Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Web-Based Drawing Tool in GWT with Usability Testing and Usability Evaluation

by

Alf Bjelkenstedt

LIU-IDA/LITH-EX-A---14/001--SE

2014-01-22
Webbaserat ritningsverktyg i GWT med användartester och användbarhets utvärdering

Examensarbete 30 hp skriven av

Alf Bjelkenstedt

LIU-IDA/LITH-EX-A--14/001--SE

2014-01-22

Web-Based Drawing Tool in GWT with Usability Testing and Usability Evaluation

Supervisor: Lena Buffoni
Examiner: Kristian Sandahl
Summery

A web-based drawing tool has been developed in Java, mainly with the library Google Web Toolkit (GWT). This project has been conducted at the company Inspectera HK AB in Norrköping and the tool has been implemented in Inspectera's E-service “Inspectera Online” (IO). This drawing tool enables users to edit uploaded maps or blueprints directly in the web browser. There exist several menus in the editor which contains categorized symbols that the user can place out in separated layers on top of the map/blueprint. The editor has general tools such as save, load, print, a marking tool, remove object, and undo functions.

The drawing tool has been integrated with a module that already existed in Inspectera Online. The tool fetches every activated inspection for the currently logged in organization. The user may then add the position of the inspections on a map/blueprint. The inspection's status and it's specific information can also be displayed inside the editor.

Sketch tools have also been developed for drawing walls, windows, doors, lines, circles, squares and texts. There is also a color selection available for these sketch objects.

The drawing tool has been tested for usability. It's been proven that the drawing tool is both effective and efficient. Hence, test users have been able to perform the intended tasks in the system within reasonable time. The tasks designed for the tests have been derived from realistic scenarios.

The drawing tool has also been scored by the test users on a Likert scale from one to five. Questions have been designed to target the quality criteria of usability. The drawing tool received an average score of four regarding criteria such as satisfaction and learnability.

The author of the thesis has also performed a usability evaluation (a heuristic evaluation) of the drawing tool. Feedback about the usability of the tool has been derived with the help of Jakob Nielsen's heuristics for user interface design. The ten most common heuristics have been used. The results of this evaluation show that the developed tool both includes and lacks features that the heuristics promote.
Abstract

On behalf of Inspectera HK AB in Norrköping a web-based drawing tool has been developed in Java, mainly with the library Google Web Toolkit (GWT). The purpose of this tool is to facilitate both the staff's at Inspectera and their client's work with different types of drawings such as blueprints for pest control, fire protection and especially drawings of the company's e-service of self-checks. Besides developing the drawing tool usability testing and a usability evaluation has been performed.

Keywords: GWT, Drawing Tool, Java, Usability, Usability Evaluation, Usability Testing
Acknowledgment

My first sincere appreciation goes to Anders Ekeberg (CEO at Inspectera) for giving me the opportunity to work in a real life setting on an interesting project which was both challenging and instructive.

I would like to thank Kristian Sandahl for being my examiner despite a very special time plan. I thank him for his advices and insight during the work with the report writing.

Last but not least I would like to thank Lena Buffoni for accepting to be my supervisor.

Alf Bjelkenstedt
Norrköping 2014
### List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO</td>
<td>Inspectera Online (Inspectera's e-service)</td>
</tr>
<tr>
<td>GWT</td>
<td>Google Web Toolkit</td>
</tr>
<tr>
<td>MVP</td>
<td>Model View Presenter</td>
</tr>
<tr>
<td>HTML5</td>
<td>Hypertext Markup Language (The coming standard, version 5)</td>
</tr>
<tr>
<td>2D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>IE</td>
<td>Internet Explorer</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>Amazon Simple Storage Service</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirement Specification</td>
</tr>
</tbody>
</table>
# Table of Contents

## Chapter 1

1. Introduction ................................................................................................................................. 10
   1.1 Background .......................................................................................................................... 10
   1.2 Purpose ............................................................................................................................... 10

## Chapter 2

2. Method ........................................................................................................................................... 11
   2.1 Prototype Development ....................................................................................................... 11
   2.2 Integrated Development Environment ............................................................................... 13
   2.3 “Fail Fast” Approach ......................................................................................................... 13
   2.4 Sprint ..................................................................................................................................... 13
      2.4.1 Sprint Log .................................................................................................................... 14
   2.5 Transparent Development .................................................................................................. 14
   2.6 Requirements Specification ............................................................................................... 15
      2.6.1 Drawing Selection View (Start View) ........................................................................ 15
      2.6.2 Create Empty Drawing View ....................................................................................... 15
      2.6.3 Import Drawing View ................................................................................................. 15
      2.6.4 The Editor View ........................................................................................................... 15
      2.6.5 General Tools .............................................................................................................. 15
      2.6.6 Drawing Tools .............................................................................................................. 16
      2.6.7 Layer Toolbars ............................................................................................................ 16
   2.7 Usability Testing ..................................................................................................................... 17
   2.8 Attendance and Roles During Tests ..................................................................................... 18
      2.8.1 Definition of Usability .................................................................................................. 18
      2.8.2 Evaluating Test Data ................................................................................................... 18
      2.8.3 Quantitative Data ......................................................................................................... 18
      2.8.4 Qualitative Data .......................................................................................................... 18
      2.8.5 Usability Evaluation ..................................................................................................... 19
         2.8.5.1 Common Usability Heuristics for User Interface Design .................................... 19

## Chapter 3

3. Theory ......................................................................................................................................... 20
   3.1 Google Web Toolkit ............................................................................................................. 20
      3.1.1 GWT Canvas ............................................................................................................... 20
      3.1.2 The GWT Compiler ..................................................................................................... 21
      3.1.3 Web Browser's Significance in GWT ......................................................................... 21
   3.2 System Architecture .............................................................................................................. 22
   3.3 Drawing Tool Architecture ................................................................................................... 23
      3.3.1 MapModule .................................................................................................................. 23
      3.3.2 SaveLoadManager ....................................................................................................... 23
      3.3.3 EditDrawingView ........................................................................................................ 23
   3.4 Saving and Management of Drawings .................................................................................. 24
   3.5 Ways of Communication ....................................................................................................... 25
      3.5.1 Server and Client ......................................................................................................... 25
      3.5.2 Database SQL Queries ............................................................................................... 25
Chapter 8
8. Conclusions.................................................................48
   8.1 Accomplishments....................................................48
Chapter 9
9. Discussion........................................................................49
Chapter 10
9. References.......................................................................50
Chapter 1

1. Introduction

1.1 Background

Inspectera was founded in 2005 and is currently represented all over Sweden. The company offers consulting services in food hygiene, quality and environment. They also handle education in these areas.

For many people in the staff at Inspectera the daily work includes editing different types of drawings. Management of these drawings is time consuming and cumbersome. The first step is to obtain a blue print, the next is to do the required changes in an editor, for example Microsoft Visio. Finally, the edited drawing is uploaded to the company’s e-service Inspectera Online (IO). The vision was to optimize and simplify this process. There was a need to be able to import drawings into the e-service, edit them online in the user’s web browser and simply push save. Hence, the idea was born to develop a web-based drawing tool for this purpose that would also be integrated with the other available services in IO.

1.2 Purpose

The purpose of this project is to develop a web-based drawing tool in Google Web Toolkit/Java that is sufficiently useful, that runs with satisfactory performance, and meets the desired requirements and functionality. The tool will then be usability tested and evaluated.

Goals set:

- Determining how successful the drawing tool became in terms of functionality and performance. Has the right graphic library been selected for managing the drawing surface? Did Inspectera get what they wanted?
- Perform usability testing and a usability evaluation of the new tool.
Chapter 2

2. Method

2.1 Prototype Development

Prototypes serve several important purposes in the development of a new product, these are: to learn something in the process of experimenting with different solutions, to make the product concepts more clear and to prove their advantages, to have a communication between the customer and the developers, to create an understanding of the product's components and their integration, and finally to create a schedule of the product development process.

There exist different types of prototypes; in the development of this drawing tool, mock-up prototypes and demo prototypes have been used. The mock-up prototype's purpose was to analyze and discuss the tool's requirements with the customer Inspectera. Later in the development process, in the end of each sprint a demo prototype was shown to the customer to demonstrate the newly developed functionality. (Sarkissian, 2008)

Below follow two figures, the first one is the mock-up prototype version 0.1 designed after the first meeting with Inspectera, the second figure is the final mock-up prototype version 1.0 approved by Inspectera.

Figure 1: Prototype 0.1
Figure 2: Prototype 1.0
2.2 Integrated Development Environment

During the development of this application the Integrated Development Environment (IDE) has been Eclipse and the Google Web Toolkit (GWT) plug in. The plug in lets you debug the application as a web application by running a local web server with the GWT development mode. You are able to do changes in the code without recompiling the source code for faster testing and debugging. However, because the Inspectera Online source code is rather large this functionality makes the environment very slow. A way to overcome this problem is to keep the code that you are working on separated in its own module while developing. Then you can integrate and compile the source code along with the large scale application.

As mentioned before, IO is a large scale product and consists of tens of thousand lines of source code. Unfortunately there did not exist any documentation of the modules in IO, and there were almost no comments in the source code. Hence, it took about a week to get an understanding of the architecture and where and how things were connected.

2.3 “Fail Fast” Approach

In order to select a framework that could fully satisfy the requirements of this drawing tool within a reasonable amount of time, a “Fail Fast” approach has been used. The reason for this was to make sure that the framework will support the functionality needed in this drawing tool while not getting stuck in the middle of the development process. In the beginning of the developing phase, after the mock-up prototype had been approved by the customer, the library GWT Canvas were selected to be the first candidate for the approach, mainly because GWT were used in the other modules. Within the first two weeks, crucial methods were built and tested, such as: drawing methods for adding different types of symbols on the draw area, methods for moving and zooming the draw area, methods for loading in a blue print that could be drawn on, etc. Luckily that one candidate (GWT Canvas) was good enough for determining that the developing could continue and the framework was set.

2.4 Sprint

In the beginning of this project a sprint plan was created together with the mock-up prototype. The sprints have been about one or two weeks long each and after every sprint a meeting with the customer (Inspectera) was held. During these meetings some requirements changed, new requirements could be added, but most of the meetings were held to demonstrate the product and to plan the next sprint. A sprint log has also been documented containing what has been done week by week. Below in Table 1 follows an example of Sprint 1.

<table>
<thead>
<tr>
<th>Sprint 1 - Week 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tasks:</strong></td>
</tr>
<tr>
<td>- Get familiar with the service Inspectera Online, study the design and functionality</td>
</tr>
<tr>
<td>- Create a mock-up prototype of the new editor</td>
</tr>
<tr>
<td>- Get familiar with the development environment GWT (Google Webb Toolkit in Eclipse)</td>
</tr>
<tr>
<td>- Trial of the GWT Canvas library</td>
</tr>
<tr>
<td><strong>Results and documentation:</strong></td>
</tr>
<tr>
<td>- Prototypes of the user interface, decision whether to use the library GWT Canvas or not</td>
</tr>
</tbody>
</table>

**Table 1: Sprint 1**
2.4.1 Sprint Log

Throughout the development process a sprint log has been documented keeping track of what has been done during each week. Below follows the first week, it has been called the “Fail fast week”. During this week fundamental functionality for the drawing tool was developed, see the sprint log of sprint 1 below in Table 2. During the fail fast week, we did not have access to the IO source code yet. Hence, focus was on the trial of the framework GWT Canvas.

<table>
<thead>
<tr>
<th>Sprint 1 – Week 5 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Mock-up prototype of the GUI</td>
</tr>
<tr>
<td>• Got familiar with Inspectera Online</td>
</tr>
<tr>
<td>• The first drawing function, it drew a picture onto the canvas, e.g. a blueprint.</td>
</tr>
<tr>
<td>• The drawing functions and functionality for adding symbols on top of the blueprint.</td>
</tr>
<tr>
<td>• Zooming functionality, zoom in and out on the canvas.</td>
</tr>
<tr>
<td>• Moving functions for the blueprint, such as moving the canvas horizontal and vertically.</td>
</tr>
<tr>
<td>• Moving functions for symbols, drag and drop them on to the blueprint.</td>
</tr>
</tbody>
</table>

Table 2: Sprint log, spring 1

2.5 Transparent Development

During the development of the drawing tool a live testing server was hosted at all time containing the latest changes and functionality. This gave the staff at Inspectera the possibility to test the drawing tool and to provide feedback and critique during the sprint meetings throughout the development process. At the end of each week Inspectera was informed of the latest additions to the drawing tool and they were provided with a link to the hosted web server. This development method requires the developer to be flexible and open minded to changes in the system requirements.
2.6 Requirements Specification

The final requirements of the drawing tool are listed in this section.

2.6.1 Drawing Selection View (Start View)

1.1 The system shall load the unit's available maps and present them to the user in a list format.
1.2 The editor shall launch and load the selected drawing when it's name is pushed.
1.3 There shall be a button to delete a drawing and the user shall have to confirm this action.
1.4 There shall be a button to open the view for creating an empty drawing.
1.5 There shall be a button to open the view for importing a drawing.
1.6 There shall be an option to change the name of a drawing.

2.6.2 Create Empty Drawing View

2.1 There shall be a text field for the user to fill in the name of the drawing.
2.2 There shall be a size selection: small, medium and large.
2.3 There shall be a format selection: horizontal and vertical.
2.4 There shall be a cancel button that returns the user to the start view.
2.5 There shall be a create button that is only click-able when the name of the drawing has been given.

2.6.3 Import Drawing View

3.1 There shall be a text field for the uploaded drawing for editing its name.
3.2 There shall be a cancel button that returns the user to the start view.
3.3 There shall be an indication during the upload process.
3.4 Once the upload process is complete, the system shall return the user to the start view.

2.6.4 The Editor View

4.1 There shall be a toolbar with the general tools listed in section 2.6.5.
4.2 There shall be a toolbar with the drawing tools listed in section 2.6.6.
4.3 There shall be a toolbar with check-boxes, one check-box for each available layer, see section 2.6.7.
4.4 Each checked check-box shall spawn a toolbar with the layer specific symbols.
4.5 Each checked check-box shall set the placed out symbols belonging to this layer visible.
4.6 Each unchecked check-box shall set the placed out symbols belonging to this layer invisible.
4.7 There shall be a field for displaying information about the marked symbol object.
4.8 The user shall be able to change the values for the marked symbol object.
4.9 There shall be an indication as long as drawing is being loaded.

2.6.5 General Tools

5.1 Marking Tool
5.1.1 A drag and drop mouse action shall move the visible drawing area or the selected marked object.
5.1.2 The left mouse button shall mark an object.
5.2.3 A click with the left mouse button on a marked symbol shall display information about the symbol.

5.2 Zoom In Tool
5.2.1 Shall zoom in the visible area of the drawing by 25% when pushed.
5.3 **Zoom Out Tool**  
5.3.1 Shall reverse the zoom level of the visible area of the drawing by 25% when pushed.

5.4 **Remove Marked Object**  
5.4.1 Shall delete the current marked object from the drawing.

5.5 **Undo Symbol Object**  
5.5.1 Shall delete the last added symbol object from the drawing.

5.6 **Save Drawing**  
5.6.1 Shall save the drawing and its objects and notify the user with a confirmation dialog.

5.7 **Load Drawing**  
5.7.1 Shall load in the last saved state of the drawing.

5.8 **Print Drawing**  
5.8.1 Shall open a confirmation dialog with a link to the generated image file.

### 2.6.6 Drawing Tools

6.1 A drag and drop mouse action shall add the selected draw object to the drawing.

6.2 Available objects shall be:
- Line object
- Square object
- Circle object
- Wall object
- Door object
- Window object

### 6.7 Text Object

6.7.1 The system shall ask the user to fill in a text field.

6.7.2 The user shall be able to place the text on the drawing with the right mouse button.

### 6.8 Undo Draw Object

6.8.1 Shall remove the last added draw object from the drawing.

### 6.9 Color Selection

6.9.1 A color selection shall affect the next object drawn by the user.

6.9.2 The available colors shall be: black, white, red, blue, green and yellow.

### 2.6.7 Layer Tool bars

Each toolbar shall contain layer specific symbols specified by Inspectera, e.g. commonly used fire protection symbols.

7.1 Layer types of the drawing shall be:
- Pest control
- Fire Safety
- Warning
- Forbidden
- Edict
- Dangerous goods
- Information
- Hygiene & Safety zones
- Inspections
- Deviations

7.2 The inspection toolbar shall be dynamically loaded with the unit's current activated inspections.

7.3 A symbol shall be placed on a drawing by the user left mouse button.
2.7 Usability Testing

This is a testing method used to make sure that the users of a system are able to accomplish the intended tasks efficiently, effectively and with a sense of satisfaction. Usability testing does not cover the functionality of a system, it evaluates the user interface design. As this drawing tool is a completely new service in IO, usability testing is a great method for discovering unknown issues and to determine whether there exist any serious usability problems in the graphical user interface (GUI). (Miami University, 2004)

2.8 Attendance and Roles During Tests

Depending on the kind of product being developed the roles involved in usability testing may differ. The following roles are usually required to attend during a test:

- A representative user
- A test host
- At least one developer (observer)
- At least one business representative (observer)

However, the author of this thesis is the only developer of this product and during the test he acted as the testing host and the observer. To aid him, a small program called “Click Counter” has been used to record the test user's number of mouse clicks per task to measure the efficiency. The observer could focus on user errors and assist the test user as the testing host instead of counting the mouse clicks. Furthermore this is a student project and the business representative has not been asked to attend during usability testing.

2.8.1 Definition of Usability

Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. (ISO 9241-11, 1998)

Usability may be measured in qualitative or quantitative terms, or both. User feedback is the best way to gain knowledge of the usability in a new system but, you may as a developer recognize some of the usability problems on your own. User feedback about the quality of the product is qualitative, but a measurable result such as how many clicks does it take for a user to complete a task is quantitative. Few common criteria used to define usability are: Learnability, Memorability, Efficiency, Error tolerability and Likeability. (Miami University, 2004)

While reporting or documenting the usability in a system you can note the severity of problems, e.g. Critical, Serious and Minor. A Critical problem stops the user from completing the scenario and have to be fixed. A Serious problem would be something that makes a lot of the users frustrated, and should be fixed, otherwise the users might just give up. A Minor problem usually annoys the users but does not keep them from completing the scenario. During testing sessions this is something the observer has to recognize and take note of. (Usability.gov, 2013)
2.8.2 Evaluating Test Data

A way to find usability problems is as mentioned before, user testing. Testing data has been collected during user testing in order to evaluate several usability criteria for this drawing tool. Usability criteria chosen were efficiency, errors/error tolerability, successful task completion rate, likeability/satisfaction and learnability. Test data has been collected through observations of the test users and by letting the users answer a questionnaire after completing the test. In addition, all the test users drawings are saved on the server and can be further examined after the tests.

The observer can identify criteria such as Successful Task Completion, Critical Errors, Non-Critical Errors, Error-Free Rate, Time On Task during a test. But, criteria such as likeability and satisfaction will be evaluated through analyzing the answers to the questionnaires. (Usability.gov, 2013)

As mentioned in section 2.6.2 two types of test data are defined, qualitative data and quantitative data.

2.8.3 Quantitative Data

This type of data includes all the data that is numerically measurable and is almost always analyzed by statistics. Chosen criteria that are possible to analyze via quantitative data collection are efficiency, errors/error tolerability and successful task completion rate. The observer plays an important role when collecting this data.

**Efficiency and Task Time**
To analyze these criteria you can keep track of how quickly the user performs the tasks. You could also measure how many clicks the users requires to accomplish tasks.

**Errors/error tolerability**
By keeping track of how many errors the users do during the tests and how severe these errors are we can measure these criteria.

**Success & Successful Task Completion Rates**
This is a fundamental criterion and easy to collect. Statistics of how many of the tasks that were fully completed by the users without any critical errors will be analyzed.

2.8.4 Qualitative Data

The qualitative data includes all the data that is non-numerical. The goal with the qualitative data collection is to cover the rest of the chosen criteria, likeability/satisfaction and learnability. The data collection will mostly consist of answers to the questionnaire.

**Likeability & Satisfaction**
Likeability or satisfaction is a subjective criterion, the users will be asked to set a score on a Likert scale of how they felt about performing tasks in the drawing editor.

**Learnability**
This is a subjective criterion designed for new users. We measure how easy it is for users to accomplish basic tasks the first time they use the system. There will be a question on the questionnaire asking the user how easy it was to perform tasks and use different tools on a Likert scale. The observer will also give his judgment of how the users performs in the observer protocol.
2.8.5 Usability Evaluation

There are several methods for evaluating the usability in a new application. One method is the well-known heuristic evaluation developed by Jakob Nielsen and Rolf Molich that helps you to structure the critique of a system using simple and general heuristics. This method is usually performed by a group of experts in usability. The experts first inspect the system individually and once every evaluator completes his inspection they are allowed to communicate with each other about their discoveries. Ten common heuristics which aid the evaluators in these inspections are listed in section 2.6.1, below. (Nielsen, 1994)

2.8.5.1 Common Usability Heuristics for User Interface Design

Jacob Nielsen in collaboration with Rolf Molich developed a heuristic evaluation for user interface design in the 1990's. By analyzing 249 usability problems, they derived the heuristics with optimal explanatory power. (Nielsen, 1994)

Below follows the standard ten common heuristics for user interface design.

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help user recognize, diagnose and recover from errors
- Help and documentation
Chapter 3

3. Theory

3.1 Google Web Toolkit

3.1.1 GWT Canvas

The graphical framework or library selected for handling the drawing surface was GWT Canvas. It uses the canvas element that is a new feature in the upcoming standard for HTML, HTML5. The canvas element handles 2D shapes and bitmap images and it consists of a drawable area coded in HTML. The GWT Canvas framework allows many functions for accessing and editing the drawing area. Java code is written with the GWT Canvas library and later translated into Java scripts by the compiler. The set of drawing functions available in the library are common to any other 2D graphical APIs. In this drawing tool functions used frequently to edit graphical objects are: `setWidth`, `setHeight`, `setColor`, `setCoordinates` etc. But, more complex functions are also being used, such as `translate` and `scale`. The `translate` function applies a translation to the drawing area. In this tool that function is often used to move the visible area of the drawing since you need to be able to work on different parts of a drawing. The `scale` function is used to zoom the visible area up and down. Maintaining a 2D coordinate system in order to handle drawing object's position is rather straightforward. However, once the visible drawable area is being moved and scaled up and down, it becomes complex. Below in Table 3, follows a code segment of the zoom-in method that translate and scales the canvas, it also applies the necessary updates to the coordinate system's variables.

```java
public void zoomIn() {
    // set zoom to 125%
    zoom = 1.25;

    // calculate the new x and y center position
    double xPosition = (-currentX * zoom) + currentX;
    double yPosition = (-currentY * zoom) + currentY;

    // move and scale the canvas
    backgroundContext.translate(xPosition, yPosition);
    backgroundContext.scale(zoom, zoom);

    // update the offsets after moving/scaling
    offsetX += xPosition*currentActiveZoom;
    offsetY += yPosition*currentActiveZoom;

    // draw and update the canvas
    updateCanvas(backgroundContext, mapContext);
}
```

Table 3: Code example of the zoom-in method
3.1.2 The GWT Compiler

The beauty of the GWT compiler is that it compiles java source code into six different permutations. Each permutation gets its own specific java scripts which works in a specific web browser also called user agent, the current available user agents in GWT are Internet Explorer 6 (IE 6), IE8/IE9, Firefox, Chrome, Opera or Safari. Cross-Browser compatibility is a goal that is rather hard to achieve. To develop a web application with full cross-browser compatibility can take months, even for a small scale application, however GWT gives you an advantage. Even though the application runs well on every user agent, it does not mean that they always behave in the same way or in the way they are intended to. You still have to spend time testing your application with different user agents. Some functionality might be missing in one of the user agents or there might be a bug etc. See the next section for examples of problems that arose when developing this drawing tool.

3.1.3 Web Browser's Significance in GWT

As mentioned in section 3.1, the GWT Canvas library is built on the HTML5's canvas element. This means that the user's browser have to support HTML5 to be able to use this drawing tool. Currently every browser supports HTML5 except Internet Explorer below version 9.0. However, the new HTML standard will be set in the future and company's that are still using older web browsers such as IE 7, have to perform updates in order to stay in touch with the new technology. Typically, Google's own web browser (Google Chrome) runs the GWT Canvas library a bit smoother than the other browsers such as Firefox and Safari, but it's not an issue, all the functions in this drawing tool is accessible in every browser.

Another functionality of the canvas element is the “toDataURL()” function, it creates an URL containing an image data URI scheme of the canvas area encoded in the Base64 format. This is a good solution for implementing the print function in this drawing tool and it is also used for that purpose. However, Internet Explorer (IE) does not support the Base64 data format and therefor it can't be used by IE users. To overcome this problem the image data is decoded and saved as a bitmap image at the server and then uploaded to the user. It works, but this solution requires an extra round trip to the server and it affects the performance negatively.
3.2 System Architecture

Inspectera Online originally consisted of four components, a Web Server hosting the website/WebAPI hosting the website/Web Client which clients connect to, a Server that handles all the communication with the Relational Database and the document/file database. The Relational Database is hosted in language SQL (Structured Query Language) and the Document/File Database is stored in the “Cloud”, at an Amazon S3 (Amazon Simple Storage Service). During the development of this drawing tool an Android application were also developed by another student. Thus, a Field HTTP API was added and used to handle the connection between the server and the Android Application. Figure 3 below shows an overview of the IO's system architecture.

Figure 3: Inspectera Online System Architecture

Image source: Inspectera HK AB
3.3 Drawing Tool Architecture

Each page or activity of Inspectera Online is built with the Model View Presenter (MVP) design pattern further explained in section 4.2. The main business logic on the client side of the drawing tool is divided into the classes “MapModule”, “SaveLoadManager” and “EditDrawingView”. There exist more views such as “DrawingView” for selecting a drawing, “UploadDrawingView” for uploading a blueprint, and “CreateEmptyDrawingView” for the creation of empty drawings.

3.3.1 MapModule

The MapModule handles the canvas area, it includes all the methods such as drawing objects, drawing symbols, drawing the blueprint, refresh, movement of objects and symbols, movement of the whole canvas, zoom in and out, selection and collision calculations (2D coordinates), and methods for adding symbols and other drawn objects such as lines, walls or texts. It also contains logic for printing the canvas, obtaining the current image data with the current activated layers.

The MapModule contains about 60 private variables for handling the 2D coordinate system, the activated/deactivated layers, the symbols and objects, the blueprint itself, etc. There exist about as many public methods available to the EditDrawingView for user interaction via the graphical user interface (GUI).

2D Coordinate system

To handle the somewhat complex 2D coordinate system it requires many coordinate variables. Important methods such as movement of the canvas, zoom in and out, movement of drawn objects, or adding more symbols and objects are all part of using these variables.

Draw Methods

There exist many draw operations. One for each specific object such as a wall, door, text or a line. Fortunately only one method for drawing symbols is required since the symbol's image data is fetched dynamically. See the MapObject in the UML diagram in figure 4, in section 3.4 for a complete model of the symbol entity.

Collision Detection Methods

Almost every interaction with the canvas area is made with the user's mouse pointer. Thus, it requires a lot of collision detection, e.g. Did the user push on one of the placed symbols or objects, or did he try to add a symbol outside the blueprint, etc.

3.3.2 SaveLoadManager

This class is responsible for handling the save and load function by using the remote procedure calls (RPC) design pattern to communicate with the server's “DrawingService” which contain logic for storing or loading the metadata of a drawing in the relational database. More on this in the next section 3.4.

3.3.3 EditDrawingView

This is the view component of the MVP design pattern, see section 4.2 for more information about MVP. It contains the graphical user interface (GUI) such as tool bars, and buttons, and the canvas area. See the GUI section 4.7 for more information of the drawing tool's GUI.
3.4 Saving and Management of Drawings

In any drawing tool a basic requirement is to be able to save and load your work. A drawing contains a lot of information and is usually saved in some kind of file structure in a save file or directly into a bitmap format located on your computer. However, this is an online editor and the uploaded blueprints (image files) are saved in the file database, and “text information” such as coordinates and identifiers belonging to drawn objects are saved in the relational database. You could save the drawing in a bitmap format in the file database directly but, then you would lose the ability to easily move and delete objects drawn on top of a blueprint after saving or loading it. In fact that was one of the basic requirements requested by Inspectera. You want to be able to place out symbols and text, by ease move and delete some of them, without altering the blueprint. Hence, the blueprint is loaded in the background and symbols and text are drawn on top in several layers.

The drawing's metadata needed to be stored in this case are for example a file identifier for an uploaded blueprint or an empty sketch board, all the drawn objects and texts which all has specific coordinates, a specific layer, a width and a height, etc. Fortunately it's quite easy with the help of tools already implemented in IO to store and load the necessary relationships and data required for a save file structure like this in the relational database. See section 3.5.1 for technical details of how this is done.

An object placed on a drawing has either been called a DrawObject or a MapObject depending on if it is just a line or text with no image data (DrawObject) in contrast to if it is an image symbol of some kind that needs an image source (MapObject). The drawing entity itself is simple called Drawing. The relationship between a Drawing and DrawObject or MapObject is one-to-many. See figure 4 for a simplified UML class diagram of the Drawing class.

```
Drawing
- id: Integer
- fileId: Integer
- mapObjects: List<MapObject>
- drawObjects: List<DrawObject>
- drawingName: String
+ getMapObjects(): List<MapObject>
+ setMapObjects()
+ getDrawObjects(): List<DrawObject>
+ setDrawObjects()
+ getDrawingName(): String
  + setDrawingName()

+ type: String
+ name: String
+ width: Integer
+ height: Integer
+ imageUrls: String
+ posX: Double
+ posY: Double
+ getType(): String
  + getWidth(): Double
```

```
Figure 4: UML class diagram of the Drawing class
```
3.5 Ways of Communication

3.5.1 Server and Client

Server-sided code that is being invoked by a client is often called a service. For this drawing tool a "DrawingService" has been created to handle the save and load functionality. By using GWT's remote procedure calls (RPC) the client can invoke the DrawingService by inheriting its interface. The RPC's gives you the ability to pass java objects back and forth between the client and the server via HTTP. When saving a drawing, the client simply sends the drawing object to the DrawingService which handles the logic and the mappings to the relational database. (GWT Project, 2013)

3.5.2 Database SQL Queries

The requirements of this drawing tool involves integration with other modules that are parts of IO. Mainly active self-checks (Inspections) shall be loaded in the editor so the user can place them on a drawing. While browsing through self-checks it also had to be possible to find out which drawings that contained a specific self-check. To solve this, two SQL queries within the java code had to be used and a new method to the DrawingService were added. The method in Table 4 returns a list of drawings which contains the self-check with a particular name.

```java
public List<Ritning> findByContainingSelfCheckName(String selfCheckName) {
    // Fetch all the map objects with selfCheckName
    Query query = entityManager.createQuery(
        "SELECT OBJECT(mapobject) " +
        "FROM MapObject mapobject " +
        "WHERE mapobject.name like :selfCheckName");
    // Use wild cards here, e.g. selfCheckName + Pest control
    query.setParameter("mapobjectname", "+" + mapObjectName + "%");
    List<MapObject> mapObjects = query.getResultList();

    // Find all the drawings where the mapObject exists
    List<Ritning> ritningar = new ArrayList<Ritning>();
    for (MapObject mapObject : mapObjects) {
        // Only add if it's a control, (deviation objects also contains the name)
        if (mapObject.getType().equals("Kon")) {
            Query queryTemp = entityManager.createQuery(
                "SELECT OBJECT(ritning) " +
                "FROM Ritning ritning " +
                "WHERE ritning.fileName = :fileName");
            queryTemp.setParameter("fileName", mapObject.getfileName());
            Ritning tempRitning = (Ritning) queryTemp.getSingleResult();
            if (tempRitning != null) {
                ritningar.add(tempRitning);
            }
        }
    }
    return ritningar;
}
```

Table 4: Method with SQL Queries
Chapter 4

4. Design and Integration

4.1 Design Choices

Due to the fact that this tool will most likely be further developed after the end of this thesis and that it's just one of many modules in IO the design choice was simple. The architectural design is consistent in every other module and there was no need to choose a different approach for this new module. The architectural design used in IO is model view presenter (MVP), commonly used in android applications.

4.2 MVP Architecture

Model view presenter (MVP) is a design pattern that is very useful in large scale applications such as IO. It gives you the benefit to have multiple developers work on the same code base and at the same time maintain legacy features and functionality. The MVP design pattern separates functionality into four main components, these are a model, a view, a presenter and a AppController. (Ramsdale, 2010)

Model
A model contains business objects. In this drawing tool there exist several models. The most important ones are:

- Drawing: This object contains all the data and information about a drawing. It has a unique identifier, an id to the uploaded blueprint or to an empty sketch, a file name, and the drawing's corresponding map objects and draw objects.
- MapObject: This is an object that is placed on a drawing that displays an image from a source. It has its specific X and Y coordinates, a width, a height, a type, a link to the source image, and an unique identifier. The type is used for dividing symbols in different layers, which gives the user the ability to show only one or more types of symbols at the time.
- DrawObject: This is also an object placed on a drawing, but it does not display an image when drawn. It can be a straight line, a circle, a text or a square etc. The object contains a unique identifier, a color, starting X and Y coordinates, and ending X and Y coordinates.

View
A view consist of user interface components, such as buttons, tool bars, text boxes and in this case also the drawing area (the canvas). The drawing tool required a total of four views, these are: DrawingView, EditDrawingView, AddDrawingView, AddEmptyDrawingView. The first view is where the user can select a drawing be opened in the editor, or to upload a new drawing, or to create a empty drawing. By selecting one of these options the user moves to another view. More on the drawing tool's GUI is found in section 4.7.
**Presenter**
The presenter contains all the logic of an application. It also involves history management and the functionality to switch to another view. Further more it contains data sync functionality such as Remote Procedure Calls (RPC) back to the server. A general rule in this design pattern is to create one presenter for every view to load and maintain the view and to handle events originated from the view's UI. The drawing module consist of four views, which means there is four presenters handling the logic of these views.

**AppController**
This component is located in the application layer. Logic that is not made specific for a single view is placed here. The AppController also contains the history management and the logic for view transitioning that is tightly coupled to the history management.

### 4.3 Fundamental Methods

As this is an editor with a graphical interface, most of the user's work is being done with the mouse pointer. Thus, a lot of the functionality has to involve the drawing's coordinate system, the user's mouse coordinates and mouse clicks. It took a lot of time to develop these features as well as to test, improve and correct them. The main methods are listed below:

- **Mouse Events, Coordinates and Collision Detection**
  Clicks and coordinates of the user's mouse pointer relatively to the exact location in the drawing's coordinate system. It involves offsets in X- and Y- coordinates depending on which area of the drawing that is displayed during the time as well as which zoom level that is currently active.

- **Active Tool**
  There is quite a lot of different tools and every tool is used and activated by the left mouse button. Thus, it requires the left button listener function to be complex and flexible. You also need to keep track of which tool is active at all time.

- **Refreshing the Drawing Area**
  The drawing area needs to be refreshed almost all the time, whenever a user performs an action, otherwise the user would get the feeling that the GUI does not respond. E.g. While moving a symbol over the canvas, the symbol is displayed at the mouse pointer at all time making it easier to place the symbol in a new position. This requires that the canvas is refreshed every time the user moves the mouse while moving symbols.

### 4.4 Following the Three-click Rule

The three-click rule suggest that a user should be able to access any information or feature of the application in no more than three clicks, otherwise they will most likely leave the website. (Zeldman, 2001)

However, this unofficial rule of usability has been criticized a lot during its years of existence. Studies shows that the users does not mind clicking as long as they feel confident that they are able to navigate and find what they were looking for. The number of clicks does not affect the satisfaction a user experience. (Porter, 2003)

After all the graphical user interface (GUI) of the drawing tool has been designed so that every feature can be reached by the maximum of three clicks. Usually the user needs only one or two clicks, e.g., mouse click one enables a layer the user wants to work with in the layer menu, mouse click two selects a symbol to be placed out from the corresponding layer's toolbar, and at last, mouse click number three puts out the symbol at the location of the mouse pointer. See section 4.7 for the more information about the GUI.
4.5 Touch Pad Support Problems

Inspectera Online is designed to be fully functional on a touch pad, such as the iPad. Therefore, the drawing tool was meant to include support for touch pads if there was enough time left during the development process.

The menus and buttons were already working properly but, interaction with the drawing area requires the user's touch coordinates.

The idea was simple, to reuse and separate the mouse event listener functions to handle the touch events as well. The only difference was the collection of the user's touch coordinates and to use these instead of mouse coordinates. After some testing and development, the drawing functions worked well with the iPad. However, during testing with other touch devices it turned out that different versions of the iOS affected the collection of the touch coordinates. Hence, the drawing tool worked fine on some versions of iOS but, not on others. Without the correct collection of the user's coordinates the drawing tool becomes unusable.

Finally, the drawing tool was designed to work with the latest version of iOS (iOS 6 at that time). Tests has been made both on the iPad and the iPhone.

Inspectera Online is currently being run with GWT version 2.3. It might be possible that the newest version of GWT (version 2.5) resolves the issues with the touch event coordinates.

4.6 Integration with the Android Application

During this project, an Android application was developed by another student. We worked together on a solution of how to modify and extend the current self-check entity (inspection entity).

Self-check extension
A user shall be able to take a photo with his/her phone and attach the photo to a self-check. The photo should then be displayed inside the drawing tool, inside the information pop up. The information pop up is displayed if a self-check placed on drawing is clicked on by the user.

My part of the integration was to prepare the server's store function of the image data and extend the self-check entity with the self-check photo. Methods for fetching the image data from the server has also been written. The self-check's with an attached image can be displayed inside the drawing tool, see Figure 5 below for an illustration of how the information pop up may look.

Figure 5: Information pop up with a self-check photo
4.7 Graphical User Interface

The graphical user interface (GUI) is built up by simple tool bars with self-explanatory images as buttons, and by hovering a button a help text appears, telling the user information about what the button does. The tool bars are divided into categories, e.g. Pest Control Symbols, Drawing Tools and Interaction Tools. Figure 6 displays a screen shot of two tool bars.

![Figure 6: Tool bars in the drawing editor](Image)

The first tool bar in the side menu, right to the drawing area contains check boxes, one check box for each layer. The user is able to choose which of the layers that shall be active. Perhaps the user only want to work with the layer self-checks, he/she then deselect every other layer in the side menu. The draw objects in the “hidden” layers will be invisible on the drawing surface. Figure 7 displays a screen shot of the layer tool bar.

![Figure 7: Layer tool bar in the drawing editor](Image)

The second tool bar in the side menu is called “Marked Symbol” which contains information about the currently marked symbol/pushed on symbol. The menu consist of text boxes displaying coordinates, the symbol’s identifier and the symbol’s width and height. By changing the values in these text boxes the user can change size and position. But, the easiest way to change the position of a symbol is to use the mouse pointer and simple drag the symbol to a new location. See Figure 8 for a screen shot of the marked symbol tool bar.

![Figure 8: Marked symbol tool bar of the drawing editor](Image)

Except of the tool bars, there is of course the drawing area, it can easily be moved around and zoomed in and out by using the interaction tools. The GUI is designed to be rather intuitive, but this will be evaluated in the usability tests. A screen shot of the complete GUI in Inspectera Online can be seen in Figure 9.
To help users to know what is going on, pop up windows with information is used, e.g. “The drawing has been saved.” when a user presses save or “You can not place a symbol outside of the drawing area.” if a user tries to add objects outside the drawing. See Figure 10 for example of a pop up.

![Figure 10: Pop up in the drawing editor](image)

High resolution blueprints can take a few seconds to load, to display this for the user a spinning loading indicator is used, see Figure 11.

![Figure 11: Loading indicator in the drawing editor](image)
There also exist a help bar, with tips and information about how to use the tools in the editor. It’s located to the right of the tool bars and the drawing area. By clicking the question mark you either hide or show the help bar. Tool’s names are written with bold text so they are easily spotted by the user. The help bar is part of the original system, there exist one help bar for each activity in IO. The help bar is shown in Figure 12.

**Figure 12: Help bar of the editor.**
Chapter 5

5. Usability Testing

5.1 Test Design

To be able to design and identify appropriate testing tasks it's important to gain knowledge of how the system will be used by the staff at Inspectera. This has been done during the sprint meetings. After each demo of the drawing tool, the CEO at Inspectera explained how the new functionality will be used or expected to be used. Two possible scenarios could then be constructed and from these representative test tasks were derived.
5.1.1 Scenario 1: No Blueprint of a Pizzeria

Anna is out in the field working for Inspectera. Her job today is to place out two traps for pest control at a pizzeria and to create a drawing with the locations of these traps. After placing out the traps, she asks the owner of the shop for blueprints of the building. The owner says that he had never seen anything like that, and he doesn’t have access to any image or PDF online. Luckily a pizzeria is rather small and it does not take long for Anna to make a sketch of the place in the online drawing tool. She logs into IO on her laptop, browse to the customers page and create a new empty drawing. She outline the pizzeria and puts the two symbols of the traps at the right location on the drawing. She saves the drawing and then she tells the owner that he can access it online in IO. Anna wishes the owner a good day and leaves the pizzeria.

Bellow in Figure 13 follows a screen shot of how a sketch of a pizzeria with placed out pest control symbols may look like in the online drawing tool.

![Figure 13: Screen shot of the drawing tool, sketch of a pizzeria with pest control symbols](image-url)
5.1.2 Scenario 2: Placing out Self-Check symbols on a blueprint

Johan is working at Ica Kvantum Hageby in Norrköping and he is responsible for all the freezers and fridges in the store. To keep track of every freezer and fridge Johan uses IO's service of self-checks. Everyday he gets reminded to check and keep track of their temperatures. Johan does not work during the weekends, thus whoever that works on Saturdays or Sundays have to do the self-checks for him. However, Johan knows that some of the self-checks were never completed during weekends and he has been hearing about few complains from his co-workers that it takes too long to find each freezer and fridge in the store. Johan decides to place out each self-check on a blueprint of the store and print it out to aid his co-workers. The self-check service is integrated with the drawing tool, so all he got to do is to upload a blueprint of the store and place out the existing self-checks. Figure 14 displays a screen shot of how a blueprint with self-checks may look like.

![Figure 14: Screen shot of the drawing tool, placed self-checks on a blueprint](image)
5.1.3 Designing Testing Tasks

By using the two scenarios from the previous section testing tasks could be derived by examining the steps a user have to take in order to complete the tasks in the scenarios. Below follows the testing tasks that has been used in the usability evaluation. These tasks cover the main functionality of the drawing tool while not being too complicated for a new user.

Task 1:
Create an empty drawing, name the drawing with your own first name and last name. Choose the small size and the horizontal format.

Task 2:
Start the drawing editor with the drawing you just created.

Task 3:
Place three types of pest control symbols in the top left corner of the drawing.

Task 4:
Outline a pizzeria with the drawing tools. The sketch have to consist of walls, one door, and two windows. See an example in Figure 15, below:

![Figure 15: Sketch of a pizzeria](image)

Task 5:
Place a fire extinguisher in the bottom right corner of the drawing. Set it's width and height to 100.

Task 6:
Save the drawing and exit the editor by navigating to drawings in the side menu.

Task 7:
Upload the blueprint located on the desktop, name it “your first name & last name T7”. e.g. Alf Bjelkenstedt T7

Task 8:
Start the editor with the uploaded blueprint.

Task 9:
Place any of the pest control self-checks in the top left corner of the drawing.

Task 10:
Save the drawing and exit the editor by navigating to drawings in the side menu.
5.2 Observer Protocol

A protocol to be filled out by the observer has been designed.

Things that shall be recorded in the protocol for each task are:
- Think-aloud observations
- Non-critical errors
- Critical errors
- Number of clicks
- Amount of Time
- Hesitations

5.3 Test User Questionnaire

By observing the test users performing their tasks, usability problems can be discovered but, a common and effective way is to collect user feedback by letting the test users answer a questionnaire. A questionnaire answered after the product has been tested is called a post-test questionnaire.

In the questionnaire you can let the user provide ratings for subjective criteria such as Satisfaction, Ease Of Use, Ease Of Finding Information etc in a Likert scale.

A questionnaire has been constructed where the user shall answer different types of questions, such as yes or no questions, Likert scale questions (1 – 5) and answer questions freely with a written sentence.

5.3.1 Yes or No Questions

- Did you manage to complete every single task during the test?

5.3.2 Likert Scale Questions

The scale is set to one to five, one means it was very difficult and five that it was very easy.

- Overall, how did you feel about performing tasks in the system?
- How was it to find a specific tool or symbol in the system?
- How was it to place out a symbol on the drawing area?
- How was it to use the sketch-up drawing tools, e.g. drawing a wall?
- How easy was it to understand the system design?

5.3.3 Free Text Questions

- Was there one or more tasks that you could not complete, please state the number(s) of the task?
- What do you think should be changed in the system?
- State two functions that you think are good and usefule in the system.
5.4 Testing Targets

Students and friends to the author of this thesis has been selected for testing the drawing tool. Jakob Nielsen wrote in 2012 that you almost find as many usability problems with five test users as you would find with many more. (Nielsen, 2012) Thereby the number of test users were decided to be only five. None of the participants did see or been told about the drawing tool before they tested it.

5.5 Conducting the Tests

The usability tests has been carried out with one user at the time. The same laptop has been used in all the tests. The test participants could choose which web browser they would like to use. Time per task and number of clicks per task has been measured. Users has been encouraged to think-aloud when they performed the tasks. The observer has filled in the observer protocol. After the participants completed the tests they were asked to fill in the questionnaire.

5.6 Goals and Requirements Set

In this section goals for the usability tests are set and explained.

5.6.1 Quantitative Goals

Efficiency and time on task

To be able to state something about these criteria you need something to compare with. Two ways of measuring these criteria (time on task) are in time and in number of clicks.

In order to set a goal or requirement regarding the maximum number of clicks per task, the number of clicks has been calculated during an optimal test case where no more clicks than needed per task has been used. Additionally by multiplying the minimal number of clicks with a constant we can set a requirement of the maximum number of clicks per task. As a start the constant is set to 1.50 which means the test users are required to complete each task with less than 150% of the clicks in the “optimal case”.

The second criteria for efficiency, time on task, is more difficult to determine. However, a fair suggestion by the examiner was to double the amount of time the developer needed in order to complete the test tasks as a requirement for time on task.
Minimal number of clicks per task
The number of clicks has been calculated during an optimal test case, see the results below in Table 5. Column three shows the set requirement of maximum number of clicks per task using the optimal number of clicks times 1.50. If the number is not an integer, its rounded up. E.g. 1 * 1.50 = 1.50, is set to 2.

<table>
<thead>
<tr>
<th>Task #:</th>
<th>Minimal number of clicks per task:</th>
<th>Requirement on number of clicks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>≤ 8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>≤ 2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>≤ 11</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>≤ 17</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>≤ 8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>≤ 8</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>≤ 12</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>≤ 2</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>≤ 6</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>≤ 8</td>
</tr>
</tbody>
</table>

Table 5: Minimal and requirement of number of clicks per task

Time per task
The time per task has been measured while the developer of the drawing tool performed the testing tasks. The requirement has then been set to an amount of twice the time the developer needed. See Table 6 below for the results. The time on task starts when the user has read and understood the task and starting to perform it.

<table>
<thead>
<tr>
<th>Task #:</th>
<th>Time per task (developer):</th>
<th>Requirement on time per task:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27 s</td>
<td>&lt; 54 s</td>
</tr>
<tr>
<td>2</td>
<td>7 s</td>
<td>&lt; 14 s</td>
</tr>
<tr>
<td>3</td>
<td>23 s</td>
<td>&lt; 46 s</td>
</tr>
<tr>
<td>4</td>
<td>46 s</td>
<td>&lt; 92 s</td>
</tr>
<tr>
<td>5</td>
<td>33 s</td>
<td>&lt; 66 s</td>
</tr>
<tr>
<td>6</td>
<td>19 s</td>
<td>&lt; 38 s</td>
</tr>
<tr>
<td>7</td>
<td>42 s</td>
<td>&lt; 84 s</td>
</tr>
<tr>
<td>8</td>
<td>6 s</td>
<td>&lt; 12 s</td>
</tr>
<tr>
<td>9</td>
<td>18 s</td>
<td>&lt; 36 s</td>
</tr>
<tr>
<td>10</td>
<td>6 s</td>
<td>&lt; 12 s</td>
</tr>
</tbody>
</table>

Table 6: Time on task in seconds
Errors and error tolerance
The goal is that no critical errors shall be made during the user tests. Few minor errors are expected to occur as well as a single or two severe errors. Of course the goal is to minimize the number of errors.

Success & Successful Task Completion Rates
The goal is that every user shall complete the whole test scenario.

5.6.2 Qualitative Goals
Two criteria has been chosen to be included in the evaluation that concerns qualitative measures, these are: Likeability/Satisfaction and Learnability. The aim is to achieve at least the average score of four on a Likert scale that stretches from one to five for each criteria.
5.7 User Testing Evaluation

The test data reported in this section has been summarized and collected from observation protocols, and questionnaires that has been filled out during a total of five test sessions.

5.7.1 Quantitative Test Data

Time per Task
As the amount of test users are small, the average task time could get skewed up by one person taking a lot of time per task while the other four users need less time. Therefore the geometric median has been used instead of the normal mean. See the results in Table 7 below.

<table>
<thead>
<tr>
<th>Task #:</th>
<th>Average time per task (geometric median) [s]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 7: Average time per task

Number of clicks per task
The mean of the five test results are reported in table 8 below. The numbers have been rounded up if not an integer.

<table>
<thead>
<tr>
<th>Task #:</th>
<th>Average number of clicks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 8: Average number of clicks per task
The observer has recorded that most non-critical errors have been made during task number 4. See the results in Table 9 below.

<table>
<thead>
<tr>
<th>Test user #:</th>
<th>Task #:</th>
<th>Comment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 &amp; 5</td>
<td>4</td>
<td>Made some mistakes drawing the pizzeria, but corrected them.</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Had problems drawing a door while sketching the pizzeria.</td>
</tr>
<tr>
<td>2 &amp; 3</td>
<td>3</td>
<td>Had problems placing the first symbol. Tried to drag and drop them.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Drew lines instead of walls at first, but corrected it afterward.</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>Mixed up “pest control self-check” with “pest control symbol”, they belong in different layers.</td>
</tr>
</tbody>
</table>

Table 9: Non-Critical Errors

**Critical Errors**

Critical errors are reported below in Table 10. It seems like lack of concentration or confusion while reading the tasks led to these errors. Users did not read the explanation texts on the tool bar buttons which could prevent the errors made during task 4.

<table>
<thead>
<tr>
<th>Test user #:</th>
<th>Task #:</th>
<th>Comment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>Drew a rectangle instead of walls while sketching the pizzeria.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>User chose the wrong size while creating the drawing.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Used undo symbol instead of undo drawing object, which led to the symbol in task 3 disappeared.</td>
</tr>
</tbody>
</table>

Table 10: Critical errors

**Successful task completion rate**

The average task completion rate during the five test sessions were 94%. Table 10 above displays that critical errors were made on task one and four. Even though every user thought he had successfully completed every task during the session.

**Hesitations**

Hesitations has been noted by the observer if a user hesitated more than four seconds. The results are displayed in Table 10 below.

<table>
<thead>
<tr>
<th>Test user #:</th>
<th>Task #:</th>
<th>Comment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 3 &amp; 5</td>
<td>9</td>
<td>Took some time before finding self-check layer.</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Took about 10 seconds before finding the layer check-box menu.</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Took time before finding the save button.</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>Took some time before the user found the “name box” and changed the name.</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Didn't find the button “create empty drawing” at first.</td>
</tr>
</tbody>
</table>

Table 11: Hesitations
5.7.2 Qualitative Test Data

By using the mean of the scores set by the users and the observer, the average score of likeability and learnability has been calculated to a score of 4 out of 5 (rounded down) which achieves the goal set in beforehand.

Likert scale questions in the questionnaire (1-5)
The test users has answered the questionnaire and scored the system after the test sessions. The mean scores are displayed below in Table 12.

<table>
<thead>
<tr>
<th>Questions:</th>
<th>Average score:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall, how did you feel about performing tasks in the system? (Likeability)</td>
<td>4,2</td>
</tr>
<tr>
<td>How was it to find a specific tool or symbol in the system? (Learnability)</td>
<td>4,6</td>
</tr>
<tr>
<td>How was it to place out a symbol on the drawing area? (Learnability)</td>
<td>4,0</td>
</tr>
<tr>
<td>How was it to use the sketch-up drawing tools, e.g. drawing a wall? (Learnability)</td>
<td>3,8</td>
</tr>
<tr>
<td>How easy was it to understand the system design? (Learnability)</td>
<td>4,2</td>
</tr>
</tbody>
</table>

Table 12: Average score of the Likert scale questions

The Observer's Score on the Test Users (Learnability measure)
The observer has scored the performance of the users after each session. The mean score was 4.

Free Text Questions
Was there one or more tasks that you could not complete, please state the number(s) of the task?
Every test user answered no.

What do you think should be changed in the system?
“Add a drag and drop function for placing out symbols.”
“There should be an eraser for deleting a marked object.” (This function actually exist.)

State two functions that you think are good and useful in the system.
“Selectable tool bars.”
“Intuitive”
“It's easy to import a drawing.”
“It's easy to place symbols onto a drawing.”
“Good with a lot of separated layers.”
“The design was easy to understand.”

Think-aloud Observations
The relevant think-aloud observations are displayed in Table 13 below.

<table>
<thead>
<tr>
<th>Test user #:</th>
<th>Task #:</th>
<th>Comment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 &amp; 3</td>
<td>3</td>
<td>“Can't place the symbol, drag and drop does not work.”</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>“Hard to see if you drew an object or not.”</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>“Where do you create an empty drawing?”</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>“Oh, so simple to place out symbols.”</td>
</tr>
</tbody>
</table>

Table 13: Think-aloud observations
Chapter 6

6. Usability Evaluation

6.1 Heuristic Evaluation

The drawing tool has been analyzed with the help of the ten common heuristics for user interface design. Even though a heuristic evaluation should be performed by a group of experts the author of this thesis has performed an inspection on his own work.

6.1.1 Visibility of system status

“The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.” (Nielsen, 1994)

Indicators

- There exist few distinct notifications. First of all while a drawing is being loaded in the editor, there is a spinning loading indication telling the user that the drawing is being loaded. The same indication shows up while using the print function when the system generates the print out.
- Another important feature is when you push on a symbol with the marking tool, a frame appears around the marked symbol which makes it easy to see which of the symbol you marked.
- If you press on a draw object, such as a wall, two red dots appears, one on the starting coordinates of the draw object and one in the end coordinates. With the marking tool you may move the draw object's starting coordinates and end coordinates.

Pop ups

- There is also few pop ups in the system, e.g. you try to place a symbol outside the drawing area. This will invoke a pop up telling the user he can't do that. There is also confirmation pop ups like “The drawing has been saved.” and optional pop ups “Exiting the editor, do you want to save your changes?”.

Menus

- Self-checks that are already placed out on a drawing has a green check marking in the tool bar, letting the user know it's already exist in the drawing.
- Check-boxes for active layers are pretty straight forward. The checked ones are active, the unchecked ones are not active. For each active layer a tool bar appears with symbols that belongs to that layer.
- A missing feature would be that there is no way to tell which of the tools or color that is active at a specific time. You simply have to remember which tool you just pressed. However, while placing out symbols on the drawing surface the specific symbol appears at the mouse pointer while hovering over the canvas making it quite easy for the user to place a symbol on the right position on the first try.
6.1.2 Match between system and the real world

“The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.” (Nielsen, 1994)

- The user interface is built up by simple menus with headlines. By hovering the buttons of different tools and symbols a description of the buttons appears explaining what the tool/symbol is.
- Most of the symbols has real-world names such as fire safety symbols, e.g. fire extinguisher or emergency exit.
- The different available layers for each drawing are named with the terms used at Inspectera, thus they are familiar to the intended users.
- Familiar terms used in common editors has been used for example zoom in, zoom out, marking tool, save and load.
- The image buttons may not have the same meaning for each and every user, e.g., an image of a floppy disk corresponds to the save function.

6.1.3 User control and freedom

“Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.” (Nielsen, 1994)

- There is no advanced undo function available. However, there is two simple undo functions that either removes the marked symbol or the last symbol/draw object that were added. No support for redo exists.
- There is no way to change the size of the canvas (drawing) area.
- If a pop up appears, there is always an “x” to push, to exist the dialog.
- You may at any time exit the editor by using the side bar menu to navigate to another activity.

6.1.4 Consistency and standards

“Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.” (Nielsen, 1994)

- The editor only support Swedish at this time. Symbols names are either used at Inspectera or commonly by the majority e.g. “emergency exit symbol”.
- The tool bars with tools are named as mentioned before according to common editor name conventions.
- As this is an online editor, the platform that is being used does not really affect the GUI.
6.1.5 Error prevention

“Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. “ (Nielsen, 1994)

- If the user tries to add a symbol or object outside the drawing area a pop up appears telling the user that it's impossible.
- There is a pop up when exiting the editor asking the user if he wants to save the drawing.
- If the user pushes “remove the last added symbol” and that symbol exists in a none active/visible layer, the user is prompted whether to remove the symbol or not.
- If the user pushes the clear button (removes every symbol and drawn objects), a pop up appears asking the user to confirm this action.
- A redo function does not exist. You might use the load function to go back to a previous state.

6.1.6 Recognition rather than recall

“Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. “ (Nielsen, 1994)

- There exists a help bar on the right side of the editor, it can be shown or minimized at any time. The help bar contains tips and information about how to use the different tools.
- The activation/deactivation of the layers are easily spotted due to the check-box menu.

6.1.7 Flexibility and efficiency of use

“Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. “ (Nielsen, 1994)

- The available flexibility right now is that you can choose which layer(s) to work on at a specific time, each layer spawns an additional tool bar. But you can not change or move the tool bars.
- It does not exist any particular function to set the GUI in a novice user state or in an expert state.

6.1.8 Aesthetic and minimalist design

“Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.” (Nielsen, 1994)

- The GUI is sufficiently minimalistic with only tool bars, buttons and the drawing area visible, except if you decide to hover over the buttons to see the explanation text.
- You can hide the help bar whenever you want.
- You can easily work with one layer at the time which only enable one extra tool bar. (One tool bar with symbols per active layer is visible.)
6.1.9  Help users recognize, diagnose, and recover from errors

“Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution. “ (Nielsen, 1994)

- Adding symbols outside the drawing area opens a pop up telling the user that it's impossible to place a symbol there.
- If the user accidentally exit the editor by clicking on another activity in the side menu he is asked if he wants to save the changes he made to the drawing. However, you can't push something like “cancel!” and stay inside the editor.
- Pushing save or load while not connected to the system due to failing net will open a pop up telling the user that there is no connection to the data base.

6.1.10  Help and documentation

“Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. “ (Nielsen, 1994)

- The help menu on the right side can be hidden or shown at any time. It consist of short texts for each tool. Tips and information about how to use them. The tool's name is written in bold style, which makes it easy to spot a specific tool you need more information about.
Chapter 7

7. Results

7.1 Functionality and Requirements

The GWT Canvas library has shown to be advanced and flexible enough in order support the functionality Inspectera demanded. The drawing tool runs with satisfying performance as long as you have an updated browser and sufficient hardware that has no problem handling images. During the development process the requirements and features of the drawing tool has been discussed, changed, and added during the sprint meetings. The final requirements, which has all been achieved, has been summarized in section 2.6. This kind of “transparent development” with almost instant feedback, with a testing server up at all time, and flexibility regarding the system requirements has shown to be a successful method in this project. A lot of functionality has been developed and tested during this short time project and the CEO at Inspectera seems sincerely satisfied with the results.

7.2 Heuristic Evaluation

The evaluation has been conducted by the developer himself and that might have affected the results. Through the heuristic evaluation, more positive than negative feedback has been derived. No serious usability problems has been found that would make the drawing tool unusable. There are of course few things that would improve the usability of the system, such as:

- indications of which tool and color that is currently active
- a redo function should be added
- a feature for changing the canvas size (the user may have low desktop resolution)
- improvement of the current undo function(s)

To sum it up, the drawing tool includes a lot of features that improve the usability of the system but, lack few important ones as well.

7.3 Usability Testing

The average user test results made the requirements that has been set for efficiency regarding time on task, while measuring time per task and number of clicks per task, except in task 4. The goal of the qualitative measures such as likeability and learnability were also achieved. There was only one question regarding the sketch tools that got a score below four (3,8) on the Likert scale. Most of the errors recorded during the test sessions were actually made during task number four, when the user was supposed to sketch a pizzeria with the sketching tools. The goal about 100% task completion rate was not achieved, although it was not that far away (94%) which proves that users are able to perform the intended task in the system. Couple of the test users tried to drag and drop the symbols onto the canvas, this intuitive function should be added if the development of this project continuous.
Chapter 8

8. Conclusions

This chapter presents the conclusion of this master thesis. Suggestions for further improvement of the developed drawing tool are also listed here.

8.1 Accomplishments

The goals presented in the purpose section (1.2) has been achieved.

- The GWT Canvas library was a good choice for handling the drawing area. It supports every feature that Inspectera asked for in the drawing tool.
- The final requirements that are listed in section 2.6 has been fulfilled, hence Inspectera did get what they asked for.
- The drawing tool is supported by every common web browser (IE, Firefox, Google Chrome, Safari, Opera) that is up to date and supports HTML5.
- Usability testing of the drawing tool has proven that it's both efficient and effective. Users are able to accomplish the intended tasks in the system within reasonable time. Test users has scored the selected quality criteria (likeability and learnability) for this drawing tool, and the average score results are 4 out of 5 on the Likert scale.
- The results of the usability evaluation (heuristic evaluation) shows that this new drawing tool includes many usability features that Nielsen’s heuristics promote but, lacks few important ones as well.

This was a short time project that resulted in a new tool with a lot of functionality and of course there is still things that needs improvement and further testing. Suggestions of how to improve the usability of this drawing tool are listed below.

Suggestions of smaller improvements:
- A drag and drop function for placing symbols onto the canvas.
- Add indication of which tool that is currently active.

Suggestions of bigger and more time consuming improvements:
- A feature that lets an administrator add new symbols to the available layers, without the need for recompilation and a restart of the service. One solution might be a dynamic UI that would build by fetching meta-data stored in the relational database.
- Development of a redo function and improvements of the undo function for handling drawing events. Perhaps with the use of the command design pattern.
Chapter 9

9. Discussion

This project has been both challenging and instructive. It was rewarding to develop a graphical tool such as this, to see what your system does graphically instead of regular programming where your functions are just being run in the background of the system.

I think that the heuristic inspection of my own development may have affected the results positively. Although, I have been in a few usability classes at Linköping University which helped me to follow usability guidelines during the development of this tool.

During the usability testing sessions it was fairly difficult to write down everything that had to be recorded in the observation protocol in time. I also felt that some of the test users felt stressed when I measured the time per task. Users might have lost focus and rushed through the testing tasks which led to errors. It's understandable that a task like task 4 (sketch a pizzeria) may require some correcting to get the walls straight or to make corrections due to any mistake made while using the sketch tools. The results show that the average user failed the efficiency criteria but, I don't see it as a usability problem in this task. If the user can correct his/her own mistakes you might consider it an achievement.

Inspectera hires IT consults for maintaining and develop their IT services. During my project the company changed IT consults and the new consults needed some time to get familiar with the system. Although, I have been able to ask the consults for support in some situations, e. g. we discussed the saving and management of drawings. The original developers (Inspectera's previous IT consults) has left out one important part in the development process, there was almost no documentation of this large scale product.
Chapter 10

9. References

Sarkissian A. (2008), Definition of Prototype Development <HTML>


Miami University (2004), Usability Testing: Developing Useful and Usable Products <E-BOOK / HTML>
Available <http://www.units.muohio.edu/mtsc/usabilitytestingrevisedFINAL.pdf> (2013-10-01)
Miami University of Ohio

Usability.gov (2013), Reporting Usability Test Results <HTML>

Usability.gov (2013), Planning Usability Testing <HTML>

John Wiley & Sons, Inc. New York

GWT Project (2013), Server Communication, Remote Procedure Calls <HTML>

Ramsdale C. (2010), MVP Part 1 <HTML>

Nielsen J. (2012), How Many Test Users in a Usability Study? <HTML>

Zeldman J. (2001) Taking Your Talent to the Web: Making the Transition from Graphic Design to Web Design
New Riders Publishing

Available <http://www.uie.com/articles/three_click_rule/> 2013-12-03
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Alf Bjelkenstedt