Matias Arpikari

Weather Map Design and Configuration User Interface

Helsinki Metropolia University of Applied Sciences

Master’s Degree

Information Technology

Master’s Thesis

15 January 2016

Abstract

Author Title Number of Pages Date	Matias Arpikari Weather Map Design and Configuration User Interface 53 pages + 6 appendices 15 Jan 2016
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Telecommunications
Instructors	Ville Jääskeläinen, Principal Lecturer Tuomo Lauri, System Designer

During the past few years in the field of weather map production there has been an increasing awareness of the importance of more informative, decorative and up to date layouts for the weather products. Additionally, the weather products should only be created on demand in order to save server computing power and storage resources.

This has led the case organization, the Finnish Meteorological Institute, to consider more modern ways to produce, configure and deliver weather data to the audiences.

This thesis concerns the creation of the design and configuration user interface for weather maps and animations. The requirements for the user interface included that it had to be effortless to use even for the non-technical end users of the case organization. Additionally, the user interface should be implemented in the existing weather delivery channel of the case organization.

The theoretical part of this thesis describes the main principles for creating a test plan to acquire information of the user interface even without writing a single line of code.

In the practical part, the acquired test results were utilized to develop an easy to use interface and a working prototype was developed. The methods and theory presented in the thesis should be applicable to other user interface design and development projects too.

The findings of this study indicate that it is recommended that the case organization utilizes the usability testing methods increasingly in the future to achieve user friendly user interfaces in the upcoming UI development projects.

Keywords	meteorology, weather, user interface, usability, paper prototyping, user experience testing

Metropolia

Contents

Glossary

Brainstorm	An HTTP based weather data API developed and used by the Finnish Meteorological Institute.
Dali	The name of the back end system for the weather image production. The name comes from Salvador Dali, the famous artist.
Geoserver	GeoServer is an open source server for sharing geospatial data.
Ilmanet	Ilmanet is an HTTP based weather product delivery system which is maintained and developed by the Finnish Meteorological Institute and used by it's employees and customers.
Ilmatie	The administration section of the Ilmanet. Only available for the employees of the Finninsh Meteorological Institute.
JSON	JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.
MetOClient UI	MetOClient UI is an open source JavaScript library developed by the Finnish Meteorological Institute which expands the OpenLayers library with visualized animation features.
Model	Computer calculated weather predictions.
qdcontour	A contouring program developed and used by the Finnish Meteorological Institute to read querydata format and draw static weather images like jpeg or png formats.
SVG	Scalable Vector Graphics
UI	User Interface
WMS	A Web Map Service (WMS) is a standard protocol for serving georeferenced map images over the Internet that are generated by a map server using data from a GIS database. The specification was developed and first published by the Open Geospatial Consortium in 1999.

1 Introduction

The case organization, the Finnish Meteorological Institute (FMI), serves as a weather research and service agency. The case organization provides the best possible information of the weather in Finland and in areas nearby. The case organization also ensures the public safety by providing information and warnings of possible airborne hazards and satisfies the need for specialized meteorological products.

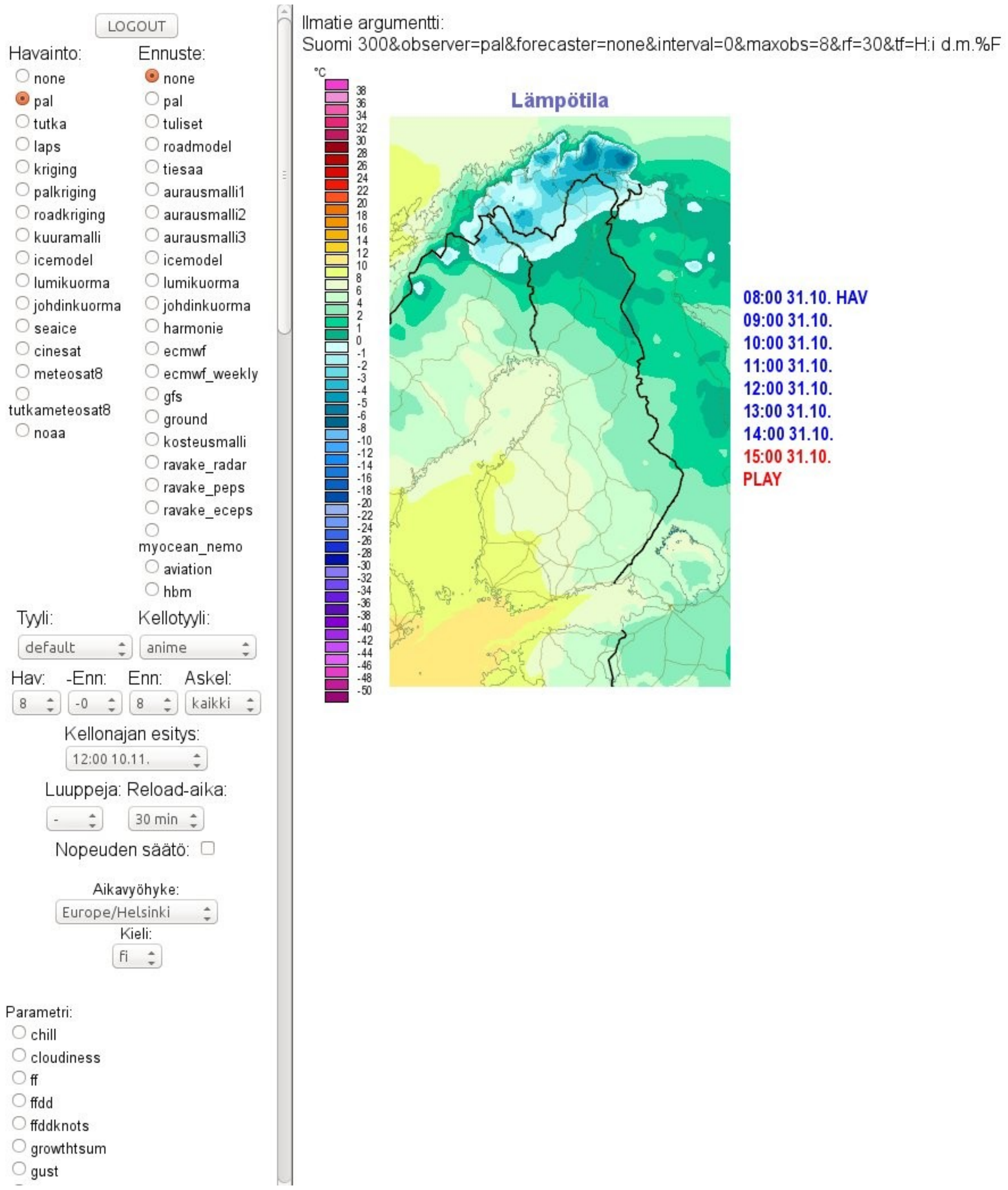
There are around 400 weather stations in Finland managed by the case organization. The stations measure different weather parameters which are then delivered near real-time to the case organization's database. In addition to ground stations weather data is gathered from many other sources such as weather radars and satellites. While the measurements show the current weather condition the weather forecast models serve the need to predict the future. Different forecast models are produced around the world. The case organization itself produces a multitude of different models and some are exchanged with other similar agencies.

The data itself is often raw, meaning that it is basically in some binary format or simple numerical data which usually do not tell too much to an inexperienced eye. This is why it is important to process the raw data into an understandable and effortless format. Images and especially animations are a good way of delivering a lot of weather information quickly and so that it is easy to understand.

1.1 Weather Data Production in the Case Organization

The Finnish Meteorological Institute has to deal with huge amounts of data. The weather data itself is often not self-evident but requires a lot of processing into an understandable format. There are definitive separations between the raw data and the actual end-user products. Therefore, the case organization has to find ways to optimize this process because now a lot of it is done by manual processes. Basically, all work revolves around the data collection, processing and delivery, and all of these steps have many challenges.

Due to this the production chain is relatively slow and the produced weather images are often not representative enough, and it is important that the case organization can produce more informative, decorative and up to date layouts, images and other weather products faster. Figure 1 shows an old animation configuration user interface.



LOGOUT

Havainto: none [✓ pal] tutka laps kriging palkriging roadkriging kuuramalli icemodel lumikuorma johdinkuorma seaice cinesat meteosat8 tutkameteosat8 noaa

Ennuste: [✓ none] pal tuliset roadmodel tiesaa aurausmalli1 aurausmalli2 aurausmalli3 icemodel lumikuorma johdinkuorma harmonie ecmwf ecmwf_weekly gfs ground kosteusmalli ravake_radar ravake_peps ravake_eceps myocean_nemo aviation hbm

Tyyli: default

Kellotyyli: anime

Hav: 8

-Enn: -0

Enn: 8

Askel: kaikki

Kellonajan esitys: 12:00 10.11.

Luuppeja: -

Reload-aika: 30 min

Nopeuden säätö: ☐

Aikavyöhyke: Europe/Helsinki

Kieli: fi

Parametri: chill cloudiness ff ffdd ffddknots growthtsum gust

Ilmatie argumentti:
Suomi 300&observer=pal&forecaster=none&interval=0&maxobs=8&rf=30&tf=H:i d.m.%F

Figure 1: Old animation configuration user interface (Animbrowser website, 2015).

In Figure 1 there is a screen shot of the old weather animation configuration user interface (Animbrowser website, 2015). While it is still functional and in use it lacks the flexibility and is rather difficult to configure. Additionally, the old weather animations are not responsive enough and are not updated frequently enough.

1.2 Current Technical Challenges

The current challenge is the automation of the production and ease of configuration of visually impressive weather images, maps and animations. Presently, the case organization can already produce weather images but the configuration is still very difficult and time consuming with the case organization's old hard-coded and highly specific drawing systems. Also, the variation of different kinds of weather image products as well as the real time requirements are rapidly growing, so the case organization needs to address this by developing a production system that answers to the needs.

For this reason a back end system Dali is currently being developed. Dali aims to produce more attractive weather images in SVG format but its configuration is currently challenging and practically impossible for non-technical persons (Heiskanen, 2015).

The design and development of the weather image configuration user interface was conducted at the Finnish Meteorological Institute. A new configuration user interface was needed because previously the configuration was by large extent a manual process. The manual process was time consuming and required technical expertise. Additionally, the images were previously all created every time new data was available. This consumed a lot of computer power. The new system creates weather images only when needed. The goal was to create a user interface that does not need any special technical knowledge of how to configure weather images and which is in production after the study was finished.

The user interface greatly speeds up the deployment time and releases the resources of the programmers for other tasks. The main end users are the sales managers of the case organization who may with the help of the user interface independently configure and create showy weather images for their customers.

The case organization has not widely utilized user interface and usability testing in previous projects so this thesis targets to serve as an insight to this kind of software development.

1.3 Objective

This study is about designing and developing an easy-to-use user interface for configuring the back end system. The technical challenge for this study relates to the need of the case organization to improve the automation and distribution of configuration of weather refining processes to configure the images by people who have different levels of technical background.

The case organization aims at producing an intuitive and self-evident system which will ease up the configuration of weather images and animations. The case organization has its own web-based weather product delivery system (Ilmanet) which is planned to serve as the host for this new configuration user interface. In addition, a research needs to be done on how much configuration will eventually be allowed via the user interface.

To create such a user interface requires knowledge of the configuration of the back end system of the weather image engine (Dali) as well as the Weather Map Service (WMS) and the Geoserver which is an open source server for sharing geospatial data. Basic knowledge of the SVG image technology and manipulation is also needed. In addition, the best practises of how to design and develop an usable browser based configuration interface were researched. The finalized user interface produces JSON configuration files for the back end according to the user's specifications. Figure 2 shows the process map of the weather map confiruration.

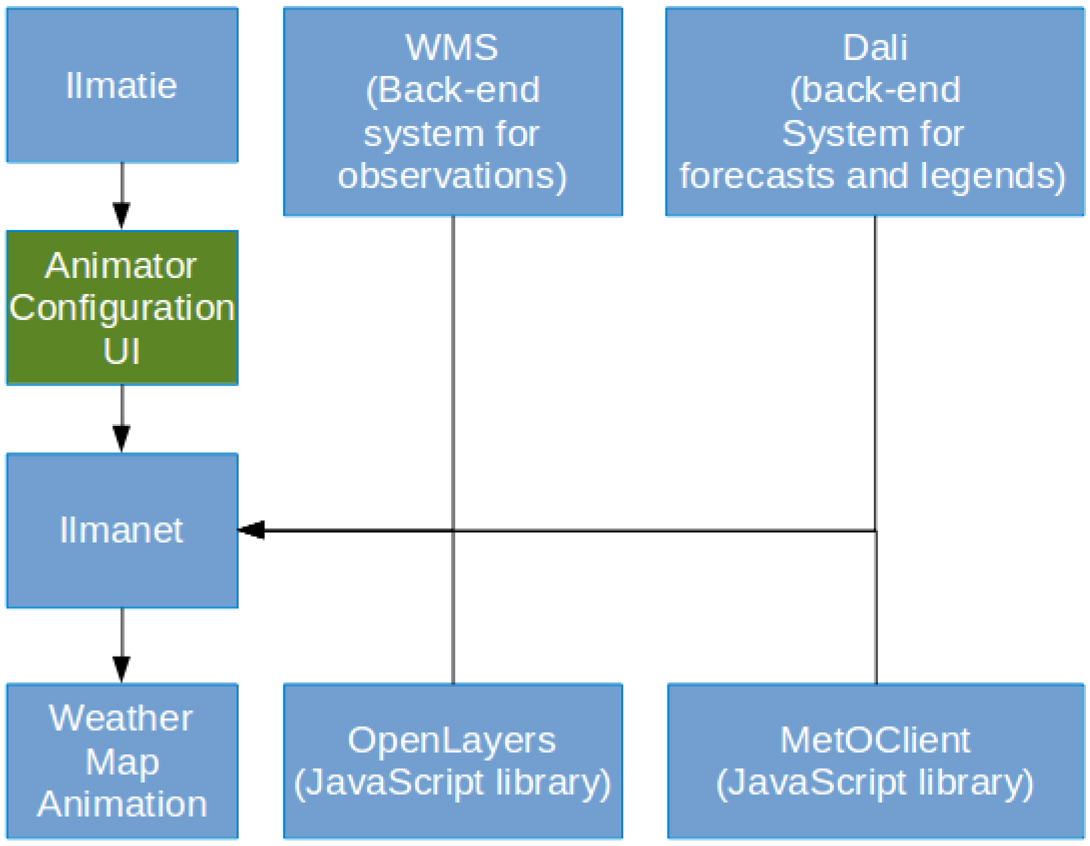

Ilmatie
WMS (Back-end system for observations)
Dali (back-end System for forecasts and legends)
Animator Configuration UI
Ilmanet
Weather Map Animation
OpenLayers (JavaScript library)
MetOClient (JavaScript library)

Figure 2: Process map of the weather map configuration.

This study aims to design and develop an easy-to-use user interface to configure the weather image production back end system. The simplified process map of the configuration is shown in Figure 2. The green box (Animation Configuration UI) in the figure indicates the main objective of this study. Other technologies needed in the project were PHP, JavaScript, CSS and HTML.

To make this improvement, the case organization had to deal with the following issues:

1. Gather information on what needs to be configured and how the UI should be designed
2. Research the best practises in user-centered design
3. Develop the UI
4. Test the UI and improve the UI according to the feedback

1.4 Scope and Content

This study describes how to design and develop a fresh and easy-to-use user interface for configuration of the weather image production system. For this to be achieved information of different configurable parameters needed to be decided and collected. The best methods of creating a self-evident and effortless user interface are also discussed. The user interface had to be easily adopted by those with little or no technical background.

The main task was to actually develop the user interface. The goal was to design and program it so that it is easy to refine and develop later in a way that nothing would be blocked out during the process. An important part of the thesis was the initial test with the paper prototypes to gather information and minimize the unnecessary work.

A test period was defined during which the testers could provide feedback and changes were made accordingly. After this the user interface was taken into production. The development of Dali is not included in this study.

Another feature out of the scope of the thesis was the accessibility. Accessibility refers to the design of products, devices, services, or environments for people with disabilities and is often considered in user interface design. Accessibility is disregarded because the end solution is used only internally by the account managers and technical personnel. This is not to say there will never be disabled persons using the configuration UI but the accessibility as a whole is a much bigger task than this study could include. Additionally, the surrounding framework is not designed to be accessible so it makes no sense to design and develop the configuration UI in such a way. Of course, some key principles of accessibility are discussed.

Since the solution in this thesis was done for a specific case organization and inside a specific product delivery channel (Ilmanet) the actual end product is not necessarily reusable. But the presented theory on how to produce scalable and rich image services can be used as a starting point for creating one's own production system. Additionally, the part of how to design and test a self-explanatory and easy-to-use user interfaces for non-technical persons is likely to be applicable in most web applications.

Since the solution was implemented into the still non-responsive framework of Ilmatie (the administration section of the Ilmanet) the responsiveness was disregarded in this study. The client application (MetOClient) is of course responsive but the development of that was mainly out of scope of this thesis. The exclusion of responsiveness speeded up the actual programming but did not set a limitation for such a technology to be implemented later.

1.5 Research Design and Structure

The design part of the user interface started with an examination of the requirements of the configuration tool. This included researching the documentation and functionality of the back end system (Dali) and finding out the parameters which needed to be configurable via the user interface. The expectations of the sales managers were also taken into account. Additionally, the user interface was programmed so that it would not narrow down the future development. Next, the current best practise user interface literature was studied in order to create an intuitive and self-explanatory user experience since the end users are inexperienced with technical configuration. Additionally, a preview functionality was added.

A prototype user interface programmed with PHP, JavaScript, CSS and HTML was presented. A series of test periods were arranged for the colleagues and the sales managers and based on the user feedback modification needs were discussed and made to the user interface.

To accomplish the requirements of the weather map design and configuration user interface research of the best practise usability literature was done. Additionally, the test and interviewing results are presented here and development decisions were made accordingly.

The thesis consists of six main chapters. After an introduction, background and basis of the weather map production were presented in Chapter 2. This includes a short description of the current situation and the obstacles detected. Second, some general and advisable principles of how to design and develop an user interface that is considered as usable are discussed in Chapter 3. Additionally, a short introduction to the usability testing is given and an example of paper prototype testing for gathering information for the user interface is presented in Chapter 3. Finally, the development of a prototype application is explained in Chapter 4 and the results of the solution are evaluated in Chapter 5 and the discussion and conclusions are presented in Chapter 6.

2 Background

This section describes the background and requirements in an automated weather map production system in the case organization. Some of the best practices in user-centred user interface design and guidelines are also presented.

2.1 Core Concepts in Weather Image Production

In order to produce automated weather images and other related rich products one has to have knowledge of weather map visualization technology. The procured raw weather data has to be refined into descriptive and understandable images. There are a number of different existing visualization tools developed by other meteorological institutes such as Metview, Synopsis and SatRep (Tervo, 2011; 25-27).

It has been a trend for some years already to allow users to interact with the web applications rather than just display the information. This greatly improves the user experience by allowing the user to decide what and to display the information. In the case of weather images this might include things such as how the user specified weather parameters, location and time.

Since the weather data is almost always location based it often makes sense to display it on a map. Another thing to consider is that weather changes over time. Thus, some kind of an animation is many times an apparent way to present the data. One might also be interested on how the weather develops around the specific location which is why zooming and dragging are useful functionalities for a pleasant user experience.

Until the zoomable and draggable user interfaces such as the OpenLayers as well as on-demand map servers such as the Geoserver were developed and became widely supported huge numbers of weather images needed to be created in advance, even though no one ever looked at them (Open Source Geospatial Foundation 2015). This consumed a lot of physical disk space and server time. The old way of doing weather map animations meant that there needed to be a massive number of already processed weather images located somewhere in the memory of the server. It is estimated that the Customer Services unit of the case organization alone created around one million images per day (Tervo, 2011; 8). These images were then fetched and displayed to the user with various technologies and they existed regardless if anyone ever downloaded them. Additionally, the known and even expected features of today like the zooming and dragging were troublesome with the old weather image systems.

2.2 Current Production System at the Case Organization

The case organization has traditionally used an in-house program called the *qdcontour* for the weather image production. Although the current system is still operational the configuration of this program is complex and time consuming which is why the case organization has decided to develop a new weather image production system called Dali. Dali is a part of the next generation weather data production system called the Brainstorm. Dali also solves the problem where the images can be created just when needed and unnecessary storage consumption can be minimized. Additionally, the reason for not relying solely to the existing systems (e.g. MapServer or GeoServer) to produce the weather images is that they had quality issues and difficulties supporting certain wanted properties (Tervo, 2011; 38-39).

The case organization utilizes the Open Geospatial Consortium (OGC) standards such as the Web Map Service (WMS) protocol to serve the maps and weather images produced by Dali.

2.3 Obstacles Solved

The case organization has already solved many obstacles of how to automate the production of attractive, rich and on-demand weather images. These solutions include the hardware and software for the map servers, enduring databases, geospatial technologies, back end production systems, etc.

MetOClient is an open source JavaScript library developed by the case organization to display weather images. It utilizes the OpenLayes library and supports both the WMS and Dali images. MetOClient was chosen to be the user interface through which the case organization serves the specially configured weather images to its customers.

2.4 Obstacles to Be Solved

Although Tervo (Tervo, 2011; 75) presented one possible solution for the configuration user interface in his thesis, he found out that it needed too much of technical knowledge. So, the case organization is still missing the self-explaining and easy-to-use user interface for designing and configuring the weather image and animation products. The user interface needs to be clear enough and easily adoptable even for the non-technical persons. How to design and develop such an interface is the main emphasis of the present study.

2.5 Requirements

Since the weather image configuration user interface is meant to be a part of the internal weather product delivery channel Ilmanet there are some limitations to consider. Ilmanet has an administration section called the Ilmatie. Ilmatie was designed so that independent plugins can be developed via the Ilmatie application interface. The main implementation technology is JavaScript and its supporting framework jQuery. Since Ilmatie is a PHP based framework the plugin needs programming in PHP too. The weather map configuration UI should output JSON formatted string that is to be stored in the Ilmanet database. The JSON needs be in the format specified by the client library MetOClient which does the visualization of the end product (see Reference 1 for an example).

Additionally, the UI should be usable for the non-technical Account Managers of the case organization. The end users need to be able to login to the Ilmatie and create a new weather animation for the customer. The end user needs to be able to do the weather animation configuration easily and without technical knowledge. Finally, the configured weather animation needs to be visible to the customer in the Ilmanet.

2.6 Standards and Technologies

This chapter shortly describes some of the key standards and technologies that need to be understood in order to create a solution. The standards and technologies involve downloading or displaying configured geospatial data in a web browser.

Scalable Vector Graphics (SVG) is an open standard of an XML-based vector image. Usually one XML document defines the SVG image and can it be scaled to any size without any loss of the quality. Since the image is basically just XML the content can be easily manipulated with any text editor or programming languages such as JavaScript and viewed in all modern web browsers. (W3C, http://www.w3.org/Graphics/SVG)

JavaScript Object Notation (JSON) is an open standard and completely language independent format to describe data objects as attribute-value pairs. The benefit of the JSON over XML is that it is considered to be easier to write and read. (W3C, http://www.w3schools.com/json/)

A Web Map Service (WMS) is a standard protocol for serving georeferenced map images over the Internet. The WMS provides an HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. The response of the WMS can be an image like a JPEG or PNG which can be displayed in a browser application. The WMS interface also supports transparency so that multiple different layers may be displayed on top of each other. (Open Geospatial Consortium, http://www.opengeospatial.org/standards/wms)

Geoserver allows users to share and edit geospatial data. It supports a number of open standards such as Web Feature Service (WFS) and Web Map Service (WMS). It is possible to create and edit different kinds of layers in the Geoserver. These layers may then be displayed in a client application. (Geoserver, http://docs.geoserver.org)

2.7 Contradictions

The case organisation already has a working production system for creating weather maps. The problem is that the configuration of the current system requires technical knowledge and plenty of time as well as resources from the developers. Additionally, the current system is unable to serve the ever growing need to produce more attractive weather maps in an efficient way. The purpose of this thesis is to alleviate these contradictions by providing recommendations and solutions to the problems in the current system.

3 Principles of User Interface Design and Usability Theory

This chapter describes how good and easy user interfaces should be designed. If one considers it any user interface is basically a form. Forms are used to collect information in an organized format from the users. Almost every form, or a UI therefore, usually includes different kinds of selectors and control inputs. It is quite easy to create a complex UI but designing a self-explanatory and effortless UI needs some planning and testing.

One should consider the following key design principles (Wroblewski, 2008; 19):

1. **Minimize the pain**. No matter how great design and functionality the UI has people actually only want what is on the other side of the form.

2. **Illuminate the path to completion.** Show the steps what it takes to complete the task to get to the other side of the form.

3. **Consider the context.** All forms have different target audiences, applications and businesses. Consider how to implement the form into the context.

4. **Ensure consistent communication** (one voice instead of many from i.e. marketing, privacy, engineering, design, business).

Effortless UI includes only what it needs and no more. Asking unnecessary information is a fault (Jarrett & Gaffney, 2008). If some information is not needed it is best to remove it completely. If it is needed only every now and then one should consider hiding it so that it is not distracting but still available if needed. (Krug, 2011; Chapter 5)

Whenever the form needs user input the first step for the user is to understand the question. Then, the user seeks for an answer. Next, the user decides whether the answer fits the question and finally user enters the answer. (Tourangeau, Rips, and Rasinski , 2000)

The more the UI makes the user to think the less likely it becomes to receive an answer. This is why it is a good principle to not make the user to think or make him think as little as possible.

Put simply, the most important content should stand out the most, and the least important should stand out the least. In other words, a reader should be able to deduce the informational structure of the page from its layout (Tidwell, 2011; 134).

Questions that should be self-evident to the user in any UI are (Krug, 2014; Chapter 1):

- What is this and what can I achieve with this?
- Where should I start?
- What are the most important things on this page?
- Why is that named like that?
- Is that a part of the form?
- Where is the functionality I'm looking for?

It is critical that the UI or form explains what the user can do with it. This should, of course, be self-explanatory but it is understandable that not everything can be made in that way. Additionally, any control input and their labels should be easy to understand and replying or controlling them should be made effortless. (Krug, 2014; Chapter 1).

Tool tips are often used to display additional instructions. Tool tips help to minimize the amount of text in the UI and thus makes it more easy to scan them.

People usually do not want to learn to use things, at least consciously. They want everything to be made easy and as self-evident as possible. According to Krug's first law of usability one should not make people think when it comes to the user interfaces (Krug, 2014; Chapter 1). People are amazingly unaware of all of the incoming information that is directing decision making. The changes which the external stimulation triggers in the system are not necessary conscious experiences. (von Fieandt, 1972).

The truth that is often forgotten is that the users actually spend most of their time on other websites. Thus, anything that is a convention and used on the majority of other websites will be burned into the users' brains and you can only deviate from it on pain of major usability problems. The former is known as the Jakob's Law of the Web User Experience (Nielsen, 1999).

This applies especially if the new user interface is created inside an existing framework. One should recycle its previously learned features and functionalities as much as possible. Every new feature or functionality has a learning curve and makes the UI less usable. Hence, the designer has to consider whether to reuse or slightly modify the existing elements or to create something completely new, thus endangering the user experience.

3.1 Usability Testing Theory

There are certain things to consider when creating and testing a new user interface. In a perfect world the testing could take a year or more and one would not need to worry about the budget. Unfortunately, this is rarely the case but the key principle is that some testing is definitely better than none at all.

It is also a great matter of debate of how to distinct or call different kinds of usability tests. Others like to separate i.e. user experience testing from the usability or the accessibility testing. All in all, testing in general is always a positive thing, no matter how you may want to call it. Furthermore, it is beneficial to familiarize one self with the certain key aspects of testing so that it is easy to collect valuable and useful information.

Most of the usability experts recommend to do the usability testing in multiple phases and with external test persons. In each subsection of testing it is worthwhile to provide the test person a short description of what is being tested and that the test person himself is not the one who is being tested, the UI and its features are. It is practically impossible for the test person to make mistakes. It is also advantageous to explain that anything the test person says can not hurt the feelings of the developer or the tester. The target is to get get as much feedback as possible and this is only possible if the test person will be honest.

Key properties of a successful tester include the understanding of the basics in user-centered design, good people skills and an excellent memory. Quick learning and flexibility to deviate the test plan will ease up the testing process. The tester needs to have a long attention span with good and empathic communication skills. Among all the other things the tester should always think about the “big picture” and try to get the most of out of the test persons. (Rubin & Chisnell, 2008; Chapter 2).

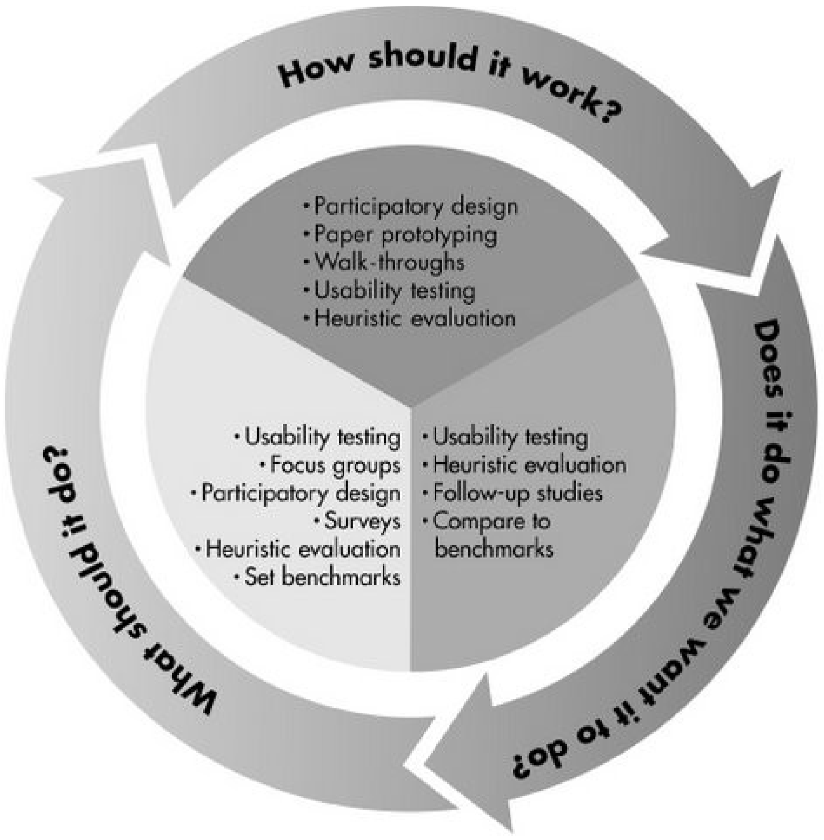


Figure 3: Questions and methods for answering them (Rubin & Chisnell, 2008, Chapter 1).

In a user-centered approach the user feedback is expected and received in each phase prior to moving to the next phase. This can involve a variety of techniques, usability testing being only one of these. The various other methods include techniques sucs as focus group research, surveys, team walk-throughs, paper prototyping and follow-up studies. (Rubin & Chisnell, 2008, Chapter 1).

Things that the tester should consider while examining the test person in any UI test include questions such as (Krug, 2014, Chapter 9):

- What are the best and worst properties of each approach?
- Which are the biggest obstacles for the user?
- After a short learning curve which properties are the most useful to the user?
- In which things the user needs help, more information or supporting documentation?
- What kind of written documentation is needed? Pre-knowledge, theoretical, perspectual, functional, examples or education?

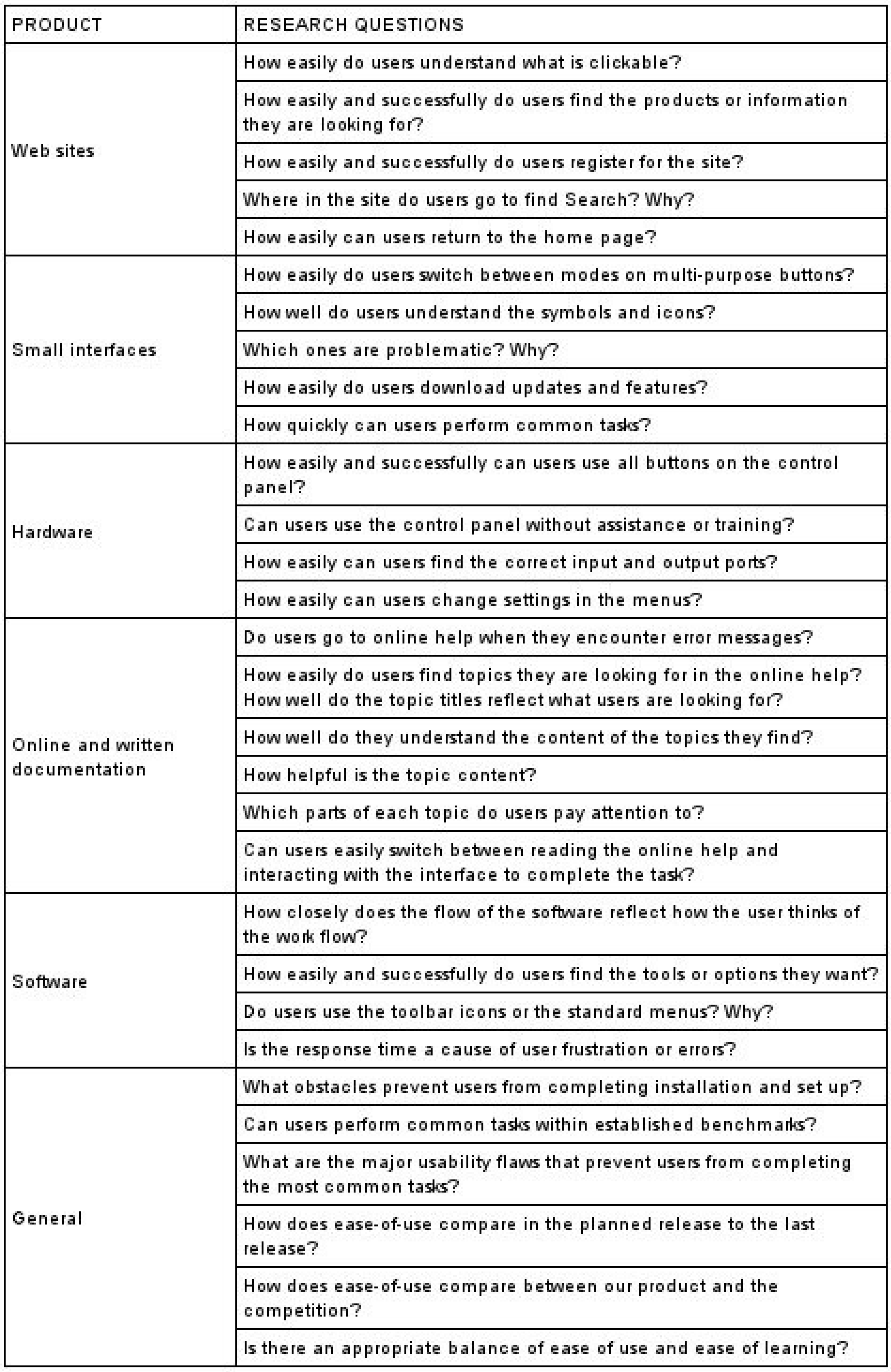
- Which of the discovered problems should be fixed and how?
- Is the UI technically well developed and is information secured?

Common Guidelines to Testers

In all phases the tester(s) should consider which are the best and worst properties of different solutions. Questions such as which are the biggest problems or after a short period of learning which are the most valuable features to the user should be discussed. Is it clear where the user needs additional information or documentation? Although problems are confronted which of those are the ones that should be handled and fixed.

The tester should also consider whether the user interface is usable. The UI should be beneficial and efficient meaning that it should do what it is supposed to do and preferably efficiently so without consuming too much of effort and time. A desirable UI is also learnable which means that an UI can be learned and memorized without doing it every time again. A simple way of finding out whether an UI is usable is to observe the end users. How do the end users find the UI? Is it needed and how do they feel about using the UI? If the end users do not think it is delightful to use the UI something is not correct. And finally, a special consideration towards the reachability should be taken into account if people with incapabilities should be able to use the UI. In this UI the reachability aspect was agreed to be disregarded in order to save time and resources.

A set of example research questions depending on the test subject can be seen in Figure 4.



PRODUCT	RESEARCH QUESTIONS
Web sites	How easily do users understand what is clickable?
	How easily and successfully do users find the products or information they are looking for?
	How easily and successfully do users register for the site?
	Where in the site do users go to find Search? Why?
	How easily can users return to the home page?
Small interfaces	How easily do users switch between modes on multi-purpose buttons?
	How well do users understand the symbols and icons?
	Which ones are problematic? Why?
	How easily do users download updates and features?
	How quickly can users perform common tasks?
Hardware	How easily and successfully can users use all buttons on the control panel?
	Can users use the control panel without assistance or training?
	How easily can users find the correct input and output ports?
	How easily can users change settings in the menus?
Online and written documentation	Do users go to online help when they encounter error messages?
	How easily do users find topics they are looking for in the online help? How well do the topic titles reflect what users are looking for?
	How well do they understand the content of the topics they find?
	How helpful is the topic content?
	Which parts of each topic do users pay attention to?
	Can users easily switch between reading the online help and interacting with the interface to complete the task?
Software	How closely does the flow of the software reflect how the user thinks of the work flow?
	How easily and successfully do users find the tools or options they want?
	Do users use the toolbar icons or the standard menus? Why?
	Is the response time a cause of user frustration or errors?
General	What obstacles prevent users from completing installation and set up?
	Can users perform common tasks within established benchmarks?
	What are the major usability flaws that prevent users from completing the most common tasks?
	How does ease-of-use compare in the planned release to the last release?
	How does ease-of-use compare between our product and the competition?
	Is there an appropriate balance of ease of use and ease of learning?

Figure 4: List of example of research questions depending on the product. (Rubin & Chisnell, 2008; Chapter 5)

The tester may use specific questions depending on what kind of user interface is being tested. Figure 4 lists some examples of such questions.

3.2 Usability Testing Criticism

Although usability testing in general is usually accepted to be advantageous in the UI development it is sometimes looked down on for various reasons. It could be questioned whether the usability testing takes just too much time. While it is usually true that the proper testing does take some time it could be argued that in the long run the testing actually saves time because many obstacles have already been solved before writing a single line of code. (Tullis & Albert, 2008; 11).

Another complaint might be that usability testing simply costs too much. This might be because people tend to think that a proper usability testing can only be done by a highly specialized individual or company. This of course is not true. Additionally, many advanced usability tools are available free of charge online. (Tullis & Albert, 2008; 11).

Usability testing opponents might argue that testing is not useful when focusing on small improvements or that the testing does not help to understand the real causes of the usability problems. But by looking at the severity and frequency of the usability issues and why they occur is an excellent way to focus resources during the design process. You can also identify where in the system users experience problems and use the metrics to tell where and even why some problems occur. (Tullis & Albert, 2008; 11-12).

One of the biggest criticism is targeted towards the matter that usability data is too “noisy”. Although this can be true an experienced tester can design the tests so that minimal noise wil occur and filter out the remaining noise (Tullis & Albert, 2008; 12). A lot of times in the UI development guesswork is used instead of usability testing data. Intuition is many times great but a fact based data is always better (Tullis & Albert, 2008; 12).

It is a widely held belief that a large sample size is required to collect any reliable usability test results. And of course, a large sample size does provide better results for sure. But often even a few test persons can provide so much more useful data than doing no testing at all. (Tullis & Albert, 2008; 13).

3.3 Creating Test Plan

Writing a test plan is helpful in answering questions such as what the user interface will be about, who will be using and what should it include. In this chapter some common principles of writing a test plan and conducting UI tests are described in more detail. Things to consider when creating a test plan include at least the following (Rubin & Chisnell, 2008, Chapter 5):

- What are the properties needed in the UI?
- What is the proper way to implement the above properties?
- Have the required properties been implemented according to the best practices of usability principles?
- Has the technical development or a mock up been done in a way that it respects the project requirements and information security?

The test plan will also serve as a blue print for the test. It is the foundation and the main communication vehicle of the whole test (Rubin & Chisnell, 2008, Chapter 5).

3.3.1 Reporting and Decision Making

Rather than solely making decisions of what to include and what to disregard and how to conduct things it is preferable to assemble a team of few members that will do the final decisions together. The tester should present the results of each test phase in a compact and understandable format. The presentation should also include the goals, scientific methods, logistics and the backgrounds of the test persons. After this the tester presents the results of the original research questions and gives the associated numerical data as well as explains the backgrounds of the questions and data. Images are often a descriptive and supportive way of presenting the results. In the end the team considers each point and decides the future steps. (Rubin & Chisnell, 2008, Chapters 11-12)

3.3.2 Gathering Information of Needed Features

It is always a good way to start a new user interface design process by researching what is really needed. Often a lot more is initially discussed and required than is actually necessary. This is where an experienced and talented tester can save a lot of time and resources by defining the substantial requirements. That being said, one should always develop the user interface so that it would limit the potential future development as little as possible. This is why it is important to remain open and upfront to the ideas emerging from the test persons and do compendious notes.

One way of gathering information is to arrange some relatively short interviews with the key persons to whom the user interface will be targeted to. In *Don't Make Me Think*, the author Steve Krug says that one should not interview too many but even one is better than none. The optimal number is around 3-8 persons. (Krug, 2014).

The background of the testers should vary. The greater the variance the better. Of course it should be considered if persons outside the target group are needed to test the UI. As a rule of thumb, if the UI is meant to serve people with different backgrounds external (outside of the target group) should be used. In this case some reasonable compensation should be considered to make the test more tempting. (Krug, 2014)

3.3.3 How to Create Usable Interface?

How to know if a user interface is usable? Of course testing is the primary way of discovering the usability of an UI but there are some general directing objectives that should be considered when designing the UI. It is important to realize that usability is not a single, one-dimensional property of a user interface. Usability has multiple components (Nielsen 1993; 26).

First of all, the most important thing is the following. Does the UI do what it is supposed to do? It makes no point of creating something that is not really needed. This is almost impossible to judge without some proper testing. Even though the UI should be created so that it is as self-evident as possible, it is important that it will be easily learnable. This is because many times it is very difficult to make everything as easy-to-use and clear. And because of the previous reason it is obligatory to make the difficult things at least re-collectable instead of learning it every time again. (Krug, 2014; Chapter 11).

The UI should be also usable, meaning that it needs to do what it is supposed to do and in an efficient way within a considerable time frame.

The end users should use the UI because they need it not because they have to. Additionally, using the UI should be delightful, or even fun! Nothing is more depressing than using an UI that makes you feel unpleasant even without actually using it.

Last but not least, a good UI is reachable when needed. Reachability includes design patterns which need to take in account things like if and when persons with incapabilities use the UI. It is practical to note that not all UIs need to be reachable but if an UI is designed ”for everyone” reachability should definitely be considered (Krug, 2014, Chapter 11).

3.3.4 Paper Prototyping

Paper prototyping, even though it sounds just like sketching and working with A4s, may be a lot more than actually drawing into a piece of paper. Paper prototyping is a common phrase for rapidly and economically test the UI ideas. Whether it is actually conducted with sheets of papers or some other way makes no difference. The point of paper prototyping is to gather instant feedback of the suggested UI and its properties without the need of actually creating a working UI. In order to do paper prototyping one should have a picture of what the UI should do. The purpose is to research if all the needed properties are included and if there is anything that can be disregarded and how to make the desired properties usable (Snyder, 2013).

Paper prototyping is a very efficient way of testing multiple different scenarios and use-cases inexpensively. It is very effortless and one can quickly make modifications even during the test period and before a single line of code is written. A good way to start is to create a mock up UI with the needed properties with as little styling as possible. This way the layout or the styling does not overwhelm the test person. Only the functionality and the actual content count.

Sketching a paper prototype is beneficial because then one will spend time thinking about the content and the priorities rather than fiddling with the design tool of choice. Creating the mock ups will be quicker, because of the clear idea at the outset of what needs to go where. Potential problems can be discovered earlier. It is easier and reassuring to move onto the finer design details if the UI works on paper. (Allen & Chudley, 2012).

When conducting a paper prototype testing one should arrange a quiet location for about an hour long session. The actual layouts can be drawn on sheets of paper by hand or with some drawing software. Today there are even a multitude of free and affordable cost online services to create layout mock ups such as the pidoco.com. Figure 5 illustrates an example of paper prototyping.

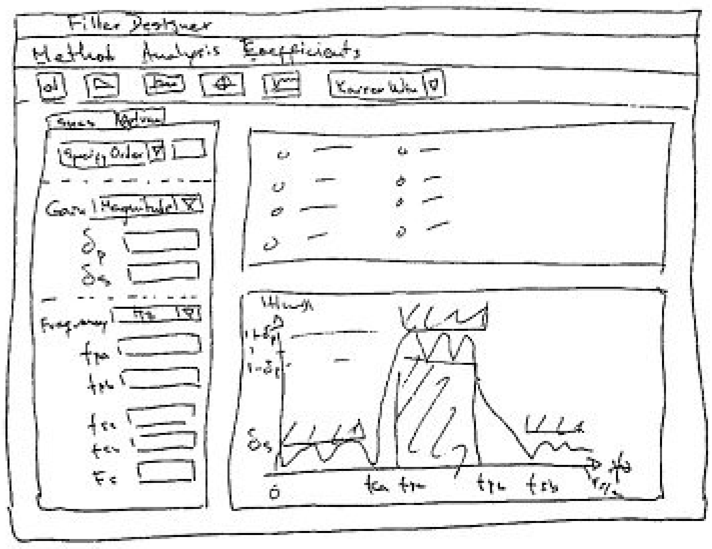


Figure 5: Rough example of paper prototyping. (Snyder, 2003; 4)

Before each test phase it is important to explain to the test person what is going to happen. This helps the test person to get oriented on the subject. Questions to ask from the test persons may include the following (Snyder, 2003; 150-151):

- What do you expect to accomplish with this UI?
- In your opinion, what are the most important features in it?
- How much time are you willing to spend to achieve what you are trying to achieve with the UI?

One will most likely be amazed by the answers given. Some of them are irrelevant but some will most likely be the ones nobody has even thought about before. This is why some kind of prototyping is critical when designing a new UI. You do not want to spend time and resources of doing something what is not really desired.

Some usability experts recommend recording the test situations or having observers to receive and catch more information. One tester can only process so much information so additional help in the form of recordings or observers may be beneficial.

3.3.5 Testing while Developing

It is always a good practise to do as many test cycles as possible. Preferably always when there is a new feature implemented. This way it is easier to modify the UI to the desired direction with minimal work.

The above, of course, is rarely the reality in life so one should at least include certain milestones and status meetings to the test plan. This way it is easier to follow the big picture and make future decisions accordingly. Too strict rules can also have a negative effect on the project.

3.3.6 Final Testing and Continuous Improvements

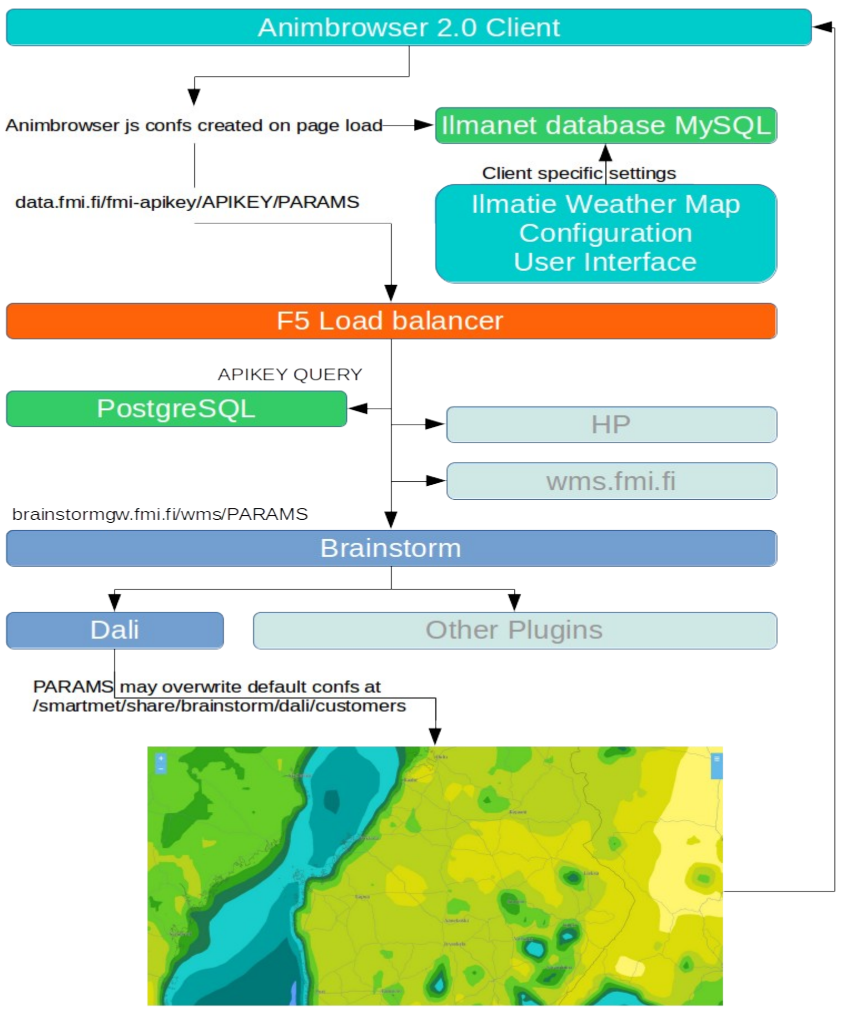
When the UI is somewhat ready, meaning that all the decided requirements are implemented and functional, at least one final test should be conducted. The test is divided between the actual end users and technical testing. The end users test the UI one final time when everything is supposed to be in place and provide feedback. The purpose of the technical review is to find possible security concerns which may have slipped from the developers. It is also important that the technical review is done by some others than the actual developers. According to the test results modification decisions are made.

4 Description of Solution

The goal of this thesis was to design and develop an easy to use weather map configuration interface that would be effortless to use even by the technically inexperienced Account Managers of the case organization. In this chapter an example of a real life paper prototyping and development of an actual working application for configuring weather maps that is in production is described.

First, the process of the user interface testing is described. Although many great testing methods exist the paper prototyping method was chosen to be used in this study. The paper prototyping is an efficient way of gathering information cost-efficiently of how to create an easy-to-use user interface for a specific task. Secondly, how to create a test plan and test execution is discussed. Third, this chapter gives examples of how to acquire information of the needed features in the new UI and how to decide what to included and what not. Last, this chapter describes how the test results were used to develop the prototype application.

The basic functionality of the new weather map work flow is presented in the following image (see Figure 6).



Animbrowser 2.0 Client
Animbrowser js confs created on page load
Ilmanet database MySQL
Client specific settings
Ilmatie Weather Map Configuration User Interface
data.fmi.fi/fmi-apikey/APIKEY/PARAMS
F5 Load balancer
APIKEY QUERY
PostgreSQL
HP
wms.fmi.fi
brainstormgw.fmi.fi/wms/PARAMS
Brainstorm
Dali
Other Plugins
PARAMS may overwrite default confs at /smartmet/share/brainstorm/dali/customers

Figure 6: Flow chart of the new animation work flow.

Everything starts by opening and defining the values in the weather map design and configuration UI (see the box named Ilmatie Weather Map Configuration User Interface in Figure 6). After the user is satisfied with the configuration the properties will be saved to the Ilmanet database. Ilmanet is the parent framework of the weather map configuration UI. Animbrowser 2.0 Client (aka MetOClient) is the client UI previously developed by the case organization. This UI is used to display the client side end product, the actual animation with selected layers, that is.

Because of the Ilmanet framework's plugin development requirements the code of the existing MetOClient was first copied, edited and refactored for the new plugin called the Animator. The Animator is responsible of the configuration UI as well as some of the additional features on the client side. The Animator plugin development is what this thesis is really about.

When loading the client side animation the Animator plugin reads the database and creates a JSON that is needed to display the chosen features. An example of the JSON can be seen in Appendix A. The layer images are called through the case organization's open data API. For this an apikey is required (registration needed). The HTTP call goes through the F5 load balancer which validates the apikey against the PostgreSQL database.

Depending on which kind of data is requested the call is then directed to either a WMS server in the case of the observation data or to the Brainstorm's Dali plugin which is responsible of the forecast data. In both cases the typical reply is a PNG image of a certain area and time with the corresponding weather data.

An example of the observation image HTTP call in the JSON:

http://2.p.wms.fmi.fi/fmi-apikey/f01a92b7-c23a-47b0-95d7-cbcb4a60898b/geoserver/wms?LAYERS=KAP:skandinavia_dbz_eureffin&TILED=true&VERSION=1.1.1&TRANSPARENT=TRUE&FORMAT=image%2Fpng&TIME=2015-10-31T10%3A00%3A00.000Z&SERVICE=WMS&REQUEST=GetMap&STYLES=&SRS=EPSG%3A3067&BBOX=181246.432,7773016.936,443390.432,8035160.936&WIDTH=256&HEIGHT=256

Here the URL consists of the server indicator number (2.p) and the actual open data API's domain *data.fmi.fi*. The fmi-apikey (*f01a92b7-c23a-47b0-95d7-cbcb4a60898b*) individualizes the user. The *geoserver* part indicates that this call is directed to the geoserver to request observation data from the WMS server. Next, the desired layers are defined (here *KAP:skandinavia_dbz_eureffin*, meaning some radar dbz data from the Scandinavian area). Additionally, some more definitions are given like the time, projection and the area. An example of the response image of a certain area and time can be seen in Figure 7.

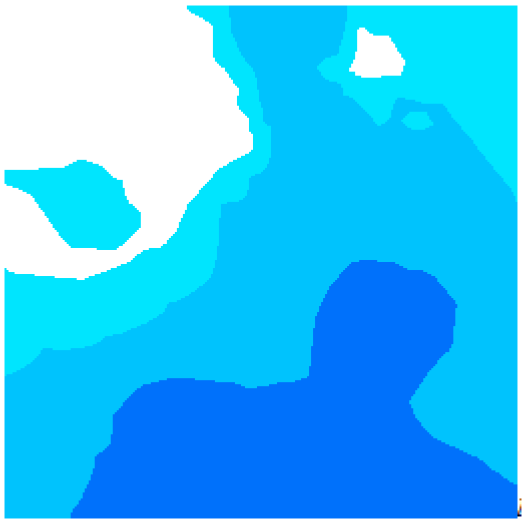


Figure 7: An example PNG image (tile) of the wind from the WMS server.

An example of the forecast image HTTP call in the JSON:

http://2.p.data.fmi.fi/fmi-apikey/f01a92b7-c23a-47b0-95d7-cbcb4a60898b/wms?LAYERS=fmi:pal:rawtemperature&TILED=true&VERSION=1.3.0&TRANSPARENT=TRUE&FORMAT=image%2Fpng&TIME=2015-10-31T14%3A00%3A00.000Z&EXCEPTIONS=INIMAGE&SERVICE=WMS&REQUEST=GetMap&STYLES=&CRS=EPSG%3A3067&BBOX=181246.432,7510872.936,443390.432,7773016.936&WIDTH=256&HEIGHT=256

The URL consists mainly of the same parameters as the geoserver call but is missing the geoserver part. This call is therefore directed to the Brainstorm's Dali plugin to acquire the forecast images.

The client application calls these images according to the JSON configuration data produced by the weather map design and configuration user interface. The images are then handled and arranged by the the client UI and the animation may start running.

4.1 Gathering Information of Needed Features

The case organization wanted to have a new user interface for designing and creating weather maps and animations. The UI was to be designed so that it is easy to understand and use even for the non-technical persons. As many times it was not totally clear what was actually wanted so a paper prototyping technique was chosen to clarify the subject. In this project actual paper sheets with mock-up UIs were used.

First, a series of meetings with the project key persons were held. In these meetings rough guidelines of what should be included to the UI were formed and written down. From these guidelines the skeleton for the feature interviews with the test subjects was created. The skeleton of the testing was divided into four different phases as illustrated in Figure 8 below.

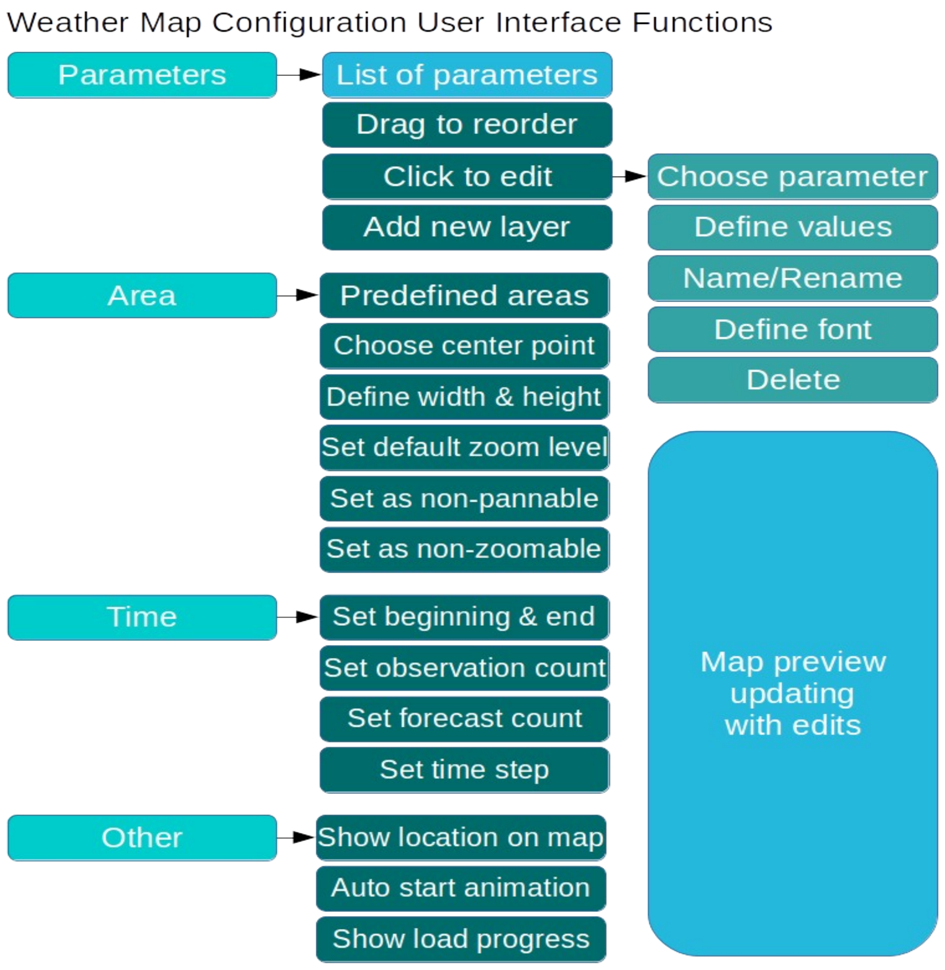


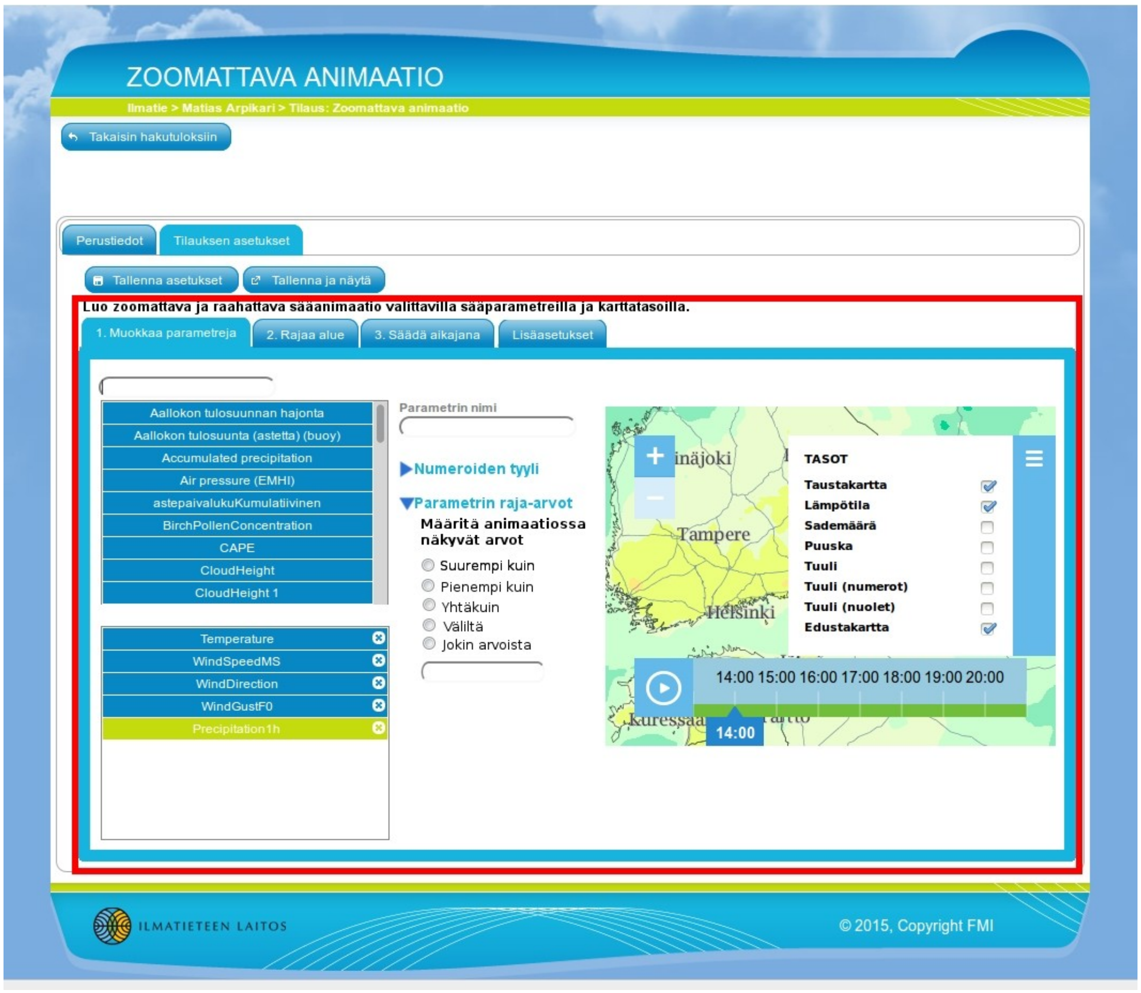
Figure 8: Weather animation configuration user interface features.

Based on the meetings it was decided that at least the properties indicated in Appendix D should be implemented. Figure 8 shows the initial views and the wire frame of the functions and features which the weather map configuration UI should have. There are four different main sections on the left side of the figure: Parameters, Area, Time and Other. All of these have their subsections, or features that each section should include. On the right side there is the automatically updating preview map.

Additionally, discussions about the unity of the layout of the configuration UI were held. The new UI should respect the existing layout and functionalities. This means that some the usability guidelines needed to be broken in order to honor the existing layout. For example, the usability guidelines do not recommend tabs in forms but because of the parent framework this rule was broken (Jarrett & Gaffney, 2011; 111-112). This should be fine since the end users have already gotten used by it in other plugins. But in this UI it was decided that numbering of the tabs makes the progress more simple. Additionally, descriptive tab texts were carefully chosen to represent the contents of each tab. In order to make things even more clear, It might make the selected tab more obvious if its color would change accordingly to the selected page. This would unfortunately again break the existing policy of the framework also so it was not applied even though it was originally recommended in some of the early layout propositions (See Appendix E).

From the very beginning an automatically updating preview of the animation was considered as a helpful addition to the configuration UI. The challenge was to make it fit to the layout so that it would not take too much room but would still be useful.

Next in the program some new layouts were created for the starting point of the layout discussion. These were created without coding a line and thus minimal time and effort were needed. One of them is represented in Figure 9. Some more mock ups created with an image editing software are available in Appendix E.



Luo zoomattava ja raahattava sääanimaatio valittavilla sääparametreilla ja karttatasoilla.

Määritä animaatiossa näkyvät arvot

Taustakartta	✓
Lämpötila	✓
Sademäärä	
Puuska	
Tuuli	
Tuuli (numerot)	
Tuuli (nuolet)	
Edustakartta	✓

Figure 9: One of the very first drafts for the layout.

In Figure 9 everything inside the red square belongs to the weather map design and configuration user interface. The outer part belongs to the Ilmanet's framework. This mock up was used to start the discussion of what to include and how to place them in the configuration UI. Not a single line of code had been written in this phase and even big alterations were easy to do.

4.2 Creating Test Plan and Executing Tests

It is beneficial to have the basic understanding of the functions and desired features of the UI before creating a test plan. This is why some mock ups were first drawn and discussed. After the project team shares a rough understanding about the UI the test plan can be written. Creating a test plan is a professional way of starting any test project. In this chapter an example of how to create a test plan for software project is presented.

The purpose of the testing was to discover which features the UI should have. Additionally, the purpose was to clarify how to develop the desired features so that the UI is effortless to use and does not require technical knowledge.

The test consisted of four different phases:

1. Test to find out which are the desired functions of the UI.
2. Test how the desired features should be implemented.
3. Test and evaluate if the functions have been implemented with the usability perspective in mind.
4. Test and evaluate the technical implementation and security.

First, the tester procured a group of testers. Usually, the more test persons there are the better. Of course, the time and resources are limiting factors. The procurement of the test persons started by sending an email to a group of potential test persons. The email consisted of a brief overview of the subject and emphasized that the testing was about the features and properties of the UI not the test persons themselves. In the end, a total of seven test persons were chosen inside the case organisation. The test persons had differing backgrounds from an Account Manager to the technical Chief of Group. In this case there was no need for external testers since the UI would only be used by those working in the case organization.

Testing: Phase 1

The tester started this test phase by describing that the goal of this phase was to design a UI for the weather map design and configuration. The purpose is to discover what the test subject thinks what the UI is about. So the first thing is to ask what does the test subject assume that the UI is capable of doing or what he/she expects to achieve with the UI. Secondly, the test subject was asked to describe the most important features of the UI. Thirdly, the tester asked how much of time the user is willing to spend to achieve whatever he/she is trying to achieve with the UI.

Overall, the test took around 30-60 minutes per test person. The test was arranged in a meeting room and the tester interviewed the test person while taking notes. No mock ups or other material was presented.

The above questions gave the tester valuable information of the desired features. Many times the desired features can be in contradiction with the willingness to spend time for a specific task. These kinds of methods may help the tester to decide which of the features and functionalities are the ones that are really needed.

The results of the test phase 1 were carefully studied and presented in the project meeting. After evaluating the results decisions were made for the future steps. The test results can be found in Appendices B, C and D.

Testing: Phase 2

In the second phase of testing the tester had created paper mock ups according to the test phase 1. In this phase the results of the previous phase were rendered as a visible layout. The layouts, or mock ups, were still very rough and just drafts without much of a structure or colors. The purpose was still to test the desired features and functionalities and how they should be implemented to the UI.

It is necessary to note that the test persons may be different from those in the previous phase. This is why the tester started this test phase again by explaining what the test was about. In this phase it was important to give the test person direct instructions of what to do. Each test person was tested separately in a quiet meeting room.

The test person was once again guided that the test was not about him/her but about the UI and that the test subject could not do mistakes. Additionally, the test person was told that the tester could not reply to the possible questions (unless the test person became totally confused of what to do next). It was crucial that the test person was asked to think out loud. He/she needed to speak as much as possible of everything running in his/her head so that the tester could follow his/her thinking process.

The test started by handing out the test person the first mock up layout of the configuration UI (see Figure 10). The tester then asked the test person to start creating a new weather map animation. The test person was asked to add a temperature layer to the animation. The tester then observed how and what the test subject did while writing notes. Secondly, the tester asked if the test person could add a new wind layer with wind arrow symbols to the animation. Again, the tester observed and made notes of what was happening and where the test person had problems. Finally, the tester asked the test person to centre the weather map so that the centre would be in Jyväskylä, Finland.

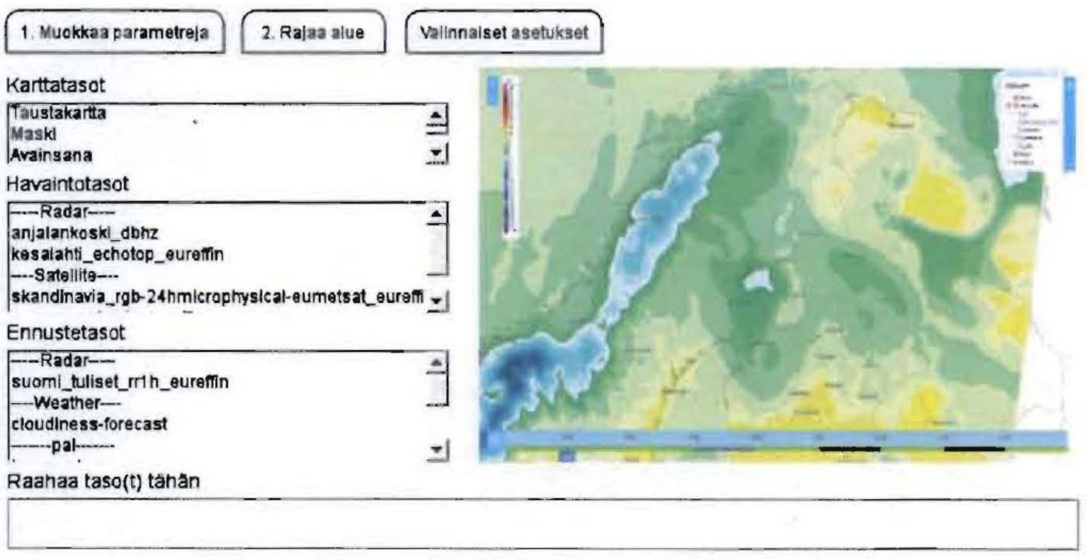


Figure 10: A mock up layout of the configuration UI.

The purpose of this test phase was to find out where and how the test subject started and what he/she did first. Additionally, it was interesting what the test subject did next and what was clear or unclear. The test subject was asked what were the features and functionalities needed and what were not. If the test subject was not speaking he/she was asked what was running in his/her mind right now and what were the things that caused the pause. The tester asked what was the thing the test person expected to happen when something that the test person did not expect happened.

This phase is one of the most time consuming but also the one that provides the most information. The tester presented the results acquired in this test phase to the project team. The results of were again extensively studied and the decisions made accordingly.

The rest of mock ups used to gather information may be found in Appendix F.

Testing: Phase 3

After the previous test phase the desired features and the functionalities of the UI were now known. Depending on the project one might want to have additional paper prototype tests here but for this project a working prototype of the UI was created according to the results of the previous phase.

The actual questions in this phase may be the ones used in phase 2 and those were the ones used in this phase. Again, valuable information was gathered. In this phase the possible changes were not that radical anymore but small improvements were made. Particularly, the main functionalities and features were already set. If one needed to do big modifications in this phase it would have meant that the previous phase had failed.

The final step was to do the reporting and present the results. After each phase the tester created a short summary of the results including the goals, scientific methods, logistics and the backgrounds of the testers. The tester then presented the results of the original questions to the project team by displaying the numerical data with graphics and provided the background of the questions and details. Finally, the results were discussed, recommendations were made and decisions of further testing were suggested.

Testing: Phase 4

In the final test phase the purpose is to validate the technical implementation and the security of the application. This phase requires technical UI and source code review. The actual testers should differ from the developer.

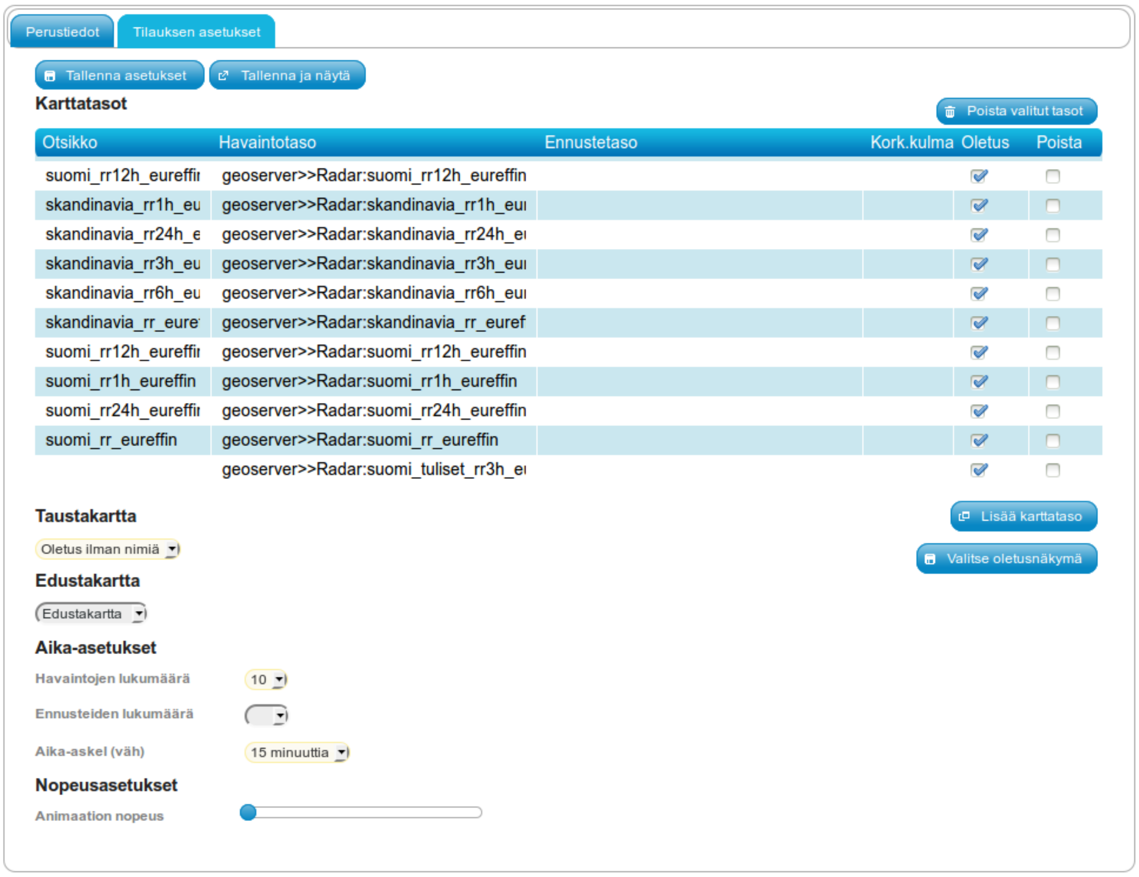
4.3 Development of User Interface

The target of this thesis was to design and develop a user interface for the weather map design and configuration. The UI was to be developed so that it would be easy-to-use even for the non-technical persons. This chapter describes the technical implementation of such a UI.

After the paper prototyping and decision making was over a functional UI was developed according to the results. Not all the wished features were implemented in the first phase but the ones that were considered the most important.

Since the UI should be implemented as a plugin for an existing framework called the Ilmanet the development started by copying an existing plugin called the MetOClient. Ilmanet has an API for new plugins. MetOClient administration UI (see Figure 11) is an earlier version of Animator (name of the plugin developed in this thesis). Because MetOClient administration section was developed rather quickly to basically demonstrate the client side functionality it lacked many of the required features but served as an excellent place holder plugin for a more developed plugin. With MetOClient it was easy to demonstrate the functionalities and requirements of the more enhanced weather animations. The development of MetOClient administration UI had not included usability testing and it was considered rather difficult to use by the Account Managers of the case organization.

Still, the work and solutions done in MetOClient helped the development of the Animator quite extensively. A lot of code had already been written in both PHP and in javascript.


Perustiedot | Tilauksen asetukset

Tallenna asetukset | Tallenna ja näytä

Karttatasot

Poista valitut tasot

Otsikko	Havaintotaso	Ennustetaso	Kork.kulma	Oletus	Poista
suomi_rr12h_eureffi	geoserver>>Radar:suomi_rr12h_eureffin			✓	☐
skandinavia_rr1h_eu	geoserver>>Radar:skandinavia_rr1h_eu			✓	☐
skandinavia_rr24h_e	geoserver>>Radar:skandinavia_rr24h_e			✓	☐
skandinavia_rr3h_eu	geoserver>>Radar:skandinavia_rr3h_eu			✓	☐
skandinavia_rr6h_eu	geoserver>>Radar:skandinavia_rr6h_eu			✓	☐
skandinavia_rr_eure	geoserver>>Radar:skandinavia_rr_euref			✓	☐
suomi_rr12h_eureffi	geoserver>>Radar:suomi_rr12h_eureffin			✓	☐
suomi_rr1h_eureffin	geoserver>>Radar:suomi_rr1h_eureffin			✓	☐
suomi_rr24h_eureffi	geoserver>>Radar:suomi_rr24h_eureffin			✓	☐
suomi_rr_eureffin	geoserver>>Radar:suomi_rr_eureffin			✓	☐
	geoserver>>Radar:suomi_tuliset_rr3h_e			✓	☐

Lisää karttataso

Valitse oletusnäkymä

Taustakartta

Oletus ilman nimiä

Edustakartta

Edustakartta

Aika-asetukset

Havaintojen lukumäärä	10
Ennusteiden lukumäärä	
Aika-askel (väh)	15 minuuttia

Nopeusasetukset

Animaation nopeus

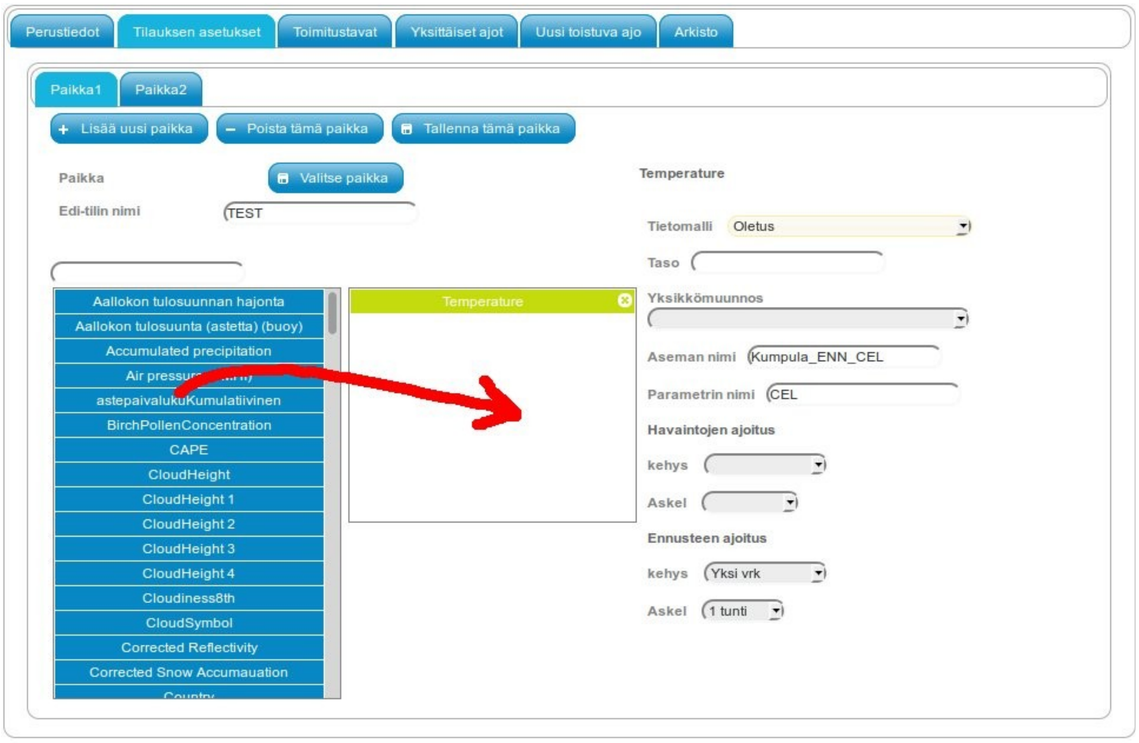
Figure 11: MetOclient configuration user interface.

After the existing MetOClient was copied and pasted as a new plugin called the Animator the UI development started. The goal here was to use the existing code as much as possible in order to speed up the development. This is why the client side experienced only a few modifications which will be discussed later in this chapter.

The development of the new plugin started by creating three different tab layouts according to the test results and the project team decisions. The first layout was responsible of majority of the weather animation configuration functionalities. The latter two are responsible mainly of the time and area selection of the animation. It is not usually a good practise to create form fields in separate tabs but since the function of each tab is quite distinct from each other it made more sense from the usability perspective. Additionally, the users of the Ilmatie are already used to tabs. The tabs were also emphasized by adding descriptive titles on each tab as well as adding numbering.

Rather than placing some uninformative titles such *Parameters*, *Time line* and *Area* more descriptive titles were placed. The titles included the main function of each tab in literal form: *1. Edit parameters*, *2. Adjust time line* and *3. Edit area*. This way it was much clearer what each tab was about. Additionally, the numbering helped the end user to follow the work flow of the configuration more easily.

Ilmanet relies heavily on javascript and is strongly event based. In the project meetings it had became apparent that if possible the javascript based Ilmanet shopping cart widget should be used as the selector for the different weather parameters. This is because its purpose and functionality were already known by the end users. An example of the shopping cart widget is displayed in Figure12. The original shopping cart widget consisted of only two areas: the one that holds all the available weather parameters and the one that serves as a place holder for the chosen parameters. The red arrow in Figure 13 illustrates how the weather parameter can be dragged from one area to the other. This functionality is used in many other plugins in Ilmatie which is why this method was considered as a good approach to select the parameters in the Animator plugin too.



Paikka

Edi-tilin nimi: TEST

Temperature

Tietomalli: Oletus

Taso:

Yksikkömuunnos:

Aseman nimi: Kumpula_ENN_CEL

Parametrin nimi: CEL

Havaintojen ajoitus

kehys:

Askel:

Ennusteen ajoitus

kehys: Yksi vrk

Askel: 1 tunti

Figure12: The shopping cart widget in another plugin.

It quite soon became clear that the implementation of the original shopping cart to the Animator plugin was not going to be effortless. This was because the new UI should have three separate areas from where to choose the available weather parameters in order to make the distinction of each group (maps, observations and forecasts) more clear (See Figure 13).

Additionally, the UI should have two different dropping areas, or baskets, for the chosen weather parameters: the one for the observations and another one for the forecasts. The existing shopping was not designed to have these kinds of functionalities. So, a decision had to be made whether to actually modify the existing shopping cart widget, or try to overcome the problems discovered in the Animator plugin. Although it probably would have been better for the overall development of the Ilmanet framework if the existing javascript code of the shopping cart widget was improved, it would have taken too much of time to confirm that all the other plugins were still functioning with the newly made editions. This is why it was decided that all the encountered problems should be handled in the Animator plugin rather than modifying the parent framework itself.

Because the animation is based on the open source javascript library OpenLayers it makes sense to call the weather parameters as layers too. The OpenLayers works by grouping different layers on top of each other. The layers may also be transparent so that the layers below are visible too. With this technology it is possible to display weather data like rain areas and contours on top of a map background.

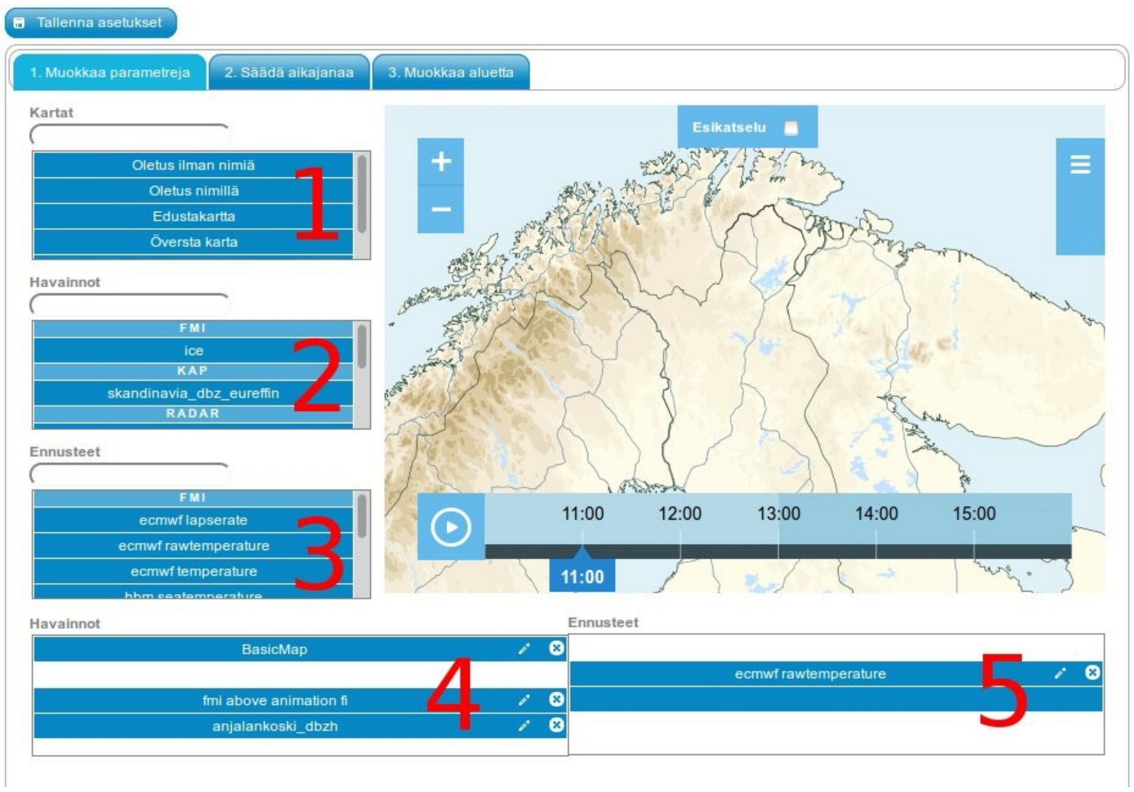


Figure 13: Animator design and configuration user interface.

In Figure 13 there are five different areas of interest marked with red numbers. The area **number 1** is for the background and foreground layers. These include the map backgrounds and for example the map borders and city names, layers which usually need to displayed on top of all other layers. Other possible layers in this area may include different kinds of overlay symbols or figures.

The area **number 2** is for weather observation layers and the area **number 3** is for forecast layers. Number 1 and 2 areas are also grouped by different producers like the weather stations, radars and models so that each parameter can be found more easily. All three areas include a search box that can be used to narrow down the parameter lists. All of the mentioned layers are fetched from the WMS application interface with a special *getCapabilites* request and rearranged properly.

The area **number 4** serves as a dropping area for the observation layers and area **number 5** serves as a dropping area for forecast layers. If observations are dragged to the forecast area they are disregarded and removed and the other way around. The areas 4 and 5 are placed side by side to make it appear more like a time line. Observations are first dragged and dropped on the left box and forecasts are dragged *after* them to the right box. This way the user should be able to associate that he/she can combine the observation and forecast layers together by placing them on the same level on both sides. If the observation and forecast parameters combine logically together the animation will first run smoothly through the observations and then continue directly to the forecasts.

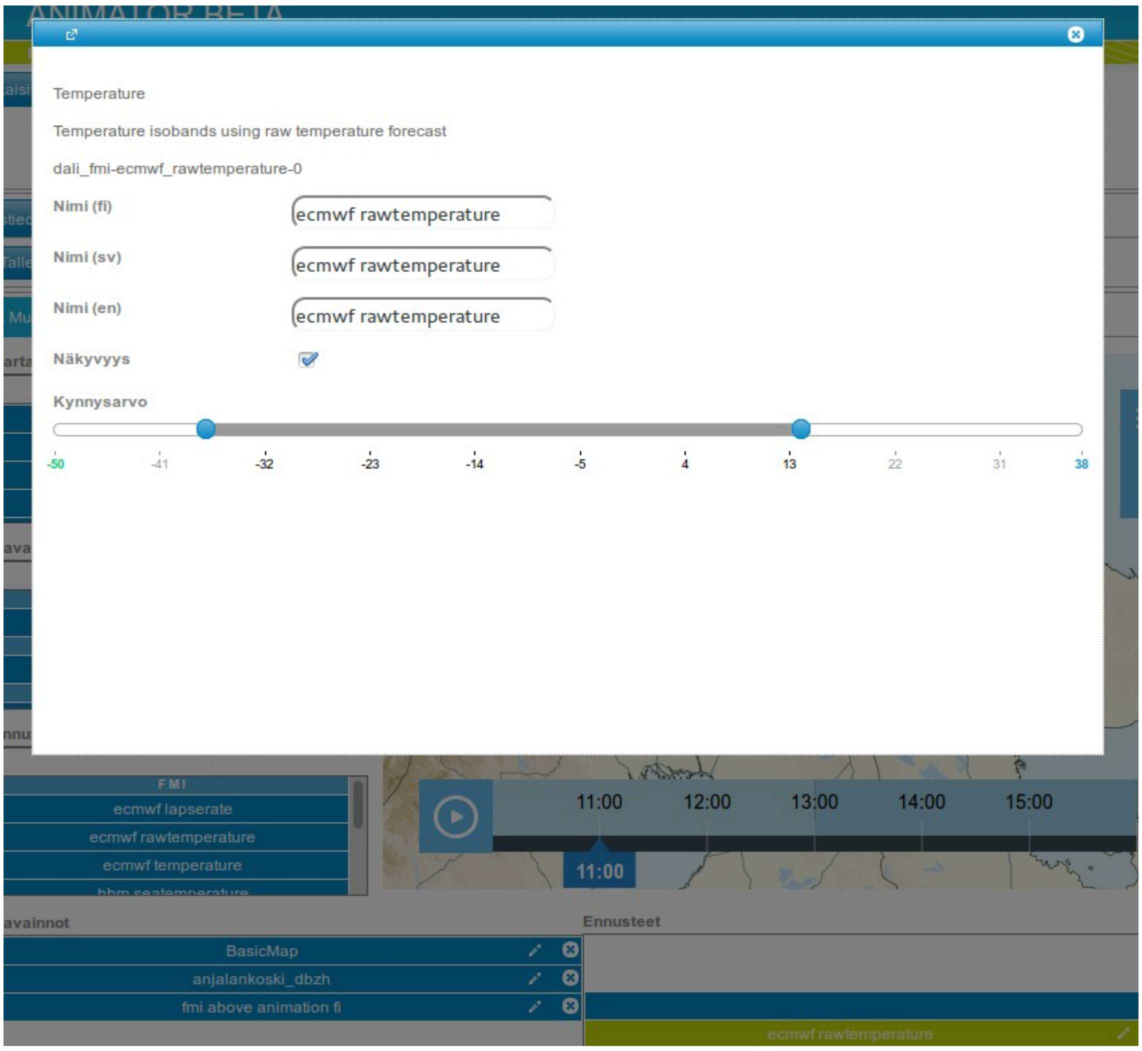
Each new line in the dropping areas represent a new layer in the animation. By default, the uppermost layer will be at the bottom in the animation. So it makes sense to start creating the animation with some background layer. Of course, the order of the layers can be later updated.

It should be noted that the animation is functional even if either one of the dropping areas is left blank. This just means that only either observations or forecasts are displayed.

When ever a new layer is dropped on the areas 4 or 5 an edit button (a pencil icon) is added to the layer. By clicking this icon the user can open a dialog window that serves as the layer property form (see Figure 14). In the dialog window the user can i.e. rename the layer, define the layer visibility in the client animation and adjust the thresholds for the parameter (if it is possible for the parameter). If a threshold is defined only the values between the threshold limits are displayed in the client animation for the parameter. In this case a set of special parameters are set to the configuration JSON which are delivered to Dali. Dali then clips the values outside of the threshold limits away. With this technology it is possible to i.e. display the temperature contours only in areas where the temperature is between 20-25 degrees Celsius.

Another icon which appears to each dropped layer is the delete button. The newly dropped layer is placed to the end of the area number 4 or 5 list from where it may be dragged to any place inside the area. When ever the dragging occurs the corresponding layer on the other side moves along.

The functionalities described earlier in this chapter were not something the shopping cart widget was designed to do so a lot of plugin specific code was needed to overcome the problems confronted to create the desired functionality. Most of them were solved by creating additional plugin specific javascript events and functions as well as AJAX calls.


Temperature

Temperature isobands using raw temperature forecast

dali_fmi-ecmwf_rawtemperature-0

Nimi (fi)	ecmwf rawtemperature
Nimi (sv)	ecmwf rawtemperature
Nimi (en)	ecmwf rawtemperature
Näkyvyys	✓

Kynnysarvo

Figure 14: Dialog window for forecast parameter properties.

On each tab a preview of the animation is displayed (see Figure 15). By default the animation is not updated every time when some modification is done but it may be enabled by checking the check box on the top of the preview area. This functionality removed the need to open the client animation every time. The preview is basically the client side animation but since it was not designed to be displayed on the administration side it required some additional programming.

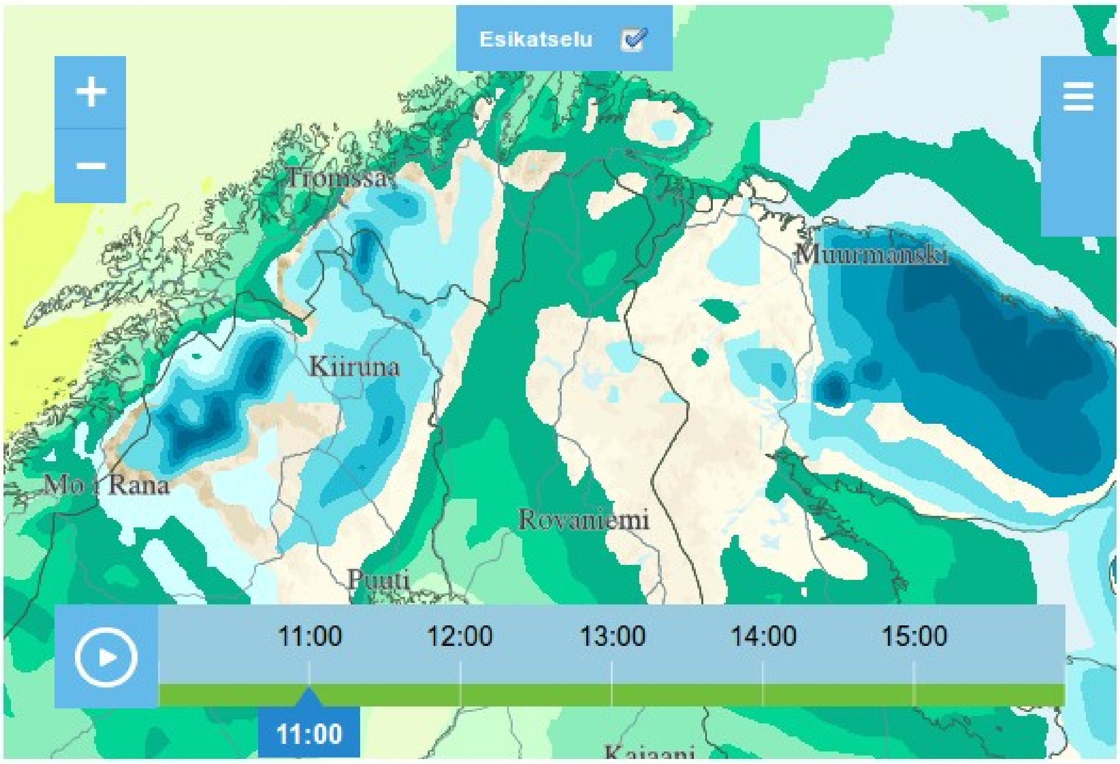
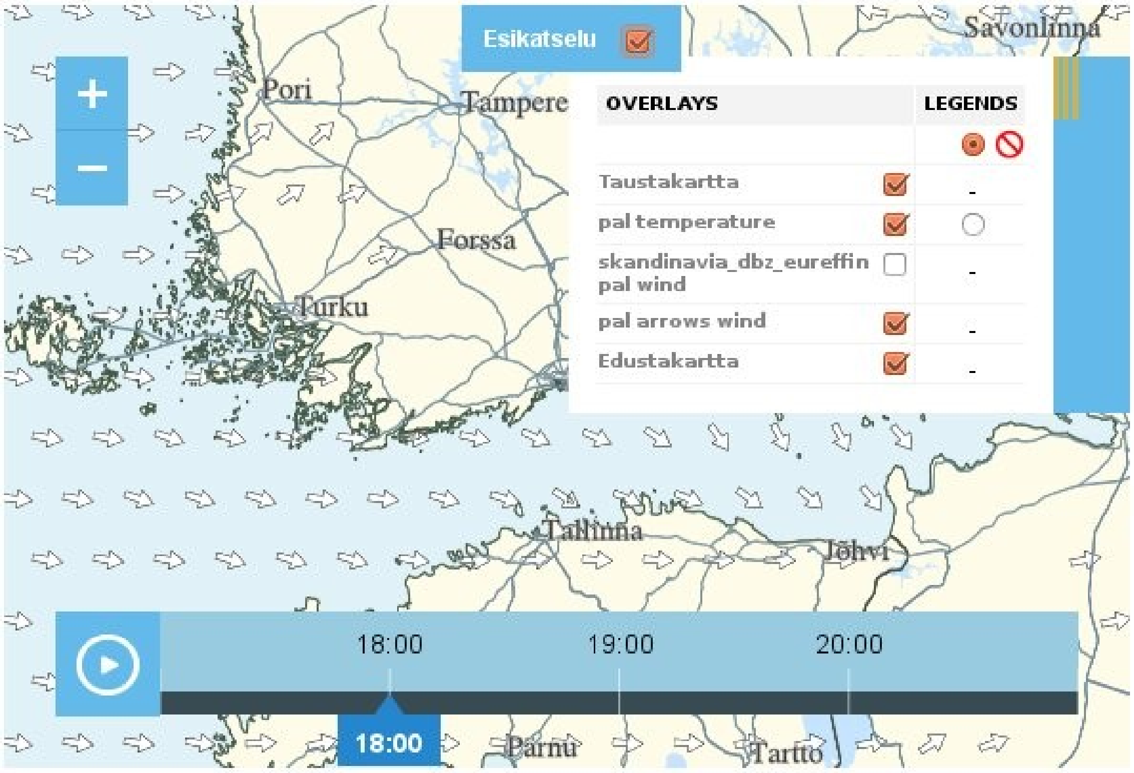


Figure 15: Preview of the animation.

In Figure 15 there are also a few other functionalities such as the time line at the bottom. This is the time line of the animation which the can be configured in the second tab of the configuration UI (see Figure 17). It is possible to configure the number of observations and forecasts displayed in the animation. Additionally, one can set the time step to something else than the default of one hour. Of course, these adjustments affect the time line correspondingly. It is not possible for now to adjust the time step separately between the observations and forecasts.

There is a layer switcher on the right side of the animation demonstrated in Figure 16. By clicking on this the UI displays a group of the layers configured for the animation.

The layers can be made hidden or visible by checking the visibility check box inside the switcher. Additionally, a special functionality to display the layer specific legend was added.

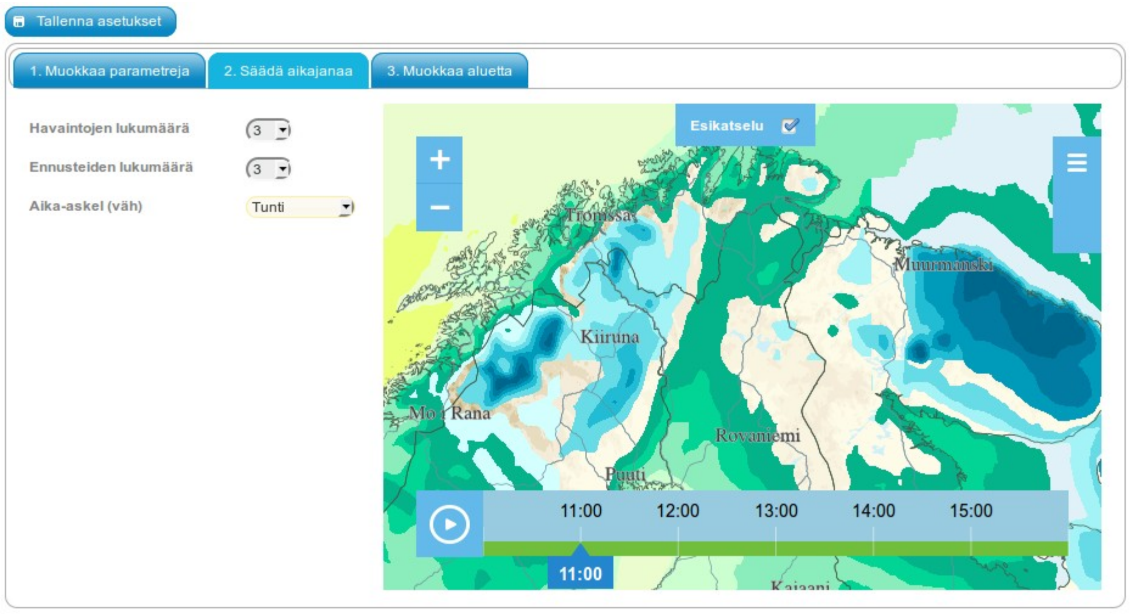


OVERLAYS		LEGENDS
		● 🚫
Taustakartta	✓	-
pal temperature	✓	○
skandinavia_dbz_eureffin pal wind	☐	-
pal arrows wind	✓	-
Edustakartta	✓	-

Figure 16: An example of the animation switcher content.

A special check box for the automatically updating preview was added to the configuration user interface. By checking it the user can see the changes made refresh automatically.

The following image (see Figure 17) illustrates the time line adjustment user configuration interface.

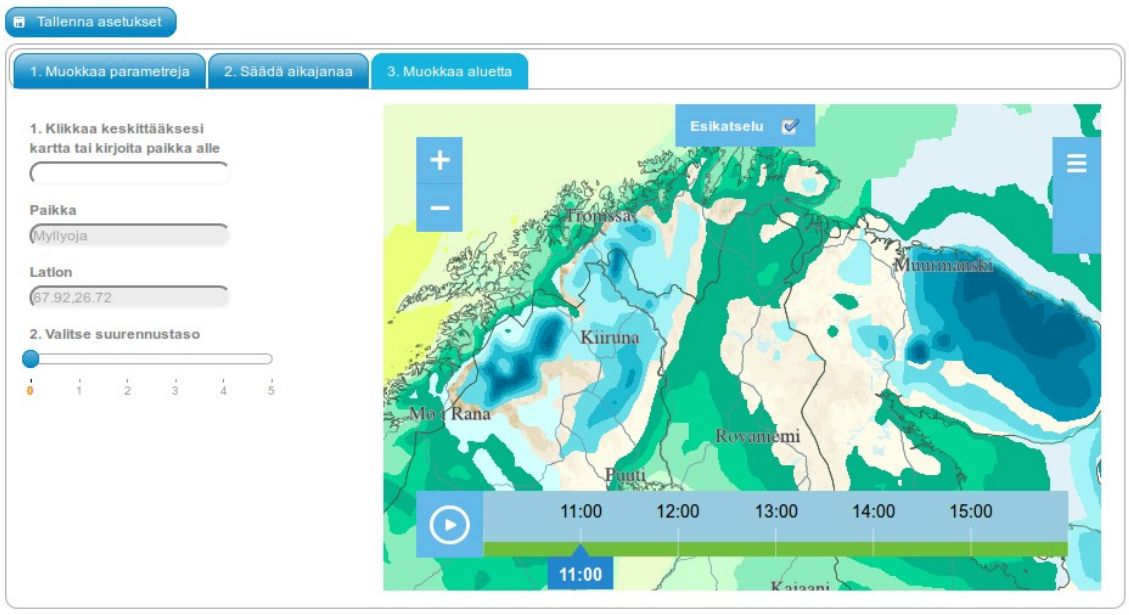


Havaintojen lukumäärä 3
Ennusteiden lukumäärä 3
Aika-askel (väh) Tunti

Figure 17: Animation time line adjustment user interface.

The need to set the default area for the animation was overcome by creating a tab where the configurator can click the preview area and centre the map to that exact point (see Figure 18). This was considered as an intuitive way of defining the centre location and it is emphasized by numbering the step and giving instructions on the page. Another way to centre the map is to type a location to the location input field and select a place from the special auto complete drop down. The case organization has developed an advanced auto complete functionality for location names. Unfortunately, within the given time span the already available auto complete widget could not be implemented here. Instead, the jQuery auto complete widget was used and modified.

When the user starts typing the location, the name and coordinates are then displayed below. Currently the location cannot be defined by providing the latitude and longitude coordinates directly. The second numbered step is for defining the default zoom level of the animation. Sometimes the configurator may want to zoom to a smaller area and sometimes he/she may want to display the whole Scandinavia by default. Here, the smaller number in the zoom level indicates a broader view. To make things even more clear the configurator may click the plus and minus symbols on the left side of the preview area too and the zoom level is set correspondingly (see Figure 15).


Tallenna asetukset

1. Muokkaa parametreja | 2. Säädä aikajanaa | 3. Muokkaa aluetta

1. Klikkaa keskittääksesi kartta tai kirjoita paikka alle

Paikka: Myllyoja

Latlon: 67.92,26.72

2. Valitse suurennustaso: 0 1 2 3 4 5

Figure 18: Animation area adjustment user interface.

The default way to store the chosen configuration properties is to save them into the MySQL database. The default way of saving the form did not work in this scenario because it included so many modifications and a specific saving functionality needed to be created.

5 Testing and Evaluation of Solution

In this section a selected set of the test results of the published application are given and evaluated. At this point the application had been used by the end users for a few months.

Since the new user interface was published the continuous testing has revealed several small areas that require improvements. Some of the feedback is targeted to the parent framework which is more difficult to change but is still valuable information for the developers.

While the improvements in the new user interface where desperately needed and the UI is appreciated certain things still bother the end users. The dragging of the layers to the correct areas is still not as self-evident as it should be. This functionality can definitely be learned but if one wants to develop an easy to use user interface it should be clear. Additionally, when adding a new layer to the observation or forecast area the layer should go directly where it is dropped. Now, because of technical issues, the layer item is dropped at the end of the list from where it can be dragged to an another position.

The preview was considered to be awesome in the paper prototyping phase but in reality it turned out to be too heavy to be instantly updated. It was just too time consuming and even frustrating to wait for the animation to be reloaded with new properties. This is why an additional functionality was added to shut down the preview functionality by default. The end users also wished that the preview area would be bigger but it is quite difficult to achieve this with the current properties of the parent layout.

Originally adjusting the time line was supposed to be totally redesigned. Unfortunately, due to the lack of time the functionality had to be copied from existing plugins. The usability of the feature is definitely not as intuitive as it should be. Additionally, a feature for setting the time step separately for each observation and forecast layers was requested but not yet implemented.

Currently, the default centre location can only be defined by either clicking on the map or typing the location to the auto complete input field. Providing the latitude and longitude coordinates directly is not yet supported.

The loading time of the animation was greatly criticised. This latency is not really a problem of the map configuration user interface but a bigger problem in the background system. The production system is not yet fully developed and requires further optimization in order the make the loading times more reasonable.

The final technical testing was done by a couple of colleagues. They gave a few technical improvement suggestions that will be implemented in the upcoming releases. The security level of the application was found to be sufficient for this kind of service.

6 Discussion and Conclusions

The goal of this thesis was to develop a user interface for designing and configuring weather map animations. The UI needed to be effortless to use even for the non-technical end users. Additionally, the UI had to comply with the parent framework Ilmanet. A considerable time was consumed to gather information on what were the real user requirements. The initial perceptions differentiated quite a lot from the end result. This is not a negative thing because the project revealed several things that were not known by anyone in the early phases of the project. This shows that many times in the early phases of a software development project it is often unknown what features and requirements of the software are really needed.

As to all application development projects none of them are ever really finished. It is a continuation of cyclic periods of testing, feature requests and development. Even though there are predefined models and examples for designing and developing user friendly interfaces all user interfaces are one of a kind and compromises are usually needed. That being said, it is crucial to research the best practises in the user experience and usability literature in order to avoid the major bottlenecks in the user interface development. Usually, it is still easier to say what one should not do than to recommend what to do. This is why testing is still so important in each project. Even a little bit of testing is better than none at all. The tester can learn considerably even by researching just one of the users trying to achieve something with the user interface.

In this thesis the goal was to develop a weather map design and configuration user interface that was effortless to use even for the non-technical end users. Because short testing phases were included in the project many obstacles were avoided and only the really needed features were implemented at first. Continuous testing was done through out the project and the end users were kept up to date. This saved a lot of development time and frustration levels remained minimal for both the developer and end users. Everybody knew what was going to happen and which features were to be implemented first.

The case organization had not previously done a lot of testing especially in advance so this was one of the first projects in that area. Overall, the feedback was very positive because the actual end users had the chance to be part of the project and to have an influence on how and what was to be developed. It really makes no sense to develop something which no one likes or wants to use.

In the end a working and live UI was published. Many compromises had to be made on the way because of the limitations of the existing parent framework but in overall the result was considerably better than it would have been without gathering the initial information and the usability testing. If the work had been done without any testing a lot of unnecessary features would have been developed. This would have consumed a lot of resources and time to things that were not really needed.

Ultimately, the processes and methods described in this thesis should enable one to design and develop a user interface that is effortless to use and requires no technical experience. Of course, each project is unique and requires an adaptive approach especially when creating the test plan.

Overall, the methods used in this thesis have received a considerable amount of acknowledgement. The methods are not new per se but most of them have not been applied widely in the case organization. The testing methods used in this thesis were not that highly sophisticated yet because of the inexperience of the tester. To become an expert more time, knowledge and experience is needed.

But as a consequence, the results of this study support the use of similar methods in the case organization in the possible future projects too.

7 References

Allen, Jesmond & Chudley, James (2012), *Smashing UX Design, Foundations For Designing Online User Experiences.* West Sussex (UK): John Wiley & Sons Ltd.

Animbrowser website (2015), [internal website], URL: http://weather.weatherproof.fi/animbrowser. Accessed 8th November 2015.

Eisenberg, David J. (2002), *Producing Scalable Vector Graphics with XML*. SVG Essentials. Sebastopol (USA): O'Reilly Media.

Geoserver*, Geoserver* [online], URL: http://docs.geoserver.org. Accessed 9th November 2015.

Heiskanen, Mika, *Brainstorm Dali JSON Reference* [internal document]. Accessed 8th November 2015.

Jarrett, Caroline & Gaffney, Gerry (2011), *Forms That Work - Designing Web Forms for Usability.* Goydon (UK): Elsevier Ltd.

Krug, Steve (2014), *Don't Make Me Think, Revisited - A Common Sense Approach to Web Usability.* USA: New Riders.

Nielsen, Jakob (1993), *Usability Engineering*. London: Elsevier Ltd.

Nielsen, Jakob (1999), *Do Interface Standards Stifle Design Creativity?* [online], URL: https://www.nngroup.com/articles/do-interface-standards-stifle-design-creativity. Accessed 10th January 2015.

Open Geospatial Consortium (OGC), *Web Map Service* [online], URL: http://www.opengeospatial.org/standards/wms. Accessed 6th November 2015.

Open Source Geospatial Foundation, OpenLayers: Free Maps for the Web [online]. URL: *http://openlayers.org/two*. Accessed 11th November 2015.

Rubin, Jeff & Chisnell, Dana (2008), *Handbook of Usability Testing, Second Edition: How to Plan, Design, and Conduct Effective Test.* Indianapolis: Wiley Publishing, Inc.

Snyder, Carolyn (2003), *Paper Prototyping: Fast and Simple Techniques for Designing and Refining the User Interface*. USA: Morgan Kaufmann.

Tervo, Roope (2011), *Master's Thesis: Product Definition Mechanism and High Level Architecture of FMI Weather Map Service.*

TheKarppinen*, MetOClient UI* [online]. URL: https://github.com/fmidev/metoclient-ui. Accessed 8th November 2015.

Tidwell, Jenifer (2011), *Designing Interfaces, Second Edition*. Canada: O'Reilly.

Tullis, Por Thomas & Albert, William (2008), *Measuring the User Experience: Collecting, Analyzing, and Presenting.* Burlington: Elsevier Ltd.

von Fieandt, Kai (1972), *Havaitsemisen maailma.* Porvoo: Werner Söderström.

W3C*, SVG Tutorial* [online], URL: http://www.w3schools.com/svg/. Accessed 7th November 2015.

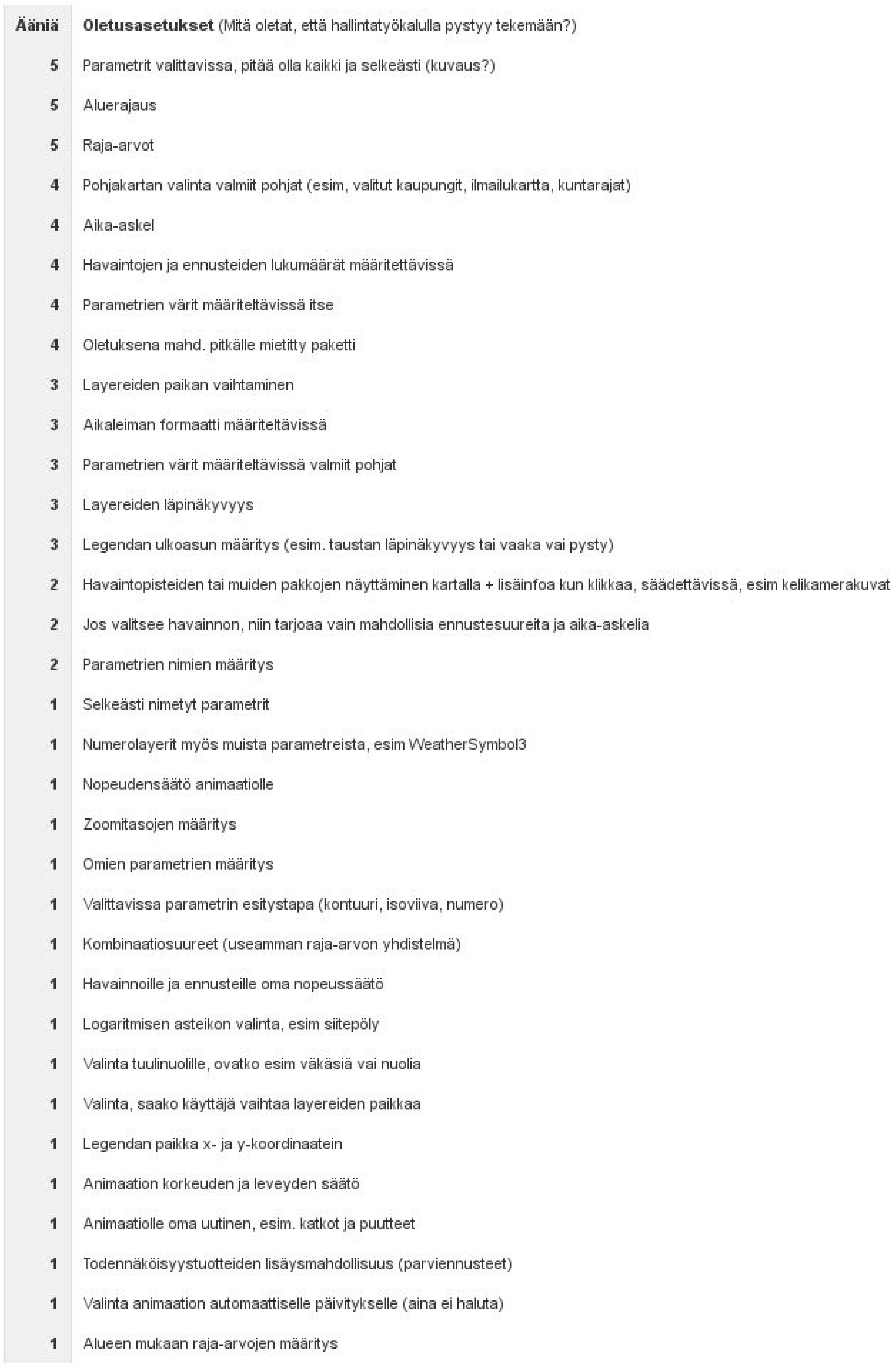
Wroblewski, Luke (2008), *Web Form Design: Filling in the Blanks.*
Tourangeau & Rips & Rasinski (2000), *The Psychology of Survey Response*. USA: Rosenfeld Media.

An Example of Configuration JSON

```
{
  map: {
    className: 'OpenLayers.Map',
    args: [
      {
        allOverlays: true,
        projection: 'EPSG:3067',
        units: 'm',
        resolutions: [
          2048,
          1024,
          512,
          256,
          128,
          64
        ],
        maxExtent: [
          -4537345.568,
          3840856.936,
          2889342.313,
          8254755.58
        ]
      }
    ]
  },
  layers: [
    {
      className: 'OpenLayers.Layer.WMTS',
      args: [
        {
          name: 'Taustakartta',
          url: [
            'http:\/\/1.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/gwc\/service\/wmts',
            'http:\/\/2.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/gwc\/service\/wmts',
            'http:\/\/3.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/gwc\/service\/wmts',
            'http:\/\/4.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/gwc\/service\/wmts'
          ],
          format: 'image\/png',
          layer: 'KAP:Europe_Basic_NoNames',
          buffer: 0,
          style: '',
          isBaseLayer: true,
          matrixSet: 'ETRS-TM35FIN',
          matrixIds: [
            {
              identifier: 'ETRS-TM35FIN:2'
            },
            {
              identifier: 'ETRS-TM35FIN:3'
            },
            {
              identifier: 'ETRS-TM35FIN:4'
            },
            {
              identifier: 'ETRS-TM35FIN:5'
            },
            {
              identifier: 'ETRS-TM35FIN:6'
            },
            {
              identifier: 'ETRS-TM35FIN:7'
            }
          ]
        }
      ]
    },
    {
      className: 'OpenLayers.Layer.Animation.Wms',
      capabilities: {
        url: '\/\/ilmatie.ilmanet.fi\/geoserver\/wms',
        layer: 'KAP:BasicMap'
      },
      args: [
        'BasicMap',
        [
          'http:\/\/1.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/2.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/3.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/4.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms'
        ],
        {
          hasLegend: false,
          layers: 'KAP:BasicMap',
          tiled: true,
          version: '1.1.1'
        },
        {
          buffer: 0,
          animation: {
            hasLegend: false,
            fadeIn: {
              time: 0
            },
            fadeOut: {
              time: 0
            },
            isForecast: false,
            endTime: 'auto',
            autoLoad: true
          }
        }
      ]
    },
    {
      className: 'OpenLayers.Layer.Animation.Wms',
      capabilities: {
        url: '\/\/ilmatie.ilmanet.fi\/dali\/wms',
        layer: 'fmi:ecmwf:rawtemperature'
      },
      args: [
        'ecmwf rawtemperature',
        [
          'http:\/\/1.p.data.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/wms',
          'http:\/\/2.p.data.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/wms',
          'http:\/\/3.p.data.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/wms',
          'http:\/\/4.p.data.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/wms'
        ],
        {
          hasLegend: false,
          layers: 'fmi:ecmwf:rawtemperature',
          tiled: true,
          version: '1.3.0'
        },
        {
          buffer: 0,
          animation: {
            hasLegend: false,
            fadeIn: {
              time: 0
            },
            fadeOut: {
              time: 0
            },
            isForecast: true,
            endTime: undefined,
            autoLoad: true
          }
        }
      ]
    },
    {
      className: 'OpenLayers.Layer.Animation.Wms',
      capabilities: {
        url: '\/\/ilmatie.ilmanet.fi\/geoserver\/wms',
        layer: 'KAP:fmi_above_animation_fi'
      },
      args: [
        'fmi above animation fi',
        [
          'http:\/\/1.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/2.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/3.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/4.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms'
        ],
        {
          hasLegend: false,
          layers: 'KAP:fmi_above_animation_fi',
          tiled: true,
          version: '1.1.1'
        },
        {
          buffer: 0,
          animation: {
            hasLegend: false,
            fadeIn: {
              time: 0
            },
            fadeOut: {
              time: 0
            },
            isForecast: false,
            endTime: 'auto',
            autoLoad: true
          }
        }
      ]
    },
    {
      className: 'OpenLayers.Layer.Animation.Wms',
      capabilities: {
        url: '\/\/ilmatie.ilmanet.fi\/geoserver\/wms',
        layer: 'Radar:anjalankoski_dbzh'
      },
      args: [
        'anjalankoski_dbzh',
        [
          'http:\/\/1.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/2.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/3.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms',
          'http:\/\/4.p.wms.fmi.fi\/fmi-apikey\/f01a92b7-c23a-47b0-
95d7-cbcb4a60898b\/geoserver\/wms'
        ],
        {
          hasLegend: false,
          layers: 'Radar:anjalankoski_dbzh',
          tiled: true,
          version: '1.1.1'
        },
        {
          buffer: 0,
          animation: {
            hasLegend: false,
            fadeIn: {
              time: 0
            },
            fadeOut: {
              time: 0
            },
            isForecast: false,
            endTime: 'auto',
            autoLoad: true
          }
        }
      ]
    }
```

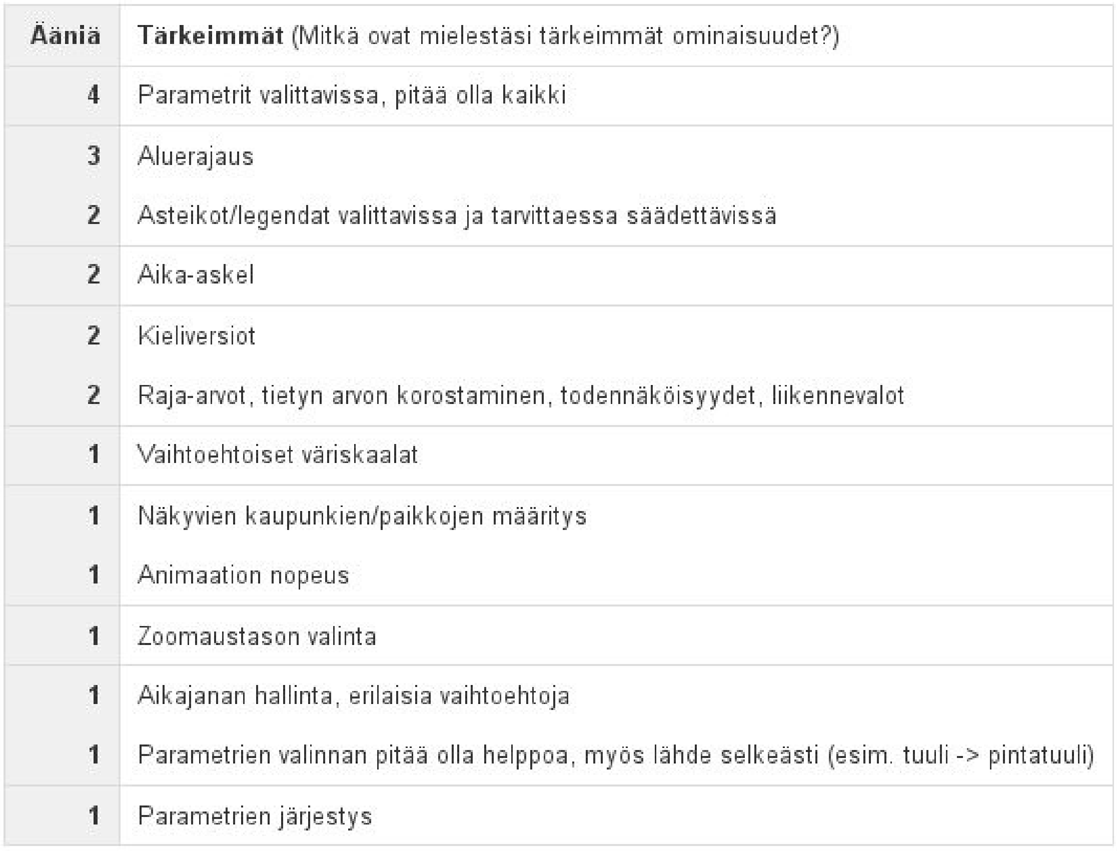
```
  ],
  showAnimationInitProgress: true,
  showAnimationLoadProgress: true,
  defaultZoomLevel: 6,
  animationRefreshInterval: 3600000,
  animationFrameRate: 500,
  animationResolutionTime: 3600000,
  animationDeltaToBeginTime: 7200001,
  animationDeltaToEndTime: 7200001,
  browserNotSupportedInfo: 'Selain ei ole tuettu'
}
```

List of features desired in the UI



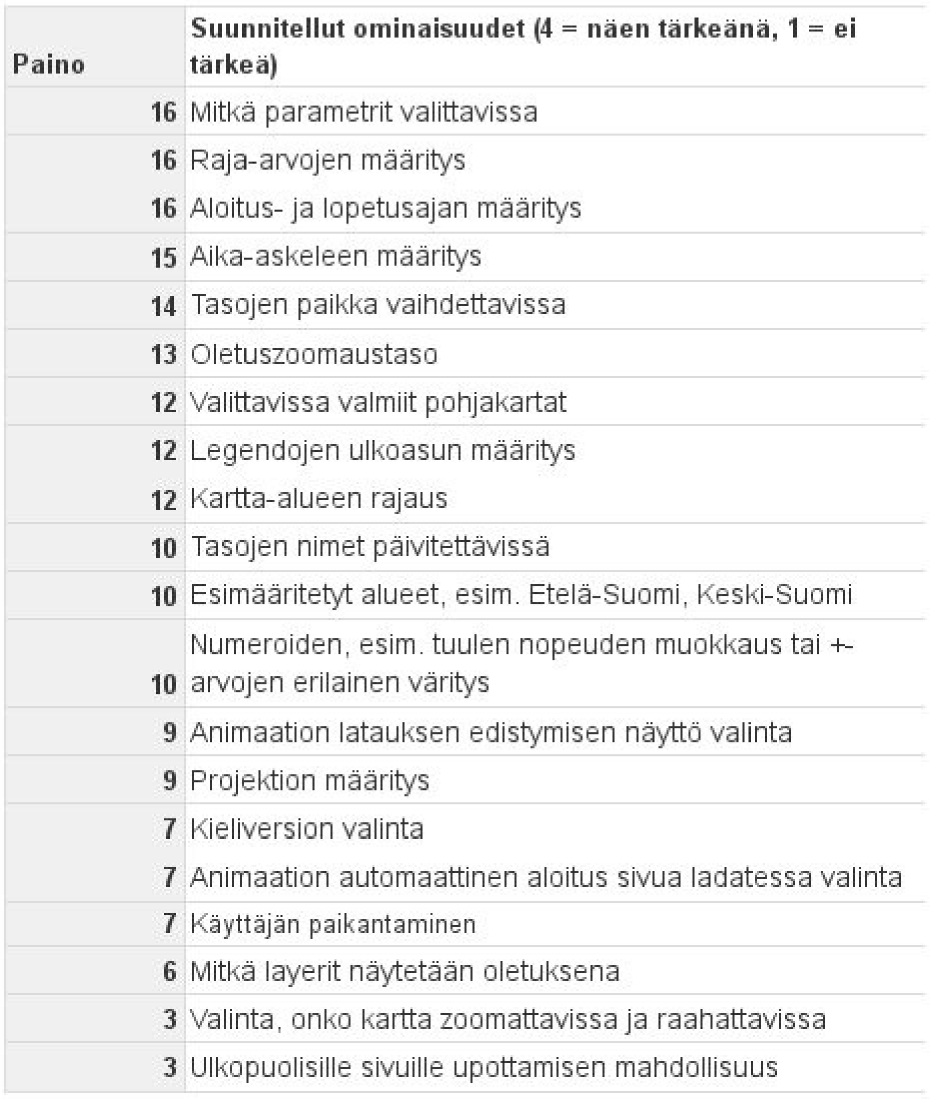
Ääniä	**Oletusasetukset** (Mitä oletat, että hallintatyökalulla pystyy tekemään?)
5	Parametrit valittavissa, pitää olla kaikki ja selkeästi (kuvaus?)
5	Aluerajaus
5	Raja-arvot
4	Pohjakartan valinta valmiit pohjat (esim, valitut kaupungit, ilmailukartta, kuntarajat)
4	Aika-askel
4	Havaintojen ja ennusteiden lukumäärät määritettävissä
4	Parametrien värit määriteltävissä itse
4	Oletuksena mahd. pitkälle mietitty paketti
3	Layereiden paikan vaihtaminen
3	Aikaleiman formaatti määriteltävissä
3	Parametrien värit määriteltävissä valmiit pohjat
3	Layereiden läpinäkyvyys
3	Legendan ulkoasun määritys (esim. taustan läpinäkyvyys tai vaaka vai pysty)
2	Havaintopisteiden tai muiden pakkojen näyttäminen kartalla + lisäinfoa kun klikkaa, säädettävissä, esim kelikamerakuvat
2	Jos valitsee havainnon, niin tarjoaa vain mahdollisia ennustesuureita ja aika-askelia
2	Parametrien nimien määritys
1	Selkeästi nimetyt parametrit
1	Numerolayerit myös muista parametreista, esim WeatherSymbol3
1	Nopeudensäätö animaatiolle
1	Zoomitasojen määritys
1	Omien parametrien määritys
1	Valittavissa parametrin esitystapa (kontuuri, isoviiva, numero)
1	Kombinaatiosuureet (useamman raja-arvon yhdistelmä)
1	Havainnoille ja ennusteille oma nopeussäätö
1	Logaritmisen asteikon valinta, esim siitepöly
1	Valinta tuulinuolille, ovatko esim väkäsiä vai nuolia
1	Valinta, saako käyttäjä vaihtaa layereiden paikkaa
1	Legendan paikka x- ja y-koordinaatein
1	Animaation korkeuden ja leveyden säätö
1	Animaatiolle oma uutinen, esim. katkot ja puutteet
1	Todennäköisyystuotteiden lisäysmahdollisuus (parviennusteet)
1	Valinta animaation automaattiselle päivitykselle (aina ei haluta)
1	Alueen mukaan raja-arvojen määritys

List of important features in the UI



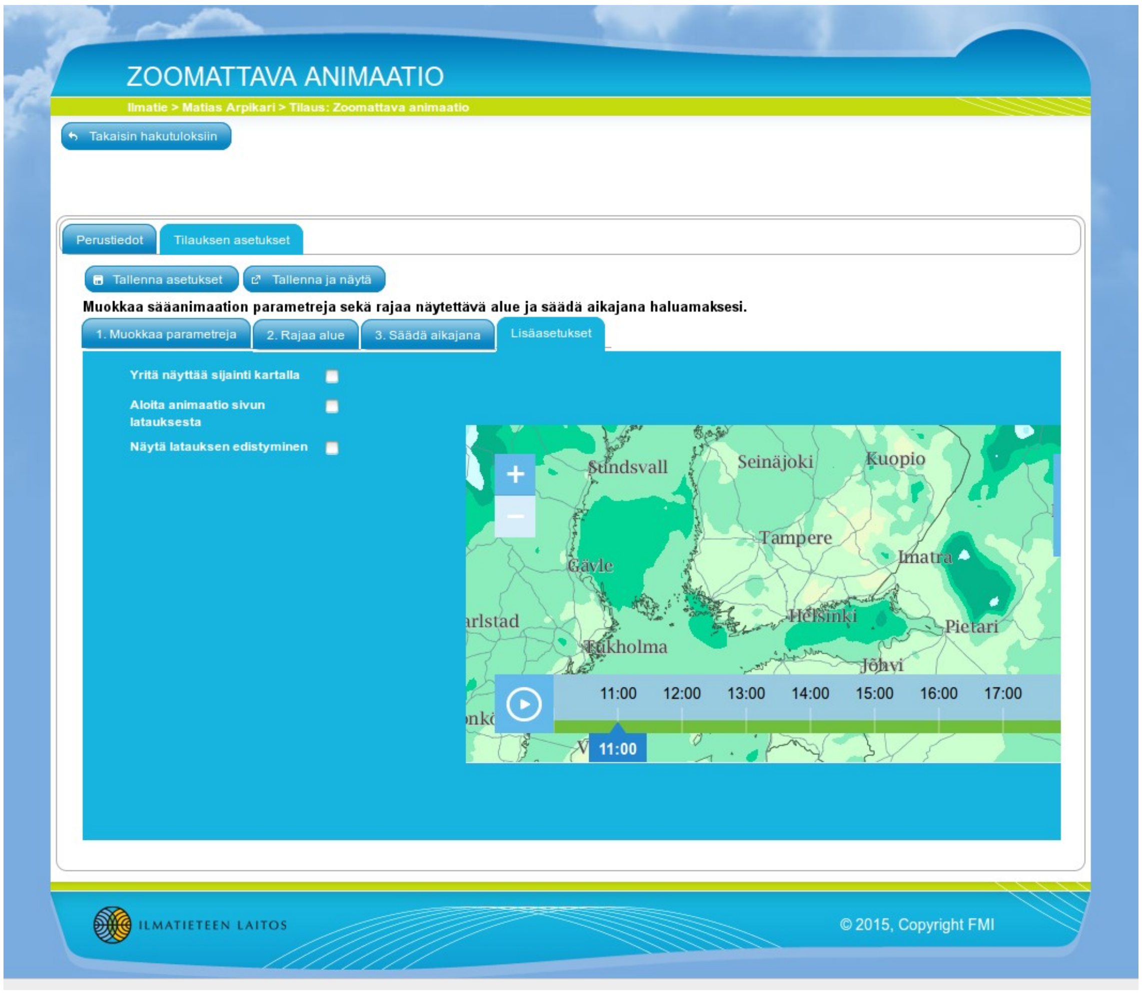
Ääniä	**Tärkeimmät** (Mitkä ovat mielestäsi tärkeimmät ominaisuudet?)
4	Parametrit valittavissa, pitää olla kaikki
3	Aluerajaus
2	Asteikot/legendat valittavissa ja tarvittaessa säädettävissä
2	Aika-askel
2	Kieliversiot
2	Raja-arvot, tietyn arvon korostaminen, todennäköisyydet, liikennevalot
1	Vaihtoehtoiset väriskaalat
1	Näkyvien kaupunkien/paikkojen määritys
1	Animaation nopeus
1	Zoomaustason valinta
1	Aikajanan hallinta, erilaisia vaihtoehtoja
1	Parametrien valinnan pitää olla helppoa, myös lähde selkeästi (esim. tuuli -> pintatuuli)
1	Parametrien järjestys

List of planned features in the UI



Paino	Suunnitellut ominaisuudet (4 = näen tärkeänä, 1 = ei tärkeä)
16	Mitkä parametrit valittavissa
16	Raja-arvojen määritys
16	Aloitus- ja lopetusajan määritys
15	Aika-askeleen määritys
14	Tasojen paikka vaihdettavissa
13	Oletuszoomaustaso
12	Valittavissa valmiit pohjakartat
12	Legendojen ulkoasun määritys
12	Kartta-alueen rajaus
10	Tasojen nimet päivitettävissä
10	Esimääritetyt alueet, esim. Etelä-Suomi, Keski-Suomi
10	Numeroiden, esim. tuulen nopeuden muokkaus tai +- arvojen erilainen väritys
9	Animaation latauksen edistymisen näyttö valinta
9	Projektion määritys
7	Kieliversion valinta
7	Animaation automaattinen aloitus sivua ladatessa valinta
7	Käyttäjän paikantaminen
6	Mitkä layerit näytetään oletuksena
3	Valinta, onko kartta zoomattavissa ja raahattavissa
3	Ulkopuolisille sivuille upottamisen mahdollisuus

Early mock ups of the UI



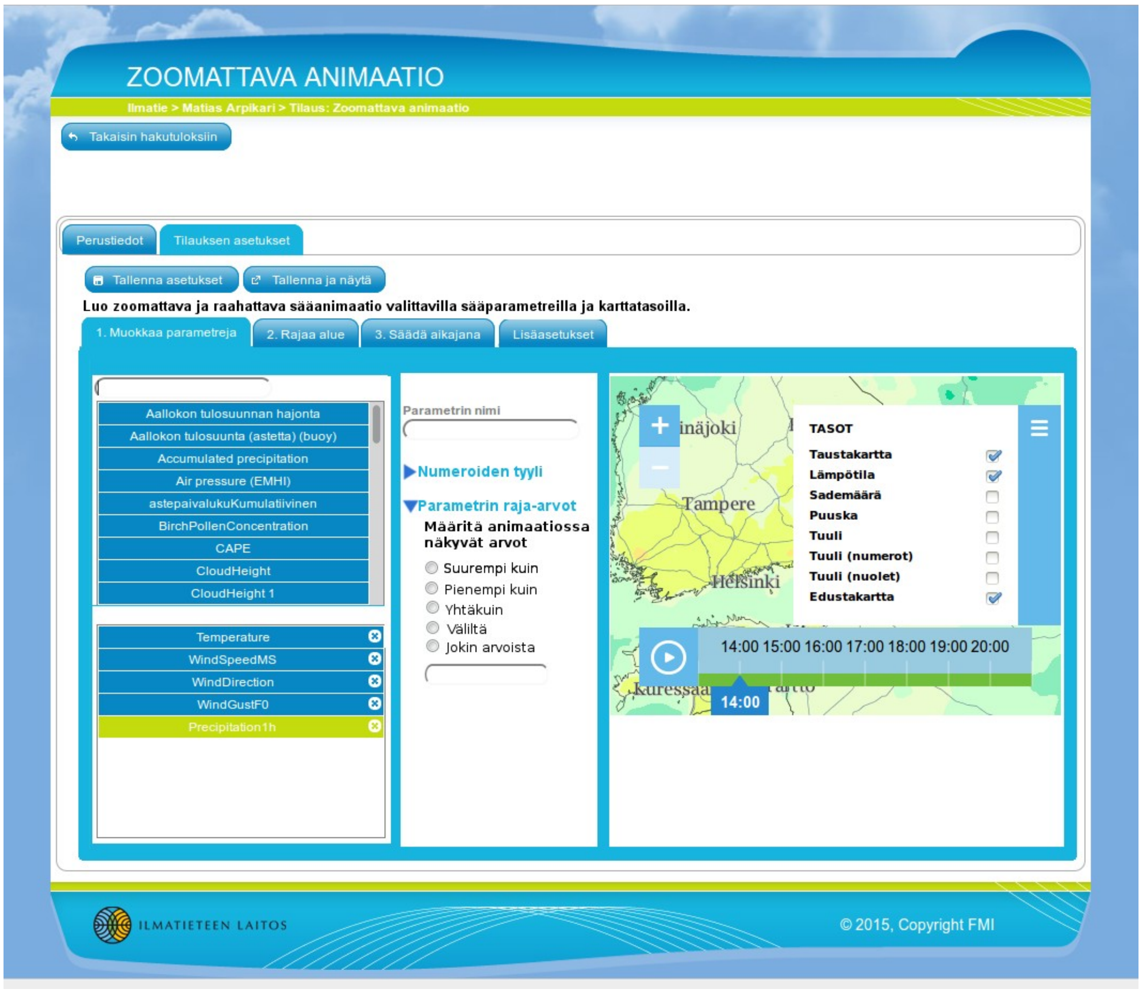
Muokkaa sääanimaation parametreja sekä rajaa näytettävä alue ja säädä aikajana haluamaksesi.

Yritä näyttää sijainti kartalla ☐

Aloita animaatio sivun latauksesta ☐

Näytä latauksen edistyminen ☐

Mock up 1: A clear way to present the selected tab with matching background colors.



ZOOMATTAVA ANIMAATIO

Ilmatie > Matias Arpikari > Tilaus: Zoomattava animaatio

Takaisin hakutuloksiin

Perustiedot | Tilauksen asetukset

Tallenna asetukset | Tallenna ja näytä

Luo zoomattava ja raahattava sääanimaatio valittavilla sääparametreilla ja karttatasoilla.

1. Muokkaa parametreja | 2. Rajaa alue | 3. Säädä aikajana | Lisäasetukset

- Aallokon tulosuunnan hajonta
- Aallokon tulosuunta (astetta) (buoy)
- Accumulated precipitation
- Air pressure (EMHI)
- astepaivalukuKumulatiivinen
- BirchPollenConcentration
- CAPE
- CloudHeight
- CloudHeight 1

- Temperature
- WindSpeedMS
- WindDirection
- WindGustF0
- Precipitation1h

Parametrin nimi

▶Numeroiden tyyli

▼Parametrin raja-arvot

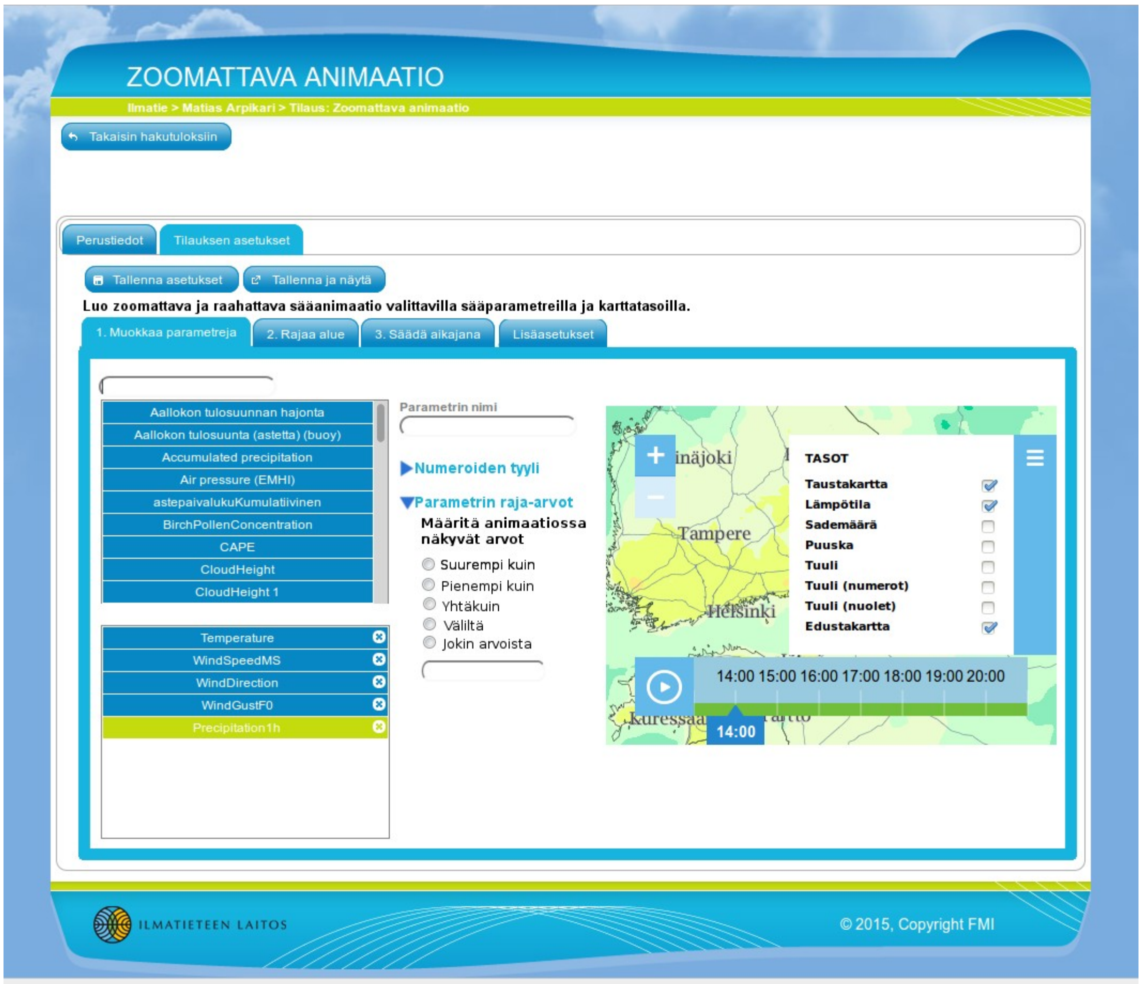
Määritä animaatiossa näkyvät arvot

- ○ Suurempi kuin
- ○ Pienempi kuin
- ○ Yhtäkuin
- ○ Väliltä
- ○ Jokin arvoista

TASOT

Taustakartta	✓
Lämpötila	✓
Sademäärä	☐
Puuska	☐
Tuuli	☐
Tuuli (numerot)	☐
Tuuli (nuolet)	☐
Edustakartta	✓

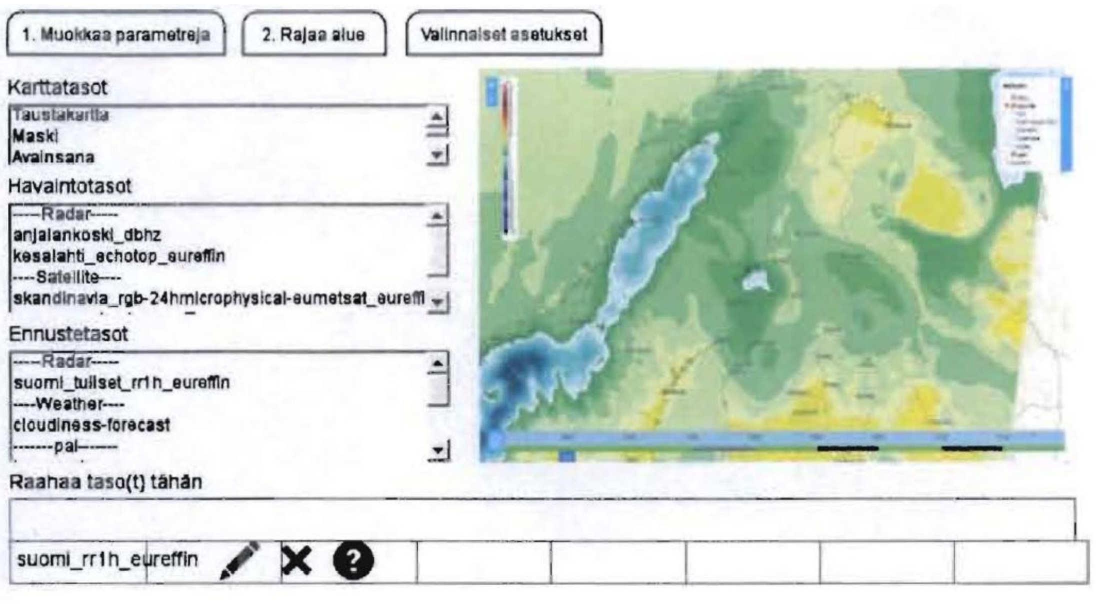
Mock up 2: An alternative way to display the selected tab and grouping of different features.

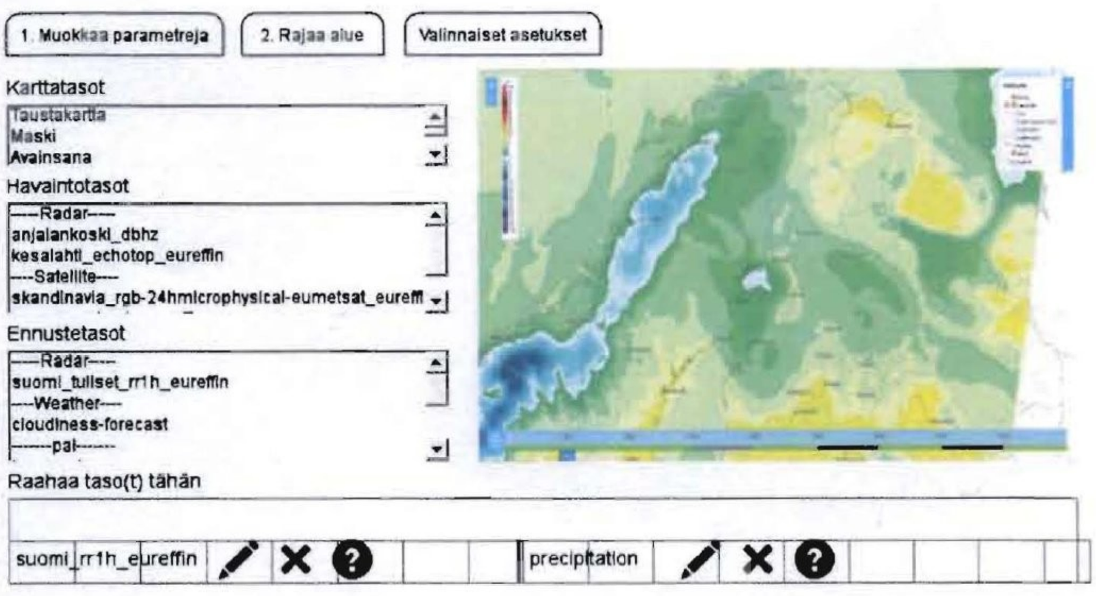


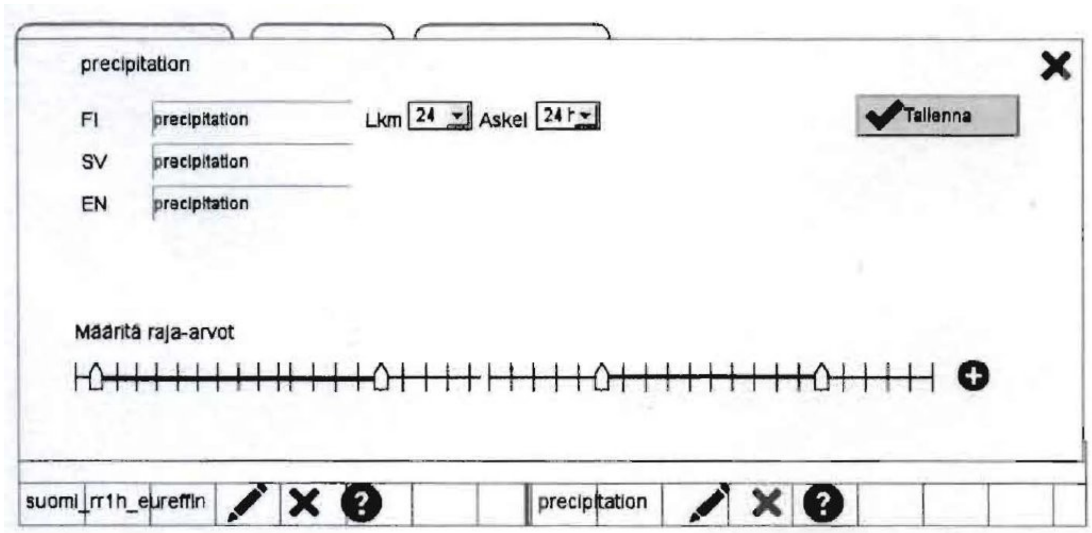
Luo zoomattava ja raahattava sääanimaatio valittavilla sääparametreilla ja karttatasoilla.

TASOT	
Taustakartta	✓
Lämpötila	✓
Sademäärä	
Puuska	
Tuuli	
Tuuli (numerot)	
Tuuli (nuolet)	
Edustakartta	✓

Mock up 3: An alternative way to display the selected tab and grouping of different features.

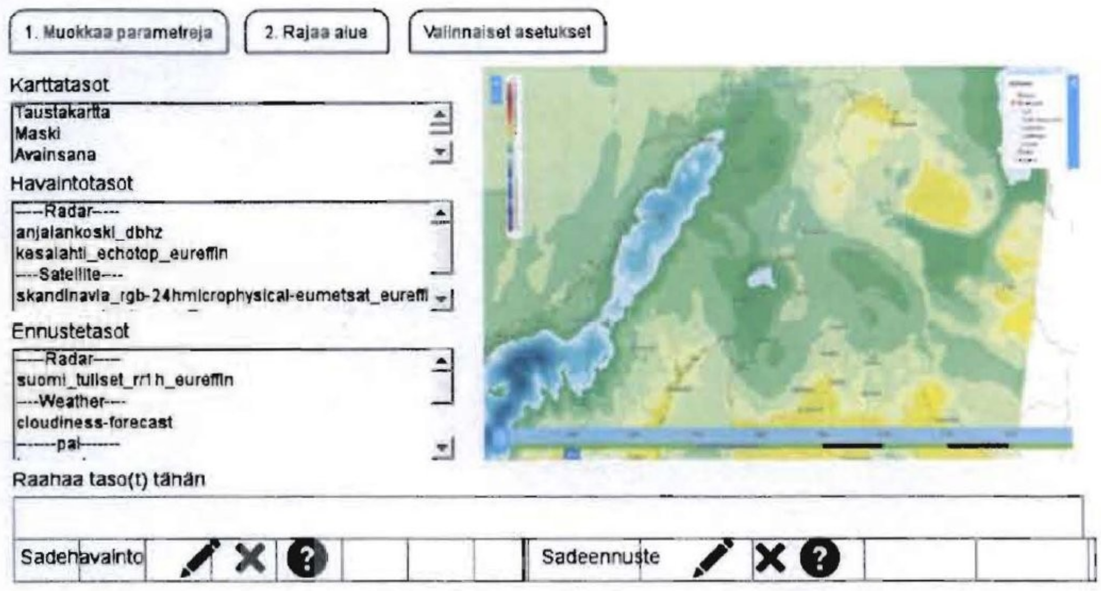

1. Muokkaa parametreja
2. Rajaa alue
Valinnaiset asetukset
Karttatasot
Taustakartta
Maski
Avainsana
Havaintotasot
----Radar-----
anjalankoski_dbhz
kesalahti_echotop_eureffin
----Satellite----
skandinavia_rgb-24hmicrophysical-eumetsat_eureffi
Ennustetasot
----Radar-----
suomi_tuliset_rr1h_eureffin
----Weather----
cloudiness-forecast
-------pal------
Raahaa taso(t) tähän
suomi_rr1h_eureffin


1. Muokkaa parametreja
2. Rajaa alue
Valinnaiset asetukset
Karttatasot
Taustakartta
Maski
Avainsana
Havaintotasot
----Radar----
anjalankoski_dbhz
kesalahti_echotop_eureffin
----Satellite----
skandinavia_rgb-24hmicrophysical-eumetsat_eureffi
Ennustetasot
----Radar----
suomi_tuliset_rr1h_eureffin
----Weather----
cloudiness-forecast
-------pal-------
Raahaa taso(t) tähän
suomi_rr1h_eureffin
precipitation



precipitation

FI	precipitation	Lkm 24 Askel 24 h
SV	precipitation	
EN	precipitation	

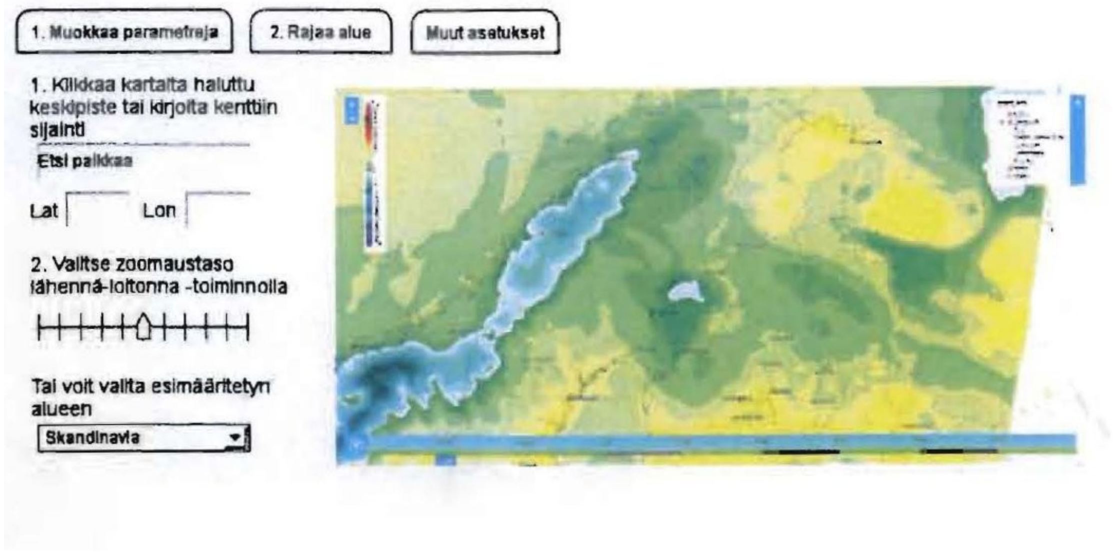


Karttatasot
Taustakartta
Maski
Avainsana

Havaintotasot
----Radar----
anjalankoski_dbhz
kesalahti_echotop_eureffin
----Satellite----
skandinavia_rgb-24hmicrophysical-eumetsat_eureffi

Ennustetasot
----Radar----
suomi_tuliset_rr1h_eureffin
----Weather----
cloudiness-forecast
------pal------

Raahaa taso(t) tähän



1. Klikkaa kartalta haluttu keskipiste tai kirjoita kenttiin sijainti

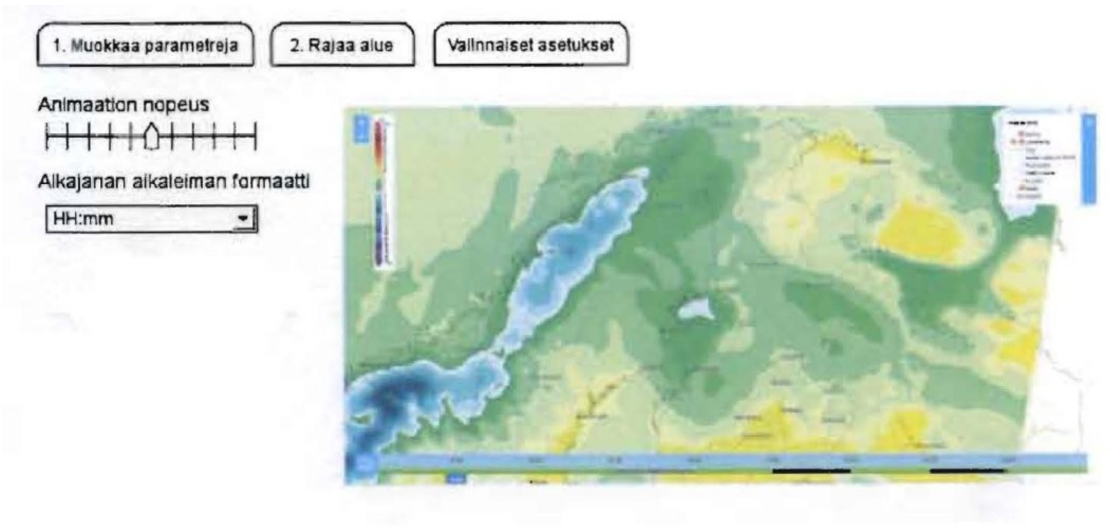
Etsi paikkaa

Lat
Lon

2. Valitse zoomaustaso lähennä-loitonna -toiminnolla

Tai voit valita esimääritetyn alueen

Skandinavia



Animaation nopeus

Alkajanan aikaleiman formaatti

HH:mm