Designing Interaction and Visualization for Exploration of System Monitoring Data

A design-oriented research study on exploring new ways of designing useful visualizations and interaction for system monitoring data using web technologies

JONATAN DAHL

Master’s Thesis at CSC
Supervisor: Anders Lundström
Examiner: Kristina Höök
Company: Spotify AB
Supervisor: Niklas Ek

2014-06-23
Abstract

System monitoring is a practice that is frequent within companies providing digital products to consumers and is a common way to help developers contribute to a good end-user experience by ensuring a high availability and good performance of the product.

This thesis is a design-driven exploratory study on designing interaction and visualization for system monitoring data, using web technologies. The design space spans over interaction design and technical domains, exploring system monitoring data interaction and visualization from an HCI perspective as well as technical possibilities and limitations of the web platform.

An artifact embodying new ideas and design visions regarding the topic is created in close collaboration with the target users. The artifact expresses possible and potentially valuable inventions regarding exploration of system monitoring data. It also emphasizes the close relationship between system monitoring and physical space and how the interaction with it can provide a useful sense of place to the data.

Technical insights and good practices regarding developing performant data visualization user interfaces is also presented and motivated, where two methods providing different strengths and weaknesses are described.
Contents

1 Introduction ...................................................... 1
   1.1 Purpose and goal ........................................ 2
   1.2 Problem definition ...................................... 2
   1.3 Scope and delimitations .................................. 2
   1.4 Definitions ............................................. 3

2 Background ...................................................... 5
   2.1 System Monitoring ....................................... 5
   2.2 Software Reliability Engineering - SRE ................. 6
   2.3 About Spotify .......................................... 6
       2.3.1 System Monitoring ................................. 6

3 Theory .......................................................... 9
   3.1 Data and Information Visualization ...................... 9
   3.2 Exploring Data .......................................... 10
       3.2.1 Data exploration techniques .................... 12
   3.3 Research and design methods .......................... 15
       3.3.1 Research through design ....................... 15
       3.3.2 Rapid prototyping ............................... 17
   3.4 Space and place in HCI ................................ 17

4 Method .......................................................... 19
   4.1 Research and design process ........................... 19
       4.1.1 Current and preferred state ..................... 19
   4.2 Target users ............................................ 20

5 Pre-study ...................................................... 21
   5.1 Introduction ............................................ 21
   5.2 Method .................................................. 21
       5.2.1 Brainstorming sessions ......................... 22
   5.3 Results ................................................ 22
   5.4 Discussion and Conclusions ............................ 23

6 Design phase .................................................. 25
Chapter 1

Introduction

There are many important factors that affect the success of a digital product and one of them, that has gotten more and more important over the last years, is the user experience. Furthermore, there are a great deal of factors that influence the user experience, and one of those factors is the performance of the product. For example, an unusually long delay of the response to an action performed by a user is likely to affect the user experience negatively.

One way to help ensure a good performance of a digital product is to monitor it in different ways. By monitoring it, developers can obtain important information that helps them keep the product performing well. This can be done using tools that collect and visualize different metrics in the product, usually related to software or hardware performance. Alerting developers in times of a severe software or hardware failure is also common.

Spotify has a large amount of services (software) and servers (hardware) that make up the internal infrastructure of their music streaming service. A service is a piece of software that usually handles the logic behind a specific feature, such as playing songs, searching for a song or handling playlists. A server is a piece of hardware on which these services run. One service can be run on multiple servers for increased performance and capacity.

The concept of monitoring services and servers like this is referred to as System Monitoring (IBM, 2013). At Spotify, there is an elaborate internal infrastructure built for this. There is software, or tools, used by certain developers to collect, store and visualize different sorts of data from these services and servers to help them gain valuable information about how they are performing, which enables them to find and solve potential or existing problems. This, in turn, helps them ensure a high availability and good performance of the Spotify services, thus contributing positively to the resulting end-user experience of the Spotify product.

System monitoring can hence be summarized as being a toolbox to help developers ensure a high availability and good performance of a digital product, and this design-driven thesis investigates new ways for developers at Spotify AB to explore system monitoring data.
1.1 Purpose and goal

Within the context described in the previous section, the purpose of this thesis is to explore visualization and interaction with large volumes of system monitoring data, having the developers at Spotify who are using monitoring tools as the target audience.

The goal is to produce a design artifact (Zimmerman, Forlizzi, & Evenson, 2007), to create new knowledge about what is possible to do regarding data exploration within system monitoring at Spotify AB. A second aim is to gain insights about technical limitations and opportunities given the choice of technologies and platform.

1.2 Problem definition

It can be difficult to find the right information quickly in large volumes of data, especially if the type of information sought for is unknown. If it is known, tools can be specifically designed to help the user extract that type of information from the data. However, if it’s unknown, a better approach might be to design a tool in a way that lets the user explore the data more freely and thus have a better chance of finding useful information within it (Keim, 2002). How a tool like this should be designed is not completely obvious though. The problem is under-constrained, and different designs need to be explored and tested to come up with useful solutions to it.

1.3 Scope and delimitations

The produced design artifact is a functional web application prototype that aims to lead to discoveries of new information regarding the subject of this thesis, and not to be released as a polished product for widespread use at Spotify AB.

The application will be using modern web technologies and run in modern web browsers but there will be no evaluation of cross-browser compatibilities or implications using legacy web browsers or technology.

Huge volumes of data being tracked and stored within companies today is often referred to as big data, being defined as ‘Big data is a term describing the storage and analysis of large and or complex data sets using a series of techniques including, but not limited to: NoSQL, MapReduce and machine learning.’ (Ward & Barker, 2001). In my personal experience, big data is often associated with business intelligence analytics with the purpose of creating valuable insights for strategic decision-making processes. Though technologies like MapReduce and machine learning occur within system monitoring practices, it’s not within the scope of this thesis and will not be a part of the project. This thesis focuses on the HCI aspects of exploring data and not the computational side.
1.4 Definitions

Spotify service
The Spotify service refers to the product that is exposed to the end-users; the desktop or mobile client where private individuals may log in using their Spotify account and play music.

Spotify services
The Spotify services are the internal software that together make up the Spotify service; the software that runs on one or multiple servers in the Spotify back-end, that handles logic such as authenticating users or delivering songs to the clients.

Spotify
Spotify or Spotify AB refers to the company and not the product.
Chapter 2

Background

This chapter provides explanations of central concepts and definitions required to fully understand all parts of the report. Terms like System Monitoring and Software Reliability are explained, both in general and in the context of Spotify.

2.1 System Monitoring

Digital products with millions of active users, like Spotify’s music streaming service, can have a large and complex back-end\(^1\) consisting of a huge amount of servers with numerous pieces of software running on them that communicate with end-users and each other. Monitoring all of those servers and systems closely helps developers to maintain them efficiently and have the product perform well, contributing positively to the end-user experience.

“System Monitoring refers to collecting key system performance metrics at periodic intervals over time” (IBM, 2013)

Usually, monitored metrics range from low-level metrics like CPU\(^2\) performance, memory management or read-write operations on the disk to high level metrics such as the amount of connected users, number of failed user logins or number of searches performed by a user, etc. Metrics like these can be visualized as time series, to show their change in value over time, or trigger alerts if their value falls below or exceeds a certain threshold. Alerts are meant to instantly notify developers when anything that could critically affect the products performance or availability occurs, for example issues that could have economical consequences or have seriously negative impact on the end-user experience. The alerted developers are then responsible for resolving the issue as quickly as possible.

---

\(^1\)The internal infrastructure of hardware and software that a digital product consists of

\(^2\)Central Processing Unit: handles most of the calculations in a computer
2.2 Software Reliability Engineering - SRE

Within technology departments there are often certain developers whose main responsibility is to ensure that the service that the company provides is available to the end-users. These developers are often frequent users of system monitoring tools. It’s important that they have access to good tools that help them find the right information quickly so that they can find and fix problems that might affect the availability of the service. They’re usually referred to as Software Reliability Engineers (Geraci, 1991).

"On-call" A developer who is on-call will be instantly notified when an alert is triggered. They are usually on-call for a specified period of time. Being on-call means that they might be awakened at any time of the day, including in the middle of the night. This type of alerts should only be triggered when something really severe occurs.

2.3 About Spotify

Spotify is a Swedish company providing a music streaming service to private individuals. Spotify was founded in 2006 and launched in 2008 and has since grown to a global service having over 10 million subscribers, over 40 million active users and acting in over 56 markets in the world (Spotify, 2014).

The mission Spotify has is to change the way people listen to and access music, by changing the way the music industry and its licensing has worked in the past. What used to be stored on physical media and could only be acquired in record stores can now be accessed instantly from anywhere in the world over the Internet, using the Spotify service.

The company has grown a lot since 2006 and so has the complexity of its product, which increases the need for effective system monitoring solutions.

2.3.1 System Monitoring

At Spotify, there is a team of developers responsible for providing an infrastructure of system monitoring tools and services. They have developed a monitoring pipeline that collects metrics, stores them and allows for retrieval of them in visualization interfaces. The target users for the monitoring team are other developers at Spotify, especially Software Reliability Engineers.

Exactly what metrics are being collected by these monitoring tools are decided by their users. They track the metrics they want - on the servers and software they work with - and send them into the monitoring pipeline for storage and then retrieval in the visualization interfaces, or monitoring interfaces, provided by the monitoring team.

In other words, the monitoring team maintains an infrastructure that enables other developers to easily track metrics that are important to them, and visualize
them in web based user interfaces. These interfaces often consist of line graphs
displaying time series, and offer interaction to allow for some data exploration. The
monitoring team constantly works on improving the monitoring infrastructure and
the visualization tools. This thesis explores new ways of interacting with and visual-
izing data in tools like these. The designs being explored are independent from the
existing visualization tools but depends on the back-end monitoring infrastructure,
i.e. the existing data collection and storage solutions.
Chapter 3

Theory

This chapter describes theory behind data visualization, data exploration and interaction techniques that have been used in the development of the prototype. The ideas behind the design-oriented research method used for this thesis is also explained.

3.1 Data and Information Visualization

The purpose of visualizing data is to help us gain insight, to acquire new information and knowledge that we wouldn’t be able to acquire by observing raw data. By looking at diagrams or charts we create mental interpretations and draw conclusions from the visualized data. The visualized data in itself is not telling us anything directly, but with the help of how it is represented visually, we are able to gain insights from it (Spence, 2007). In this way, how data is visualized will probably affect our ability to draw conclusions from it, and that’s one of the areas to be explored by this thesis.

Spence (2007) further proves the point that the result of observing charts and diagrams is created entirely in the mind of the viewer by demonstrating a few examples of data visualizations. Today, it might be easy to assume that data visualizations are results of calculations done by computers. However, computers are rather a tool that makes it easier to create the visualizations. Figure 3.1 is a classic example within data and information visualization history. It’s the french engineer Charles Joseph Minards visualization of Napoleons march to, and retreat from, Moscow. The visualization is quite self-explanatory and gives the viewer a lot of information about the expedition without having to spend a considerably higher amount of time to read about it in a book, or similar. And it’s easy to remember.

Another interesting example is one where a cholera epidemic in 1800’s London was effectively stopped thanks to insight gained from correlating data, which was made possible as a result of the way the data was visualized. Figure 3.2 shows a map over the Soho district in London. By establishing that the deaths from cholera were clustered around one of the water pumps, which was made possible with the
visualization, the epidemic could be stopped.

The process of visualizing data can be split into 4 phases. (1) capture the data, (2) prepare the data, (3) process the data and (4) visualize the data (Schroeder & Noy, 2001). The first step happens in the start of the monitoring pipeline at Spotify, where developers set up what metrics to track and send them into the monitoring pipeline for storage. Step 2 and 3 is partially handled by the monitoring infrastructure when retrieving the data from a monitoring interface. This project focuses on the visualization step and to some extent on the processing step.

Thanks to the advancement of digital technology we are now able to explore data and gain information using methods that weren’t available before the information technology era. It’s nowadays possible to interact with data in real time to increase our ability to gain insights from it, using exploration techniques such as filtering, linking and brushing that I will explain in the following sections.

### 3.2 Exploring Data

The amount of data tracked and stored at companies today, including Spotify, can be vastly large. The bigger the volume of data, the more complex it may be to interpret; to find valuable information within it. If it’s hard to understand the data, calculating complex algorithms - for generating insights and valuable information - might not be such a good approach. Instead, it’s could be more valuable to design solutions that allows humans to explore the data and use their perceptive skills to mine information from it (Keim, 2002).

The process of exploring data is, according to Keim (2002), also a hypothesis generating process. The exploration leads to insights that change the hypothesis.
Figure 3.2. Map of London's Soho district in 1845 showing locations of cholera deaths and water pumps. (Tufte, 1983)
iteratively, or create new hypotheses. He also states that human exploration of visualized data often leads to a higher confidence in the validity of the findings.

Shneiderman (1996) phrased the "Information Seeking Mantra" as: *Overview first, zoom and filter, and then details-on-demand.* When exploring data, the user should be first presented with an overview of all the data. The user can then, using interactive tools, zoom the visualizations and filter out irrelevant data to find interesting patterns. When the user has succeeded in filtering out a subset of the data that is the most interesting to their current hypothesis or exploration goal, the user can request additional details about this data.

### 3.2.1 Data exploration techniques

Simple and low dimensional data can often be easily visualized and explored using basic line graphs, x-y plots and histograms. However, more complex data with many dimensions may need different approaches to be able to be efficiently explored by a human user. Keim (2002) classifies data exploration techniques for complex and large data sets based on three criteria:

**(1) The data to be visualized**

The dimensionality in data is decided by the number of variables to be found. For example, comparing different car models could mean that each car has variables like mileage, horse powers, weight, construction year and more. This data is considered multidimensional and requires other types of visualization and interaction techniques to be able to be effectively explored.

Some data may be of the relational type where different pieces of the data, called "nodes", have a relationship to one or more other nodes. Examples are e-mail conversations among people or the file structure of a hard disk.

**(2) The visualization technique**

In addition to 2D and 3D visualizations such as line graphs, bar charts and x-y-z plots, there are some other visualization techniques that can be used to make sense of complex data that is of higher dimensionality or relational/hierarchical.

When visualizing and exploring high dimensional data, *parallel coordinates* is a good solution (Figure 3.4) for better understanding the data or to find interesting variable correlations or patterns. The principle of this technique is to display all the dimensions (variables) parallel to each other and have lines connect all variables that belongs to the same entity. As in the car example mentioned earlier, each line would be a car.

For hierarchical data, the *treemap* is a useful visualization solution to allow for exploration of the data. The treemap is a rectangular area subdivided by every level of the hierarchy, where each child node are grouped inside their parent node. The treemap was created by Ben Shneiderman (Shneiderman, 1992).
To give further meaning to the visualized data and communicate a certain dimension, different encodings such as color, position or size can be applied (Iliinsky & Steele, 2011).

(3) The interaction and distortion technique

Interaction is a crucial component of data exploration. Not only does it provide the steps necessary for the "Information Seeking Mantra" mentioned in the previous section, but it fills the gap between them (Keim, 2002). Interaction allows the users to actively explore the data by manipulating the visualizations in real time, allowing for finding relations and correlations within the data. There are a couple of commonly used interaction techniques for exploring data. Among other, there are brushing, linking, zooming and filtering:

**Brushing**

Brushing is to change encoding in one or more items as response to a user interaction with another element (Spence, 2007). In figure 3.5 it’s shown by changing the non-selected items into a gray color, to make them appear dimmed and help the user focus on the chosen set of elements.
Linking

A useful way of making sense of large multidimensional data sets is to split them up into multiple visualizations, that each represents fewer dimensions, and link those together. The linking is an interaction method where a users action in one component is reflected in another, in a way that is meaningful for the user. An example is highlighting a subset of the data in one component that will highlight the same data in the other components (Spence, 2007). Figure 3.5 demonstrates this.

Zooming

When dealing with large amounts of data, zooming is a good way of being able to both get an overview of the data and at the same time being able to drill down do explore the details on a low level.

Filtering

Similar to above, when exploring large volumes of data it’s important to be able to partition it and investigate interesting subsets of the data. One way of doing this is by using the brushing technique to allow the user to filter the data in real time.
3.3 Research and design methods

Given the exploratory and design-oriented topic of this thesis, a methodology using a model for design research is applied. The execution relies on a combination of the design-oriented research model and rapid prototyping.

3.3.1 Research through design

The term design has previously been more common in HCI practice than in HCI research, and has often been associated with usability engineering; the process of modeling requirements and specifications for user needs and shaping the product accordingly (Zimmerman et al., 2007). In recent years, design has found its way into
HCI research and Zimmerman et al. (2007) has developed a model describing and motivating the role of design within HCI research and how to evaluate its outcomes.

In this model, they highlight some of the strengths of interaction designers way of working and how that can contribute to both HCI research and HCI practice. One of the biggest strengths of designers is their ability to tackle under-constrained, or "wicked", problems - complex problems with conflicting ideal outcomes, opposing stakeholder goals and many other unknown or complicated variables that makes it hard to define one optimal solution (Rittel & Webber, 1973). The way designers do this is by continuously re-framing the problem by iteratively creating and critiquing design artifacts that aims to transform the world from its current state to a preferred state (Zimmerman et al., 2007), eventually proposing a solution to the under-constrained problem.

They describe 5 ways interaction designers contribute to HCI research and practice: First, in their process of finding a preferred state they create opportunities and motivation for research engineers to develop new technologies that hasn’t previously existed. Second, the created artifacts embodies their ideas, which facilitates the communication to the HCI practice community that can bring them to life in commercial products. Third, they facilitate for the HCI research community to engage in under-constrained, ‘wicked’, problems that are difficult to tackle using traditional methods. Fourth, interaction designers make research contributions using their biggest strength; re-framing problems by iteratively trying to design the right thing. Fifth, they motivate the HCI community to discuss the impacts the design artifacts and ideas might have on the world.

Evaluation criteria

Since the design oriented way of performing research differs from traditional ways within HCI, so does the evaluation of the outcomes. Zimmerman et al. (2007) outlines the evaluation criteria as follows:

Process
In interaction design research, there is no expectation that repeating the process will yield the exact same results. Instead, the process will be judged as a part of the whole research contribution and needs to be thoroughly described and motivate all rationales and decisions.

Invention
The result of an interaction design research project must be something new, that contributes something novel back to the HCI research community. To demonstrate this, a literature review must be made to situate the result and prove that it’s an advancement of existing knowledge. By elaborately articulating the invention, the details about the invention is communicated to the HCI community, guiding them on what to build. This directly relates to the second type of contribution that interaction design researchers can provide to the research community mentioned earlier.
Relevance
In traditional research methods, validity is one form of evaluation of the results. This is typically done by either benchmarking the performance increase of the new solution or by disproving the null hypothesis.

In the research through design approach, this is not always applicable. There is no way of proving that the solution is the right one. Two designers following the exact same method are highly unlikely to come up with the same results.

Instead, the relevance is evaluated. Relevance is positioning the contribution in the real world and evaluating its impact. For example, is this useful? Does it transform the world to a better state? The "better", or "preferred", state must also be elaborately articulated and there needs to be motivations on why it is better.

Extensibility
The final criterion for evaluating an interaction design research contribution requires that the community are able to build upon the outcomes. Either by applying the methods used on future research or by leveraging the knowledge produced by the resulting artifacts. The research needs to have been documented well so that this is possible.

3.3.2 Rapid prototyping
Research through design might be considered to implicitly include rapid prototyping methodologies, but to further clarify the principles and benefits of the method, it’s here described in more detail.

When rapidly prototyping a software, the development process is done iteratively and in close collaboration with the target users. Features are delivered incrementally and the requirements change over the course of the development process according to the continuous feedback gathered from the users (Luqi, 1989).

The motivation for the use of this method is the lack of exact requirements and knowledge about the artifact that was going to be produced. Rapid prototyping allows for exploration of requirements and features iteratively and minimizes the risk of errors or bad design decisions thanks to the close involvement of the users.

3.4 Space and place in HCI
The later stages of the project lead to the discovery of a connection to some concepts that exists within interaction design, the notion of space and place. The metaphor of space has been used a long time by interaction designers to design user friendly systems (Harrison & Dourish, 1996). A typical example is the desktop metaphor when designing user interface for computer operating systems (such as Windows, Linux and OS X).

The concept of place shares properties with space but is also different from it in a couple of ways. While a space is a three dimensional physical room, a place is all
that but with some additional properties. Place is defined by how it relates to its context and surroundings and by how we adapt and appropriate the space it exists in. A house is a space but my house is a place.

The point that Harrison and Dourish (1997) make is that it is actually the notion of place that frames our behavior when interacting with systems, especially the ones providing collaborative features, and not space, as previously believed. While the application created in this project is not collaborative in the same sense as the systems they are describing, it’s however interesting to view how system monitoring is closely tied to physical space and how interacting with this application gives these physical spaces a sense of place.
Chapter 4

Method

This thesis follows a design-oriented research methodology and is initiated with a theoretical enquiry on relevant theory for this project that is combined with a contextual pre-study followed by a design phase, which are described in detail in their own separate chapters following this chapter. The design driven approach of the thesis follows the model of Zimmerman et al. (2007) of how interaction design contributes to human-computer interaction research, as described in their concept of Research Through Design.

4.1 Research and design process

The project starts with a pre-study to map out the existing system monitoring tools at Spotify to understand what their purpose are and how they are being used by the developers. A study is also made on the practice of system monitoring in general and why it’s important for companies with digital products of this size. This is combined with an inquiry about data visualization with an emphasize on data exploration. Guidelines and good practice for how to design interaction for data exploration and how to visualize data are investigated. The pre-study also results in possible starting points for the development av a functional prototype.

This is followed by a design phase where a design artifact is produced using ideas and knowledge generated from the pre-study. The artifact is rapidly prototyped and iteratively tested on the target users. The prototyping process does not include lo-fi and hi-fi sketches but instead real development of a web application directly from the start, the motivation for this being the requirement of using live monitoring data for the purpose of design exploration.

4.1.1 Current and preferred state

In accordance with the Research Through Design method this thesis applies, the current and preferred state of the world must be articulated and motivated so that
it is possible to determine the relevance of the design artifact, the purpose of the
artifact being to shift the state from current towards the preferred.

**Current state**

At Spotify AB, there are some ways of visualizing and interacting with system
monitoring data.

**Preferred state**

At Spotify AB, there are new and useful ways to visually explore and interact
with system monitoring data.

### 4.2 Target users

The main users of the monitoring tools at Spotify are the so called *Software Reliabil­ity Engineers*, which makes them natural to have as target audience for this
project as well.

The fact that this audience possesses deep knowledge about the system moni­
toring and back-end architecture of Spotify influences the design of the prototype as certain assumptions can be made regarding how they will perceive elements of
the interface and how they will interact with it. There is no need for extensive ex­
planations on what data is available and how to interpret, for example, time series
that are generated from it, as they are already used to working with tools providing
that type of presentation.
Chapter 5

Pre-study

5.1 Introduction

Given the open ended nature of this thesis, the project started with brainstorming sessions to explore interesting challenges and opportunities regarding designing for system monitoring.

The mission of the monitoring team is to enable developers\(^1\) to quickly identify and fix technical issues, which is why that was made the goal of this project as well. This is especially important in times of serious technical issues that might have a severe impact on the user experience of the Spotify service, such as user being completely unable to play music, that could lead to economical consequences for the company.

The aim of the pre-study was to explore and define the current situation regarding system monitoring at Spotify AB; to dive and look into how things work, what tools exist, how people work and what the monitoring infrastructure looked like, and eventually find good starting approaches to the project.

5.2 Method

Three meetings were held during the first few weeks, where I and three developers brainstormed interesting problems and challenges related to system monitoring. A goal with these meetings was to understand was system monitoring really meant and what possible opportunities there were for proposing new and useful design ideas.

I also reviewed and tested the existing system monitoring tools being used within Spotify AB to help me understand basic concepts about system monitoring and how the developers use these tools.

\(^1\)Service Reliability Engineers
5.2.1 Brainstorming sessions

The participants in the brainstorming sessions were three developers at Spotify AB:

- **Developer 1**: Front-end web developer in the monitoring team. Designs and builds interfaces for the monitoring tools. Has been at Spotify for 1 year.

- **Developer 2**: Back-end developer and team leader for the monitoring team. Has been at Spotify for 3 years.

- **Developer 3**: Web developer and team leader for all front-end web developers. Has been at Spotify for 1 year.

5.3 Results

By studying the existing monitoring applications I learned how developers used them and what the objective of using those tools were. Typical use cases are: investigating ongoing technical issues, *post-mortems* and capacity planning. *Post-mortem* means to investigate a technical issue after it’s been resolved, to determine what happened, why, and what knowledge can be gained from it to know how to prevent it from happening again in the future. Capacity planning is to ensure the systems have sufficient memory and processing power, etc, and plan for upgrades.

The brainstorming sessions generated three major ideas on possible design explorations, all of which would provide features missing from the current monitoring solutions:

1. **Sharable graph dashboards**
   
   The first idea that emerged from the brainstorming sessions was to design an application where developers could create their own dashboard of monitoring graphs. These dashboards would then be able to be shared among other developers. The value of this would be that developers would create dashboards for certain troubleshooting cases, i.e. for a specific type of problem that might occur not only once. Then other developers would be able to more quickly troubleshoot a problem by reusing an existing dashboard that someone else created for a similar problem, and thus be able to efficiently solve the problem.

2. **Graph annotations**
   
   A second idea was to translate real life interaction into digital by observing how developers interact with and discuss monitoring graphs in person and then design a solution that would allow the same type of interaction digitally. This can be described as a distributed digital model of real life interaction that would not only potentially enable larger scale collaboration between developers but also persistent storage of the communication.

---

2 A collection of visualization were each visualization would show relevant system monitoring data.
(3) Health overview of services

The third idea was to design a solution that would enable developers to more effortlessly get an overview of the overall status of all the services in the Spotify infrastructure. This solution would rely on defining one or a few specific metrics that would define the "health" of a service; whether it’s functioning normally or not. This would provide developers with insights regarding the overall "health status" of the entire infrastructure and could possible lead them to earlier discoveries of potential system failures.

The last idea gained most interest among the developers and was decided to be the starting point for the project.

5.4 Discussion and Conclusions

The brainstorming sessions and the review of existing tools provided the knowledge required to be able to start exploring useful design solutions for data visualization and interaction. By knowing how the tools worked and how the developers used them, along with what the developers considered was missing from the current solutions, the project was able to start transitioning into the design phase and the actual creation of a design artifact.

Commonly across all the ideas were the challenges of providing developers with the right information at the right time, and enabling that to happen during a short amount of time. This is especially challenging when the type of information sought for is unknown, which supports the idea of designing a solution that allows developers to explore through a "top-down" approach, as suggested in design proposition 3.
Chapter 6

Design phase

6.1 Introduction

Following the pre-study, the development of the prototype started with the initial idea to provide a service health overview that would help developers to more quickly identify potential system failures, or knowing where they had occurred. However, this eventually proved to have some difficulties tied to it and the prototype design changed direction. The new form was a data exploration tool where information could be correlated to the physical location of their sources. This also opened up for the exploration of the HCI concepts space and place and their representation in the prototype.

6.2 Method

The prototype creation required quick development of a functional application that could be directly tested on the target users, as well as decisions about what the data visualization process would look like from a technical point of view. No analogue lo-fi or hi-fi sketch prototypes were made, the digital application logic was developed directly from the start. The reason for this was the benefit of being able to use the system monitoring data instantly and have the interaction and visualization with it tested on developers from the beginning.

As part of the rapid prototyping, informal and spontaneous user testing was performed continuously to gather feedback to help making new design decisions. These user sessions consisted of demonstrating for, in total, 5 service reliability engineers, 2 monitoring engineers and 5 web developers the latest progress of the prototype and gather their feedback on what features that could potentially improve its usefulness.
Figure 6.1. Server and services overview (treemaps). Visualizing size hierarchy between different grouping of servers and services.
6.3 Results

The first version is the visualization of the back-end infrastructure in a summarized overview, the interface of this version can be seen in figure 6.1. It comprises three treemaps showing different hierarchical groupings of the servers and services of the Spotify back-end, where the size is measured in number of servers they consist of. The topmost treemap shows the size relationship of the data centers$^1$ across the world. The middle treemap shows the size relationship of the pods, which are sub groups in each data center, and the treemap at the bottom shows the size of the internal services.

The method behind defining service health was to pick one or a few key metrics that could be considered to hold all information about how any service is performing. Since all the different services function very differently, its health is also defined differently. There was, however, one metric that was believed to be enough for defining the health of any service. That was latency$^2$. If the latency would heavily diverge from a "normal" value, it would signal an anomaly that would represent 'bad health' for the service.

However, I and the target users soon came to the conclusion that it could still vary heavily from service to service, which would make it more difficult to create one definition of service health for all services. The same amount of latency divergence could have different amount of impact on different services. We also realized that the action of designing a tool that, by itself, judges the data - i.e. decides whether the data is represented something 'good' or 'bad' - puts an immense responsibility on the designer of the product. There were also speculations about the possibility to involve machine learning in this design to programmatically compute and tweak different threshold values for this metric, or for the choice of metric, but it was soon discarded as being too out of scope for this project. This lead to the decision of going for a more objective design that instead would allow for the user to judge the data and for the application to present the data unbiased.

The first manifestation of this new direction of the prototype is shown in figure 6.2. It consists of a line graph (top) that is able to show a system performance metric over the period of one week, starting from now on the right hand side and spanning to one week ago on the left hand side. Below it is a treemap showing all the servers, their geographical placement and their internal groupings at each location. The interaction consists of users being able to query any metric stored in the monitoring pipeline$^3$ from a text input field and then interact with it by using the mouse to draw a rectangular selection in the line graph that is reflected in the treemap.

During one of the informal user testing sessions, facts about a previously, for me, unknown data source in the Spotify back-end that held more information about the servers appeared. This new data source made it possible to add another grouping

---

$^1$A collection of servers in a physical location
$^2$The time interval between a request and response of an action in a service
$^3$See section 2.3.1
Figure 6.2. Final prototype, early version. A line graph (above) and a treemap (below) displaying a system performance metric (a time series spanning one week) and the physical grouping and geographical placement of the data sources.

dimension to the treemap. The new treemap also shows how the servers are grouped in racks which heightens the resolution of the physical placement information (see figure 6.3).

6.3.1 Description of the Final Prototype

The final version can be summarized as an objective and exploratory tool where the users can explore the system performance data of the Spotify back-end as time series, and correlate that data to the physical location of their sources. The final user interface is shown in figure 6.3, it consists of two visualizations that are interactively linked together by the input of the user: a line graph and a treemap. Details about the final prototype are explained in the next section.

Line graph

The line graphs retrieves temporal system monitoring performance data from a data source that exists within the monitoring back-end at Spotify. The data it retrieves is processed by the application and then rendered in the web browser onto the screen. The processing consists of transforming the incoming
data structure to a form that is compliant with the functions that calculates the screen coordinates of the data points that make up each line.

After some experimenting with different methods of rendering large amounts of lines in the browser the *canvas*-method was chosen, which is more performant than the *svg*-method for a large volume of data points (see section 7.3.2 for an explanation of the two methods).

The line graph shown in figure 6.3 is showing user activity over one week. The peaks and dips signifies day and night cycles. Different groupings of lines can be observed in the line graph, and some of them are off phase with others. This signifies their ties to different physical locations, which connects them with different time zones.

**Treemap**

The treemap visualizes all the servers in the Spotify back-end, the way they are grouped, the relationships of the groups and their geographical and physical placement. It retrieves its data from a source within the Spotify back-end, processes it and renders it in the browser. The processing of the data includes transforming the structure of the incoming data to a hierarchical structure that is appropriate for the functions that calculates the sizes and positions of the rectangles that make up the visualization.

**Interaction**

As previously established, interactivity is a fundamental criteria when designing for data exploration. The interactivity in the final prototype consists of a real time linking of selections in both visualizations. The user can brush (select) a subset of the data in the line graph and the sources of the selected data will be highlighted in the treemap (figure 6.4 and 6.5). This allows for exploration of interesting patterns in the line graphs and their correlation to physical grouping and geographical placement of their data sources. For example, selecting a peak in one group of lines and then selecting a peak in another group of lines that is off phase with the first one would inform the user that the first group of lines belong to servers and services located in Europe while the other are located in North or South America.

Following the 'Information Seeking Mantra', the treemap offers details-on-demand when the user hovers the mouse cursor over a specific server (figure 6.6). That action provides the user with information about the name of the server and groups it belongs to, which the user can use in other monitoring tools to gain more knowledge about that server and the situation being explored.

### 6.3.2 Technologies used

This prototype was built using web technologies, these included HTML5 (W3C, 2014a), CSS3 (W3C, 2014b) and JavaScript (ECMA International, 2011). To sup-
Figure 6.3. Final prototype. A line graph (above) and a treemap (below) displaying a system performance metric (a time series spanning one week) and the physical grouping and geographical placement of the data sources.

Figure 6.4. Final prototype. The selection made in the line graph is linked and mirrored in the treemap.
port the creation of this static web application, two JavaScript frameworks were used: d3.js (d3.js, 2013) and Angular.js (angular.js, 2014).

Application and Platform

The prototype is developed as a static web application hosted internally at Spotify. A static web application is a website that is not using a server for computations. The application is downloaded in its entirety and run solely in the web browser. The reason for this architecture is that all functionality needed for the server-side (back-end) already exists internally at Spotify. That includes different web servers providing APIs from which it is possible to retrieve different kinds of monitoring data. The prototype uses those APIs to retrieve its data for the visualizations.

Angular.js Angular.js is a popular JavaScript framework maintained by Google to create application logic, such as navigation between different views and handling data transfer between back-end APIs and the visualizations.

The reason for the choice of this framework was it’s solid and structured application logic functionality and that it’s well established within the static web application community.
Visualizations

The visualizations are built using a combination of JavaScript, HTML5 and CSS3. JavaScript for computations and processing of the data retrieved from the monitoring back-end (See 2.3.1), and the overall application logic regarding visualizations. HTML5 for structure and CSS3 for presentational aesthetics. The use of these technologies allows for rich interactivity of the visualizations, which is the fundamental base for exploring data.

D3.js  D3.js is a powerful JavaScript framework to craft custom data visualizations. There are other frameworks that come shipped with finished components such as line graphs, bar charts and other standard types of diagrams, but d3 is a set of tools and functions to allow developers to design their own custom charts and diagrams, which can range from simple line graphs to more advanced multi-dimensional visualizations.

This was chosen as a visualization framework because of its flexibility in allowing developers to create custom visualizations. A framework like this is ideal for exploratory design-oriented projects.
6.3.3 Value and usefulness

The goal throughout the process has been to create something useful for the target users, and especially exploring what could be useful, given the current state of the design artifact but also what it could become, which is one important factor to consider when evaluating design outcomes from interaction design research. Based on the feedback from the users, there are some value propositions for this prototype in its current state:

Capacity planning

All servers, or nodes, in a large back-end like this carries different amount of network traffic, or load. This needs to be evenly distributed among them to function as efficient as possible, and support a good user experience without latency and delays. By visually seeing the geographical grouping of servers and what services they are connected to, and a specific metric value at a certain time, it would be possible to determine whether some places are over- or under-provisioned regarding computational power and network capacity, etc. For example, a graph showing a group of servers having all their bandwidth used up - and they’re all in the same location - could mean that a certain group of users will experience latency problems in the near future, which in turn could lead to them being unable to log in or play songs.

Post-mortems

When a serious issue has occurred, developers investigate what happened, and why, to learn from it and prevent it from happening again. This is referred to as post-mortem. By correlating a certain metric with the geographic placement of their sources (servers) it would be possible to gain insight about the cause of the problem, if its somehow related to the organization, grouping or interrelationships between the servers.

6.4 Discussion and conclusions

6.4.1 Process

The form of the prototype changed continuously throughout the design process. Ideas that initially seemed useful and interesting later proved to be either too difficult to implement or not as interesting anymore. This is the strength of rapidly prototyping a design artifact to create knowledge within a given context; the problem is constantly re-framed and better and more interesting results are yielded.

There were no specifically planned milestones or organized user testing sessions throughout the design phase. The designer (me) being constantly physically present next to the target users allowed for spontaneous but effective user testing of the
prototype and ideas. I believe that this ease of interacting with the target users contributed positively to the evolution of the prototype.

### 6.4.2 A sense of place

The final application prototype allows for looking into closed rooms, into physical spaces located far from where the observer is. It’s a digital, simplified representation of physical spaces; servers organized in racks that are placed in large server halls in different locations around the world.

Bringing back Harrison and Dourish (1996) ideas regarding space and place in interaction design, I would not argue that there is a sense of placeness that frames the interaction and user behavior in the prototype in its current state, according to their definition. However, if one would envision a state where users are able to appropriate and adapt this virtual space together through their interaction by contributing persistent data and content of their own, generated from their behavior and usage of the prototype, a placeness that more closely resembles their definition could be created. One usefulness of this could be found in how this could influence future behavior of users by potentially helping them solve problems more quickly by relying on relevant content created by previous users.

The design of the prototype has been influenced by spatial properties of the real world, such as grouping and hierarchy of the servers. Additionally, another type of placeness can be observed in the prototype through the interactive linking of the visualization components. The relation between the physical locations and the temporal data is what differentiates these virtually represented physical spaces from each other and adds a dimension of place to it. This place information can be useful for developers when searching for information in the data in the way that different metric values have different meaning depending on the relative time and location of their sources. The significance of place contextuality of the data can be observed by envisioning a scenario where all servers that handles user logins are located in the same geographical location, and a power outage occurs, then no users are able to log in. However, if the servers are instead distributed over several locations, the traffic from users affected by an outage in one location could be transferred to another location that isn’t affected.

Paradoxically, these spaces are still represented spacelessly, there is no notion of distance between different servers or server groups nor between data centers across the world. Distance has meaning when sending digital information between physical locations as it affects the time it takes to reach its destination, digital messages are transferred very fast but not instantaneous. However, in this visual representation, the distance has been completely abstracted away. This might not necessarily be preferable, but its likely not vitally affecting the quality of the exploration experience of the user. This topic could be further investigated in another research project though.
Chapter 7

Discussion

Throughout the process of this thesis, insights were gained regarding the design process and the final design artifact, as well as the role of space and place within system monitoring. Those findings are discussed and evaluated in this chapter.

7.1 Research and design process

A strength I observed with the research through design methodology is how the design is shaped together with the target users; how it defines the feedback and design decisions loop. Instead of having discussions about possible or impossible features and trying to envision what might be and demand constructive and creative feedback from the users based on something that doesn’t yet physically exist - that is just an idea - it’s instead possible to present the physical creation and they can see it for what it is, and understand the possible opportunities and concepts of it. This way it’s easier for the users to give feedback on the product. To communicate an idea to someone is much easier and more effective when showing it, embodied in a physical object, than using verbal language explaining its features and properties and expecting them to fully understand the scope of the vision.

Zimmerman et al. (2007) points out that it’s also important to not only consider the final result as something "final", but rather as something that has created insight and knowledge in the process and might become much more, given further iterations and tests. This is very accurate regarding the prototype created in this thesis. Knowledge regarding technical implementation solutions and interaction with and exploration of system monitoring data has been gained. Additionally, many ideas and suggestions about features have been expressed throughout the process but has not been manifested in the prototype. But can, however, potentially be implemented in the future.

My interpretation of the preferred state, that Zimmerman et al. (2007) stresses as an important part of design research in their model, is that it is something to aim for but not necessarily achieve by all means. Both for the reason that the knowledge
generated in the process is that which is of value and because I believe that the *preferred state* is impossible to achieve. The reason being that it’s not possible to define it in a complete and absolute manner to be able to actually validate whether it’s been achieved, nor the design artifact. In the case of this project, the design artifact has definitely shifted the current state towards the preferred state but it hasn’t completely fulfilled it. Not even including all the possible extensions and visions of what it could be. However, I think the goal has been achieved through the generation of new ideas and opportunities for what is possible to do regarding exploration of system monitoring data at Spotify AB.

### 7.2 Data and place in system monitoring

Another interesting discovery generated from the final prototype is how it emphasizes that system monitoring is tightly linked to the physical world. The final prototype presents an interface for visually correlating data to the geographical and physical placement of the sources. What has been created is a virtual *space* connecting physical *places*. The prototype enables users to look into closed rooms in different places of the world, concurrently. The feat to link temporal system performance data to the physical placement of their source in this way is new within the scope and context of this thesis; system monitoring at Spotify AB. Based on Zimmermans et al. (2007) criteria for evaluating design artifacts in design research, this is the *invention* criterion of the design.

This feature is also what adds a dimension of *place* to this visual representation. The linking of selections in the performance metric data to the server data gives the representations context in the form of geographical and physical location as well as time and grouping which adds more meaning to the data. For example, certain patterns in the data have different meaning whether it’s day or night, whether their sources are located in Europe or America or whether the servers are grouped in the same rack. All of this is likely to have meaning when correlated to other patterns in the system monitoring data. Examples of this is, as mentioned earlier, that user activity patterns are heavily tied to their geographical location as well as time of day. Or when a piece of software running on multiple servers, the servers should be as physically separated and distributed as possible to minimize the risk of software failure in case of technical hardware issues that could affect multiple servers if they are physically grouped tightly together.

### 7.3 Technical evaluation

The web as platform imposes a couple of technical restrictions. Most notably, computational power, both for processing data and visualizing data. There are many different choices that can be made to affect the performance of the prototype. The implications of the different options are explained as well as which ones are best suited for what situation:
7.3.1 Processing data in the browser

Processing large volumes of data requires strong computational power. A web browser on a single agent running a JavaScript application is limited in resources, compared to the large cluster of servers collecting and storing the data in the Spotify back-end. Therefore, the data received to the web browser should be pre-processed, preferably down-sampled using a sensible algorithm, so that the volume received can be processed in a reasonable amount of time by the browser. There is no point in having more data points than the amount of pixels the screen is able to render. This time variable is very likely an important factor affecting the user experience of the application.

7.3.2 Rendering in the browser

Visualizing, i.e. rendering, large volumes of data in a web browser will also require significant amount of computational power. A sensible thing to do in the processing stage, in the browser, is to further down-sample the data so that the amount of data points available in one axis is not higher than the amount of pixels in the pixel range required to display it on the screen.

Two techniques were used for the rendering of data. The first one uses multiple svg elements in the web browser and the second one uses one canvas element in the browser. Both of which are official html5 elements. These two techniques pose different strengths and weaknesses:

**SVG:** The technique of rendering visualizations using svg elements means producing a very high amount of html elements on the page. Each html element rendered on a web page takes up memory and processing power, and a large amount of them will eventually lead to very bad performance, which in turn creates a bad user experience. So, using svg elements for a high volume of data has a high risk of leading to bad performance of the web browser.

However, the benefits of using svg is that they are easy to implement interaction for since web browsers allow for very easy attachment of different input listeners to every html element, or node, it contains. Therefore, adding interactivity - such as listening to click or drag events on the mouse - to an svg visualization requires relatively little work.

**CANVAS:** Visualizing using canvas differs from svg in the sense that there is only one element being rendered in the browser. There’s only one canvas element, as opposed to the svg method where all visualization elements are comprised of many different svg elements.

The canvas is one element only containing pixels in different colors, which is much cheaper performance-wise for the browser to render. So in the processing phase, all screen coordinates of the visualizations are calculated on the canvas and painted as a pixel with a certain color.
The disadvantage of canvas though, is that it’s much harder to implement interactivity to the visualizations. Because the canvas is just a 2-dimensional area containing painted pixels, instead of an abstract tree of html nodes. There is no way to attach any input event listeners to it, as it would be when having many html nodes as in the svg method.

It’s still possible to implement interaction with the visualization, though it needs a lot more work.

7.3.3 An accessible platform

A big benefit of using the web as platform for developing an application such as this is the high level of accessibility it provides for the users. A web application does not require any cumbersome acquirement procedures, such as downloading and installing a software on the system, which is a time consuming task, and requires a higher gain expectation from the user to be willing to go through with it.

Since a web application is instantly accessible by a the user through a web browser, I think the conversion rate from non-users to users are much higher than if they would have to download and install a software. I believe it’s easier to let a user use an application and discover its value by themselves rather than having to explain it, or even trying to convince, them about it.

7.4 Extensions

There are many ideas that are not incorporated in the final prototype, but are useful and could provide great value if implemented.

More details

There exists a lot more information about the servers visualized in the treemap than what is shown. Not all servers are actually active, some are being initiated and some are not being used at all. This information would be useful to visualize, and allow for the users to filter and correlate the status of the data sources (servers) with the data.

Correlate different data types

Currently, it’s only possible to correlate one data type, one metric, with its data source. By overlaying multiple metrics in the line graph and linking them to their sources in the treemap with color encoding, further insights could be gained about the state of the back-end servers and software. For example, bandwidth usage of incoming traffic is likely to correlate with the number of connected users.

Modes of exploration

Further opportunities of exploring the data using human perception would possibly be created by allowing for changing the scales of the line graph. Examples could be to have logarithmic scales, normalized scales or integrals.
It’s hard to say what exact purpose this would serve but since the whole meaning of this prototype is to explore the data and not just display it for a pre-defined reason, it might be useful features to have.

It could prove useful to be able to add, divide or subtract values between different metrics. In the example from the previous section, this could be exemplified as dividing the number of bits of the incoming traffic by the number of currently connected users; incoming bits per user. Again, it’s hard to predict exactly what insights this information would create but it’s aligned with the basic concepts of data exploration; to let human perception do the work.
Chapter 8

Conclusions

The purpose and goal of this project was to explore new ways of visualizing and interacting with system monitoring data at Spotify AB. The work resulted in a design artifact that embodies knowledge about some of the things that are possible to achieve regarding data exploration within this context and an additional emphasize is also given to the relation between system monitoring data and physical space, and its meaning within system monitoring at Spotify AB.

This research contribution shows that by geographically and physically contextualizing system monitoring data, a sense of place is added that helps developers correlate system performance with physical placement of software and hardware, to predict future performance related bottlenecks or investigate whether past incidents were related to the physical placement of the systems. It also exposes possibilities for providing general assistance in the developers work on investigating technical issues and ensuring a high availability of the Spotify service by potentially allowing for new types of interaction with the data, such as overlaying metric visualizations or performing math operations on different sets of metrics as means of exploring possible correlations. The design artifact produced materializes these design visions and acts as a conduit to communicate these ideas to the developers at Spotify AB, increasing the likelihood that they will result in products for widespread use within the company.

Technical insights regarding good practices and effective solutions to ensure performant and useful user interfaces for data visualization when using web-based technologies have been gained as well, where the recommendation is to use the pixel graphic based canvas method for very large volumes of data, and the node tree based svg method when the amount of data is smaller and the requirement for interactivity is higher. This further emphasizes the strength in creating design artifacts for research contribution in the way this feeds back technological knowledge to the human-computer interaction and the computer science community.

Hopefully the results and conclusions of this thesis will serve as inspiration for further development of useful monitoring tools within Spotify AB as well as a resource for continued exploration and research on this topic.
References
