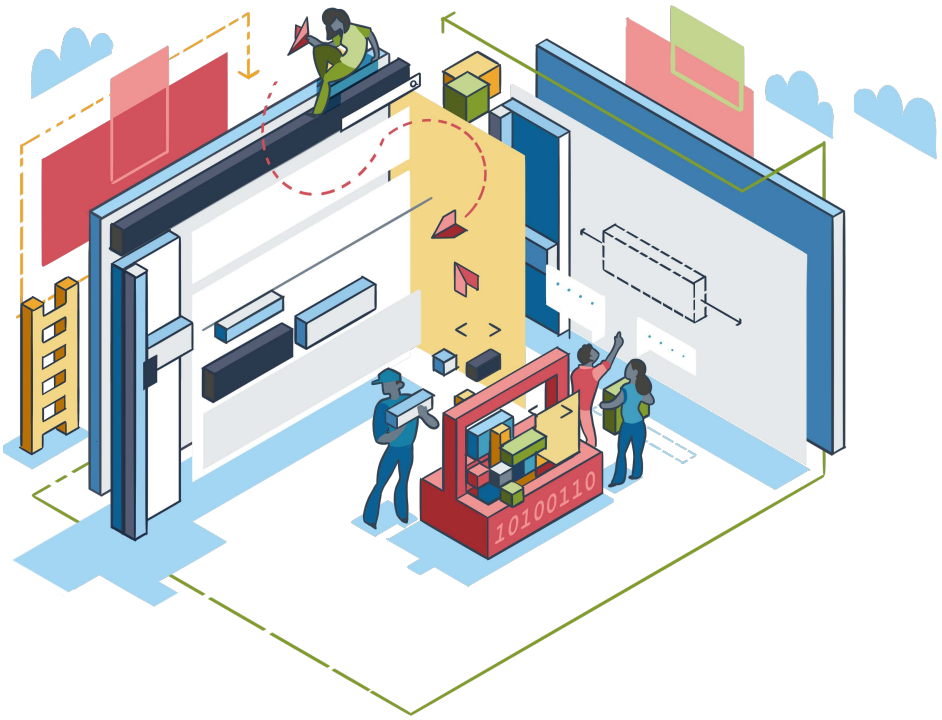
CHALMERS
UNIVERSITY OF TECHNOLOGY

MASTER THESIS



Designing a Design System

Bringing Teams Together Through a User Centred Perspective

MALIN LILJA & FREDRIKA LUNDKVIST

DEPARTMENT OF MATERIALS AND INDUSTRIAL SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019
www.chalmers.se

ABSTRACT

A great way of saving both time and money is making sure work is not repeated. Today, many companies have started to create design systems to enable for more efficient digital product development, while also aid communication between the people involved. At the region of Västra Götaland in Sweden there has been a start to creating a design system, called *"komponentkartan"* or *"the component map"* translated into English. However, the design system has been created in a way that has left it inconsistent, and instead of helping the designers and developers with communicating it has created some dual meanings and double interpretations of things.

The purpose of this master thesis was to understand what a design system is and research best practices to design and improve a new design system for the region of Västra Götaland's IT department. The design system would primarily be used for the IT department's development of remuneration systems. The work included benchmarking and case studies to understand design systems from an outside as well as an inside perspective, complementing the literature review.

The work had also a heavy emphasis on the users of the design system and their needs. Through iterations with user research the design system's requirements were defined and were used as a basis for the concepts. The concepts were iterated in three steps with users and went from low fidelity to high fidelity prototypes.

The result was not just a design system website, but also a process of working so that the users could make the design system stay relevant for their products. The final design system included a on boarding process for new users as well as showing the benefits of the system in a clear way to those not actively using the system, but has more control over it, such as the organisation. It has guidelines to help a developer of the design system to make coherent content, creating recognition and better understanding for an experienced user of where to find information and how. The final design system aids a user to edit and contribute, using the help of GitHub as a base. In these ways, the final design creates a better user experience, and the process of a design system can be better established and worked with at the region of Västra Götaland's IT department.

Keywords: *design system, style guide, pattern library, components, user-centred design, UX, user interface, UI*

ACKNOWLEDGEMENTS

Throughout the work of this master thesis there have been people involved who have contributed to our result. We would like to begin the report by expressing the gratitude that we have towards them.

We would like to thank all of those at the region of Västra Götaland that took the time to participate in our user studies. The input we got from you has helped us move forward and it would not have been possible if we did not get the chance to talk to you. Everyone has been positive to our visits and your feedback has been invaluable to us when we created our concepts and iterated them.

We would also like to thank the people who have provided us with guidance and help during the work of the project. Firstly, we would like to thank our supervisors from VGR trough InUse, Louise Broo and Rebecca Hallqvist, who have helped us with feedback and guidance of how to proceed. It has helped us stay on the right course with the project and we value the support you have given us. Without you we would not even have been able to meet such a variety of users, introducing us and helping us getting along in our research.

Further, we would like to thank our supervisor from Chalmers, Pontus Wallgren for all the help and support with finding out what a design system is. It helped immensely to early on figure out that the two words of design and system are big and difficult to interpret. You have always been positive to our work and encouraged us to reach further, but also set milestones with goals to reach at specific times. Without that, we could have researched the whole thesis on what a design system actually is.

We would also like to thank inUse for the opportunity to write the master thesis with you. We have felt very welcome and have gotten incredible tips and support from you.

Last, but not least, we would like to thank each other, for having produced a successful project as the outcome of good collaboration and support of one another. But also, thank you Chalmers for these five years, it has been great fun.

Malin Lilja

Fredrika Lundkvist

TERMINOLOGY

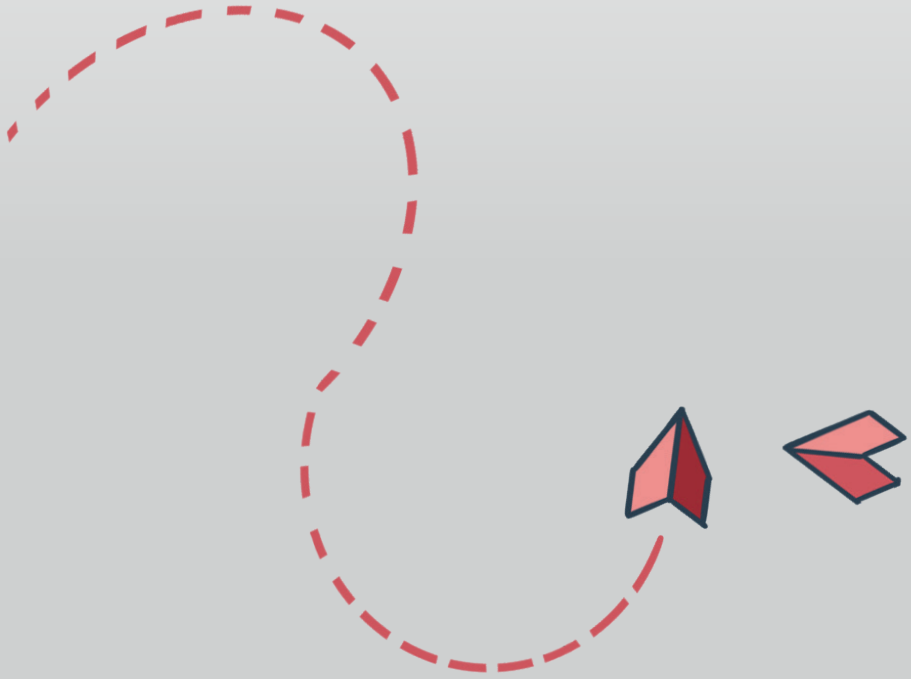
Agile	Is the name for a flexible working process in software development. It is a way to iteratively work with a product.
Angular	Angular is a platform for how to program applications for the web, based on using JavaScript. The components for VGR component library are created by using Angular.
API	This abbreviation is short for application programming interface, which is a specification of clearly defined methods to communicate with a set of various functions.
Component	A component is a reusable part of a user interface, for example a button or a checkbox. If the component is working properly, the code behind it can be reused.
Design debt	This term is about possible concepts and solutions that was overseen in order to accomplish the long-term goal, trying to make the solution as simple as possible (Mazur, 2018). An example can be skipping user testing or missing inconsistencies in the product due to time pressure. When these types of things occur, it can in the end build up onto something that will get worse and worse.
Design system	See page 10 for full description, but briefly it is the foundation for a common language at an organisation which gather values, design guidelines and its building blocks to enable for creating products.
Element	An element can be considered as the smallest parts that makes up a component. Can for example be a box or text.
GitHub	GitHub is a website that hosts service for version control of code repositories.
Pattern	A pattern can be made up of several components. A pattern has a specific use purpose and makes a specific goal available for the end user which can be reused across products.
Pivotal tracker	This software is an agile development project management tool that aids the collaboration in teams since it shares a prioritised backlog of what should be done and have been done in the project.
Scrum	Scrum is a teamwork framework of how to develop, distribute and maintain complex products.
Sketch	Sketch is an application to create vector graphics and is only available on Apple products. It is a tool that helps creating user interfaces and aids with user experience.

Technical debt	This is a term coined by Ward Cunningham that talks about that technology gets released too quick to the market, so that it is not fully finished yet, causing extra costs. The technology should be tested to understand faults that may possibly occur, not focusing on getting the project out to the market as quick as possible.
VGR	Stands for Västra Götalandsregionen and is the county council of Västra Götaland county in Sweden. It is mainly responsible for the health and welfare of Västra Götaland.
WCAG	This is an abbreviation of *"Web Content Accessibility Guidelines*" which are guidelines set up to aid everyone to be able to browse and use internet services and sites independently of how they access the web.
Wireframe	This is a type of prototyping used frequently within UI design, to create a generalised idea of the system.
Zeplin	Zeplin is a software in which Sketch-files can be opened to help in the collaboration between the demand setting designers and the user interface creating developers. For example, a developer can open a Sketch-file in Zeplin, and then they can find information of pixel widths of components and lettering, which will later help when transforming the sketch to final code.

TABLE OF CONTENT

Introduction

This part of the report is an introduction to the master thesis project. The introduction includes a background to the project and the company where the project was performed, alongside with the aim and research questions. The introduction also contains how the work of the project was structured.



1 BACKGROUND

When developing new applications, it is always preferable that the process goes quick and frictionless. To build prototypes in simpler and quicker ways opens for more iterations during the product's development, which can lead to better outcomes. However, the task is far from simple if every product needs to be built from scratch every time.

If an organisation or a company oversee several different products, it is likely that parts of the interface from those products can be reused, if they are organised in correct ways. When it comes to building interfaces for new applications, a design system can be a good aiding tool and process. A design system offers user interface building blocks and set practices, which can reduce friction when developing new products while also ensuring a higher quality. Development becomes more efficient, since the framework of how to build the user interface already exists. This way, time can more efficiently be spent on important things that are more critical, instead of rebuilding foundations that already exists and are documented in the design system.

Design systems have recently become more popular around organisations and companies when discussing digital products. However, they are not anything new and have existed for a longer time but under different types of names (Shevchuk, 2018). A design system can bring different parts of the organisation together and it can help with communication between those involved that have an interest in the system.

1.1 THE COMPANY & THE PROBLEM

Västra Götalandsregionen, abbreviated VGR, is one of Sweden's largest employers with over 55 000 employees, excluding consultants (Västra Götalandsregionen, 2018). It is an organisation controlled by political decisions in the region of Västra Götaland in Sweden. There are different areas within the organisation of VGR that all have their own political board. Examples of these areas are healthcare, public transport and culture (Västra Götalandsregionen, 2018). The organisation gets supported by, amongst others, an IT-department that works with either implementing, customising, or creating specialised systems that are to handle and work with certain things in the different areas that VGR acts in.

The IT department constantly have applications that they work on, which includes both old as new systems. This since development and maintenance are core parts in their work. Within health care, a system has been created to supply caregivers in the so called *"vårdval",* which is care choice. Citizens chose which clinic that is their *"home clinic"*, meaning which health clinic they are to turn to when necessary. The system VGR IT is responsible for is a remuneration system which remunerates the different clinics according to how the citizens make their choice. The system is built on the core foundation that it should be correct and straightforward, ensuring that those working within the remuneration system should not be able to do anything wrong. The system talks directly with economic systems, and therefore the factor of correctness is greatly important.

When the remuneration system was built, there appeared other areas within the health care which required similar systems. This brought up the suggestion of why the systems could not work the same way, since they had the same purpose. This way, the new systems could save time on development since the interface parts simply could be reused and not remade from scratch. The other systems were as well supposed to have similar interfaces, and by *"reusing"* parts from the first system, the developers and designers could easier assemble a new system.

These interface parts, such as buttons and checkboxes for example, were called components. When these common and frequently used components were identified the developers and designers started to build what they started calling *"komponentkartan"*, or *"the component map"* translated into English. In *"komponentkartan"* these components were gathered, to ease access to those that needed to use them. In the collection of these different components there were documented examples of the code, showing how the component acts and looks but also describing how it can and should be used. This collection has helped when building the new remuneration systems and has speeded up the process of how front-end code normally is developed. It has expanded outside the remuneration application area as well, helping other parts of the IT-department to build their applications.

However, the component collection as it is built today has emerged between projects. It lacks in both consistency as structure, and its users, which are mainly designers and developers, do not have a unified view of how to use, work, and develop it. The documentation of the components is inconsistent and there is an unclear process of how to maintain and develop what can be found in the design system. The different components are structured in different ways and their level of detail vary between who has created the respective component and with which purpose.

As it has already helped teams to quickly iterate and build front-end code there is a need for "*komponentkartan"* to become a standardised way of working. A possible way to do this is to evolve the *"komponentkartan"* into a design system. A design system should include the components and the documentation, but also give support in how to work with design and development to create a good experience, both for the users of the design system and the users of the products built with the design system. The design system could not just help with reusing ways of creating applications, but also keep the organisation coherent and clearly connected through its products and systems. Throughout the project, this should be investigated by user studies and research, to try and understand which solution of a design system that best would fit VGR IT.

2 PROJECT PROCESS

Throughout the project the work followed the double diamond process (Design Council, 2015). The work started with identifying the problem, followed by a phase of discovering more about the problem space at hand and other areas connected to it. Afterwards followed a phase of defining more concretely what that problem was, narrowing it down from the discovering of the problem. From the problem definition there was a develop phase to open and discover the solution space, which created concepts and ideas that were investigated and evaluated alongside with users to understand more about their potential. These ideas were then taken into

the deliver phase where they were narrowed down to become a solution to the original problem. The double diamond process is about expanding and narrowing down in two different batches, to go from a problem to problem definition into a final solution. A visual of this process can be seen in figure 1 below.

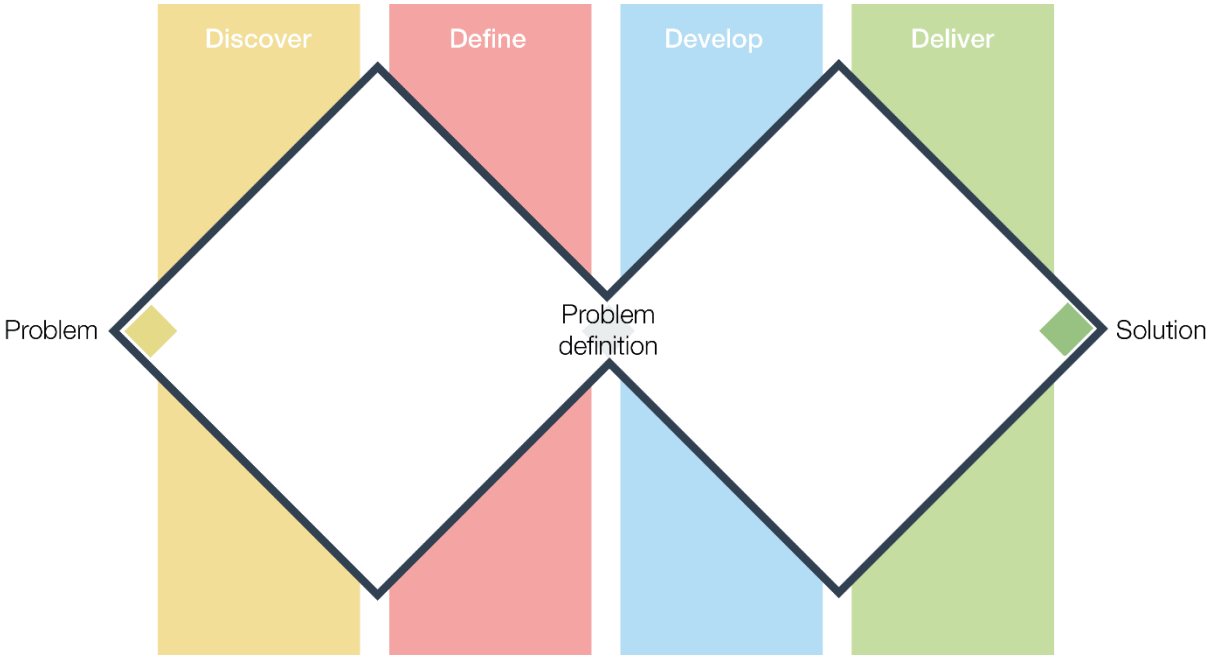


Figure 1. Graphic over the double diamond process. Authors own image.

3 AIM

With this thesis the aim was to understand and further develop the current tool, *"komponentkartan"*, of the region of Västra Götaland's IT department and design it into a consistent and well-functioning design system. The thesis aimed to investigate and make suggestions to the design system to make it as beneficial and as useful as possible for its users, concerning both the layout of the system as well as the process behind it.

3.1 RESEARCH QUESTIONS

The project dealt with answering the following research questions:

- What is it that makes a design system advantageous and when is it applicable?
- What needs do the users of the design system have and which user group should be prioritised, why and how?
- How should a design system be designed, concerning its structure, content, documentation and process, to best fit the users' needs?

The answers to these questions will be presented in the form of requirements and guidelines, but also a proof of concept, showing these in a context. The prototype and guidelines that was

set up was evaluated in iterations with the user, constantly trying to improve the concepts of the iterations.

4 DEMARCATIONS

During this thesis, the aim was not to create an entirely new design system from scratch for Västra Götalandsregionen IT. The work during the thesis focused on developing and alter the existing, established *"komponentkartan"* and other tools that already were being used daily. The research did not primarily investigate what components should be in the documentation but did rather investigate how the existing components should be documented, used and developed in a better way to become so called user patterns, or maybe even better components. Also, the thesis investigated and designed how the patterns of the system should be presented and used.

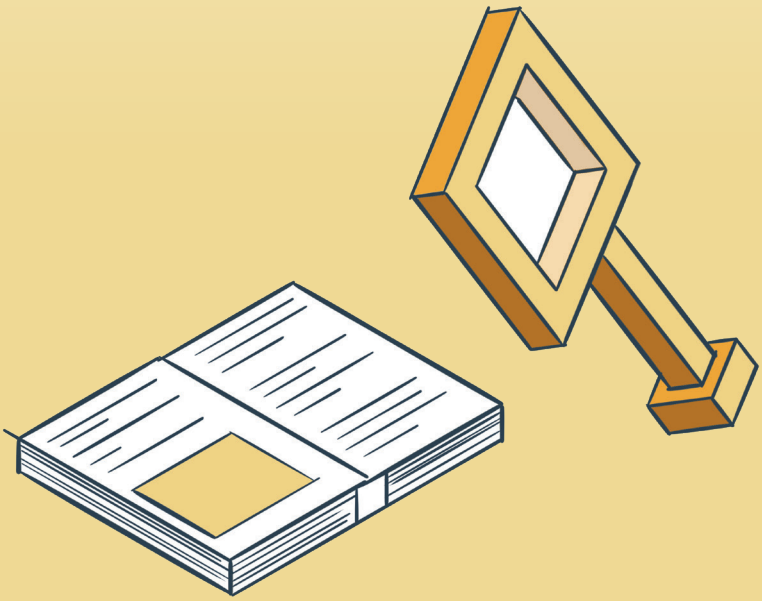
Another limitation comes from the decision VGR have made to not have reactive components changing with different screen sizes or devices. This limits their products and the design system to just cover applications made for desktop. Therefore, the research in this project has not prioritised covering reactivity.

The prototypes created in the project were not coded to work as a functioning website but were rather wireframes showing the proof of concept. It visualised and explained how the system would look, but also demonstrated how the system could be used and why it is useful. The prototypes' text was created according to set guidelines and insights during user research that demonstrated the content, rather than being definite words and formulations.

The focus of this project mainly concerned how to make the design system useful for users to create digital content, and not focused on developing the branding aspect of design systems.

Discover

The initial phase for the project was discover, which opened for further themes and possibilities for the project to proceed with. It researched what a design system is with the help of literature and research of existing design systems. During this phase, there was also an initial investigation of the users to "komponentkartan", to discover how the current solution was apprehended and used. The most important findings were gathered in the end and brought into the next phase.



5 METHODOLOGY DISCOVER PHASE

In this phase the methodologies picked served to gain both theory and knowledge about design systems to open for possible solution spaces. The methodologies used were to provide a basis for the continued work and their result covered subjects such as theory, best practices, and initial user research. The purpose of these methods was to discover design systems and lay the groundwork for the next phase. This chapter describes the different methods used, and their results and conclusions can be found in the following chapters.

5.1 LITERATURE REVIEW

A literature review is a method to find, collect and evaluate the literature in a chosen area. It provides a current state of the topic and summarises what is known about the subject. The method also serves as a starting point for the research and for new ideas (Royal Literary Fund, n.d.). The literature study for the project aimed to discover themes and theory concerning design systems to enable for new perspectives and insights. As both *"design"* and "*system"* are big words that are difficult to easily define, the method created a starting point to talk about design systems. It identified how the topic could be discussed, but also defined. Its purpose was also to find out best practices in the field and how the theory connects to the success of design systems, but also what causes it to be unsuccessful.

5.2 BENCHMARKING

To discover and identify what opportunities there are with a design system, a benchmarking was made. According to Maylor (2010) a benchmark is a reference point and a standard by which different factors are judged. It serves as a temporary fixed point, which is decided by a set of relevant measures or metrics. A benchmarking's objective is to find out how the best performers are doing and how these results are achieved.

The benchmarking made in this project was of a brief and generic nature as it served to gain knowledge about how others are currently implementing their design systems and how they are doing it. The benchmarking set out to investigate design systems of different fields of work, to see what the success factor can be within different areas. The research served to find how others are trying to achieve the best result with what could be accessed from an outside perspective. The benchmarking was made using resources available online to research the different investigated design systems.

The benchmarking was performed by creating generalised wireframes of the different pages in the systems, that represented the content and structure on the pages. The wireframes were firstly made traditionally with paper and pen and showed an overview of how the website was structured and how it would work during interaction. The wireframes made it possible to compare the overview pictures, to ease the process of finding common, differentiating and unique factors within the systems. After each system's wireframes were created, the different systems were first described according to certain categories such as structure, examples and

maintenance to mention a few, but these were then also compared. To ease with the comparison illustrations of the different content were made.

5.3 CASE STUDIES

Case studies were performed to gather qualitative information about other design systems from an inside perspective. The case studies used semi structured interviews to gather the information (Wikberg Nilsson, Ericson, & Törlind, 2015). The method mentioned in section 5.2, the benchmarking, had the perspective of looking at design systems from an outside perspective, whilst the case study's aim was to look at them from the inside by interviewing a person experienced of using the design system. The main purpose was to find out more about the teams and processes behind the design system, to understand what factors had influenced the design systems the most when it came to overcoming problems, but also to create and find benefits. It also served to find out the interviewee's relation to it and the design system's relation to the company being investigated.

5.4 INITIAL USER STUDY

The initial user study with users at VGR IT was performed by following a semi-structured interview template with prepared questions which were considered important to be answered throughout the interviews. To keep the interview semi-structured means that there was room for explanations and possible elaborations on certain questions (Wikberg Nilsson et al., 2015). Alongside with the planned questions there was a participating observation together with the interviewees. A participating observation includes that the observers sees, listens, asks and observes what the users are experiencing with a product, to further understand the way they interact with the specific task (Wikberg Nilsson et al., 2015). The interviews and observations were done by each interviewee's workspace, to make it possible to see their computer so that they could explain more thoroughly about the component collection *"komponentkartan"* and its demonstrative website *"komponentkartan demo"*. This created a medium for both the interviewers and the interviewees to discuss around and ease the communication.

For the initial user study there were four participants; two designers and two developers, that all had a strong connection to *"komponentkartan"*. The participants work in two separate projects, both regarding the remuneration systems mentioned in section 1.1. The interviews were audio recorded to aid the researchers with keeping focus during the interviews and observations and to also help with the analysis later.

The interviews were afterwards analysed via transcription of the audio recordings and then also grouped into categories to find the most common problem areas and possible demands, similar to the KJ technique (Martin & Hanington, 2012). The results from the analysis gave a bigger understanding of what should be more investigated in the define phase of the project, since it explained where the problems were.

6 LITERATURE REVIEW

There exists no real or defined standard definition of what a *"design system"* is, and the term is used in different ways. This chapter will cover the outcome from the literature review, describing theory about design systems. The chapter will first discuss what design systems has been defined as by some authors and then discuss, according to the literature, what factors it is that make a design system useful and beneficial. It will cover how design systems are structured, documented, organised, adapted and maintained.

6.1 DESIGN SYSTEMS THEORY

Design systems are not directly defined in a core concept that covers all types of design systems. A design system serves other applications by offering a common set of components and practices across teams. But widely, the term design system and what it is have differences depending on who is explaining it.

Kholmatova (2017) defines a design system as:

> *A design system is a set of interconnected patterns and shared practices coherently organized to serve the purpose of a digital product.*

The patterns referred to in the quote are elements that are combined to create interfaces. These patterns can be divided into functional and perceptual patterns. The functional patterns are the building blocks that bring action and purpose to the interface, for example a button taking the user to the next page. Perceptual patterns include styles and how the aesthetic of the system should be, for example which colour to use to bring attention to certain details. Beyond the mentioned patterns are also the so-called practices of a design system, that are how the programmer or designer chooses to use the patterns and how they get combined. For example, a practice could be guidelines of how things should be used.

Suarez, Anne, Sylor-Miller, Mounter and Stanfield (n.d.) defines a design system as following:

> *A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build any number of applications.*

In this definition, the design system is described as having a collection of reusable parts which are guided in the way they should be used, by the help of documentation for example. These parts are created by combining multiple elements according to specific standards. Again, a definition states a design system as something cohesive that gets put together by defined parts. In Suarez et al. (n.d.) it gets explained that the design system's *"what"* and *"how"* is critical to decide in order to create a good user experience. This since it is difficult from a start to get everybody to agree on what a *"standard"* should be.

Guth (2018) discusses as well that it does not exist any standard definition of a design system, just different opinions. Guth (2018) describes design systems according to a *"system"* view and that it is an interconnected group of elements that are organised in a way that focuses on achieving a certain goal. The elements of a design system are said to be anything from how processes at a company is carried out, to design principles and assets, alongside with the brand.

The core gets argued to be a documentation of a unified set of design rules and patterns that can create a scalable design. A design system is said to not be for the final product that it creates but are rather for those who will create and build that product.

There is also a methodology described by Brad Frost (2016) called *"Atomic Design"* which also breaks down design systems into smaller stages of interface creations. It gets called stages instead of parts which the others have used, since the smaller parts build of each other and are based on each other. Instead of patterns and components, Frost calls it atoms, molecules, organisms, templates and pages and ties the methodology to biology, trying to aid a user or creator of a design system to understand the logic behind the building blocks of a design system. With Atomic Design and design systems, it does not get considered as a linear process, but rather that it is a mental model of how to design and create applications and systems. With a design system, it can create a cohesive end product that has had the help to be developed by a system that have collected the parts that build up that product.

These are four different types of definitions and methodologies but a thing that appears important in them all is that the design system should ease some kind of process. It can be the process of creating the new digital product, or even a process of using the design system itself. The design system is a tool for the creation process, either within or outside the system.

6.1.1 ORGANISING THE DESIGN SYSTEM

In literature, design systems are said to be made up of several parts, making the whole. A design system is said to contain a style guide, pattern library and a component library, but they are sometimes easily mistaken for each other and the words and their definition can vary. This can happen since there is no set definition and different people have their own interpretation of what the words mean. Even though they are related to each other, they can be differentiated. All of the parts that a design system consists of handles the way code is created, and what it in the end will look like, for the application being made or developed. Below the terms are described and how they are related. The way the terms are described to below is how they will be referred to in the rest of this report. The descriptions are based recaps of the discussions about design system described in this section.

Design System – A broad concept covering how digital applications are built according to a set of standards. A design system is a process that has collected a set of standards, principles and guidelines. It includes different kinds of assets and tools to achieve these standards, like pattern libraries of interaction patterns and code, alongside documentation on how to use these. A design system should be a living, everchanging product that oversees its content and keeps it updated, making it evolve alongside the applications tied to it. It should have defined ways of working with it, to make it sustainable over a longer period of time.

Pattern Library – A pattern library is a part of the design system. It is a collection of *patterns*, which could be interaction properties or aesthetics as well as code, depending on what the purpose of the library is. A pattern is made up of different smaller parts, that are the building blocks of creating that pattern. With a pattern you can show a type of interaction and how something like for example a button can be used in context with other components to have a meaning, but also how it can act and be in an application. A pattern consists of different smaller components, but the components in turn is made up of smaller elements. For example, a

component can be a button, but the elements to that button is text on the button, and the shape of the button. This shows that a pattern cannot be created without either components or the smaller elements. To see an illustration of this hierarchy, see figure 2 below. Generally, the pattern library offers the tools and assets for creating applications, documenting the importance of even the smallest parts in a user interface.

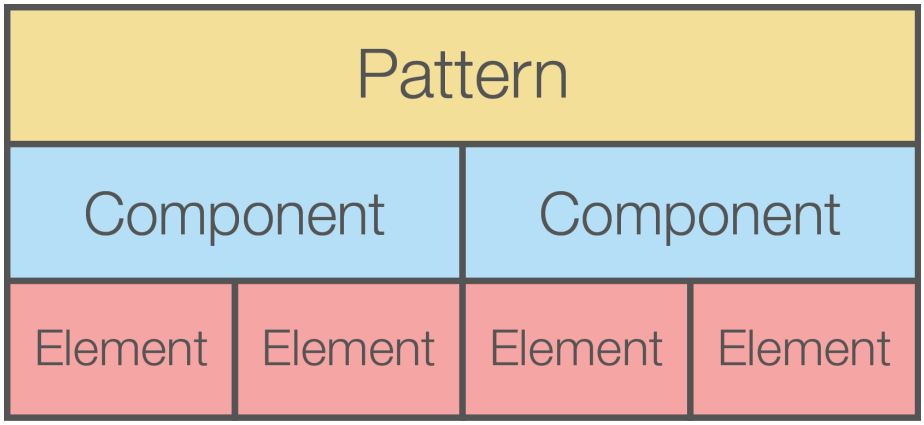


Figure 2. An illustration showing how patterns, components and elements are parts of each other. Authors own image.

Style Guide – The style guide is also a part of the design system, setting guidelines to follow when creating new content in the design system or within the organisation. It contains static documentations of how things are supposed to be used and how products are supposed to look and feel. This to aid the users of the design system to be able to make as much out of it as possible. It takes into concern the brand of the organisation that have created the design system.

6.1.2 GUIDING PRINCIPLES

A good design system, regardless of the technologies and tools behind it, follows the following guiding principles, as stated by Suarez et al. (n.d.):

- *It is consistent.* The way the parts of the system, the components and patterns, are created and managed follows a predictable pattern.
- *It is self-contained.* The design system is treated as a separate, standalone dependency that needs maintenance; it is a product serving products.
- *It is reusable.* The components and patterns are created and built in such a way that enables them to be reused in different contexts.
- *It is accessible*. Any application built with the design system is usable by as many people as possible, regardless of how they are using the web.
- *It is robust*. For every application the design system is applied to, the application should perform accordingly and with minimal bugs.

6.2 BENEFITS OF A DESIGN SYSTEM

A design system can be advantageous in many ways and can help organisations and businesses. In this section of the chapter the benefits extracted from the literature review will be explained.

6.2.1 EFFICIENCY

A design system can help the efficiency of a team in the work of creating a new application. According to Kholmatova (2017) there are some arguments of how a design system can help, regarding both time savings and product delivery. With a design system, time can be saved on building up the new application for the company or organisation, since the simple framework is already created and can be applied, so that more focus can go into the details and create as good of an experience of the system as possible. It can also save time when it comes to changing the whole site, since components and patterns will be connected all over and the system will be easier to maintain. The product launch of the product can also occur quicker, and be more verified, since if the system can become completed early, the more tests can be made to verify it.

Frost (2016) mentions similar savings, with saving time when having a design system. To have a design system creates more manageable chunks of code and design, so that when a change needs to be done, a lot more than just one thing can change at once, making updates more efficient. It is also expressed in the text by Frost that systems do not need to be built from the foundation up anymore but has somewhere to start thanks to a design system. A design system is said to speed up the workflow of teams, which is a big part and factor when saving time, and money. A design system can also speed up the production time so that high quality work can be launched faster.

Guth (2018) agrees with both Kholmatova and Frost, saying that a design system that is set up and defined, and is frequently being used, can create an environment where things gets done. By things being done, it does also make the process of creating new applications shorter, speeding up decisions and the development.

6.2.2 IMPROVING COLLABORATION & REDUCING SILOS

A design system can help with bridging the gap between designers and developers. Without a functioning way of handling design and code, information risks to be siloed and not distributed across the whole team or even not available when a person finishes their job at the company. Inconsistent processes can result in an unsustainable code base which is neither helping the developers nor the designers, or their ability to collaborate (Chan & Hayes, 2016). Design systems can improve transparency and eliminate knowledge silos. It can also help to reduce the design and technical debt as it distributes knowledge and make it available across every team (Vella, 2018).

In the onboarding process the design system can be an invaluable tool. It enables new members to the team to catch up with the design language without having to put strain on other team members to teach them everything. It will also improve the new members productivity, for

both designers and developers, as they can independently design and develop with the help of the design system (Vella, 2018).

6.2.3 END USER PERSPECTIVE

For the end user the design system can provide a consistent experience across applications. As the users recognise how things are meant to be used there is a reduction in the risk of what Spool (2011) calls the *"Magic Escalator of Acquired Knowledge"*. The model explains the user's knowledge in relation to what there is to know about the application and how much they need to know to make use of it. A large change to a design causes a disturbance in how the users can apply their current knowledge onto the system, and the users will spend time and energy relearning the experience in order to climb back up the escalator of acquired knowledge to the place they were before the change. An illustration of this thinking can be seen in figure 3 below.

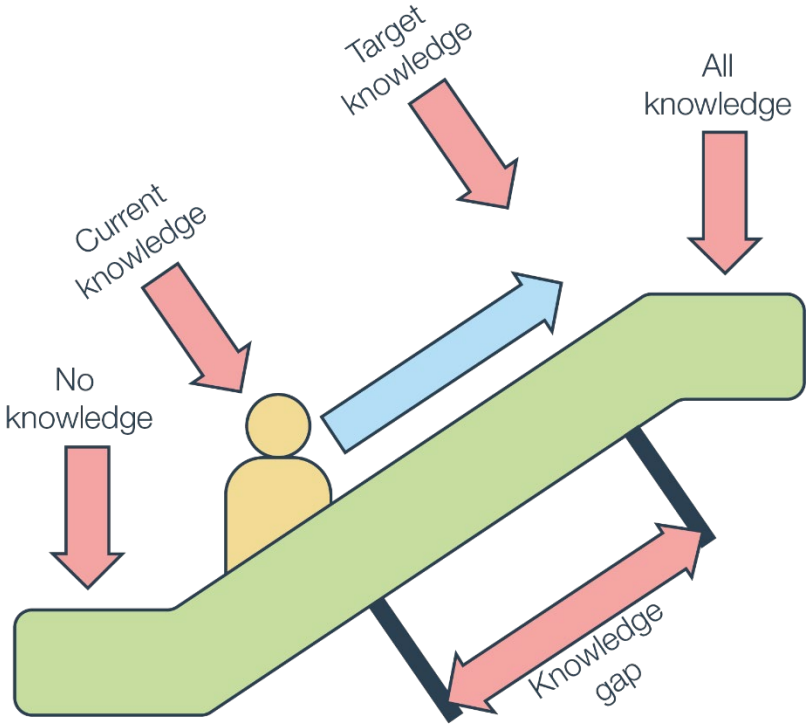


Figure 3. The magic escalator of acquired knowledge as described by Spool (2011). Authors own image.

To make big changes in the way applications look and acts can become difficult. For an end user the consistency and recognition during the experience of the application is valued novelty. To have a design system maintains a coherent view of an application, and to apply the style guide on the patterns will ensure a more coherent view. Design is not just about the looks, but also about the experience behind. (Christenssen, 2019).

6.3 DOCUMENTATION

> *If it's not documented, it doesn't exist.*
> (Suarez et al., n.d.)

No matter how the design system is built, a main cornerstone of it is the reusability and self-contained aspects. The documentation for the design system plays an important part as it will enable new people to understand the content of it better and not be dependent on other people having to explain it for them, while also making sure that the resources of the design system are used coherently.

6.3.1 CONTENT

Documentation describes and shows information to usually explain a certain thing. Documentation is important in a design system, since it explains the different parts of it, also explaining how those are to be used. By having a well-documented design system, it should be possible to use the tools provided without having much prior knowledge of the system itself. The documentation can be written and structured in different ways, depending on what should be described, to what purpose, and towards which user of the design system. However, good documentation takes effort and does not come for free. A high-quality component documentation is a sign of an effective library, which is the work of planning, effort and process. The components should be written in such a way that it drives effective design decisions and speeds up development times (Curtis, 2018).

To achieve consistency the most important task is to define the rules of the design system. This means documenting the rules and making sure that everyone follows them. A well-documented design system, with code standards and best practices in place, will benefit developers as well as designers across the whole organisation, as they will be able to use and contribute to the design system (Suarez et al., n.d.). Frost (2016) is also reasoning similarly as he writes that the documentation of a user interface should contain insights from each discipline involved in creating it. These insights should be incorporated in the living design system. He states that effective pattern libraries have a designated space for defining and describing components for user interfaces, which could range from accessibility to performance, even to aesthetics and more.

6.3.2 TARGET

A design system needs to consider broader audiences to not come across as too vague or too technical. If it does it can intimidate other disciplines and make them believe that the resource is not meant for them (Frost, 2016). A design system works to bring different disciplines together, therefore the documentation should do likewise.

As in any product development it is important with research to know who will be using the design system and how. The design system will more likely be used if it fits into the workflow of the teams (Suarez et al., n.d.). To know the workflow, you need to know the users in the workflow. To ensure high quality on the documentation in the design system, it has to be

recognised that the different users of the system has different needs. Since they have different needs, there has to be prioritisations made between which user's needs that are the most important to focus on. The two main audiences of a design system are developers and designers, but the documentation will serve these different users to a varying degree, as well as the other users involved that are neither designers nor developers. There is a trade-off to be made, depending on objectives, frequency of use, content depth, quality, cost of production and relevance to the work carried out (Curtis, 2018).

> *Getting 50 engineers to code well-designed components is more likely to result in cohesive, efficiently built experiences compared to 50 designers reading tomes of guidance about decisions already built into that code.*
> (Curtis, 2018)

6.3.3 CONTEXT

To understand a design system, understanding the context is key. A problem with a lot of design systems is that they do not provide information of when, how and where their components get used. Without providing context it makes it more difficult for designers and developers to know how global a pattern is, which could lead to loosing track of what it has been used for and which pages would need to be revisited (Frost, 2016).

There are a lot of pattern libraries that shows a demo of each of the user interface patterns, but it should be clear what the demo is showing. When showcasing a pattern of a user interface in a library, it needs to demonstrate context for the design system users to understand how and where to properly use them. Design patterns are dependent on context and does not exist separately from each other (Frost, 2016).

However, even if a lot of things can be documented alongside each pattern it is not feasible to try to cover everything from the start, especially for small teams. Kholmatova (2017) argues that to see tangible benefits sooner, a team should start with an overview documentation that is lightweight and covers the main patterns. Once there exists a simple foundation for the documentation, it is possible to improve the pattern library as time goes by.

6.4 PROCESS

As a design system is supposed to be living and changing alongside the applications it is used for, it is necessary for the design system to live as a process. It should be used in a context with users, and if the design system cannot be adapted to its users' processes, it will most likely not be used at all.

6.4.1 ADAPTION

As a design system is supposed to be constantly evolving, the roll-out process is not as important as having an intentional and thoughtful strategy to ensure that adaption is treated like

the crucial part of the building-process as it is. All parts of the design system will evolve, but also the way the design system is adapted and shared will evolve with time (Suarez et al., n.d.).

It is also important that the design system is put into the centre of the development, or else it risks ceasing to reflect the current state of the products it serves and becomes outdated. When the pattern library, that is managing the design system, cease to be accurate the website and product maintenance once again becomes inconsequent. At that point it has ruined the thoughtfulness that went into creating the design system in the first place. To be successful in the long term there needs to be a shift of the outlook on what is created, rather than thinking the final applications are the only responsibility. It must be recognised that the design system is the foundation to the final product and pattern library (Frost, 2016).

In the process it is fundamental to decide how new patterns should be added to the pattern library. Effective pattern libraries and teams have systematic approaches integrated in their process. However, the different workflows might vary a lot across different organisations. No matter how it is performed, with synchronising for example, it is important for the team to practice the same approach. It means they have a common view of the structure, naming and understanding of the purpose with the system (Kholmatova, 2017).

6.4.2 GOVERNANCE

The result and maintenance of a design system is dependent on how the design system is handling ownership, or so-called governance. In the context of a design system, governance does not necessarily mean that there needs to be a team solely handling the responsibilities with it, but there needs to be a thorough plan to handle it. A design system without a way to handle the responsibilities within will be struggling with not becoming obsolete. Having a clear governance plan is essential for the design system to thrive and adapt as time goes on. A solid governance plan needs to handle questions concerning adding and removing patterns, accesses, updates and responsibilities, to mention a few (Frost, 2016).

In general, the governance models can be divided into two different opposing principles of centralised- and distributed model (Kholmatova, 2017). However, there are team models that fits outside of these which could be the solitary- and federated model as described by Curtis (2015). These models are described below together with an explanation of how to assemble the different teams.

Centralised Model

The centralised model has a clear ownership of the design system and there is a person or a group of people primarily managing the system. This means that they are defining the patterns and rules and plan what is going to happen with the design system as well as managing all the resources. They have veto power over the system and can steer the system into a focused direction and it is their responsibility to ship the features to the users of the design system. Main advantage is the responsibility and ownership the group has over the system, as it will lead to the system getting evolved and maintained properly. The downside to this model is that it can lack context to the actual product teams' constraints and needs.

Distributed Model

The distributed model has no clear ownership of the system and everyone who uses it is also responsible for maintaining and contributing to it. This provides autonomy for individuals and teams to make their own decisions. It can also enable the system to be more agile and resilient, but also serve the user's needs better. However, this model relies on people's own motivation and engagement in the design system and risks becoming deprioritised.

Solitary Model

The solidary model sees one team as the builder of the system, which is often supporting their own products, but keeping it open for others to use it as well. This team model is quite poor, even though it can provide value to teams that lack capacity in design and front-end gage. The teams using the system risks becoming divergent as the team building the system, no matter how altruistic they are, will be having their own product in mind when developing the system.

Federated Model

The federated is built upon having multiple teams represented in a committee that are the decision makers of the system. The committee's responsibility is to ensure the direction for the design system and then build upon that and communicate it by having supporting artefacts like a style guide or pattern library. Not everyone in the product teams gets to participate and the teams will adapt the outcome or ignore it at their own risk. By having a federated model, it becomes more likely that the design system keeps its relevance to the product team and their different products. It will also be more likely that a federated model limits the impact of biases as it is representing many perspectives and eases the adaption of the design system as it will communicate to more product teams directly. However, there are also downsides introduced with this model, which can be practical challenges in the decision making. There must be a balance to where the decisions happen, pausing the larger decisions until everyone can discuss it, while minor decisions still can happen naturally.

Composing the Team

As there are different ways of applying a governance model to a design system there are also possibilities concerning what people should be involved. To make the design system team blend both engineering and design is a strength, as it will create the necessary bridge between design and engineering. There is also a strength in half-time capacity, as it creates visibility for the key products to be reflected in the design system and minimise bias towards a certain product (Curtis, 2017). However, there could be a risk with not having people working full time with the design system as the design system often have a very different timeline of that of a feature. A design system has a holistic approach and unlike the shipping of a feature, which often has a set deadline and shipping dates, it can get deprioritised. It can also stress the people working part time with the design system and product, as the product feature will most likely need to be done in a shorter time perspective (Mounter, 2018).

Another way to compose the team is to balance continuity with rotation. By balancing a few people working full time, from different disciplines, with people that works across products rotating with staffing the design system team, it can create an adaption of the design system.

By rotating more teams, the different teams get to understand the design system on a deeper level and can start feeling a connection to it, wanting to contribute and evolve it. Also, the products can have more influence over the design system without the design system getting deprioritised.

Integrating a new member to the design system team is a great way to test the quality of the design system. It can give a measurement of how well it is documented, the code quality and how it adheres to the promises made within the design system (Curtis, 2017).

6.4.3 LANGUAGE

When a design system is up to date it is an aid in communication between different teams, since it creates a common language (Kholmatova, 2017). Without a shared language it is not possible for multiple people to effectively create together. It is therefore preferable to define the names of things between professions, so that possible deviations can be detected and then fixed so that all parties involved is referring to the same thing. Kholmatova (2017) also argues that design is a form of language, and through the way design is displayed, it is possible to communicate to the user what should be done, to learn the interface quicker, reducing their cognitive load.

Frost (2016) agrees with what Kholmatova mentions and says that by having a design system the work becomes more collaborative between team members. This since the shared vocabulary of a design system will speed up the process achieving work that will not have to be redone later. By having the shared language there will not be blockades of misunderstandings that need to be eliminated before moving on.

As well in Suarez et al. (n.d.) a strong argument about design systems is that they improve the designer-developer communication and helps the translation between graphic looks and programmed code.

7 BENCHMARKING

In this project five different design systems were investigated for the benchmarking. These companies were Atlassian (Atlassian, 2018), Australian Government (Australian Government, n.d.), FutureLearn (FutureLearn, 2019), GOV.UK (GOV.UK, n.d.) and Shopify (Shopify, n.d.). These design systems were chosen since they had different structures and targeted different types of markets and users. This was preferable since it was possible to understand differences between different companies and markets. The chosen design systems were open to the public, displaying their information and design systems openly. The outcome of the benchmarking was to help with understanding different design systems and spark inspiration for the evolvement of the VGR IT design system.

7.1 RESULT

The main and most relevant insights are presented for each of the design systems included in the benchmark. The result of the different design systems was also compared and analysed to be able to pinpoint the most relevant differences.

7.1.1 ATLASSIAN

Atlassian is a software development company focusing on helping teams with the products of their own. The company has 3 000 employees and they have more than 4 000 apps on their marketplace. They also have a large team with dedicated resources towards the design system to develop and maintain it. In the introduction to the design language they are displaying the resources available, but also linking to the Atlaskit which is another resource in the Atlassian design system. An illustration over how the design system show and combine different kinds of information is shown in figure 4 below.

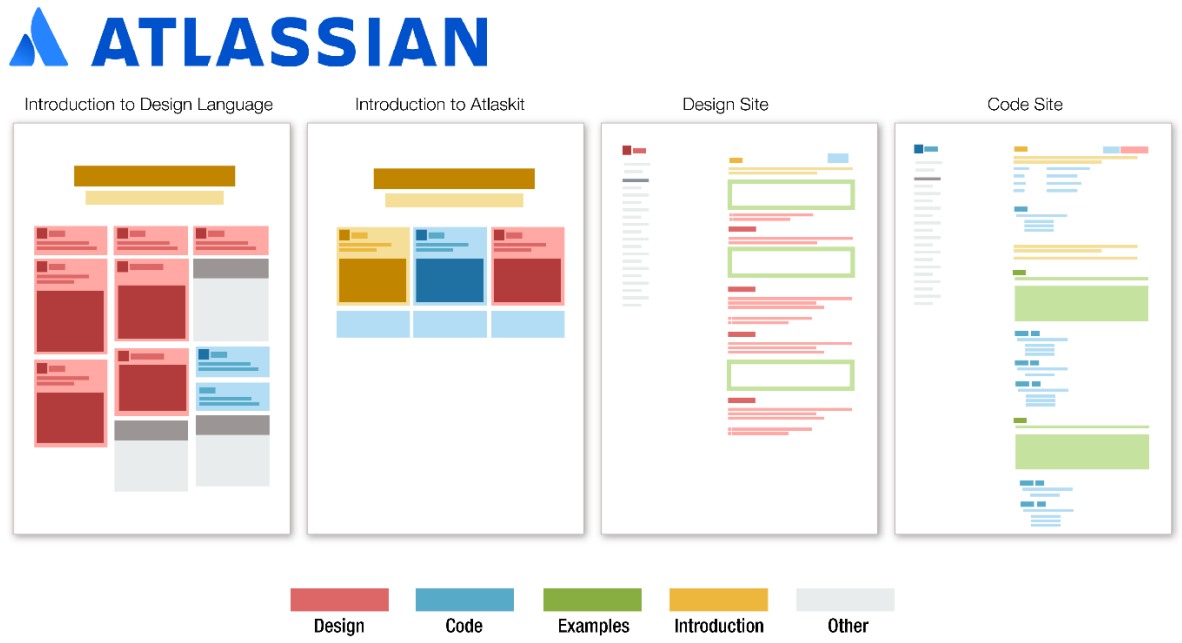


Figure 4. An illustration over how information about design, code, examples, introduction and other information is placed and structured, done with the help of shapes, in the Atlassian design system. Authors own image.

The Atlassian design system is extensive and their design language includes a brand guide, marketing guide and product guide, here called design site, together with its belonging API guide, called packages, in their own code site belonging to the Atlaskit. All of it is cross referenced and linked between sites. Even though they are separate modules they are linked to each other to create a coherent design system and brand expression.

The design site examples are just static pictures while the examples on the code side are interactive. The design system is extensive and contains a lot of details concerning how to use the components, regarding both design as code. They have separated the design and code to optimise for the different audiences the system has.

7.1.2 THE AUSTRALIAN GOVERNMENT

The Australian Government design system was created to improve people's experience of government sites and is a design system that focuses on their components, the progress of these and their community where the site gets further developed. The design system is an initiative of the Digital Transformation Agency and is maintained by a multidisciplinary team. The code repository, the code library, is available to download on GitHub and they have a separate test-site. An illustration over how the design system show and combine different kinds of information is shown in figure 5 below.

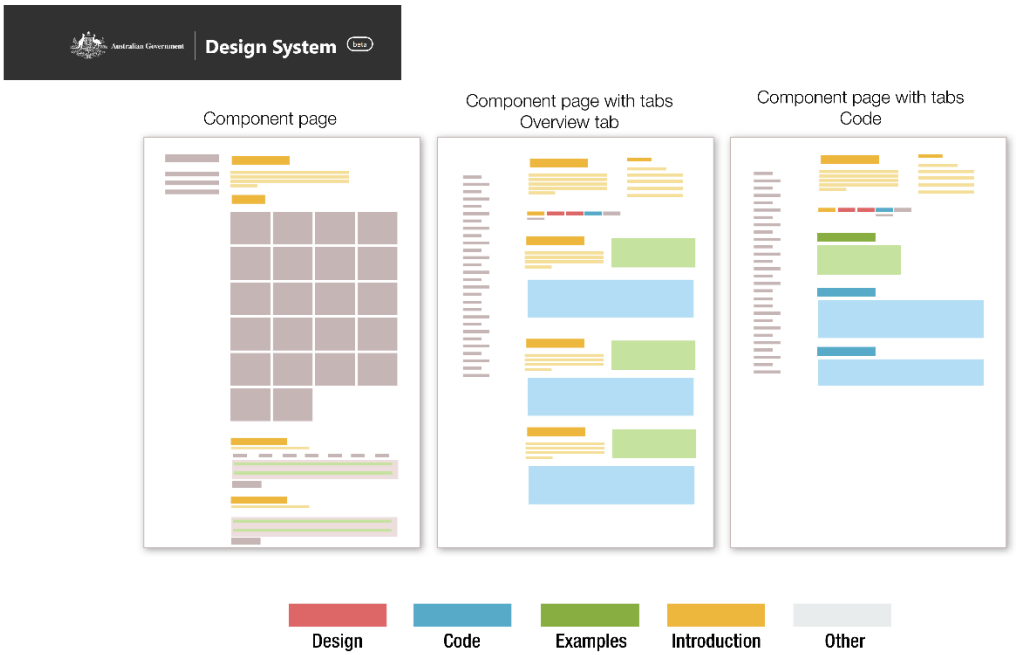


Figure 5. An illustration over how information about design, code, examples, introduction and other information is placed and structured, done with the help of shapes, in the Australian Government design system. Authors own image.

In their component page the components are showed with icons and organised depending on how complete they are. Below the components there is an area displaying what is in progress and is suggested for changes further on, but there is also a link to their community. When picking a specific component to find more information about it, there is an overview page explaining the component and the different versions of the component, together with an interactive example and lines of code. The other tabs are *"Rationale"*, *"Accessibility"*, *"Code"* and *"Discussion"*. The *"Rationale"* focus on design guidelines while the *"Accessibility"* covers colour blindness and keyboard accessibility, among other things. The *"Code"* tab includes code examples and the different properties that are possible to change within a component.

7.1.3 FUTURE LEARN

FutureLearn's design system follows another denomination than the others, calling its smallest part for atoms, and then larger ones for molecules, adapting the Atomic Design thinking of Frost (2016), but modified. Alongside these atoms and molecule descriptions is a style guide and available sketch file, and instructions of how to use the atoms and molecules in code. The source-code is not open to view; however, it is a work in progress. In total there are about 200 employees at FutureLearn. An illustration over how the design system show and combine different kinds of information is shown in figure 6 below.

Figure 6. An illustration over how information about design, code, examples, introduction and other information is placed and structured, done with the help of shapes, in the Future Learn design system. Authors own image.

The Atoms page has boxes with the names of the atoms. An atom page contains how to use the atom and has visual static examples. Code usage has some descriptions as well, but there is no complete overview of the documentation of the code.

7.1.4 GOV.UK

GOV.UK's design system was created to make their different services consistent, with arguments taken from user experience research. It was created for developers and designers to easily use the code in their projects. The system includes a style guide, components, patterns and a community where the system can be discussed for improvements or contributions. The design system is used by a large number of teams and has a multidisciplinary team responsible for it, which receives contributions and suggestions from users and stakeholders of the system via the community feature. An illustration over how the design system show and combine different kinds of information is shown in figure 7 below.

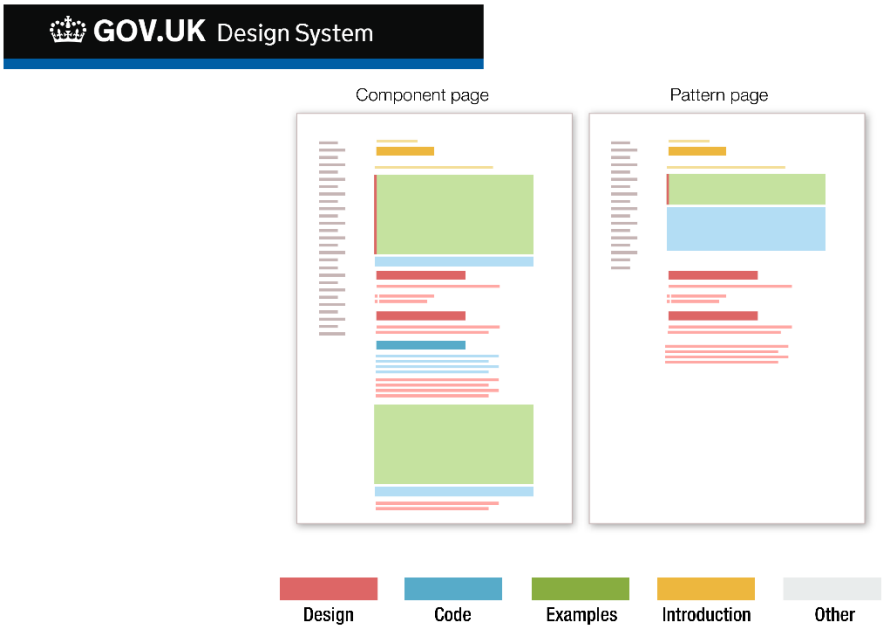


Figure 7. An illustration over how information about design, code, examples, introduction and other information is placed and structured, done with the help of shapes, in the GOV.UK design system. Authors own image.

In a component page there is an explanation of when to use the component and how. It also starts off with an example with corresponding code, showing the primary version of the component and then examples of the variants. A patterns' page describes how to use the components when wanting to build a design-solution for the user to perform a certain task. These patterns could be filling in email-addresses, telephone numbers and other tasks.

7.1.5 SHOPIFY

Shopify's design system Polaris is centred around building a great experience for their customers. The design system was created to let developers build the best type of experience much quicker, with thorough documentation regarding components, patterns and design guidelines. The Shopify design system Polaris is available for download on GitHub and in Polaris there is a guide on how to get started using their design system. An illustration over how the design system show and combine different kinds of information is shown in figure 8 below.

Figure 8. An illustration over how information about design, code, examples, introduction and other information is placed and structured, done with the help of shapes, in the Shopify design system Polaris. Authors own image.

The components are organised after what purpose the certain component has, for example if it is an action component or feedback component. They also have a lot of best practices in place to ensure their products built with the design system are as good as possible for their customers, which are merchants.

7.2 COMPARISION & SUMMARY

From this benchmarking, almost all the researched design systems follow what Kholmatova (2017) means with design systems, with patterns that are built up by different smaller parts. Only FutureLearn have adapted more to Frost's ideas about Atomic Design (Frost, 2016).

Some systems are quite alike, having distinctive hierarchical categories concerning the size of the UI object described. Atlassian's design system does not show code examples alongside with the component descriptions, but have a whole other external site connected to the *"main part"* of the design system. This site explains how the components are built in code alongside with downloading so called *"packages"* with the premade components. Other design systems, like Australia.gov and Shopify's Polaris have a more mixed approach to design and code and

includes the documentation for both on the same component page. Noteworthy of the Polaris language and the categories for their components are that it is all very central to their core idea, which is creating pleasant experiences for their merchants.

The different systems have different focuses on the receiver of the information, or who is looking for information. Both Atlassian and the Australian government have separate tabs in a way where it is possible to decide as a user to filter which kind of information you want to have, for example about design or the code. Atlassian even has an entirely different site that still is tightly linked to the source coded design system. The way the different design systems present their components also differ, from menus with names to a grid of icons symbolising the components.

Although the systems were analysed from what was available publicly, it did not give the whole picture of the different design systems. It does not give an idea of how the teams behind the design systems work with feedback, it just displays how it can be gathered. Interesting aspects to make note of is that the Australia.gov have their change log easily accessed and the contributors to the components are displayed. They also have an overview of what components are released and what components are in progress. Another aspect of noteworthy feedback is the documented research connected to the components in the GOV.UK design system, as it serves to verify and suggest what there is to be done with the component concerning usability testing or issues.

Noteworthy is the different approaches to what is shared publicly and how they approach the community around the design system. Both Australian government and GOV.UK put a heavy emphasis on the community and how to contribute to the design system. Atlassian and Shopify do have open source code, but they do not have it as much in the spotlight as the government design systems has. In the government design systems, there are transparency surrounding what is in progress currently and what is going on, trying to make it accessible for the contributors with the purpose to feel a common ownership towards the design systems.

8 CASE STUDIES

For the case studies, UX-professionals out in the field who are users of design systems in different ways were reached out to. In the case studies it was investigated why the design systems were started, how they functioned, which problems had occurred and what the biggest benefits of implementing the systems were. For this, three different companies were investigated. Below is a brief explanation of each company;

- **Case A** - The researched company is global with around 3 000 employees around the world, that works with optimising certain parameters within their business area.
- **Case B** - The researched company is similarly to **case A**. It is a global company with approximately 20 000 employees worldwide. They work with various technical services, amongst others providing service for the everyday consumer.
- **Case C** - The researched organisation works for the public sector in a European country, aiding their citizens and providing services.

8.1 REASONS FOR CREATING A DESIGN SYSTEM

The design system of **case A** was created due to the company wanting to handle projects under time pressure, but also projects with technical debt. The design system was created to help with reducing the time for creating new front-end and the company realised it would help to create applications by already having interactive components complete. This since the components already had built in coherent design and visual expression. The company offers several services, which all looked different from each other, and which all belonged to different product owners of those services. This created a large variety of elements and even colours that were all out of sync as there was no standard way for doing certain tasks in the development of the products. Currently, no customer is buying every one of the company's services as a package, however that is the ultimate goal. But if the customers would buy all services, they could get utterly confused from the variating systems and applications.

Before the design system in **case A** got set up, the selling point for those in higher, leading positions was that a design system would save time and create a more sustainable coherence between the different programs. Another selling point was that the system would be less dependent on individuals, since there would exist documentation about why things were constructed the way they were. The design system was allowed to be created, but not to a full extent, ending with no appointed product owner due to the lack of formality in the decision.

The company of **case B** is a large worldwide company which has a large market share of their business in Sweden. The company started to create a design system when they realised that their website was very inconsistent concerning how they made certain components, for examples search bars or buttons. The company has multiple designers that are responsible for their own different pages on the site. This could make it look quite different depending on where you looked on the site, since the pages were created from the separate designers' own preferences. The idea of a design system arose with the purpose to stay to the same unified vision, making designers cooperate, having a similar layout throughout to create a coherent look for their users.

Case C's design system belongs to a European country's government that decided to create a design system for their web applications, both to their external services towards the public and internal services. The purpose of the design system was to create a gathered system to maintain coherence and quality throughout their organs.

The idea of a design system for **case C** initially started because there was trouble with the cooperation between teams within the large organisation. Several teams of developers worked with different packages of components that other developers needed. There were several difficulties when these developers found the packages, such as finding them and then also understand them. This became unnecessarily complicated, and therefore the idea of creating a design system was born. The thought behind creating the design system was to remove friction from building a product, leaving time for more iterations. This way the designers could think less about how to create the basics and refine the more complex things on the site.

Initially for **case C**, it was difficult to persuade those that held the budget to invest in the system. Eventually this was solved, but it took around two years, and the leaders needed a lot of convincement from both designers as developers to be fully on board with the idea that a design system was needed.

8.2 THE DESIGN SYSTEMS FUNCTIONALITY

The design system of **case A** is internal and not displayed to the public. This company chose to adopt the philosophy of Brad Frost's Atomic Design (2016) for the design system's structure. Firstly, they had to clean up amongst the messy existing components, colours and other smaller parts the different services of the company contained. Sorting components into atoms, molecules and organisms in the way that Frost (2016) describes it has in hindsight been regarded as aggravating and a methodology that works better in theory than in practice. This since it was quite difficult to classify things in the set categories when it was not always obvious were the certain component belonged. The Atomic Design was also explained as not creating a clear enough interaction pattern with the smaller parts or components. When viewing the documentation in the system, it is possible to either just show graphically what a component consists of or show in text why things look and act the way it does within it.

The design system of **case B** is internal and not displayed to the public, and it is only accessible through the company's internal system. To structure their components in their design system the company has also adopted the Atomic Design philosophy, similarly to **case A**. The UX-team has a Sketch-file with their components and the developers has their library corresponding to the Sketch-file, however the UX-designers are generally not aware of how the developers use their code library connected to the system. The company has a common library they call *"the Component Bank"* where they share components between teams.

To maintain and develop their process and teamwork in **case B**, the designers and developers have their own meetings discussing the system. Unfortunately, these meetings tend to overlap, so designers cannot be a part of the developer's meeting and vice versa. When creating a new component for the system, the product owner gives certain orders to the designers of what to design for a new component. The designers then check if the component already exists in the system, and if it does not, they have to bring it up to a design council for discussion. However, if there is not enough time for creating the new component thoroughly, it will be roughly made and not so good. Most often, there are time pressures on the component creation, ending with the design system having inconsistent components, which contradicts why it was created in the first place.

The design system of **case C** is open to the public and follows the hierarchical structure of smaller parts being built into bigger ones, with elements, components and patterns. The system has around ten to twelve people in a multidisciplinary team that work solely with maintaining and developing the design system. This team include developers, designers, a project leader and an operations manager amongst others, who all work together to make the design system as good as possible. The team of the design system maintenance include user researchers, that tests the components in the system, to show quality throughout the system. Each component in **case C**'s design system has verification information, showing if the component has been verified or tested, which will help in the convincement of both designers and developers to use it.

A growing a community has been very important for **case C** since their design system is open to the public and is available for anyone to download. The fact that it is available for anyone is a very large difference from **case A** and **B**. The design system for **case C** uses the community as a method to make their business design recognised, but also to centralise knowledge. The

design system is treated as an artefact that brings a lot of teams together, letting them focus on collaborating and building something good made for all.

8.3 PROBLEMS & SOLUTIONS

Case A had a difficult task as they had to start over on a clean slate when creating the design system, since the different programs and services varied tremendously from one another. Things varied a lot in looks between the different systems and to clean up between the different variations proved to be quite difficult and time consuming, but nevertheless an important stage in creating the final design system. There was no appointed product owner in **case A** either, making it difficult to drive and develop the design system forward. It made the designers being those pushing for it, and they got the developers to help with the implementation of it. The teams working with the services, and the design system, are teams situated in Sweden and India, making the communication more difficult to maintain and sustain in a good way. A problem for **case A** was also emotional, since the developers felt as if their work could become redundant due to code becoming more easily accessed and easy to copy. Therefore, they had to be convinced of the fact that their jobs would not disappear, but rather become more detailed and focused rather than having an *"overall"* focus. The design system cannot contain all possible components, just the general ones. Therefore, they could focus further on detail design and special components rather than having to spend time on building the basic structure over again for each new product.

For **case B** a problem is that the design system is not online. As the design system consists of two separated libraries, one appropriate for developers and one for designers, it is difficult for the developers to see the designer's version of the components and vice versa. However, the system is completely shared between the different designers and developers respectively, ensuring that they stay to the same design language and components. Currently, **case B**'s site does not perfectly meet the Web Content Accessibility Guidelines, the *"WCAG"* (W3C, 2018). In the future the company preferably wants to start doing more to work with their sites accessibility to ensure that they can meet all their users' needs and demands.

Case C did, similarly to **case A**, have problems with the developers and designers feeling as if their work would become redundant with less creative freedom. However, this was proven to be unlikely as there always will be problems to solve that you cannot plan for, which requires freedom and creativity from their sides. A design system is supposed to be general, enabling both developers and designers to focus on fixing specific details for their own products. In **case C** there was also a problem with the first trial that did not work properly and required a start-over to build an entirely new thing. However, this was planned since things do not tend to turn out perfectly the first time around. It also took more time than planned to develop the system, changing from the planned six months to an end duration of 18 months until a viable product could launch.

8.4 BENEFITS

To have a design system has brought several benefits to the different cases investigated. For **case A**, they have gathered all their different components in one place, to have a better overview of what they have created and can use for future work and development. The company and development teams have also found that they can save both time and money on product development. The work and development also becomes less dependent on individuals since documentation becomes gathered and displayed instead of being contained inside developers and designers heads. Another discovery has been that it created a more sustainable coherence between the different services that they offer, since over time all their services will look and act the same as they will work according to the design system.

For **case B** a big benefit has been the communication and mutual understanding of the different components between developers and designers, both in the way that they are named but also comprehended. This since they have two libraries that are meant to correspond, showing the same information in different forms, where one is directed at the designers and the other is directed towards the developers. As well for **case B**, they have found that the design system helps with continuity since the company is consultant heavy, and people frequently rotates. The design system can help with transitions between people and the newly started people can in a quick, easy way start working and also understand the existing structure of the site.

Also, for **case C** the mutual understanding has been a benefit. **Case C** has rather used the design system as a mediating object to discuss around, while **case B** has used it more to ensure that things are titled the same way. In both **cases B** and **C**, they have found that their sites become more coherent to the public since they are displayed publicly, which is a difference from **case A**. For **case C** the design system has helped hundreds of teams inside of the organisation to stay to the same guidelines and ways of creating and talking about different components on their site. For **case C**, it brings their teams together since they can all contribute to it, suggest improvements and working with developing the company further, making a large organisation feel as if they are all a part of the system and can contribute. A big benefit also mentioned from **case C** is that they quickly can create sites that are minimally viable, which can leave space for quick iterations in the development process.

8.5 CONCLUSION CASE STUDIES

The case studies provided information that was not possible to find out through the benchmarking, providing a better understanding of how an implementation of a design system affects the company and what is difficult and beneficial with starting a design system. Those details are not possible to understand through the polished fronts that the design systems published online provides.

From these case studies it was possible to find some common themes in the investigated companies. All cases had one aspect in common for starting a design system, which was to create a better coherence in their service or systems. They also started a design system to be less independent on individuals in an everchanging environment and wanted to create better cooperation between the people involved.

The cases investigated mainly used Atomic Design by Brad Frost as a foundation, however not being entirely satisfied with the outcome of that. Two out of the three cases investigated were private design systems, only getting utilised by those working at the company, whilst one was open to the public and showed what happened with the development of the system. One of the cases were basically built up of two different design systems that kept their contents updated and synced with separate meetings that discussed the development of the system. For one of the cases the organisation works multidisciplinary between team to maintain the system, and quality test their components by performing user tests on them.

There had been some problems when developing the different design systems of the cases, with for example having to start from the beginning, sorting through content, being tedious work. There had also been a common problem having the designers or the developers doubt their freedom with implementing a design system. They had to respectively be convinced that their work would become more meaningful, leaving space for creativity and detail work instead of working on the shell of a new application. A problem not always thought of as is the development time for a design system, as it can take quite a long time before being able to make a fully working and good release.

In all cases it is shared that a design system has brought a lot of benefits, such as easing communication and mutual understanding between team members and other teams. A design system can save time and money in different ways and will help the continuity between workers and new jobs, while also bringing a large organisation together. Most importantly it has brought coherence between the different applications and systems provided by the cases, thereby fulfilling an aspect to why they created the design systems in the first place.

9 INITIAL USER STUDY

An initial user study was performed together with the users developing and using the existing solution for component collection and documentation at the region of Västra Götaland IT department, the so called *"komponentkartan"*. The documentation that existed was of the code but could also be found on the belonging demonstrative website which displayed interactive examples together with text regarding how it could be used for example. The user study was performed to understand in which part of the documentation that different users were active, but also what they used the information for and what their opinions were about it. With the background of the literature study, benchmarking and case studies, the initial user study was made to discover and more thoroughly start to understand the existing system, identifying possible problem areas and important factors with the system.

9.1 RESULTS

Currently, the system that VGR has created that is similar to a design system is called *"komponentkartan"*, which translates into *"the component map"* in English. The system has a history of mixed people working with it over time, and no one has taken responsibility for it, making the documentation incoherent.

9.1.1 HOW KOMPONENTKARTAN FUNCTIONS

The current system includes a code repository, which has gotten the name of *"komponentkartan"*. But there also exists a demonstrative website that displays what is in the code repository, called *"komponentkartan demo"*, adding the word demo to the same name that the repository has. Both these are hosted open source on a website called GitHub, available to the public to download and use. The demonstrative website contains interactive examples together with explanation texts for different components, together with further UI information. There has been an issue in separating these two, and between designers and developers the term and word of *"komponentkartan"* gets interpreted differently, since the designers refer to the demonstrative website when saying the name, whilst the developers refers to the code repository. There have been difficulties with creating the documentation for the demonstrative website and it is not coherent throughout. Currently, the documentation and testing are done on the same site, and the relationship between these and the code repository could become clearer. For development, it had also been stated that to become a component in the *"komponentkartan"*, the component needs to be general. A component should not be created with the sole purpose of only one system but should be able to be applied to several. However, it is not possible to still apply something general to everything, and for some systems the teams still have to create their own components. The current solution of the repository and the demonstrative website was created for basic needs and not all possible needs.

Since the demonstrative website *"komponentkartan demo"* is built on an inconsequent foundation, it is still considered to undergo development by most of its users, not yet being a mature system. A problem with the tool is that it can introduce a type of slowness, since other projects could be waiting for things to be implemented in the system, since they use components from the repository. The repository together with the demonstrative website may speed up the development process, but the developers still has to be competent enough to build their own components. Currently, no one is owning the system and works with its progression and development, which have halted it. It is due to this that things have progressed slowly and has no standards or guidelines to follow.

The documentation that currently is displayed on the demonstrative website is copied from previously created component pages with some changes in the text, not following a standardised template. Those creating new components have tried to replicate and keep a similar structure to what someone else earlier have done, whom in turn probably have done the same thing when creating that content. Those developing *"komponentkartan"* have had the intention of trying to make it appear more coherent, whilst making it deviate in practice. It was also expressed that all documentation is not desirable to be viewed for all the users, but if one type of user does not want to see it, it does not mean that all of that kind of users do not want to see it at all. Developers, for example, want to understand the structure and parameters of a component and do not want to read extensive text, but that does not mean that they should not be able to read it if they would want to. The documentation should be at an appropriate level for the user, without added extra complexity.

One part of the documentation on the demonstrative website is examples. They can be experienced as too much for some developers, who would rather prefer fewer, more thoroughly worked out examples that could cover a wider variety of components and usage. The developers experienced that some examples are too styled and not applicable to their project.

However, designers express that they want to understand how components work in context, to understand how they act when clicked upon and so on, since they then better can understand how the interaction works. This shows that developers and designers want different information from the examples displayed on the demonstrative website. However, some examples are more complex than others, and there have been several *"example pages"* created that combines different components in different ways. These pages have been created mainly for code testers of the *"komponentkartan"*, so that they can test and further understand a component in a context and not just as a standalone example. This have created a lot of pages with multiple examples and demonstrations with components on the demonstrative website, which does not have any further explanations to them.

Another part of the component documentation on the demonstrative website regards the accessibility of the regarded component. This section describes how the component follows the guidelines of web content accessibility, the WCAG (W3C, 2018), and it is important that this stays put for further work with *"komponentkartan"*. It was expressed through the interviews that a backlog would be preferable, that is a backlog that is more manageable than the current ones that exist, to ease with the understanding of what has happened with the system.

9.1.2 BENEFITS WITH KOMPONENTKARTAN

There are a lot of benefits with the *"komponentkartan demo"* demonstrative website, and one prominent thing is that sites and systems gets built in a similar way, and the basic components for the site is already done before starting development. This increases development speed and facilitates for new design and detail design. It is also good since novices gets a quick overview over which components exist in the demonstrative website *"komponentkartan demo"* and can aid their onboarding process. It is a tool that work with ensuring a coherent work, where things will look the same in the end since it has followed the same guide. This agrees with the benefits found in section 6.2.

9.1.3 HOW KOMPONENTKARTAN GETS USED

When the developers use the demonstrative website *"komponentkartan demo"* they do not directly copy the HTML-code that stands there but do rather collect it from the repository *"komponentkartan"* instead. This puts the demand that the components should be general and well-functioning. However, since the developers import code from the repository, they do not know exactly what it will end up like, and do not know exactly how it will act, which can be a problem sometimes. The developers need the documentation to better know what they can do with the component. The designers work with a *"Sketch"* file built up of what the components look like visually, created from what they look like on the demonstrative website. This has been done separately from the repository and the demonstrative website to aid the designers in their design work.

The demonstrative website, *"komponentkartan demo"*, is used by designers to know which components have been created before and they can thereby understand which they can use, or if they need to create a new type of component. The repository and the demonstrative website

can be used by other teams than just those developing and maintaining it, but there exists an issue with them using it without a designer. This since designers may use the components differently and combine them in different ways than a developer may do. This could possible lead to lowered end-user experience. The demo website is a demonstration over what it could look like and not a tool to directly solve problems with. The components as defined by the code repository and the demonstrative website are utterly most used by developers since they are told to use them, and thereby have to.

The developers work with pair programming to help one another to find faulty code within *"komponentkartan"* and they share their knowledge with each other to make the final product as good as possible. The developers tend to find the unusual use cases, those that tend to be difficult to identify whilst the designers identify the most common ones, showing that they complement each other and can make the system very well-functioning.

The cooperation between developer and designer is that the designer sets the demand alongside with a Sketch-design file that the developers creates code after. These Sketch-files are built up of different components displaying what a page should look like on the site, easing the work for the developers since they can take and modify code from the different components. This file is shared between teams so that no one has to remake all of the components graphically again. It is very important though that the names for the *"graphic"* components in the software Sketch corresponds to what they are named in *"komponentkartan"*, to ensure good communication and language.

9.1.4 WHAT KOMPONENTKARTAN IS LACKING IN

In the system, the developers want the information of how the components code is built and what is in the code. They would also like to understand the flexibility of a component, what can be done with it and how that is supposed to be done. One thing missing according to the developers is a tutorial that covers how the code repository *"komponentkartan"* and the demonstrative website *"komponentkartan demo"* should, and could, be used.

For the system, the designers would like to know more about if a component is verified or tested with a user, which is not currently displayed. The designers would like to understand why components look the way they do, their history and why things are placed a certain way in the component for example. They would also like to see what is currently being worked on with the demonstrative website. The designers also think that to read a lot of text is more complicated than seeing and interacting with the components for real as demonstrative examples, which is done via the demonstrative website.

Some things on the demonstrative website are invisible to the designer, such as spacing and how the components can be combined. This information could possibly be explained in the API-section, but the designer does not understand the documentation explained there. During certain occasions, the designers and developers have not had a common view on what a component is, and the definition is difficult to directly explain as well.

There existed a divided view of how people should be able to contribute to *"komponentkartan"* code library and the demonstrative website. Some developers expressed that they think the ones contributing should have knowledge into the technical demands so that they can express the

correct things, whilst some thinks that anyone should be able to lay suggestions. This is something that should be further investigated. However, those contributing can also be the ones creating and using the system, which possibly could create biased content. It is important to think about that the components should be kept general, since the more specified components become, the more likely it is to introduce errors into the system.

It may be considered odd that the developers and designers are the ones setting requirements to *"komponentkartan"* and its code repository and demonstrative website, since they are both suppliers and consumers of it. This could make them biased to what goes to the different libraries.

The biggest thing identified throughout as a common, re-occurring theme is that there is no product owner to the system, making its progress slow, unclear and biased from the users. This will be important to investigate in the future as well, ensuring that problems can be better solved with a future version.

10 CONCLUSION OF DISCOVER PHASE

The phase of *"discover"* opened for interesting subjects and points to elaborate and look further into for the phase of *"define"*. Three clear subjects were identified in the form of the content in the design system, the design system working as a language, and finding further benefits with the design system. These themes are further described below, but for the next phase these themes were to become further investigated to find the needs of the users of the design system.

10.1 CONTENT

The content of the design system is important to consider, together with how it is presented, to whom.

Connection of content – Several parts were identified to belong to a wider description of a design system, like the components in the code repository *"komponentkartan"* and the demonstrative website *"komponentkartan demo"*, together with ideas from the users on guidelines and a better process. For the next phase, the define phase, the aim was to understand more about how they were connected and how they better could become connected, to better become distinguished and presented in a final product.

Structure – The overall understanding of the current system was investigated initially in this phase, however going forward it was important to break the system and its content down to its essential parts. This was to be further investigated, concerning how they can be structured to optimise for the users of the design system.

Documentation – From the discover phase it was made clear that the current documentation on the demonstrative website was insufficient. The documentation was inconsistent and for the users it was not clear who the documentation was intended for and what they were supposed to use it for. The amount and handling of documentation was also an area for further investigation, as there needs to be a correct balance of the documentation for it to be useful. In

the next phase the documentation and its prioritisation were to be defined, based on the different user groups and their needs.

Examples – Like the documentation the examples were also found to be inconsistent and with an unknown intended user. For the progression of the design system it was necessary to understand what purpose the examples serve and how the examples could serve the users of the design system in a better way.

10.2 THE DESIGN SYSTEM AS A LANGUAGE

A design system is larger than just a pattern library. To apply the design system thinking to VGR's remuneration systems, it is not enough that their pattern library works well, but the process and collaboration around it also needs to work. To be able to find out how this kind of thinking could be applied to VGR, there were a few subjects regarding the design system language that had to be defined.

Common language – To enable and ease communication the literature suggests bringing the terms and words used in the system together to make sure they have the same meaning across all teams and users. A groundwork for collaboration is to understand each other, therefore the language used in the design system should become coherent, being a considered part of the development. In the next phase, the possibilities of working with this were to be identified.

Collaborative tool - Further work in the project were to investigate the design system as a collaborative tool between designers, developers and other users to define how this best could work to fit the respective users' needs. The collaboration could possibly be enhanced by a design system, and therefore the next phase was to investigate this considering the different users' needs.

Contributions – By contributing to the design system, its users will get a stronger bond to it. Contributions to the system would need to be further researched and understood to decide which ways best could work to enable contributions. As there can be different types of contributors to the system, their roles were to be taken into consideration of how they possibly could contribute.

Feedback loop – Continuous improvement requires feedback and a way of handling feedback. In the define phase it was to be further investigated how feedback currently was being handled when the system received it, but also what user needs existed regarding receiving and giving feedback. It the next phase, the feedback loop was to be further researched to understand how it could be applied in the current design system to further serve the users.

10.3 BENEFITS

As a design system does not exist without context there were subjects which needed to be dealt with for the design system to be able to be as useful as possible in its context. The benefits needed to be further identified, showcasing why it is beneficial to actually have and implement a design system.

Product owner – A thing noticed through both the case study and the initial user study was that a product owner can help structure the system to bring it forward. The possibilities of having a product owner, or a constellation which purpose is to own the product, should therefore be investigated. This did not exist for VGR IT before, and the benefits that are possible to get from a design system cannot survive in the long run if it is not maintained.

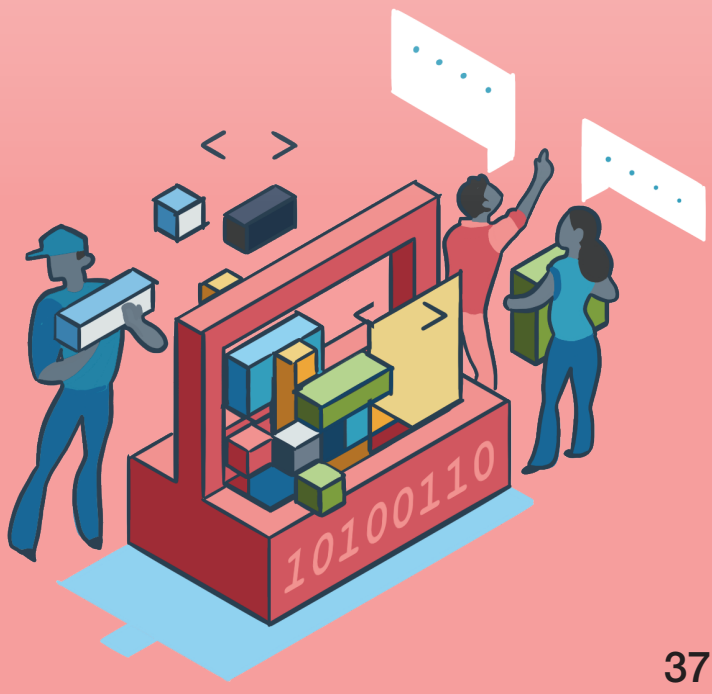
Other teams – It became evident through the initial user study that other product teams outside of those developing the repository *"komponentkartan"* and its demonstrative site had used and benefited from them. Therefore, the next phase was to investigate how those teams used it, and what they found beneficial or possibly lacking with it, to create benefits for them as well.

What is applicable to VGR – There are a lot of both successful and not as successful practices covered in the discover phase. A challenge to bring into the define phase was to understand and define what is applicable to VGR's context. Many benefits with having a design system was discovered through this phase, but to be beneficial they need to be adapted to the organisation and specific case that VGR IT has. This meant that it had to be defined which benefits would make the biggest impact for their organisation, making a design system fit in the right way.

Risks – Finally, for the next phase an important thing to look at would be what risks possibly exists with the found recommendations, benefits and structure. Most importantly in relation to the risks, there should be suggestions created for how those risks could be conquered, avoided or minimised.

Define

This section concerns the second phase of the project; define. The define phase took in the themes and subjects from the discover phase with the purpose to define them further. The phase set a definition of what a design system is and defined in more detail how the current system at VGR IT worked and how its different parts were connected. During this phase the users' needs and their profiles were also defined. The phase ended with insights which were taken into the next phase of creating concepts.



11 METHODOLOGY DEFINE PHASE

This phase started with stating a definition of what a design system actually is, based on what was found in the literature review to create one definition to follow for the remaining of the project. From the first phase, it became obvious that there exists no clear definition of what a design system is, and by setting a definition it was easier to continue with the project.

To then also properly understand the current solution at VGR, the project also more clearly defined exactly what *"komponentkartan"* is and how it works, before proceeding with the project.

After these definitions were made, the project used user-centred research methodologies to find out more about the themes discovered in the first phase. The methods served to gain more in-depth insights about what user groups there are and define them. Also important with the users was to understand which of them that should be most prioritised and supported by the design system. These methods ended with creating user profiles from the insights, to create guidelines for the develop phase. This chapter describes the different methods used, and their results and conclusions can be found in the following chapters.

11.1 USER INTERVIEWS

To gather qualitative data about the users of the design system interviews were chosen as the main method of extracting information. The interviews were customised to two different user groups: developers and managers. The developers were prioritised since they were considered the most important users of the system and needs to be understood in order to make the design system work for them. The second group of managers is a collective name for those in charge of the development, who decides whether to use tools to try and increase efficiency or not. This since their needs need to be met as well for the design system to get accepted and become a central part of the development. For this round of interviews and evaluation, the designers were left out, as the insights about them was considered sufficient from the discover phase.

11.1.1 INTERVIEW WITH DEVELOPERS

The interviews with developers in this phase of the project were performed to gain more insights about them to define the different dependencies that the current system has together with development. These relations in the process were important to understand when developing the existing system, since it showed how things were connected and what the developer's opinions on those connections were. It created more needs and guidelines to follow in further development.

The sample of interviewees was chosen based on the experiences the developers had with *"komponentkartan"* and the demonstrative website. It was necessary to find interviewees that had knowledge about how things had emerged and were interconnected to get the desired result.

For this study six developers were interviewed, which worked mainly with the health care remuneration systems, with one working outside these systems but still within health care

closely tied to it. One of the interviewees was also a tester of the components and the demonstrative website *"komponentkartan demo"*. The interviews were semi-structured (Wikberg Nilsson et al., 2015) and used pen and paper together with words on post-its as a mediating object to help the discussion around the different parts of the process. The interviewees got to both discuss around pre-planned questions, but also use the mediating object to discuss around to aid with provoking thoughts. The post-it notes had prewritten words on them, such as *"komponentkartan"*, *"Angular"*, *"documentation"* and *"backlog"* to mention a few, see figure 9 below for a picture of what this looked like. The interviewees were asked to put these words in relation to each other and explain that relation, whilst also explaining their interpretation of the word. The order in which these words were given was pre-planned to go from the bigger parts linked to the code repository's code and demonstrative website, to Angular and then also the parts about their working process, such as Scrum-work and its importance. The participants could also choose to write their own words on a differently coloured post-it note to fill in gaps if they found there were any. After this map was drawn up with the relations, there were follow-up questions for the developers to answer. Questions that regarded what they had created, to understand what they thought about the specific process they had described. The interviews were filmed to later help with analysis and recollection.

Figure 9. The mediating tools used for the interviews with developers in the define phase.

11.1.2 INTERVIEW WITH MANAGERS

To be able to find the concerns of the managers that oversee development at VGR IT, two interviews were held with different managers. In the discover phase it was made clear that the

managers as well are a user of the design system since they decide whether it is to be used or not. However, the needs they have for it are different from that of the primary users of developer and designer. The interviews were centred around what they thought about their own work and what they value with development, but also what they know about of the current *"komponentkartan"* and its demonstrative website. Those interviewed were one object manager responsible for the applications developed towards the business of VGR and the other person interviewed was a Scrum master for one of the remuneration teams within health care.

11.2 KJ ANALYSIS

The interviews were analysed by using a modification of the KJ-analysis (Martin & Hanington, 2012). This meant that from the transcriptions of what the interviewee said, certain comments and opinions were highlighted. These text bits were later grouped together with similar quotations to create themes upon which areas was identified that were important to think about for the concept creation further on. The identified areas then were translated into needs, demands or guidelines for what the respective user groups wanted out from a design system, and what is important to make note of and think about during further development. The analysis was made in a digital manner instead of physical notes.

12 DEFINING DESIGN SYSTEM

A definition of what a design system is was defined in this phase to ease further work, bounding together several definitions to have one unified view.

Throughout literature, it became clear that a design system is a process that should work according to set standards and rules. The design system contains several parts that are documented to enable for others to access the information. These parts are the code repository, the style guide and the pattern library. The system itself can with all of this become a language and a mediating tool between different types of users, such as designers and developers.

The code repository is where the code exists and becomes assembled and structured. It contains the programming language and follows programming frameworks to create more efficient code. The repository is where the different reusable component exists, since their code is programmed into it.

Then there is the style guide, which is an explanation of how to create visual coherent content, such as which colours or typography to use, correct spacings and which format to follow on page layouts. The style guide is more connected to the brand and can contain guidelines as well as specifications of how to think about expressions when creating content.

What gets called the pattern library contains the reusable parts of a user interface, and it gathers the documentation about the code and styles together with these reusable parts. The reusable parts can be components and user patterns, where the patterns usually consist of several components interacting in a certain way. The pattern library could be compared to *"the face"* of the design system and is often published on a website. The pattern library makes it possible to work with the resources of code and style together, providing documentation for it in the

form of text and interactive examples to better understand. It is also important that the pattern library stays in sync with the code repository and the style guide, keeping all information up to date and not static.

Since the pattern library bounds different parts of the design system together, it can be misinterpreted of being a design system. However, it is different from a complete design system whilst still being an important tool in it. A design system has as earlier mentioned a set of principles and rules to adhere to. For example, this could be that every component and pattern should follow the WCAG, and if those rules change, so should as well the content in the design system. The principles and rules are reflected in the design system by introducing standards, that the users have to follow when developing the content in the system. It creates design standards between the pattern library and style guide, such as a usability focus that aids the use of the components in the pattern library. It also creates code standards between the code repository and pattern library, which structures code and provides help of how it should be written to help different developers to cooperate.

The style guide sets design guidelines to the pattern library, and the code library sets code guidelines to the pattern library. These guidelines derivate from a set of principles and rules made by the organisation and they tell of how things should be done or if there is a special goal to achieve with the design system. Such a goal could be that all the components are following WCAG, both in design and functionality.

But as firstly mentioned, the most important thing is that a design system is a process. It is a product created to serve other products, and therefore it needs to evolve, either ahead or alongside those products. The process of a design system does as well have to fit into the organisation that choses to have and start a design system. If the process that a design system is cannot fit into the organisations process, it will become neglected and left behind, which in that case becomes unnecessary work. To see a whole figure over what a design system is, see figure 10 below.

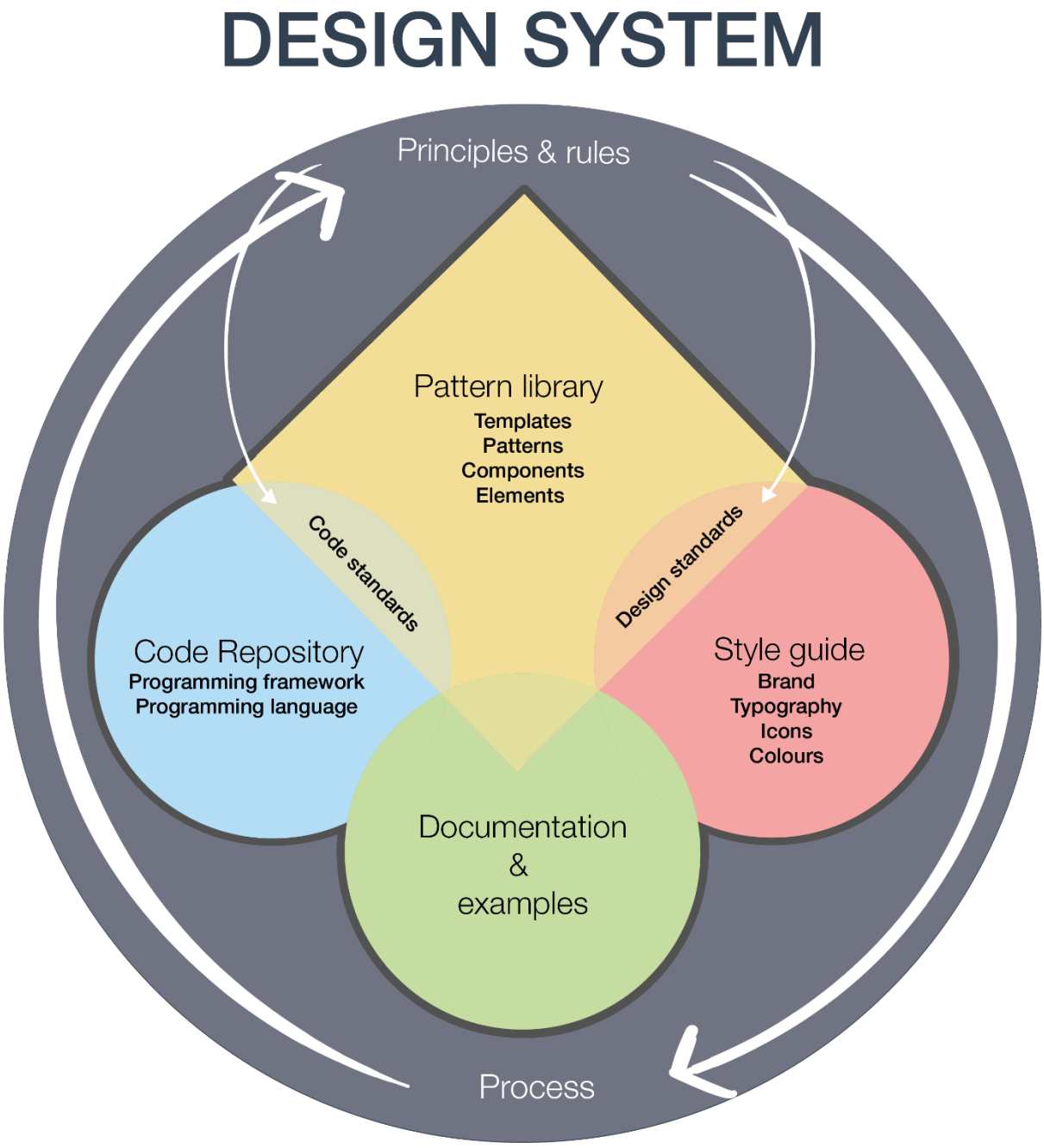


Figure 10. An illustration over how a design systems parts may be related, working together in the process that a design system is. Authors own illustration.

The output that will be from a design system is the products built with the design system, rather than the design system being something that exist separately from the products. The fact that a design system is said to be a process is that it needs to constantly be worked on, improved or alter to fit in with the changes happing around it. There can be updates in software that needs to be adapted to and as technology changes, as well does the design system have to adapt to it.

13 DEFINING KOMPONENTKARTAN

The existing solution at VGR, *"komponentkartan"*, was found out to work in a very inconsistent way. From the research done in the earlier phase, the current code repository and pattern library could be defined and compared to the design system definition made in chapter 12.

13.1 EXPLANATION

"Komponentkartan" is a tool that currently gets used and developed between projects at the IT department of VGR, especially by the teams working with remuneration systems. What is called *"komponentkartan*" is a code repository that contains the component code that helps during development of remuneration systems for the health care. It has also helped teams outside of the remuneration systems to help with their respective products.

A code repository is a collection of code, sorted into different categories, similar to a library. A repository is divided in different types of code, such as visual coding, website coding or arithmetic's needed to perform code in different programs. In the repository of code, it is possible to create so called *"components"* which can be reused. The biggest advantage with having these components gathered in the repository, is that when a button is used for example, there is no need for having to change each instance that uses that button, but you only need to change it in one place in the code and it will change all the buttons. In the case of *"komponentkartan"* it holds both the code for the components and the pattern library website. So the code repository does just display code, not visuals. In figure 11 below it is possible to see a screenshot of the code repository.

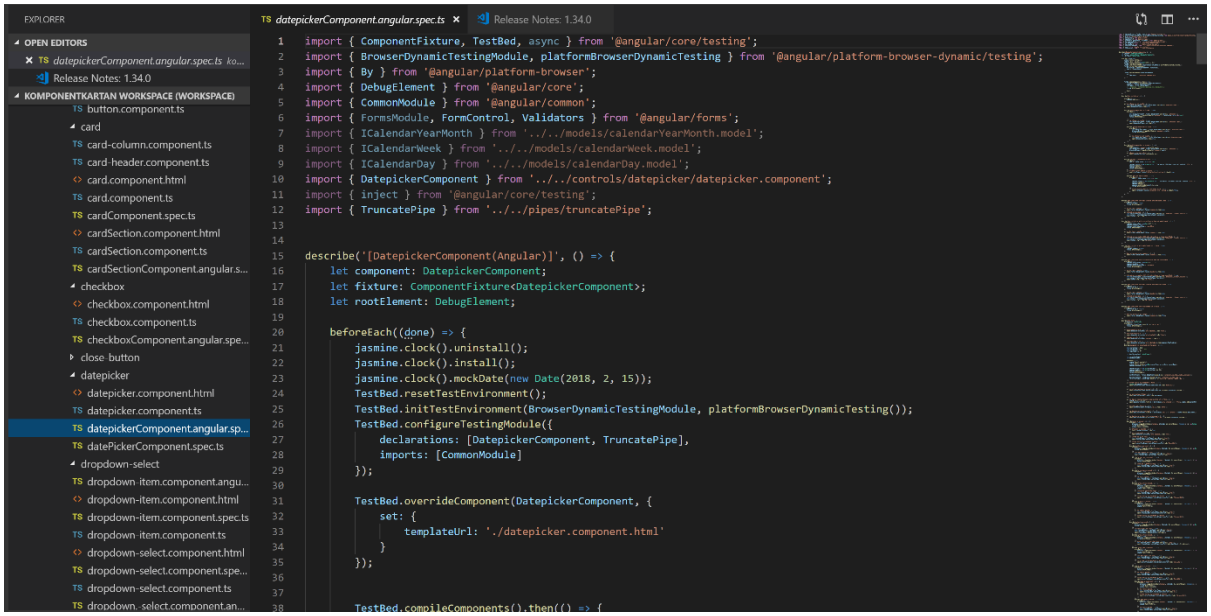


Figure 11. A screenshot of the component named "Datepicker" in the Code Repository, opened in Visual Studio Code.

There is also a demonstrative website that shows documentation about what is in the code repository, with interactive examples and describing text. This website is called *"komponentkartan demo"* and the code that structures the site can actually be found in the

repository, so the demonstrative website is nested within the repository. This has understandably caused confusion, and the names of the repository and the demonstrative website have been mistaken for each other.

The demonstrative website can be considered a pattern library since it shows the functionality of the components through interactive examples alongside documentation in the form of text about how they function. For example, a page about a specific button is not just a picture, but it is also possible to click on the button example to interact with it, to see how it reacts to the action. However, the documentation about components is inconsequent throughout the different pages on the site. It becomes clearer the more you look around that the quality and thoroughness varies significantly. To see an example of what the documentation about a component looks like on the demonstrative website, see figure 12 below.

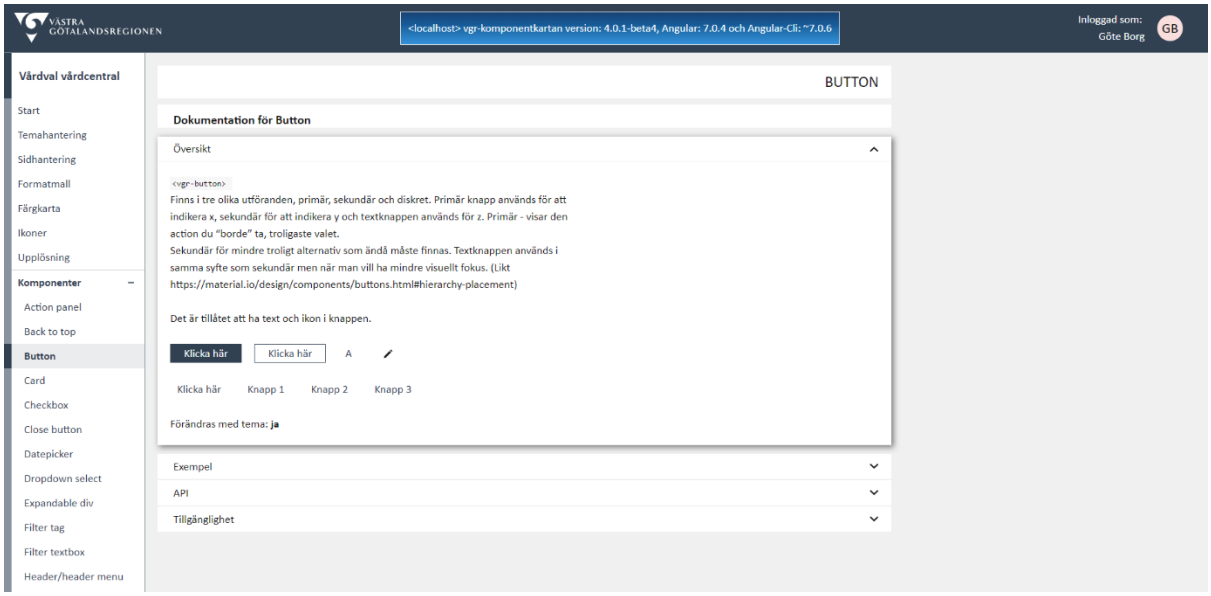


Figure 12. A print screen of the demonstrative website "komponentkartan demo", here displaying the layout of the documentation regarding a button component.

For a component description page, there is also information regarding the API, which is of interest of developers as well as designers, since it explains the flexibility of a component. Flexibility is what properties are possible to change and alter within a component. Another important part of the documentation is the so called *"Accessibility"* section, which tells the reader if a component follows the standards of the Web Content Accessibility Guidelines. It is a description of for example if it works with different web browsers, or what happens when you use the keyboard to navigate.

But beside from having the component documentation on the demonstrative website, there are also example pages made up of several components, to display them in a bigger context. These have low to none documentation and were created for code testers to be able to test that the components work in a context, and not just as stand-alone elements.

Also, there are pages of documentation regarding which colours to use, which fonts to use and which widths to think about when creating applications and sites from the collection of components. Normally this type of information is found inside of a style guide and currently this information is quite scattered and not contained under any gathering points.

13.2 COMPARSION TO THE DESIGN SYSTEM MODEL

These parts with the code repository and the demonstrative website could possibly be considered a design system. However, *"komponentkartan"* and its demo website does not have a proper process behind it, left without an appointed product owner, and does therefore not evolve and get developed well enough. This since they have been created between projects without having a clear goal, which have left it inconsequent without clear principles and standards to follow. Only developers have been able to edit the text on the demonstrative website as well, since it is nested within the code repository. This has also been a factor to the work being halted in its development, meaning that designers and developers have not been able to cooperate with an efficient development of *"komponentkartan"*.

Going back to the definition made in chapter 12, the code repository could be directly referred to as being a *"code repository"* within a design system, and the demonstrative website could be compared to a *"pattern library"*. There also exists two available versions of the demonstrative website, one that exists internally on the VGR network, and another that can be accessed online directly. The internal version gets properly synched with the code repository, but the public one has been created on the side of the repository, and whilst still referring to it, its text does not get updated properly. This means that someone that views the publicly displayed version uses a correct code repository whilst reading documentation that can be outdated and may be incorrect. This again, is a problem in the process and the development behind *"komponentkartan"*, that needs to become better managed and updated to become a design system. A figure of how the repository and demonstrative website is towards the definition made in chapter 12, together with the sync of the two different existing demonstrative websites, can be seen in figure 13 below.

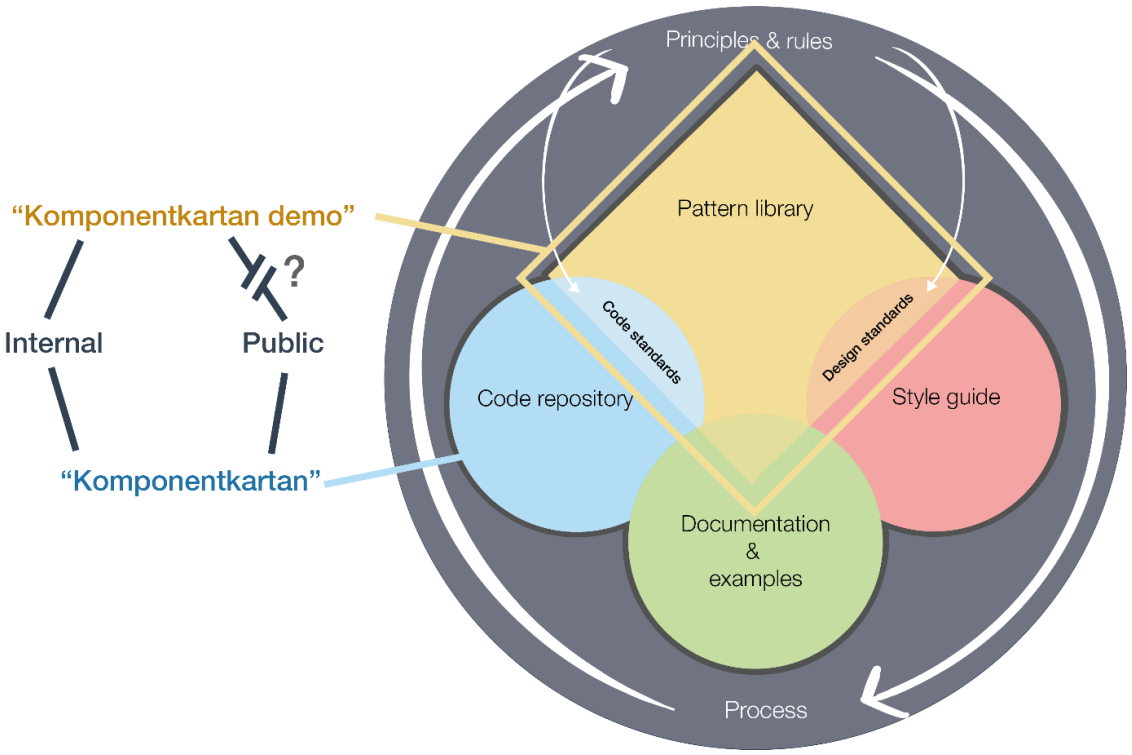


Figure 13. An illustration over how the code repository and demonstrative website of "komponentkartan" refers to the design system definition made in the project. The synching problem between the two different versions of the demonstrative website is also displayed. Authors own illustration.

14 INTERVIEW RESULT & ANALYSIS

When starting to understand the current solution, and defining what is was, how it worked, and what was problematic, a new round of interviews were performed with developers and managers. The interviews performed with developers were made to understand how the development process relates to the code repository and the demonstrative website. The interviews with managers were performed to understand what they appreciate and value in a development process. For the different interviews a total of eight people were interviewed. In this chapter it is possible to read about the result and analysis from those interviews.

14.1 INTERVIEW DEVELOPER

The total amount of developers interviewed were six. The different developers were spread out over five different teams, working within health care and a majority worked with the remuneration systems for which *"komponentkartan"* was created. The interviewed developers had different roles of developing, from front-end developers, full-stack and back-end, to code testers. Those interviewed had varying relations to *"komponentkartan"* and the demonstrative website *"komponentkartan demo"*. Some of the developers had been a part in creating or developing it, while one interviewee barely had heard of it.

During the interviews the interviewees were asked to, with the help of mediating tools, create a picture of how different things were connected to each other. Some drew lines and made illustrations while others simply just grouped the post-it notes into different areas. The way they put the notes out on the paper did not matter as much as the way they discussed differences and relations between them. An example of what it looked like can be seen in figure 14 below.

Figure 14. An example of what it looked like after an interview with a developer, when using mediating tools such as paper, post-its and pens.

14.1.1 SIMILARITIES

In all the maps made by the developers their own product was in the centre. For this process, they used *"komponentkartan"* and its demonstrative website to build the product. In this kind of process, there was always tie ins with tests and project management of that product.

The developers all agreed that on the demonstrative website *"komponentkartan demo"* there exists examples and documentation that helps when using the code repository

"komponentkartan" for their final product. Therefore, the code repository stands closer to the end product they create, rather than the demonstrative website which is further away.

The designers and the way they work was a process that most developers did not fully understand, but most had some knowledge about their tools and the way they create sketches for them to work with.

Most of the developers talked about flaws in the current documentation on the demonstrative website, which they occasionally noticed whilst using it as a tool. Also, the examples documented on the site were considered inconsistent and it happened that they could not acquire the information they needed to use a desired component.

14.1.2 DIFFERENCES

The people not working directly with developing *"komponentkartan"* had a hard time understanding the process of the teams who worked with it. Neither did they know how their own product would be affected if there were changes made to the code repository. The developers working with developing *"komponentkartan"* had knowledge of future changes and could understand what implications it might bring since they worked with it alongside their assigned project. Additionally, the developers outside of the creation and development of *"komponentkartan"* was not using the internal version of the demonstrative website and used the version that did not get updated which is publicly available online. The developers outside of the development were not aware that the other developers used another version that was accessed locally on the VGR network. Those developers had therefore been using outdated documentation that did not fully match the components, which had led to problems during the development of their products.

Those developers that either work with developing *"komponentkartan"* or have a connection to it, tend to have a better understanding of what is going on with the system. This of obvious reasons, but this still means that those not closely linked to it feels as if they do not get enough information about updates and maintenance of the system, which they would like to have.

14.1.3 CONCLUSION

From this it can become concluded that there are four different groups of developers that currently uses the code repository and demonstrative website, and these are either in a team with a UX-designer, or not, but are also more or less involved in the development process of *"komponentkartan"*. The fourth group is also code testers which do not have that much insight into the development but tests and ensures the content in the code repository.

The product that each team works on is separated from the demonstrative website, but the code repository affects the final product since it uses components and code from the repository. Therefore, it sets a high standard that it should be fully working and thorough.

The documentation is lacking in coherency and the examples appears to be quite scattered. The different developers need better structure to make it easier for them to find and navigate within

the system. There is also a problem in keeping the demonstrative website and its documentation apart from the code repository with the code, mainly to the names being so similar.

There exists a wish of better knowing what is going on with *"komponentkartan"* and its demonstrative website for those outside the development of it. They need to be able to know what to use and how to use it, without looking at outdated documentation.

14.2 INTERVIEW WITH STAKEHOLDER

From the interview with the stakeholders it became clearer that they are positive to the thought of *"komponentkartan"*, how it can help their developers and designers to do a better work with their products. They could see the benefit of things being developed faster, providing less interruptions. Since processes are an important way of an organisation to work, a design system can aid the structure during the development of new applications.

However, this sets demands on the content of the documentation, and it should be able to be used by more than one team to ensure the benefit of having the design system. It is extra important that the content of the design system follows legislative demands and can aid its users with components that already are following WCAG. The documentation should also be able to evolve as the world around it does, creating a need for governance that can work further with it.

15 USER PROFILES

The results from the define phase is below summarised in user profiles for the different roles of users that were interviewed and researched. In this phase most emphasis was put on the developers as they were considered the most important users of the design system. Primarily the developers are the users which needs need to be met, but the designers as well have important needs that has to go in line with these. It was also discovered in this phase that managing roles has an interest in understanding a design system better and they have an important role which determines if the tool should be worked with or not. Therefore, the needs of managers were also important to consider in further development. This ends up in three different main users of developers, designers and managers. However, there are also subgroups tied to the designers and developers depending on how involved they have been in the development of the earlier code repository and demonstrative website. The users and their subgroups are illustrated in figure 15 below.

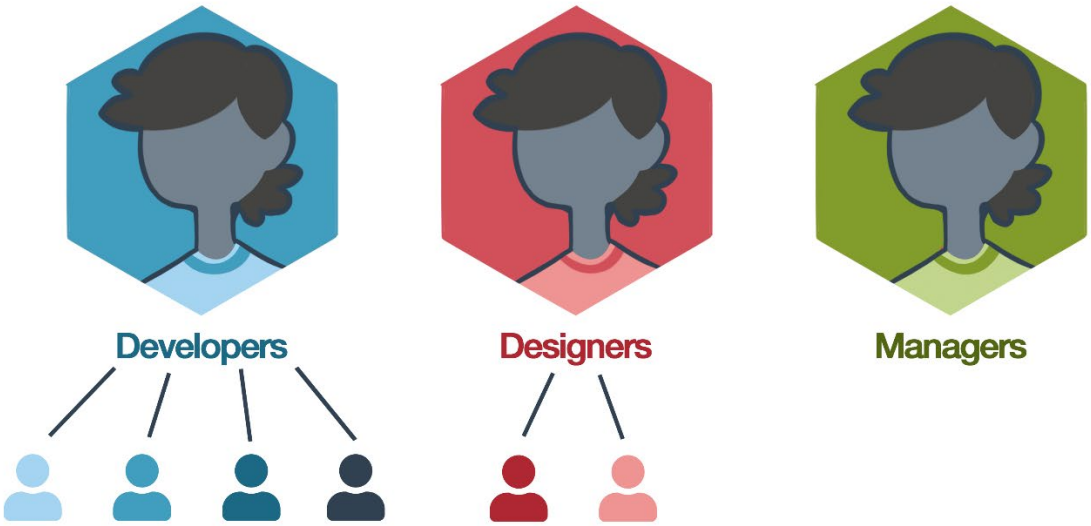


Figure 15. Illustration of the three main user groups and their subgroups.

The way that user profiles were used was to gather and present the most influential insights and needs belonging to the users and group them into profiles that represented the different users. In the profiles there is a mix concerning what they do in their job and why. But there is also information about which motivations and goals the different users have, in relation to what *"komponentkartan"* currently is. The needs of the users get presented as guidelines in the respective profiles. Further on the demonstrative website will be referred to as the pattern library since it is what the demonstrative website strives to be in a design system.

> *"Everyone uses a design system, even if they don't interact with it directly."*
> *This, above everything, drove home just how relevant it is to include input from everyone.*
> (Schultz, 2019)

15.1 DEVELOPERS

The developers are users of the system when they use it to develop their products with the help of it. The different types of developers that currently acts within the design systems are called primary-, secondary- and consuming developers, but also code testers are included here.

15.1.1 PRIMARY DEVELOPER OF THE DESIGN SYSTEM

The primary developer of the current design system are mainly front-end developers in collaboration with one or more UX-designers. While developing their assigned product, these users also develop the pattern library and its belonging code repository for the design system. Often the components in their own product end up as a general version of a component in the pattern library. The developers have built the components as well as the pattern library with its documentation and examples showing the components in action. This user group is both the provider of the design system and user of the design system, making them both developers and consumers of the design system.

Documentation and Examples

The primary developer user considers the pattern library to be lacking in documentation and the user is not sure whom it is supposed to appeal to as they are not dependent on the documentation themselves. As they are the main developers of the design system, they are the ones who need to develop it, but they need support in what to write and whom the documentation and examples are meant for. Therefore, the design system should:

- Provide documentation and example creating aid to this user. For the primary developer it should be known what they need to document and what to make examples of when creating new content. This way they can get help when creating new content, which in turn makes other users of the system understand the information better. This aid should help with answering the following questions for the primary developers:
    - What documentation and information should be displayed?
    - For whom is the documentation?
    - What least viable content is needed?
    - Where is this documentation accessed?
- Provide information about what is updated and not as the documentation cannot be old and outdated.
- Include documentation in the user's process.

Structure of the Design System

The primary users are well up to date with how the design system is structured and what content it has, but they realise themselves that it can appear messy for other users. As they put a lot of effort into making the code repository, they would like others to see what it is and what they can do with it, but to make that possible others need to understand the code. The pattern library's purpose is to show the components, how they work, and their possibilities, but if other users of the design system cannot make it work due to structure problems it does not encourage them to keep using the design system. Therefore, the design system should:

- Be referred to in correct ways and have descriptive names of the included parts of the design system.
- Have a good visual representation of the code in the design system as the code repository is not visible to all the users.
- Provide a defined structure that is to be followed when creating new content.
- Ensure the quality of the components. The components should therefore be tested, and the design system should ensure this.
- Provide a unison understanding of the versions and how to handle breaking changes as the users currently do not know how others work with versions of the design system.
- Enable testing without interfering with the representation of the design system.

Process

Generally, these primary developers have a very good knowledge of the development behind the design system since they work with developing it. However, they might not always need to use the documentation as they rather look straight into the component code or ask another team member how they have used it previously. The primary developer can assign themselves to tasks and prioritise what is to be done when developing the design system, however it is usually done through their own perspective of what needs to be done. Therefore, the design system should:

- Have a central part in the development work.
- Have governance that can decide the direction of the design system and ensure that the components will not be too application specific.
    - The governance should prioritise tasks and can distribute assignments to the developers.
    - The governance needs to be impartial to the products the design system creates, to ensure that its content is generic.
- Provide rules and a standardised way of developing and creating components to ensure a high quality throughout the design system.
- Be appealing to other teams to spread the usage of the design system.

Product Impact

The components are all approved according to the WCAG guidelines, which aids the users of the design system to have approved, qualitative components in their final products. To use and develop premade components can simplify the process, since you can reuse them and quickly build up new applications when necessary. This way, time and energy can be put on developing the product. However, since the primary developers creates components for the system based on their own application, they are both the user and developer of the components, creating a possibility for biased opinions. Therefore, the design system should;

- Have well-working components that aids in creating new applications.
- Consistently work with maintaining WCAG when creating new components.
- Have a way of governance that can take on an impartial view and help to not create biased components.

15.1.2 SECONDARY DEVELOPER OF THE DESIGN SYSTEM

The secondary developers are in teams with UX-designers and are developing a product with the design system, however they sometimes help with developing the design system on the side of primarily developing their own product. This contribution to the design system occurs much more rarely than for the primary developers. The secondary developer is similar to the primary developer in many ways, but the main difference is the ability they have to influence the design system. Currently there exists a collaboration between the secondary developer’s and the

primary developer's teams, but as the secondary developer's team are not supposed to work with developing the design system, their influence over it is limited.

Documentation and Examples

There have been some problems with starting up a new project and using the current pattern library as a foundation, this since it has lacked in information. The secondary developers have knowledge of the background to the system but prefer not to look on the code but rather the documentation. The documentation is therefore required to be up to date, since it can otherwise break the products created with the design system. Therefore, the design system should:

- Have well-working documentation that correctly represents the components.
- Provide enough information of how to start up a new project with the design system.
- Have correct and up to date documentation and examples to show what is possible with the components, otherwise the users will not be able to use the components efficiently.

Structure of the Design System

Similar to the primary developers, these users are using the code repository and with it the pattern library to build their application. However, they are not always able to build the components themselves due to them not working with developing the design system. A difference from the primary developers is that these users might have had insights into the code repository before but would rather use the demo site for documentation currently since they are out of the loop of the development and code of the design system. Therefore, the design system should:

- Be properly updated and have a correct visual representation corresponding to the code repository.
- Provide information on why and how a component is built in a certain way.
- Provide information about why a component looks and works like it does.
- Provide components of high quality.

Process

The UX-designer of the teams with the secondary developers represent their team at the design council. But since the current *"product owner"* role belongs to another team, these users' influence over what becomes prioritised in the design system is limited. If these users find bugs in the systems or faulty information, that may not become as prioritised as those issues reported by the primary developers. These users are somewhat informed of what is going to happen with the design system, since their UX-designer is a part of the council, so they can know what is planned. However, they do not know when those planned things will be performed or implemented. The pattern library is a tool for the secondary developers that they build their product with. Therefore, the design system should:

- Have a clear governance, to help prioritise issues from all the different users of the design system.

- Show what is planned, and when changes will be made in the system, aiding planning for the different users' products.
- Clearly show the front-end so that it easily can be imported and used in a project.

Product Impact

The secondary developer's product is an application that currently is similar to the primary developers. The primary- and secondary developers also have a similar way of working and have a similar collaboration with the UX-designers of the teams. Therefore, the design system should:

- Maintain the components and make sure they do not break and delay product launch.
- Correspond to the visual design the UX-designer makes based on the pattern library.

15.1.3 CONSUMING DEVELOPER

This user is a developer that works in a team that does not have any UX-designers and uses the pattern library and code repository, but they are not a part in the development of it. To these users, the design system is a help for them to develop new products in a more efficient manner. When they use it as a tool, they expect a certain level of quality. Since these developers does not have a designer on their team, this means that no one is responsible for the design in that team but is loosely combined into the developers' work. These developers are usually not divided into front-end or back-end developers, making them full-stack developers, often finding the front-end to be necessary but not as interesting. These developers have also been found out to mostly use an incorrect version of the pattern library. They have therefore been working with obsolete documentation to the code repository, making them not correspond correctly.

Documentation and Examples

The consuming developers needs high validity and correct information out from the design system, since they have lesser of an opportunity to get help from the primary- and secondary developers. These users need more guidance and help through the pattern library to ensure that they use the design system as well as they can with as few complications as possible. These developers depend more on the pattern library rather than the code repository, since they need more help. The help they need and get should include design advice, since the team does not have a designer. This way, by using the pattern library, they need good help with the design. It is not just about taking and adapting the components, but it is also about using it in a good way together with other components. Therefore, the design system should:

- Provide a way for the user to quickly adapt the design system and understand how to use it.
- Provide enough documentation and examples so that the user can understand the content properly.

- Have documentation in the pattern library that is correct and up to date and matching the code repository.
- Provide guidance concerning good design practice.

Structure of the Design System

For these users, a design system can help with setting up a design that is coherent throughout the application. Currently however, there are no set rules of how and when to use the components. These consuming developers are not very familiar with the structure of the design system. For example, this user is confused about what can be considered a bug or feature with a component when they are developing their own product. The users will not know and will not have an easy way of finding out, which they need to be able to do. This so that they can decide whether it is worth changing the component in their own application, or if they should submit a bug issue instead. If there is not a component in the system that fits the consuming developer's needs, they take it and modify it according to their needs. The users have had a hard time knowing what version to use and have used the online version, that does not get updated, instead of the most updated one. This since it is troublesome to keep up with the version and what is going to change in them. Therefore, the design system should:

- Support transparency and consistency in which version the design system is.
- Have rules and guidelines of how to use the components.
- Have descriptive, easy to understand, names for the different parts of the design system as the consuming developers are not familiar with any history of the design system.

Process

When using the system, these developers notice bugs or other faults in the components and report it in through issues at GitHub. However, they have no overview or comprehension of what is planned to be developed further and the only way they can contribute and have their opinions raised is through GitHub. Occasionally they can also contact a person in the teams with the primary- and secondary developers of the design system. The consuming developer users have no insight in how the process of the design system works and uses it as a third-party tool. The consuming developers are only familiar with the issues the design system has that gets reported on GitHub and are not able to see the whole backlog for the system. The consuming developers do not have the interest of learning the code repository thoroughly and see how the components are built. They just want to have a tool they can access for the front-end development. The issues reported to GitHub as well do not have clear instructions of when and how a reported issue will be investigated, when it will become accepted or rejected. The changes these users suggest or makes to a component can be accepted, or otherwise just be used locally for their own product. Therefore, the design system should:

- Help the user start using the design system as the user must find it easy to use the design system to proceed with using it.
- Provide information about what is planned to change or be fixed in the design system.
- Provide information of how feedback on reported problems gets handled.

- Be properly managed and updated so that the visual representation of the code on the demonstrative website and the code repository corresponds to each other.

Product Impact

The consuming developers is concerned about the customers satisfaction. They find it to be an advantage to use well-functioning, good-looking components to help develop their product. The consuming developers get help from the design system so that they can start with creating a new application quickly. When using the design system, they consider design to come *"for free"* with the premade components. These users have a need of things looking nice. However, they think this can be done by the components in the design system and common sense. The users of this group consider the design system to be useful developing and maintaining existing systems, but not fitting to larger new systems since they need to be re-built from the foundation. This concern comes from that the menus and headers are placed in the same way, which necessarily will not fit an entirely new product as it might not be a similar application to the other user group's applications. Therefore, the design system should:

- Support teams without a designer to make good design decisions.
- Have high quality components.
- Show that it can support parts of an application and not necessarily create the whole of it during new development.

15.1.4 CODE TESTERS

A code tester is someone that tests the code produced by the developers, but to a degree is a developer themselves. A code tester checks of if the produced code is living up to the demands set for the product. To be able to test things properly, a code tester needs examples of the components or site that it is trying out. The current design system does not have a backend, so the testing is said to not be so difficult, but the components do usually need to be put in a context to try the functionality of it. The testers are not directly a user of a design system but is still an actor working in and with the system. Therefore, the needs of the code tester need to be taken into consideration. Currently, all testing gets done on the pattern library. Components are also unofficially tested in the final products that the testers try out. If a fault gets detected it becomes reported to the team working with the design system, and therefore bug-reports gets placed as an issue on the GitHub page for the system. To help during the tests of the component on the website, there are pages of examples built to help and check off working components. Currently in the system, there are a lot of pages doing this, but not really helping anyone else than the tester or maybe a designer seeking inspiration. With this in mind, the design system should:

- Have tested and verified components to ensure quality.
- Support testing by a set procedure and environment.

15.2 DESIGNERS

The designers are also users of a design system, since it helps them to both understand and communicate with the developers. Two different designer groups were identified and found during the user profiling; those involved with the design system and those that are not.

15.2.1 UX-DESIGNER INVOLVED IN THE DESIGN SYSTEM

The UX-designer is responsible for designing the interaction and the flows for the product that a team works on, having the end user in mind. They decide what the product should work and look like to create the desired user experience. This kind of user of the design system currently have the main responsibility of the visual pattern library. They do currently also work as an unofficial product owner in a design council with other UX-designers. These users collect the visual components in a Sketch-file, which corresponds to the interactive examples of the components in the pattern library. This to aid them with using and placing the components in wireframes when developing their products. This shows that the UX-designers work closely with the pattern library to ease the communication with the developers, using the components so that the developers easily can identify components from the pattern library. These users take inspiration and help from a design system, trying to understand which components work well together and why. However, this information appears is flawed for some components. Currently, there is no information about user tests of the components, and it is difficult for the designers to understand what has worked with a component or not.

Since the UX-designers works with the pattern library as a foundation, it requires that the pattern library website stays well updated towards the code repository. With all of this in mind, the design system should:

- Ensure updated components, matching the code repository with the pattern library.
- Improve collaboration and productivity through easing communication and mutual understanding in the team.
- Provide enough information of why components are created, designed and assembled the way they are.
- Providing information concerning user tests of the components.

15.2.2 DESIGNER CONSUMING THE DESIGN SYSTEM

This type of designers, that are not involved in the development of the design system and just consumes it, does not currently exists. However, they could possibly exist in the future if the design system evolves. This kind of designers would probably also be UX-designers working with developing products together in teams with developers. However, they would not be involved in the backlog like the UX-designers that actually are involved in the development are. These users would use the design system for inspiration, making sure that they use components that already exist before designing something new. To help them to use and understand the system correctly, the design system should:

- Provide and sell benefits with using the design system.
- Aid understanding of how components work, are built up and constructed.
- Explain why each component exists and for what purpose.
- Aid consuming designers with adapting the design system to their team by being easy to start with.

15.3 MANAGERS

There are managing positions higher up in the organisation that are potential users of the design system, since they decide whether it should be used as a tool for development and efficiency or not. Therefore, they need to have a way of getting insights of how the system is used and what it can do.

15.3.1 HIGHER MANAGER

A highly put manager can be a user of the design system. These managers are responsible for the whole IT department to deliver the applications to the customers. If these managers would know and understand how the teams work with the design system, they would understand how product development benefits from it. The managers expressed that they want to constantly make the business and organisation more effective, lowering costs. The thought of a design system is that it could help structure ways of working and the way things are done throughout the organisation. As a stakeholder in the system, they also have certain expectations on it. Therefore, the design system should:

- Follow laws and important standards, such as the WCAG.
- Have a well-arranged front, keeping information updated and visible.
- Be possible to be used by teams concerning different varieties of applications.
- Enable for teams to create productivity sooner and be visible to the user.
- Have clear governance to ensure that the system will endure.
- Provide rules concerning how it should be used to assure the quality of applications built with the design system.

15.3.2 TEAM MANAGER

The team manager is the person responsible for the work of a smaller team, ensuring that the team will not be distracted by external factors and complications. They are not directly in charge of code or the product but are there to ensure a deliverance of it and to work with managing distractions. However, they are often a big part of the team and are therefore also a stakeholder in the design system that could help the team. Considering this, the design system should:

- Enable collaboration between and within teams.
- Create unity within the team and provide enjoyable and meaningful work for the team members.
- Reduce friction when developing the product.

16 CONCLUSION DEFINE PHASE

In the define phase the term of *"design system"* and with it *"komponentkartan"* have been defined as well as the different users of the design system and their needs. Below are the main insights concluded before the next phase of the project is introduced.

Up to date system - All the users and stakeholders in the design system need a well-updated system that will help with quality, consistency and information flow. They all have the end goal of producing products of good quality with the least amount of friction as possible.

Governance - There is no responsible person or group designated for the design system currently. What has been called governance can have many meanings and it is not as much about control as it is to have collaboration and development under control. With so many users having an interest in an updated, well-functioning and qualitative system, there should be a way of governing the design system in an appropriate way that favours as many users as possible.

Consistent documentation - The documentation and examples also get expressed as inconsistent from the users, and it needs to become standardised what to write, how to write it, and in which way. The different users have different needs concerning the documentation and an important aspect to regard is how the documentation is created and maintained. It is not enough to find what the best documentation of the design system is. But it should also be more understanding of how documentation should be created and how it can be a natural part of the development and process of the design system.

Maintenance - The design system cannot just be decided, created and designed, but needs to be developed as time goes on. It needs to be living and could possibly help with communication between the users to bring the work forward. A thing very appreciated with the design system is that it ensures high quality of the components since it follows the guidelines in WCAG. However, the information could be improved by having information about whether a component has been tested or not, to show that it has been verified. Once again it comes back to the process and how different users are included in the process. It also concerns how the users of the design system interpret the ingoing parts of it and the communication around it, which leads to the need of it to be more open to the users of the system.

Accessibility - Another aspect shared amongst the user groups are to ensure high quality in the design system, which will be inherited in the products. For this particular design system, accessibility is a central part. The WCAG criteria needs to be met in every product produced with the help of the design system.

Version control - An aspect that mainly concerns developers are the version control. The different kinds of developers have different accesses to new versions, which causes dissonance

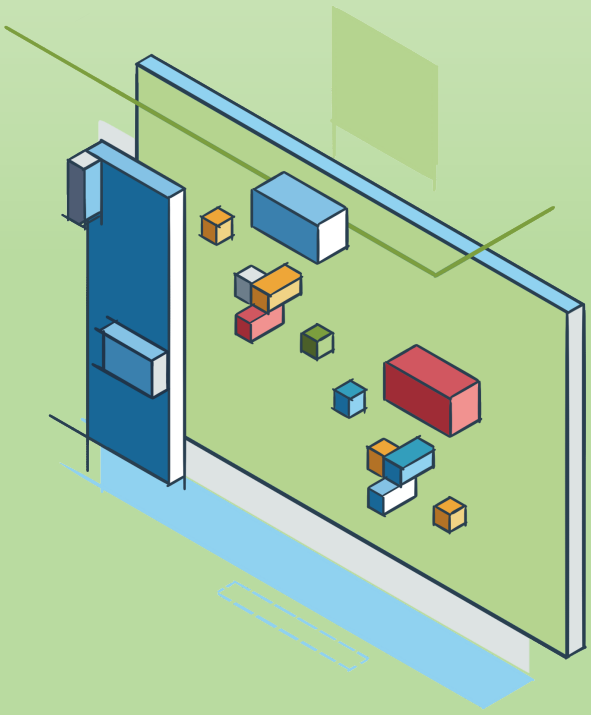
among the teams. Especially the consuming developers need to understand which version that they should use and how updated that version is. This since most of these users have been using a wrong version and read outdated information in the pattern library, since the version they used had not been synced with updates to the code repository. For the consuming developers to get out as much as possible from the system, version control is an important standpoint that should be more clearly communicated to all the users of the design system.

Inclusive - Throughout the interviews with the consuming developers, there have been an expression of that they feel left out the process. They do not want to get shut out from the process. They want to feel as if it is possible for them to contribute, showing that their contribution matters. It can be a help to communicate between teams, helping preparation of updates and what is planned to be implemented into the system.

Authority - An important insight from the define phase is that the *"komponentkartan"* needs to gain the weight it deserves. It needs respect to be able to be developed properly, and this understanding needs to come from the users classed as *"managers"*. Without the managers' understanding of the improvements it brings it will be hard to achieve the design systems progress as there will not be any changes made in the process.

Develop

This section concerns the third phase of the project; develop. In this phase the insights from the first two phases were made into concepts of how the final design system could work. These concepts were tested, evaluated and discussed together with various users between the different user groups. The concepts were iterated a few times for the final result to become as good as possible.



17 METHODOLOGY DEVELOP PHASE

For the develop phase the main goal was to produce concepts that were to be tested, evaluated and verified with the identified users. To do this, a workshop with users was carried out to help create ideas for a final solution from their perspective. There were also braindrawing carried out to create a wider range of ideas after the workshop. The concepts created were iterated in three turns, and how these iterations were carried out is also described.

This chapter describes the methods used for the phase of how to create concepts and iterate them, and the result of these methods can be found in the following chapters.

17.1 WORKSHOP

In the beginning of the develop phase, a workshop took place. The purpose of the workshop was to gather different users of the design system and initiate discussions around how the design system is supposed to work and what their respective views on it were, finding out if there was a common view among the users or not. It also served to discuss the different users' needs and find out if they could find ideas and solutions to fit their respective roles. The workshop also had a wrapping up purpose to take these ideas and find ways of how they could be implemented in their future design system process.

The process of the workshop was divided into two parts: *"Structure & content"* and *"Process & feedback"*. The workshop was loosely based on the process explained in Wikberg Nilsson et al. (2015) and took into consideration how to plan and introduce the workshop. The workshop was held for two hours and the facilitators took part in the workshop. The facilitators kept track of the time, introduced and changed the subjects, whilst ensuring that every participant got to speak. The facilitators also answered questions as they were brought up.

The sample size of the workshop was five people. The criteria of the sample were people working within VGR IT and belonging to the user groups previously mentioned in chapter 15, with as large spread of the different user groups and their subgroups as possible. Most of the participants were developers, and these belonged both to the secondary- and consuming developer groups. The other participants was one designer involved in the development of the current system and one team manager. The goal of the sample was to get as large spread as possible, but not more than six participants, as every participant would get their room to speak.

In the beginning of the workshop the participants were given a brief explanation of what a design system is, see the definition mentioned in chapter 12, but also what the goal of the workshop was and the expected result. In the end the workshop was wrapped up with a conclusion and what the material was going to be used for in the proceedings of the project.

In the first part of the workshop, *"Structure & content*", the participants were asked to work together to form their view of what should be prominent in the design system structure and documentation. With the help of cut outs of different shapes, colour and sizes, representing elements of documentation, the participants would form their own pages. They were to do this whilst explaining why the different pieces were structured in the different ways. The red pieces symbolised design information, the blue represented code information, the yellow was

introduction information, the green was examples, and the grey could be names or other things that had to be specified. See figure 16 below for a picture over this material. The pages the participants were to make was an introduction page, a component page and a pattern page for the design system. The cut-out pieces were to facilitate for creativity and lower the effort of creating a layout for the pages, allowing the participants to discuss and try out to puzzle together different pieces. They were also allowed to draw on the pages. For the participants to have a familiar thing to relate to, the demonstrative website *"komponentkartan demo"* was printed and given to the participants as reference.

Figure 16. Picture over the material for the workshop, with the different colour coding for design, code, examples, and so on.

The second part of the workshop was called "*Process & feedback"*. The task the participants were to work with was how the result from part one, *"Structure & content"*, could be realised. They were asked what steps would be necessary to make and how it would be possible to implement. The participants were to do this whilst also discussing whose responsibility it would be to realise these things. Whilst discussing these questions, the participants were asked to draw a line and pinpoint these steps on a whiteboard. The starting point was today, which was represented with a picture from the current pattern library "*komponentkartan demo"*. At the final vision point were the layouts for the various design system pages created during the workshop. The visual outcome and result from the workshop were not considered as important as the discussion around it, which as well was communicated to the participants.

The entirely of the workshop took two and a half hours, with more dedicated time for the first part, rather than the second.

17.2 BRAINDRAWING

The aim of the braindrawing session was to create a larger set of ideas of different solutions to a set of pre-defined areas, focusing on quantity over quality as a large solution space would be preferred for this part of the development phase. The authors of this report participated in the braindrawing and did a modified version of the brainwriting 6-3-5 method found in Wikberg Nilsson et al. (2015), making it more similar to braindrawing. Each of the chosen areas were

ideated around for three minutes and were performed separately by two people, who after the three minutes switched papers. The switch of papers would allow the next person to build further on the first person's ideas. It was also possible to start drawing new ideas with, or without, inspiration from the other person's ideas. This was done in two turns, so each person had each of the papers twice. After the first of the area's ideation was completed, the theme and ideas were explained and discussed amongst the participants. After that, the next area was ideated around in the same way. The areas that were ideated around were:

Start page - This area regarded how to catch the interest of a new user of the system, and how to help guide experienced users to where they want to go in the system.

Component and pattern page - This area concerned how information was to be presented about the components and patterns in the design system, but also what information or features could be included. The concepts would give examples of what this content could be and how it could be prioritised and structured.

Feedback and community - This area concerned how feedback could be introduced in the design system and in what ways it could be done. The feedback could also be connected to a community, a place where people could work together to improve the design system. This area aimed to discover what possibilities there were regarding feedback and community, and also how to connect it with the users of the design system.

After the ideas were produced for the different areas, the best ideas were marked out and described with their relative benefits and disadvantages, to create a clearer view of what each idea meant. Thereafter, different features were paired together to create different concepts. These were then made into wireframes that were taken into evaluation with users.

17.3 FIRST ITERATION & EVALUATION WITH USERS

The first concepts were generated from the workshop and braindrawing sessions. These were evaluated together with users with the purpose to find out their opinions concerning the different produced concepts and their parts. The evaluation would be the first part of the iteration of ideas concerning the design system. In this evaluation concepts of the first page of the design system and a component side was showed to the users. The fidelity of the presented material was hand-drawn wireframes, as the focus was to evaluate the content rather than how things looked.

The sample size of the evaluation were three participants that were users from different user groups. The user groups were primary developer, secondary developer and a designer involved with the development of the current pattern library and code repository. The sample size was rather small as the concepts were to be iterated further in a shorter amount of time rather than placing a score on each concept.

The evaluation took the form of semi-structured interviews similarly described to what is says in Wikberg Nilsson et al. (2015). Some of the questions were pre-planned, but a lot of room was left for discussion that would come up during the evaluation process. The interviews also were transcribed during the interview process. During the interviews the users got to look through the wireframes and mark directly on them with green and red markers what they liked

and not with each concept. It was also possible for the participants to write comments with a pen or make suggestions of how the concepts could be altered. The order the concepts were presented in was the same for all users.

After the tests the most appreciated features and things with the concepts were identified and analysed to take into the next iteration. The analysis was based on what the users had said and how they had marked and commented on the different wireframes for the concepts.

17.4 SECOND ITERATION & EVALUATION WITH USERS

After the first iteration the features most appreciated in that iteration's concepts were iterated further and put into two new concepts. These concepts had two different focus areas. One focused on what was feasible with today's build of the pattern library. The other concept disregarded what is possible today with the pattern library. It focused on being a concept that could be adapted in the future that needed to make larger changes to the build of today's pattern library and process. The concept less similar to what the demonstrative website looks like currently was evaluated first by the users, to go away from the users comparing or finding similarities with what the site looks like currently.

Similarly to the first iteration, the evaluations took the form of semi-structured interviews (Wikberg Nilsson et al., 2015). Some of the questions were pre-planned, but a lot of room was left for discussion that would come up during the evaluation process. Four users were interviewed for the evaluation, with one user from each of the user groups; primary-, secondary- and consuming developer as well as a designer involved in the development of the system.

The evaluation was done via a computer were wireframes of the two concepts had been created digitally. These wireframes were also done into working prototypes. This created the possibility for the users to interact with the concepts and *"click around"* to get a feel of how the concepts would work. The interviews were screen recorded, so that both the voice of the participant was heard, as well as showing how the user navigated and moved around on the screen. It created a better connection between what the users said and what they did by interacting with the concept prototypes.

After the evaluations an analysis was done of the result, listening to the feedback from the different users. From this feedback, it was possible to find trends of what the users preferred the most within the two different concepts.

17.5 THIRD ITERATION & EVALUATION WITH USERS

The third evaluation with the users was done after creating a concept based on the user feedback on the concepts for the second evaluation. The concept presented was done to high fidelity wireframes, looking similar to what it should look like in the end, based on what the demonstrative website currently looks like.

As the other iterations, this iteration had evaluations that took the form of semi-structured interviews (Wikberg Nilsson et al., 2015). Some of the questions were pre-planned, but a lot of room was left for discussion that would come up during the evaluation process. Three users were evaluated for this concept, where one was a secondary developer, one was a designer and one was a representative from the organisation.

The evaluation was done like the second evaluation, on a computer, where screen recording was used to get both recordings of what the users said, but also how they interacted with the concept prototype.

The analysis of this iteration was based on the feedback from the users. The feedback and what the users thought about this final concept were altered for the final concept. Therefore, the last concept and output of the project is based on this concept's look and feel.

18 WORKSHOP

The workshop was split into two main parts. The first part, titled *"Structure & content"* covered how the pattern library would be supposed to be built and what purpose it had. The exercise had a base in how the pattern library looked before, but the goal was to create a layout for how it could look. The second part had the title of *"Process & feedback"* and the exercise was to discuss a roadmap to change the existing solution of *"komponentkartan"* into a new design system.

18.1 OUTCOME

Below the outcome of the workshop will be presented. The results are presented in the order they were discussed in the workshop.

18.1.1 STRUCTURE & CONTENT

During the first part of the workshop, *"Structure & content"*, the participants discussed different pages for a future design system, shaped after what content they would like to see there. They discussed a lot and did thereafter lay out the different pieces of papers according to what they discussed. When the participants discussed, they realised that they had different views on things and could bend their wishes to better fit with what the others wanted quite well.

Start Page

For the first page they created, the start page, the different users agreed on a lot of the features they had placed there. In the end, the users wanted a clear pitch for the start page, a so called *"elevator pitch*", selling in the system to a first-time user of the system. Then they would like that different roles in an easy way should understand what benefits they can get from the system. The roles that were noted on the paper by the participants were *"for a manager"*, *"for*

a designer", "for a developer" and *"for a system architect"*, which would each have their own explanation of why those roles should use the design system. The outcome for what a start page could look like can be seen in figure 17 below.

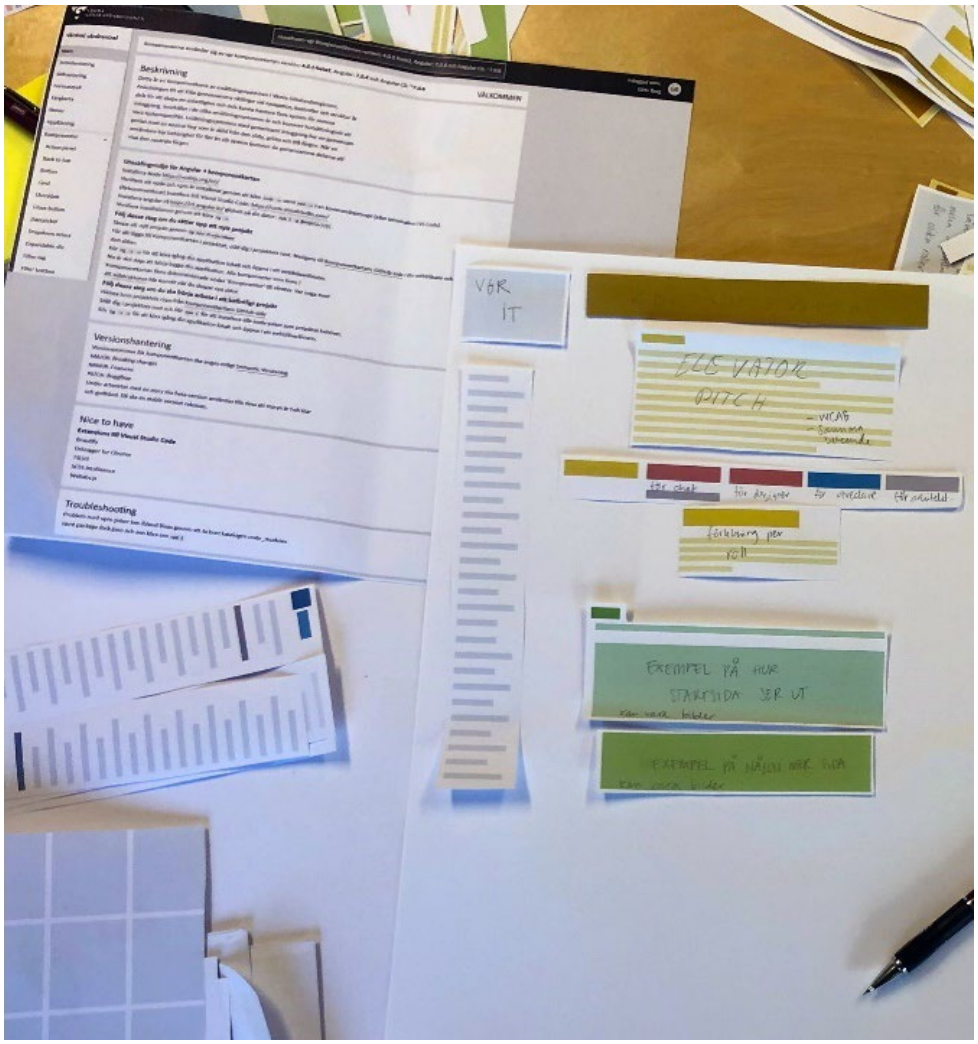


Figure 17. A picture over the outcome from the workshop and the creation of the start page.

The purpose for the start pages the participant agreed on was going to be an overview of what the design system includes and how it can be beneficial to use. A related idea was to have the start page as a *"tease"* of the design system and links to make the viewer able to read more, which could be pictures or separate pages.

A question discussed was that the pattern library does not have to be built with the components from the code repository. To not rely solemnly on the code repository would open up for possibilities, but there would also be a significant change in the process of how to document.

In the workshop the branding of VGR IT was discussed and the relation to the design system. Although the teams working with the remuneration system within health care are the main users the brand would be positively affected with connecting these applications and collecting them under one brand. For this to happen VGR IT, the design system needs to be recognised as one of their products and the value for having it should be understood.

It also was discussed and brought up how important it is to promote that the components and the content of the design system are made with WCAG in mind. As VGR is a public organisation it is very important that they are leading when it comes to accessibility.

Component Page

For the second page, a component page, the participants discussed its structure quite a lot, this more than putting out the cut-outs on the paper. They discussed a lot what they would want to see coming from their specific roles. The developers expressed that they do not mind seeing information if it stays relevant for them too. Information that was more role-specific could be hidden and opened on demand and the information had to be split in a way that made sense and was clear about who the information was meant for. Below in figure 18 it is possible to see what this looked like.

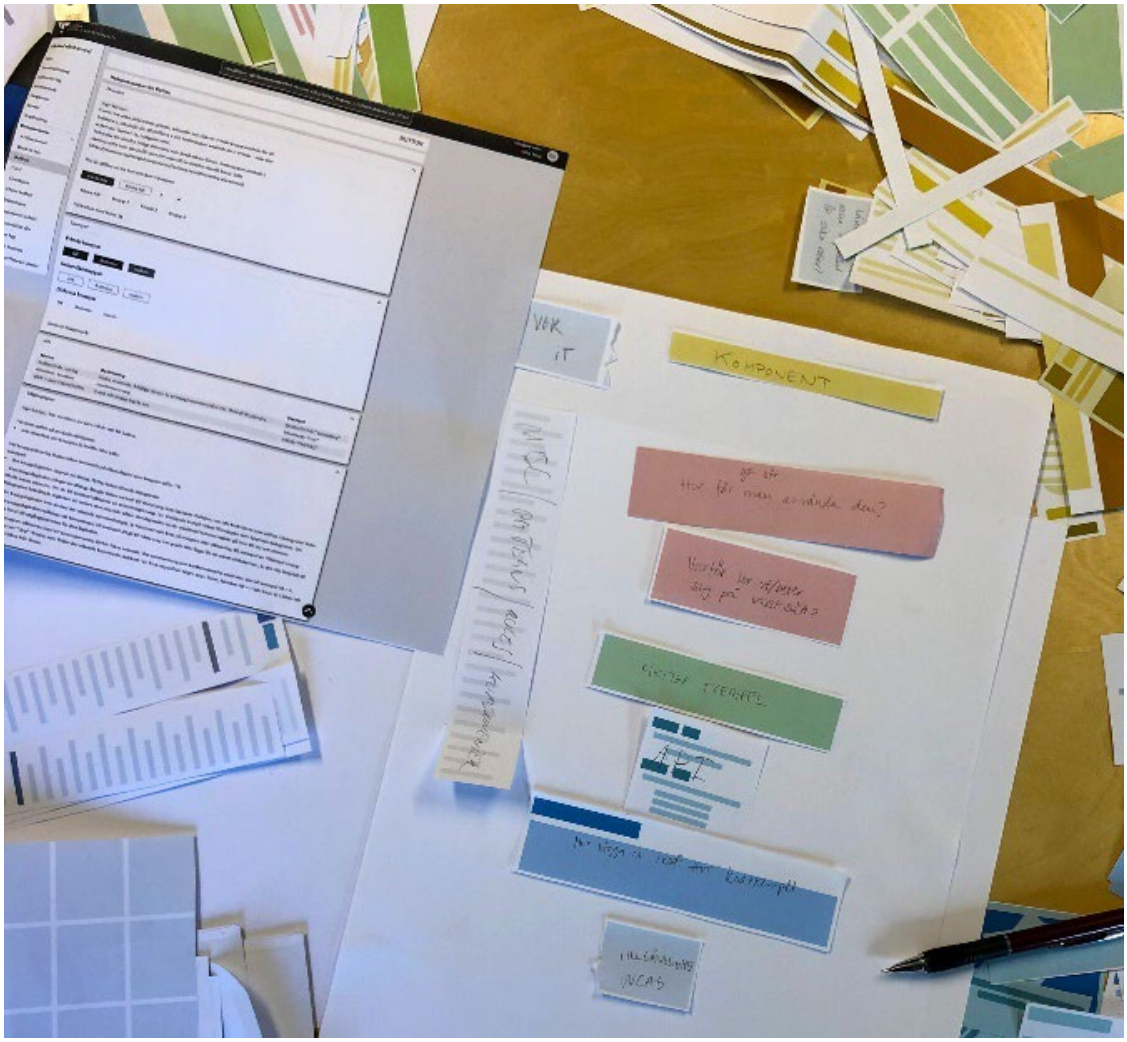


Figure 18. A picture over the put-together component page done by the participants of the workshop.

The designer wanted to see why a component was made in a certain way, clearing up difficulties with understanding why a component is made and design decisions behind it. The participants also wanted to know what is allowed and not to do with a specific component and why it should behave in a certain way. All the participants expressed that they would like to see the possibilities and abilities of the component. For the developers there needs to be a description of the API so that they can understand the flexibilities in the code, but also a code

example to describe to them how they would put the component in their code. All the participants agreed that an interactive example would be better than pictures of the component. This since pictures are not optimal and risks to be obsolete when the component's code changes.

The developers expressed that they are not interested to read up on WCAG, and they considered it to be self-explanatory that the components had been made with WCAG in mind. The only time it would be useful was if there would occur a case where they would need to consider it whilst implementing the component. It was also brought up that it would be useful to know in what ways it followed WCAG, but there would be a limit to what is useful.

Pattern Page

When the participants reached the pattern exercise there was a long discussion about what could be considered a pattern and if there would be different patterns for designers versus developers, or if patterns were for designers and components for developers. In the end of the exercise they had not reached a consensus of what a pattern page layout would look like, but the discussion had touched relevant subjects and content for introducing patterns. Important aspects that were discussed was to not have patterns scare away developers to use them and finding the right level of what a pattern should be and what level of uniqueness a pattern should have. The participants needed a clear definition of what a pattern was and what was different from a component and template.

Patterns were generally hard for the participants to make sense of as they could be considered components. For the developers it was not unusual for components to make up other components, so the definition of a pattern being components put together did not make much sense to them. It also was discussed that patterns possibly could become complex if it consists of other components. Therefore, a code example could get long, and would be difficult to describe with for example an API description. The designer had a bit of a different view of this and argued that patterns have a different purpose than components. The designer still thought that it still was good to have documentation about patterns, and that it was necessary to have.

Generally, the participants decided on a similar layout like the one in the component exercise, see figure 18, but they did not have time to build it.

All Pages Together

The pages together had similarities to what the demonstrative website *"komponentkartan demo"* looked like currently. Some developers wanted to see the API description to know what they would be able to do with it while others would like the examples to be visible before the code. The current menu system was also mentioned and the inconsistency with the menu items and that some of them could be grouped together to make more sense.

Another subject that was brought up was the order and grouping of the design system. It was discussed what would be included in the pattern library and what guidelines, or other parts, of the design system that should be kept outside of it. An example regarded code guidelines, if they only should be written on GitHub or if it should be put in the pattern library with a risk of being to redundant. A subject touching this was also how the organisation and process should work for contributing to the system. The UX-designer felt like they were put aside and could

contribute more to the documentation if they would be able to edit the text themselves. Today, the developers are the only ones qualified to make changes to the text on the pattern library. In the design system it would also be necessary to look over the names of the components to make sure they keep to a standard.

18.1.2 PROCESS & FEEDBACK

The first subject that was brought up in this part of the workshops exercise was on what server the design system was supposed to be hosted. During the time of the workshop the design system had no owner and it made decisions difficult. The system would in the future need a team that could work together on problems, but there was no consensus amongst the participants of how this team could be formed. A suggestion was that the responsible team was switched around and the responsible team had two weeks of maintenance each. An issue related to this was where the design system should get the finical aids that it needs from, as it does not currently have any funding or resources. The picture produced during this discussion, can be seen in figure 19 below.

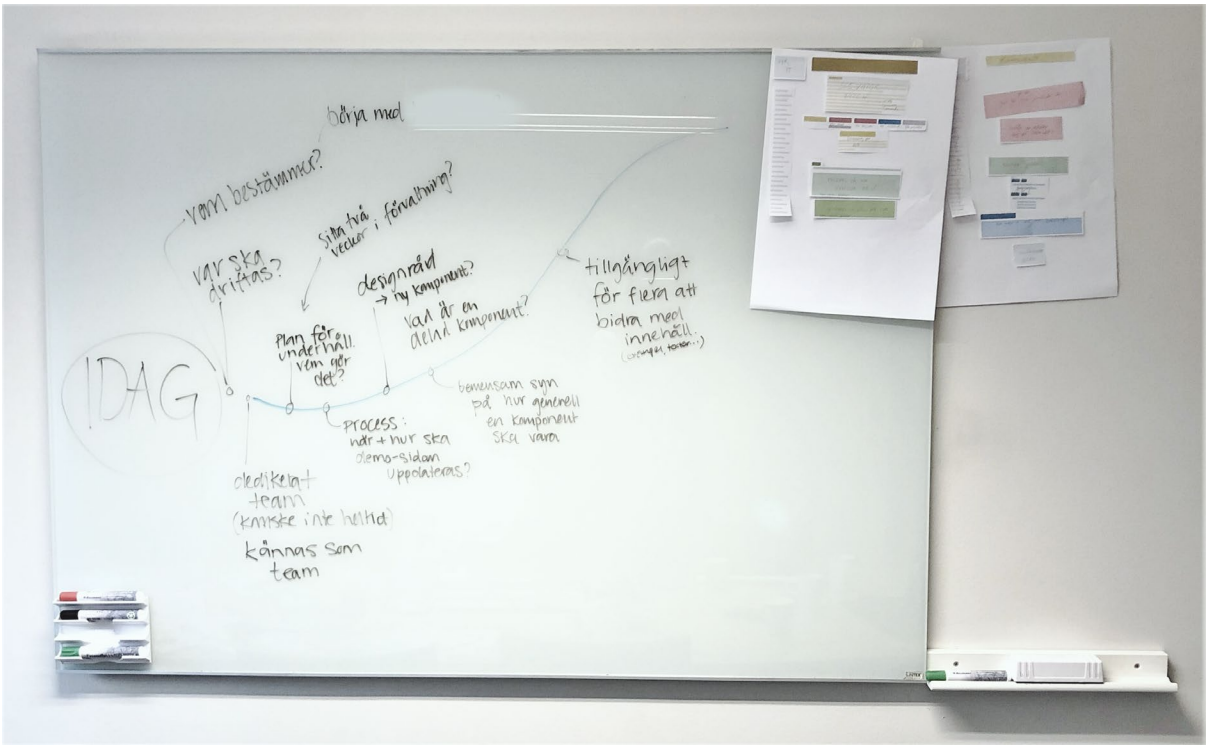


Figure 19. A picture over the timeline with actions produced with the workshop participants.

When developing new components there was also a discussion about a need for someone, or a team, responsible for deciding how that component should be created. The discussion was that that person, or team, should be careful when judging the uniqueness and generality of a component, having guidelines of how to do that.

18.2 DISCUSSION

During the workshop, the participants were presented with existing examples from *“komponentkartan demo”* with the ulterior motive that it would ease their process with grasping the abstract assignments provided during the workshop. This so that they could have a reference point to compare to during discussion, creating a basis. Even though it did help their discussion, it can also have created a type of bias in the participants, since the way they put the different parts together were similar with what the structure looks like today on the demonstrative website. They did not think outside of their existing box that much but kept re-arranging the existing system. However, when they arranged the different parts there were good discussions, and a lot of valid points were said that was considered for the future work.

The pictures and outcomes that the participants of the workshop produced was not directly considered to be complete concepts. What was created during the workshop should in other words not be a standalone concept. The most important thing to consider from the creation, is the discussion of what the participants found to be the most important things to consider and what is missing in the current code repository *"komponentkartan”* and the demonstrative website *“komponentkartan demo”.*

It was preferable to have a team manager in the form of a Scrum master participating in the workshop. This provided good opinions coming not just from those working in or with the system, but from someone that would like more insight into it without working with it as a tool. This gave good input in how to sell the idea of a design system to someone not that knowledgeable in the system.

An insight gained from the workshop was that different users have different views on what an example is, and what it is supposed to be. There was a discussion revolving pictures versus interactive examples, and what benefits there was with either version. Pictures can be enough for someone not working with the system, that is just investigating what it is. However, for someone working with the system, they want interactive examples to better understand how different parts are to act and behave, for example getting a visual understanding of what happens when you click something.

There was also a discussion on what is role specific, and into which different roles that a design system could be divided into to try and ensure that everyone gets the insight and knowledge that they require from the system. According to the participants, examples of roles that could be interested could be manager, designer, developer and various other roles. The way of dividing roles and find appropriate areas was taken into further consideration during the brainstorming. However, going further it is important to think about not dividing information to much, since the design system is supposed to be the single source of truth, not hiding away useful, important information.

19 BRAINDRAWING

For the braindrawing three different areas were investigated; a start page, a pattern or component page, and also some type of feedback or community. For each area, a lot of ideas became produced, with varying degrees of realisation capabilities. The different ideas were discussed between the participants, and in the end, the favourites and the most feasible ideas for each area was taken into the next stage. These ideas were marked with dots on the papers to specify which ideas were good to work and consider further, see figure 20 below for what it looked like.

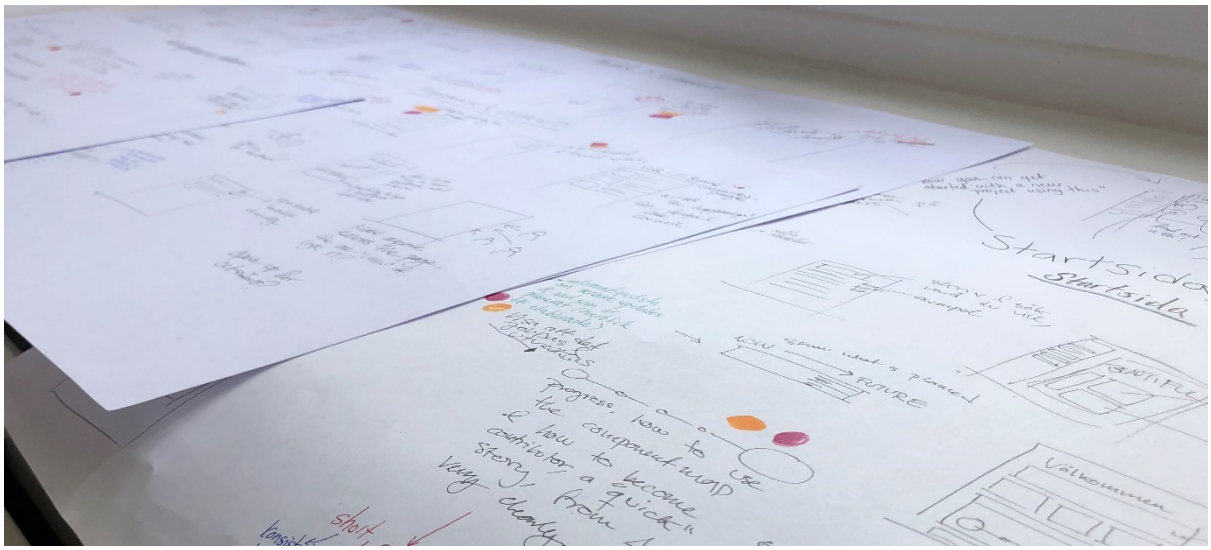


Figure 20. A picture over how the ideas were sketched down on papers for the certain areas for the braindrawing session. The different dots on the papers separates feasible ideas from the mass of ideas.

The marked-out ideas were taken into a document, and each idea was described and clarified. In the document, the respective strengths and weaknesses for each idea were specified, which was done to further understand what each idea stood for and implied.

After this, different ideas from the different areas were put together into concepts, with some concepts having either one or more ideas from each area. The concepts were based on what could work together, following certain themes, such as favouring different types of roles to the system and different end-goals. The way of pairing features together was done with post-it notes set out on printed out papers. In the end it became five different concepts. An overview over what the concept creation looked like can be seen in figure 21 below. These concepts were made into wireframes and are presented in this chapter.

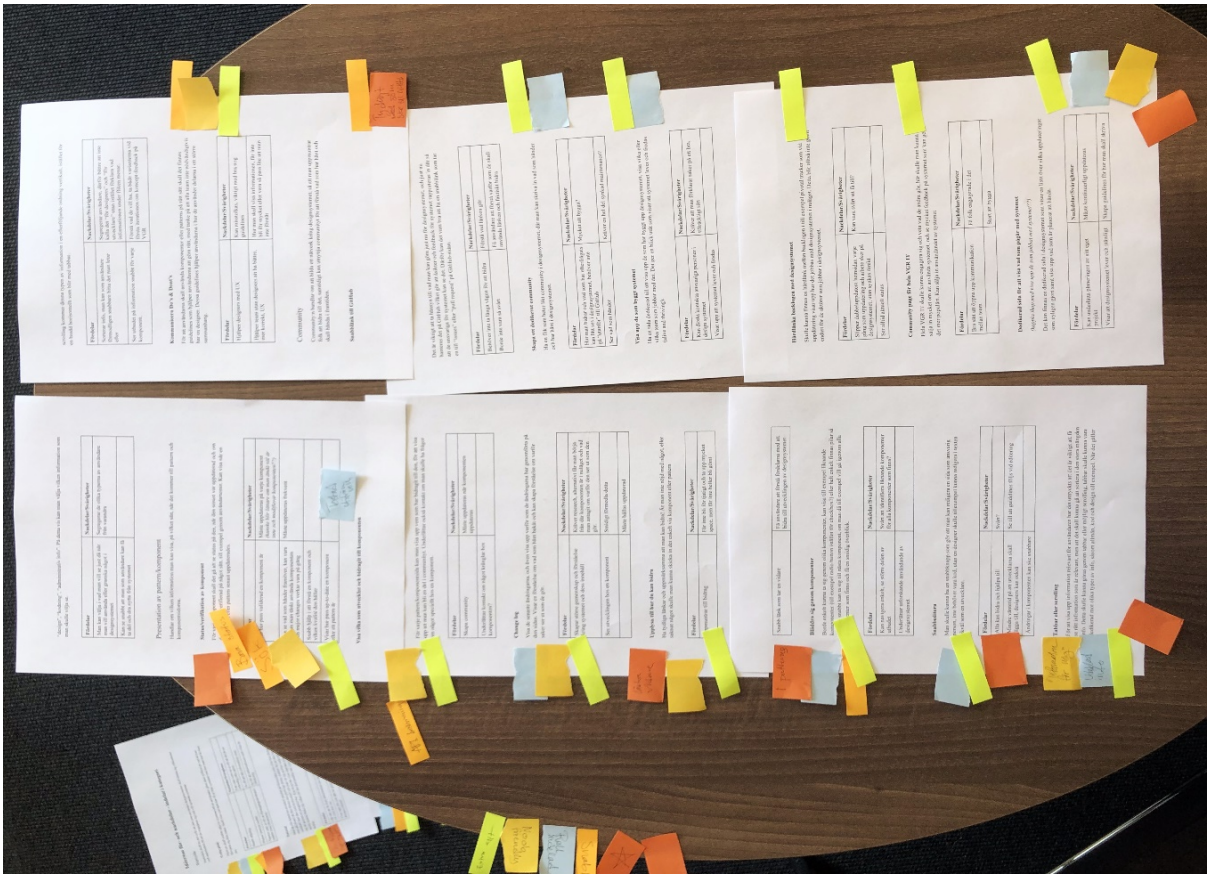


Figure 21. The papers printed out with the features for concepts of the first iteration, with the features getting paired together with different post-its.

19.1 CONCEPTS

From the workshop, something evident that was frequently and thoroughly discussed was that a user should be met by a pitch as soon as they enter the design system, to quickly understand what it is and get a sense of the benefit with it. Therefore, the pitch was adapted and used in different ways through each of the concepts. Otherwise, different kind of information was placed differently, and some concepts had features that some did not have. The different concepts are described below.

Concept 1 - Role Adapted

This concept focused on adaption towards different roles, so that the users identified could be able to quickly find information adapted and suited for them. The start page of this concept contained a clear pitch but shared the page with different type of links acting as shortcuts to specific pages in the system. The shortcuts were meant to serve new as experienced users. Role descriptions were also displayed early and was a dedicated area that shortly explained how the system could benefit different types of users.

The component page was supposed to show how the component had been tested, which was adapted to different roles and not just displayed generally. In the concept it was possible to edit the page, no matter what role you belonged to, so that any user could help with contributing to the development of the documentation. The information on the component page was sorted in

different categories, divided by role and possible interest in the documentation. To handle community and contributions, the system had a dedicated community, where different roles could go in and read and find information about what was planned and had happened recently in the system. To see what the wireframes looked like, see figure 22 below.

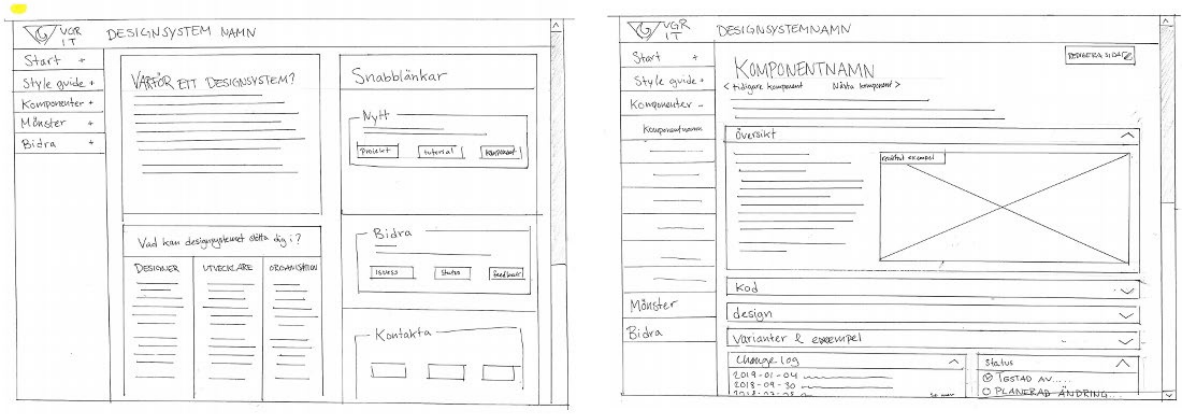


Figure 22. Wireframes created for the first concept, here shown what the start page and component page look like for the concept.

Concept 2 – Speedy

This concept was to aid a user to quickly get to their wanted information, showing correct information in a short and consist way. The start page contained a short pitch, followed by a search bar with shortcuts to get to different parts of the system. The links were to aid new- as established users.

On a component page the status of a component was displayed, showing if it was verified or not. This so that a user quickly could get an overview of how fresh and updated the component was. The documentation regarding the component did also contain information and illustrations of what can be done and not with a component. In the concept there was a quick way of navigating between components, so that a user quickly could get to the previous or next one. To see what the wireframes looked like, see figure 23 below.

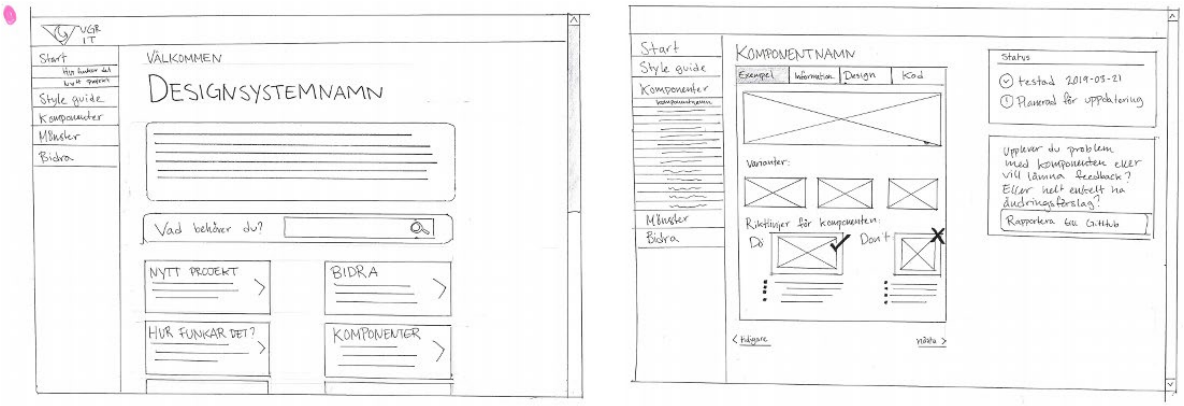


Figure 23. Wireframes created for the second concept, here shown what the start page and component page look like for the concept.

Instead of a dedicated community, this concept constantly encouraged users to contribute by having a shortcut to GitHub where the users could do just that.

Concept 3 - Logic Thinking

This concept started with a pitch and a bullet list structuring up the most valid points of why a design system is needed. The thought was to have a quick overview of what is possible to find in the system, described from the view of different users and their interests. The start page was more stripped than the others and did more clearly dive into the different roles.

The component page showed status and verification of the component early, to declare the quality of the component in a quick way. This box did also contain tags that for example quickly could help a user search in the issues gathered on GitHub. Next to the name of the component was the so called *"selector name"* of the component. Information was sorted by tabs, to create a quick way to provide a clear width of information. For this concept, the component page also showed which patterns the component possibly was a part of. The page was possible to be edited by different users of the system, to aid the development of the system. The concept was linked to GitHub, and to encourage contributions there was be a dedicated page informing users of what was happening in the system. To see what the wireframes looked like, see figure 24 below.

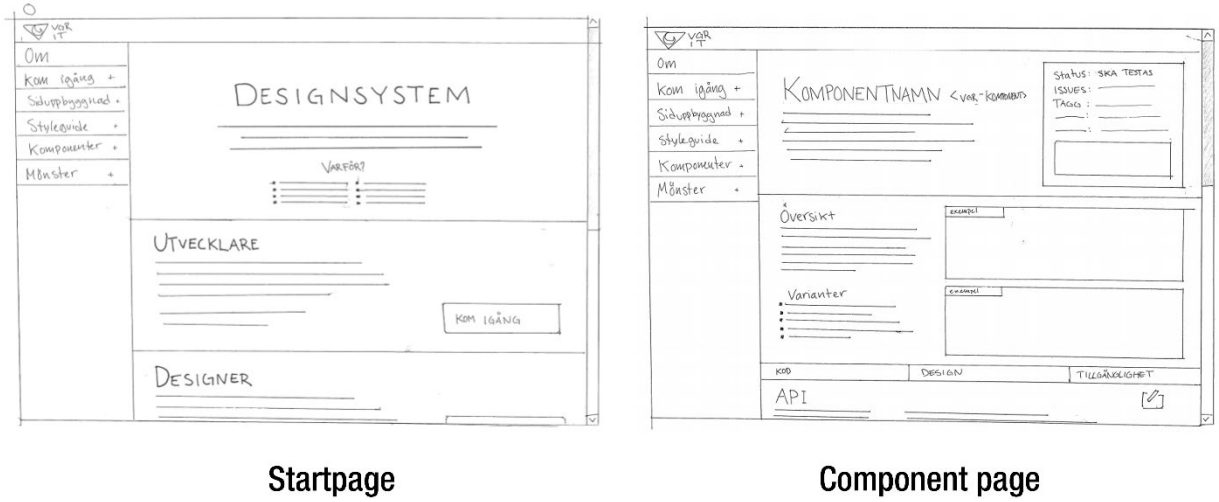


Figure 24. Wireframes created for the third concept, here shown what the start page and component page look like for the concept.

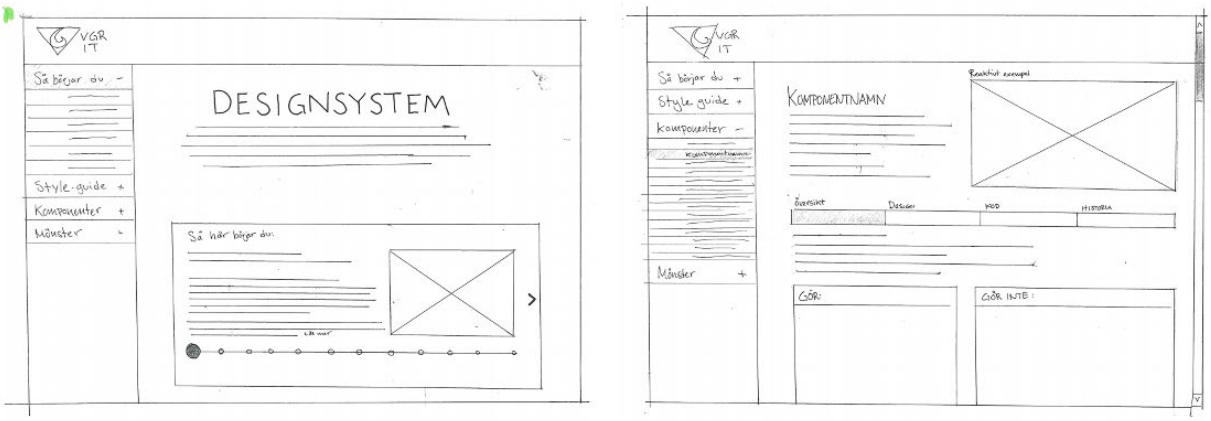
Concept 4 - Beginner Friendly

The fourth concept was meant to primarily help a new user of the system. Therefore, the start page did, after the initial pitch, have a tutorial that showed what the system consisted of and how it was to be used. This way, a new user could get a complete introduction to the system immediately.

This concept had a start page for all the different component, displaying previews in the form of small pictures which showed what the components looked like, to help with recognition of what component to find more information about.

A component page in the concept explained at the top what the component was, together with an interactive example. After that, information was sorted in tabs to show a width, so that the

user could chose which information to display. The tabs for the concept was *"Overview"*, *"Design"*, *"Code"* and *"History"*. The documentation was to be helpful for a first-time user, showing a change log for the specific component so that a new user could understand how it had been changed. In the concept, there were also encouragement of how to contribute if problems were found within the system site. To see what the wireframes looked like, see figure 25 below.



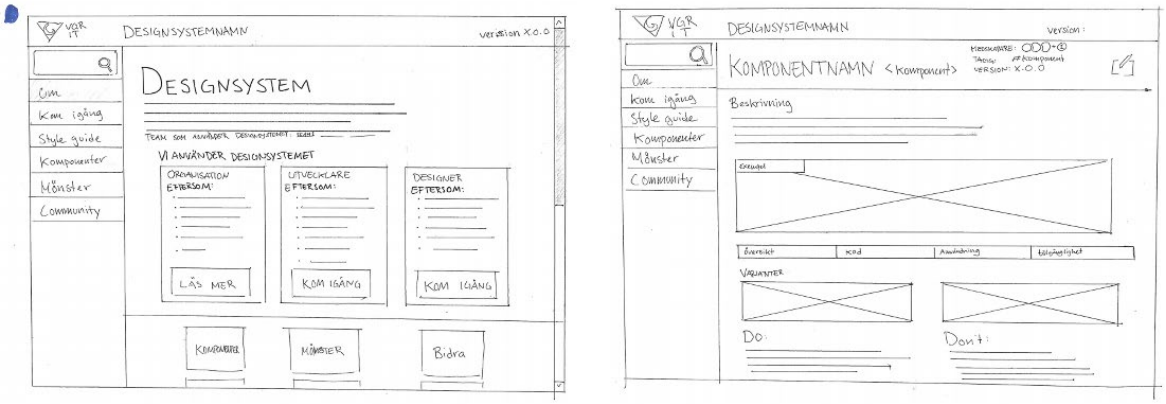
Startpage **Component page**

Figure 25. Wireframes created for the fourth concept, here shown what the start page and component page look like for the concept.

Concept 5 – Information-Packed

The last concept created during the first iteration was packed with a lot of information in different ways. On the start page, after the initial pitch, there were shortcuts to different teams that had used the system. This existed if a user would want more input from them about using the design system. Afterwards, different roles and information about the system benefits was presented. To also help a user to where the needed to go, there was information about the different sections in the design system, such as components, patterns and contribution.

A component page contained a lot of information as well. The component page showed status, verification, contributors to the component, and informed of what to do and not with the component. To aid in the development of a component, it was possible to edit the page as well. The structure of the information was in tabs with the labels *"Overview"*, *"Code"*, *"Usage"* and *"Accessibility"*. To see what the wireframes looked like, see figure 26 below.



Startpage **Component page**

Figure 26. Wireframes created for the fifth concept, here shown what the start page and component page look like for the concept.

For this concept, it also existed a dedicated community, showing who had created the system, alongside with a backlog of what had happened in the system. It also showed what was planned with the system, letting a user in on what was happening and had happened, to feel as if they belonged more into the community of the design system.

20 FIRST ITERATION & EVALUATION WITH USERS

The first iteration of concepts created during the braindrawing was evaluated with users to see the advantages and disadvantages with each of the concepts presented in section 19.1. It was also done to get feedback on what to discard and what to bring into the next iteration of ideas. This was done by letting the users go through hand drawn wireframes of the concepts, allowing them to express what they liked and not with the help of green and red markers. In figure 27 below it is possible to see what this looked like.

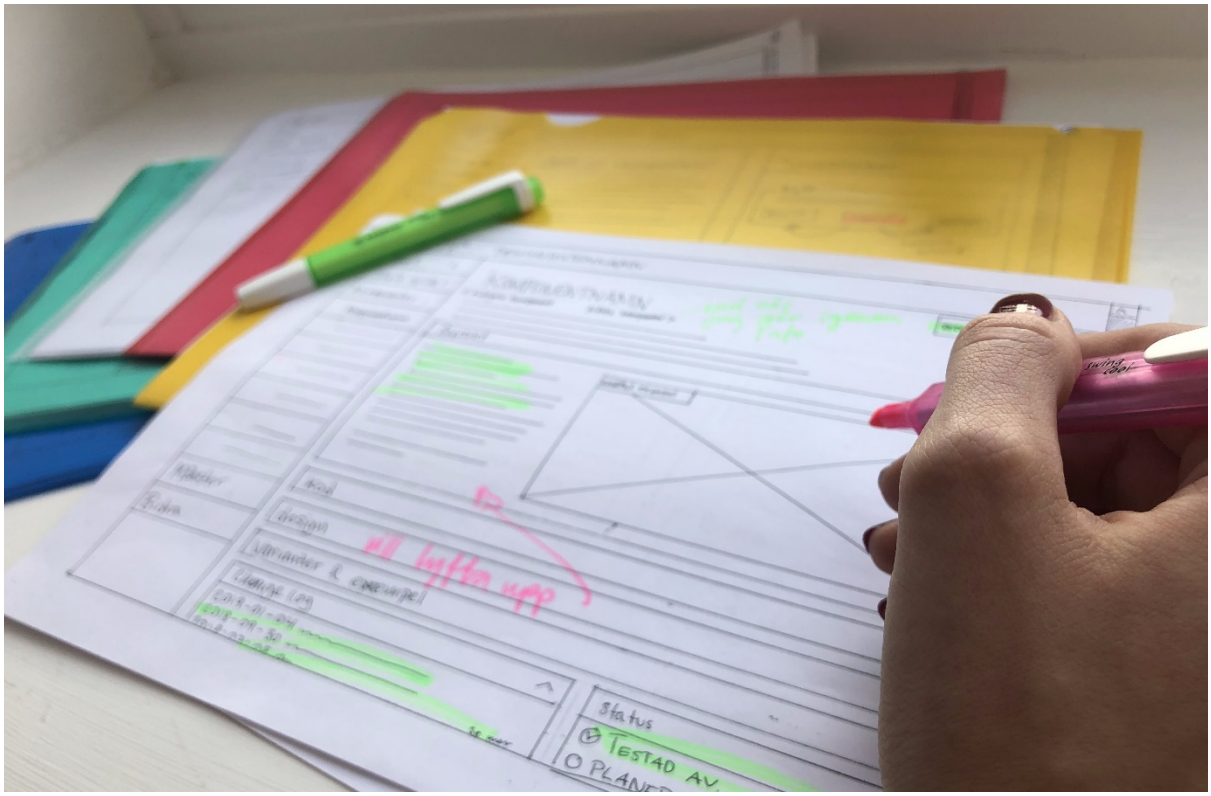


Figure 27. A picture of the first iterations evaluation, where the users got to mark on the paper with green and red markers what they liked and not with the concepts.

20.1 OUTCOME

Below the evaluation results will be presented. It will describe the main takeaways from the feedback provided by users.

Concept 1 – Role Adapted

For the first concept, there were different opinions about the quick access shortcuts. Some enjoyed it, but most thought it mainly was in the way, misguiding and potentially misleading users to where they should navigate.

There was confusion about what *"New"* meant for the users that were developers. One developer thought it could be more hidden and not as promoted, but the other developer thought it could be featured news rather than how to start new projects. Generally, all the users had different opinions of what *"New"* could be and the word might have confused them. One developer thought the design system introduction took too much space but thought the idea of having tips of what the design system could do for your user group was helpful. The designer liked the idea of the motivational introduction, but also the featuring of a link to quickly provide feedback and the information about who to contact. The users had different opinions of what the labels for the documentation should be like and their content, as well as placement. One of the developers and the designer thought overview and design should be first, even expressing that it possibly could be merged.

None of the users found the *"Previous - & next component"* links useful as they would not browse the design system in that order. One developer found the introduction test to be excessive when there was an overview text. The designer thought the documentation for the component felt hidden and lacked a clear overview. Also, it was unclear where they could find accessibility information. The designer thought the placement of having variants and examples separated from the main example was strange but liked the option of being able to edit the page. One of the developers was sceptical to having an interactive example within what they referred to as a *"list"*. It is difficult, sometimes impossible, to have interactive components within another component. All the users liked the idea of having the edit button on top of the page, but there are issues connected to how the pattern library is built today if that would become adapted.

The designer and the developers liked the change log at the bottom of the page, although the developers were a bit hesitant to how it could be up to date. The designer liked the idea of seeing the components status and if it had been tested, while one of the developers thought that everything put in the pattern library should already been tested before being added to the pattern library.

Concept 2 - Speedy

All the users liked the idea of being able to search for components and thought it fitted to be displayed on the first page since it could get them quick access to what they needed. However, there would be limitations with a search bar and the designer expressed these concerns, but the developers thought it would be possible to implement to an extent. Therefore, a search option could be possible, but how accurate it could be was unsure.

The quick-access boxes with links were generally considered a nice idea but the content needed to be thought through so it would not be redundant with the menu to the side. The content of the boxes would also need to be prioritised of what to put at the top. The introduction to the design system was of an appropriate size.

On the component page the developers thought that the do's and don'ts belonged to design and they were not interested in seeing them. The designer liked the idea but did not know what would be left to the design tab if the overview had the do's and don'ts. The designer also liked having the main example of the component with the variants below, however the developers did not think interactive components could fit within the small area that was drawn, only if the examples were shown as pictures and not as real, live components.

The developers also expressed concerns considering the large space the status and GitHub report boxes took up on the page. This since it took up space from the more important information by having the different types of information in columns. One developer liked to see that the component was planned for updates but did not like the idea of showing that it had been tested. This since they thought that it would be self-evident that the component had been tested before being put into the pattern library. One developer thought that the GitHub reporting box could be placed elsewhere, or be different, and not as visible.

Concept 3 – Logic Thinking

The developers thought the idea of having the *"Get started"* section for different roles was appealing but questioned what content it would lead to. One of the developers would also have liked to prioritise the designer section above the developer section and move the bullet points to the organisation section, as they thought it fitted better there. There was a bit of confusion of what the label *"Organisation*" meant, if it was the design system organisation or the VGR organisation.

The designer thought that the examples of the applications built with the design system was a heavy proof of its usefulness and selling points. One of the developers were also positive, while the other had no opinion.

On the component page the designer liked the idea of seeing the selector component name used in the code since they thought it could help them to understand the developers better and ease communication. One developer thought the opposite as they thought it would just get confusing of which name to use when talking about the component. The same developer liked to see the issues and have a tag for these issues related to the component. However, it would be a bad idea to highlight issues with the component directly and would rather have them in another place than at the top. All the users were confused with the variants, experiencing them to be in text which they did not like. One developer did not like the idea of having the examples so small, especially if it was to be interactive examples.

Generally, the tabs were liked, but the content in these tabs needs to be considered together with their labels. The developers thought the code tab would include code examples rather than API and have the API in a separate tab and rename the code tab to examples.

The edit button appeared to go unnoticed in this concept.

Concept 4 – Beginner Friendly

The start page was rejected as it was too focused on new users of the design system. They all would like a getting started page, but they did not think it would be a good idea to have it featured at the first page. One developer did not like the progress bar as it would not say what is going to be next and difficult to overview.

All the users were positive to the introduced component start page, which was an extra page for this concept. The developers thought it was a good idea with the start page for the components if the picture for each individual component describes the component instead of being an interactive example. The component start page was also considered as being a good help to retrieve and recognise components easier.

The introduced history tab was not disliked, but it was thought of to be too difficult to maintain comparing to the benefit it would have. The developers though the do's and don'ts belonged into the design and did not consider them to be of relevance for developers. They would rather see examples of variants in the overview tab. One developer did not like the label code and would rather see the label examples, which would include code examples. One developer liked the idea of getting an overview and be able to hide and display content with the tabs.

Concept 5 – Information-Packed

On the start page all the users liked the feature to be able to search. The designer liked the idea of displaying which teams that used the design system, but one developer expressed concerns of being held responsible if the name of the team was displayed on the first page. All the users liked to see the version of the design system, but the version needed context to be beneficial. Although the way the design system is versioned needs to be considered, as every update is regarded as an update of the whole pattern library. The possibility of versioning components was brought up. The designer liked to see the versioning to be able to understand the developers better.

One developer disliked having the *"Getting started"* as its own menu item and would rather see it below the *"About"*. One developer and the designer liked the testimonials from the different roles on the first page. As a designer, the designer thought that their role more would be interested in patterns rather than components, so they would like the pattern box to be placed first. One of the developers did not like the label *"Contribute"* on one of the boxes and would rather have it to be a word like community that could link a user to a community page.

On the component page the designer liked to see the contributors to the component, while one of the developers were not sure it would contribute anything or be interesting to see as it is possible to find that information on GitHub. Once again, the designer would like to be able to edit the pattern library and its pages. One developer and the designer also liked to see the tag related to the component to be able to discuss it and what tag to use when writing about it. The designer liked the label "*Usage"* instead of *"Design"* but was a bit hesitant to the placement and the mix up with the information from overview.

The users liked the idea of having a community page but was afraid it would be redundant with the pivotal tracker backlog. One developer expressed concerns considering how to update the community page even though they thought it could be a good idea. Generally, all the users liked having information about why there would be a community and what was to be followed as well as how to contribute. The designer would have liked the possibility to express the agenda surrounding the system about what was happening and what was to be discussed on the community page.

20.2 EVALUATION

Throughout the tests, it was found preferable from the users with all-person access to the editing of pages. This was taken into consideration for further iterations, and there was previously evidence of this in earlier interviews as well. This since the designers felt difficulties to currently make a small change in the system, and there should be a better way for developers and designers to work together with the development and maintenance of the system. Therefore, some type of editing solution, for more than just developers, should be created.

Versioning and how to work with mentioning versions needed to be further considered when going further. It needed to be considered whether versions should regard the whole content of the design system or if a small change in the text could get away with not being an official update of the version. In the future, maybe it does not have to be a whole new release of the system as soon as a minor update gets implemented.

The updating and synching of the system was mentioned as being considered complicated by the users. It was therefore important to think about the process around the design system and not just what it should look like and contain. There should be suggestions of how this practically could work.

One thing important to bring from this iteration, was that those working with the help of the current demonstrative website *"komponentkartan demo"* and the code repository thinks that the components are already verified and tested. It was complicated and confusing to the users that testing takes place within the system, and nothing currently says whether it is verified or not. Therefore, the idea of a separate demo site was taken into the next iteration, where the components could be tested by code testers before being put in as documentation. This ensures higher quality and more well-functioning components.

Things to Iterate Further

When going further to the next iteration there were some identified improvements that were to be considered.

One thing discovered, was that the structure of menus can be further investigated, looking into what should be in them, in what order, and then also figuring out the reason to why something should be a menu choice.

In further iterations, it was to be investigated more about the start page, considering the inputs from the first user evaluation to understand what can be hidden away. This to find a possible balance of how much you can either help or guide a beginner of the system.

Going further, it was to be investigated more about the content of the start page for the components and the patterns, to set down more guidelines of what information can be found there, and which information should be possible to find there.

The community-aspect did as well need more iterations to reach an understanding of how to create new things and feel a sense of belonging to the design system. Further it was considered how to mirror the development process to those interested, keeping them updated and in the loop.

Going forward, it was also considered how to edit the content of the design system, looking into different content management systems, and how editing technically could be solved for the design system.

When it comes to information the benefits of having information displayed in different ways was to be considered, such as tabs versus dropdowns. It was to be considered what information about updates and versions that were to be shown, and where they would be displayed on the different pages. These were to consider when a user would like to see that type of information as well.

In the evaluation, there were some split views concerning the information and documentation about WCAG. Therefore, the way of how to describe information about it were to be iterated.

21 SECOND ITERATION & EVALUATION WITH USERS

For the second iteration two different concepts were created with input from the evaluation of the concepts in the first iteration. The first concept disregarded what is current practice and introduced things that could be viable to implement but would take some effort to initially adapt. The second concept was based on what currently is possible and focused on improvements to the existing pattern library and demonstrative website *"komponentkartan demo"*. This to easier adapt the current practice. The two different concepts were created to understand how much effort there would be to change the current solution, and what compromises a user would want to make to get a better, more coherent system. After these wireframes were created, they were tested together with users to get their inputs. The fidelity of these concepts was higher than during the first iteration and contained more placeholder text of so called *"lorem ipsum"*-text. This text did in a clearer way describe just how much information there was meant to be, rather than just being lines as in the first iterations paper concepts.

21.1 OUTCOME

The outcome explains the two different concepts more in depth, explaining what they contained and how they were structured.

First Concept

In the first concept the possibility for non-developers to edit text on the pages was added. This feature was to be enabled through using GitHub to directly edit text files from there. Otherwise, the largest difference from the other concept was the layout. The menu of this concept was structured so that by pressing a menu heading, it would take you to a start page for that menu choice. The wireframes created for this concept were prototyped, so that you could get to another page by pressing a button for example. To see the depth and number of wireframes created for this concept, see figure 28 below.

Figure 28. The wireframes for the first concept of iteration two, showing the different levels of the concept.

The concept had a search bar to easier navigate and find wanted information in a quick way. The concept consisted of a start page, again with a pitch selling the design system, alongside quick links to guide the user to certain parts of the system and an example of an existing application using the design system. See figure 29 below for the wireframe of the start page.

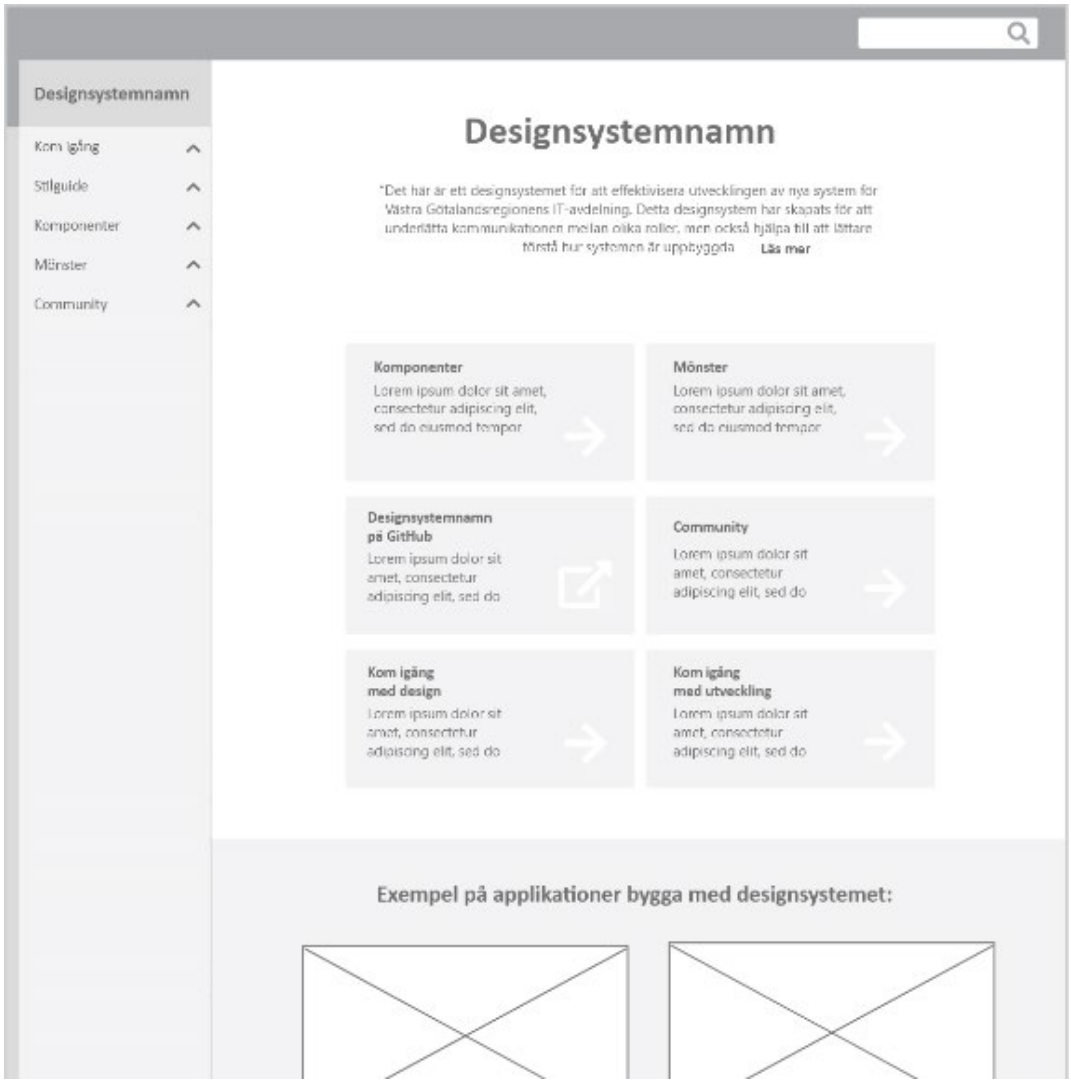


Figure 29. A picture over the wireframe for the start page for the first concept of the second iteration.

Then the concept contained information about how to get started, a style guide, components, patterns and type of community called just that; *"community"*. For all the menu choices there existed start pages, explaining what that section does and what differentiates each section from each other, such as components from patterns. The component start page is supposed to have small illustrations of each component, aiding recognition for a user looking for a specific component. However, each component should also be listed in the side menu so that it gets easy to navigate between components.

The getting started section was about helping someone to learn more about the design system, but also how to get started with either design or development. For development and design there were different stages to follow in a sequential order, meaning that both were linear onboarding processes.

The style guide collected all the different style guide parts from the previous demonstrative website, hiding away certain style guide related information that otherwise took up space in the side menu.

At the top of a component page there was information about the version of the specific component, its code selector name, and a short introduction to what the component was. Then

there was a tab system with the tabs of *"Example"*, *"Design"*, *"API"* and *"Accessibility"*. The tab example showed interactive examples, and its code, whilst the design section described guidelines of how to use the component, also describing the different variants of the component. The API displayed a short table of what flexibility the component has, and the accessibility section described how the component follows the WCAG. See figure 30 below for wireframes of the component start page and a specific component page.

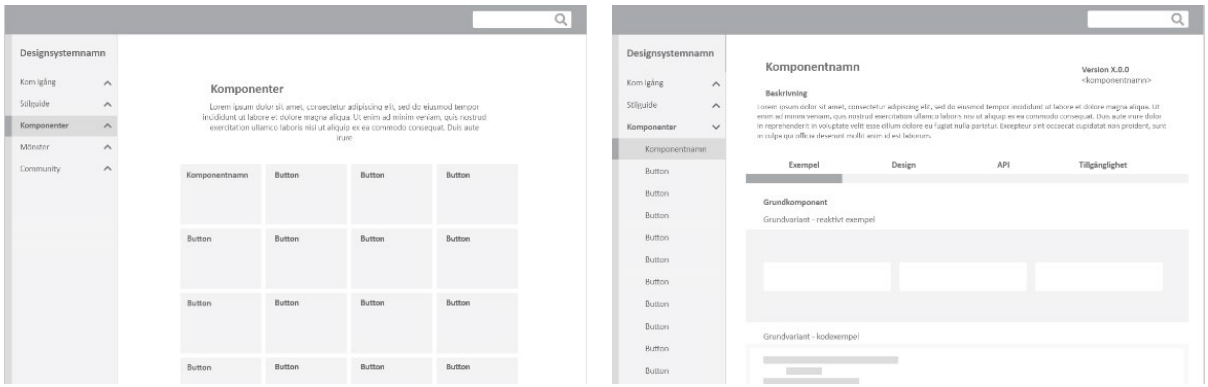


Figure 30. A picture over the first concept of the second iterations component start page and a specific component's page.

A pattern page on the other hand contained information about what the pattern concerns and which version it was. It had a short introductory text, then two different tabs of *"Example"* and *"Design"*. The tab example showed the ingoing components that the pattern consists of together with an interactive example of the pattern with a code example. The design tab had information about best practice when using the pattern. The design tab also showed whether or not the pattern had been in any user tests and what response and feedback it had received.

The community was about enlightening a user of how the design system gets worked on, which guidelines it has, showing the status of its content. The status was shown via a table, displaying which components of the system that was planned to be implemented, or if it was on-going or even completed.

Second Concept

The second concept evaluated was more inspired and structured as the current demonstrative website *"komponentkartan demo"*. Even so, it had new features which the other concept did not have, such as instead of code examples displayed there were links directly connected to the source code of the page on GitHub. By having the link, the user would be able to see the code in a context that would not need as much maintenance as it would change as the component would change in the code repository. To see the depth and number of wireframes used for this concept, see figure 31 below.

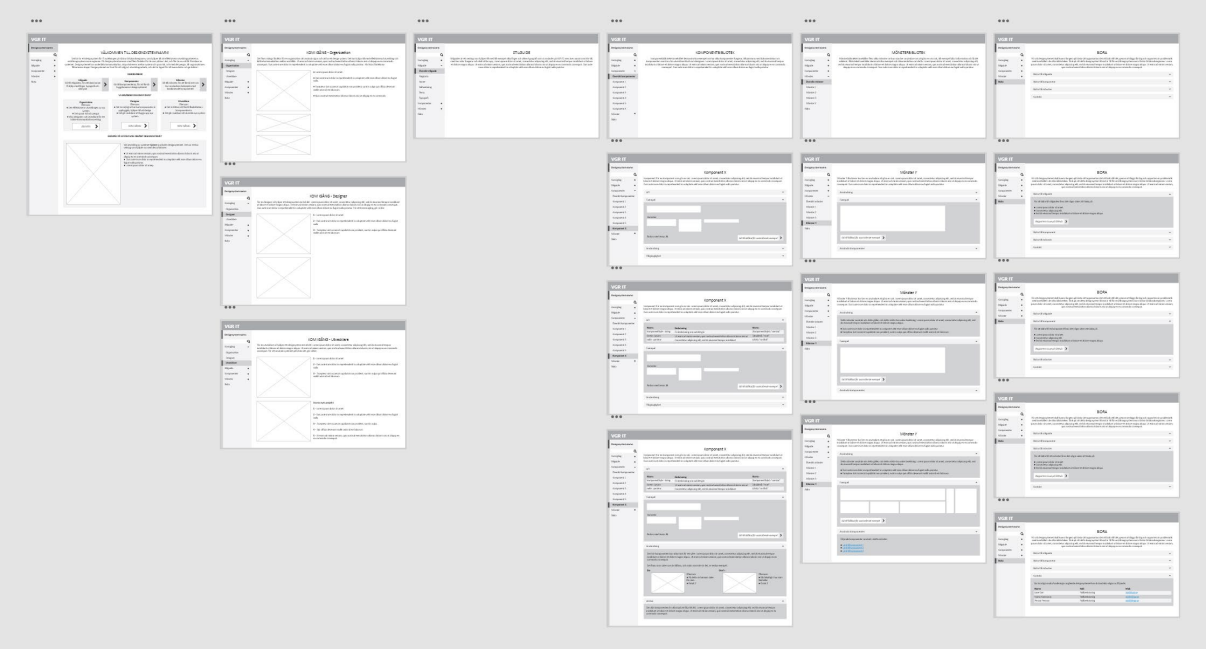


Figure 31. The wireframes for the second concept of iteration two, showing the different levels of the concept.

Similar to the first concept of this iteration, the concept had a search bar to easier navigate and find wanted information in a quick way. The second concept also had a start page with a selling pitch, but also quick links guiding a user to different parts of the design system that they might want to access. The start page also contained information about how the system could benefit certain roles, but also an example of a system that had used the system and been successful. To see what the start page looked like, see figure 32 below.

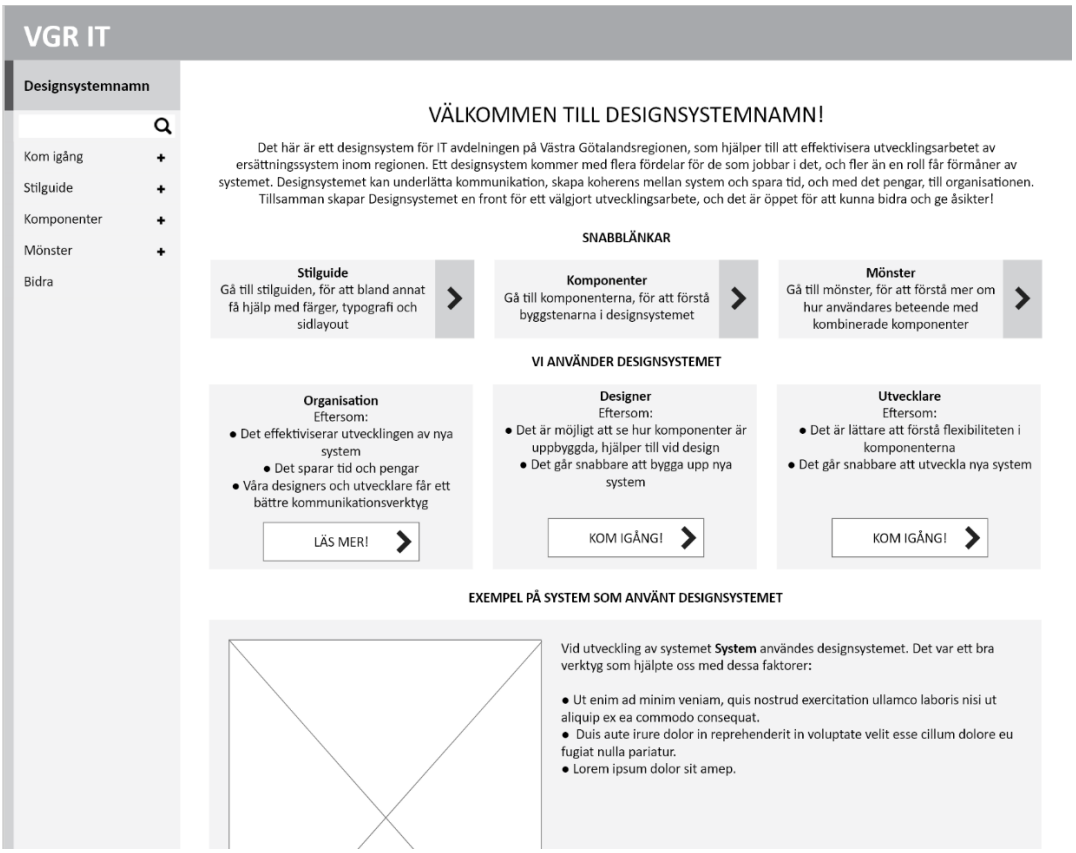


Figure 32. Wireframe over the start page for the second concept of the second iteration.

The other different areas in the concept were *"Getting started"*, *"Style guide"*, *"Components"*, *"Patterns"* and *"Contribute"*. Instead of having a start page on the dedicated side menu choice, the list showed when expanding a menu choice had the option of *"Overview"* at the top. The overview was to work as a start page but did not have to be clicked upon by a user if not desired. The overview acted in a similar manner with describing the content of the menu choice.

The getting started was similar to the first concept in this iteration, with how to get started as a designer or developer in a step-by-step manner. But here the term of *"Organisation"* as well was introduced, explaining how having a design system could help more than just developers and designers, benefiting more roles.

The style guide does once again collect the documentation found on the demonstrative website, that regards things such as theme, typography and colour chart.

A component page for this concept contained short information about what the component were, alongside four different drop-down list with the headings of *"API"*, *"Example"*, *"Usage"* and *"Accessibility"* which were placed on the page in that particular order. However, the example drop-down choice was open as a default showing interactive examples and the possible variants of the component. The API showed a table over the possible flexibility of the component. The usage dropdown contained information about why a component was created and for what purpose, alongside information of what to do and not with the specific component. The accessibility drop-down showed information about how the component followed the WCAG's. To see what a component page looked like in the concept, see figure 33 below.

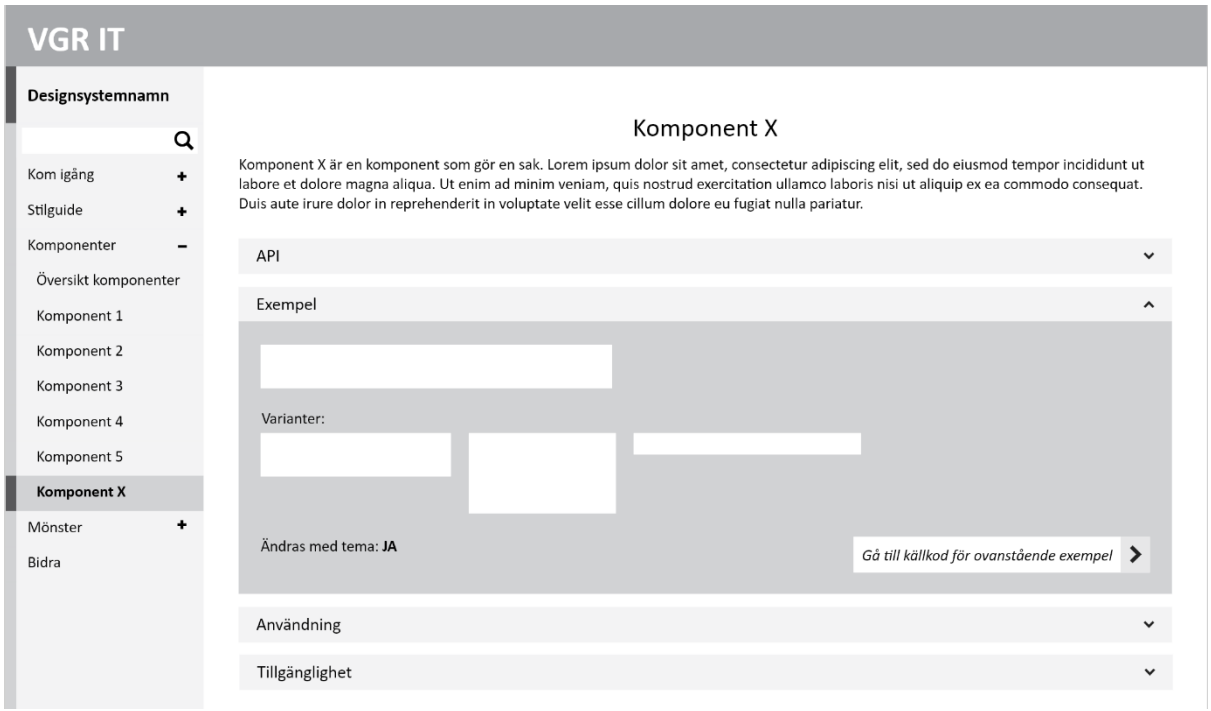


Figure 33. A wireframe over what a component page looks like in the second concept of the second iteration.

The pattern page was similarly structured to the component page, with an introductory text and drop-downs of *"Usage"*, *"Example"* and *"Used components"* placed in that certain order with *"Example"* opened as default. The usage described what the pattern is used for and what needs to be considered when using it, the example shows an interactive example and used components displays which components is used in the pattern.

The contribute menu choice was a single page with information about what to do as a user if the user would want to contribute to the style guide, components or patterns. On the contribute page there was also contact details to those responsible for the system. After each contribution description there was a link away from the page to where on GitHub this could be reported.

21.2 EVALUATION

The two concepts presented earlier in this chapter were evaluated together with users and the results from those evaluations can be read here.

First Concept

For the start page, it was appreciated by some that it was not similar to what the demonstrative website looked like. There were some opinions as well about the example displaying what had been done with the help of the design system and that it should be information that is more hidden away and not as displayed to a frequent user. However, all users appeared to like the fact of a short text selling in the system for a potential user.

The getting started instructions felt appreciated by the users, and it felt as if there was a lot of potential to make it better than the help provided today. However, some expressed that some things may be difficult to divide into designers and developers. They also then said that some information could be shared between roles, not excluding information since it belongs *"more"* to another role than it does to another.

For the style guide there were no strong opinions, mainly just agreement that the things placed under the style guide belongs there. However, there were questions regarding how adaptable it should be, if people outside of VGR as well could use and adapt the design system. Since the design system is said to be for the remuneration systems with the region, this should not be a priority in future work with the system.

The component start page was met with positive responses from the designer, since it was a good way to help with recognition, not having to remember the exact name of the components. From the developers, it was a neutral response, with some questions regarding how easy it could be to make every possible component into an easier sketch. It was as well expressed that the developers probably rather would use the list in the side menu, rather than a grid of components, since it is a fast way to get to the wanted tab.

The component page was met with good responses. Most of the users evaluating the concept thought it was good with tabs, to get an overview of the content. Although, one of the developers preferred the accordions, like the current demonstrative website has, as they liked blocking out the information they did not need. Regarding the examples there were some questions regarding what they would illustrate and how they would work, but it was agreed upon that interactive examples would be the best to display under the *"Examples"* tab. The tab would not either have to show all possible examples, just the most relevant ones. It was also preferred to see how the code example relates to the interactive example.

The design tab was considered as okay, and the designer was the only one with true interest in it. To call it design was considered as positive, since it created an idea of what you can find under that tab.

The API tab was the one page that the developers expressed that they mostly would use. The name of API was said to be the correct name. However, one developer was not sure they would find the information they needed in the design or API tab. To be able to use the API it needs to be more properly updated and it should contain better descriptions. It would also be favourable if it would be explained in such a way that a designer or non-developer could understand what is possible to do with the component.

In the accessibility tab the users wanted to see how the end user would use the component. One developer found the information there as irrelevant to them, since they already knew that each component already considers accessibility and the WCAG's. The label *"Accessibility"*, in Swedish, was a bit hard for one of the participants to relate to WCAG and the name can become a bit misleading, and then maybe WCAG would be a better name for the information it contains.

There were some difficulties surrounding the pattern page as the wireframe lacked real content and the users had to imagine what could be there. The participants all had the idea of patterns being for designers as it would tell how to put the components together in a use situation rather than being technical. One thing discussed by several users was whether the code examples on the pattern page need to be there or not. Patterns are more of suggestions of how to fit the need of a user pattern, and therefore they can change in their layout between applications. It was said that pictures or interactive examples would be enough to display.

The introduced community was generally thought of as positive, but it lacked clear purpose. A problem with the community was that the participants did not really know who it was for, or if the concept was meaningful with few teams. This since a community would require significantly more maintenance than the current demonstrative website has. The participants liked the different information found under the community page, but they were a bit hesitant to have it under something called just *"Community"*. They would be interested in the status of the components but having it all in a list would be excessive and would take away focus from the more interesting things, like what is planned and what currently gets worked on.

Second Concept

The start page was considered to be more compact than the first concept and the participants liked the very directed pitches of having a design system towards different roles. To have a directed pitch towards *"Organisation"* was liked. The quick access links to the style guide, components and patterns was once again not that appreciated by the users, since they felt as if they could find that information in the side menu.

For the *"Get started"* pages it was noted that it had to be consistent with the links and buttons. Also, there needed to be a well-thought of getting started part as there is not certain that a developer does not want to get to know how to get started with the design and vice versa. The label *"Organisation"* was a word that the participants thought fitted better than the *"About"* label in the first concept, since it was clearer who the information was directed towards. One developer thought the layout, with pictures next to the text, would be a problem when

developing for different screen resolutions. The other users though thought it was easier reading the information when in different columns.

For the component overview page, the first concept was mainly preferred as it simply gave a better overview. On a component page most of the participants preferred the layout of the first concept as it was easier to understand the width of information when in tabs. However, one of the developers still found the accordion layout better as they liked being able to close things they did not wish to see. One developer also suggested that it would like the system to remember which accordion was opened, having it as the same default accordion when going into a new component. The fact that *"Example"* was the default option when opening a new component was met with criticism, since if it is the first to open, there were questions about why it was not on the top.

On the pattern page there was the same reasoning as in the previous concept. The developers agreed that a link to the source code could be a good idea if the pattern was complex. There was also a discussion if there would need to be an interactive example at all, or if it could just be a picture as an illustrative example instead.

The contribute page put another focus and perspective on the information there, rather than having it named community which the first concept did. Like the community page in the previous concept, the page needed to be further defined for the users to make sense of it and it needed to be distinct from the issues-page on GitHub where issues currently get reported.

Things to Iterate Further

When asking the participants what they wanted on the get started pages, they answered that it should be more step-by-step instructions for developers, however still expressing freedom for them to work. The designer expressed that design may not be as linear as the developers work, but as a designer it may be more applicable to get tips, creating clearer headlines for the getting stated page. This could show a designer how to use the design system as a tool for design. For the next iteration the onboarding process and pages was refined to become more specified.

There was a bit of confusion about what the purpose of the quick links on the start pages, as well as the examples of applications that had used the design system. The quick links were expressed as unnecessary and will be iterated until the next concept. Although the example could be useful for some roles and new users, it did not feel necessary to have on the start page.

For the next iteration, the component and pattern pages were to be based on the first concept as it was more appreciated as was expressed to give a better overview and a cleaner and more collected look with the tabs. The designer noted that the design of the tabs looked a bit like a progress bar, which would need to be redesigned for the next iteration.

In the next iteration's concept, the possibility to edit on GitHub was to be used as a valid way of editing between roles. By letting other users than developers edit there could be a better documentation of rules and standards, but also reduce the effort of keeping the pattern library's documentation up to date.

For the next iteration there needed to be an iteration of the community and contribute page. For the users it was important that the design system focused on those that were using it and be more reserved towards big words that would imply that other people outside of VGR could use

it as well. The new page would rather collect the tools and information necessary than being a way to host the community or contributions, going more towards this iteration's second concept's solution.

For the design system there needs to be clear guidelines for what a component and pattern respectively is. The design systems website, the pattern library, should be the central tool when accessing information about the design system and how it works. Therefore, in the next iteration there needed to exist more refined examples and example texts of what the documentation should contain. Going further, the wireframes was to be of higher fidelity as there needed to be more content for the users to give feedback on.

22 THIRD ITERATION & EVALUATION WITH USERS

For the third iteration there was only one concept created of the design system website. For this concept, there was no placeholder text, and it was instead filled with examples of actual text to create better context. This gave the concept a lot higher fidelity so that the users evaluating it could understand the message from the different pages in a better way. The concept was created with the input from the second iteration. On overview picture of the depth and number of wireframes created can be seen in figure 34 below.

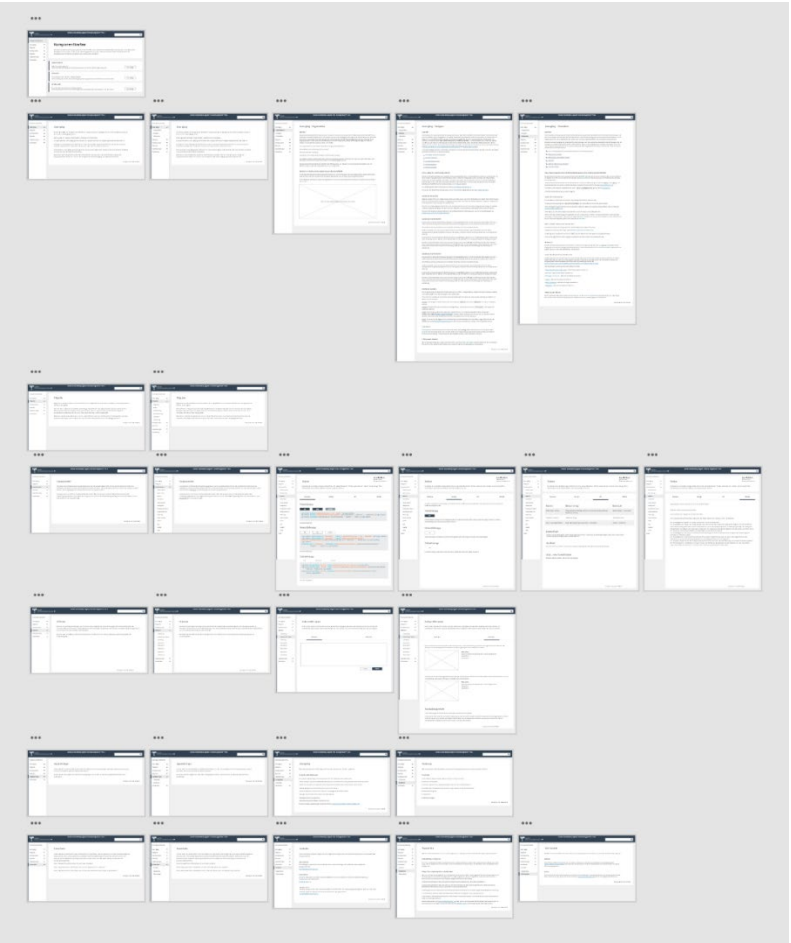


Figure 34. A figure over the depth of the third and final iteration, displaying the number of wireframes connected in it.

22.1 OUTCOME

This concept was designed to not be demonstrative of the components but was rather displaying and showing documentation about the components to the users of the design system. This meant that the components were tested away from this site, being already assured of its quality before getting put up on the site with belonging information.

For this concept the quick links on the start page were removed due to excessive redundancy from what could be found in the side menu. Instead there were pitches directed towards the roles of *"Designer"*, *"Developer"* and *"Organisation"* and links that would take the user to the respective get started pages. They would also be placed horizontal as it would be easier to implement for responsivity for the different desktop window sizes.

The side menu had some menu options such as *"Get started"*, *"Style guide"*, *"Components"*, *"Patterns"*, *"Updates"* and *"Collaboration"*. The menu choices were all supposed to have start pages that describe what that section is about to help a new user differentiate between the different menu choices. This page could be accessed by pressing on the menu option.

The concept of this iteration combined the insights from the previous iterations and divided the *"Get started"* section into *"Designer"*, *"Developer"* and *"Organisation"*. The pages were given appropriate text to show the users what the focus and content was thought to be fitting on the respective pages.

In the concept the users with permission could be able to edit the page text on GitHub. The way it works is that they edit a text file with markdown directly online that is merged with the code for the website. The edit option would be present for every page except for the example tab with interactive examples, since these would be difficult to edit if the users would not be a developer.

For the component page the information was divided into tabs with a general information placed above the tabs. Above the tabs, there was also information showing what version the component was of and its code selector component name, as well as if it changes with the theme. The tabs were *"Examples"*, *"Design"*, *"API"* and *"WCAG"* which worked similarly to earlier concepts. To see the layout of the component page, see figure 35 below. The example tab displayed different interactive examples with belonging code, and the design code described the different variants of the component and how it was to be used. The API had short list describing the flexibility of the components, but also had longer descriptions about each flexibility below. The tab about WCAG tells how the component is adapted to and follows accessibility guidelines.

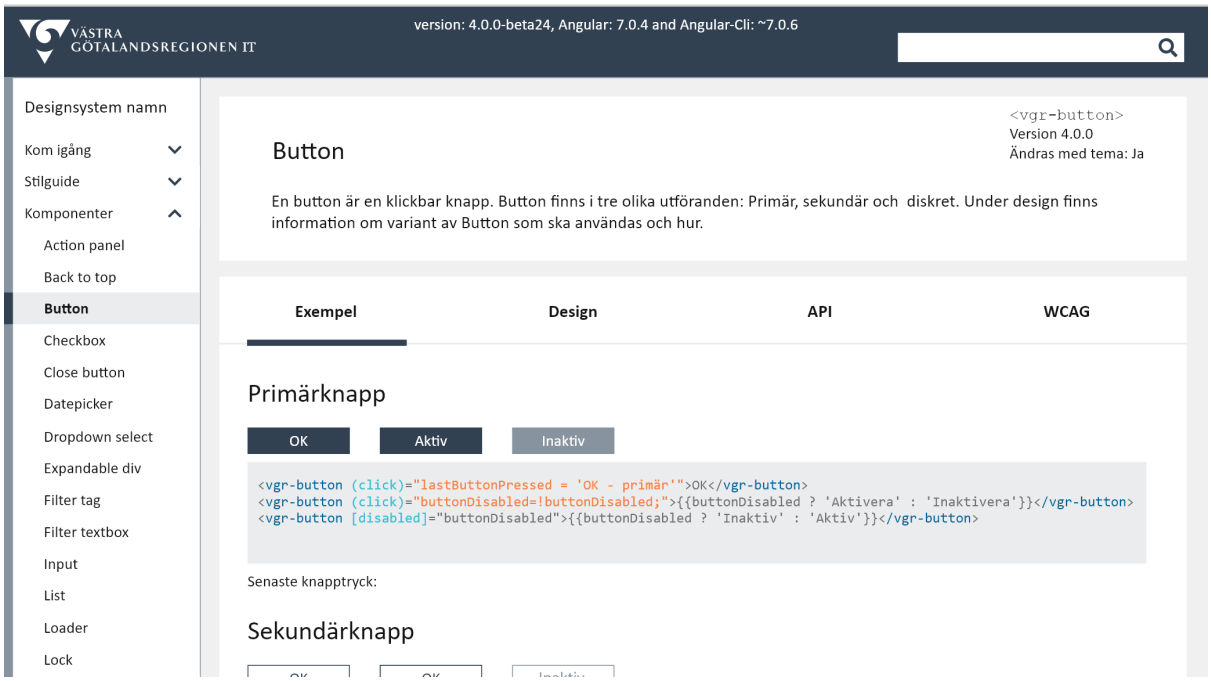


Figure 35. A wireframe over the component page for the third iteration's concept.

The pattern page had a similar layout to the component page. However, there was no code example for a specific pattern, since it is a combination of several patterns and is an illustration of a specific user case. In the concept there was an interactive example, instead of pictures or videos, that illustrated the pattern. A patterns page had the tabs of *"Example"* and *"Guidelines"*, where the guidelines described how and when to use the pattern, and information whether the pattern had been tested with a user or not.

The *"Updates"* menu item included a change log and a roadmap. The change log described what was different from the previous major version of components while the roadmap included what was to come soon, but also further ahead.

The concept also introduced *"Collaboration"* instead of community or contribute as it was to put more focus on the small scale of the design system, uniting the teams using the design system. It was also chosen as an option to give the users a place to learn more about how to possibly collaborate, who to contact, how issues were reported and what channels and tools the design system uses to communicate.

22.2 EVALUATION

It was made clear to the users that this concept had lifted out the demonstrative parts of the design system website and allowed for testing on another site before bringing in the finished, unique component into the design system. The users thought this idea was good and would improve cleanliness of the design system website as the pure separate, test and demonstrative site could handle all expectations and variations of components.

In the evaluation there was feedback on dividing the getting started pages. In the concept it had been mixed, both for developers and designers, with how to start using the design system but also how to develop with the help of it. From the evaluation it would be better to make the two

cases more distinct, but also to emphasise what standards and rules should be followed when developing.

For the component start page it was appreciated with the clarification of what a component was. It was also brought up during the evaluation to only keep unique components on the design system website, but write under the design tab what the component could be altered into and what patterns it was included in. An example of this was to remove the current *”close button”* component and just have it as a design variant of a button component instead.

The pattern page’s response was positive as it removed the complex, often excessive examples, on the current demonstrative website and replaced them with user patterns instead. It would serve to limit the generalness and provide set practices across teams. It would also limit the set of components as some of them would become patterns.

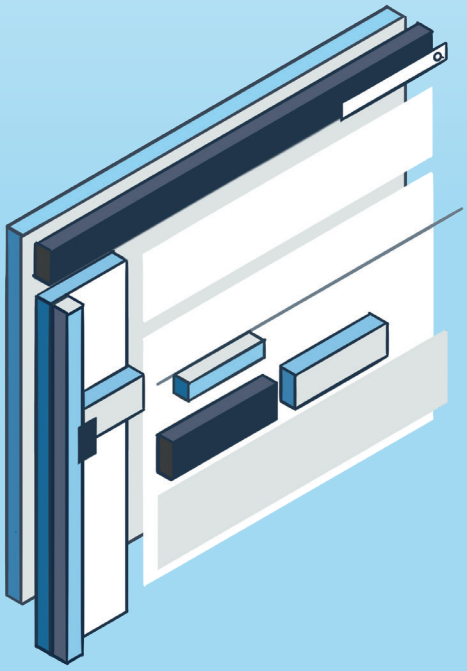
The API description was similar to that of previous concepts but introduced a more elaborate explanation of each of the APIs in a list under the table with brief examples and descriptions. The list would not interfere with the table, which has the purpose to give a quick overview, but to support the table. As it is very important for all users in the category of developers to understand the API and know what can be done with the component it was important that the properties were well described. It was also expressed as a wish from the designer’s perspective to be able to understand the API better. The ease of editing trough GitHub would also be able to help the information being up to date on the different pages.

The menu option named *“Collaboration”* was received well and had a more balanced approach to the process of the design system. It focused more on the teams at VGR IT and their specific tools and how to improve the design system. However, there needed to be a distinction that the design system primarily would support remuneration applications for VGR and that other suggestions might not be prioritised if there only is funding to support their own applications.

The outcome of this iteration lead to guidelines concerning the content for a final design system website, which was similar to this final concept.

Deliver

In this phase the refinement of a design system concept is described in detail as the result of the project. The final delivery was guidelines that were exemplified in wireframes and illustrations. These could become a groundwork for the organisation to follow and build on further.



23 THE NEW DESIGN SYSTEM

In this chapter the final concept for the new design system is explained alongside guidelines of how to fill it with appropriate, coherent information. The design system aims to be the single source of truth. This way, quality can be ensured and reduce the risk of having inconsistencies.

The design system earlier was and should still mainly be for the remuneration systems within the region of Västra Götaland, as they are the teams maintaining the design system. Going further with a more coherent design system, the needs of the remuneration systems need to be primary, whilst it still can be open for others at the VGR IT department to use the system.

The design system will consist of several ingoing parts, like the definition made in chapter 12. For this final concept, the ingoing parts of the design system for VGR should become clearer for the user. The parts should have their respective purposes, coming together as a complete design system. The different parts that should exist should be the language that affects the whole design system, but also the design system website, the test site and the code repository that makes up the design system. These parts are illustrated in how they make up the design system in figure 36 below. Behind these parts are also the process of the design system which should be led by the design system team.

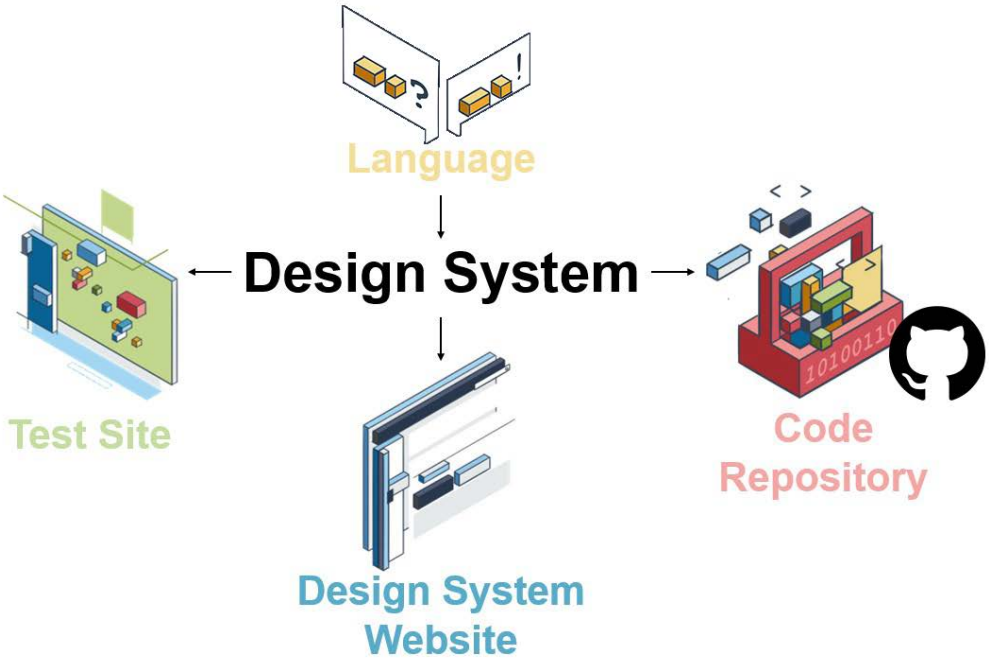


Figure 36. Illustration of the different ingoing parts of the final design system. Authors own illustration.

23.1 LANGUAGE

For the design system it is important to phrase and name things in appropriate ways. Without a common language the collaboration process becomes hindered and causes confusion. Therefore, it is essential with a language that everyone can understand and communicate with.

For example, the actual name of the design system should be non-descriptive and just represent the design system as a holistic system. The current name used for the system, *"komponentkartan"*, gets translated into *"the component map"* in English and has the interpretation of being a map of some kind. The fact that it gets called a map, comes with certain expectations and thoughts about what it denotes.

The other parts of the design system such as the code repository for example, should as well have appropriate names, explaining the different parts differences. But also, the names of the components should be conventional, and the meaning of the names should be agreed upon between different roles and match that of a standard. Examples of conventional names for different components are *"button"*, *"modal"*, *"accordion"* and *"date picker"*. The names given to the component should also be complied with, not mixing different languages such as Swedish and English. The language used throughout the system should also be proper and correct, without complicating descriptions and documentation. The information found in the system should be able to be comprehended by managers as well as designers and developers.

23.2 DESIGN SYSTEM WEBSITE

The design system website should be the face of the design system and the tool to collect and distribute information to the users of the design system. It should include documentation for the components and patterns. It should also explain the processes in the design system and include an onboarding process to the design system for the new users. Although there can be information and necessary parts of the design system hosted elsewhere the design system website's purpose is to collect these to one central place and mediate to the users where to find these resources.

The design system website should therefore be the pattern library, where the process as well gets displayed. A pattern library is a central part of a design system and becomes the face of the design system itself. The pattern library should get more status and be used as a tool to connect all the other tools that the design system consists of.

The design system website should also be lifted out from the component code repository to be able to handle versioning better. To separate these, it is also necessary to ensure that the website will not be as nested with the components. The pages with text should be able to be edited directly on GitHub by those with the permission to do so. The design systems coded components and the design system website will become easier to talk about. They will not be as mistaken for each other, as they would be two different things, and not one thing nested into each other. The design system website should mainly consist of components from the design systems components, with the website rather being built with it rather than being a part of it as one big website component.

23.3 TEST SITE

The test site, which also should include room for making demonstrations, should now become separated from the design system website. Having a site dedicated to testing and showing demo should remove the excessiveness and confusion that has occurred when it was mixed with documentation. The test site should not have to serve all the user groups, unlike the design

system website, and can be fitted to the developers and code testers needs. However, it can be a site to see the components and patterns in a bigger context but should not be presented as the site to trust with how it is disposed.

A recommendation for the test site would be to allow for experimentation in an environment that is already set up. It can allow for a more playground focused area, which sole purpose is to try new things and let users try to build things with the components.

The test site is to be a complement to the design system, where the different components can be tested by code testers to ensure the functionality and flexibility of the components. This is to be done before the components in the code repository gets placed in the pattern library and the design system website, ensuring high quality and tested components. A figure of how this could work can be seen in figure 37 below.

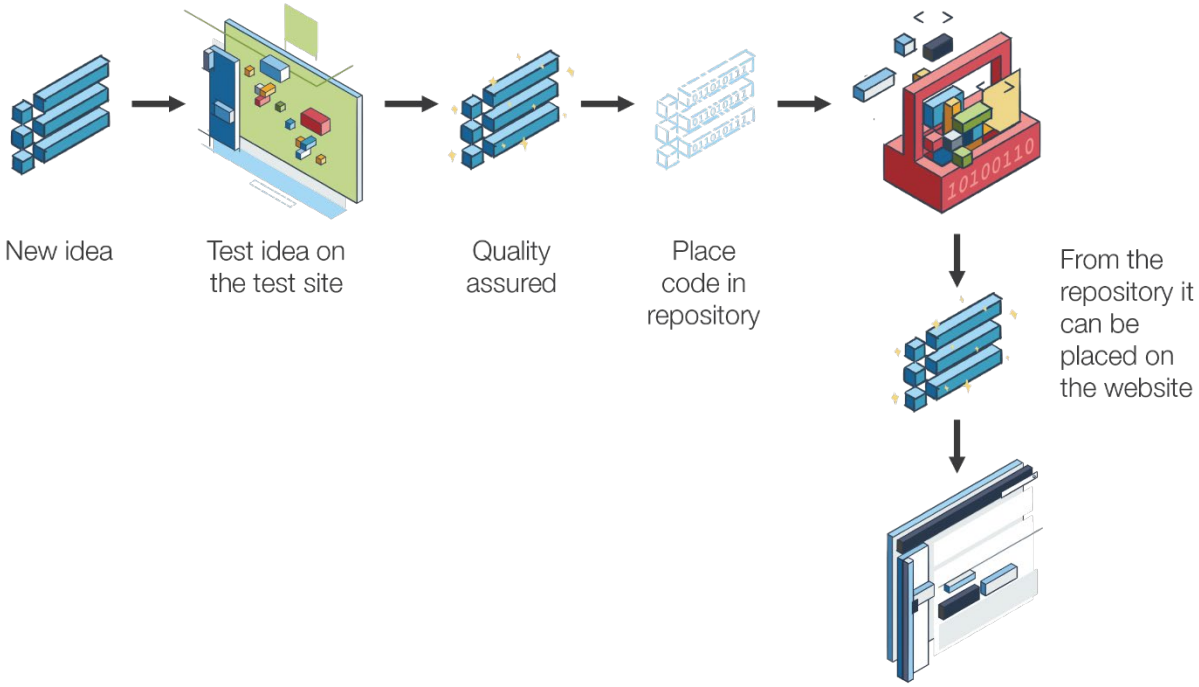


Figure 37. An illustration shows how new contributions and ideas gets implemented in the design system, ensuring the ideas quality and possibly improving it before placing in the code repository to be able to put it up on the website.

23.4 THE CODE REPOSITORY

The code repository holds the design system components. This design system's code repository should now be stripped down of the documentation and should solely include the components. This way the code repository can be easier updated and can focus on the components. The components can then have their own versioning. Meaning that they will not have to have their version changed if the design system website's text gets updated as it was before. This makes the version changes more comprehensible than they were before.

The code repository should be hosted on GitHub, similar to how it was hosted there before. As there also will need to be code repositories for the design system website and the test site, these should also be hosted on GitHub. The repositories are after all what builds up the different sites, and therefore the other parts of the design system require repositories. The important part is that they are not nested in each other anymore and more clearly separated. The different

GitHub pages should collect the issues and bug reports made for the design system components, the design system website and the test site respectively. The GitHub page for the design system components should also include how the issues should be submitted, and the code standards the components should follow.

24 GUIDELINES FOR THE DESIGN SYSTEM WEBSITE

To aid the work with the design system there have been guidelines specified to do just this. The guidelines concern the content of the design system, which now has the different options of the introductory menu items of start page and getting started. Then there is the different documentation of style guide, component documentation and pattern documentation. To help and enlighten a user to know what is happening in the system, what is planned, or how they can contribute, there now will be the two menu items of *"Updates*" and *"Collaboration"*. The different headings and their meaning are further explained in this chapter. In the appendix it is possible to see the wireframes for the suggested design system website, with example content.

24.1 LAYOUT

The layout of the design system website should change a bit from the original demonstrative website *"komponentkartan demo"*. Some new things should be introduced, and the information should become structured in a new, more logic way.

The design system website should have a search function, since this was appreciated during evaluation of concepts with users. Not everyone may use it, but a simpler search function can help a user to use the system in a smoother way.

The menu on the side is divided into certain menu items, which should be a start page, *"Getting started"*, *"Style guide"*, *"Components"*, *"Patterns"*, *"Updates"* and *"Collaboration"*. This is a way of hierarchically structuring the menu so that a new user can get help, and at the same time get an idea over what the design system incorporates. The order of the menu goes from generality, at the top helping everyone, going into more narrow areas such as how to collaborate with the system when established with it. Therefore, it is important that each heading in the menu should have their respective start page, so that a user can learn and understand what each menu item's area is about and so that it stays with its original purpose.

For the menu, it should be possible to click on the name of a menu item to get to its start page, but if a user do not want to go to the start page, it can click on the symbol to its side, a so called chevron, which display what is beneath that menu item. This navigation, of not having to always go to the start page, aids experienced users to quickly find the information they desire.

The *"back to top"* helper that earlier have been on the demonstrative webpage should still be on the design system website. This since it is a good way to quickly get to the top of the page, and if a user would like to change to another tab, they can quickly get to the top to do so.

It should also be shown at the top of the page which is the current version of the code repository of the design system. This should be so that a user can double check if it may be a new version from the last time that they used the design system website. If it is a new version, the users can always look under the *"Updates"* section to know what is new or has been changed.

To aid development of the system, and to aid collaboration between designers and developers working with the system, the different pages in the design system website should have an *"Edit text on GitHub"* button in the lower right corner. This link takes the user to the respective page in the design system at GitHub and makes it possible for someone to edit the text, if they have the permission to do so. Any kind of user could suggest a change in the text, but those with permission can easier edit the documentation more directly, easing development and cooperation. However, it may be too complicated for designers to change the interactive examples, so these pages of the system do not need that function and should be left to be edited by developers.

24.2 START PAGE

The start page should help someone new to the system to understand what it is and what purpose it has. The start page should be arranged in a way that catches the attention of a user and should firstly express the bigger benefits of the system, to get them on board wanting to use it. Therefore, the start page should contain:

- A big headline with the name of the design system
- A pitch at the top of the page, quickly and briefly describing what the design system is, also what it can be good for.
- After the pitch, there should follow *"cards"* that tells the benefits of the system dependent on which role or association a user could have to the system. These different roles are different types of stakeholders, which can also be called organisation, designers and developers.

The start page should be there, as earlier mentioned, to grab the attention of potential users and sell the benefits of the system in a good way. It is also there for new users, so that they are met with what the design system is, what it is useful for, and information about where more information can be found.

24.3 GETTING STARTED

There should be a start page for the getting started section of the design system, guiding new users that may have missed the information on the start page of the website. The information in the text for that start page can contain:

- Information about what getting started means and why it is important to read and understand the information in the section.
- Information about each menu item under *"Getting started"*, and what interest there may be for different users in each of the categories.

- Information about that it is encouraged to read all of the information, no matter which position a user may have, to gain a better understanding of what the system is, what is possible to do with it and also how it can help.

24.3.1 ORGANISATION

The getting started for the organisation should sell benefits with the system, aiming to capture the interest of those in charge and managers within the organisation. The organisation page for how to get started with the design system can contain:

- An introduction to the design system, selling in why the system is beneficial to have, expressing things such as:
    - The system creates coherence.
    - End-users will have a better user experience with the products.
    - The design system contains building blocks, easing for new systems to be developed, which increases efficiency during development.
    - It is possible to save time and money.
    - It gets easier to follow the web content accessibility guidelines (WCAG) since the components in the system are created with those in regard.
- There should after the introduction be information about how the content of the design system gets used, and with that also try and display examples of what a final product could look like. A good way of showing this implementation of the design system can be to display an application before and after using the design system, selling benefits in a way that has more context.

24.3.2 DESIGNER

As a designer getting started with the system, there is no real linear process of which steps to take to get started. The getting started guide for the designers should rather be tips and tricks of how the system can be used to help a designer with their work. Since the design system contains a lot of technical information regarding code and similar, it is good to aid a designer with what they can make use of. To help navigation on the get started page, there should be links taking the user to a specific part on the page, similar to an interactive table of content. This to quickly take the user to the information they need. These links should be the follow up of an introductory text. To see what this kind of link could look like, see figure 38 below.

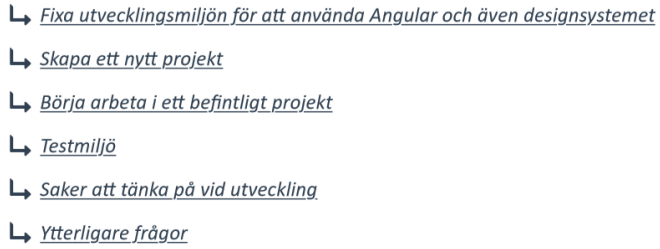


Figure 38. A picture over what an interactive table of content could look like, where the arrows illustrates that by clicking the link it is possible to get guided to the chosen area.

In short, the designer getting started page should contain this information:

- There should be an introductory text describing how the design system helps a designer in their work, containing:
    - How it aids communication with developers.
    - That it helps to maintain a coherent look for an application or system.
    - That the design system follows WCAG and therefore the components are adapted after those rules and regulations.
- Information about how to get access to the Sketch-library, where there are vectorised illustrations of each component, which helps designers to create user interfaces. Therefore, it should contain:
    - Information of what Sketch is.
    - How to get started using the components.
    - Contact details to the person responsible for it.
- Information about how to use the different tools available in the design system, such as the style guide, components and patterns together with the test site. It should describe:
    - How to use those tools.
    - What information that can be found and how that information can be interpreted and used, for example tips of how to interpret the API descriptions and how that can help in the design work.
- Information of where to find the test site. It will help the designer with visual examples of what has been done with the design system earlier and let them understand the context behind the components better.
- Information about how to proceed if there are further questions about the design system. This page can link to the *"Contact"* section under *"Collaboration*" in the menu to guide the user further.

24.3.3 DEVELOPER

The get started page for developers should help them to set up and start using the design system, following a quite linear, step-by-step path to help them. The page should start with an introductory text over what can be found on the page, but also benefits of using the system. The introduction can also shortly explain that development follows WCAG and helps a user to find out more information about that. The structure of the site should be similar to a designer, with quick links after the introductory text that can take a user to the part they are interested in reading on the page. The information on the page should be:

- Step by step instructions of how to set up the development environment.
    - Explanation of that the design system uses Angular and helps to guide them to external resources if they never have worked in Angular before.

- Step by step instructions of what to do when creating a new project, with detailed descriptions, put on a level that can help those new to the design system.
- Step by step instructions of what to do when starting to use the design system on an already started application. There should be thorough instructions here as well.
- Information of where to find the test site. It will help the developer see examples of what has been done with the design system earlier and let them understand the context behind the components better.
- Information about things that are good to have as a developer working in the system, such as plug-ins and tools.
    - For each thing, it should be described what that thing is, and also an explanation of *why* it is good to have just that thing.

24.4 STYLE GUIDE

The style guide should gather information regarding the look and feel of the design system and will describe the thoughts and philosophy behind application content. This means that the style guide helps to create additional components or applications that go in line with the desired expression. This so that the whole application or product created with the help of the design system should look and feel as if it belongs and fits in. It should have a start page describing what it is and what can be found within it, written for someone that is new to the system. It should contain information and documentation such as theme information, what formats are possible, symbols, typography and colour chart, to mention a few.

24.5 COMPONENT DOCUMENTATION

The component documentation consists of a start page that introduces a new user to what a component is, and then also all the different existing components with its documentation. The documentation about the different components should as well be easy to access via a list in the side menu.

24.5.1 COMPONENT START PAGE

The component start page should have a text describing and explaining what a component is in a clear way.

- The text should help to keep the definition of the components consistent over time.
- The text should explain how the components are unique and reusable.
- The text should explain what a component page includes and what information can be found there.
- The component start page should have a link to take the user to know more about contribution to components.

24.5.2 COMPONENT PAGE

A component page should contain documentation gathered about a component. It should serve to help the user use the component in the best way possible and help them find the information they require.

- The component page should have a short description about the component, what it is, what makes it unique and what it enables for the end user.
- The component selector name should be located in the same place across components, to the right of the name of the component.
- There should be information about during which version of the code repository that the component was last updated. This way a user can understand how updated it is compared to which version the system currently is at. This can then be easily compared to the version description at the top of the header.
- Information about whether the component changes or not with different themes should be visible and in the same place as the information about which version the component is at.
- The information should be divided into the following tabs *"Example"*, *"Design"*, *"API"* and *"Accessibility"*, where *"Example"* is marked as default when entering a component page.

To see a picture over what this page could look like, see figure 39 on the next page.

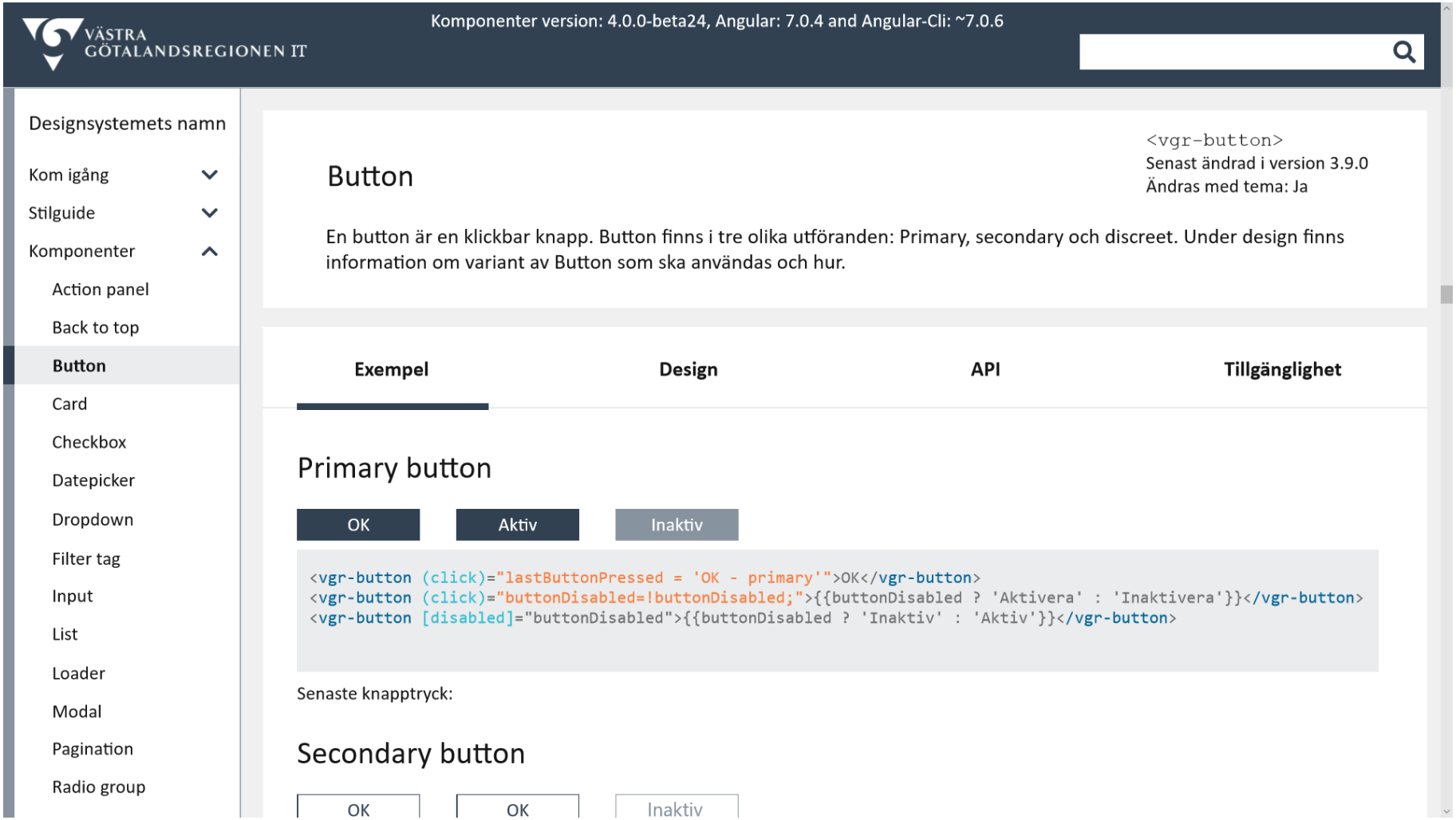


Figure 39. The wireframe over a component page for the final concept, displaying the "Example" tab in the documentation.

Example Tab

Example is the first tab visible to let the user see the component and be able to relate it to the descriptive text.

- The tab should at first show an interactive example of the component.
- The information can include examples of variants of the component, but it should not become extensive and cluttered.
- Below the interactive example, there should be code example included of each of the interactive examples.
- If the code example is too long, it can be hidden under a *"show more"* button.

Design Tab

Design should be the second tab after *"Example"*. In the design tab there should be more information of the purpose of the component and what is possible to do with the component, but also what guidelines exists concerning the usage of it.

- It should include when it is appropriate to use the component.
- It should talk about variants, but that can be shown in pictures.
- It should include links to the patterns that includes the component.
- It should include a section of user testing and possible validation if the component has been tested with end users.

API Tab

API is the third tab and includes information about the properties and possibilities with the component and how it is coded.

- The API is introduced in an overview table with name, description and short example of each of the API.
- Below the table there is a longer explanation of each API with the purpose to give a better explanation for other users than developers to understand what is possible to do with the properties of the component.

Accessibility Tab

The last tab is the accessibility tab. It serves to make the accessibility focus of the design system visible.

- The text should describe how the component aids accessibility of different kinds.
- The text should explain how the WCAG guidelines have been implemented.
- It should explain to what level the component is accessible, AA or AAA by the WCAG criteria's, as all of the components should at least reach the AA standard.

24.6 PATTERN DOCUMENTATION

The documentation about the patterns should consist of both a start page that introduces a new user to what a pattern is, and the documentation should also include all of the different existing patterns. The documentation about the different patterns and an overview over which exist, should also be easy to access via a menu item in the side menu.

24.6.1 PATTERN START PAGE

The pattern start page should be similar to a component start page, being a text expressing to a user what a pattern is, what it can be used for, and what it is beneficial for. The start page should emphasise the purpose of having patterns.

24.6.2 PATTERN PAGE

A pattern page should contain documentation regarding the pattern, also expressing why it is useful and for which use cases it serves. On top of the pattern page there should be:

- An introduction which describes what the pattern is and what it means for the end user.

The page should be divided by tabs, to easier separate information to help the user find the correct, necessary information it requires. There should be two tabs for the pattern page, and those should be the *"Example"* and *"Guidelines"* tabs.

Example Tab

The first tab is *"Example"* and shows the pattern.

- The example does not have to be interactive, but it should serve to illustrate the pattern in the best way possible.
- If the pattern is interactive there should be a link to the source code of the example below the actual example.

Guidelines Tab

The second tab is guidelines and with the help of text and pictures, the pattern should be explained. The documentation about these guidelines should include:

- When the pattern is used.
- The different variants of the pattern that exists.
- Instructions of what the best practice is.
- Results from user tests where the pattern has been tested.

24.7 UPDATES

Updates should be a menu item to make a user more aware of the process surrounding the design system. The user will get onboard with what has happened in the development of the system, such as latest updates and information about what is planned to be made with the system in the future. There should also be a start page for the menu item of *"Updates"*, which should contain information about what can be found under it. Such as a changelog and a roadmap, describing the difference between the different choices. The start page should also explain the benefits of having this information displayed, and how it can ease for the users of the system to understand and take part of that information.

24.7.1 CHANGELOG

One menu choice for the updates section of the design system can be the changelog, describing what has been changed in the latest update of the system. This way, a user can backtrack what has happened if it is interested about whether a component that they use for their system has been affected or not. On the page there should be information about:

- What has changed in the design system during the last major change.
- Why those changes were made.
- A link to the GitHub page, which hosts all of the changes made to the system from the beginning.

It is also possible for the changelog to show all the changes ever made, but the information showed should be at least the latest update. This to make the users more active in what goes on with the system. To only display the latest update and then refer to the GitHub page creates an easier maintenance. This alongside the *"edit page"* option will make it possible for every team member of the design system to easily contribute and help with the maintenance of the page.

24.7.2 ROADMAP

The roadmap is a way for the users of the system to understand more about what is planned ahead. This way the different users of the system can prepare or understand how future changes may affect their applications. Under this menu choice, there should be information about:

- What is planned long term for the design system.
    - Dates for when releases are planned.
    - Information about what is planned for those releases.
- What is planned in the near future.
    - What the next changes will be.
    - What the biggest changes in those planned changes are, describing minor and major changes.

This page of the design system needs to be updated and maintained, since when a change is implemented, it should move into the changelog instead, and the new, planned changes can be displayed and re-written on this page. With the *"edit page"* options, it should become significantly easier for all of those in the design system team to make changes, and not just developers.

24.8 COLLABORATION

The collaboration menu-choice is about how a user can collaborate within the design system. It is how to make contributions to make progress in the design system and giving it maintenance. This is a way to bring in the users of the design system into the process of it, similar to the menu choice of *"Updates"*. The collaboration part should contain information of who to contact in need of help, how to report problems with the system, how to contribute to it, and what to do to stay frequently updated with what happens in the design system.

Like the other menu choices, the collaboration menu-choice should have a start page. The start page is once again helpful for a new user and reminds the experienced users what the purpose of it is. The introduction page does therefore require certain information. It should contain:

- Information of why you should want to collaborate within the design system as a frequent user, describing the benefits of doing it. Such as:
    - Staying updated.
    - Using the design system to its maximum potential.
- Information about what collaborating to the system may include, describing the menu choices belonging to the collaboration choice, information such as:
    - Who to contact when in need of help or aid.
    - How to contribute.
    - How to come with contributions.
    - How to stay updated with what is going on.

24.8.1 CONTACT

Under contact there should be information about who to contact. There should as well be information about different responsible roles in the design system and how they possibly could help a user. The documentation should be:

- Name and title of the different responsible roles.
- Information about the different roles, and what kind of questions they may be able to answer.
- Information about how and when these people can be contacted.

24.8.2 REPORT

The users of the design system may want to report faults in the system, rather than suggesting changes and new components and patterns. Therefore, the option of *"Report"* should be placed over *"Contribute"* in the menu items for *"Collaboration"*. On the page of reporting, there should be information about:

- How to report a problem, preferably linking to GitHub, underlining why it is important to report properly to help the development of the design system.
- Explanation of what a possible problem may be, to aid someone with reporting in a problem.

24.8.3 CONTRIBUTE

The possibility to contribute to the system should be clearly expressed, to encourage users to help with the development of the system. This section should contain information about:

- How to make suggestions for a new component or pattern.
    - Instructions of the information a new suggestion should contain:
        - Such as if the errand concerns a pattern or a component.
        - Motivation of why the new suggestion should be implemented.
    - Criteria's that need to be fulfilled for the suggestion to be considered or sent in.
        - It should be general, not being specified to one specific system.
        - Be unique, being different from the existing components.
        - Added with the suggestion there should be an example or illustration of that specific suggestion.
        - A motivation is required to explain why the suggestion should be taken into consideration.
        - If the suggestion is a replacement of an existing component, motivate why the change should take place.
    - Information of where and how the suggestion should be sent in.
    - Information about the feedback time on a suggestion, and also how feedback should get provided on the suggestion.
- How to edit pages of the design system on GitHub.
    - Highlighting that this is prioritised by those working in the system, such as the team responsible for the design system, but that suggestions are welcomed.
    - Express that if a user is not authorised their change will come in as a suggestion, without guarantees that it gets accepted.

24.8.4 OUR CHANNELS

There should be a menu item of the channels in which the design system acts and is discussed. Right now, the different channels are GitHub and Slack. The users of a design system may have interest in knowing actively what is going on, not just via the development tab in the menu. Therefore, they can have contact in the different channels that the design system acts. On this page, there should be information about:

- The different channels that exists and what type of information that can be found on the different channels.
- Information about how to get connected to those channels.

25 GUIDELINES FOR THE PROCESS OF THE NEW DESIGN SYSTEM

A design system is more than just its documentation. It needs to be a living thing that is maintained and constantly evolved. Therefore, the process and the work behind it should be set out to ensure that it will be maintained and keeps up with the demands that is put on the design system. In this chapter follows suggestions of guidelines of how to achieve a sustainable process, with a base in what has been found out through interviews, user evaluations and literature.

25.1 DESIGN SYSTEM TEAM

To make sure the design system lives on there needs to be governance. For the final idea of the design system there should be a core design team of the product owner and a technical lead together with a team of rotating design system team members. The technical lead supports the product owner by being responsible for the technical questions such as implementation, feasibility, or the complexity of issues. For an easier illustration of how the team could function, see figure 40 below.

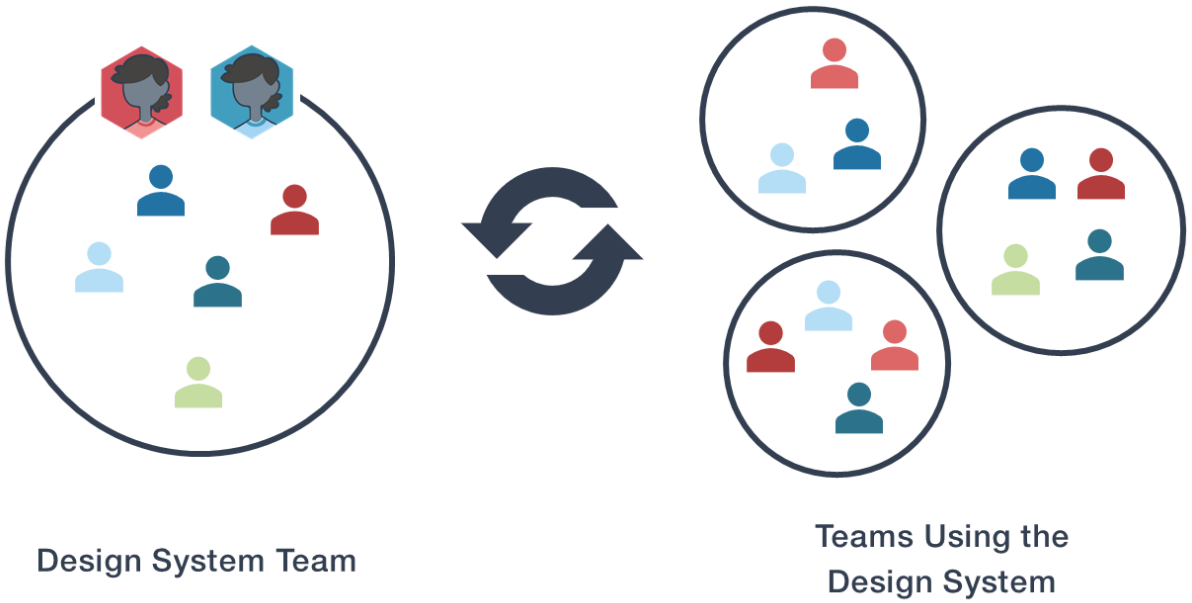


Figure 40. An illustration of the suggested team model, with a permanent product owner and technical lead that has rotating members from teams with an interest in the design system. Authors own image.

The two roles of product owner and a technical lead should be constant within their roles. However, the supporting team around them working with the design system can rotate. There are several teams at VGR that currently uses the existing code repository and the demonstrative website. The teams using them either develops the content of them or does frequently use them and does therefore have an interest in its development. As said, this happen over multiple teams, and therefore the teams with interest in having a design system should help with developing it for the better. It could then be possible to have a rotation with people that in intervals helps with both maintaining and developing the system. This helps with getting continuous insights from each interested team, helping with ensuring that the different teams' demands and wishes are considered during development. This should create less biased views on the content which can help several teams, and not just benefit one. It should as well with keeping the components and patterns general, helping as many users as possible.

The members of the design system teams are supposed to work on their respective products beside being on the design system team. However, the design system team should dedicate some of their time to complete design system work.

25.2 CONTRIBUTIONS

When someone outside of the responsible team of the design system comes with suggestions of what to add to the system, this should first come to the product owner, which then decides if it passes the criteria's described under the contribution section of the design system. After the approval from the product owner, it should be taken up into consideration with the rest of the team, discussing if it is worth adding to the system. At this point, there should be some feedback to the person sending in the suggestion, of whether the suggestion will be taken into consideration or not.

If the suggestion gets accepted, there needs to be documentation written about it and it needs to be added to the correct place in the system. This should be done by the team members working with the system at that time. But before it gets added, the code for the component need to be written and the functionality of the contribution needs to be tested within the test site before it can be added into the design system website and code repository.

If rejected, the person sending in the suggestion needs to get feedback on why the suggestion was not accepted. However, the feedback e-mail should still encourage the one sending in the suggestion to still contribute and come with further suggestions.

This means that a contribution suggestion needs to be approved by both the product owner as the design system team.

25.3 STANDARDS & GUIDING PRINCIPLES

The design system should aim to follow a set of standards and guiding principles. These should match the organisations goals and be guiding when deciding about the design systems future and prioritisations. One of these guiding principles could be to create accessible applications in line with accessibility laws from the EU directive, which leads to the standard to follow the WCAG. There could be other guiding principles as well and their purpose is to have a unified view of what the design system aims to accomplish.

Other standards that are more hands on is that of code standards. Developers should have a code standard so that the code is structured in a certain way throughout the whole system. This way, users that are developers, that uses the code, should get the same type of structure throughout and be able to collaborate and understand each other's code better. This information about code standards is best stored on the GitHub page's wiki as it is there the code repository should be hosted. There should be information on how to access these standards on the design system website.

When writing and adding documentation to the design system, there should be certain standards that are followed. To add a component or pattern documentation, the suggested guidelines should be followed, to ensure that coherent content gets added.

There should also be set up how to work with the design system, to decide this and follow it when working. The design system should be a thing that becomes actively worked with, a living process to improve development.

25.4 ADAPTION

To adopt the design system there needs to be a low barrier to start using it. The design system can help the organisation with onboarding processes. However, the user needs to know that the design system exists if they are going to use it. The design system can gain benefits from becoming more visible and be shown to more than just the teams working and knowing about it. Information about the development of the design system, its benefits for the teams using it, and how it works can be set up on bulletin boards or even the intranet of VGR. That way the

word may be spread, and more potential users of the system could be found. Any way of spreading the knowledge about having a design system should increase the adaption and trust in the design system. Although the design system is said to mainly be focused to enable for development of remuneration systems, the design system needs to be visible to be able to grow in the future.

To ensure that the design system is adapted at VGR the system needs to become a part of their everyday work. There could be open meetings held, where all the different users and stakeholders can come and listen to what is going on. If people can come and discuss, they could possibly feel more like of a part of it. That way the system could become more central and a part of the organisation, or at least for some certain products.

The design system should be seen as something holistic, and as it grows it is important to think about is that it should not loose perspective of what it is good for and how it helps. It is also important to notice that if the design system gains traction there will be a need for new users to get help from the design system team and the team needs to devote time for helping them.

25.5 OPEN SOURCE

This concept is based on it being open source. Both the source code and its code repository for the design system website, components and the test site should be available as open source code. It is an approach to make the design system being more visible. Being visible is an important part in making the design system get the attention and funding it needs. This to be able to stay up to date, or even grow and become of more importance for other parts of the organisation. Having it accessible and open should reduce the friction working with the system, while also being able to use the tools the open source hosting site, GitHub, offers.

As there should not be any hindrance for others to view the design system website it is preferable for it to be hosted in a way that makes it public. It enables developers and designers to work with it away from office, but it also sends a message of transparency. It should become easier to adopt the design system if the design system website is available in a quick and easy way.

26 SUMMARY DELIVER

The final outcome of this project is a description of how the new design system should be structured, maintained, and what information it should contain. The final delivery is guidelines of what the design system should be like and what the work behind it should be to ensure a better coherence and efficiency. These are made into wireframes suggesting what it could look like.

The most important thing about the new design system website is that is should separate the part of demonstrating components from the documentation about them, ensuring verified, working components on the design system website. The code repository for the components should also become more stripped, only containing the components, not text or other types of information, easing the implementation and use of it.

The design system website should no longer look like a remuneration system, instead is should become a collection of documentation and information rather than a demonstration website. The website with the design system should become more divided and structured to understand the different areas of information that it contains. The side menu has the menu choices, from top to bottom, of *"Getting started"*, *"Style guide"*, *"Components"*, *"Patterns"*, *"Updates"* and *"Collaboration"*. This structure helps someone new to the design system website to understand the different ingoing parts of it, whilst someone experienced quickly can go to the information they want. By having areas of the design system showing what has changed within the system, what is going on, and how to get help and how to contribute, the design system opens the process to become more transparent. Starting to show that it is not static but a living and everchanging thing.

There is also a dedicated start page that sells the concept of the design system in favour for selling the benefits of it to potential users. But also, each area within the design system, the different menu choices, should have its own dedicated start pages to better explain to the users what that area implicates and what documentation it contains.

The getting started information is divided into organisation, designer and developer, where the organisation section sells in the benefits and argues for why using the design system is preferable. The designer and developer pages are about helping them to get started with the system. The designer information is more tips about in which way the different information on the website can aid them. For the developers, the information is more linear, from how to set up a new project to how to add components from the code repository to existing projects.

The documentation about the components should be changed from the original documentations name of *"komponentkartan demo"*, since it was found complicated that it got compared to a map. Instead it should have a collective name that can describe the design systems ingoing parts in an easier, clearer way.

The name of the components should become more prominent, and its documentation should have a short description to what a specific component is. Belonging to that introductory part of the documentation, the code selector component name should be displayed, with a version number of that specific component, but also information about if the component changes with different themes. The default start when entering a component should be *"Example"* showing interactive examples of the component, and its belonging code. The next tab should be *"Design"*, describing why the component looks the way it does describing when to use the different variants. Thereafter, the tab *"API"* will follow. It will have a short description of the flexibility-options of the components, but there should also be more thorough information about the different flexibilities, written so that any user could understand it. The last tab, *"Accessibility"* should describe how the component is accessible and how it follows WCAG.

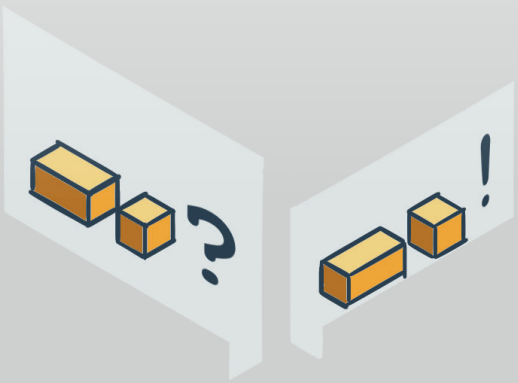
The new design system website has patterns introduced. The documentation about them should exist to explain user patterns and present important findings found from user research. It should also have an introductory part, explaining what the pattern is and when it could be used. Thereafter there should be two tabs; *"Example"*, displaying what the pattern looks like, and *"Guidelines"*, describing when to use the patterns, if it has been user tested and what to do and not with the pattern.

The *"Updates"* and *"Collaboration"* are to open up the process of the design system to the users, helping them understand both how the system has evolved and how it is planned to be developed. It will also include how to collaborate with the system, helping it to evolve further.

But besides from what the design system website should look like, the process of working with the design system should be more dedicated, having both a product owner with a technical lead together with a rotating team of designers and developers that helps them to develop the design system. The rotating team should be users within the teams using the system, not making it biased to one team, instead evolving with all of those using it. The design system should also become more open to other teams not working with it. It will serve to make everyone feel a kind of belonging to the system, so that it becomes an established part of the development work at VGR.

Discussion

This part of the report discusses the project, its process and outcomes. This section does as well discuss suggestions for future work.



27 PROJECT PROCESS

The process of the project followed the double diamond process, where the work diverged and converged in two turns. In the earlier phases it was very important to understand what a design system was, since this was considered very difficult to initially get a grip of. A lot of time was spent on understanding what makes a design system successful and beneficial, and the research was important to ensure that the work with the design system was made in a proper way. The words *"Design"* and *"System"* are two difficult, big words, which comes with certain expectations and prejudices. Therefore, it was difficult at the start of the project to define what a design system was. However, it was proven to be necessary to spend the time on it as there was no set definition of a design system. In the long run of the project it was beneficial for the project to make a self-defined definition, to draw up which parts of a design system that exists and how they are structured. To then also stick to and follow that definition throughout the rest of the project helped a lot with not getting stuck between phases and concept creations.

It was very beneficial to the project to spend time on doing initial interviews observing and understanding the different parts that exists at VGR. There was more to the system than firstly understood. Therefore, the second interviews were spent getting help from the users to create an idea over how everything was connected, getting response on if they thought it was as messy as it appeared to be. The outcome from those interviews was that it actually was messy and confusingly structured and an unclear process, which was valuable to understand when concepts were made. The concepts were given a different purpose, with not just a polished front but also a better working process behind it. Without the thorough initial two rounds of user research it would have been a risk to get stuck on the looks of the demonstrative website and create a result that would be unattainable.

The developing phase of the project started with a workshop which helped with realisations about the work behind the design system. The discussion among users provided the different attitudes and more insights into the work behind the design system and how they were working as an organisation. For this project it has been prioritised that the design system should make the user's work more purposeful, so it was important to know it would improve the user's situation and have them discuss it. Information could have been lost by just talking to the users one by one, and the valuable discussions between users would have gone lost. It was identified as necessary for the solution to be sustainable over time and make help the users to reach their goals, but also to improve collaboration between users.

Even if a nice façade and conceptually utopian, amazing system is what the design system could be, it still needs to be able to work and be maintained. Therefore, a new outcome for the project became to decide and make suggestions about realistic implementations, also suggesting how the maintenance and development of the system could work as well.

Three iterations were made for the layout for the design system website, trying out different types of functions. In the first evaluation of five concepts there were a bit too many concepts as it was noted that the participants became tired and lost interest when they had around two concepts left to evaluate. If the evaluation was to be remade it would have been preferable to have fewer concepts or having the participant rest between evaluations. However, having a pause would increase the duration of the test, which would not be preferable as they did the evaluation instead of their appointed work. The result of the evaluation might have been

affected by the users' loss of interest in the end. However, the method of letting the users mark and comment on low fidelity prototypes with markers was a good way of evaluating it. This way, the users did not consider the concepts as ready, not intimidating them to give proper feedback. The red and green markers was also a good way for the users to in hindsight get a good grip over which concepts they liked the most, since there was a lot of green on those that they found the most positive. It was a good way for a user to remember what they liked and not when going through multiple concepts. Additionally, they were also able to go back and look at all the concepts at the same time.

For each iteration during the development phase, the concepts became of higher and higher fidelity and there were less and less options of concepts. The first round was to create a lot of ideas, understanding more about which features were appreciated and not, and as the iterations went on, more content could be added, providing information to what a final concept should contain. Preferably, one more iteration could have been done, but there was not a lot of time, and therefore time was put on creating a high-fidelity prototype during the last iteration to get proper feedback on the content. Another iteration would consider the concepts in more detail and preferable would have been to make a usability test of it.

Luckily, during the process of the project the design system was given an appointed product owner. By getting a product owner, several of the user's needs could be solved. In the end of the project it became possible to talk and discuss the findings with the product owner to ensure the suggestions could become considered and what it would take to make it realisable. It was a positive way to get valid feedback and understanding that the work put down will create value for the organisation. The product owner took the findings of the project in consideration and did later act upon them. Some changes that were made during the duration of the project was that the double site of *"komponentkartan demo"*, that was publicly available and of a faulty version, was put down. Other improvements were done to the design system process as they were given insight into how the different users experienced the design system. There was also start of a design system team, with inspiration taken from suggestions in this project.

This project related to software development and was a new way for the authors to think about product development. Software development makes it possible for products to change quickly and things will frequently be implemented or updated in the system. This project took place for six months, and during the time a lot of changes were made to the original demonstrative website and code repository. Therefore, different suggestions from the thesis project could be implemented during the duration of the project as well. In hindsight it has been important in this kind of project to constantly save pictures of what the current state looks like, since it may change every other week.

28 FINAL CONCEPT

The aim for the project was never to develop something new, since the existing demonstrative website "*komponentkartan demo"* had already gained users. As it already has gained traction it was seen as too much of an effort to suggest it to change completely, without knowing if the result would be better. The possibility of using a third-party tool to create and store new design system components was discussed. A tool like this would help store the components and code

and make it possible to build the components in there. But since the competence and initial way of working with the pattern library already existed it would be more beneficial for VGR IT to not be dependent on a third-party tool and instead maintaining their own components in their website and code repository.

It could be argued that using already existing, well-developed, components from another design system could be beneficial. There are design systems that are open to use, but the issue is that you will not have ownership nor any control over it. A design system like this is open to use and will be maintained by another company or organisation. It has similar issues like the third-party tool, and for VGR IT it was important that they could have control over their applications and design system.

The final concept has considered the workflows and mental models of the users and it has been of high importance that the design system fits with within their needs and process. There has been a constant consideration of what would be the best for the users compared to how hard it would be to adapt and maintain. Of course, there is no end to what is possible to do with a design system, but to make the process of a design system maintainable it needs to fit into the process of VGR IT as well. This project has put a lot of focus in getting the website of the design system to get the recognition and focus it needs to be the central tool in the design system. If the main focus with this project would have been on making an optimal, good looking system, the final concept may not have been possible to implement by VGR IT and would not have brought benefits to its users. Therefore, a top priority through the project have been to consider what is possible versus what the users optimally would want, this to make the development of the design system sustainable.

For this final concept, there has been a suggestion with guidelines regarding the process behind the system regarding team models. It is important to understand that what this project suggest is just that; a suggestion. The organisation themselves have to decide exactly what they want to do with that information. They have looked into the suggestion but there needs to be a discussion about a balance of what is reasonable to implement and follow when it comes to a design system team. There must be a model that works within the context. It has to be made sure that the team working with the design system do not get the double the amount of work just because they also are working with developing the design system. Unfortunately, this was not an area that the thesis could test or evaluate within its time frame. There also needs to be support from the organisation to let the users implement suggestions. Otherwise they will not have time to make use of the suggested improvements and just go back to the requested products since they do not have the resources to make the design system a reality. In all, it is necessary to evaluate the model to ensure social sustainability.

It must also be realised that if more users start to use the design system, the more help they will need from the design system team. The progress of the design system has to be made in a sustainable way and the limitations of the design system team must be recognised, especially if they have dual responsibilities both for the design system and their own product. Once again, it will need the recognition and resources to be able to cope with an increasing number of users and products. If it does not get the resources it needs the design system team will spend more time aiding other users than developing the design system, which will risk the design system becoming obsolete.

There is a suggestion about how to write and use common language for the documentation on the suggested design system website. Recommended is that the design system website should choose one language for the components, either Swedish or English, in the way they are named and documented. Currently, the code repository uses English names, but the concern language of VGR is actually Swedish. It therefore becomes quite confusing in the documentation when the component selector name for example can be *"button"* whilst it in the documentation gets referred to the Swedish name instead of the chosen name which then is the English one. Swedish however can be difficult to implement together with coding, since they tend to only use and follow the English letters and naming conventions. This makes it more difficult to combine with Swedish, since there are some special Swedish letters in the Swedish alphabet. Also, to translate component names into Swedish may become misguiding, since the English names are more established amongst developers. Which language that VGR IT wants to use is up to them. But to avoid confusion between developers and designers it is recommended that one language is used to name components. The name chosen should then be stuck to throughout the documentation.

The final concept aims to keep the design system open source. VGR IT is related to VGR which is a public organisation controlled by public decisions. It is an organisation funded by taxes and they should not have anything to hide except information concerning people's privacy. Transparency is favourable as it would be more ethical towards the taxpayers who are paying for it in the long run. A design system will require resources in the beginning, but once it's in place it will most likely bring benefits and save money, which ultimately will enable the taxpayers' money to be better used than to pay for the same work twice. It could also be beneficial for VGR to see the value in collaborative efforts. This since it will reduce silos and enable teams to share knowledge and work, even though it is difficult to administrate within the organisation.

As the design system collects front-end, common design patterns and guidelines, together with values of VGR, there should not be any big risks with having the system open source. It should rather be seen as an advantage keeping it open source, since it will enable users of it to access it when they are out of office. It will also show that VGR IT is advancing and work with development in a modern and open way, which also can help in attracting new employees. It can give more transparency into VGR's process, by openly showing it, which possibly could increase or create trust for the organisation.

The risk with having it open source is of course copy-cats. However, there is a rather low risk that anyone would use the design system to build their application, since there is a lot more to a VGR IT application than the user interface and its building blocks. There are other design systems out there that could be used instead that are more flexible to use than VGR IT's components which are optimised for a certain type of applications.

29 RECOMMENDATIONS FOR FUTURE WORK

The concept proposed in this project should not be seen as the finish line for the design system. It rather proposes a groundwork to continue building from, since a design system is not a static

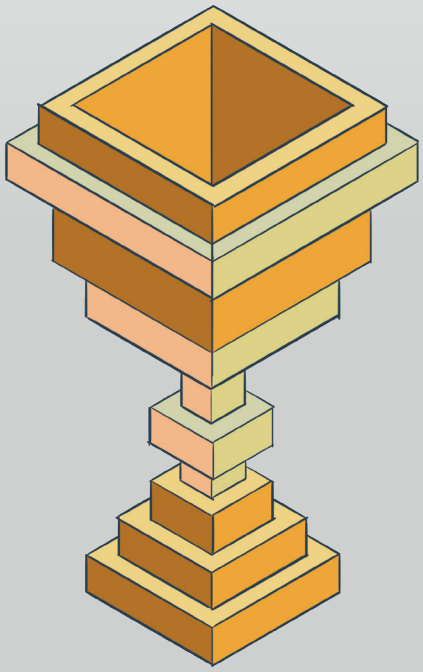
thing, but a product and a process that needs to evolve and develop next to the products created with the help of it.

This project has just touched the details and top of what a design system could enable. The demarcations of this project led it to be focused on adapting it to the remuneration systems and make the design system work for them and their users in the best way possible. To start with, the pattern library is an important part of getting ideas adapted and spread, so that the design system can last over time. Although it is necessary to start small with the details, there is a great potential with design systems when it becomes central for the organisation. Not only in getting the details right, but to have a collected and unified view of what the organisation does and how it is reaching its goals. Ultimately, a design system should permeate the whole organisation and bring together values and what goals the organisation aims to serve on a higher level. A topic for future work would be how the whole organisation could make use of a design system on a higher level and how a design system like that could be adapted.

If the design system is to be adapted at a larger scale and by more teams at VGR, there is a need to have it include responsivity for how the end users are using the web. There are an increasing number of devices that can access the web and it is no longer given that all the end users only use their desktop to do their work, or that the desktop is the best way for them to do their work. The design system is there to support the products made with it, so if there is a demand for a product available across different screen sizes the design system should aim to support it. At the start of this project it was already determined that it only would support desktop, however the question of responsivity will sooner or later have to be faced.

Currently, there also exists a style guide for the VGR public websites. The goal behind this is quite unclear, but a future possibility could be to put the two systems together, reaching the entire VGR. It is a style guide more focused on branding, which could benefit and complement the information found on the design system website. Branding is an important part of a design system, but it was not in the scope of this project since the design system designed was limited. Going forward there is a great potential to discover the possibilities of having a unified design system for all of VGR.

References



Atlassian. (2018). *Atlassian Design System*. Retrieved January 31, 2019, from Atlassian Design System: https://atlassian.design/

Australian Government. (n.d.). *Design System*. Retrieved January 31, 2019, from Australian Government Design System: https://designsystem.gov.au/

Chan, D., & Hayes, I. (2016). *Building Empowering Style Guides with Practical Research*. Retrieved March 29, 2019, from Clarityconf: https://www.clarityconf.com/session/building-empowering-style-guides-with-practical-research/

Christenssen, T. (2019). *What It Means to Be a Designer Who's Creative*. Retrieved March 20, 2019, from UX Collective: https://uxdesign.cc/what-it-means-to-be-a-designer-whos-creative-fc64cf8bd27c/

Curtis, N. (2015). *Team Models for Scaling a Design System*. Retrieved April 21, 2019, from Medium: https://medium.com/eightshapes-llc/team-models-for-scaling-a-design-system-2cf9d03be6a0/

Curtis, N. (2017). *Designing a Systems Team: Models and Lessons Learned to Scale a Team for an Enterprise*. Retrieved April 21, 2019, from Medium: https://medium.com/eightshapes-llc/designing-a-systems-team-d22f27a2d81d/

Curtis, N. (2018). *Documenting Components: Serve a System's Audiences with Well-Architected Content*. Retrieved January 23, 2019, from Medium: https://medium.com/eightshapes-llc/documenting-components-9fe59b80c015/

Design Council. (2015). *The Design Process: What Is the Double Diamond?* Retrieved January 24, 2019, from Design Council: https://www.designcouncil.org.uk/news-opinion/design-process-what-double-diamond/

Frost, B. (2016). *Atomic Design.* Brad Frost Web. Retrieved January 23, 2019, from http://atomicdesign.bradfrost.com/

FutureLearn. (2019). *Design System*. Retrieved January 31, 2019, from FutureLearn Design System: https://design-system.futurelearn.com/

GOV.UK. (n.d.). *GOV.UK Design System*. Retrieved January 31, 2019, from GOV.UK Design System: https://design-system.service.gov.uk/

Guth, M. (2018). *The Value of Design Systems for Scalable Growth.* n.p.: Edenspiekermann.

Kholmatova, A. (2017). *Design Systems: A Practical Guide to Creating Design Languages for Digital Products.* Freiburg: Smashing Media AG.

Martin, B., & Hanington, B. (2012). *Universal Methods of Design.* Beverly, MA: Rockport Publishers.

Maylor, H. (2010). *Project Management* (4th ed ed.). Harlow, England : Pearson Education Limited.

Mazur, M. (2018). *What is Design Debt and Why You Should Treat It Seriously*. Retrieved February 4, 2019, from UX Collective: https://uxdesign.cc/what-is-design-debt-and-why-you-should-treat-it-seriously-4366d33d3c89/

Mounter, D. (2018). Design Better Podcast. *From Design Silos to Design System*. (A. Walter, & E. Woolery, Interviewers) InVisionApp.

Royal Literary Fund. (n.d.). *Essay Guide: Literature Reviews*. Retrieved March 27, 2019, from Royal Literary Fund: https://www.rlf.org.uk/resources/what-is-a-literature-review/

Schultz, A. (2019). *4 Takeaways From the Clarity Design Systems Conference*. Retrieved March 14, 2019, from Framer: https://blog.framer.com/4-takeaways-from-the-clarity-design-systems-conference-3b4ae777aa77/

Shevchuk, M. (2018). *Do You Need Design System in 2019?* Retrieved March 27, 2019, from Medium: https://medium.com/@TechMagic/do-you-need-design-system-in-2019-32fb697b2126/

Shopify. (n.d.). *Polaris*. Retrieved January 31, 2019, from Polaris: https://polaris.shopify.com/

Spool, J. (2011). *Riding the Magic Escalator of Acquired Knowledge*. Retrieved February 20, 2019, from uie: https://articles.uie.com/magic_escalator/

Suarez, M., Anne, J., Sylor-Miller, K., Mounter, D., & Stanfield, R. (n.d.). *Design Systems Handbook.* Retrieved February 5, 2019, from DesignBetter.co: https://www.designbetter.co/design-systems-handbook/

W3C. (2018). *Web Content Accessibility Guidelines (WCAG)*. Retrieved February 21, 2019, from W3C: https://www.w3.org/WAI/standards-guidelines/wcag/

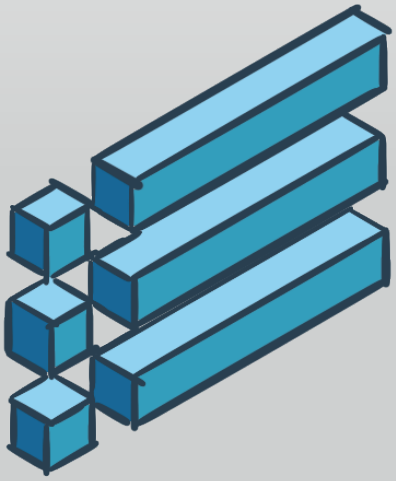
Vella, M. (2018). *Design Systems, and How Your Company Benefits from Them*. Retrieved April 23, 2019, from Modus: https://moduscreate.com/blog/design-systems-and-how-your-company-benefits-from-them/

Wikberg Nilsson, Å., Ericson, Å., & Törlind, P. (2015). *Design: Process och Metod.* Lund: Studentlitteratur.

Västra Götalandsregionen. (2018). *Organisation och Verksamhet*. Retrieved January 25, 2019, from Västra Götalandsregionen: https://www.vgregion.se/om-vgr/organisation-och-verksamhet/

Appendix

In this section is the appendix of the final concept in, presented in the wireframes created as one of the final deliverables of the project.



Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Designsystemets namn

Komponentkartan är ett designsystem som har till syfte att underlätta och kvalitetssäkra utvecklingen av webbaserade applikationer inom VGR IT. Genom att samla byggstenar och gemensamma riktlinjer skapar designsystemet en enhetlighet och förhindrar att arbete som redan gjorts repeteras.

ORGANISATION

Tillhör du organisationen?
Vill du veta mer om fördelarna med designsystemet och hur du kommer igång med det?

Kom igång!

DESIGNER

Är du designer med intresse i designsystemet?
Vill du lära dig mer om hur man kommer igång med designsystemet, och fördelarna man får av det?

Kom igång!

UTVECKLARE

Är du utvecklare med intresse i designsystemet?
Vill du lära dig mer om hur man kommer igång med designsystemet, och fördelarna man får av det?

Kom igång!

Kom igång

För att så snabbt som möjligt kunna förstå hur designsystemet är upplagt och hur det kan hjälpa just dig så finns denna **“Kom igång”** guide.

Denna guide är indelad i *Organisation*, *Designer* och *Utvecklare*.

- Organisationens kom igång guide förklarar hur designsystemet kan hjälpa organisationen på olika vis.
- Designerns kom igång guide förklarar hur man kan använda designsystemet inom sin roll, hur man kan använda informationen i systemet.
- Utvecklarens kom igång guide förklarar hur man kommer igång med systemet, vad man behöver installera och vad man bör tänka på..

Självklart är det möjligt att ta nytta från informationen på olika sidor, så se till att läsa in dig på vad designsystemet handlar om, och hur du skall göra för att komma igång och börja använda det!

Redigera text på GitHub

Kom igång

För att så snabbt som möjligt kunna förstå hur designsystemet är upplagt och hur det kan hjälpa just dig så finns denna **“Kom igång”** guide.

Denna guide är indelad i *Organisation*, *Designer* och *Utvecklare*.

- Organisationens kom igång guide förklarar hur designsystemet kan hjälpa organisationen på olika vis.
- Designerns kom igång guide förklarar hur man kan använda designsystemet inom sin roll, hur man kan använda informationen i systemet.
- Utvecklarens kom igång guide förklarar hur man kommer igång med systemet, vad man behöver installera och vad man bör tänka på..

Självklart är det möjligt att ta nytta från informationen på olika sidor, så se till att läsa in dig på vad designsystemet handlar om, och hur du skall göra för att komma igång och börja använda det!

Redigera text på GitHub

Designsystemets namn

Kom igång
Organisation
Designer
Utvecklare
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Kom igång - Organisation

VARFÖR?

Designsystemet hjälper till att skapa koherens mellan de produkter som skapas inom ersättningssystemen på VGR IT. Vissa användare kan tänkas använda mer än ett system, och är de inte uppbyggda eller fungerar på samma sätt kan en användare bli förvirrad och kanske till och med råka göra fel. Designsystemet erbjuder färdiga byggstenar som kan användas för att bygga upp system, vilket sparar tid på utvecklingen av det som är grunden, och gör det möjligt att lägga mer kraft på detaljer och att optimera systemen så bra som möjligt.

Som organisation kan det vara bra att använda designsystemet eftersom:

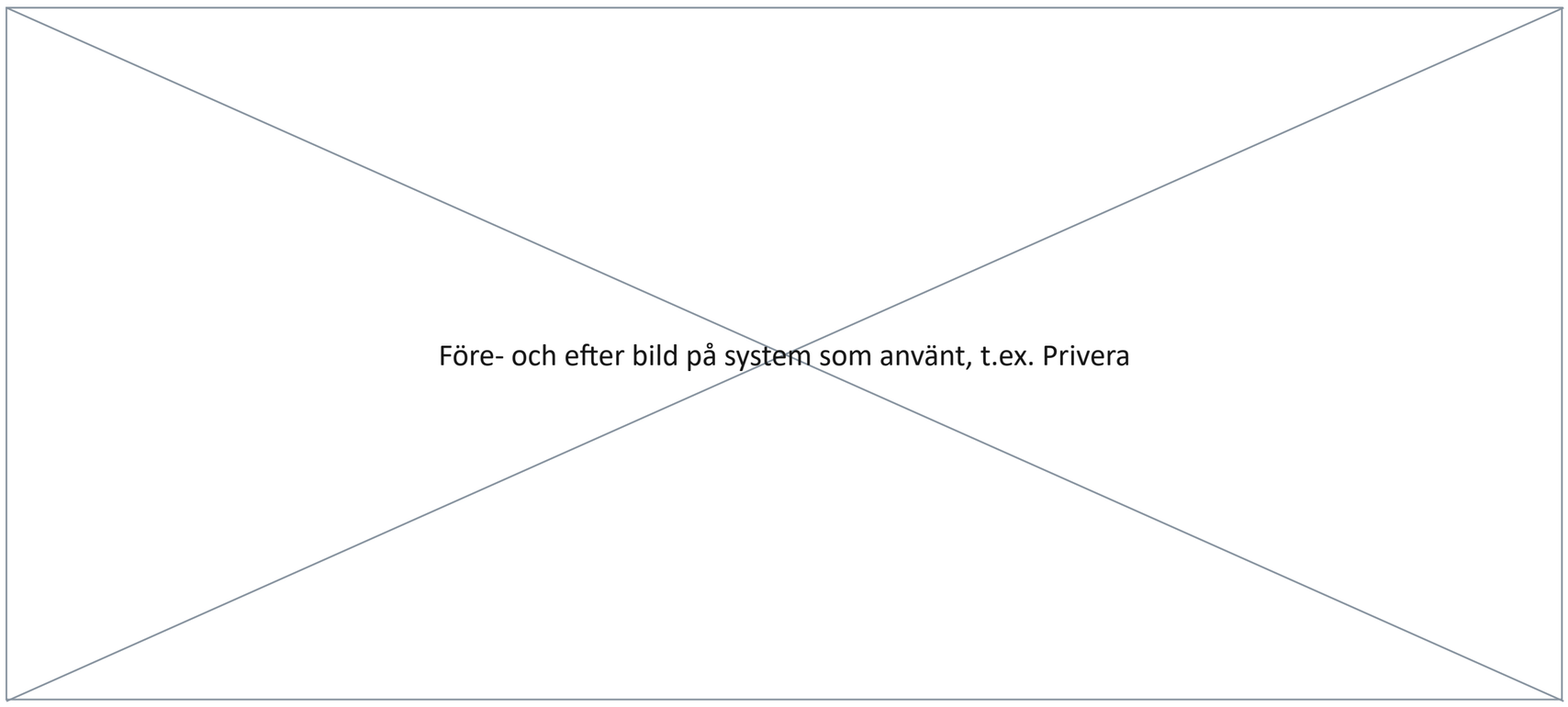
- Det effektiviserar utvecklingen av nya system.
- Det kan spara tid och pengar.
- Designers och utvecklare får ett bättre kommunikationsverktyg.
- Innehållet i designsystemet följer Web Content Accessibility Guidelines (WCAG) till en hög nivå, vilket underlättar inom offentliga sektorn för att enklare kunna följa och förstå tillgänglighetskraven.

Designsystemet är beräknat att ha hjälpt till med **X %** besparing, och tjänster har kunnat utvecklats snabbare och mer effektivt, vilket har gjort att slutprodukterna är av en hög kvalité.

EXEMPEL PÅ SYSTEM SOM ANVÄNT SIG AV DESIGNSYSTEMET

För att säkerhetsställa att introducerade komponenter och mönster fungerar som de är tänkta så finns det en testmiljö för detta. För att se mer hur komponenter interagerar tillsammans och kan skapa mönster, gå till testmiljön.

Är du nyfiken på existerande system och applikationer som har använt systemet kan du läsa hur de gjorde och hur det gick till här under.



Redigera text på GitHub

Designsystemets namn

Kom igång
Organisation
Designer
Utvecklare
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Kom igång - Designer

VARFÖR?

Som designer är designsystemet ett bra verktyg att använda sig av. Det underlättar vid kommunikation med utvecklare då man kan använda sig av de byggstenar som går att hämta från designsystemet, såsom komponenter och mönster, för att bygga UI's som utvecklarna sedan kan göra om till kod. De byggstenar som finns i designsystemet, komponenter och mönster, är anpassade efter **Web Content Accessibility Guidelines (WCAG)** 2.1 nivå AA (EN301549), vilket är ett lagkrav, och eftersom att VGR är en offentlig aktör är det extremt viktigt att följa detta. Tänk därför alltid på kontraster i det du skapar och hur du tänker att ditt innehåll skall fungera och agera. För att läsa mer om WCAG finns mer information att hitta på funka.com/lagar-och-regler/webbtillganglighetsdirektivet samt webbriktlinjer.se/wcag.

I denna kom-igång guide finns information och tips om hur du kan använda designsystemet. Se länkar under för att snabbt komma till den del som du är intresserad av!

↳ *Få tillgång till Sketch-biblioteket*
↳ *Använda stilguiden*
↳ *Använda komponenter*
↳ *Använda mönster*
↳ *Se exempel*

FÅ TILL GÅNG TILL SKETCH-BIBLIOTEKET

Det finns ett Sketch-bibliotek där innehållet i komponentbiblioteket finns i vektorgrafisk form. Detta kan vara bra att ha så att du slipper skapa alla komponenter från början när de redan finns tillgängliga. Sedan är det bara att infoga lämplig komponent när det är lämpligt under tiden du designar ditt system. De komponenter som finns i systemet uppdateras kontinuerligt, för att se till att du har nyaste versionen av en komponent, se till att använda nyaste versionen av Sketch-biblioteket och jämför med hur komponenterna ser ut i designsystemet.

För att få tillgång till Sketch-biblioteket, kontakta ansvarigdesigner@vgregion.se.

Om du är ny till Sketch finns det bra guider om hur man kommer igång på deras hemsida: sketch.com/docs.

ANVÄNDA STILGUIDEN

Stilguiden hjälper till om du måste designa utöver de komponenter som finns i biblioteket, och hjälper till vid utformningen av dessa. De komponenter som finns i designsystemet är redan utformade till att följa riktlinjerna i stilguiden. Stilguiden är till för att få till en enad front genom hela systemet.

Efter att du har fått tillgång till Sketch-filen och vill börja använda den så är det sedan möjligt att även infoga och använda de ikoner, typografi och färger som beskrivs under Stilguiden i designsystemet, vilket går att hitta i menyn till vänster.

De ikoner som används i designsystemet är från gratisbiblioteket hos Font Awesome, se mer om ikonbiblioteket på fontawesome.com/icons?d=gallery&m=free.

ANVÄNDA KOMPONENTER

Komponenter är återanvändningsbara byggstenar i ett användargränssnitt, och kan till exempel vara knappar och listor. I detta systemet är komponenterna ordnade efter bokstavsordning för att få en översikt på de komponenter som finns.

Informationen på varje komponent-sida innefattar olika flikar som man kan bläddra mellan.

En flik är exempel, som visar upp och demonstrerar hur en komponent beter sig vid interaktion. Dessa är oftast interagerbara och går att trycka på för att se hur de fungerar, vilket kan vara bra vid design då det visar hur komponenten beter sig.

En annan flik är vad som kallas API-beskrivning och är en specifikation på vad som är möjligt att göra med en komponent, vilken flexibilitet den har. Här kan det vara bra att titta om du vill förstå hur en komponent agerar och möjligen kan göra.

Det finns också information om hur man använder komponenten, kallad **"design"** som förklarar på vilka olika sätt man borde och inte borde använda komponenten, för att hjälpa till med hur man skall designa med komponenten. Man skall tänka på att komponenten används i sammanhang, så därför måste det större sammanhanget tas i beaktning vid användning.

ANVÄNDA KOMPONENTER

Komponenter är återanvändningsbara byggstenar i ett användargränssnitt, och kan till exempel vara knappar och listor. I detta systemet är komponenterna ordnade efter bokstavsordning för att få en översikt på de komponenter som finns.

Informationen på varje komponent-sida innefattar olika flikar som man kan bläddra mellan.

En flik är exempel, som visar upp och demonstrerar hur en komponent beter sig vid interaktion. Dessa är oftast interagerbara och går att trycka på för att se hur de fungerar, vilket kan vara bra vid design då det visar hur komponenten beter sig.

En annan flik är vad som kallas API-beskrivning och är en specifikation på vad som är möjligt att göra med en komponent, vilken flexibilitet den har. Här kan det vara bra att titta om du vill förstå hur en komponent agerar och möjligen kan göra.

Det finns också information om hur man använder komponenten, kallad **"design"** som förklarar på vilka olika sätt man borde och inte borde använda komponenten, för att hjälpa till med hur man skall designa med komponenten. Man skall tänka på att komponenten används i sammanhang, så därför måste det större sammanhanget tas i beaktning vid användning.

SE EXEMPEL

Vill du kunna se större exempel om hur man skulle kunna lösa upplägg på hur olika sidor ser ut kan du gå och titta i testmiljön där alla komponenter och mönster testas innan implementering. Genom att trycka omkring och förstå hur komponenterna fungerar i sammanhang blir det möjligt att skapa en större förståelse kring dem.

YTTERLIGARE FRÅGOR

Har du ytterligare frågor kan du gå in på sidan Kontakt som finns under Samarbete i designsystemet där mer information finns om hur man kan bidra till systemet, men också vem man kan kontakta gällande vissa ärenden.

Kom igång - Utvecklare

VARFÖR?

Som utvecklare är designsystemet ett bra verktyg att använda sig av. Det underlättar vid kommunikation med designers, då det som de skapar i ett UI enklare kan översättas till kod. Att använda ett designsystem för att utveckla underlättar när man skapar grunden för en ny applikation eller system, och visar även upp flexibiliteten i de existerande komponenterna.

För att kunna komma igång som utvecklare finns det vissa saker som man kan behöva tänka på för att komma igång med att utnyttja och använda designsystemet till sin fulla potential. Denna komma igång guiden innehåller information om hur du startar upp ett nytt projekt och hur du använder materialet från designsystemet. Se länkar under för att snabbt komma till den del som du är intresserad av!

FIXA UTVECKLINGSMILJÖN FÖR ATT ANVÄNDA ANGULAR OCH ÄVEN DESIGNSYSTEMET

Designsystemet använder sig av Angular-komponenter så därför krävs det att vissa saker installeras. Har du inte använt dig av Angular innan finns det en tutorial man kan göra på angular.io/tutorial. Följ denna process vid installering för att skapa utvecklingsmiljön:

1. Börja med att installera Node och verifiera att node och npm är installerade genom att köra `node-v` samt `npm-v` i en kommandoprompt. Det är även rekommenderat att installera IDE Visual Studio Code på code.visualstudio.com.

2. Installera sedan angular-cli globalt på din dator: `npm i -g @angular/cli` genom länken cli.angular.io.

3. Verifiera installationen genom att köra `ng -v` .

SKAPA ETT NYTT PROJEKT

För att skapa ett nytt projekt och använda dig av designsystemet, följ då dessa steg:

1. Skapa ett nytt projekt genom `ng new ProjectName` där “ProjectName” är namnet på ditt projekt.

2. För att lägga till designsystemet i det nya projektet, måste du ställa dig i projektets root på GitHub, vilket du hittar på designsystemets GitHub-sida.

3. Kör `ng s -o` för att köra igång din applikation lokalt och öppna i ett webbläsarfönster.

4. Nu är det dags att börja bygga din applikation och alla komponenter som finns i designsystemet finns dokumenterade under fliken “Komponenter” i sidmenyn. Var noga med att din sidstruktur blir korrekt när du skapar nya sidor. Sidstrukturen kan du hitta under Stilguiden genom att trycka här på Sidstruktur.

BÖRJA ARBETE I ETT BEFINTLIGT PROJEKT

För att börja arbeta med designsystemet i ett befintligt projekt, följ då dessa steg:

1. Hämta hem kodrepot för design systemet från designsystemets GitHub-sida.

2. Ställ dig sedan i projektets root och kör `npm i` för att installera alla node-paket som projektet behöver.

3. Kör sedan `ng s -o` för att köra igång din applikation lokalt och öppna i ett webbläsarfönster.

TESTMILJÖ

För att se till att komponenterna och mönstren är testade och fungerande så testas de i en testmiljö. I testmiljön är det möjligt att se hur kod kan interagera och lösas på olika vis. För att få inspiration om hur du kan lösa saker i applikationen du jobbar med kan du även titta på källkoden till testmiljön.

SAKER ATT TÄNKA PÅ VID UTVECKLING

Detta designsystem har komponenter som är anpassade efter Web Content Accessibility Guidelines (WCAG) 2.1 nivå AA (EN301549) eftersom VGR är en offentlig aktör. Se därför till att det du utvecklar utöver och tillsammans med kod från designsystemet likväl är anpassat efter WCAG. Mer information går att hitta på funka.com/lagar-och-regler/webbtillganglighetsdirektivet samt webbriktlinjer.se/wcag.

Vid utvecklingen kan det vara bra att ha följande verktyg:

- Extensions till Visual Studio Code - *eftersom det hjälper på detta vis.*
- Beautify - *eftersom det hjälper på detta vis*.
- Debugger for Chrome - *eftersom det hjälper på detta vis*.
- TSLint - *eftersom det hjälper på detta vis*.
- SCSS Intellisence - *eftersom det hjälper på detta vis*.
- Wallaby.js - *eftersom det hjälper på detta vis*.

YTTERLIGARE FRÅGOR

Har du ytterligare frågor kan du gå in på sidan Kontakt som finns under Samarbete i designsystemet där mer information finns om hur man kan bidra till systemet, men också vem man kan kontakta gällande vissa ärenden.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Stilguide

Stilguiden är dokumentation och information om hur applikationer och system skapades med designsystemet skall se ut och agera.

Det som finns i stilguiden är statiskt, det vill säga informationen däri gäller konstant genom system och är oftast svår att ändra. Den hjälper till att göra att även det som skapas utöver de återanvändningsbara komponenterna skall kännas och se ut, men även efterlikna, VGR’s designspråk.

Stilguiden innehåller bestämmelser om hur saker faktiskt skall vara som tillexempel vilket typsnitt som skall användas, vilka färger som används och när, men även information om hur siduppläggg skall vara.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Färgkarta
Ikoner
Sidhantering
Temahantering
Typografi
Upplösning
Komponenter
Mönster
Uppdateringar
Samarbeta

Stilguide

Stilguiden är dokumentation och information om hur applikationer och system skapades med designsystemet skall se ut och agera.

Det som finns i stilguiden är statiskt, det vill säga informationen däri gäller konstant genom system och är oftast svår att ändra. Den hjälper till att göra att även det som skapas utöver de återanvändningsbara komponenterna skall kännas och se ut, men även efterlikna, VGR’s designspråk.

Stilguiden innehåller bestämmelser om hur saker faktiskt skall vara som tillexempel vilket typsnitt som skall användas, vilka färger som används och när, men även information om hur siduppläggg skall vara.

Redigera sidan på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Komponenter

Komponenter är återanvändningsbara byggstenar av ett användargränssnitt som kan användas till ett antal olika applikationer. På varje komponentsida finns det dokumentation om vad komponenten är, hur den ska användas, vilket API som finns tillgängligt och exempel på hur den kan implementeras.

Komponenterna som finns i komponentkartan stävar efter att vara generella och återanvändningsbara för ersättningsapplikationer. Om något inte fungerar, eller om du har ett förbättringsförslag, finns det information under Samarbeta hur du kan gå till väga.

Redigera text på GitHub

Komponenter

Komponenter är återanvändningsbara byggstenar av ett användargränssnitt som kan användas till ett antal olika applikationer. På varje komponentsida finns det dokumentation om vad komponenten är, hur den ska användas, vilket API som finns tillgängligt och exempel på hur den kan implementeras.

Komponenterna som finns i komponentkartan stävar efter att vara generella och återanvändningsbara för ersättningsapplikationer. Om något inte fungerar, eller om du har ett förbättringsförslag, finns det information under Samarbeta hur du kan gå till väga.

Redigera text på GitHub

Button

<vgr-button>
Senast ändrad i version 3.9.0
Ändras med tema: Ja

En button är en klickbar knapp. Button finns i tre olika utföranden: Primary, secondary och discreet. Under design finns information om variant av Button som ska användas och hur.

Exempel | **Design** | **API** | **Tillgänglighet**

Primary button

OK Aktiv Inaktiv

```
<vgr-button (click)="lastButtonPressed = 'OK - primary'">OK</vgr-button>
<vgr-button (click)="buttonDisabled=!buttonDisabled;">{{buttonDisabled ? 'Aktivera' : 'Inaktivera'}}</vgr-button>
<vgr-button [disabled]="buttonDisabled">{{buttonDisabled ? 'Inaktiv' : 'Aktiv'}}</vgr-button>
```

Senaste knapptryck:

Secondary button

OK OK Inaktiv

```
<vgr-button [buttonStyle]="'secondary'" (click)="lastButtonPressed = 'OK = secondary'">OK</vgr-
button>
<vgr-button [buttonStyle]="'secondary'" (click)="buttonSecondaryDisabled=!
buttonSecondaryDisabled">{{buttonSecondaryDisabled ? 'Aktivera' : 'Inaktivera'}}</vgr-button>
<vgr-button
[buttonStyle]="'secondary'" [disabled]="buttonSecondaryDisabled">{{buttonSecondaryDisabled ?
          'Inaktiv' : 'Aktiv'}}</vgr-button>
```

Senaste knapptryck:

Discreet button

OK Aktivera Inaktiv

```
<vgr-button [buttonStyle]="'discreet'" (click)="lastButtonPressed = 'OK - discreet'">OK</vgr-button
<vgr-button [buttonStyle]="'discreet'" (click)="buttonDiscreetDisabled=!buttonDiscreetDisabled">{{buttonDiscreetDisabled
        ? 'Aktivera' : 'Inaktivera'}}</vgr-button>
<vgr-button [buttonStyle]="'discreet'" [disabled]="buttonDiscreetDisabled">{{buttonDiscreetDisabled ? 'Inaktiv'
        : 'Aktiv'}}</vgr-button>
```

Senaste knapptryck:

Button

`<vgr-button>`
Senast ändrad i version 3.9.0
Ändras med tema: Ja

En button är en klickbar knapp. Button finns i tre olika utföranden: Primary, secondary och discreet. Under design finns information om variant av Button som ska användas och hur.

Exempel | **Design** | **API** | **Tillgänglighet**

En Button kommunicerar handlingar en användare kan ta. Det finns tre olika utföranden med olika Visual loudness och de ska användas på följande sätt:

Primary button

OK

Primary button symboliserar det troligaste valet en användare kommer att ta. På en sida är den högst i hierarkin och flera primary button ska inte placeras bredvid varandra.

Secondary button

OK

Secondary button symboliserar ett mindre troligt alternativ och är lägre i hierarkin än primary button.

Discreet button

OK

En dicreet button symboliserar också ett mindre troligt alternativ och är lägst i hierarkin.

Mönster som använder button

Avbryt eller spara

Redigera text på GitHub

Designsystemets namn
Kom igång
Stilguide
Komponenter
Action panel
Back to top
Button
Card
Checkbox
Datepicker
Dropdown
Filter tag
Input
List
Loader
Modal
Pagination
Radio group
Input

Button

<vgr-button>
Senast ändrad i version 3.9.0
Ändras med tema: Ja

En button är en klickbar knapp. Button finns i tre olika utföranden: Primary, secondary och discreet. Under design finns information om variant av Button som ska användas och hur.

Exempel **Design** **API** **Tillgänglighet**

Namn	Beskrivning	Exempel
buttonStyle : string	Ändra utseense. Möjliga värden är primary/secondary/discreet. Default är primary.	[buttonStyle] = "secondary"
disabled : boolean	Inaktivera knapp	[disabled]="true"
click = new EventEmitter	Event när knapp trycks ner (click) = "OnClick()"	(click) = "OnClick()"

buttonStyle

"primary" ger primary button, vilket är den kanpp som är ifylld. "secondary" ger secondary button, vilken har en med outline. "discreet" ger discreet button, vilken har text utan ram.

disabled

Kan vara "true" eller "false". "true" ger en knapp som inte går att klicka på och har ett blekt utseende.

click = new EventEmitter

Förklaring på vad det kan vara och hur det fungerar.

Button

`<vgr-button>`
Senast ändrad i version 3.9.0
Ändras med tema: Ja

En button är en klickbar knapp. Button finns i tre olika utföranden: Primary, secondary och discreet. Under design finns information om variant av Button som ska användas och hur.

Exempel **Design** **API** **Tillgänglighet**

<vgr-button> har en intern div vars roll är satt till button.

Förutom rollen så används attributen:

aria-disabled, om knappen är inaktiv eller aktiv

Vid knappaktivering flyttas fokus beroende på vilken åtgärd som knappen utför. Till exempel:

Om knappåtgärden öppnar en dialog, flyttas fokus till inuti dialogrutan.
Om knappåtgärden stänger en dialog, återgår fokus normalt till den knapp som öppnade dialogen, om inte funktionen som utförts i dialogrutan leder till ett annat element. Om du till exempel aktiverar en avbrytningsknapp i en dialogruta återgår fokus till knappen som öppnade dialogrutan. Om dialogrutan bekräftade åtgärden att radera den sida från vilken den öppnades skulle emellertid fokusen logiskt gå över till ett nytt element.
Om knappåtgärden inte avvisar det aktuella sammanhanget, är fokus normalt kvar på knappen efter aktivering, till exempel en 'Tillämpa'-knapp.
Om knappåtgärden indikerar ett kontextbyte, till exempel att gå till nästa steg i en guide eller lägga till ett annat sökkriterium, är det ofta lämpligt att flytta fokus till utgångspunkten för den åtgärden.
Om knappen aktiveras med ett kortkommando, förblir fokus normalt i det sammanhang som kortkommandot användes. Om till exempel Alt + U aktiverar en "Upp" -knapp som flyttar det aktuella fokuserade objektet i en lista en position högre upp i listan, kommer Alt + U när fokus är i listan inte att flytta fokus från listan.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Mönster

Mönster är vedertagna lösningar som är knutna till en specifik typ av användningsfall. För detta designsystem är mönstren främst funna inom ersättningssystem och informationsinsamling. Alla mönster innehåller beskrivningar på hur och när mönstret ska användas samt exempel.

Mönster kan innehålla en eller flera komponenter och beskriver hur de ska användas i sammanhanget av ett användningsfall.

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Validering
Avbryt eller spara
Filtrering
Mönster 4
Mönster 5
Mönster 6
Mönster 7
Mönster 8
Uppdateringar
Samarbeta

Mönster

Mönster är vedertagna lösningar som är knutna till en specifik typ av användningsfall. För detta designsystem är mönstren främst funna inom ersättningssystem och informationsinsamling. Alla mönster innehåller beskrivningar på hur och när mönstret ska användas samt exempel.

Mönster kan innehålla en eller flera komponenter och beskriver hur de ska användas i sammanhanget av ett användningsfall.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Validering
Avbryt eller spara
Filtrering
Mönster 4
Mönster 5
Mönster 6
Mönster 7
Mönster 8
Uppdateringar
Samarbeta

Avbryt eller spara

Avbryt eller spara är ett mönster som ger användaren möjlighet att spara eller avbryta vad de har ändrat. Om användaren väljer avbryt ska det vara tydligt vad som avbryts, lika så ska det vara tydligt vad som har sparats.

Exempel **Riktlinjer**

Avbryt Spara

Designsystemets namn

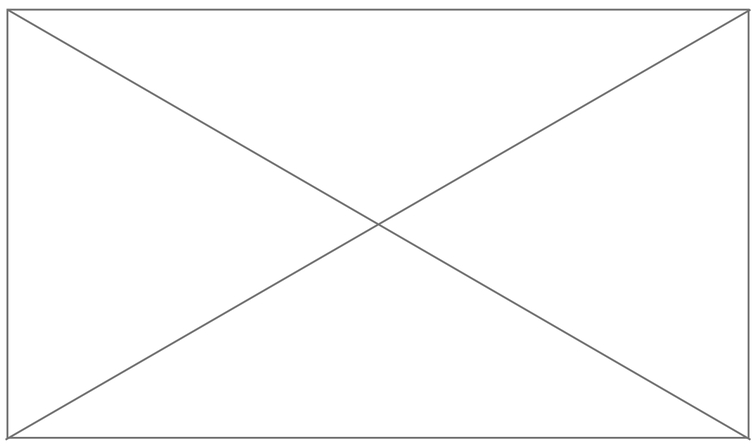
Kom igång
Stilguide
Komponenter
Mönster
Validering
Avbryt eller spara
Filtrering
Mönster 4
Mönster 5
Mönster 6
Mönster 7
Mönster 8
Uppdateringar
Samarbeta

Avbryt eller spara

Avbryt eller spara är ett mönster som ger användaren möjlighet att spara eller avbryta vad de har ändrat. Om användaren väljer avbryt ska det vara tydligt vad som avbryts, lika så ska det vara tydligt vad som har sparats.

Exempel **Riktlinjer**

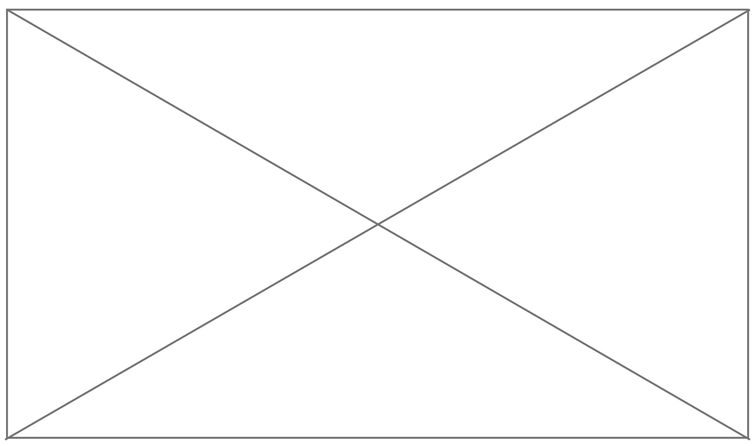
Spara och avbryt kan behövas på flera olika sätt och passa in i olika sammanhang. Vad dessa ska ha gemensamt är att de ska följa riktlinjer som är framtagna för att användaren ska känna igen sig hur denne avbryter och sparar.



Tänk på att:

Spara och avbryt ska alltid placeras i nedre högra hörnet
Guideline 2
Guideline 3

Det finns också andra sätt att göra avbryt eller spara på. Det går bra att använda en secondary button och en dicreete button i den här kombinationen, där spara är till höger och består av en primary button.



Tänk på att:

Spara och avbryt ska alltid placeras i nedre högra hörnet
Guideline 2
Guideline 3

Användningstester

I ett användningstest 2019-04-25 så utvärderades mönstret med resultatet:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Uppdateringar

I denna delen av systemet går det att hitta information om vad som antingen har hänt i systemet det senaste, men även vad som är planerat att hända framöver.

För att se vad som nyligen har hänt, finns changelogen, och för att se vad som är planerat framöver finns roadmapen.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Changelog
Roadmap
Samarbeta

Uppdateringar

I denna delen av systemet går det att hitta information om vad som antingen har hänt i systemet det senaste, men även vad som är planerat att hända framöver.

För att se vad som nyligen har hänt, finns changelogen, och för att se vad som är planerat framöver finns roadmapen.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Changelog
Roadmap
Samarbeta

Changelog

Här i changelogen finns information om vad som har hänt och är nytt i systemet.

SENASTE UPPDATERINGEN

Den senaste uppdateringen av designsystemet var 4.0.1 (beta) för vilken detta hände:

- Fixed a bug in List when multipleExpandedItems are not allowed and closing list-items did not emit any event.
- Fixed text for button select/deselect all in Dropdown multiselect when not bound to a form control.
- Modal background now correctly covers menus and headers.
- Input component: a bound form control is not changed on a DOM blur event.
- Fixing a bug in header menu where clicks get captured.
- Changed icons to use vgr-icons.
- Possible to prevent collapse of list-items in list.

För att se tidigare uppdateringar, följ denna länk till Changelogen på designsystemets GitHub sida.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Changelog
Roadmap
Samarbeta

Roadmap

Här i roadmapen står information om vad som är planerat att ändras i designsystemet framöver.

PLANERAT

Under 2019 har följande datum satts för release av major-versioner:

- Version 5: 22 maj 2019

Till version 5 planeras och uppskattas följande saker att kunna implementeras:

- Förbättringar i felmeddelande, eventuellt en egen komponent för felmeddelanden
- Deklarativ radio group
- Angular 8.0
- Rättning av buggar

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta

Samarbeta

I denna delen av systemet finns det information om samarbete, det vill säga vem man skall kontakta angående vissa ärenden, men också rapportera in problem eller föreslå ändringar till systemet. Skulle man vara mer intresserad över arbetet kring designsystemet kan man också se till att vara delaktig i de kanaler som komponentkartan finns.

Under Kontakt finns information om vem man kontaktar.

Under Rapportera finns information om hur man rapporterar in i systemet.

Under Vår kanaler finns information om hur man kan få veta mer om arbetet kring systemet.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta
Kontakt
Rapportera
Bidra
Våra kanaler

Samarbeta

I denna delen av systemet finns det information om samarbete, det vill säga vem man skall kontakta angående vissa ärenden, men också rapportera in problem eller föreslå ändringar till systemet. Skulle man vara mer intresserad över arbetet kring designsystemet kan man också se till att vara delaktig i de kanaler som komponentkartan finns.

Under Kontakt finns information om vem man kontaktar.

Under Rapportera finns information om hur man rapporterar in i systemet.

Under Vår kanaler finns information om hur man kan få veta mer om arbetet kring systemet.

Redigera text på GitHub

Designsystemets namn

Kom igång

Stilguide

Komponenter

Mönster

Uppdateringar

Samarbeta

Kontakt

Rapportera

Bidra

Våra kanaler

Kontakta

Om du behöver hjälp med något, eller har några funderingar kan du kontakta ansvarig person som jobbar med designsystemet.

Ägare Ägsson

Produktägare, kan bland annat kontaktas om du har funderingar över allmänna saker angående designsystemet.

produktagare@vgregion.se

Kod Kodsson

Ansvarig utvecklare, kan bland annat kontaktas om du har problem med kod och implementering av designsystemet i ditt projekt

kod@vgregion.se

Design Ersson

Ansvarig designer, kan bland annat kontaktas för att få insikt i hur utvecklingsarbetet går till, eller hur man skall använda sig av de riktlinjer som är dokumenterade i designsystemet

produktagare@vgregion.se

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta
Kontakt
Rapportera
Bidra
Våra kanaler

Rapportera

Här finns det information om hur du kan rapportera in problem.

RAPPORTERA IN PROBLEM

Om du har problem med något i designsystemet kan du alltid rapportera in detta under designsystemets GitHub sida under Issues. När du rapporterar in ett issue finns det viss information som måste fyllas i, se till att fylla i dessa så utförligt som du möjligen kan.

Redigera text på GitHub

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta
Kontakt
Rapportera
Bidra
Våra kanaler

Bidra

Här finns det information om hur du kan bidra och ändra i designsystemet.

REDIGERA INNEHÅLL

Det är uppmuntrat att redigera information på sidorna som finns i designsystemets websida. I varje sida finns det en möjlighet att redigera på GitHub längst ned. Om du har tillåtelse kommer ändringen direkt in, annars måste den godkännas.

Vill du ändra en existerande sida skall du tänka på att detta görs med mark-up, vilket innebär att rubriker, brödtext och länkar skrivs på olika sätt.

Rubriker: När det gäller rubriker så finns det olika nivåer, där `#Rubrik` är större än `##Rubrik` och sedan i nerfallande ordning.

Brödtext: Brödtext skrivs på normalt vis med vanliga tecken, men fet stil skrivs som `**fet text**` där texten inuti asteriskerna blir fet.

Länkar: Vill du skapa en länk skrivs detta med hakparenteser och en kopplad webb-länk till det, till exempel `[Länka till något](https://link.com/page)` där det innanför hakparenteser blir det som syns på sidan och det innanför parenteserna blir webblänken som man vidarebefordras till vid klick på texten.

Annat: Om du skulle vilja lägga till mer komplicerad typ av dokumentation eller information, lägg i så fall ett issue på

FÖRESLÅ NY KOMPONENT ELLER MÖNSTER

Har du en idé eller tanke angående en ny komponent eller mönster? Då finns det möjlighet att föreslå och rapportera in det till designsystemet. Dock finns det några krav som måste uppfyllas för att kunna lägga till en komponent eller mönster. I listan nere kan du bocka av och fylla i följande uppgifter att följa med ditt förslag.

□ *Komponenten/mönstret skall vara generell och kunna vara användbar över flera olika applikationer.*

□ *Komponenten/mönstret skall vara unik, så se till att en liknande inte finns innan du föreslår en ny. Tänk också på om komponenten skulle kunna vara en variant på en existerande komponent.*

□ *Skicka gärna med ett bildexempel eller liknande på komponenten/mönstret du tänker dig för att förtydliga din motivering.*

□ *Du skall kunna motivera varför som komponenten/mönstret är viktigt att ta upp i designsystemet.*

□ *Om tanken är att komponenten/mönstret skall ersätta en existerande skall du se till att också bevisa eller starkt motivera varför som ändringen skall ske.*

Detta förslag skickas in till designsystem@vgregion.se för att överses. För att veta att ditt förslag tas under beaktning får du ett svarsmail inom X-Y dagar. Du kan även hålla utkik på Uppdateringar om ditt förslag väljs att skall implementeras

Designsystemets namn

Kom igång
Stilguide
Komponenter
Mönster
Uppdateringar
Samarbeta
Kontakt
Rapportera
Våra kanaler

Våra kanaler

Här finns det information om de kanaler som designsystemet är vaksamma i, samt där det diskuterat kring det.

GITHUB

Vill du få mer ingående detalj om vad som händer i systemet, se backlog och changelog, är det möjligt att gå till GitHub-sidan för designsystemet. Där publiceras arbetet bakom och är en förlängning av den dokumentation som går att finna i designsystemet.

SLACK

Om du vill kunna ta del av informationen som delas under utvecklingen av designsystemet kan du ansluta dig till vår Slack-kanal. Detta kan du göra genom att skicka ett mail till designsystem@vgregion.se och be om tillgång till Slack-kanalen.

Redigera text på GitHub