requirements

The thirteen functional requirements that were indentified in the preliminary project. User stories for these functional requirements can be found in appendix B.1.

Requirements		FI	NI
FR1	Display challenge solution	•	
FR2	Extend Leet-O-Meter	•	
FR3	Display challenge statistics	•	
FR4	Modify high score list	•	
FR5	Email address as username	•	
FR6	Modify reset password function	•	
FR7	Improve authentication		•
FR8	Collect hacking attempt content	•	
FR9	Collect time on hint requests	•	
FR10	Categorize challenges by attack type	•	
FR11	Collect player background data	•	
FR12	Varying level order	•	
FR13	Surveys	•	

Table 3.1: Functional requirements status. FI = fully implemented, NI = not implemented

Game-based learning requirements

Requirements		F	NF
NFR1	The game shall be expanded with more challenges to work better as a teaching tool.	•	
NFR2	The challenges shall reflect today's web security vulnerability landscape.	•	
NFR3	The game shall provide understanding of the simulated vulnerability with the respecting best practice counter-measures.	•	
NFR4	The game shall be self motivating.	•	
NFR5	The difficulty of the challenges shall advance from level to level.	•	
NFR6	If a challenge is initially too hard for a player to solve, there shall be mechanisms providing help.	•	

Table 3.2: Game-based learning requirements status. F = fulfilled, NF = not fulfilled

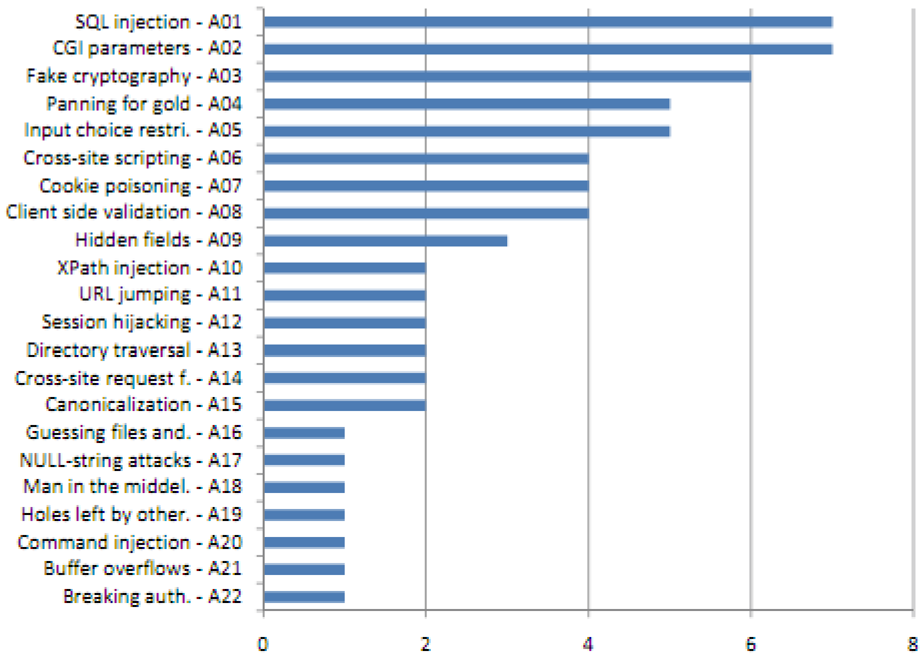


Figure 3.3: hACME game attack type distribution

Security requirements

Requirements		F	NF
NFR10	Variables shall be bound to database queries using prepared statements.	•	
NFR11	All input data shall be validated or escaped.	•	
NFR12	The session identifier shall be randomly generated with high complexity.	•	
NFR13	The system shall never reveal registered users' identity.	•	
NFR14	Users shall upon registration be informed about security measures for awareness of their login credentials.	•	
NFR15	The system shall enforce a strong password policy.	•	
NFR16	User passwords shall not be sent over a communication link in plaintext nor be possible to replay.		•
NFR17	The system shall protect sensitive data.	•	
NFR18	It shall not be possible to enforce an account lockout.	•	
NFR19	If an request results in modifying state or data, the source of the request shall be validated to be from the application itself.	•	
NFR20	Sensitive files shall be protected against unauthorized access.	•	
NFR21	The system shall never display revealing information to the public.	•	
NFR22	Vulnerabilities simulated in the game challenges shall not propagate to the rest of the system.	•	

Table 3.4: Security requirements status. F = fulfilled, NF = not fulfilled

Appendix C

Database

This appendix contains the hACME game database table size at 20.01.2011.

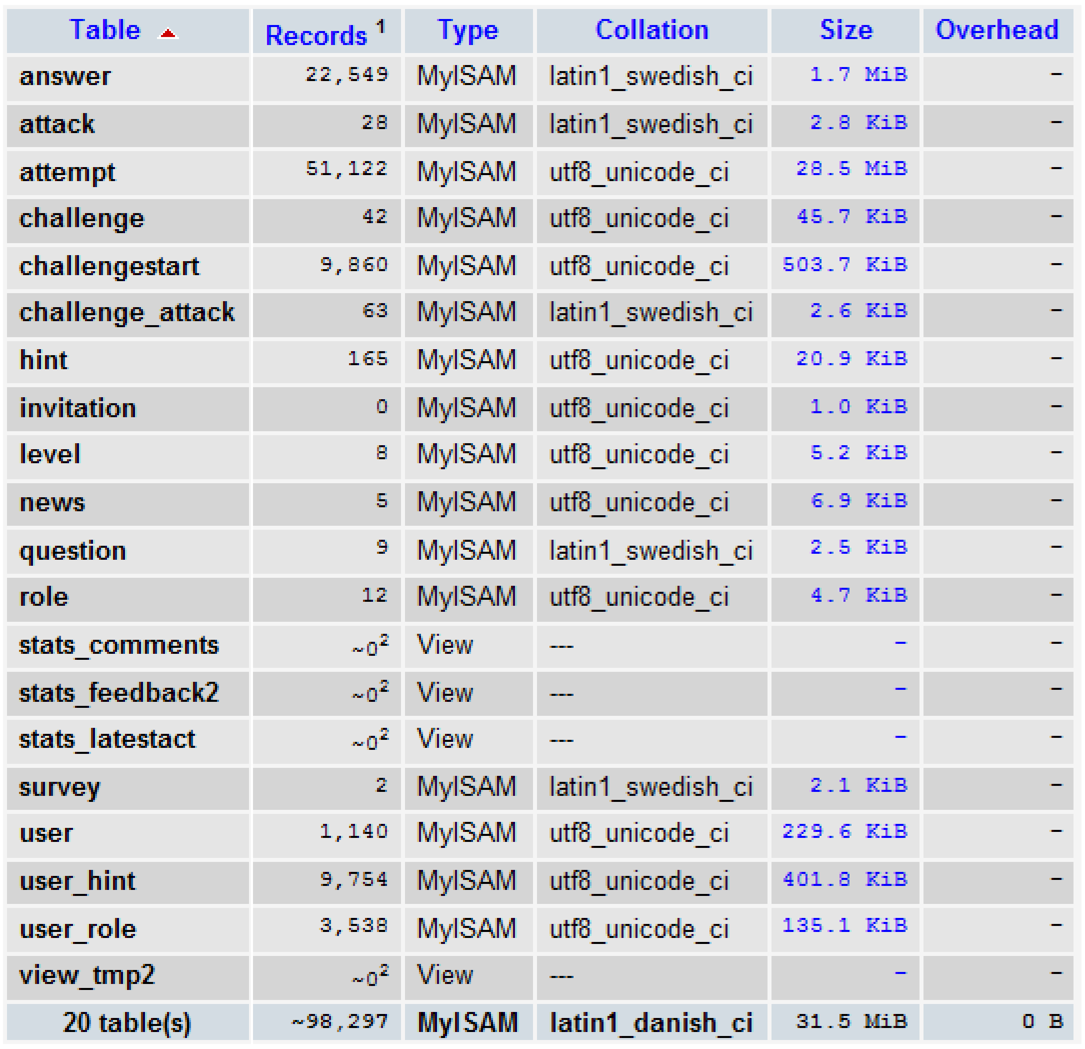


Table	Records[1]	Type	Collation	Size	Overhead
answer	22,549	MyISAM	latin1_swedish_ci	1.7 MiB	-
attack	28	MyISAM	latin1_swedish_ci	2.8 KiB	-
attempt	51,122	MyISAM	utf8_unicode_ci	28.5 MiB	-
challenge	42	MyISAM	utf8_unicode_ci	45.7 KiB	-
challengestart	9,860	MyISAM	utf8_unicode_ci	503.7 KiB	-
challenge_attack	63	MyISAM	latin1_swedish_ci	2.6 KiB	-
hint	165	MyISAM	utf8_unicode_ci	20.9 KiB	-
invitation	0	MyISAM	utf8_unicode_ci	1.0 KiB	-
level	8	MyISAM	utf8_unicode_ci	5.2 KiB	-
news	5	MyISAM	utf8_unicode_ci	6.9 KiB	-
question	9	MyISAM	latin1_swedish_ci	2.5 KiB	-
role	12	MyISAM	utf8_unicode_ci	4.7 KiB	-
stats_comments	~0[2]	View	---	-	-
stats_feedback2	~0[2]	View	---	-	-
stats_latestact	~0[2]	View	---	-	-
survey	2	MyISAM	latin1_swedish_ci	2.1 KiB	-
user	1,140	MyISAM	utf8_unicode_ci	229.6 KiB	-
user_hint	9,754	MyISAM	utf8_unicode_ci	401.8 KiB	-
user_role	3,538	MyISAM	utf8_unicode_ci	135.1 KiB	-
view_tmp2	~0[2]	View	---	-	-
20 table(s)	~98,297	MyISAM	latin1_danish_ci	31.5 MiB	0 B

Appendix D

Development of hACME Game

This appendix contains describes which tools and libraries are used for development of hACME game.

D.1 Dependencies

The following tools and libraries are neccessary to install in order to further develop hACME game and the administrative interface. The version number is also mentioned. Newer versions might also work properly.

- MySQL version 5
- Java SDK version 1.5
- Maven version 2

D.2 Application Servers

GlassFish

GlassFish an open source application server developed by Sun Microsystems. GlassFish is the desired application server and is present on *www.hacemgame.org*.

Jetty

Jetty is a Java-based HTTP application server developed as an open source project as part of the Eclipse Foundation. It can be used in further development of hACME game.

D.3 Installation Guide

A more comprehensive installation guide can be found in Hagen and Taraldset (2009), Appendix G.