Usability evaluations of vector graphic editors

Elin Svedin
Abstract

Usability evaluations of vector graphic editors

Elin Svedin

In this thesis are vector graphic editors evaluated with help of different usability studies. The usability studies include heuristic evaluations, cognitive walkthroughs and usability evaluations. These studies were done in order to find appropriate usability requirements for web based graphical editors and to implement a new editor for the company KSU. The user groups in all studies are therefore co-workers at KSU and most studies focuses around their old graphical editor DVdraw.

The findings during the evaluations are divided into two lists, one with specific requirements for KSU and one with general requirements for all graphical editors. The thesis also present a design solution based on the requirements, design guidelines and design principles found appropriate for a graphical editor.
Contents

1 Introduction 3
  1.1 Problem description 3
  1.2 Research questions 3
  1.3 Delimitations 4

2 Background 5
  2.1 Kärnkraftsäkerhet och utbildning 5
  2.2 Technology 5
    2.2.1 Dynamics and variable names 6
    2.2.2 New project 7

3 Theoretical framework 8
  3.1 Interactive system design 8
    3.1.1 Human-computer interaction 8
    3.1.2 Usability 8
    3.1.3 User experience 9
  3.2 Computer graphics 9
  3.3 Web applications 9
  3.4 Design principles 10
    3.4.1 Nielsen’s 10 usability heuristics 10
  3.5 Design guidelines 12
    3.5.1 Feedback 12
    3.5.2 Consistency 13
    3.5.3 Memory 13
    3.5.4 User differences 14
  3.6 Design rules 14
    3.6.1 Colour 14
    3.6.2 Icons 15

4 Methodology 17
  4.1 Walkthrough 17
  4.2 Heuristic evaluation of DVdraw 17
  4.3 Designing test plan for usability studies 18
  4.4 Usability Evaluation 19
  4.5 Compilation of findings 20
  4.6 Cognitive walkthrough 20
  4.7 Findings from Cognitive walkthrough 20
5 Results
  5.1 Heuristic evaluation ............................................. 21
  5.2 Usability Evaluation .............................................. 23
  5.3 Cognitive walkthrough ........................................... 28
    5.3.1 Method draw .................................................. 28
    5.3.2 Janvas .......................................................... 31

6 Requirements ......................................................... 33
  6.1 Requirements vector graphics editor ............................. 34
  6.2 Specific requirements ............................................. 38

7 Design solutions ..................................................... 40
  7.1 Overall design .................................................... 40
  7.2 Dialog field ....................................................... 41
  7.3 Left side menu .................................................... 41
  7.4 Right side menu ................................................... 47

8 Summary ................................................................. 53
  8.1 Discussion ......................................................... 53
  8.2 Conclusion ......................................................... 53
  8.3 Future work ......................................................... 54

Appendices ........................................................................ 56
  A Scenarios Cognitive walkthrough .................................. 57
1 Introduction

The way humans interact with computers have changed significantly through the last decades. In the early days, computers were expensive, huge and hard to use. As a consequence were computers only used by specialist and scientist. This changed with the launch of the personal computer (PC) in the 1970s [1]. The PC made it possible for anyone to buy a computer with a graphical user interface which they could use and store in their own home. Computers are a big aspect of our daily life and we interact with computers almost daily. Consequently, the user demands on computers and software increased and badly designed programs could be devastating for a company. Human computer interaction (HCI) have therefore become an important study for companies to embrace. HCI is a field that studies how humans interact with computers and how to develop technologies that help humans interact with computers.

1.1 Problem description

The company Kärnkraftsäkerhet och utbildning (KSU) works with nuclear safety and education. In order to train new personnel KSU are using simulators that simulate the nuclear reactors. These simulations can be displayed with help of vector graphics (section 3.2). The program used to create new vector graphic images is out of date, complicated and have limitations that makes it hard to use. At the moment only specialists create images in the vector graphics editor. However, limitations in the program makes it tedious to use even for them. Therefore, KSU decided to do an evaluation of the old editor in order to decide the user requirements of a new graphical editor. Their goal is to create a more user-friendly editor that reduce the learning curve and enhance the user experience for both experienced and novice users.

1.2 Research questions

- What are the user requirements for a web based vector graphics editor?
- How can the requirements be used when designing the new editor?
1.3 Delimitations

The user evaluations in this study will be small and evaluate at most 4-5 participants. The evaluation will primarily focus on the drawing tools and navigation bars. The usability evaluations in this thesis should be done several times in order to have an accurate requirement list and design solution. Due to time limitations will the usability evaluations in this thesis only be conducted once and contain the first draft of requirements lists and design solutions.
2 | Background

The background gives an introduction of the company KSU and the technology they use today.

2.1 Kärnkraftsäkerhet och utbildning

*Kärnkraftsäkerhet och utbildning* (KSU) handles nuclear safety, education of personnel and contribute to safer nuclear power operations in Sweden. Their most important task is to teach operating- and maintenance personnel at the Swedish power plants. KSU have used simulators and simulation technology for training of personnel and validations of plant modifications since the 1970s [6]. The training is done in full scale replicas of the power plant control room which gives detailed simulations of the plants processes. [7].

2.2 Technology

In KSU:s development systems and graphical simulators, computer graphics is used in order to visualize the control room and simulation state. For visualisation the DataViews framework is used [Web page], it is a powerful but old framework available on multiple platforms. To create new graphics an editor called DVdraw (Figure 1) that run on a silicon graphics machine is used. DVdraw is an interactive software tool that can create two-dimensional drawings with dynamic components. To allow highly scalable graphics that looks good when zoomed in; vector graphic is used.

![Figure 1: Overview of DVdraw](image-url)
2.2.1 Dynamics and variable names

To animate and allow interactivity, objects in the graphics can have dynamics and variable names. These variable names corresponds to the names in the simulation software. It is important that all variable names are correct so the right value from the simulator corresponds to the correct object. An object with a variable name can also have dynamics added to it. The dynamics are rules for how the object should change depending on the values given from the simulator. The different dynamic classes can be found in table 1. All dynamic rules have thresholds that tells the object when it should change. For example, a lamp could have threshold \([0, 0.5),[0.5, 1]\), where the first range tells the lamp to be dark green, and in the second range light green.

<table>
<thead>
<tr>
<th>Object dynamics:</th>
<th>Explanation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>The object switches color.</td>
</tr>
<tr>
<td>Text</td>
<td>Display the value given from the simulator.</td>
</tr>
<tr>
<td>Change sub-drawing</td>
<td>Object switches between different sub drawings.</td>
</tr>
<tr>
<td></td>
<td>For example, a button that is pressed in and then pressed out.</td>
</tr>
<tr>
<td>Rotate</td>
<td>The object rotates.</td>
</tr>
<tr>
<td>Movement</td>
<td>The object moves horizontally or vertically.</td>
</tr>
<tr>
<td>Visibility</td>
<td>The object is visible or invisible.</td>
</tr>
</tbody>
</table>
| Fill                      | The object is filled with a specified color, horizon-
|                           | tally or vertically.                                 |

Table 1: The different dynamic classes used by KSU

An object can also be assigned an input command. These commands tells the object how it should behave when a user interact with it. The different input commands can be found in table 2. The input command could for example be an image link.

<table>
<thead>
<tr>
<th>Input command:</th>
<th>Explanation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle Button</td>
<td>Toggles between on and off.</td>
</tr>
<tr>
<td>Spring Button</td>
<td>Stays on as long as user is pressing the button.</td>
</tr>
<tr>
<td>Push Button</td>
<td>Is on for a second when pressed then turns back to off.</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Switch</td>
<td>Switches between several positions. All positions are connected to a variable which all have an on/off value.</td>
</tr>
<tr>
<td>Switch Menu</td>
<td>Open up a menu with options that the switch can turn to.</td>
</tr>
<tr>
<td>@drawing</td>
<td>Link to another image</td>
</tr>
</tbody>
</table>

Table 2: The different input commands used by KSU

### 2.2.2 New project

The DataViews framework is old, inflexible and expensive; therefore KSU is looking at different options to replace DataViews. KSU evaluated available upgrade options. However, none of the commercial solutions evaluated were compelling. Therefore, KSU decided to begin a new project with purpose to evaluate the use of modern web technology and open standards. The project led to a prototype that displays SVG-images in a web browser, the images are animated with help of javascript and JSON-files. The use of web technology removes troubles with updates, installation and licenses (section 3.3).

When the project got positive feedback KSU decided to continue with the prototype. The next step is to add a web based editor. The current editor (DVdraw [1]) can be used to edit graphics for the new system. However, images drawn in DVdraw have to be converted which complicates the editing process. DVdraw takes a long time for new users to learn, and only specialists edit graphics. Instructors have expressed an desire to create graphics for their classes. Therefore, a more user friendly editor would be of benefit for KSU.
3 | Theoretical framework

This section explains concepts that appear in the thesis. The theoretical framework contains chosen design guidelines, design principles and design rules for the graphical editor. These design recommendations and rules will be used during the usability evaluations, and during the design process.

3.1 Interactive system design

Interactive systems are components, devices and software systems that processes information and respond to users interactions [1]. All technology depending on user input to work are called interactive systems. Interactive system design is about designing systems that supports people, are enjoyable and creates an interactive experience for the users. When designing interactive systems the designer needs to keep in mind what the users can do rather than thinking about what technology can do [1].

3.1.1 Human-computer interaction

Human-computer interaction (HCI) is the study on how humans uses technology and how to design technology to be more user friendly [11]. HCI focuses on how to design a system that is easy to use, easy to learn and usable for the intended user group [1]. It is important to find troublesome features that are tedious for the users in order to find a solution. The features should also match real world convention to ease the use and reduce learning time for the user. Since HCI focuses on the users, systems created with this approach tend to benefit from better user satisfaction, less user difficulties and lower development costs [15].

3.1.2 Usability

In (ISO 9241-11) usability is defined as "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". An effective system have appropriate information and functions and an efficient system allows user to do things with the appropriate amount of effort [1]. In order for a system to have high usability the users goals have to be bridged together with the physical system.
3.1.3 User experience

User experience (UX) includes all aspects of usability and desirability of a system from the user’s perspective (ISO 9241-210)\cite{13}. To have a good UX the system must be simple, elegant and meet the user’s needs without any complications. Usability is a big part of UX as well as usefulness, desirability, credibility and accessibility \cite{13}. The user interface is an important part of UX. However, total UX are concerned with the experience of the whole system and not only the user interface. \cite{12}

3.2 Computer graphics

Computer graphics has been around almost as long as computers themselves. In the 1960s until the 1980s the most common graphic displays were built on vector graphics \cite{4}. Vector graphic images are created with mathematical equations called vectors. A vector is a line drawn between two coordinate points and these points decide the size and direction of the vector. These vectors are then used to draw lines, points, curves, polygons and other geometrical primitives \cite{14}. Vector graphic images does not have an fixed resolution which makes it possible to scale them to any size without loosing resolution.

3.3 Web applications

Computer application runs in the OS (operation system), these applications have to be downloaded and installed on a specific computer in order to work. Consequently, this process have to be done all over again to use the program on another computer. Web applications runs in the web browser and are stored on a server instead of the local computer. This means that the application can be accessed from any kind of device with a web browser and a Internet connection. This saves memory on the computer, retrieving information gets more flexible and in most cases the user can use a more lightweight device such as a tablet or a mobile phone. Web applications are programmed for web browsers and not a specific OS. Consequently, the developer is able use the same code when developing for different web browsers. Web applications are easier to update than computer applications. Small fixes can be made and launched directly without having to wait for the next version to be released.
3.4 Design principles

The design principles are clear rules of the thumb that provide advice when issues and problems occur. They are usually high-level and context-free design goals based on research about how people learn and work (theories of HCI). The principles can be used during the development process as recommendations, they can also be used when evaluating prototypes and existing systems [1]. The following sections state usability design principles for interactive systems.

3.4.1 Nielsen’s 10 usability heuristics

Nielsen’s 10 usability heuristics are usability design principles developed by Jakob Nielsen in 1994. These 10 heuristics are based on Nielsen’s earlier work with Rolf Molich in 1990.

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

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.

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.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

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.

**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.

**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.

**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.

**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.

**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.
3.5 Design guidelines

Design guidelines are more specific and low-level compared to the design principles. They are rules the designer can follow in order to achieve the goals stated in the design principles. They often give a quick answer on how to design different features in the system. However, design guidelines are easily misinterpreted and misapplied. It is therefore important to keep the number of design guidelines as low as possible, as well as having a clear description for each design guideline. The following sections state design guidelines that will help the designer achieve the goals given in section 3.4.

3.5.1 Feedback

All user interactions should give feedback to the user. No feedback means that nothing has happened.

Noticeability and visibility

The feedback should be visible for the user when it is needed. Relevant feedback should come to the users attention without users needing to search for it. Relevance of feedback should be indicated with color and placement of feedback. The user should be able to control how much feedback the application should provide.

Progress indication

The application should have progress indications for long tasks explaining when a task is completed or what is left to do to complete a task.

Error prevention

The feedback should be used for error prevention. The application should always tell the user what is going on and give the user an cancel or undo option.

Readability

The feedback should be readable and use a language that the user understands. It should be kept short and informative.
3.5.2 Consistency

The application should be consistent internally and externally to other vector graphics editors. Inconsistencies should only be applied to distinguish differences in content or user actions.

Avoid ambiguities

Ambiguities in a interface leads to errors. In order to avoid this it is important to have a clear distinctions between similar features. For example, "mark" a feature that selects several objects and "group" the feature selects several objects and then grouped together.

External consistency

The user’s interaction should have the same appearance, meaning and operation across different applications. For example, shortcut keys like cut, paste, delete and undo should follow standards from other applications. The interface elements should be applied as they were designed. For example, a radio button should only be used when one choice of selection is available. The layout of the application should match similar applications, in graphical editors navigation, tool options, and styling of objects often have the same positions.

Internal consistency

Keep colours, typography and language consistent throughout the whole application. Icons, buttons and links must be consistent throughout the whole system. The interactions dialogue boxes, animations and transitions should be consistent.

3.5.3 Memory

The user should not have to memorize a lot of information. Relevant information should be present for the user at all time, other information should be easy to retrieve.

Chunking

Avoid big menus by grouping meaningful information into larger units. Menus and navigation bars should not contain more than seven items.
Time limitations

Make important information presented persist. Do not flash an alert onto the screen, instead force the user to interact with the system and acknowledge the message [1].

Use selection

Selection should be used in pick lists. For example, users should not have to remember all fonts available and how they are spelled. Instead they should be able to search for the font and select the correct one.

3.5.4 User differences

The application should be able to provide different amount of information and tips depending on the expertise of the user. The application should be able to lead novice users, keep intermediate users on track and get out of the way for expert users [5].

Shortcuts

Shortcuts should be used for physical affordance. This means that "hot keys" can be used by the user instead of menus, icons and button command choices. For example, 'Ctrl' + 'S' keys can be pressed for saving a file. The "hot keys" should be visible for the user. For example, should the 'Ctrl' + 'S' be visible to the user as a tool tip.

3.6 Design rules

3.6.1 Colour

The following color rules should be followed when designing the graphical editor. In Aaron Marcus’s book "graphic design for electronic documents and user interfaces" [9] the following rules are stated for designing with colour.

Use a maximum of five, plus or minus two, colors

For novice users, four distinct colors seem appropriate. This number allows extra room in short-term memory, which can store five words or shapes, six letters, seven colors, and eight digits.
Use foveal (center) and peripheral colors appropriately

Use red and green in the center of the visual field, not in the periphery. The edges of the retina are not particularly sensitive to these colors. If red and green are used at the periphery, some signal to the viewer must be given to capture attention, for example, size change, blinking, etc. Use adjacent colors that differ by use and value (lightness).

Use colors that exhibit a minimum shift in color/size if the colors change in size in the imagery

Use light text, thin lines, and small shapes (white, yellow, or red) on medium-dark and dark backgrounds (blue, green, red, or dark gray) for dark viewing situations. Reserve the highest contrast in figure-field relationships for text typography. Use dark text, thin lines, and small shapes (blue or black) on light backgrounds (light yellow, magenta, green, blue, or white) for light viewing situations.

Do not use high-chroma, spectrally extreme colors simultaneously

Some figure-field relationships, like strong contrasts of red/green, blue/yellow, green/blue and red/blue, create vibrations, illusions of shadows, and after-images. Avoid to use these simultaneously.

Use familiar, consistent color codings with appropriate references

In table 3 are different colours and their reference stated. The references are taken from the western colour convention.

<table>
<thead>
<tr>
<th>Color</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Stop, danger, hot, or fire</td>
</tr>
<tr>
<td>Yellow</td>
<td>Caution, slow, or test</td>
</tr>
<tr>
<td>Green</td>
<td>Go, ok, or clear</td>
</tr>
<tr>
<td>Blue</td>
<td>cold, health/finance, or water</td>
</tr>
<tr>
<td>Warm colors</td>
<td>Action, response required, or spatial closeness</td>
</tr>
<tr>
<td>Cool colors</td>
<td>Status, background or information</td>
</tr>
<tr>
<td>Grays, white</td>
<td>Neutrality</td>
</tr>
</tbody>
</table>

Table 3: Western colour conventions
3.6.2 Icons

The icons in the vector graphics editor should follow standard from existing graphical editors. The icons should also be checked with Horton’s icon checklist in table 4. The icons should be drawn in a vector graphics format so they are scalable (section 3.2).

<table>
<thead>
<tr>
<th>Is the icon...</th>
<th>Example question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understandable</td>
<td>Does the image spontaneously suggest the intended concept to the viewer?</td>
</tr>
<tr>
<td>Familiar</td>
<td>Are the objects in the icon ones familiar to the user?</td>
</tr>
<tr>
<td>Unambiguous</td>
<td>Are additional cues (label, other icon documentation) available to resolve any ambiguity?</td>
</tr>
<tr>
<td>Memorable</td>
<td>Where possible, does the icon feature concrete objects in action? Are actions shown as operations on concrete objects?</td>
</tr>
<tr>
<td>Informative</td>
<td>Why is the concept important?</td>
</tr>
<tr>
<td>Few</td>
<td>Is the number of arbitrary symbols less than 20?</td>
</tr>
<tr>
<td>Distinct</td>
<td>Is every icon distinct from all others?</td>
</tr>
<tr>
<td>Attractive</td>
<td>Does the image use smooth edges and lines?</td>
</tr>
<tr>
<td>Legible</td>
<td>Have you tested all combinations of colour and size in which the icon will be displayed?</td>
</tr>
<tr>
<td>Compact</td>
<td>Is every object, every line, every pixel in the icon necessary?</td>
</tr>
<tr>
<td>Coherent</td>
<td>Is it clear where one icon ends and another begins?</td>
</tr>
<tr>
<td>Extensible</td>
<td>Can i draw the image smaller? Will people still recognize it?</td>
</tr>
</tbody>
</table>

Table 4: Top-level headings of Horton’s icon checklist with example questions. (Derived from ’Designing interactive systems’ [1]).
4 Methodology

The methodology explains the research approach taken in order to collect data about different vector graphic editors. It explains the used methods, why they were chosen and how findings were represented and analysed. The Evaluation used three different usability studies, heuristic evaluation, usability evaluation and cognitive walkthrough. The methodology is built up in a chronological order, from beginning to end.

4.1 Walkthrough

When developing an application there is always a specific user group that you develop for. Therefore, is it crucial to understand the users, how they solve tasks and in which environment. The reason for the walkthrough was to gain a better understanding on how DVdraw is used at KSU and how they work on a day-to-day basis. The walkthrough was done with help of a co-worker with experience of DVdraw. The co-worker showed the editors different features and how they work with DVdraw at KSU. Existing images, done with help of DVdraw were also shown and explained. A discussion about what they expected of a new editor and the biggest issues with the current system was done with help of two other co-workers. The notes taken during the walkthrough were used throughout the whole development process, as a replacement of personas and scenarios.

4.2 Heuristic evaluation of DVdraw

After the walkthrough an heuristic evaluation was conducted in order to find problematic areas to focus the usability studies around. This approach was taken because of the complexity of DVdraw, it would have taken days for a novice user to evaluate the whole system. The evaluation was focused on finding and explaining problems not solving them. The findings of the evaluation does not have any severity rating and they are seen as equally serious. Nielsen’s heuristics (3.4.1) were used in the evaluation.
4.3 Designing test plan for usability studies

A test plan should state: the purpose of the evaluation, practical details, information about the participants and the tasks chosen to be performed [2]. The findings of the heuristic evaluation and the walkthrough worked as a foundation when deciding what tasks the participants should perform during the usability evaluation.

Purpose

KSU has decided to look into the possibility to make a new web based graphical editor to draw images of their control panels. The purpose of this evaluation is to find the unnecessary features, the complicated features and the most used features in the current editor (1).

Participants

The participants of this study are co-workers at KSU. The goal is to have two different user groups, expert and novice users. The reason for having two different user groups is to get out as much information about the editor as possible. The novice users will give information on how hard the system is to learn and point out tasks that are not straight forward. The experienced users, which are the ones that actually uses the editor on a day to day basis will give information on what features that are good and bad in the existing editor. There are at least two participants in each user group.

Figure 2: Image that participants should draw during the usability studies.
Tasks

The participants in the usability study will be asked to perform the tasks stated in table 5. The user have to complete all tasks. However, help can be given to the participants if they get stuck during the session.

<table>
<thead>
<tr>
<th>Tasks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open a panel named test.v</td>
</tr>
<tr>
<td>Zoom and Pan so the image fills the whole drawing area.</td>
</tr>
<tr>
<td>Create a new image and make the image look the same as Figure 2.</td>
</tr>
<tr>
<td>Give the three buttons different names.</td>
</tr>
<tr>
<td>Reorder the Buttons so they are behind the turquoise rectangle.</td>
</tr>
<tr>
<td>Make 'test'-text have a red font-color and move it to the black rectangle.</td>
</tr>
<tr>
<td>Select the three buttons again and move them in front of the turquoise rectangle.</td>
</tr>
<tr>
<td>Save image and give it a new name</td>
</tr>
</tbody>
</table>

Table 5: Tasks during the usability evaluation.

4.4 Usability Evaluation

The usability study began with a short introduction which explained the purpose of the study. The participants where asked if it was okay to use an audio recorder during the session, and the importance of thinking aloud was explained. After the introduction participants were given the task list, novice users were introduced to the system in order for them to understand the provided tasks. When the tasks were finished an after session interview was held. The intention of this interview was to allow the participant to speak more freely about the system, and their overall user experience. At the end of the interview the participant was asked to look at Nielsen’s heuristics and go through the system one last time. This was done after the participants had gotten familiar with the system in order to not confuse them in the beginning and change their first impression of the system. The usability studies were conducted on 5 participants in total, two experts, two novice and one user that had some experience with the system. The participants in the different user groups pointed out different critical problems with the
system. However, the number of participants in each user group felt enough since each group pointed out similar problems. It did not feel necessary to add another user group since the system is going to be used mainly by these two specific groups [1].

4.5 Compilation of findings

The findings from the usability study and the heuristic evaluation are ordered by different areas of the system in a report. The report describes each problematic feature without any severity rating. The findings from the usability study and the heuristic evaluation were then combined in a requirement list for a graphical editor at KSU. The requirement list was divided into two parts: one for functionality and one for usability. The functional requirements describe what kind of features should be implemented and the usability requirements explain how they should look and behave. Solutions given by the user during the evaluation were also evaluated and added to the usability requirements. When user groups had contradicting opinions on the same feature, data from the walkthrough were taken into account.

4.6 Cognitive walkthrough

When all findings had been evaluated and added to the requirements list, cognitive walkthroughs were conducted on other existing graphical editors. This was done in order to find more requirements and to probate the existing ones. A cognitive walkthrough needs to have two different kind of scenarios: common and uncommon with critical sequences of activities[1]. These different scenarios can be found in appendix A. During each scenario and interaction with the system, the evaluator asked these questions: Will the users try to achieve the right thing? Will the users notice that the correct action is available? Will the users associate the correct action with their intention? Will the user understand that they have made progress?[1].

4.7 Findings from Cognitive walkthrough

Findings from the cognitive walkthrough were represented in a report ordered by each feature evaluated in the graphical editors. The final findings were combined into a requirement list. These requirements were later used as guidelines when designing the graphical editor. Some of the findings were also added into the requirement list for KSU:s graphical editor.
5 | Results

This section states the findings from the different usability evaluations explained in section 4. These findings were later used as basis for the graphical editors requirements and design.

5.1 Heuristic evaluation

The following problems were found when evaluating DVdraw with help of Nielsen’s 10 usability heuristics. All findings were seen as probable issues for both novice and experienced users. Therefore, all findings have the same severity rating.

Unclear and limited instructions

Instructions to the user are given at the top of the application. These instructions have a yellow color and are placed in a big dialog field. There are a limited number of instructions given to the user, and it is hard to follow what actually happens in the system. The instructions are sometimes unclear and hard to understand.

Few error-messages

The system does not give proper feedback when a problem occurs. There are a few error-messages implemented in the system that appear in the same dialog field as the instructions. The error-messages have the same color as the instructions, which makes them hard to separate. Consequently, is it hard for the user to recognize when they have made progress.

No implemented help-section

There is no help-section implemented in the application. There are some menu options that indicate that a help-section could be placed there. However, these menu selections only give general information on the system and contact information to the company that created DVdraw.

No automatic refresh

The system does not refresh the screen properly when editing, it is up to the user to selected the 'refresh' option which will update the drawing. Con-
sequently, objects can disappear from the users sight and novice user (not familiar with the ‘refresh’ option) might believe that the object is deleted.

**No undo/redo**

There is no undo/redo feature implemented in DVdraw. This could be a problem when working on bigger projects, if something goes wrong the user can be forced to redo the whole project. There is however an ‘undelete’ function which allows the user to recover the last deleted object.

**Marked menu options**

To draw a geometrical primitive, a menu option must be selected. The menu option changes appearance when selected in order to indicate progress to the user. However, after creating a geometric primitive the appearance is the same, indicating that the user can keep drawing, which they can not. This could be confusing for the user since the wrong feedback is given from the system.

**Big menus**

The menus in the system are big, which reduces the size of the drawing area. The most menu selections are visible for the user at all time. However, not all menu selections work at all time. For example, the options for text styling are visible when drawing or editing another geometric primitive. These text styles can be chosen without having any functionality. The big menus makes the system feel complicated and hard to navigate. Consequently, this could lead to longer learning time for novice users.

**Confusing menu selections**

A user should not have to wonder what will happen when selecting something in the menu. Therefore, is it important that all menu selections are descriptive and clear. Some menu selections in DVdraw have confusing names. Some names seem to mean the same thing while some are hard to understand at all.
5.2 Usability Evaluation

The following problems and solutions were found by the study participants during the usability evaluation.

**Hard to write and edit text**

The novice users could not decide whether they should chose the ‘vectext’ or ‘text’ menu option when writing text. They did not understand the difference between them and tried to find some explanation elsewhere in the system. One of the experienced user explained that KSU only uses ‘vectext’ since it is scalable. The ‘escape’ key has to be pressed in order to save a text to the drawing area. This was not clear for the novice users, they tried to press outside the text-box instead. Consequently, all novice users felt confused and thought that they had chosen the wrong menu selection. Experienced users also felt that this was a bad implementation and suggested that a text-box could be deleted by pressing the delete option or leaving the text box empty. It took some time for novice users to understand how to style text. They looked for styling options in the left side menu and explained that it felt natural that styling of text should appear at the same place as styling of geometrical primitives. One user thought that the styling options in the top menu were styling for the whole view and not for text. Experienced users would have liked an option for changing font-size instead of scaling text.

**Panning and Zooming is annoying**

One of the experienced users thought that zooming and panning were the biggest problem with the whole system. Zooming and panning an image leads to a lot of clicking, since they have to be reselected every time the user presses the drawing area. This slows users down and gets annoying after a while. Novice users understood zooming and panning, they also thought it was easy to find them in the system. However, the feature was hard for them to use. They would have preferred an option that brings the user back to the original view. The novice users suggested that the keyboard arrows could be used for panning, whilst the experienced users suggested an scroll-bar. They also suggested that the scroll-wheel on the mouse could be used for zooming. One experienced user suggested that zooming should be centered to where the mouse pointer is placed. Experienced users like the 'zoom to' option which allows user to zoom to a specific place by marking it.
Group objects does not work as intended

Group 'All', 'All in' and 'Part in' are different grouping selections that all users liked. All users liked that they could use the mouse pointer in order to group several objects. However, the group feature was not functioning as the novice users intended. They felt that this feature marked several objects instead of grouping them. The reason for this was that the grouping disappeared when drawing something new or when the user pressed outside the group, this is actually marking since grouping several objects indicates that they are combined into one object until the user decides to ungroup them. They suggested that both features should be implemented, hence one group feature that makes several objects one whole object and one marking feature that have the same functionality as the grouping feature in DVdraw.

Hard to select background, fill and edge color

All novice users found changing colors to be a big issue. This feature had to be explained to them in order to make progress. Once it was explained it was manageable, however they did still not understand when colors where changed. When they wanted to change the background color on the drawing area they navigated to the configuration tab, this felt like the natural place for it to be. Different coloring options can be chosen in the menu right next to the color palette. This menu selection can be toggled between several coloring choices ('Edge', 'Fill with edge', 'Edge with fill', 'Fill' and 'Transparent'). Experienced users explained that this implementation can be annoying since a user has to press through all options once more if they misses the one they wanted. The users liked the names 'Fill' and 'Edge' and thought them to be more explanatory than 'Background' and 'Foreground' color that is often used in graphical editors. It is not clear for the novice users if the background is going to be visible in the resulting image, and it is not clear how big the resulting image is going to be. They would have liked an drawing area with a height and width attribute as well as background color. The user should then be able to change these attributes.

Snap and Grid off by default

'Snap' and 'Grid' are used almost every time a user draws an image in DVdraw at KSU. The users explain that images drawn without these two features often have to be redrawn since they do not follow scale and objects are placed wrong and lines are very hard to draw straight. 'Snap' and 'Grid' are turned off by default. Consequently, the users have to do two extra steps before they can start using the system. Another problem is that novice users do
not understand the importance of using these features or forget to use them, which leads to a great deal of redrawing at the beginning. The experienced users suggests that these two features are turned on at start up. They want to keep the options for grid color and grid scale. Novice users did not understand why 'Snap' and 'Grid' where placed under the zoom menu. They stated that placing it under the view menu felt more logical.

Confusing names

Novice users had a hard time navigating correctly through the system. This mostly depended on the menus names, that where confusing for the user. Some users expressed a feeling that some of the menu selections seemed to mean the same thing. For example, icon, drawing and image or vectext and text. Some menu selections had abbreviations instead of full names. For example, 'cmds' and 'cp delta'. These abbreviations were hard to understand for the novice user. Some names did not explain the feature they represented and some of the drop down menus had content that the novice users felt should be placed under other menus.

Big menus

The novice users felt that the menus were too big, this made them overwhelming. One user suggested that icons could be used instead of names to save space. The experienced users did not have any problems with the big menus. However, they did point out some menu selections that were unnecessary or could be combined.

Grouping of menu selections

All users felt that some menu selections should be placed together in order to make the system easier to use. For example all styling of objects should appear on the same place and nearby the color choice. They also suggested that styling options should only appear when they could be used. One user suggested that all drawing tools should lie next to each other and tools for grouping, zooming and panning should be placed nearby each other. In general they all wanted a clearer demarcation between different groups of tools.

Placement of dialog field

The users did not give the dialog field much attention. Novice users found it when they had difficulties solving a task. One user suggested that the
dialog field should be placed at the bottom of the system, since that felt more natural. Experienced users did not take any notice of the dialog field at all which they thought was a good thing since it otherwise might draw attention from the drawing area.

The system is not responsive

One novice user tried to decrease the window size. When doing so the whole system got a weird layout and the names on the menu selections disappeared after a while. The user stated that if you want to use the system on a smaller screen or just using half of a screen and this not possible.

No keyboard shortcuts

All users missed the ability to use the keyboard for different actions. Delete, copy and paste were the most missed shortcuts.

Overview of sub drawings

The novice users had a hard time moving around in the file system. They would have liked to have comments to the files and directories in order to better grasp the content. All users expressed an desire to preview the files before opening them in order to reduce the amount of attempts to open the right file.

Add dynamic

The experienced users have additional programs that helps them set up the dynamic in an image. DVdraw can add dynamic to objects, however is the process tedious and therefore have the users chosen to use other programs for this feature. The experienced users would like to be able to set up dynamic within the editor to make the process easier. The dynamic in the images in a important part and therefore is it also important that novice users can use this feature without having to learn another program.

Overall experience

All users felt that DVdraw was flexible and had a lot of potential. However, the design choices were bad and the system felt old. One experienced user explained that it could take up to two weeks for a new user to learn DVdraw completely. This is a long learning time when it comes to a graphical editor without a lot of special functionality. It is also costly for a company since
they can not count on the newly hired to produce anything the two first weeks. The users liked that the system had a dark color theme and said that it was easier for the eyes to look at. However, they would have liked that the background of the drawing area was white by default, to have a better contrast from the rest of the system.
5.3 Cognitive walkthrough

The cognitive walkthrough were conducted in different web based graphical editors. This section states the results of these walkthroughs and the overall user experience.

5.3.1 Method draw

SVG-edit is a fast web-based open source SVG editor [3]. Method draw is created by Mark MacKay and it is a fork of SVG-edit. The intention of method draw was to create a more user friendly editor. Method draw have a different Graphical user interface from SVG-edit, that uses other colors and icons (Figure 3). Mark MacKay have also removed and modified some features to give the editor a more simple and pleasant experience [8]. Method draw works in all modern web browsers.

![Figure 3: Overview of Method draw](image)

Results

Changing size of the drawing area was hard since the implementation was different from other graphical editors. In order to change a value the user had to hold the mouse pointer over the attribute they wanted to change and then move the mouse up or down in order to decrease or increase the value.
Usually when changing a value in an input field, the user can just select the attribute they want to change and type the correct value. No instructions or information is given which confuses the user and makes them wonder if they have taken the right action. However, the feature was nice when realizing how to use it, and it gave direct feedback if the user scaled the drawing area to the correct size. Selecting several objects could be done by holding down the mouse pointer and dragging over the wanted objects. It could also be done by holding down "shift" and click on the wanted objects. There was no indication of this in the system and the menu selection that says "Keyboard shortcuts..." does not work. Consequently, the user have to try out different keyboard commands in order to find the correct one. Zooming could not be done to a specific part of the drawing, even though the mouse pointer changed to a magnifying glass which indicates that the image will be zoomed in wherever the user clicks. There where no indication on how to zoom out from the image. Trying out different keyboard commands was the only way to make progress. In order to get to the correct area pan had to be used. Panning left and right was harder to figure out since the mouse pointer was a magnifier glass even when hovering over the scroll bar that could pan left or right. Drawing a square was done by choosing to draw a rectangle and then holding down "shift". It was indicated that a square could be drawn when hovering over this tool. However, was there no indication on how to do this. The same problem appears when trying to draw an circle. There where a "snap to grid option" in the menu. However, did it not seem like this tool works since nothing changed even though it was selected in the menu. No progress or indication of error were given to the user. There were no way to give an object a name. It was hard to notice the correct approach for this action. The correct way was to create an group of objects, then could the name of the group be displayed by double clicking on the group. This made the editor move up one layer and enabled the user to edit the created group. The name could not be changed here and the user had to go into the SVG-code and change the ID-name by hand. It was impossible to give an standalone object an ID which could be very problematic. When exporting an image to raster graphics there were sometimes errors in the code that made the image look different from the original image. The errors where not indicated to the user, the only way to find out was to open the SVG-code and look for the problems and change them by hand.
Overall experience

The layout of Method draw looks clean and gives a good overview. The grouping of the icons and features feels logical. It is easy to recognize where features can be found. The features that could not be found where not implemented in the application. The icons in the editor have the same style as icons in other graphical editors. This makes the user recognize the different features instead of recalling them every time they use Method draw. The icon size is slightly bigger then in other editors. A tool tip was displayed when hovering over an icon. The tool tip did not appear right away and when selecting a tool it disappeared right away. This is a good thing since giving a tool tip every time the user hovers over a tool gets annoying and is unnecessary for experienced users. Method draw crashed several times during the evaluation and some features had bugs. This makes the editor feel unstable and should not be chosen when drawing for commercial purposes or bigger projects.
5.3.2 Janvas

Janvas is a vector graphics editor which allows creation of vector graphics in HTML, SVG and PHP. It also allows the user to insert their own CSS and javascript files, making it possible to create a whole website in this editor [10]. Janvas is a project made by Riccardo Della Martire. At the moment Janvas is a free to use tool. However, after the next release a user will only be able to save 10 images a month for free.

![Janvas GUI](image)

Figure 4: Overview of Janvas

Findings

Changing color and size of the drawing area was straightforward and easy to use. The two different properties had good names which made the user associate their intention with the correct action. Under properties there is an option called 'Snap grid’, this indicates that a grid layout could be turned on. However, this action could not be found in the system and changing the value does not change anything visible. Giving objects names (ID:s) are straightforward and easy to achieve. When selecting the input field is it highlighted which indicates that something can be written in the field. However, no progress indication is given so re-selecting the object feels necessary in order to see that the object actually has the correct name. The path tool is quite hard to use and the different path-selections are not straightforward. It is not only the path tool that is hard to understand, some of the other icons are not
so explanatory which confuses the user and the right action are sometimes hard to find. Coordinates of a specific node in a path can not be changed. However, are coordinates for the whole path displayed even when a specific node is selected. This leads the user to believe that they can change the coordinates of that specific node even though it is for the whole path. A specific node can not be deleted by pressing the 'delete' option, the whole path is instead deleted. Deleting a specific node can be done with help of the path tool. The zoom-tool is easy to spot in the menu options. When pressing the icon the cursor changes to a magnifier glass which gives the user direct feedback. How to zoom is not straight forward since no instructions are given in the system. When saving a file both 'Save' and 'Save as' can be chosen. By convention these two selections mean different things, but in Janvas they work exactly the same and opens up the same tab.

**Overall experience**

Janvas has a nice design and the colors chosen for the editor matches well together (Figure 4). Displaying tool tips when hovering over icons would make them more comprehensive. All styling of objects are placed on the right side of the screen and all tools are placed together at the top of the screen. It feels logical to have the two types of features together. Janvas can only load and store files to Google drive which forces all users to create a Google account in order to use the editor. In order to save an image as raster graphics the user has to select view as PNG or view as JPG, and then download the image from the new tab. If the user tries to save file as normal in raster graphics the whole image will be white with an text saying 'undefined'. There are some pop-up windows in Janvas which gives the user some additional features. These pop-up windows can be resized and moved around allowing the user to have the most frequently used features easily accessible.
6 | Requirements

The conclusions in this section are drawn from the results in section 5. It states the most critical problems and how they should be solved in a new graphical editor. All the concluded problems are stated in two different requirement lists (??), which also contains more general requirements stated in section 5. The requirements have no severity rating and all are considered to be equally important and grouped by the associating feature or action.

**Remove, change or explain unclear features in the system**

The lack of error messages, instructions and other proper feedback in DVdraw made it hard for novice users to move forward with a task. Therefore, more feedback describing errors, what is going on or what a user is expected to do is needed. The feedback should also be placed so it is noticeable for the user and make a clear distinction between the different types of messages. It is important that the dialog field can be minimized in order to keep out of the way for experienced users. The menus in DVdraw are big and included unnecessary, unclear and rarely used features. Consequently, menus in a new graphical editor should be created with only the most frequently used features displayed. Unnecessary features should be removed, rarely used features should be grouped into under menus and unclear features should be renamed or explained with help of a tool-tip or be displayed as an icon representing the features functionality.

**Change tedious tools**

All drawing tools should be implemented so the user can focus on the drawing area instead of the menu. Therefore, selected tools should be kept selected until user selects another tool to use. All tools should have key-shortcuts, this enables the user to keep the mouse on the drawing area and still select new tools in the menu. It should be easy for the user to move around in the image. It is therefore important that the zooming and panning tools are easy to use and access. The selecting of fill and edge color should be easy to use and enable the user to choose any colour that the user could want, including transparent colours. To improve the color picker a colour palette should be added that contains the most frequent used colors and the users last used colors. The snap and grid feature should be on by default and the user should be able to change the snap size, grid size and grid background.
The new graphical editor should have undo/redo implemented to help the users recover from mistakes. The undo/redo should always be at hand for the user and easy to use.

**Handling variables and dynamic**

The graphical editor should enable the user to add variables and dynamic types to the objects in the image. The user should be able to search for variable names in the simulator.

**Color theme**

The editor is implemented in an already existing system with a dark color theme and therefore should the colors be adjusted to those already implemented for internal consistency. However, two additional contrast colors should be added in order to distinguish when something is focused or not. The drawing area should be white as default to follow external consistency from other graphical editors.

**Add overview of sub drawings**

The user should be able to see the sub drawing before adding it to the image. The images in the file system should therefore be displayed as miniature images in a list. It is also important that the user can search the file system easily and should therefore be able to search for keywords in the images.

**Overall conclusion**

Features in the new editor should not slow the experienced users down. It should therefore be possible for the users to turn off tool tips and hide unnecessary instructions. Drawing and editing tools in the system should be easy to use and follow standards from other graphical editors. The number of ‘clicks’ to achieve a goal should be as low as possible without removing functionality for the user. The system should be flexible in a way that the user can change the editor to their preferences and experties.

### 6.1 Requirements vector graphics editor

Table 6 states the features required by a graphical editor and their associated usability and functionality requirements.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Usability/Functionality requirements:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Should be vector text. The text should be easy to create and edit. Should be scaled with font-size. The text-box should be removed when a user clicks outside of it and the text-box is empty. Otherwise the text should be saved.</td>
</tr>
<tr>
<td>Pan</td>
<td>Panning the view should be done with help of keyboard arrows or by holding down left mouse-button and drag. User should be able to pan the view until another menu selection is chosen.</td>
</tr>
<tr>
<td>Zoom</td>
<td>Zooming should be done with help of the mouse scroll-wheel. Zoom should be centralized to where the mouse is. Should have a 'zoom to' selection in the menu (Mark an area in the drawing with help of holding down left mouse button, function should be aborted when pressing right mouse button).</td>
</tr>
<tr>
<td>Group</td>
<td>Should group several objects into one. Should have a group name without changing the names on the underlying objects. Grouping should stay until user decides to ungroup them. Should have different grouping selections, for example, 'All', 'All in' and 'Part in'. User should be able to group things with help of mouse. All marked objects should be grouped together if the 'G' key is pressed on the keyboard.</td>
</tr>
<tr>
<td>Drawing area</td>
<td>Color selection of the drawing area should be separated from other color selections. User should be able to have a dark or light transparent background. Change size of drawing area.</td>
</tr>
<tr>
<td>Edge, Fill</td>
<td>Should Be represented by a icon with clear distinctions between edge and fill. Should have transparent color option.</td>
</tr>
<tr>
<td>Menu names</td>
<td>The menu names should clearly state what features it represent. The names should be short and informative, abbreviations should be avoided.</td>
</tr>
<tr>
<td>Menu Icons</td>
<td>It should be clear for a user what the different icons represent. The Icons should be recognizable from other graphical editors (Follow some kind of standard).</td>
</tr>
<tr>
<td>Grouping</td>
<td>Grouping of tools and features should be logical. For example all drawing tools should be in one group and all file features should be combined under the same menu selection. Zoom and pan should be in the same group. Styling features should lie close together.</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Dialog field</td>
<td>Should inform user when something goes wrong in the system. Should inform user how to use more complicated features. Should inform user when something is finished, e.g saving an image. Should be placed at bottom of system.</td>
</tr>
<tr>
<td>Responsive design</td>
<td>System should be usable even on small screens. Should keep dimensions and not change menus.</td>
</tr>
<tr>
<td>Resize</td>
<td>Object should be resized either by dragging in corners of element, or by setting height and width of object.</td>
</tr>
<tr>
<td>Scale</td>
<td>Should work as resize, user holds down 'shift'-key to scale in two dimensions at once.</td>
</tr>
<tr>
<td>Reorder</td>
<td>User should be able to bring object(s) 'To front', 'To back', 'Forward' and 'Backward'. Bring object(s) 'In front of' a specific object. Should be able to reorder objects with keyboard arrows, 'Forward' (up) and 'Backward' (down).</td>
</tr>
<tr>
<td>Align</td>
<td>Should have Left, Right, Bottom, Top, vertical and horizontal alignment. Align several objects to each other.</td>
</tr>
<tr>
<td>Geometric primitives</td>
<td>Should have Rectangle/Square, Ellipse/Circle. User should be able to hold down 'shift' - key in order to draw a circle or a Square. Should be drawn by selecting two points in drawing area (After first click should a dotted line show how the finished will look). Drawing should be aborted when clicking right-mouse button.</td>
</tr>
<tr>
<td>Pencil tool</td>
<td>User should be able to draw lines freely with help of mouse pointer.</td>
</tr>
<tr>
<td>Path tool</td>
<td>User should be able to set path node by left click on mouse. A path should be finish by clicking on right-mouse button.</td>
</tr>
<tr>
<td>Feature</td>
<td>Requirement</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Selection tool</td>
<td>User should be able to select one or several objects. One object should be</td>
</tr>
<tr>
<td></td>
<td>selected by pressing it. Several objects should be selected by using the</td>
</tr>
<tr>
<td></td>
<td>mouse and either drag over all wanted objects or pressing a new object and</td>
</tr>
<tr>
<td></td>
<td>holding down the 'Ctrl' key.</td>
</tr>
<tr>
<td>Load image</td>
<td>Should at least support SVG files.</td>
</tr>
<tr>
<td>Saving image</td>
<td>Should at least be able to save an image in SVG-format.</td>
</tr>
<tr>
<td>Import subdrawing</td>
<td>Should be able to import an SVG-file.</td>
</tr>
<tr>
<td>Delete</td>
<td>User should be able to delete selected object(s). Delete should be triggered</td>
</tr>
<tr>
<td></td>
<td>either by pressing the delete key or by selecting delete in the menu options.</td>
</tr>
<tr>
<td>Eye dropper tool</td>
<td>Should be able to pick up both fill and edge color.</td>
</tr>
<tr>
<td>Undo</td>
<td>Should be triggered when user presses 'Ctrl' - 'Z'. Should be implemented in</td>
</tr>
<tr>
<td></td>
<td>the menu options. Should support undo at least 5 times.</td>
</tr>
<tr>
<td>Redo</td>
<td>Should be triggered when user presses 'Ctrl' - 'Y'. Should be implemented</td>
</tr>
<tr>
<td></td>
<td>in the menu options. Should be able to redo everything that has been undone.</td>
</tr>
<tr>
<td>Copy</td>
<td>Should be able to copy all selected items. Should be triggered when user</td>
</tr>
<tr>
<td></td>
<td>presses 'Ctrl' - 'C', or menu selection.</td>
</tr>
<tr>
<td>Paste</td>
<td>Should be able to paste everything that is copied. Should be triggered when</td>
</tr>
<tr>
<td></td>
<td>user presses 'Ctrl' - 'X' or menu selection.</td>
</tr>
<tr>
<td>Color Palette</td>
<td>Should display color code. User should be able to change color by typing</td>
</tr>
<tr>
<td></td>
<td>the color code. Should have at least 20 different colors as default colors.</td>
</tr>
<tr>
<td></td>
<td>Should have transparent color option. Should have possibility to create</td>
</tr>
<tr>
<td></td>
<td>gradients.</td>
</tr>
<tr>
<td>Active icons</td>
<td>The active tool should be highlighted and easy to recognize. Different</td>
</tr>
<tr>
<td></td>
<td>modes should have different mouse pointers, for example, panning, zooming</td>
</tr>
<tr>
<td></td>
<td>drawing and selecting.</td>
</tr>
</tbody>
</table>
Hovering icons should give information about what it is and keyboard commands associated with this tool. For example, hovering pan tool will give the user the following information "Pan tool 'Ctrl' + 'P'". The keyboard commands given is a shortcut for selecting the tool.

Table 6: Requirement list for a vector graphics editor

<table>
<thead>
<tr>
<th>Feature:</th>
<th>Usability/Functional requirements:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snap</td>
<td>Should be turned on by default. Should have the following configurations, Snap range.</td>
</tr>
<tr>
<td>Grid</td>
<td>Should be turned on by default. Should have the following configurations, grid size and grid color. Default grid size should be 5pt.</td>
</tr>
<tr>
<td>Add dynamic</td>
<td>Should have all different dynamic types. Should be able to connect dynamics to an ID instead of a variable name. Several variables should be connected to one ID. An object should be able have more than one dynamic.</td>
</tr>
<tr>
<td>Change dynamic</td>
<td>User should be able to change the attributes of the dynamic. For example, colors, thresholds and sub drawings.</td>
</tr>
<tr>
<td>Sub drawings</td>
<td>Should be ordered in different folders depending on type. For example all buttons in one folder. Every sub drawing should have a preview so the user knows how it will look before using it in the image. User should be able to search the system with help of key words. For example, 'Yellow' and 'button' should display all yellow buttons.</td>
</tr>
<tr>
<td>Variable names</td>
<td>When entering a variable name a list of all available variable names should appear. The list should display all names matching the users input. Variable names that does not exist in the simulator should be allowed, but highlighted and easy to spot.</td>
</tr>
</tbody>
</table>

6.2 Specific requirements

Table 7 states the specific features needed for a graphical editor at KSU. The table also states their associated usability and functionality requirements.
<table>
<thead>
<tr>
<th>Instrument scales</th>
<th>Should be able to set rotation degrees, number of ‘ticks’ on instrument and a value for each ‘tick’. Should be able to set unit of measurement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision control</td>
<td>Should have a revision control with a graphical user interface (GUI).</td>
</tr>
<tr>
<td>Colours</td>
<td>Colour palettes default colours should change depending on which simulator the user draws for.</td>
</tr>
</tbody>
</table>

Table 7: Special requirement list for KSU
7 | Design solutions

This section describes the different design solutions taken in the new system in order to meet the design guidelines, design principles, design rules and user requirements given in chapter 3 and 6.

7.1 Overall design

The overall design of the editor can be seen in Figure 6. The color and font theme for the editor is the same as the navigation bar in browsIS (Figure 5). The navigation bar has a dark theme with white and turquoise accent colors and Open Sans as a font style. There are two different font sizes 18px and 16px. An additional font size (14px) were added to the editor. The new smaller font size is used for input and dialog fields.

![Figure 5: Navigation bar in browsIS](image1)

The editor features are grouped into two separate menus. The two different menus are placed on each side of the drawing area. A dialog field is placed at the bottom of the editor. It can be opened or closed and the background has an opacity at 50% in order for the user to notice objects on the drawing area even if the dialog field is covering that part.

![Figure 6: Overview of vector graphics editor at KSU](image2)
7.2 Dialog field

The dialog field displays (Figure 7) all instructions, feedback and error messages to the user. The different types of messages have different colors. Instructions are yellow, feedback is green and error messages are red.

![Figure 7: Opened dialog field](image)

The dialog field has the default settings to open when a new message arrives and stay open in 10 seconds then close. However, these settings can be changed by pressing view messages, enabling the user to keep the dialog field opened or closed. If the dialog field is closed only parts of the latest message is displayed for the user and it will not open when new messages arrive. If the user chooses to open the dialog field again it will use the default settings.

![Figure 8: Closed dialog field](image)

7.3 Left side menu

The left side menu contains drawing, Zoom/Pan, eye-dropper, undo/redo and subdrawing tools (Figure 7). These features are represented by descriptive icons. The currently selected tool is indicated by a change in the shadows and icon changes color from white to turquoise. Above the tool icons are two rectangles displayed, these two rectangles represent the edge and fill color picker. The color picker is opened by selecting one of these rectangles. There are three selections ('File', 'Edit' and 'View') at the top of the menu. These menus will open up different drop down menus if the user hovers the mouse pointer over them.
**File selection**

By hovering over 'file' in the left side menu a drop down menu appears for the user (Figure 9). The user can choose to create a new or load an existing file. If a file have unsaved changes the user can choose to save changes, discard changes or cancel the action. The user can also choose to save a file by selecting either 'Save file' or 'Save file as'. Save file as will allow the user to change or add a page name, add user name and a comment to the image. The save file option will save the file without any need for extra user input. Selecting 'Import image' will give the user an overview of all images available in the system, the user can then choose to import one of these images into the current image as a subdrawing. Save subdrawing can be used by the user in order to save an selected object or a group of objects in the image as a sub drawing. When an image is saved it is stored as versions with 2 optional parameters, user name and comment. If a user have made some changes and accidentally saved the image they can recover from that error by loading an older version.

![Figure 9: File selection](image)

**Edit selection**

By hovering over 'edit' option the user is given to option to cut, copy or delete an selected object (Figure 10). If no object is selected then nothing will happen. The user is also given the option to paste an already cut or copied object. If an object is not copied or cut nothing will happen.
View selection

By hovering over the 'view' selection the user is given functionalities to manage and configure the view (Figure 11). By selecting configuration a new dialog field is opened that enables the user to change the grid color, background color, page height, page width, grid-line width, grid size and snap distance (Figure 12). By selecting the colored box next grid or background color a color picker is displayed to the user enabling them to choose any color preferable. Changes made in the configuration is saved by selecting 'Ok'. The user can also choose to turn on and off snap and grid. This is done by clicking on the turquoise tab and it will toggle between on and off. Selecting source opens up a new dialog field which enables the user to view and edit the source code of the SVG-image. By selecting shortcuts a new dialog field is opened for the user presenting all available shortcuts in the system and what they do.
The color picker can be opened from various places in the editor. In order for the user to change edge or fill color the user can select the two different rectangles in the left side menu (Seen in figure 11), this will open up the color picker displayed in figure 13. The color picker has a color palette which contains the most and latest used colors in a project. When choosing a new color the user can either type the color code (hsv, rgb, rgba, hsl and hex) or choose a color manually with help of the tools in the right side of the color picker. The previous color is displayed to the user alongside the new color in order for the user to get an better overview. To choose a new color the
user has to click the 'Choose' button, selecting something outside of the color picker will close it.

![Color picker](image)

Figure 13: Color picker

**Tools**

The tools in the left side menu are displayed as icons (Figure 14). The selection tool is represented by a mouse pointer. This tool enables the user to select one or several objects in an image. Instructions on how to select several objects is displayed in the dialog field. When this tool is selected it is also possible to move the selected objects in the image. The pan tool is represented by four connected arrows. This tool enables the user to move around in the image. It is also possible for the user to select one object by double tapping it. The zoom-to tool is represented by an magnifier inside an rectangular. The user can always zoom by using the scroll-wheel on the mouse. However, this tool enables the user to mark a specific area in the image that the user can zoom to. The path tool is represented by a pen and a line drawn with connected dots. This feature enables the user to draw figures with help of connected straight and curved lines. Import an image (sub drawing) enables the user to quickly import sub drawings to the image. When selecting this tool the current sub drawing is given to the user in the dialog field as well as instructions on how to change current sub drawing. The user can place the sub drawing anywhere in the image by just selecting a spot in the drawing area. The pencil tool is represented by a pencil and a drawn line. This feature enables the user to draw freely in the drawing area. A user is able to draw a rectangle or circle by selecting the circle or rectangle icons. They can be drawn anywhere in the drawing area by selecting a start
point and moving the mouse. The eye dropper tool enables the user to pick up colors from drawn objects in the image. It can pick up both edge and fill color if the selected object has those properties specified. The text tool enables the user to add a vector text to the image.

![Figure 14: Tool Icons](image)

**Tool tip**

When a user is hovering over a tool for more than one second a tool tip is displayed that describes what tool the icon is representing (Figure 15). If selected tool has a short cut is this also presented in the tool tip between two quotation marks.

![Figure 15: Tool tip](image)
7.4 Right side menu

The right side menu contains 4 tabs with drop down menus. The whole menu is responsive, meaning that the size of the menu is changing depending on how many tabs the user chooses to open. This enables the user to have more space for the drawing area when needed.

![Right side menu](image)

Figure 16: Right side menu

Properties

The properties tab contains properties given to a specific object (Figure 16). The properties given are height, width, x-and y position, rotation and stroke width. These properties can be changed by selecting the input field and change the value. If several or no object is selected then only stroke width can be changed.
**Text properties**

When the user wants to edit or add a text, the properties field is changed to match the properties of the text object (Figure 17). Width and Height are displayed but cannot be changed instead the user has to change the font size in order to change these attributes. The user can also edit the font style and choose to align the text to the left, right or center.

![Properties](image)

**Figure 17: Text properties**

**Dynamics**

The 'Dynamic' tab gives the user a possibility to connect values from the simulator to different dynamic types. When the user selects the tab, a preview of the dynamic connected to a selected item is displayed (Figure 18). All objects in an image have an ID that can be changed and connected to different dynamic types. The user can choose to delete all dynamic connected to an ID by clicking the 'Delete' option.
If the user would like to add a dynamic type or change specific parts of an already existing dynamic they can select the 'change' option, which will open up a new dialog field. At the top of the dialog the user can select input commands. The input commands specifies how the objects in a image should behave when a user clicks on it. For example, a user can choose the input command 'push back' which indicates that the applications should send a value to the simulator for one second and after that send the original value. The input commands can also be switches or links to other web pages or panel images. The input command can connected to an variable in the variable database or to a link. One object can have several output dynamics connected to it. In Figure 19 the object (with ID ii20922dropstate_203_) has one output dynamic connected to it. This dynamic is by itself connected to a variable in the simulators variable database. The user can choose to copy the dynamic to rapidly create a new similar dynamic type. A user can also choose to remove the whole dynamic or just the dynamic-type connected to the variable.
By selecting 'New' the user is able to add another variable to the list and a dynamic-type connection. The different dynamic types available for the user is displayed in figure 20.

A user can add a threshold by selecting the value interval. By selecting an already existing threshold a new dialog is opened where the user can change
the threshold values, change dynamic output or delete the threshold. Figure 21 shows how the dialog looks when changing a threshold value for the color dynamic. In this dialog can the user also change the color connected to that specific threshold.

![Change Dynamic dialog](image)

**Figure 21: Change dynamic dialog**

**Align**

By pressing the 'Align' tab six different align features are displayed to the user (Horizontal, Vertical, Top, Bottom, Left and Right align) displayed in figure 22b. All features are displayed as icons and by hovering over an icon a tool tip is displayed for the user. Selecting any of the align features will display instructions on how to use the feature in the dialog field. The user can align objects to the image or to other objects.

![Reorder tab](image)  ![Align tab](image)

(a) Reorder tab  (b) Align tab

**Figure 22**

51
Reorder

By pressing the 'Reorder' tab five icons are displayed to the user with different reorder selections (Figure 22a). The user can select 'To top', 'To bottom', 'Up', 'Down' and 'In front of'. By hovering over an icon and keeping the mouse pointer there for one second will display the name and functionality of icon as a tool tip. When selecting 'In front of' the user is given additional instructions in the dialog field telling the user how to use this feature.
8   |  Summary

8.1  Discussion

This section will discuss the methodology in this thesis and what could have been done differently given more time.

Methodology discussion

After concluding the results from the Usability evaluation it was clear that the test plan was more suitable for the experienced users than novice users. A more divided test plan would have been more suitable. Where the novice users should be given a longer task list with much more basic tasks to complete in the beginning, novice users should also have been given more time during the usability evaluation, allowing them to explore the system more. When concluding the results all requirements were put together into one list. It would have been interesting to see if the result would have been different by dividing the requirements into novice and experienced users requirements.

Given more time...

The steps 4.2-4.5 in the methodology should have been done several times on the new prototype in order to confirm the findings from the last evaluations and finding more requirements. This is time consuming but an important approach since it can either be an confirmation that the prototype is moving in the right direction or if it is implemented wrong. When the prototype was completed should an evaluation been conducted to conclude if the prototype actually meets both experienced and novice user demands. This would have been a crucial in order to find out if the research approach was enough to conclude requirements from both user groups. Due to lack of time were only a limited amount of functions evaluated, the focus was on graphical design and where functions should be displayed. The visual display is only a small part of an graphical editor and given more time should all functions be evaluated.

8.2  Conclusion

The conclusions of the research questions and the thesis are presented in the following sections.
User requirements

The first research question was to examine what user requirements a new web based editor should have. The most important requirements from the usability studies can be found in the requirements lists 6.1. Some features in DVdraw needed to be implemented differently since they were tedious to use for both novice and experienced users. The study participants also wanted to add additional features that they thought would make the editor more user friendly. The editor should allow the user to keep as much focus as possible on the drawing area and also have an responsive design so it can be used regardless of screen resolution. Limit the amount of features visible for the user when entering the editor in order to avoid confusion.

Design solution based on requirements

The second question was to examine how the user requirements could be used when deciding the design solution. All features that was unnecessary or redundant were removed. All features that felt tedious to use were implemented differently based on suggestions from the study participants. All features were grouped depending on their functionality, these groups were then placed in different menus in the application depending on placement in other web based graphical editors. Tool tips and a dialog field were implemented in the editor in order to give the user information and feedback while using the editor. Features that the users expressed that they needed were implemented, both completely new features and features kept from the old editor. The overall design of the editor were kept simple and as small as possible in order to give place for a drawing area.

8.3 Future work

To finalise this research usability studies should be performed on the new implemented prototype. This would tell if the methodology was designed correctly and if the prototype indeed fulfils all user requirements. This should be done repetitively until all user requirements are fulfilled and the feedback from the usability studies is overall positive.
Bibliography


Appendices
A | Scenarios Cognitive walkthrough

Common scenarios

Scenario One
The user is asked to draw an exact replica of a given image. It is important for the customer that every object in the image has the correct position and size. All added elements should also have the correct name given to them.

Scenario Two
The user has to draw very specific details to an image and therefore need to zoom and pan to a specific place in the image.

Scenario Three
The user makes a mistake when drawing an object and wants to undo the last changes to the object without deleting the whole object.

Scenario Four
The user is asked to do a logo for a company. The user is given some examples and a list of requirements for the logo. The user is free to be creative and draw the logo however (s)he thinks is suitable.

Uncommon scenarios

Scenario One
The user is asked to save the image in raster graphic format instead of vector graphic.

Scenario Two
The user is asked to animate several of the objects that are drawn in the image.

Scenario Three
The user is drawing an image after a while (s)he notices that one object is under all other and wants to move it to the front without having to move all other objects.

Scenario Four
The user is asked to open an existing image and do some changes on a given image. It is important for the customer that the changes appear on the exact right place and that the new added objects have the correct size. All added elements should also have a correct name given to them.