Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Workflow graph editing and visualization in HTML5 and Javascript

by

Marcus Alfredsson and Eric Lundmark

LIU-IDA/LITH-EX-A–15/018–SE

May 19, 2015

Linköpings universitet
Institutionen för datavetenskap

Final thesis

Workflow graph editing and visualization in HTML5 and Javascript

by

Marcus Alfredsson and Eric Lundmark

LIU-IDA/LITH-EX-A–15/018–SE

May 19, 2015

Supervisor: Anders Fröberg

Examiner: Erik Berglund

Abstract

Being able to run applications written in a single language on multiple platforms is a strong incentive for migrating applications to the web. This along with the possibility to avoid the sometimes problematic procedure of installing software, makes the case even stronger. This thesis investigates how to migrate a workflow graph editing system into a web technology in order to publish it on the web. We will evaluate a number of different technologies such as WebGL, HTML5 canvas and SVG. SVG is deemed as the preferred technology due to its advantages when it comes to interaction. As it can leverage JavaScripts event system we get a potent way of handling events without writing a single line of code. When combining this with the framework D3JS we achieve a great tool for writing workflow management systems.

Contents

Chapter 1

Introduction

1.1 Background

This master thesis was conducted at Ida Infront in Linköping, Sweden. Ida Infront is a company that is eminent in e-Government. They produce systems for e.g. document handling, process support and case management. They have 70 employees spread out among their offices in Linköping, Stockholm and Sundsvall.

1.2 Domain

The graphs that we will handle can be considered relative small. They will consist of a maximum of 200 nodes and 200 transitions. The application will be designed to handle these quantities.

1.3 Problem description

There exists many different technologies to build systems that visualise information. These technologies have their respective advantages and drawbacks. The problem in this case is to find a technology that address the needs of a workflow application.

1.4 Purpose

The purpose of this report is to investigate which different technologies exists for implementing a workflow graph editing and visualisation tool. Our research question incurred from the issue in migrating graphic intense software to the web. The research question investigated in this thesis is:

- How can a workflow graph editing and visualisation tool be implemented in web technology?

1.5 Approach

In this study, the starting point is a feasibility study where focus will lie on a literature study. The literature study will result in a number of hypothesis, which will be analysed after the case study. The case study will consist of implementing the workflow tool, which will be performed with an agile work approach. After the case study is complete an evaluation of the used technique will be performed. In the evaluation the hypothesis will be confirmed or denied based on the empirical findings from the case study.

1.6 Limitations

These are the limitations of this thesis:

- A limitation is that the researcher of this case study are the participants that perform the study.
- We have limited experience of Scrum.
- Due to the limited amount of time we will not have the recommended 30-days per sprint.
- The implemented tool shall be represented in a 2D format.
- The selection of frameworks is limited to those free of charge.
- As we are inexperienced programmers the implementation may not be the best solution.

Chapter 2

Method

This chapter will describe the method used in this study. First, it consists of theory behind the method and at the end our approach as a conclusion of the theory.

2.1 Literature study

Webster and Watson(2002) describes a framework, consisting of three steps when identifying relevant literature:

1. **Start with leading journals** as these will likely contain the major contributions to the subject.
2. **Go backward** by reviewing references to determine prior articles.
3. **Go forward** by reviewing citations.

The review is considered complete, when there are no new concepts discovered. [35]

2.2 Requirement elicitation

Requirement elicitation is the process of identifying the system to be developed. One of the most important techniques are interviews. Paetsch et al.(2003) states that interviews allow mistakes and misunderstandings to be identified and cleared up. [25] There exists two different kind of interviews:

- **Closed** - Pre-defined set of questions
- **Open** - No pre-defined questions, instead the engineer and stakeholders engage in a discussion

Paetsch et al.(2003) concludes that the researcher get a rich collection of information during open interviews, but it can be hard to analyze due to the amount. [25]

2.3 Case study

When conducting a case study, it can vary depending on the purpose of the study. One research methodology does not fit all case studies, it is therefore important to establish the purpose. [30] Robson(2002) describes four different methods for conducting a case study:

- **Exploratory** - When the purpose of the research is seeking new insights or generating ideas for future research.
- **Descriptive** - Used for research when the purpose is to describe a situation or phenomenon
- **Explanatory** - When the research is seeking an explanation to a situation or problem.
- **Improving** - Used for research that is trying to improve certain aspects of the studied phenomenon

Depending on which type of purpose is defined, different methodologies can be applied when analysing the empirical findings. This is why it is important to know the type of purpose before continuing with a case study. [29]

According to Runeson and Höst(2009) there are five major steps that need to be walked through in a case study:

1. Case study design - described under the section "Case study design and planning".
2. Preparation for data collection - described under the section "Case study design and planning".
3. Collecting evidence - described under the section "Data collection".
4. Analysis of collected data - described under the section "Data analysis".
5. Report - The report will consist of the hypothesis confirmation, which is presented in the discussion section.

2.3.1 Case study design and planning

In the beginning of the case study it is important to define the objectives of the case study. A plan should be defined to make sure that all elements are included and in that plan the following questions must be answered:

- **Objective** - what to achieve?
- **Case** - what is studied?
- **Theory** - frame of reference?

- **Research questions** - what to know?
- **Methods** - how to collect data?
- **Selection strategy** - where to seek data?

These questions will handle the first two steps of the five major steps in Runeson and Höst(2009), that should be walked through in a case study. [30]

2.3.2 Data collection

When collecting data there are usually many different data sources included to minimise the effect a single data source can have on the empirical findings. In a case study it is also important to take different opinions of different roles into account, like different projects and products. [30]

Lethbridge(2005) has identified that data collections techniques can be divided into three different tiers:

- **First degree** - the research has direct contact with the subjects and collect data in real time. This can for example be interviews, focus groups and work diaries. This technique require the most resources and has lowest reliability of the three. On the other hand it is the most flexible, leading to e.g. that research questions are interpreted in the correct context.
- **Second degree** - the researchers collect data in real time without interacting with the subjects. Can be used when observing video records for example and documentation analysis.
- **Third degree** - the researchers analyse the already available data. This can for instance be analysis of requirement specification and failure reports. This technique has the lowest requirement regarding resources and has the highest reliability. However this method is also the least flexible, as there can not be any alteration to the research question.

[19]

According to Runeson and Höst(2009) data can be collected either in a qualitative or quantitative way. Quantitative data collection usually consists of numbers and facts, which can be seen as statistics. This gives a very clear picture of the empirical findings but also gives a quite shallow understanding. Qualitative data collection offers a deeper understanding in the empirical findings. [30]

Work diaries are a first degree data collection technique, as described by Lethbridge(2005). Work diaries are a qualitative data source as they consist of reflections rather than numbers and facts. Work diaries are a good way to obtain accurate information about developers work practices.

This method has advantages when it comes to reporting events. For instance, this method report events and decisions as they occure rather than in retrospective. Random sampling of events also gives researchers a good way of understanding how software engineers spend their day without requiring a great deal of observation. Work diaries has some disadvantages which needs to be considered. One of them being that participants require to recall events of significance with accuracy. Another disadvantage is that the diary can infer with the engineers normal workflow. For instance, if the diary should include how often the participants ask for help from colleagues, this can lead to the participants avoid asking for help. [19]

2.3.3 Validity

Validity is vital when conducting a case study. It is crucial to know that the study provides trustworthy results. The results should be accurate and the researchers point of view should not be biased and reflected in the study. [30]

Construct validity measures if the operational measurements are what the researchers expect and that the participants interpret the research question in the same way as the researchers. For instance if a research question is interpreted differently by researchers and interviewed persons, there is a threat to validity. [30]

External validity checks to what extent the empirical findings can be generalized and to what extent the findings are of value to people not a part of the investigated case. In the analysis regarding this aspect, researchers will try to find in which extent the empirical findings can be used in other cases. The intention is to be able to generalize findings so they can be used in other case studies with similar characteristics and therefore the findings are relevant for these cases as well. [30]

Internal validity is measured when causal relations are examined and which impact underlying factors have. When researchers examine how one factor influence another, a risk exists that other factors are affected. If researchers are unaware of this correlation there is a threat to validity. [30]

The *reliability* aspect is concerned to which extent data and analyse, depends upon the researchers performing the case study. The study should reach the same result with another research team. Threats to validity can be an unclear description of how data is collected or if certain interview questions are poorly described. [30]

2.3.4 Triangulation

One of the most important aspects of empirical findings is that they are reliable and validated. To arrive at reliable and validated findings, there is a need for structure. Triangulation is a protocol of different ways to collect empirical findings that are valid and accurate. There are four different protocols for triangulation:

- **Data source triangulation** - This protocol checks if the case remains the same at different times for a data source. If the observed and reported data source has the same meaning when observed and reported under different circumstances, the data source fulfills the triangulation protocol.
- **Investigator triangulation** - In this triangulation protocol several researchers investigate the same case. This results in several interpretations of the empirical findings and therefore, a more accurate interpretation. Researchers that can be used in this triangulation protocol are colleagues or a panel of experts.
- **Theory triangulation** - Whenever two investigators compare their findings, there is a theory triangulation, since two investigators never entirely interpret data in the same way. When findings are compared, and matched, the findings are valid. Otherwise more research is required to produce findings that is interpreted in the same way by both investigators.
- **Methodological triangulation** - This protocol uses multiple methods for data collection, to make sure that the empirical findings will be complete and nothing is missed.

[33]

2.3.5 Data analysis

This is the fourth step of five in Runeson and Höst(2009) steps towards a successful case study. The analysis of data will be based on qualitative empirical findings. The intent of the analysis is to derive conclusions from data and to have a clear chain of evidence. Each part of this chain must carry sufficient information from the study and every decision taken by the researcher must be presented. [30]

Qualitative data collection is seen as a flexible technique and it is therefore important to analyse data in parallel with the data collection. This technique gives a better and more complete analyse, because analysing data during the collection process can lead to new insights. These insights can be used to alter some approaches in the data collection process to arrive at better empirical findings and a better analysis of the empirical findings. [30]

Analysing qualitative data is usually done with hypothesis and when analysing a case study with explanatory purpose, hypothesis confirmation is the most common technique [30]. Hypothesis confirmation can be confirmed through analysing the empirical findings. [32]

2.3.6 Scrum

The rapid development of the web sets high demands on the development process. There is a need to quickly adopt to new requirements in order to accommodate the demands of the users. [5]

Agile methodologies is an attempt to achieve this, as stated in the agile manifesto, "by rapid, incremental delivery of software" [6].

There are several different agile methods and one of the most popular is Scrum. Meyer(2014) describes "the most distinctive characteristic of Scrum" as the "closed-window rule". It is stated in the agile manifesto that agile processes are open to change. But Scrum does limit the window where change can be introduced, it is only allowed if the change does not disrupt the current iteration. [21]

Scrum is built up by small groups of people called Scrum teams. Scrum teams work in short iterations called sprints. A sprint consists of 30 consecutive calendar days. Every sprint starts with a planning meeting where the product owner and the team decides upon what should be done during the sprint. [21]

The work that should be performed is described in a Backlog. There are two different Backlogs, the Product and the Sprint Backlog. The Product Backlog is a dynamic list of requirements which evolves as the project evolves. During each sprint a selection of items from the Product Backlog that will make up a new release is put into the sprint Backlog. The sprint Backlog is an overview of the work that should be done during the current sprint. [21]

The Scrum team is divided into three roles:

- **Product Owner** - The representative of the stakeholders in the project
- **Scrum Master** - Ensures that the rules of Scrum are followed, not a regular project manager
- **The team** - Members that perform tasks from the backlog

[31] The product owner is the face of the stakeholders. The product owner should work closely with both the Scrum Master and the team trying to figure out how to get the most value from the business. The requirements produced goes into the Product Backlog. The product owner is responsible for managing the backlog in such a way that functionality with the most customer value is produced first. [31]

The Scrum Master is responsible for teaching and making sure that the rules of Scrum are followed. It is also the responsibility of the Scrum master to make sure that the Scrum process fits the organisation, if not it should be tweaked so that the expected benefits of Scrum is realized. The Scrum Master is also responsible for maximising return on investment and removing any barriers that disables the customer from directly driving the development. [31]

The team is responsible for developing the functionality described by the backlog. It should be performed in a top-down approach, developing the most valuable functionality first. The team should be self-organizing, self-managing and cross-functional. The members are responsible for the success of each iteration and as such, the whole project. [31]

Coram and Bohner(2005) states that releasing a functional product in short cycles allows the product to be evaluated as the development continues. The evaluation can then act as a foundation when deciding and changing the priority of upcoming features making sure the product evolves in the desired direction based on customers needs. [9]

It is not only the aspect of customers being able to influence the direction that the product evolves in, that ensures quality. Quick releases also gives the possibility to test often, or as said in Begel and Nagappan(2007), "When you integrate early and often, the product can be tested early and often, too". [7]

2.3.7 Our approach

Our approach starts with conducting a literature study as part of the feasibility study. The feasibility study contains requirement elicitation, which consists of analysing the previous system and interviewing the product owner in an open interview. The purpose of the interview is partly to check that conducted requirements were correct and to acquire new requirements which is not a part of the previous system. From the result of the feasibility study we derive some hypothesis which are answered in the discussion section. The empirical data from the case study consists of a developer diary. This diary is written on a daily basis and documents every day reflections of using the chosen technique. We use Scrum as development methodology in the case study. Scrum is used to provide a flexible methodology that can handle alternations in requirements during the development process. Scrum is also used so we can get feedback from the product owner and product supervisor about which aspects of the product that can be improved. In the discussion section we argue for the case, that our empirical findings are valid and reliable using triangulation. From the result of the literature study we follow a series of hypothesis. These hypothesis are be analysed with regards to the empirical findings from the diary, leading to that the hypothesis being confirmed or denied in the discussion section.

Chapter 3

Theory

This chapter will describe theory about workflows and visualisation technologies.

3.1 Workflow management systems

A workflow management system is used to help people reduce workload by organising business processes, also known as workflows. In a workflow management system it is important to analyse the workflow, in order to make sure that all important concepts are included. The important concepts are:

1. **Case** - A case can be described as the purpose of the workflow. Each case has a limited lifetime, with a starting and ending point in the process. Under this lifetime each case consists of a specific state depending on which tasks that has been performed. A case can have attributes, that can e.g. control under which circumstances a task can be superseded. A case can also have conditions which reflect the progress in a case and thereby reflect which tasks that have been performed and which that are left.

2. **Task** - A workflow is structured by a sequence of tasks and can be e.g. checking personal data or stamping papers. A task can be either automatic, semi-automatic or manual. A automatic task is performed exclusively by a machine, a manual task is performed exclusively by humans and a semi-automatic task is performed by humans assisted by machines.

3. **Process** - This part describes in which order that certain tasks should be carried out to complete a successful case. A process can also consist of several subprocesses. The process has a limited lifetime and the lifetime is the same as for the case.

4. **Routing** - Routing describe how workflows can choose to perform different tasks. Routing can perform tasks in four different ways:
    - Sequential - executes tasks in order, one after each other.
    - Parallel - executes several tasks simultaneously.
    - Selective - executed one of several paths, depending on input.
    - Iteration - executes a task several times.
5. **Enactment** - This part describes how a task is triggered. The trigger often depends on the type of task e.g. a manual is triggered when a person takes initiative to start it, and an automatic task triggers when an required resource becomes available.

These properties should be represented in all workflow managment systems. [1]

3.2 Visualisations on the web

This section will describe different approaches to visualisation on the web.

3.2.1 Raster graphics

"Raster graphics are image files made up of individual color pixels or dots". This can for example be an image taken by a digital camera. The downside with raster images is that it is made up of individual color pixels or dots, the image will not handle scaling as the image will be seen as unclear and grainy. [27]

3.2.2 Vector graphics

Vector graphics are made up of mathematical formulas. The image is based on vectors which make the image cope very well with scaling, compared to raster images. Vector graphics can be used as the underlying technique in native file formats such as the scalable vector graphic(SVG) format. [27]

3.2.3 Document Object Model

The Document Object Model(DOM) is a representation of a website. It can be interpreted and manipulated through Javascript. The model is built up as a tree structure where html, the document, is the root element. There exists several different node types, such as:

- Element nodes.
- Text nodes.

A notable thing about elements is, as stated by Keith(2005), that ”not all elements contain attributes, but all attributes are contained by elements.”. [15]

3.2.4 Flash

Flash is a vector graphics format standardisation. It has been one of the most used and widespread technologies, when it comes to visualisations on the web [24]. Though it is a format for vector graphics, a flash file can contain a combination of raster and vector graphics [26]. Flash uses Actionscript in combination with simple frame-based animation to ”create high-impact moving images to make attention-grabbing Websites”. These websites are often used in commercial contexts [28]. Being widely used in commercial contexts has made Flash a target for attackers. To be able to use graphics written in Flash, the user needs to install a plug-in in the browser. This plug-in has proven to be a security risk as it opens a hole in the sandboxed environment, in which websites run. [2, 10]

Flash programs are stored as pre-compiled executables, and as a result the file size is large compared to HTML and Javascript files. There are also issues with accessibility and performance. Screen reader support only exist in Windows and a lot of Flash applications has poor support for use of keyboards. Flash applications are known to be CPU intensive, especially on Linux and Mac. The flash format is also not indexable as it is stored in an executable and not the web page, resulting in the information not being available in search engines. This issue can though be avoided by creating a second web page, where instead of the content, a description is provided. [34]

In 2010 Apple publicly announced that they would not support Flash on their products iPad and iPhone. Instead they favored their own technologies, now incorporated in HTML5. [4]

3.2.5 VML

Vector Markup Language(VML) defines a certain format for encoding information of vectors with additional markup. Markup that describes how information may be edited and visualised. [8]

VML is now a deprecated technique and as stated by Microsoft ”SVG, implemented in Windows Internet Explorer 9, provides the functionality needed to replace VML in websites and applications that use it.”. Microsoft also states that VML is deprecated for Internet Explorer 10 standard mode. [23]

3.2.6 Canvas

Canvas is an HTML element which got capabilities that allows for rendering raster graphics. It is one of the most popular technologies for drawing graphics in web applications. The canvas element itself does not allow for drawing graphics. But it provides a context that allows access to the canvas area through Javascript, which then can draw pixels. [20] Johnson(2008) explains that

> "Since this rendering is procedurally generated, only the final image must be stored in memory once the Javascript code is executed; the canvas raster image is cached and only re-generated on request." [14]

Canvas is an immediate-mode API. There is no model stored within the graphics library that describes what to be drawn. Instead the instructions are sent directly to the application when a new frame is to be drawn. [22] There are no DOM elements generated inside the canvas element [34]. As a result of this the graphics library does not support events, this needs to be implemented by the developer.

Canvas has the ability to draw text using fonts, but it is an expensive operation. Canvas needs a lot of resources, compared to other technologies, to draw vector fonts as it needs to rasterize the given font before drawing it. There is support for raster fonts, but the same issues apply here as with regular graphics, it is not zoomable. [11]

In an article investigating image manipulation in Internet applications, Steenbergen(2010) concludes that "Canvas has the strongest papers for image manipulation since it has the capability to access and change a single pixel on the drawing canvas using a few lines of Javascript code. Any existing image effect can therefore be recreated within the browser". He further states that "the only limiting factor to the technology is the Javascript processing speed of the web browser." [34]

3.2.7 WebGL

WebGL is designed as a Document Object Model(DOM) API which creates 3D graphics in the web browser. WebGL is based on OpenGL, which offers the similar API but in a desktop environment. WebGL is also fully integrated with browsers, an application built with WebGL can use Javascript infrastructure and DOM fundamentals. WebGL gives another rendering context on the canvas element and can therefore be cleanly combined with HTML. [16] Using WebGL gives several advantages, such as an API that is widely accepted as a 3D graphics standard. WebGL has gained traction lately and is now supported in every major browser. This allows for implementation of cross-platform hardware-accelerated 3D graphics applications. [11, 17]

According to Hui et al.(2012) other benefits of using WebGL is that due to the hardware-accelerated 3D graphics, battery savings can be made [13]. However this is contradicted by Garaizar et al.(2012), which states that WebGL should not be used for applications with low power consumption [11]. One benefit of using the hardware-accelerated graphics is that it gives a fast application, since more of the hardware capabilities are used than in other technologies. WebGL also use less CPU with high performance compared to the standard canvas technique. [13]

3.2.8 SVG

”Scalable Vector Graphics (SVG) is an XML dialect used to describe vector graphics wherein images are composed of drawing elements rather than raster pixel colors”. A picture in an SVG model consists of a list of paths, basic shapes and text, each one of these XML elements has its own appropriate graphical attributes. The vector image is then turned into a bitmap by the web browser. One great benefit of SVG visualisations is that they become very concise, since the entire pixel plane does not have to be specified, only the visual elements for the figure. One downside for SVG is if the visual elements exceeds the bitmap size, the visualisation will become very memory expensive. [14]

SVG is in contrary to WebGL and VML open source. It is also an official W3C recommendation but was not present in the initial HTML5 specification. However SVG has now been adopted to the specification. [18] It is supported natively in the browser, reducing security risks. [24] SVG is a retained-mode graphics API in contrary to canvas and WebGL. This means that the graphics library keeps an internal representation, a model, of the graphics elements. Every time a new frame is to be drawn the model is first updated, allowing the library to handle initialisation, state maintenance and cleanup. [22] The model consists of elements in the DOM and this allows for user interactivity through Javascript as event handlers can be attached to the elements. These event handlers can listen for events such as click, making it easy to implement functionality for moving objects. [11, 14] One other advantage described by Garaizar et al.(2012) is the zooming operations which SVG handles well.

As with all technologies there are some disadvantages, one technique can not be best in every aspect. For SVG the most concerning disadvantage is that it can be perceived as slow. This problem arises when SVG has to handle a big number of shapes. [11]

3.2.9 Canvas vs. WebGL

As previously stated WebGL is hardware accelerated and canvas is not. Hui et. al.(2012) has performed a comparative study on visualising objects. In the study the difference between the two technologies can be seen.

Test number	Sprite number	FPS test result (Browser Platform)			
		Chrome 20		Firefox 10	
		Canvas	WebGL	Canvas	WebGL
1	1	60	60	60	60
2	10	60	60	60	60
3	100	57	60	57	60
4	1000	32	28	29	58
5	10000	9	31	5	20

Figure 3.1: Results from the study performed by Hui et. al. [13]

The study shows that the difference in performance grows larger as the the number of sprites increase. But it is not until there is a vast amount of sprites that a significant difference can be detected. The bottleneck in rendering graphics, using HTML5 canvas, resides in the Javascript engine. This is a result of canvas not being able to leverage the GPU in the same way as WebGL. Instead the CPU is used, the difference can be seen in figure 3.2 where the CPU load increase exponentially as the number of sprites increase when using HTML5 canvas.

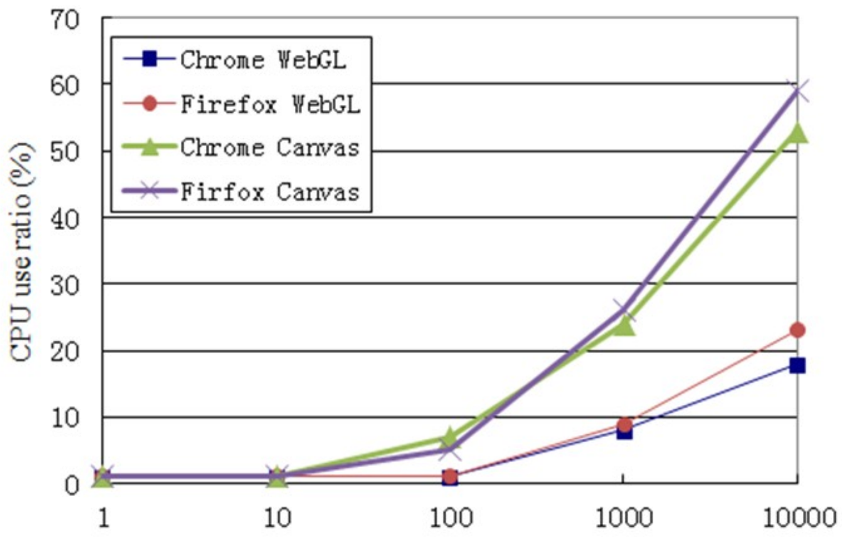
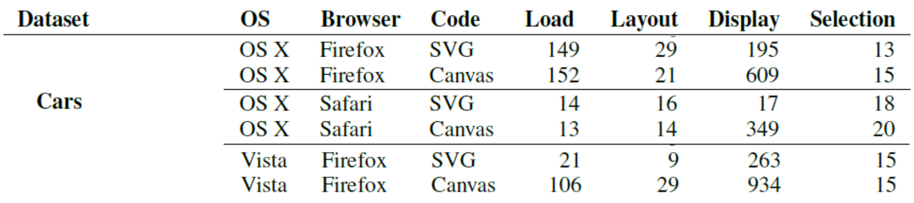


Figure 3.2: CPU use rate from the study performed by Hui et. al. [13]

It has been discussed that the CPU usage may have an impact on the battery life of mobile devices. As the clock frequency increases on the CPU the power consumption increases. Keeping the clock frequency to a minimum is essential for managing the power consumtion. [12,13] Here it can be seen that WebGL has a clear advantage when there is more than 10 sprites.

3.2.10 Canvas vs. SVG

Johnson and Jankun-Kelly(2008) performed a comparison between Canvas and SVG to determine which has the best performance in information visualisation [14]. The result of their comparison is shown in figure 3.3 and 3.4.



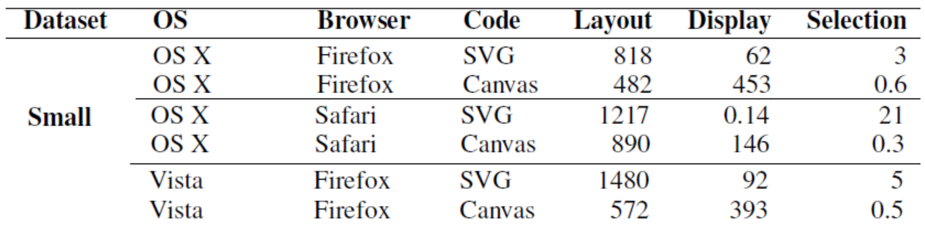
Dataset	OS	Browser	Code	Load	Layout	Display	Selection
Cars	OS X	Firefox	SVG	149	29	195	13
	OS X	Firefox	Canvas	152	21	609	15
	OS X	Safari	SVG	14	16	17	18
	OS X	Safari	Canvas	13	14	349	20
	Vista	Firefox	SVG	21	9	263	15
	Vista	Firefox	Canvas	106	29	934	15

Figure 3.3: Results on car drawing comparison, measured in milliseconds, by Johnson and Jankun-Kelly(2008) [14]

Figure 3.3 shows different performance metrics that describe how the canvas and SVG perform when drawing the cars dataset. The metrics that are used measure

- **layout** - creating an SVG/canvas structure.
- **display** - actual rendering.
- **selection time** - time from click to selected.

Johnson and Jankun-Kelly(2008) also describe how Canvas and SVG perform on drawing more complex images, where SVG also is the slightly better alternative. What can be conducted from this table is that SVG is better at image processing than canvas in this case of creating SVG/canvas structure and rendering. [14]



Dataset	OS	Browser	Code	Layout	Display	Selection
Small	OS X	Firefox	SVG	818	62	3
	OS X	Firefox	Canvas	482	453	0.6
	OS X	Safari	SVG	1217	0.14	21
	OS X	Safari	Canvas	890	146	0.3
	Vista	Firefox	SVG	1480	92	5
	Vista	Firefox	Canvas	572	393	0.5

Figure 3.4: Results on small datasets, measured in milliseconds, by Johnson and Jankun-Kelly(2008) [14]

Figure 3.4 shows performance metric for small datasets and how canvas and SVG performs for this dataset. Johnson and Jankun-Kelly(2008) describes that canvas and SVG both have advantages in different areas. In this comparison SVG is better for performing rendering and canvas is the better option for creating SVG/canvas structure. [14]

3.2.11 Comparing canvas, WebGL and SVG

Canvas, WebGL and SVG has been compared in a study by Andrew and Wright(2014). The comparison is based on five randomly generated datasets, which are measured in dimensions/record. One important factor in this comparison is that the study is only based on visualisation and not on interactivity.

To compare these technologies, WebGL and Canvas uses FluidDiagrams(FD) in 3D respectively 2D and SVG uses Data-Driven Documents(D3). FluidDiagrams is a Javascript framework that can use both WebGL and canvas. FluidDiagram uses WebGL for both 2D and 3D rendering when available and uses canvas when WebGL is not available. D3 is a Javascript library that uses SVG. [3]

Each element created in SVG provides a memory overhead. Canvas and WebGL does not have this issue as it does not retain a model in the DOM. However, this has an impact on the performance of the application. When creating applications that relies heavy on user interactivity, the memory overhead can be ignored as implementing interactivity in WebGL can not be done as effective as when using the DOM. [14]

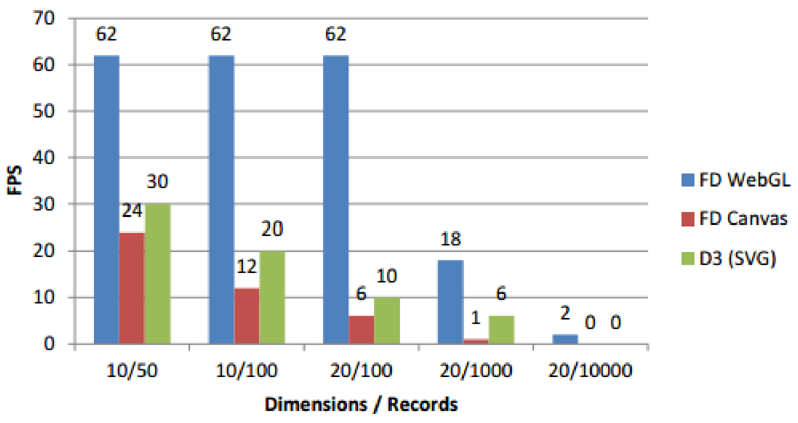


Figure 3.5: FPS for canvas, WebGL and SVG, by Andrews and Wright(2014) [3]

The result of this study shows that WebGL is superior to both Canvas and SVG when it comes to frames per second(FPS). WebGL is the best technique, in this study, for all datasets. The next best option is SVG, which comes in second place for all datasets. [3]

Chapter 4

Results

This section will present the result of the literature study, requirement elicitation and case study.

4.1 Literature study

This section will present the result of the literature study, which aspects of workflows that need to be supported by the chosen technology and which technology that is the best fit for workflow management systems.

4.1.1 Key findings from workflow management systems

From the concepts of workflows there arise some important aspects of workflow management systems:

- **Structurable** - The system must be able to create tasks in a way that allows for a sequence to be created.
- **Re-arrangeable** - Tasks must be re-arrangeable in a sequence, the system must support the ability to move tasks.
- **Visualisation** - It is important to visualise a sequence of tasks and the entire workflow. This importance of clarity make it necessary to zoom and pan the flowchart.

These findings must be supported by the chosen technology.

4.1.2 A comparison of technologies

All of the technologies mentioned in the theory section have their respective advantages and drawbacks, and some are obsolete. VML is not supported in Internet Explorer 10. Flash has issues with security and is no longer supported by Apple. This has led us to ignore both VML and Flash in this study as the application must be supported in future browsers and at least Internet Explorer 11.0.

Using WebGL will result in a fast application that uses less CPU compared to canvas and SVG. One advantages of using WebGL is the cross-browser support, which is important in order to reach out to the most users possible. But cross-browser support does not mean that performance is equal on all platforms. This was an issue in Flash where there were a severe difference in performance between different platforms. The experience of Flash on Mac OS X and Linux did not come close to the the one on Windows. WebGL does however provide a some what equal experience cross-device and cross-browser.

When comparing Canvas, WebGL and SVG, we see that WebGL clearly has the upper hand when it comes to FPS. One factor to the much higher FPS in WebGL is the hardware accelerated graphics, which gives a faster application.

The comparisons in the theory section only handled visualisation and this is only one part of the tool which will be implemented. The key findings of workflow management systems were that tasks shall be structural, rearrangeable and that it shall be possible to display the whole workflow with zooming capabilities. As described, editing is just as an important part and can be the more difficult to implement. Therefore it is also an important part to look at advantages and disadvantages of canvas, WebGL and SVG to see if any of the techniques are more suitable in an editing application. These are the aspects that are most important when further comparing the technologies.

WebGL and canvas are immediate mode API's, where on the other hand SVG is a retained mode API. This is an advantage for SVG over WebGL and canvas, if the developer prefers simplicity over control. Since SVG has the internal representation of the scene integrated into the library, it is easier to use if the representation fits the domain. Using SVG, the developer has to accept how the technology handles internal representation of images, which is not the case in WebGL and canvas.

Looking at advantages and drawbacks for SVG, we can see that SVG has an advantage when it comes to redraws and movement of shapes. This comes from the integration with DOM elements. This is an important feature when it comes to workflow visualisation and editing tools, as described in the key findings. One other important feature is that SVG can handle zoom operations well, which is part of the visualisation finding of workflow management system. The memory overhead disadvantage that is mentioned will not be a problem. This is due to the fact that we are utilizing the built-in event system. As stated earlier it would be less efficient to implement the same behaviour in technologies such as WebGL.

Canvas lack support of events, since this is a desirable feature in the graph-editor this is a big drawback. Text readability is important in our case and is something that is very expensive to perform with canvas. These two drawbacks makes canvas a bad match for the graph-editor.

Most of the advantages of WebGL is concerning 3D. Since this tool will be in 2D, these advantages does not affect this case. WebGL suffers from the same problem as canvas when it comes to interactivity. Moving items around is an expensive operation as the whole scene needs to be re-rendered.

As can be seen by the advantages and disadvantages of each technique, SVG is the most suitable technique for an editing tool. One of the more important feature is the ability to move shapes, which is an easy task to perform in SVG and more difficult to perform in canvas and WebGL. Even if WebGL has better FPS for visualisation, we can not know that the FPS is better in WebGL when it comes to movement of objects and shapes.

4.1.3 Hypothesis

By selecting SVG as the underlying technique we strive to achieve a set of goals. These goals are the result of the feasibility study and reflected in the hypothesis. The hypothesis are formulated as:

1. All of our requirements are possible to implement using SVG
2. The graph-editor will not lose performance when displaying a great number of nodes(according to the domain).
3. We will not be forced to implement our own event model in order to achieve interactivity

In order to implement a graph editor, one must choose a technology which fits the domain. This is required to ensure that the requirement specification can be implemented. The first hypothesis is declared to answer if this is possible.

One of the main concerns when choosing technique, was the great deal of interactivity involved in a graph editing management tool. Hence, it is important that SVG implements an event system that fits our application. The second hypothesis is formulated out of this concern, that our application will not appear to be slow when interacted with. This performance aspect of our application will be controlled by observing the frame rate per second(FPS).

Using an SVG technology ensures that events such as moving objects has support from SVG, allowing developers to implement this functionality relatively easy. This functionality is one of the more important, when developing an interactive tool such as a graph-editor. This is described in the key findings of workflow management systems, where features like moving object are important in order to re-arrange a sequence of tasks in expected order. The third hypothesis is formulated to ensure that this is the case when using a framework that is built on SVG technology for workflow management systems.

The formulated hypothesis will be analysed and confirmed or denied in the discussion section.

4.2 Requirements elicitation

The requirements elicitation resulted in a set of properties which are important to our specific project rather than general to all graph-editors.

- **Active** - Ensures that upcoming bugs are handled in the framework
- **Internet Explorer 11 (IE11)** - A minimum requirement is to support IE11
- **Touch** - In order to support smart devices, touch events must be available
- **Route points in essence curved lines** - Desirable feature to be able to structure the layout of the flowchart

To check if the project still remains active, the commit history was examined to determine if someone was working on the project continuously. External dependencies, support for Internet Explorer 11 and touch support was determined by looking at the projects website and github page. Route points was more of a subjective opinion on how difficult it would be to implement in the corresponding framework, it does not reflect how easy it is to perform in reality.

4.3 Selecting framework

A thorough investigation of existing visualisation frameworks resulted in the following matrix.

	Technique	Active	IE11	Touch	Route points
jsPlumb	SVG and VML	uncertain	yes	yes	maybe
JIT	Canvas	inactive	yes	no	maybe
PixiJS	WebGL and Canvas	active	yes	yes	hard/impossible
Raphael	SVG and VML	active	yes	yes	hard/impossible
JointJS	SVG	active	yes	yes	possible
SigmaJS	WebGL and Canvas	active	yes	yes	possible
VisJS	SVG and Canvas	active	yes	yes	hard/impossible
D3js	SVG	active	yes	yes	possible

From this table two frameworks emerged as the most suitable for the graph-editor, D3JS and JointJS. When deciding between these two, we went with D3JS over JointJS. We did this as the community around D3JS is a lot more active. It is also the most used and there were a lot of examples.

4.4 D3JS

D3JS is a visualisation framework that uses SVG as underlying technology. D3 uses JavaScript to manipulate HTML documents based on data. It uses the same vocabulary as standard technologies such as HTML, CSS and SVG. This is useful as there is no need for a patch to be able to use new functionality in the given standards.

4.4.1 Selections

Selections is the way that D3 handles data. A selection is a set of elements in the document and these can be joined with data. D3 provides enter, update and exit selections and these are the selections that different D3 operations can be performed on:

- **Enter** is a set of placeholder elements, created to bind with new data.
- **Update** is a set of elements that is bound to existing data.
- **Exit** is a set of elements that has its corresponding data removed.

Using these selections allows us to set styles and do computations only on those elements that require it.

4.5 Design decisions

In this section the most important design decisions are described and analysed.

4.5.1 Platform

The requirements specified that the editor should support both tablet and desktop platform, which requires the support of touch events. However the main focus was on desktop platform and getting existing functionality ported from the previous application. This resulted in tablet support to be a low priority putting it at the bottom of our backlog, resulting in never being implemented. The choices we made allow tablet support to be implemented in the future.

4.5.2 Extra library for data-bindings

To avoid writing a lot of code to handle data-binding, which is not central for this thesis, we decided to add a library to handle the view part of our application. After evaluating RactiveJS and ReactJS we decided to go with the latter. This was mainly because of the structure which React provides. Also, Ractive did not feel as easy to use and intuitive as React. The documentation and community surrounding React was also a big factor when deciding which one to go with.

4.6 Key findings from developer diary

This is a summary of the most important findings. They are extracted from the diary produced by the developers during the case study:

- A need for databinding, since D3 is only a visualisation framework and the support for updating data in forms is limited.
- Developers had difficulties knowing which solution was best for creating SVG shapes, HTML or Javascript.
- It would be beneficial to set coordinates to an SVG group. Present solution is to transform the element instead.
- D3 can not be controlled from css entirely, some properties need to be controlled through JavaScript. Placing the styling in two different locations.
- D3 has a lot of support for interactivity, such as movement and zooming. However, a difficult task was to move a SVG object between two different SVG elements.
- The ability to enter an array of data into a D3 selection and later look at the data before setting attributes and other features at no cost. This makes it easy and efficient to make minor changes to the same type of element e.g having dashed lines.
- SVG groups is a central part of using an SVG technology like D3 and is mostly used for grouping SVG shapes together.
- Using enter, update and exit selections can improve performance. Using selections in the correct way will minimize the amount of times the DOM is modified, increasing performance. If not used correctly every element can be altered every time.
- Code structure can sometimes be confusing. Chaining makes it hard to reason about the code and which value that is modified.
- Even though the D3 API, for the most part, is well documented, there are some methods that are lacking complete documentation and thereby are difficult to understand what they return and how they work.
- D3 has behavior for interpolating lines, which was useful when adding route points to transitions.

4.7 Implementation

From the requirement elicitation, details about the implementation emerged. The requirements unveiled three main components:

- Flowchart
- Node
- Transition

There is also an apparent need for some sort of component that can be used to control input of data. We decided to go with a menu.

4.7.1 Flowchart

The requirements state the need of a component that can hold child components of different types. This component is referred to as a flowchart. The flowchart is split up into two different objects. One object holds the data associated with a flowchart, seen in listing 4.1. The other is the graphical representation, which consists of a single SVG element. This is the drawing surface of the application which we will add child elements to, seen in listing 4.2.

```
function FlowchartModel(name, type, description,
    warningTimeout, storageTime){
        this.name = name;
        this.description = description;
        this.createDate = '';
        this.warningTimeout = warningTimeout;
        this.storageTime = storageTime;
        this.type = type;
        this.startNodeId='';
        this.properties = [];
};
FlowchartModel.prototype.constructor = FlowchartModel;
```

Listing 4.1: Flowchart data model

```
<svg id="flowchart"></svg>
```

Listing 4.2: Flowchart graphical model

Adding child elements is done using D3. In D3 you can add items by appending them to a selection. A simple example can be seen in listing 4.3. The returned element is a D3 selection allowing the developer to chain commands.

```
select('#flowchart').append('rect');
```

Listing 4.3: Adding single element

But if you want to append many items and bind them to data there is a more efficient way, this is by doing a join. The code in listing 4.4 describes how to join elements with data in D3.

```
select('#flowchart').selectAll('rect').data(data).
    enter().append('rect');
```

Listing 4.4: Adding items in batch

First we select all rects that are children to svg and bind some data to the selection. The enter() command will then return the enter selection. As described earlier the enter selection is a set of placeholders for elements not having any data bound to them. With append('rect') we will replace these placeholders with rect elements bounded to a specific data item. This process of joining data to elements is used to a great extent in our application. It allows us to react to data and set attributes accordingly using dynamic properties.

4.7.2 Nodes

One of the two main components that a flowchart should be able to contain as a child is a node. Nodes are represented in the same way as a flowchart, with a data model and a view representation. One difference is that we use inheritance to achieve our node hierarchy, seen in figure 4.1. This is used as we have a number of different node types with some common properties.

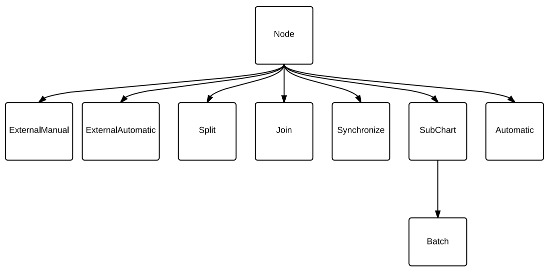


Figure 4.1: Node hierarchy

```
function Node(name, properties, description,
                    warningTimeout, alarmTimeout) {
        this.name = name || '';
        this.description = description || '';
        this.createDate = new Date();
        this.warningTimeout = warningTimeout || '';
        this.alarmTimeout = alarmTimeout || '';
        this.properties = properties || [];
        this.x = 0;
        this.y = 0;
}
```

Listing 4.5: Base type inherited by all node types

The base type in listing 4.5 can be extended as described in listing 4.6 using JavaScripts prototypal inheritance.

```
Automatic.prototype = Object.create(Node.prototype);
Automatic.prototype.constructor = Automatic;

function Automatic(name, plugin, pluginParameters,
                    properties, description,
                    warningTimeout, alarmTimeout) {
```

```
        Node.call(this, name, properties,
            description, warningTimeout, alarmTimeout)
            ;
        this.type='Automatic';
        this.plugin = plugin || '';
        this.pluginParameters = pluginParameters ||
            [];
}
```

Listing 4.6: Example of inheritance

Using inheritance in this way allows us to reuse code and take advantage of polymorphism. An instance of a specific type of Node can now be checked to see if it is an instance of Node, and more specific, an instance of Automatic.

A node's graphical representation is built up by three different SVG elements:

- **g** - a SVG group element
- **rect** - a SVG shape element
- **text** - D3 text element

We build our nodes by creating SVG groups and then appending SVG shapes to that group. We first append the rect element, and secondly, the text element to our g element. By adding the text as the second object we make sure that the text will be placed on top of the rectangle. The end result can be seen in listing 4.7.

```
<g class="node" transform="translate(290,10)">
    <rect class="node-rect" width="40" height="20"></
        rect>
    <text class="node-text" x="1" y="15">Skapa</text>
    <rect class="avatar" width="7" height="7" x="1" y
        ="1" rx="1" ry="1"></rect>
    <text class="avatar-text" x="2.5" y="6.5">A</text>
</g>
```

Listing 4.7: SVG representation of a node

The purpose of building nodes like this is to be able to treat both the text and the rect as one element. The group element allows us to achieve this, as we can add attributes to the g element. In listing 4.8 we see how adding a SVG group containing a node looks like.

```
var node = svg.selectAll('.node').data(data).append('g
    ').attr('class', 'node');
node.append('rect');
node.append('text');
```

Listing 4.8: adding node in D3

The first append in listing 4.8 will return a SVG group which is what the node variable will contain. We can then append different SVG shapes and D3 elements to this group, in order to create node types. Also notice the attr method which stands for attribute. This method handles manipulation of SVG and D3 elements. In this case we set a class name on the group to later be able to recognise that this group contains nodes. The node SVG group can later be used for movement instead of moving every object which is a part of the node.

4.7.3 Transitions

The second component a flowchart should be able to have, as a child, is a transition. A transition is a possible route between two nodes. As seen in listing 4.9, a transition's data model is similar to the other data models, an object holding data.

```
var Transition = function(){
        this.name = '';
        this.description = '';
        this.from = '';
        this.to = '';
        this.type = '';
        this.properties = [];
};

Transition.prototype.constructor = Transition;
```

Listing 4.9: adding node in D3

The transitions graphical representation is a bit more complex due to the requirement to support route points. If we would only have had the need for a straight line between two nodes we could have used the SVG line element. Instead we need to use a path element. The path element is the most powerful of the basic shapes, as it can be used to create all of the other shapes. We use the path element as it can interpolate over a set of dots. This will result in a curved line, which cannot be achieved with SVG line.

4.7.4 Menu

The menu emerged as an essential part of the application. This was a result of the amount of data connected with the different parts of the flowchart, which should be possible to edit. As the nodes have different properties we need to be able to display the fields connected with a specified node, transition or flowchart. This was solved by checking the selected object and determining from that what should be visible. In order to handle this flow of data and control we introduced a second library called React.

We created three different menus called TransitionMenu, NodeMenu and FlowchartMenu.

4.7.5 Route points

Route points provides a way to alter how transitions are drawn in the flowchart. By adding route points to a transition, the transition will need to pass through these. A path is built up by several lines. This gives us a line between the set of points provided. However, this results in very sharp edges. To get a more fluid appearance, we apply an interpolation function, giving us a curvy line.

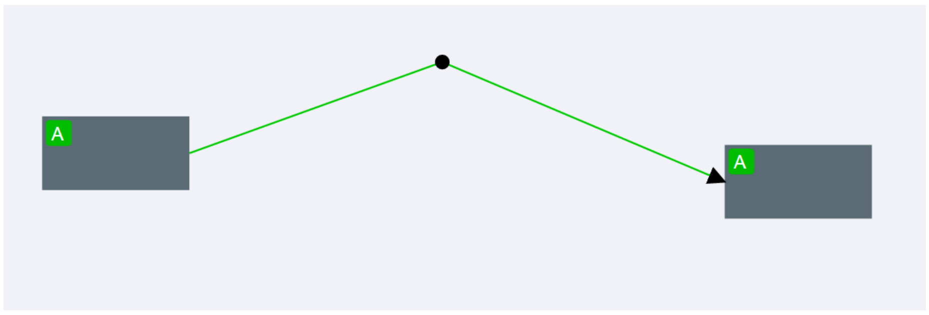


Figure 4.2: Print screen of route point without interpolation function

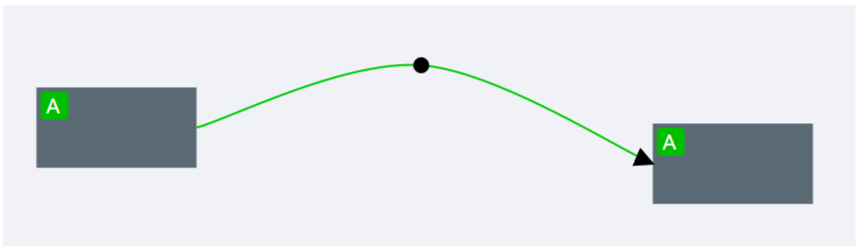


Figure 4.3: Print screen of route point with interpolation function

4.7.6 Event handling

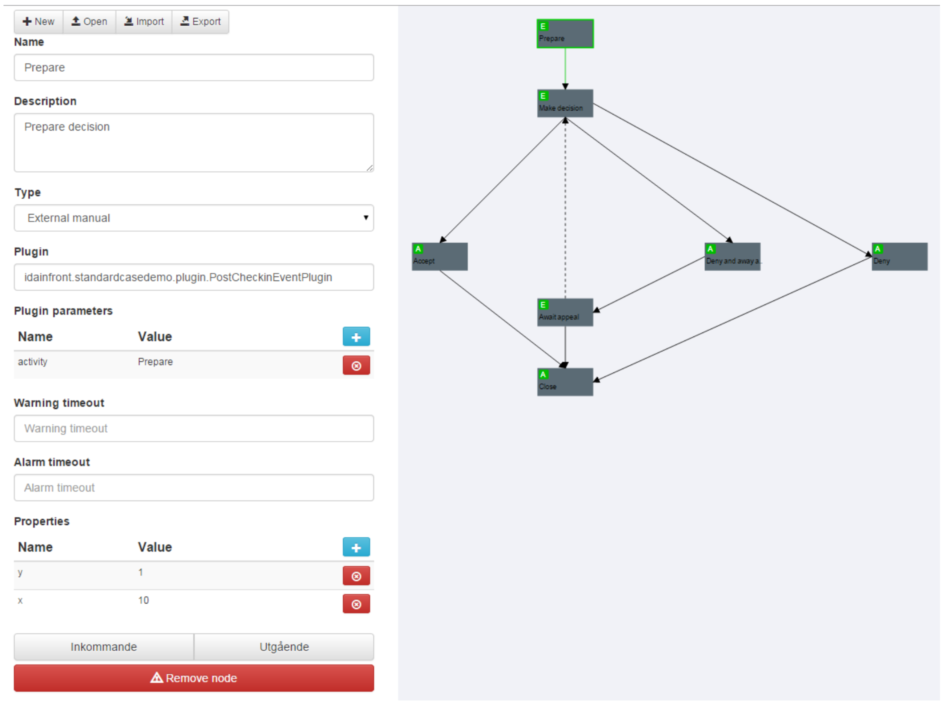
As described earlier there exist a possibility to join data with elements. By doing this we can access the data connected with an element that is involved in an event, in the event function. D3 leverages JavaScripts event system and extends it with its own special events. One of these special events is extra useful in a Workflow Management System, the drag event. Our pre-study shows that it is essential to be able to move nodes around in the flowchart, in order to improve readability. The drag behavior which can be applied using D3 is shown in listing 4.10

```
d3.behavior.drag()
        .origin(function(d) { return d; })
        .on('dragstart', function(){
            d3.event.sourceEvent.stopPropagation();
        })
        .on('drag', function(d) {
                d3.event.sourceEvent.stopPropagation()
                    ;
                d.x = d3.event.x;
                d.y = d3.event.y;
                self.props.onNodeMoved(d);
        })
        .on('dragend', function(d){
                d3.event.sourceEvent.stopPropagation()
                    ;
                self.props.onNodeMoved(d);
        })
```

Listing 4.10: Drag event implemented in the prototype

4.7.7 Graph editor

In figure 4.4, we provide a print screen of the graph editor. This figure displays nodes, transitions and a menu. The selected item is marked with a green border. As a node is selected, the menu represent this state by showing information about the selected node.



Name
Prepare

Description
Prepare decision

Type
External manual

Plugin
idainfront.standardcasedemo.plugin.PostCheckinEventPlugin

Plugin parameters

Name	Value
activity	Prepare

Warning timeout
Warning timeout

Alarm timeout
Alarm timeout

Properties

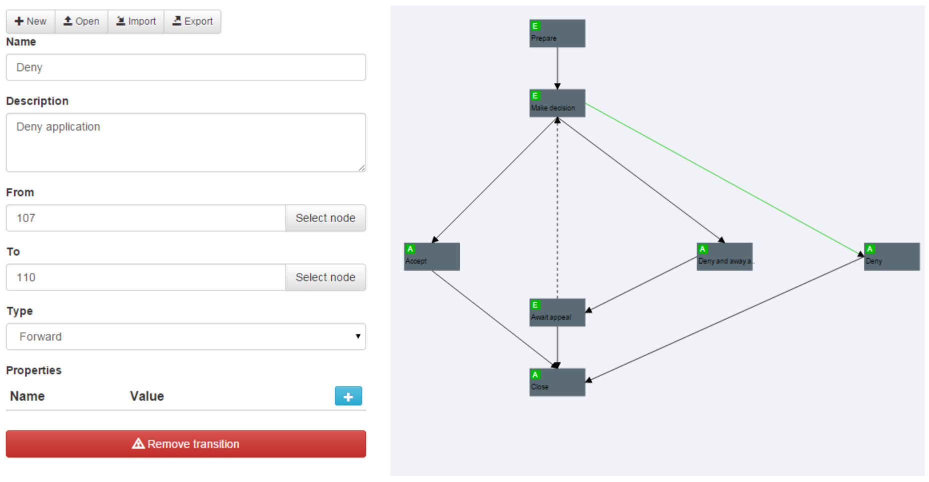
Name	Value
y	1
x	10

Inkommande | Utgående

Remove node

Figure 4.4: Print screen of the graph editor

The menu will always display information about the current selected item. This can be seen as we select a transition, the menu will switch. This can be seen in figure 4.5



Name
Deny

Description
Deny application

From
107 | Select node

To
110 | Select node

Type
Forward

Properties

Name	Value

Remove transition

Figure 4.5: Print screen of the graph editor

Chapter 5

Discussion

This section will analyse and discuss the result provided in the previous section.

5.1 Method

This section will analyse the methods applied in this thesis.

5.1.1 Diary

Using the findings from the diary as empirical findings needs to be motivated by how we have handled the disadvantages of using a diary. One of the disadvantages of using a diary is that it can infer with the normal workflow and lead to an unnatural workflow. This disadvantage was mitigated by only doing observations at the end of the day, such as reflections and important design decision.

Since there are two participants, we get two different versions of the diary. We are thereby given different perspectives on the findings. These versions can be used to, when analyzing, to compare how the perspectives differ. One other approach would be to discuss important findings at the end of each day and later document these. However, this would influence the findings and the different perspectives may be lost.

When writing in the dairy, entries should have been a lot more specific and exhaustive. During the evaluation it was hard to draw conclusions, instead one had to try and remember which issues occurred during the development process. The entries mainly consisted of the work produced during each day. It should have consisted of more reflections and thoughts about the implementation.

There was also a lack of commitment when writing the diary, resulting in poor quality. Then entries were often short and sometimes, stated the same thing several times. From time to time there were also empty entries. The routine was simply none existent.

5.1.2 Triangulation

As was described in the method section, triangulation can be used to make sure that empirical findings are validated and reliable. The empirical findings of the literature study as well as the case study has to be triangulated. This triangulation is performed to make sure that this thesis provide valid and reliable empirical findings as foundation for this thesis.

The literature study is triangulated both through theory and investigator validation. We have achieved theory triangulation as both investigators have found relevant information to the literature study at different sources. These findings have later been compared to check what different source describe about the same topic. By achieving theory triangulation we can be sure that the result of the literature provides accurate and reliable information. We also triangulate our findings in the literature study through investigator triangulation. That is, we have both investigated the same material to give different interpretations of the meaning on the findings in the literature study. This triangulation gives a more accurate interpretation of the result in the literature study.

Now that we know that the literature study provides reliable and accurate information we can use this data as part of the case study. The case study can be triangulated through investigator triangulation. The empirical findings that are triangulated in this case is the diary. The diary can be triangulated using investigator triangulation because both researcher wrote diaries through out the case study, providing different perspectives on important findings during the case study. This gives more valid and reliable empirical findings since its more likely for two researchers reflecting on all important aspects than one.

5.1.3 SCRUM

Using SCRUM during the development has allowed us to adapt to change. We can not say that it has been positive nor negative, as the project has not changed direction during the development process. Our backlog has been consistent throughout the sprints and the order of the user stories has not changed. We could probably have used a waterfall model as well as SCRUM if we had known this at the beginning. But, it is impossible to know which direction a project will go in the beginning, and it is better to plan for change as it is time consuming to change methodology.

The backlog has not been used in the right way as the user stories has not been picked in the order they appear. Instead we have just picked the stories we feel is the most important ones at the beginning of each sprint. This has probably resulted user stories that do not maximize customer value, being implemented. The reason to this could be the lack of a designated product owner. Having a designated member in the team to manage the backlog would probably allow that person to spend more time on ranking and ordering tasks. As time was tight we often overlooked parts of the method, that was more of a administrative nature.

As neither of us have enough experience regarding the rules of SCRUM, none could take on the role of SCRUM master. This lead to no SCRUM master being appointed and that the SCRUM methodology, may not have been exercised in the right way.

Our sprints were not the recommended 30 days, instead we had 14 day sprints, as our total development time was very limited. The short sprints were not recognized as a serious problem. Only impact this had on us was that we had to plan less stories per sprint.

It was not until the third release that we had a working release candidate. This was not a major problem as this was a prototype and not a product. Though it could have been handy at the demo meetings, after each sprint, to have a working release. We also pondered about having a code stop the day before, when no new code is allowed to be merged into the development branch. The purpose with code stops would be to minimize new, potentially error prone, code to be merged with the code base and not noticed until the demo.

5.2 Result

In the results section we formulated a set of hypothesis that was extracted from our pre-study:

1. All of our requirements are possible to implement using SVG
2. The graph-editor will not lose performance when displaying a great number of nodes(according to the domain).
3. We will not be forced to implement our own event model in order to achieve interactivity

5.2.1 First hypothesis

As researchers and developers in this case study, we can confirm that this hypothesis is fulfilled and that SVG is well suited for building visualisation applications depending on a high functioning event system. We draw this conclusion as most of the requirements listed in appendix B has been implemented. Requirements that has not been implemented is due to lack of time and not because a lack of support from SVG.

One other aspect that can confirm this hypothesis, is the key findings from workflow management systems. That is, the graph-editor can structure nodes in a sequence by drawing transitions between nodes, the graph-editor has the ability to move nodes and it is possible to visualise entire workflows.

However there was a need of using a binding framework as well, mostly for visualisation of information related to a graph. This was something that SVG was missing in order to be the complete solution for our domain. However including to much functionality in a framework can also result in overhead. We appreciate the possibility to decide which binding framework to use our self instead of being forced to use some implementation.

5.2.2 Second hypothesis

We discovered quite early that performance was not going to be an issue. We did a simple test by adding a lot of nodes, and it was not until we had around 6000-7000 that we noticed that the program was sluggish. But as our domain is around 100 nodes, we are not even close to the limits of D3.

From the diary we can conclude that when not using placeholders selections enter, update and exit, performance decreases drastically. This conclusion motivates the importance of using the right methods in D3 in order to arrive at the most efficient solution.

5.2.3 Third hypothesis

JavaScript provides an event system which can be connected with SVG elements. This can be conducted, from the diary, as one of the great features of SVG. The event system provides the developer with a way to implement behaviours, like movement and zooming, without being forced to write them from scratch.

As most of the events in D3 is not library, but standard specific. As a result all browsers that support SVG will support D3, when support for selections is added. This entails that it is possible to utilize new events added to the SVG standard without a need to update D3. This would limit the risk if the library would stop receiving updates.

We like D3 as it enhances SVG but not hiding it. It adds the ability to manipulate documents based on data and does it really well.

5.2.4 Low level of abstraction

One issue with D3 was the level of abstraction. As you are working almost directly with SVG, you work in terms of shapes, like circle and line. This makes D3 domain agnostic which has its pros and cons. A higher abstraction level, that is domain specific, would probably decrease the development time. This would in that case be the result of working in terms of nodes and transitions instead of shapes. Our framework study showed that frameworks with this kind of abstraction are not cheep and if they are, they lack tests and the documentation is inadequate.

5.2.5 Performance does not matter

In the beginning of this study there were a lot of focus on the performance of different technologies in different situations. We have discovered that these kind of applications does not use a lot of resources and does not need the most efficient technique. The calculations performed are light and the amount of elements can be considered few. Most importantly there are no spectacular animations performed.

5.2.6 Diary analysis

This section analyse and discuss important findings from the diary.

SVG groups

The graph-editor uses SVG groups in a large extent and mostly for grouping several SVG elements together to define different shapes. However SVG groups does not have a property for global coordinates to the root SVG element. If the SVG element is the only part on display this is not a problem. In our case we have a menu to the left on the screen, leading to that we have to calculate how large the menu is and offset this to the global coordinates on the screen. If our graph-editor is plugged into another system and this system has a left-sided menu on its own, this will lead to more problems.

Visualisation data binding

The data binding functionality provided by D3 has made it easy to connect data with SVG elements. Since the data is connected to the element it is possible to reach this data inside D3 methods. With this ability it is possible to check the data and set attributes accordingly. Reaching data in this way allows us to alter data in a very convenient way, reducing the amount of code.

Menu data binding

Data binding for the menu was not as obvious to obtain with D3 and therefore we decided to go with another binding framework for the menu.

Interactivity

D3JS provided us with a number of events for handling interactivity, such as moving object and zooming. This was one of the functionalities that was vital in the selection of framework and D3JS has proven that these event are easy to implement and requires few lines of code to make them work.

One of the benefits of using an SVG technology is that when zooming, shapes will still be sharp. This is a something that would have required extra functionality in another technology, but not necessary when using SVG.

Chapter 6

Conclusion

As we only worked with SVG in a D3 context, we can only say that pairing these two technologies together has worked well for us. Workflow management systems are a lot about connecting data to a process and visualising it. The initial belief of using efficient technologies being critical to the system proved to be false. We concluded that the key points when choosing technique is more about writing as small amount of code possible by e.g. using a high abstraction.

We have also seen indications that there is a need to be able to apply automated tests, so the technique should have good support for this. Though we have not been able to investigate this leaving it for future work.

6.1 Future work

If more time had been available it would have been interesting to compare a flowchart written in SVG to a identical one written in a framework using canvas or something similar that does not support event handling. The comparison should check how much more code that is written and the performance difference.

Automated tests is a big part of software development. It would be interesting to investigate to which extent you can execute end-to-end(E2E) tests when using different visualisation technologies. There will probably be differences in how easy E2E tests is to performed, depending on if the technique utilises the DOM or not. Being such an important factor this could be a deal breaker when deciding technology.

Bibliography

[1] Wil van der Aalst and Kees Max van Hee. *Workflow management. [Elektronisk resurs] : models, methods, and systems.* Cooperative information systems. Cambridge, Mass. : MIT Press, 2002, 2002.

[2] Mustafa Acer and Collin Jackson. Critical vulnerability in browser security metrics. *Proceedings of W2SP*, 2010.

[3] Keith Andrews and Benedict Wright. Fluiddiagrams: Web-based information visualisation using javascript and webgl. *Eurographics Conference on Visualization (EuroVis)*, 2014. Editors: N. Elmqvist, M. Hlawitschka, and J. Kennedy.

[4] Gary Anthes. Html5 leads a web revolution. *Commun. ACM*, 55(7):16–17, July 2012.

[5] M. Aoyama. Web-based agile software development. *Software, IEEE*, 15(6):56–65, Nov 1998.

[6] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[7] Andrew Begel and Nachiappan Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 255–264. IEEE, 2007.

[8] World Wide Web Consortium. *Vector Markup Language*, 1998 (Accessed: 2015-02-11). `http://www.w3.org/TR/1998/NOTE-VML-19980513`.

[9] Michael Coram and Shawn Bohner. The impact of agile methods on software project management. In *Engineering of Computer-Based Systems, 2005. ECBS’05. 12th IEEE International Conference and Workshops on the*, pages 363–370. IEEE, 2005.

[10] P. De Ryck, M. Decat, L. Desmet, F. Piessens, W. Joosen, T. Aura, K. Jarvinen, and K. Nyberg. Security of web mashups: a survey. Katholieke Universiteit Leuven, IBBT-DistriNet, Leuven, 3001, Belgium, 2012.

[11] Pablo Garaizar, MA Vadillo, and Diego Lopez-de Ipina. Benefits and pitfalls of using html5 apis for online experiments and simulations. In *Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on*, pages 1–7. IEEE, 2012.

[12] Lee Garber. Gpus go mobile. *Computer*, 46(2):16–19, 2013.

[13] Xu Hui, Wei Lihao, Wang Tian, and Luo Xiaoben. Webgl based html5 application performance analyzer. *Journal of Convergence Information Technology*, 7(23), 2012.

[14] Donald W. Johnson and T. J. Jankun-Kelly. A scalability study of web-native information visualization. In *Proceedings of Graphics Interface 2008*, GI ’08, pages 163–168, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[15] Jeremy Keith. *DOM scripting. [Elektronisk resurs] : web design with JavaScript and the Document Object Model.* Berkeley, Calif. : Friends of ED ; New York : Distributed to the book trade by Springer-Verlag, c2005, 2005.

[16] Khronos. *WebGL and OpenGL*, 2009 (Access 2015-02-11). `https://www.khronos.org/webgl/wiki/WebGL_and_OpenGL`.

[17] Khronos. *WebGL overview*, 2011 (Accessed: 2015-02-11). `https://www.khronos.org/webgl/wiki/Getting_Started`.

[18] E.N. Kim, D.P. Schissel, G. Abla, S. Flanagan, and X. Lee. Web-based (html5) interactive graphics for fusion research and collaboration. *Fusion Engineering and Design*, 87(12):2045 – 2051, 2012. Proceedings of the 8th {IAEA} Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[19] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical Softw. Engg.*, 10(3):311–341, July 2005.

[20] Wen Tao Liu. Applied information technology in graphics algorithm implementation based on the web canvas. *Advanced Materials Research*, 908:543–546, 2014.

[21] Bertrand Meyer. *Agile! [Elektronisk resurs] : the good, the hype and the ugly.* [Zurich?], Switzerland : Springer, c2014, 2014.

[22] Microsoft. *Retained Mode Versus Immediate Mode*, Accessed: 2015-02-11. `https://msdn.microsoft.com/en-us/library/windows/desktop/ff684178%28v=vs.85%29.aspx`.

[23] Microsoft. *VML is no longer supported*, Accessed: 2015-02-11. `https://msdn.microsoft.com/en-us/library/ie/hh801223(v=vs.85).aspx`.

[24] Andreas Neumann and Andréas M Winter. Time for svg—towards high quality interactive web-maps. *International Cartographic Association*, 2001.

[25] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 308–308. IEEE Computer Society, 2003.

[26] MP Peterson. Maps and the internet: an introduction. *Maps and the Internet*, 1:1–16, 2003.

[27] Olympus Press. *Vector & Raster graphics in offset printing*, 2009 (Access 2015-02-11). `https://www.khronos.org/webgl/wiki/WebGL_and_OpenGL`.

[28] Stian Reimers and Neil Stewart. Adobe flash as a medium for online experimentation: A test of reaction time measurement capabilities. *Behavior Research Methods*, 39(3):365–370, 2007.

[29] C. Robson. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Regional Surveys of the World Series. Wiley, 2002.

[30] P. Runeson and M. Host. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131 – 164, 2009.

[31] Ken Schwaber. *Agile project management with Scrum. [Elektronisk resurs]*. Redmond, Wash. : Microsoft Press, c2004, 2004.

[32] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng.*, 25(4):557–572, July 1999.

[33] Robert E. Stake. *The Art of Case Study Research*. SAGE Publications, 1995.

[34] Thomas Steenbergen and Michael S Lew. Analysis of using browser-native technology to build rich internet applications for image manipulation. *arXiv preprint arXiv:1101.0235*, 2010.

[35] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review. *Management Information Systems Quarterly*, 26(2):3, 2002.

Appendices

Appendix A

Diary

A.1 First sprint

The diary will consist of todays work and important reflections and major findings regarding the framework and workflow tool.

Bad selection of frameworks in SVG that has the required functions for a flowchart application. Jointjs seems to be a little unstable and may require some rewritings.

Since there are not that many different good options when it comes to flowchart frameworks, it feels like a big limitation to also have to use the best technique.

Changing framework to D3js. Mostly because D3js is a widely used framework and can be seen as well developed framework with quick response to documented bugs on their github.

2015-02-23

Marcus: Today we started our first sprint, consisted of creating modal window for creating flowcharts. Struggled a little with compatibility with IE. Reflections are how node objects will be created and how much objects oriented thinking can be used for these objects. No major findings.

Eric:

2015-02-24

Marcus: Created an node type and realised the importance of interactivity with the nodes to be able to move nodes and transition and will require a lot of work if the technique does not support this. **Eric:** To allow updates in the user interface when the model is changed we need some kind of databinding. D3 is only a visualisation framework, and as such it does not support this feature. D3 does support updating data from forms, but in a more limited fashion. I have investigated different solutions to this problem and settled with RactiveJS. Working with implementing it in the development pipeline.

2015-02-25

Marcus: No implementation regarding framework today. Working on Node types.

Eric: Been working on the menu. The development pipeline is now working.

2015-02-26

Marcus: Been working with drawing nodes. The SVG area moves sideways when clicking on screen. Also the get bounding box of text element does not handle characters that go under the writing line, like j g etc. can be solved with either padding or chosing another method for calculating text width and height. //side note: this was solved with checking text length instead of text width and therefore not using boundingbox.

Eric: Been working on the menu. Been looking at how to work with D3 and templates. Question is if it is better to represent d3 elements as templates and manipulate attributes through moustaches or to create them in JavaScript. Moustaches is a HTML template engine supported by RactiveJS

2015-02-27

Marcus: Continued drawing nodes and have been looking at if drawing them in javascript or html with two way binding is the best way. **Eric:** Menus needs to be tested in order to determine if they work. Creating the view part of nodes etc. is placed in templates to separate view and logic. This direction is still being evaluated, it is still unclear if D3JS can be leveraged in this way. Events seems to be smooth to implement.

2015-03-02

Marcus: Creating nodes in html with two way bindings and tied it up with the menu bar. Two-way binding makes it easy to draw many nodes with a small amount of code.

Eric:

2015-03-03

Marcus: Started working on moving nodes in flowchart. Since we use a svg group element that contain both node(SVG rect element) and text(SVG text element) it gives some difficulties. The limitation with SVG group is that it does not have coordinates, but can be translated to different locations in the parent group. The benefit of having relative coordinates in the group is that it becomes easier to place e.g. node text.

Eric: The ability to use css together with SVG seems severely limited. Issues with Ractive, does not work as good as expected, problem with performance is already present. The structure of the code is also not that intuitive.

2015-03-05

Marcus: Adding functionality to move nodes. Making nodes move was relatively easy task to perform, since d3 has event that handle movement of nodes and therefore easier to implement movement. One difficulty is to truncate text to the size limitation of the node box. Since text is drawn in an svg text element it does not support all common css tricks for this task. One other issue that had to be addressed was to get coordinates relative to the flowchart, which has some offset for menu bar.

Eric: In order to handle the issues with performance I have elaborated with ReactJS. It seems already more robust and measurements points at an increase in performance. This can however be a result of some errors in the implementation in RactiveJS as it was somewhat unclear how to structure the code and implement some parts of the program.The result of switching framework seems to be positive.

2015-03-06

Marcus: Been working on append text to nodes with SVG and adding all different types of nodes to flowchart. appending text works well when used in svg group so text is relative to the node instead of the flowchart. Events in D3 also works well with click events on different elements.

Eric: Implemented functionality in react. The structure is much better. Trouble with dragging nodes form a menu onto the drawing area, context menu is used instead. The issue was dragging nodes from one SVG surface to another, seamless. This was determined to difficult for us.

A.2 Second sprint

2015-03-09

Marcus: Changed binding framework from ractive to reactjs to get a better structure. This causes some problems when wanting to render javascript code instead of HTML, which is necessary for nodes since events with drag and drop are easier to perform when elements are rendered through javascript. **Eric:** Implemented the context menu in react and other parts.

2015-03-10

Marcus: Struggling with learning structure of new framework together with d3js. The problem when drawing nodes in HTML when trying to create composable html object. **Eric:** Issues with D3JS and composing reusable objects was discovered. The ability to encapsulate nodes into objects and then create them by running ’new Node’ was limited.

2015-03-11

Marcus: Learned some of reactjs and was able to develop the same functionality regarding drawing nodes as with ractice, only offset problems left. Think D3 works well with appending data array for to later append SVG groups, also very easy to call drag behavior for nodes. **Eric:** Worked with offset issues when adding nodes through the context menu. The coordinates must be relative to the upper left corner of the SVG surface and not the window. This may require some computation to be done if not available as a property.

2015-03-12

Marcus: Behavior for nodes. Needs to delete all nodes before drawing all nodes, result of sending D3 all node data for every draw and just not the new nodes. The was later solved with enter and exit selections. **Eric:** Worked with implementing menu functionality. Discovered problems with D3 and removing nodes. It seems as if we need to remove and create the element tree to accomplish it.

2015-03-13

Marcus: Worked with adding and removing properties in the menu. **Eric:** Worked with integrating our project into wicket and caseapp.

2015-03-16

Marcus: Worked with changing properties and plugin parameters. So have been working with react js and nothing with d3. Using HTML events for changing from text to input field to be able to edit text. React js has a nice data flow with event handling for keys which was used today. **Eric:** Worked with parsing XML files described with chartdescription.

2015-03-17

Marcus: Mostly work regarding layout with bootstrap for key-value input field. Also added text to node, where it seems that not all nodes and all text for nodes can be appended in the same statement, which is the result of what an append returns. In this case a text element cant be appended on a rect and this is one of the reasons why svg groups are used. **Eric:** Worked with parsing XML files described with chartdescription.

2015-03-18

Marcus: Finished working on properties and plugin parameters and then worked on rewriting some code structure. Also looked at transitions, where it seems like events on a circle cant stop dragbehavior for dragevents on the same svg group. So dragging a circle which is located in a SVG group that has a dragging behavior results in dragging the circle, even this is tried to be prevented in events on the circle. **Eric:** Worked with parsing XML files described with chartdescription.

2015-03-19

Eric: Worked with wicket and publishing of resources so that charts can be accessed through a HTTP GET.

2015-03-20

Marcus: Worked with zooming, which according to example from the API should be in a overlaying SVG group with in SVG groups beneath. D3 handles zooming so not much work needs to be done to implement the zooming behavior. **Eric:** Worked with testing integration with wicket. Collecting data from inside of React using jquery AJAX calls was easy.

A.3 Third sprint

2015-03-23

Eric: Fixed import from server. It uses a mock API during production. Have also looked at different ways of limiting pan and zoom.

Marcus: Worked with bugs regarding zooming behavior in D3js. One issue regarding D3js zooming behavior is that one using zoomend event, every zoom event can only last 50 ms. Which leads to difficulties when knowing when a zoom operation is finished.

2015-03-24

Eric: Worked with transitions. Encountered implications with react and updating location of a node. In order to move the position of the transition there is a need to update the node data on drag. This will cause the update of state to be called a lot during drag, causing an update of data in D3JS.

2015-03-25

Eric: Worked with transitions. Using enter, update, and exit selections drastically increases performance. Discovered that if used incorrectly the application tends to be very slow. The code is often hard to interpret and it is not always clear that the same statement can mean different things depending on the order. Data the is only updated stays in the DOM and it is up to the developer to limit the amount of updating to keep the application responsive. **Marcus:** Finished zooming functionality. Using d3 linear scale to restrict zooming window and calculate current zoomed window by using domain function on x and y coordinates. It requires some calculations when want to get the upper right corner of the zoomed screen to be able to translate the zoom function. The zooming behavior is somewhat lacking in documentation and specially the domain() function, which required a lot of thinking to figure out and is still not perfectly clear. But the domain function and the zooming function can be used to calculate the current zoomed window.

2015-03-26

Marcus: Worked on drawing transitions from XML import and placing transitions based on angles. There was some confusion when the y-axle went downwards, causing the angle to be inverted for the y-axle. **Eric:**

2015-03-30

Marcus: Worked with creating new transitions, adding menu for transitions. The ability to use functions with datum for operations on selection is very often useful and is a nice feature in d3js. **Eric:** It is now possible to create new transitions. The ability of looking at the data before setting attributes and other features at no cost, makes it easy and efficient to make minor changes to the same type of element. E.g having dashed lines,

2015-03-31

Eric: Worked with re-factoring large parts of the d3 code. The problem with not being able to keep all styles in css has forced us to extract some into a globals object and some into the CSS. This separation of information is not desired but unfortunately necessary. **Marcus:** Started looking on route points, mostly laborating regarding how to interpolate lines.

2015-04-01

Marcus: Worked on route points. Arrived at the conclusion that svg paths will be used to interpolate lines. Paths are a line generator which is used to interpolate over points and create curved lines. paths are part of svg.

Easy to implement in jsfiddle, harder to implement in the application. **Eric:** Problems when trying to add knees to transitions. The knees are stored as an array under the transition object. We want to add them directly to the flowchart. Adding them together with the transitions would force us to use a group and the coordinates would have to be relative to the group. But adding them to the flowchart would force us to iterate over every transition and pick out the knee data before adding them.

2015-04-02

Marcus: Came across some issues when adding route points to the flowchart, where the transitions hold the data that we want to use to add route points, e.g. a circle. However we cant append a circle on the transition selection since this selection is of type path. This could be solved by iterating over every transition again and append circle to all route points. However this seems unnecessary and will instead be solved by only drawing route points for a transition once it has been selected.

A.4 Fourth sprint

A.5 Evaluation

Appendix B

Requirements

1. The system shall be responsive
2. The system shall have two modes, view and edit
3. The system shall support Internet Explorer 11.0
4. The system shall have a rightside menu
5. The rightside menu shall contain all the different node types
6. The rightside menu shall switch to another view displaying information about the selected item, when selecting an item
7. As a user I shall be able to zoom the flowchart by scrolling in on a location
8. As a user I shall be able to zoom by pinching when using a mobile device

B.1 View mode

1. As a user I shall be able to move nodes
2. As a user I shall be able to filter out information in the flowchart
3. As a user I shall be able to view flowcharts
4. As a user I shall be able to locate the route the courier has taken
5. As a user I shall be able to keep track of where the courier is located at the moment

B.2 Edit mode

B.2.1 Flowchart

1. As a user I shall be able to create a flowchart with the following properties
   (a) name, string, not empty
   (b) description, string (optional)
   (c) create-date, date (optional)
   (d) warning_timeout, integer (optional)
   (e) alarm_timeout, integer (optional)
   (f) storage_time, integer (optional)
   (g) start-node-id, string, not empty (optional)
   (h) type, case or broker
2. As a user I shall be able to change the following parameters of the flowchart
   (a) name
   (b) description
   (c) type, between case and broker
3. As a user I shall be able to add nodes to the flowchart
4. As a user I shall be able to add transitions to the flowchart
5. As a user I shall be able to remove nodes from the flowchart
6. As a user I shall be able to remove transitions from the flowchart
7. As a user I shall be able to move nodes
8. As a user I shall be able to connect nodes with a transition
9. As a user I shall add a minimum of one node
10. As a user I shall be able to save the flowchart to disk
11. As a user I shall be able to save the flowchart to server
12. As a user I shall be able to upload a saved flowchart from file to server
13. As a user I shall be able to open file from disk
14. As a user I shall be able to import flowchart from server
15. As a user I shall be able to create a new flowchart

16. As a user I shall be able to add a timer event to the flowchart with the following properties
    (a) name, string, not empty
    (b) event-type, either of
        i. close
        ii. open
        iii. flush
    (c) when, string, not empty
    (d) period, string, not empty
17. As a user I shall be able to structure the nodes in swim lines
18. As a user I shall be able to move to a subchart by first selecting a subchart node, then select go to subchart in the leftside menu

B.2.2 Nodes

1. As a user I shall be able to create a node of type Automatic with the following parameters
    (a) name, string, not empty
    (b) description, string (optional)
    (c) chart-id, string, not empty (backwards compatability)(optional)
    (d) create-date, date (optional)
    (e) plugin, string(fully qualified class name)
        i. plugin-parameter, key-value pair (optional)
    (f) warning_timeout, type integer (optional)
    (g) alarm_timeout, type integer (optional)
    (h) properties, key-value pair (optional)
2. As a user I shall be able to create a node of type ExternalManual with the following parameters
    (a) name, string, not empty
    (b) description, string (optional)
    (c) chart-id, string, not empty (backwards compatability)(optional)
    (d) create-date, date (optional)
    (e) plugin, string(fully qualified class name) (optional)
        i. plugin-parameter, key-value pair (optional)
    (f) warning_timeout, integer (optional)

   (g) alarm_timeout, integer (optional)
   (h) properties, key-value pair (optional)

3. As a user I shall be able to create a node of type ExternalOther with the following parameters
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)
   (d) create-date, date (optional)
   (e) warning_timeout, integer (optional)
   (f) alarm_timeout, integer (optional)
   (g) properties, key-value pair (optional)
4. As a user I shall be able to create a node of type Split with the following parameters
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)
   (d) create-date, date (optional)
   (e) warning_timeout, integer (optional)
   (f) alarm_timeout, integer (optional)
   (g) properties, key-value pair (optional)
5. As a user I shall be able to create a node of type Join with the following parameters
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)
   (d) create-date, date (optional)
   (e) warning_timeout, integer (optional)
   (f) alarm_timeout, integer (optional)
   (g) properties, key-value pair (optional)
6. As a user I shall be able to create a node of type Synchronization with the following parameters
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)

   (d) create-date, date (optional)
   (e) warning_timeout, integer (optional)
   (f) alarm_timeout, integer (optional)
   (g) properties, key-value pair (optional)
   (h) event-listener, string
   (i) event-type, either of
      i. close
      ii. open
      iii. flush
   (j) status, string (optional)

7. As a user I shall be able to create a node of type SubChart with the following parameters
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)
   (d) create-date, date (optional)
   (e) plugin, ”subChart”
      i. plugin-parameter, key-value pair (optional)
   (f) warning_timeout, integer (optional)
   (g) alarm_timeout, integer (optional)
   (h) properties, key-value pair (optional)

8. As a user I shall be able to create a batch node with the following properties
   (a) name, string, not empty
   (b) description, string (optional)
   (c) chart-id, string, not empty (backwards compatability)(optional)
   (d) create-date, date (optional)
   (e) plugin, ”batch”
      i. chart, string, not empty
      ii. count, integer, integer
      iii. timeout, integer (optional)
      iv. plugin-parameter, key-value pair (optional)

9. The system should set SubChart as type by default on Batch nodes

10. As a user I shall be able to change the following properties on a automatic node:

   (a) name
   (b) description
   (c) warning_timeout
   (d) alarm_timeout
   (e) type of node
   (f) key-value pair of properties
   (g) plugin
   (h) key-value pair of parameters

11. As a user I shall be able to change the following properties on a ExternalManual node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties
    (g) plugin
    (h) key-value pair of parameters

12. As a user I shall be able to change the following properties on a SubChart node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties
    (g) plugin
    (h) key-value pair of parameters

13. As a user I shall be able to change the following properties on a ExternalOther node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout

   (e) type of node
   (f) key-value pair of properties

14. As a user I shall be able to change the following properties on a Split node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties
15. As a user I shall be able to change the following properties on a Join node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties
16. As a user I shall be able to change the following properties on a Synchronization node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties
17. As a user I shall be able to change the following properties on a Batch node:
    (a) name
    (b) description
    (c) warning_timeout
    (d) alarm_timeout
    (e) type of node
    (f) key-value pair of properties

    (g) plugin
    (h) key-value pair of parameters

18. As a user I shall be able to add a retry policy to a node
19. As a user I shall be able to add intervals, with count and interval, to the retry policy
20. When specifying the subchart in a subchart node there should be a possibility to create a new flowchart
21. When specifying the subchart in a subchart node there should be a possibility to select an existing flowchart

B.2.3 Transitions

1. As a user I shall be able to create transitions between two nodes of type Forward with the following properties
    (a) name, string, not empty
    (b) description, string (optional)
    (c) create-date, date (optional)
    (d) from-node-id, string, not empty
    (e) to-node-id, string, not empty
    (f) properties, key-value pair (optional)
2. As a user I shall be able to create transitions between two nodes of type End with the following properties
    (a) name, string, not empty
    (b) description, string (optional)
    (c) create-date, date (optional)
    (d) from-node-id, string, not empty
    (e) properties, key-value pair (optional)
3. As a user I shall be able to create transitions between two nodes of type Back with the following properties
    (a) name, string, not empty
    (b) description, string (optional)
    (c) create-date, date (optional)
    (d) from-node-id, string, not empty
    (e) to-node-id, string, not empty
    (f) properties, key-value pair (optional)

4. As a user I shall be able to change the following parameters of a transition
    (a) name
    (b) description
    (c) type of transition
    (d) key-value pair in properties
5. As a user I should be able to add key-value pair to properties
6. As a user I should be able remove key-value pair if any exists in properties
7. As a user I shall be able to add route points to a transition