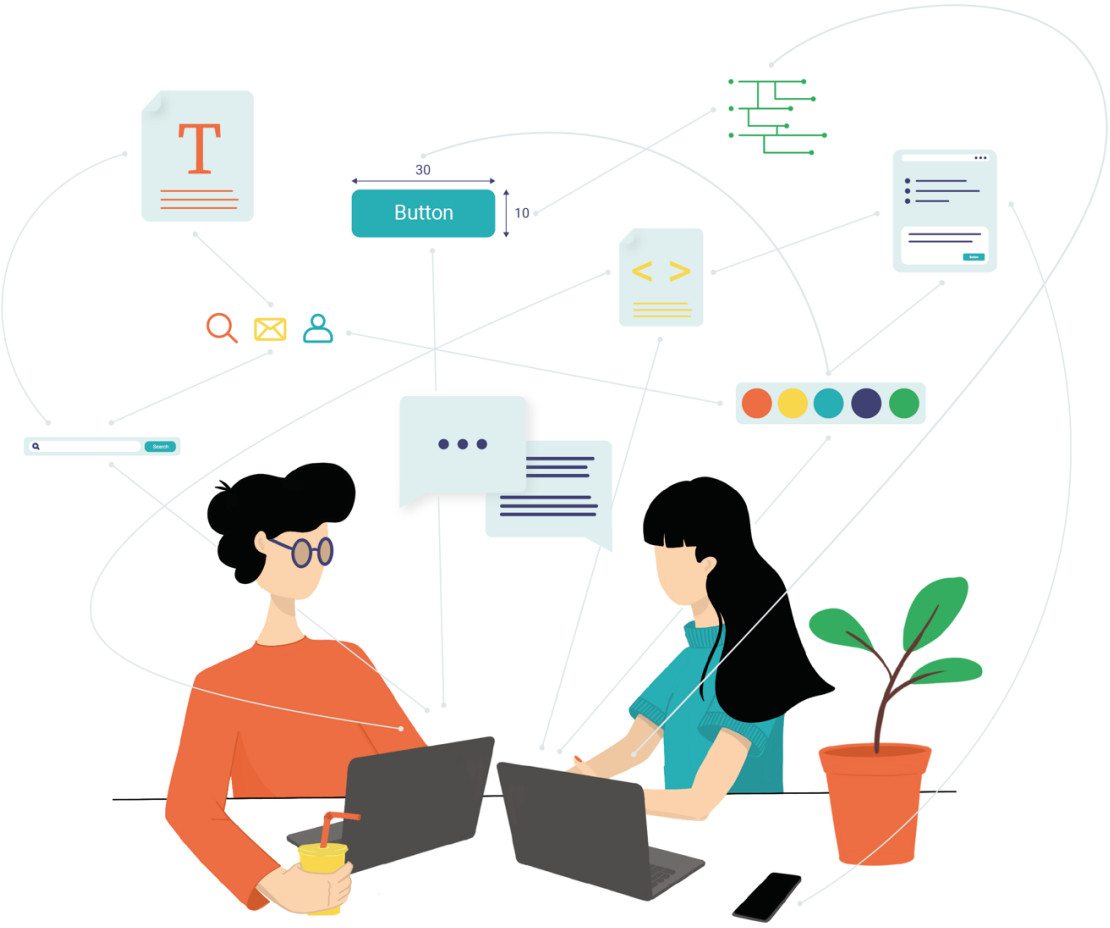
CHALMERS
UNIVERSITY OF TECHNOLOGY



Designing an Implementation Support Framework for Design Systems

A Constructive Design Research

Elin Davidsson & Theodor Hennings

Master's Thesis in Industrial Design Engineering

DEPARTMENT OF INDUSTRIAL AND MATERIAL SCIENCE
Division Design & Human Factors

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

Designing an Implementation Support Framework for Design Systems
A Constructive Design Research

Elin Davidsson
Theodor Hennings

Cover Illustration: Illustration of the concept of design systems
Cover made by Elin Davidsson

Chalmers University of Technology
SE 412-96 Gothenburg, Sweden
Print: Repro Service Chalmers

Gothenburg, 2020

Designing an Implementation Support Framework for Design Systems

A Constructive Design Research

Master of Science Thesis by

ELIN DAVIDSSON & THEODOR HENNINGS

Examiner & Supervisor: Bijan Aryana

Department of Industrial Material & Science - Design & Human Factors
Chalmers University of Technology - Gothenburg, 2020

Abstract

In this study, constructive design research has been used to investigate the implementation of design systems in organizations. Constructive design research is a research approach which employs the design process to gather insights and knowledge in a particular area.

A study has been conducted at the web-development company Universal Avenue. The aim was to identify the need of the designers and developers to suggest a design system concept. The aim was also to suggest a generalized framework for how to implement design systems in other organizations to meet the needs of designers and developers.

A design system can be defined as a collection of elements that will support the designers and developers in the development process. A design system is an open system which will need control mechanisms to stay relevant and up to date. The theory defines the content of a design system in more detail, as well as declaring how design systems could be designed and when it should be introduced in an organization. Literature about lean management in relation to design systems was included to support the construction of the design system framework.

The methods used in this study were benchmarking, observations, interviews, card sorting, co-creation workshops, and KJ analysis. These methods generated insights about design systems used in other organizations, as well as insights regarding the needs of designers and developers at Universal Avenue.

The field study resulted in an exposition of the current working process and design system at Universal Avenue. It also resulted in the creation of user profiles of the primary users of the design system. Based on the user's needs, guidelines for the creation of a design system were listed. The insights from the field study served as foundation for the concept creation of Universal Avenue's new design system, regarding what it should contain and how it should be implemented. The concept created for Universal Avenue, together with insights from the literature and the benchmarking resulted in a generalized implementation support framework for how to implement design systems in other organizations.

It has been concluded that the most prominent needs of designers and developers regarding the design system could be linked to the collaboration within and across team boundaries, as well as to the structure of the code. As a second result of this study, the main building blocks of a design system could be defined as design elements, guidelines, and component libraries. Finally, design systems should be implemented as an iterative process which supports the constant improvement and maintenance of the design system.

Key words: Design Systems, Constructive Design Research, Lean Management, Process Design.

Acknowledgements

First and foremost, we want to thank Universal Avenue for their warm welcome and continued support throughout this master thesis. We especially want to dedicate a thank you to Jovan Velkoski for supervising this project and for always being there whenever we had questions. Also, many thanks to the designers and developers at Universal Avenue that have contributed with their knowledge and input to our user studies, it has been a pleasure working together with you.

We also want to thank our supervisor and examiner at Chalmers, Bijan Aryana. His support in how to structure and perform this constructive design research have really helped us in tackling the complex domain of design systems in a great way.

Lastly, we want to thank Chalmers University of Technology for five incredibly fun and thoughtful years, now we are eager for new experiences.

Elin Davidsson and Theodor Hennings,
Gothenburg 2020.

Terminology

Branch - Using branches is a way to isolate development work so that it does not affect the main code directly. When work in a branch is done, it can be merged into the main repository.

Component - Is a grouping of elements that together create a functionality.

CSS - A code language used for describing the presentation of a document written in a markup language like HTML.

Design system - A design system is a collection of reusable elements, components and guidelines that aims to support the designers and developers in the process of building digital products.

Element - Smallest building block in the construction of graphical interfaces. Text, colors and icons are all examples of elements.

GitHub - An online service for code repositories that allows file sharing and version control.

HTML - Is the standard markup language for documents designed to be displayed in a web browser.

Kanban - Is an agile project management tool that helps visualize and manage workflows.

Library - An organized collection of components to be reused in web-development.

Ruby on Rails - A web-application framework that includes everything needed to create database-backend web applications.

README - A file located in directory that contains information about other files or the file structure.

Repository - A central location in which code is stored and managed.

React - A JavaScript framework for building user interfaces.

Sass - A preprocessor to CSS. Introduces variables and other features that CSS does not natively support.

Source of truth - A way of structuring programs so that everything is referencing the same source. A change in the source of truth propagates throughout a product.

Story - A story is used to describe a task that needs to be done by a product team, e.g. when implementing a new feature or when making an improvement in the existing product.

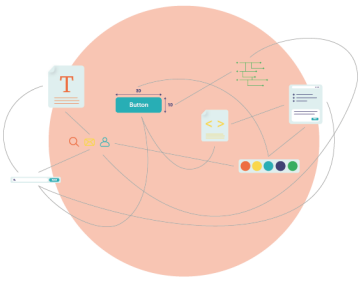
Token - Token is a named variable that can be shared between products. For example, spacing, opacity and time.

User journeys - A user journey is a series of steps which represent a scenario in which a user might interact with the designed product.

Table of Contents

01.
Introduction

In the introduction, the background to this study, and the company at which this study has been performed, will be presented. This will be followed by an introduction to the research approach that were used, the aim of this study, and the research questions which will be answered as well as the objectives. Lastly, the limitations of this study will be declared.



1.1 Background

Today, many organizations struggle with how to optimize the cooperation between designers and developers when building new digital products. Many attempts have been made to solve this problem, but few seems to be solving it perfectly. Recently, big companies like Atlassian and Salesforce have been creating *design systems* to decrease the gap between designers and developers and to speed up their development processes.

Design systems often contain rules, guides, design *elements* and *component* libraries that could be reused in web-development. The purpose of a design system is to create common language for designers and developers, to create a more streamlined and efficient working process. Which hopefully will contribute to more unified and user-friendly products for the end-user.

1.1.1 The Company and the Problem

Universal Avenue is a company developing a digital service for the business to business (B2B) sector. The service is providing solutions to other B2B companies and to small and medium sized enterprises (SME) that want to become more digital and grow their businesses. The platform allows salespersons to distribute to their networks, buyers to manage their hardware and, sales partners to connect sellers and buyers (Universal Avenue, 2020). Today's team consists of about 50 people located in both Gothenburg and Stockholm.

The product teams at Universal Avenue consists of five smaller teams, working on different parts of the product. In each team, there is one designer and at least two front-end and one back-end developer. Each team also consists of a product owner responsible for the progress of the team. The CTO is leading the whole product development process together with three other lead roles; Front-end lead, Back-end lead, and a Head of Design. Designers and developers work side by side in the development process and most of the designers have programming skills to decrease the gap between UI/UX and front-end.

At Universal Avenue, attempts have been made to create a design system to support their product development process. They currently have a component *library* to store coded components that can be reused while developing their products. This code library is meant to work as their *source of truth* and as a template for existing and future projects. However, there is a lack of structure for how to update and use this code library in their design process, and the different product teams have different approaches when using the component library which makes it inefficient and not as useful as it could be.

The designers at Universal Avenue uses a vector graphic editor called Sketch to create design mockups of the products. In Sketch, the designers have graphical libraries from which they reuse graphical components while designing. There is however no way to synchronize the code library and the graphical library automatically. The lack of automatic synchronization together with the complexity of the system has led to

differences between the components contained in the code library and the graphical library, a gap that will grow bigger as work continues.

1.2 Aim

The aim of this study is to use constructive design research, through a design project at a web development company, to develop a framework for how they could implement a design system that would be beneficial for designers and developers. The study aims to use the gained insights at the company to propose a general framework for how to best implement design systems in other organizations.

1.3 Research Questions

As described earlier in this chapter, many organizations are struggling with how to optimize the collaboration between designers and developers. The current situation at Universal Avenue confirms this problem, as there are deviations in the way their designers and developers are working in relation to the current design system. Hence, the questions to be answered in this study are;

- What are the designers' and developers' needs regarding a design system?
- What should the design system contain, and how should it be implemented to meet the needs of, and be advantageous for, both designers and developers in the development process?

1.4 Research Approach and Design Process

The approach used in this study is constructive design research which is referred to as a research approach where the design process or the construction of concepts holds an important role for gaining knowledge and insights (Koskinen,Zimmerman, Binder, Redstrom & Wensveen, 2011). In constructive design research, the building of concepts is a central part in the research, were the aim is to connect the designed concepts to the underlying theory (Wensveen, 2018). The efficiency and usefulness of a design system is highly influenced by the context in which it is used. As a result of high contextual bias, it was decided that the research would follow the field subcategory of the constructive design research that emphasizes contextualization of the research area (Koskinen et.al, 2011)

The design process used in this study is inspired by the double diamond framework for innovation (Design Council, 2020). The process is adapted to better meet the aim of this study and consists of four different phases; explore, define, iterate and generalize (figure 1). The explore-phase aims to generate a better understanding of design systems in general by reading about prior research and performing a benchmarking of already existing design systems. The define phase aims to investigate the current situation at the case company Universal Avenue and to define needs and problems in relation to design systems in an organizational context. That is followed by an iterative phase were a concept of the design system framework for the case company is to be developed. In the last phase, the insights and knowledge gained through the previous phases will be used to develop a more generalized framework for design system implementation.

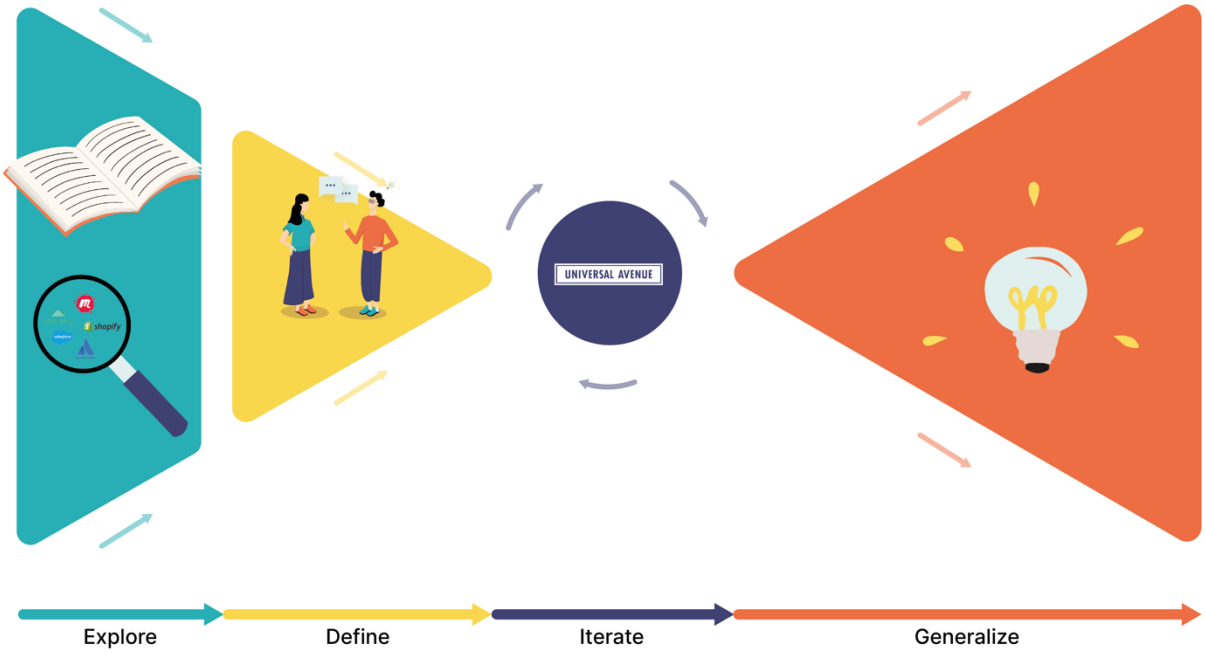


Figure 1. Overview of the design process

1.5 Objectives

- Read prior research on design systems to gain knowledge about its content and usage.
- Observe the collaboration between designers and developers at Universal Avenue to map their current working process.
- Interview designers and developers to map the current working process to find the pain points in the current design system.
- Develop a concept of a design system framework for Universal Avenue that would support the implementation.
- Iterate the concept of the design system framework to meet the needs of the designers and developers.
- Generalize the framework so that it can be used by other organizations aiming to implement a design system.

1.6 Limitations

The empirical part of the constructive design research has been limited to only investigate the current situation and the need of a design system in the context of one company. The company is divided into two main offices. Only one of the offices has been included in the user studies of this project.

The study only considered designers and developers even if there are secondary users and stakeholders that are likely to be affected by the implementation of a design system. The study was also limited to focus on the implementation of a design system and not the actual building of e.g. components and libraries.

Testing of the implementation support framework in practice had to be excluded due to limitations in time and will be left for future work. Halfway through the project, the Covid-19 situation, unfortunately, limited the accessibility to the company and the users, and the remaining co-creation sessions and evaluations were to be held remotely.

02.
Theory

The theory presents the most recent research on design systems and how they should be defined and implemented in companies to best facilitate their development. Both major and minor building blocks of design systems have been defined. There will also be theory about lean principles and how they relate to design systems.

2.1 Design Systems

The term design system has been a flourishing topic among UX designers and product developers in the last few years and more and more digital organizations are adopting the concept into their organizations (Gothelf & Seiden, 2016). On the web, one can find several open-source design systems were giants like Google, IBM and Uber do not hesitate to share the structure behind the way they are developing their websites and products to the public. But the traces of design systems can be linked to a movement that Bauhaus started over 100 years ago (Vesselov & Davis, 2019). Bauhaus shifted the focus from design as just decorative elements to focus on its function (Vesselov & Davis, 2019).

> *"Design systems are more than the latest fad; they are a natural evolution in the intersection between art, technology, and industry."*
>
> (Vesselov & Davis, 2019, p. 5)

Bauhaus created a unified design language supported by guidelines which later became an influence for the development of today's web and graphic design (Vesselov & Davis, 2019). As the response to the rise of the internet in the late 1900s, the popularity of computers increased considerably and so did also the interest and need for better web-design. This time was the starting point for the design systems as they are known today (Vesselov & Davis, 2019). The main purpose of a design system is to maintain consistency, reduce time and, increase collaboration between designers and developers (Churchill, 2019; Devanney, 2017). In summary, design systems are an attempt to systemize the web-development process in organizations.

2.1.1 Definition of Design Systems

To define what a design system is, the concept needs to be broken down. Systems thinking, which is a holistic framework for approaching complex things (Flood & Carson, 1993) have been used to define design systems. Flood and Carson (1993) describes a system as; "an assembly of elements related in an organized whole" (p. 7). The elements within a system are characterized by some interchangeable attributes, and the existence of the elements must be agreed upon by all users of the system in question. What further distinguish a system from another is how the elements within the system are related and what impact they have on each other. The elements can be connected through e.g. flows of materials, information or energy (Flood & Carson, 1993).

A system can be either closed or open, where a closed system has a clear boundary that separates it from any external environment. An open system, on the other hand, is affected by flows from the external environment (Flood & Carson, 1993). Lastly, systems also have a tendency to change due to entropy and external factors if there are no control mechanisms that preserve it from changing (Flood & Carson, 1993).

In contrast to the theoretical definition of systems, design systems can be defined by looking at how it is described and used by designers and developers in the field. Two different definitions of design systems are cited below. The first citation is from the

book Practical UI Patterns for Design Systems written by MacDonald (2019) who is an experienced product designer.

> *"A design system is a single source of truth for shared parts and processes, such as components, patterns, and guidelines, to build consistent products."*
>
> (MacDonald, 2019, p. 144)

The second citation is from the book Building Design Systems written by Vesselov, and Davis (2019), two UX designers with extensive experience within the field.

> *"A series of documented elements, components, and regions that include both design and front-end guidelines. The documentation contains live code examples, allowing cross-functional teams to easily reuse styles and components in several instances across an application. A design system also includes underlying design principles, rules, and guidelines that help a team build one or multiple products."*
>
> (Vesselov & Davis, 2019, p. 16)

Both definitions include the distribution and documentation of the design system's building blocks. They also give examples of how the documentation could be structured into e.g. patterns and guidelines. MacDonald (2019) describes the design system as a single source of truth, while Vesselov and Davis (2019) says that the design system will ease the process for cross-functional teams, both definitions refer to the aim of the design system to make the product development process more efficient. What also is recurrent among the two definitions is the consensus that there is more than one single way of structuring things when building a design system. A design system contains several building blocks, all contributing to the product development process by serving different stakeholders to develop uniformed and user-friendly products (figure 2).

Figure 2. Illustration of the concept of design system

Connecting those definitions to the system theory generates a greater understanding of which elements that exist in a design system and how they are connected to each other. These elements are building blocks such as component-libraries, patterns, and design principles, but also the actors such as designers and developers. The connections and the flow between elements in a design system are dominated by information and digital materials i.e. components and pictures. A design system can be considered as an open system since its content will be affected by external factors such as the customers.

Based on the theory above, this report will define a design system as follows;

- A design system is a collection of documented and informative elements together with consuming and constructive actors, such as designers and developers.
- The documented element aims to support the actors in their design and/or development process.
- The elements of a design system are connected through a constant exchange of information and material.
- A design system is an open system which is affected by external factors, such as the end-user.
- Each design system will need its own control mechanisms to stay relevant and up to date.

2.2 Building Blocks of a Design System

Design systems are too large and complex to efficiently be developed as one piece and are hence divided into several building blocks. These building blocks are made in a modular way in order to allow design systems to facilitate the development process (Frost, 2016). These major building blocks that help structure the design system often contain smaller building blocks that are used in the development of applications. In Atomic Design, Frost (2016) refers to the smaller building blocks as atoms and molecules, which are called *elements* and *components* in the design system defined by Vesselov and Davis (2019). This report will use the naming coined by the latter since elements and components have a wider adoption in the web development industry.

A summary of the different literature definitions of the building blocks of design systems has been illustrated in table 1. Building blocks located in the same row are each other's equivalents. In literature, elements, components, style guides, libraries, and guidelines are the terms used for defining the building blocks of a design system. These building blocks will be described in more depth later in this chapter.

Table 1. A comparison of the literatures' definitions of the building blocks of a design system.

Frost (2016)	MacDonald (2019)	Veselov & Davis (2019)
Brand identity	Brand guides, Visual style guides	Style guides
	UI libraries, Code libraries	Component libraries
Design language	Design guidelines	Usage guidelines
Code style guides	Code style guides	Technical guidelines
Patterns	Patterns	Regions
Voice & tone, Writing	Voice & tone	Content

2.2.1 Elements

Web applications are composed of several components that together construct the website, the smallest objects used when constructing components are called elements, see figure 3. Examples of elements are; colors, icons and typography (Vesselov & Davis, 2019). In some design systems, *tokens* are added as elements and these can, for example, be spacing and animation variables (Salesforce, 2020). What both tokens and elements have in common is that they are variables that handle basic styling and are therefore easy to share between teams and products.

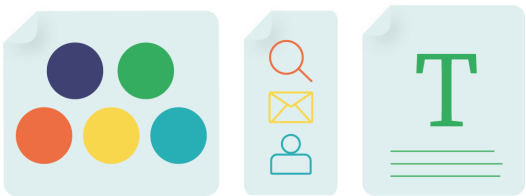


Figure 3. Examples of elements used in web-development.

2.2.2 Components

Even though the notion of component is much older than the web itself, it was not until 2002 they were used in web development (Vesselov & Davis, 2019). Before that point, websites were often built as whole entities and every page on a website contained all information. If a change was made to a recurring "component" on a website, for example, a header, the same change had to be made for every page. As websites grew it became more and more tedious to manage all pages individually and a need to efficiently manage them all emerged (Vesselov & Davis, 2019). The adoption of JavaScript in the mid-2000s also helped in driving the shift from creating whole pages to creating smaller reusable components which have become one of the core principles of design systems used in web development.

Components are collections of elements or other components that together create functionality. Examples of components are buttons, search fields or modals. What all components have in common is that they should be reusable throughout the application (Frost, 2016). Examples of components can be found in figure 4.

Figure 4. Examples of components used in web-development.

2.2.3 Style Guides

Style guides are a set of documentation supporting product development. Vesselov and Davis (2019) argue that style guides are one of the more static components of a design system that often contain information about icons, typography, and colors. Frost (2016) applies a broader definition where he simply states that it exists multiple variations of style guides and that they often include information about the brand, components, and patterns as well. Frost's (2016) definition of a style guide is more diverse than Vesselov's and Davis' (2019), but the items are still of static nature as a result of their connection to the brand and company. MacDonald (2019) takes this one step further and calls the style guide for a brand guide that puts greater emphasis on the brand's inclusion.

> *"Brand guides, brand kits, or visual style guides lean toward visual matters of a brand's identity, including logos and icons, color palettes, typography, and photography. They are sometimes mixed in with editorial style guides."*
>
> (MacDonald, 2019. p 147.)

All definitions return to the same basic elements; colors, icons, and typography. They might also include logos and how they should be used in marketing material. Style guides are often located as static documentation on a website and depending on the company they can be public or private.

2.2.4 Libraries

A majority of the information used during the development of websites can be found in libraries. There are primarily two types of libraries, graphical libraries (Yarkov, 2019) that contain the design of the elements and components, and code libraries that store all elements and components used in development (UXPin, 2020).

The design libraries contain representations of components that can be used when designing. They are often used to facilitate the creation of coherent website designs (Vesselov & Davis, 2019). These libraries are often made and maintained with the help of graphical vector programs such as Figma or Sketch. These programs allow for graphical elements to be grouped together and saved as components to later be reused across the designs when creating mockups.

The code libraries contain components that can be reused in development. Using components in web development is a way to structure the code and to raise efficiency. By using libraries, different teams in an organization share components minimizing the number of times the same code has to be written. Coded components are often stored in repositories such as *GitHub* to allow for easy access and version control (UXPin, 2020).

2.2.5 Guidelines

Guidelines come in different variations but the most common are design, pattern, and code guidelines. The different guidelines can be documented at separate places in a design system but are often gathered and displayed on a website as the design system matures (MacDonald, 2019).

Design Guidelines
Design guidelines explain how to design elements and components and often include best practices and do's and don'ts when using the components. How strict or lose these guidelines are varies depending on company and product. Strict design guidelines can speed up the development process but also have a higher risk of impeding on the creative freedom of the designer (Radford, 2018).

Pattern Guidelines
Patterns are more general guidelines that apply to the structure of a larger part of the website, *user journeys*, or how to structure the code.

Design patterns in code was described as solving specific problems in a recurrent context (Gamma, E., Helm, R., Johnson, R., & Vlissides, J., 1994)). They also go more in-depth and describe a design pattern as having four distinct parts; the name of the pattern followed by what problem it solves, how the problem is solved, and the consequences of using the pattern. MacDonald (2019) defines a design pattern as

consisting of three parts; a description of what the pattern does, the problem the user is facing, or why the pattern is needed as well as a context for when to use the pattern.

Patterns are used by both designers and developers to solve and systemize problems in a similar way, yet with a completely different focus. MacDonald´s (2019) definition puts a lot of emphasis on the end-user of the product and why the pattern is used. Each pattern needs to be explained and argued for in detail as they cannot be classified as objectively right or wrong. According to MacDonald (2019), patterns are a collection of best practices and are prone to change. Gamma et.al (1994) focus more on the problem and how an array of problems can be solved in a structured way by using patterns.

In a design context, patterns describe how components should be used to achieve the sought-after outcome of user journeys. It is a way to keep track of the bigger picture and keep a unified style and ensure a good user experience throughout an application. Patterns often introduce a context to a collection of components (Frost, 2016).

Code Guidelines
Code can be written in a multitude of ways and much like regular languages it needs grammar and syntax to be understood by its users, in this case, developers. Guidelines and rules explaining how to code are often stored in a code style document. This document is often separated from the other parts of the design system and can instead be found in a *README* file in the code *repository* (MacDonald, 2019). Explicitly code guidelines bring up the formatting of the code, for example, spaces or tabs, code patterns, and naming conventions (MacDonald, 2019).

2.3 Designing a Design System

Experts within the design system field claim that design systems are not a one-time project, instead, it should be seen as an internal product, where the designers, developers, and other stakeholders within the organization are the users (Curtis, 2016; Trender, 2017).

If design systems should be seen as products under constant development, it is hence reasonable to address product development processes when building a new design system. This has been done by Devanney (2017), Curtis (2016) and MacDonald (2019) where they stress the importance of identifying the organization's context, needs, and touchpoints in today’s systems or processes, which all are important parts in the early stage of any general product development process. Some examples of activities that could be included in the discovery of the current state are;

- Map the touchpoints in the organization, to understand today’s process and the ones involved (Devanney, 2017)
- Define the needs of the teams (MacDonald, 2019)
- Determine what is most essential in your organization and identify the pain points in today’s process that would benefit the most from a design system (MacDonald, 2019)
- Perform an inventory of today’s interface (Trender, 2017; Vesselov and Davis, 2019)

Vesselov and Davis (2019) advocates looking at a design system like a language and to document and establish a clear vision for this language could be a good starting point for a new design system. One way to start documenting the language is to create component libraries based on the most common components identified during the inventory (Vesselov & Davis, 2019).

To prioritize the building blocks according to their relevance for the product is one of the first steps in building a new design system (Hacq, 2019). Common ways to prioritize and organize the building blocks are through workshops or other interactive sessions where the stakeholders are invited to go through and sort among what has been collected during the inventory of today's interface (Hacq, 2019). Another way of approaching the construction of a design system is to gather inspiration by looking at the way other organizations have built and structure their systems. Useful insights could e.g. be regarding what building blocks to include, how to sort them, and how to document information about usage (Vesselov & Davis, 2019). The next step would be to actually start building the first components of the design system. This can either be done graphically, typically the design guidelines or brand guides, while functional UI components could be built and implemented into code component libraries (Vesselov & Davis, 2019).

However, a design system is not built and implemented overnight. Implementing a new design system requires an understanding of the organization's unique process and all relevant stakeholders should feel involved and onboard during the construction of a new system (Lopes, 2017). To allow for a smooth transition into a new design system, it could be implemented through several iterations, where the most-used components and patterns are the first to be defined and added to the design system (MacDonald, 2019). When the most important parts are in place, the organization can expand the design system as they develop new products or features.

2.4 Introducing a Design System

There is an ongoing discussion regarding when to introduce a design system into an organization. Devanney (2017) argues that small enterprises might not have the need for design systems since the communication between different disciplines comes naturally when they sit close together. When a company grows, the communication complexity will increase and share knowledge organically without any framework will be harder the bigger the company gets. According to MacDonald (2019), there are some cases in which the implementation of a design system could be especially beneficial to implement and those cases are; if the organization has multiple office locations which makes it more difficult to collaborate, there is a need for faster employee onboarding, if diversity in the naming of UI patterns or components is a becoming problem or, if the company is lacking a source of truth.

As MacDonald (2019) mentions, the available time, skills, and resources for implementing a design system is something to consider. Devanney (2017) argues that there is a breaking point of complexity when there are around 20 designers working in the same product development process and that such organizations should consider applying management tools, like design systems, for structuring their processes.

2.5 Lean Principles and Lean UX in Relation to Design Systems

As said earlier, the main purpose of a design system is to systemize and make the web development process more efficient. Analogies can be found between lean management principles and design systems, which Gothelf and Seiden (2016) describe in their book Lean UX. This chapter aims to declare some of the most prominent lean principles that can be applied on a design system.

The lean management philosophy originates from Japanese culture and the Toyota family who started the Toyota Automotive Company which have become a role model and strong influence for management and process systems all over the world (Liker, 2004). The lean ideology intends to increase the efficiency and productiveness of production processes by shortening the time it takes to deliver value to the customer and by eliminating waste (Liker and Hoseus, 2008, Pakdil and Leonard, 2017). Lean is described as an employee-driven approach and empathize on the importance of a company culture that supports collective growth and constant improvement (Liker & Hoseus, 2008; Pakdil & Leonard, 2017).

The Toyota lean philosophy relies on 14 ground principles (Liker & Meier, 2006), four of these principles have been considered as especially interesting with respect to this study.

Principle 2 - Create a continuous process flow to bring problems to the surface
According to the Toyota Production System, flow is created when the production is fully optimized and idle time between the stages of the production has been eliminated (Liker & Meier, 2006). A well-established process-flow will not only speed up the production, but it will also tighten the connections between the operators in the system which, according to Liker and Meier (2006), will make problems surface much faster.

Principle 3 - Use “pull” systems to avoid overproduction
The third principle implies that products should only be produced when requested by a customer. At Toyota, this means that stocks should be small but refilled frequently (Liker & Meier, 2006). The customer consumption should drive, or “pull”, the production to minimize wasted value in terms of overproduction.

Even if the lean philosophy originates from managing automotive manufacturing processes the step towards web-development is not as far as one might think. Relating principle 2 and 3 to web development, production flow, and pull systems can be established by introducing a design system with defined components to be consumed directly by designers and developers. This will decrease the time spent on waiting for colleagues or on over producing components.

Principle 6 - Standardized tasks and processes are the foundation for continuous improvements and employee empowerment
The lean management approach strives to standardize best practices to ensure high-quality processes. The goal is to constantly improve the standardized tasks as new best practices arise, in that way, Toyota Production System could ensure a learning

organization (Liker & Meier, 2006). Creating a shared understanding is also stated as an important factor according to the authors of Lean UX, Gothelf and Seiden (2016). Standardized tasks can help to create collective knowledge. This, in turn, will generate a more efficient process since e.g. the dependency on other teammates will decrease.

Principle 14 - Become a learning organization through relentless reflection and continuous improvement

When reaching a stable and standardized process, continuous improvements becomes natural since waste and problems can more easily be detected (Liker & Meier, 2006). Toyota Production System also stresses the importance of having a human-centered approach to learning, to leave room for reflection and personal growth will contribute to a learning organization (Liker & Meier, 2006).

Another approach that originates from lean management, and especially lean startup methodologies, are the Build-Measure-Learn feedback loop (DelVecchio, White, & Phelan, 2013). The Build-Measure-Learn approach advocates constant testing of so-called minimum viable products (MVPs) to create feedback loops with input from the users. The feedback will support the improvement of the product according to the needs of the users. After each pivot, a new MVP will be released to the user so new feedback could be obtained (Ghezzi, 2018; DelVecchio, White, & Phelan, 2013).

Introducing a design system will imply more standardized work, best practices will be documented in guidelines and standard components will be stored in shared libraries. A design system will also be dependent on continuous improvement to support the product development process and meeting emerging market trends and customer needs. The four lean principles and the Build-Measure-Learn feedback-loop can help to define the settings of a design system and has served as inspiration for the design system for Universal Avenue and the implementation support framework designed in this study.

03.
Method

The methods used in this constructive design research are primarily based on design methodologies including user research and process design. The methods were used to gain knowledge about design systems in organizations, and to investigate the current working process at Universal Avenue to design a new concept of a design system framework. The method chapter has been divided into three different phases; benchmarking, user studies, and concept creation.

3.1 Benchmarking

Benchmarking is a method that aims to compare different phenomenon's against predefined metrics (Maylor, 2010). Vesselov and Davis (2019) argue that investigating other design systems could be helpful by providing inspiration for what to include when building a design system, and also how to structure and document its content.

In this study, benchmarking was used to compare different open-source design systems regarding their content, structure, and usage. A total of five open source design systems were investigated. The selected design systems belong to a mix of large and small-medium enterprises, to see if there was any difference in the content and structure depending on the size of the company. The main building blocks of each design system were identified, and the content was described and compared to the findings from the literature review.

3.2 User Studies

The user studies were divided into two phases; initial observations and interviews to collect data, and secondly a more in-depth interview combined with card sorting to gather information about the current structure of the development process at the company. The qualitative data collected during the user studies was e.g. needs, motivators, frustrations, and opinions related to the working process and design systems. The data collected was compared to the findings from the literature review and used as the foundation for the design phase of the study.

3.2.1 Observations

Observations were conducted to elicit information about the organizational behaviors, which could be hard to elicit through query-based methods. The observations performed at the company have been both direct and participating observations with an unsystematic structure. The observations were performed in the early phase of the user studies to collect data about the company structure and the different teams' work processes. The observations were mainly performed by participating in sprint reviews, sprint planning, and design retrospectives by listening to the discussions that were held during meetings. The data collected from the observations were used in a brainstorming session with the primary goal to result in relevant questions for the first round of interviews.

3.2.2 Initial Interviews About the Current Working Process

As the second step of the user studies, interviews were held with two designers and two frontend developers from different teams within the organization. The aim of the initial interviews was to collect qualitative and subjective data about the existing product development process. The interviews were designed to be semi-structured, with open-ended questions. The structure of a semi-structured interview allowed for comparison between the participants since the interview touched upon the same topics (Bohgard & Karlsson, 2015). Depending on the profession of the participant, the interview questions were adapted to stay relevant.

To get deeper insights into the designers' and developers' daily tasks, questions related to their working process were asked. This was further complemented by asking the participants to illustrate their process by writing down some of the most characteristic tasks they performed on a daily basis. This was followed by questions regarding the group dynamics, both within and between the teams. The interview further aimed to gather some first indications of conflicts and potential needs in relation to the current working process. The interviews were transcribed, and the most important parts were color-coded for further analysis.

3.2.3 Second Interview About the Current Design System

Based on the insights from the initial interviews, a second interview was designed. The goal of the second interview was to gather more information about the designers' and developers' mental models, attitudes, and wishes regarding design systems. The interview started with a card sorting session, which will be described in more detail in section 4.2.4. The card sorting resulted in a system map for which, questions were asked. A total of three designers and three developers participated in the second round of interviews. The interviews were transcribed and prepared for analysis.

3.2.4 Card Sorting

To gain deeper knowledge about the current development process at the company, a card sorting session was included in the second interview round. Card sorting is a way to use notes to create an overview of a structure or a system (Spencer, 2009).

The participants were handed a number of cards with predefined keywords collected from the first round of interviews. The keywords on the notes were a collection of tools, processes, and generic words used in web development. The participants were asked to categorize and group them according to Universal Avenues' current processes and explain all their choices out loud. A number of empty notes were given to the participants for them to add new words in case the provided ones were insufficient. After the first round of sorting, they were asked to connect every group to a profession and also mark in which area of the development process they themselves spent the most time working in. The last step of the card sorting had the participants sort the cards and categories them in a structure according to how they wanted to work. Figure 5 illustrates an example from one of the card sorting sessions.

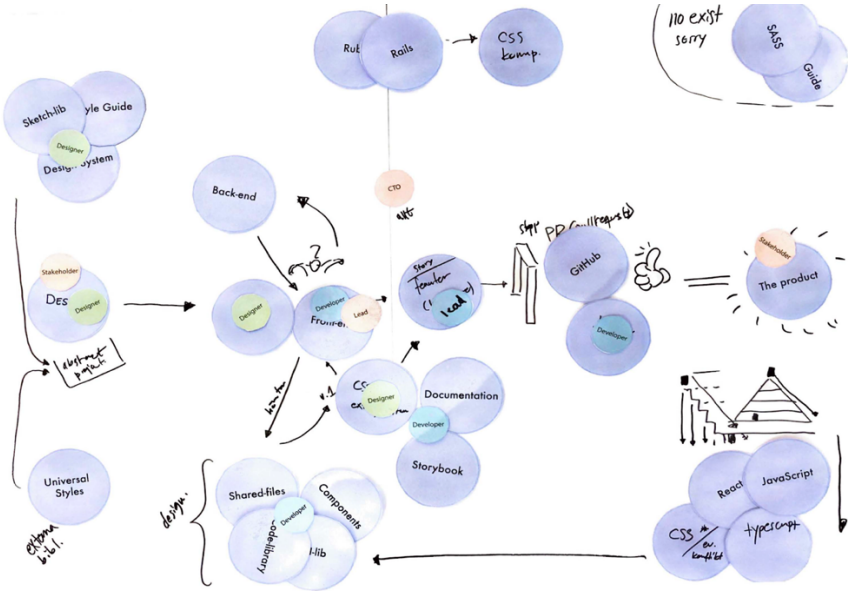


Figure 5. Example from one of the card sorting sessions.

3.2.5 Analysis of User Studies

Through qualitative data analysis the data was transcribed, decoded and, structured which made it possible to find similarities, trends and patterns (Bhatia, 2018). Defining these factors will increase the chance that the relevant problems will be solved. Such trends or patterns could, for example, be; ideas, concepts, or behaviors (Bhatia, 2018). The results of the analysis acted as the base for the following design phase and the summarized data was presented and discussed together with the participants in the studies as an evaluation step.

The analysis method used was KJ-analysis. The KJ-analysis method is used for decoding and sorting data into groups, often on small notes of paper (Scupin, 1997). The process of analyzing data with the KJ method includes four steps; label making, label grouping, chart-making, and written or verbal explanation (Scupin, 1997). Label making is the creation of notes, often a single word or a short phrase. Label grouping is a process of sorting the previously created notes into groups. These groups often represent different problem areas. The second to last step, chart-making, is when the data is summarized and displayed in an appropriate chart. The last step is to explain the data and deconstruct it so that valuable insight can be made.

The data gathered from the first and second round of interviews were analyzed with the KJ analysis and the data from the card sorting was used to create a relation map framework. Frameworks are used to highlight the key relationships and help understand the context for which the map is made (IDEO, 2015). The relation map was used to map the current working process and as inspiration and input for the design system framework.

The outcome of the user study analysis was defined user profiles for both designers and developers and a detailed description of the current design system at Universal Avenue together with the defined problems.

3.3 Concept creation

The design process aimed to elaborate on the defined problems that emerged from the user studies. The first step was to generate different concepts for the design system construction. The next step was to define its implementation process which then could be evaluated with the users. The concept creation consisted of three different phases that all will be presented below.

3.3.1 First Ideation Session

The first iteration session was made internally in the project group and aimed to generate ideas and early concepts. The session focused on defining and ideating on the content of the design system as well as its implementation process. The methods used were brain-drawing and brainstorming. Brain drawing is a way to explore ideas by drawing (IDEO, 2015) and brainstorming is a method that helps a group of individuals to increase their creativeness and generate ideas (Wilson, 2013). The best ideas were summarized at the end as method deliverables (Wilson, 2013). These two methods were used in combination with the material from the card sorting and the guidelines from the user profiles. The ideas from this first ideation session were summarized and digitized to create a better overview.

3.3.2 Co-Creation Workshop with Designers and Developers

After the internal ideation session, a co-creation workshop was designed to extract additional insights and ideas from the two user groups. The workshop was designed to work in a digital format and an online app called Figma was used as a collaboration-tool, which allowed all participants to work in the same document and to see each other's cursors. The workshop was divided into three main parts; design system building blocks, create new components and, implementing the design system.

In the first part of the workshop, the participants were asked to ideate on the content of the six different parts of the design system that had been identified through the analysis of the user studies, and further defined in the first internal ideation session.

Material was provided to help the participants in defining the content of each building block. For each of the six categories, the participants had to answer questions by dragging the provided material into different areas, the participants were also allowed to add additional text and symbols, see figure 6. The questions to be answered by the participants focused on the content of each building block, who would be in charge of the maintenance and, where the content of each part should be documented and stored.

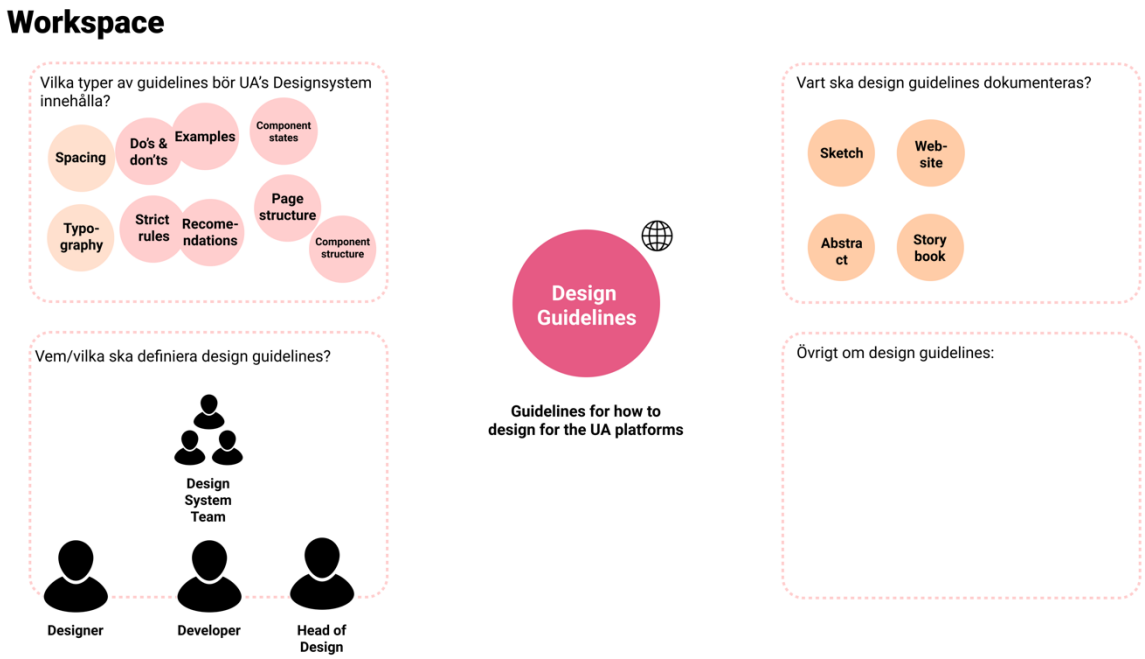


Figure 6. An example of the workspace used to define the content of the design system.

The second part of the workshop focused on the process of implementing a new component into the different global design system libraries. This was done by letting the participants place crucial events on a timeline that had a defined start and end. The workshop provided some pre-written examples to work as inspiration for the participants, but they were also free to add their own content to the timeline or change in the pre-written material, see figure 7. The pre-written starting point was; "The need for a new component is detected" and the endpoint of the process was; "The component is implemented in all global component libraries of the design system".

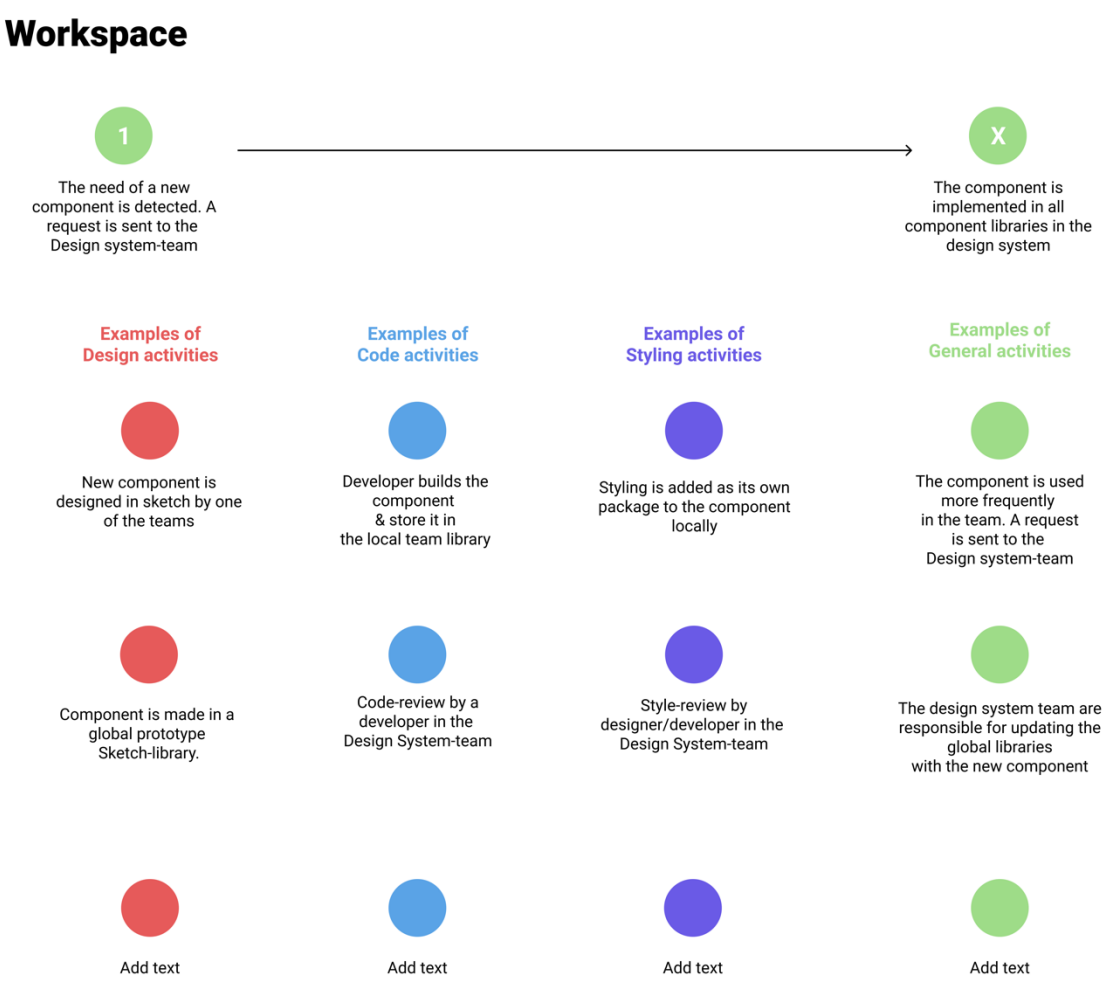


Figure 7. The workspace with the timeline and the pre-written events.

The third and last part of the workshop aimed to ideate the process of implementing a design system at the company. The approach was the same as for part two, the participants got to place different events on a timeline, where some of the events were pre-written for inspiration. The starting point for this part of the workshop was; “The first building block of the design system is initiated.” and the endpoint was; “A first version of the design system is ready to use by all teams at Universal Avenue.” In addition to the pre-written events, tags that could be used to indicate time was also provided for the participants to use, see figure 8.

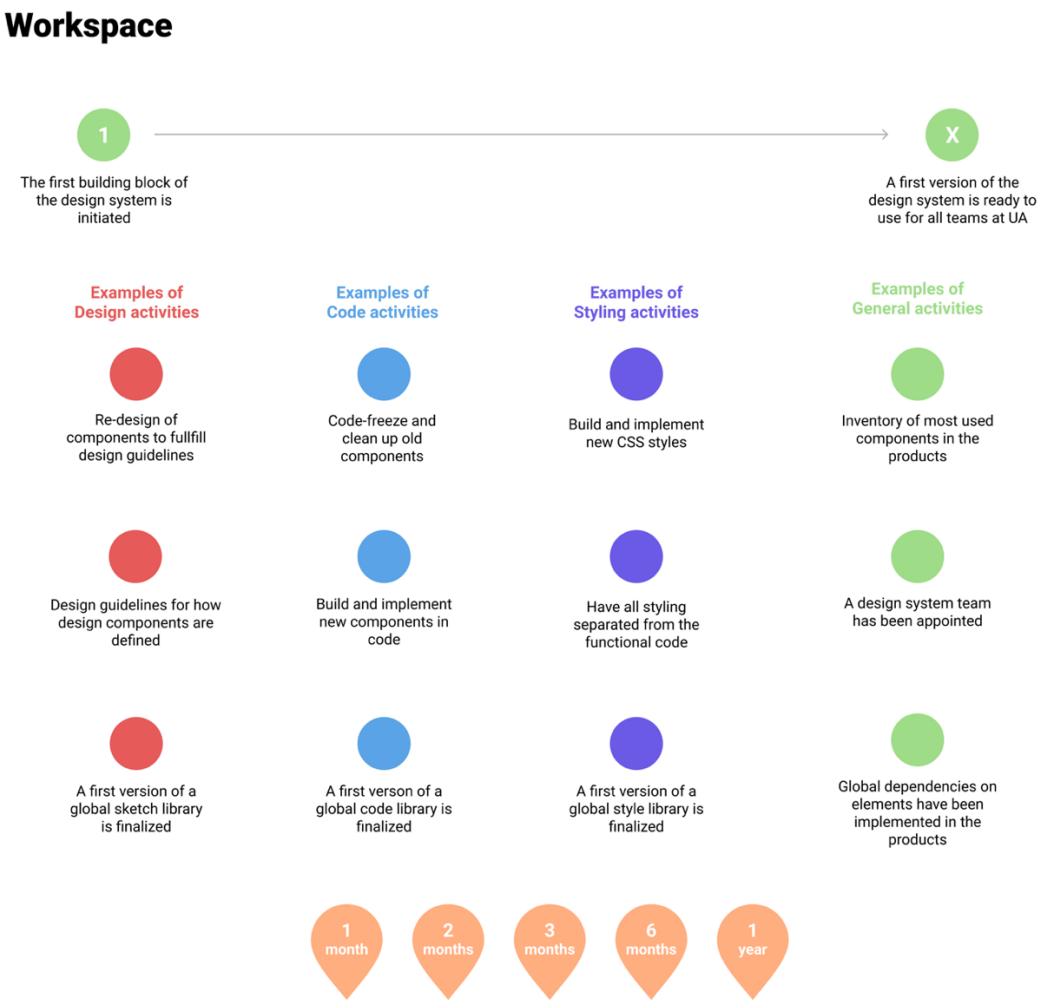


Figure 8. The workspace with the timeline and the pre-written events and the time indicators.

The workshop was performed on two occasions with a total of four users, one designer and one developer were paired together for each session. Both the audio and the screen were recorded during the workshop and was transcribed after the sessions. The material that was used during the workshops was also saved for further analysis. The transcribed data were analyzed through a simplified KJ-analysis where the data was grouped into different categories.

3.3.3 Concept Iterations

The concept for the design system framework was evaluated together with the designers and frontend developers at the company and took place in two iterations. The first iteration focused on the contents of the design system while the second shifted the focus to the implementation process.

Base Concept
The data collected during the user studies and benchmarking acted as a base for the ideation and conceptualization of a first version of the design system framework. This concept contained a process for component development and overall library structure.

First Iteration
During the co-creation sessions, the users generated ideas for an implementation process for the base concept. The concept of the implementation process was visualized and refined before it was evaluated together with the participants by sending them the visualized process. Between the base concept and the first iteration, the focus shifted from the content of the design system towards the process of implementing a design system.

Second Iteration
Based on the insights from the evaluation a second iteration took place where minor tweaks were made to the process of implementing a design system and the structure of the component libraries. In practice, the tweaks meant that some sections of the implementation process were simplified and expanded upon in a separate process.

3.3.4 Concept Evaluation

The co-creation workshop was used to evaluate the first iteration of the concept and took into account the feedback from both designers and frontend developers at the company. All iterations of the concepts also went through expert reviews that allowed the designers and frontend developers to give their feedback and influence the continued improvement of the concepts.

3.4 Methods in Relation to Result Outcome

Table 2 represents an overview of the result chapters in relation to the methods that have been used in this study to reach the results.

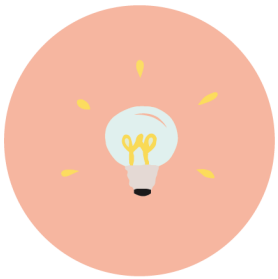
Table 2. The result chapters in relation to methods used.

Results chapter	Methods used
4.1 Benchmarking Results	Benchmarking (3.1)
4.2 Current Working Process at Universal Avenue	Observations (3.2.1), Initial Interview (3.2.2), Second Interview (3.2.3), Card Sorting (3.2.4)
4.3 Problem Areas	Analysis of User Studies (3.2.5)
4.4 Current Design System at Universal Avenue	Observations (3.2.1), Initial Interview (3.2.2), Second Interview (3.2.3), Card Sorting (3.2.4)
4.5 Relation Map of the Current Design System	Card Sorting (3.2.4)
4.6 User Profiles	Analysis of User Studies (3.2.5)
4.7 Defining a Design System Concept for Universal Avenue	Analysis of User Studies (3.2.5), Design Process (3.3)

04.
Results

The result chapter will present the findings from the benchmarking together with a description of the current working process and design system at Universal Avenue. This will be followed by a presentation of the user profiles and guidelines for how to develop a design system that will meet the needs of the user profiles. Lastly, the finalized concept of the design system framework for Universal Avenue will be presented in detail.

Only the final concept of the design system framework will be presented in this chapter, see appendix II for early concept iterations on the content of the framework.



4.1 Benchmarking Results

This section presents the findings from the benchmark. A total of five different open-source design systems made by large and small-medium enterprises have been investigated and compared according to the layout, content, and usage. An overview of the companies is presented in table 3.

Table 3. Overview of the benchmarked companies.

	Business Area	Founded	Employees
ATLASSIAN	Software development of tools to support collaboration and project management.	2002	3 000
Meetup	Community platform organizing meetup-events.	1999	300
salesforce	Customer relationship management services.	1999	49 000
shopify	Software company developing e-commerce solutions.	2004	5 000
Wonderbly	Publishing company that provides personalized children's books.	2012	100

All design systems included in this benchmark follow the general structure of how a design system usually looks according to the findings from the literature review. Meaning that they follow a hierarchical structure where the building blocks can be broken down into smaller components and elements. A compilation of the design system's main elements has been listed in figure 9. The compilation shows that color, typography, grid, and spacing were the most commonly used elements in the benchmarked design systems. Some of the systems, primarily the more mature, includes even more elements such as icons and animation as well.

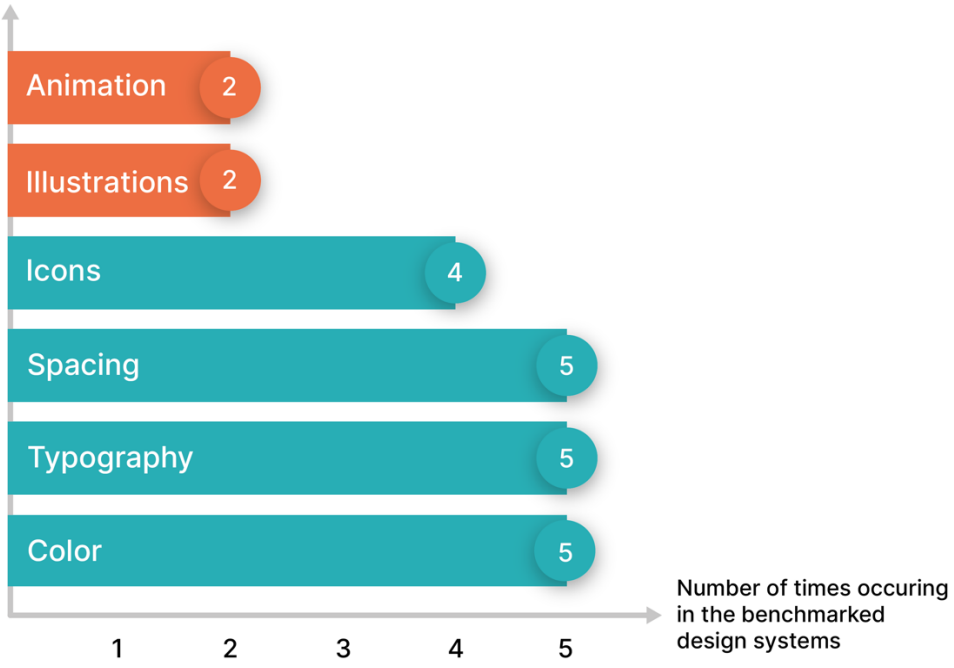


Figure 9. Elements used in the benchmarked design systems.

All the benchmarked design systems also contained guidelines to support the product development. The design system's different types of guidelines have been presented in figure 10. Code examples and illustrative examples were used by all design systems that were investigated. Most of them also included typography, icons and accessibility guidelines. However, the guidelines were distributed differently on the design system websites. Some companies, like Salesforce and Atlassian, had clustered all design guidelines under one single menu tag. Shopify's and Meetup's guidelines was mainly displayed in relation to each element, while Wonderbly presented the guidelines in relation to the components.

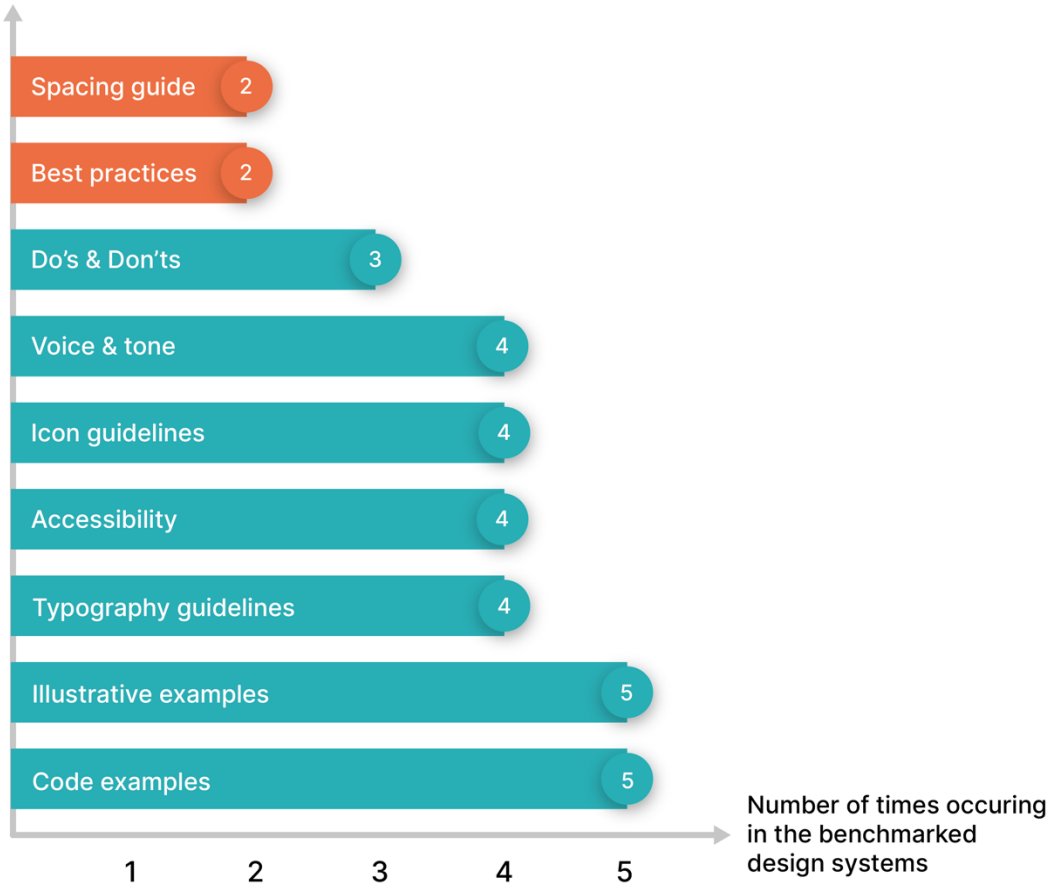


Figure 10. Guidelines used in the benchmarked design systems.

Component-lists are also recurrent in all the design systems. Atlassian, and to some extent Meetup's Swarm, had chosen to make two separate component lists in their design systems. The different lists either focuses on the design and context in which the component should be used or, it focused on the code and how to implement the component into the product. The other design systems only had one list of components, and by clicking at one of the components in the list the user got access to both design guidelines and code examples related to that specific component.

What was found when comparing design systems of smaller enterprises, like Meetup and Wonderbly, to the Atlassian's and Salesforce's systems was the increased complexity of the design system as the companies gets bigger. The bigger and more complex systems also had a stricter way of representing the information regarding their design system's building blocks. This is reasonable due to the great amount of information included, which was not the case for the smaller enterprises. They did not have the same need for such a strict structure in their systems since they did not contain the same amount of information.

A more detailed analysis of the benchmarked design systems can be found in Appendix I.

4.2 Current Working Process at Universal Avenue

Universal Avenue is using an agile approach for software development and work in two-week sprints with quarterly reconciliations regarding objectives and key results. The development teams are structured to include persons from different disciplines but at least one product owner, one designer, one frontend and one backend. It is possible for the product owner to be working in the project under another title, for example, a frontend developer can also be the product owner. During each sprint, the development team manages their work by writing so-called *stories* that e.g. describe what functions or designs that should be added to the product. An example of a *story* would be; "as a salesperson, I would like to have a list of all my customers". Stories are a way to relate the development process to the needs of the end-user. The stories are managed through a *Kanban* board where the stories are sorted into different columns.

Each sprint is followed by a sprint review organized by the CTO and the product owner of each product team. During the sprint review, the team presents what has been built during the last two weeks. The sprint review is followed by a sprint planning meeting where the team writes the new stories for the upcoming sprint and prioritizes them in the backlog of the Kanban-board.

4.2.1 The Designer's Working Process

The designer's working process starts with the designer choosing a story to work on. The design process depends on the story, but generally, it is either a new function or a redesign that is to be created. When starting with a new story, the designer often does a check-off with his or her stakeholders to get the context of the new design. When the context is known the design process starts in Abstract, which is a version control software for Sketch-files. If there already exists a version of the design a *branch* will be created in which the redesign will be developed. If there is no current design, the designer has to create a new project in Abstract.

When a new design is designed in Sketch, the designer usually has a demo session with the management team and the stakeholders to get feedback on the design. The demo session is followed by iterations if needed. Then the design will either be formulated into a new story for development or the designer will style the design in *CSS* before handing it over to a developer. This process depends on the team, where all teams have different approaches for how to hand over a design to the developers, the approaches might also vary within the team depending on the task.

After the function has been added to the design by the front-end developer, the designer performs a design-review and comments on the developed component from a design perspective. If the designer approves the design of the feature and another developer approves the code, it can be implemented in the live product.

4.2.2 The Frontend Developer's Working Process

The developer's working process starts with choosing a story from the backlog of the Kanban-board. This is followed by the developer creating a new branch from a master

file from GitHub and continues to work on the story in that new code-branch. When finished with the task, the developer opens a pull-request at GitHub. Then, another developer will have to make a review on the pull-request and either approve the changes or leave feedback. When the changes in the code have been approved by at least one other developer, it can be merged into the master file and will then be a part of the live product. Depending on the story, the developer could also ask for a design-review from a designer at this stage.

4.2.3 The Current Code Structure

The Code structure at the company can be divided into three parts. First is the backend that contains all data and database structure. Followed by frontend that handles all the functionality and implementation of the visuals. The visual styling is written in the preprocessor *Sass* or CSS depending on product.

The different parts of the product can be written in different languages and have access to various libraries. These libraries are to some extent shared between the different products in the company if the products are based on the same languages. Two out of three product teams use JavaScript and *React* for the frontend while one team is using Ruby and *Ruby on Rails*, which is also used by the backend. Code is shared between the teams working on the same product and to some extent between the products that use the same languages. How the different products share code and how Sass and CSS affect the code can be seen in figure 11.

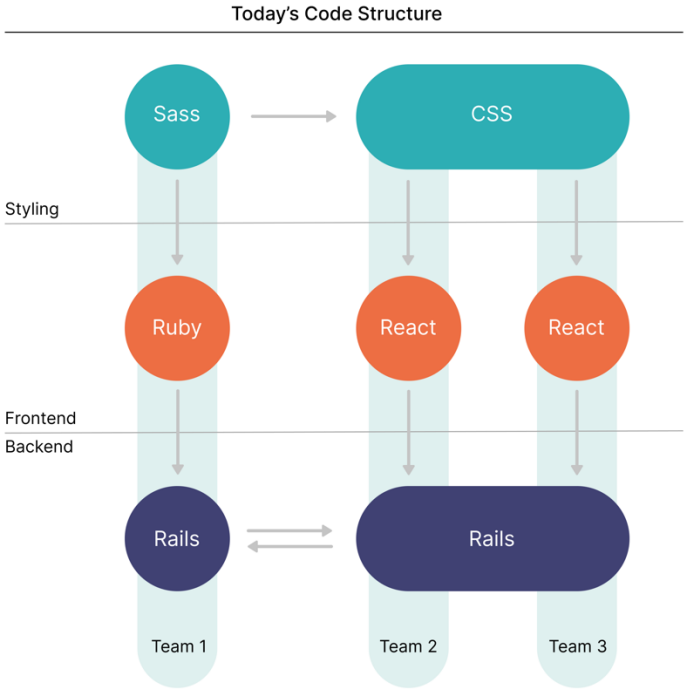


Figure 11. The current code structure at Universal Avenue.

4.3 Problem Areas

This chapter presents the major problem areas found during the first interview round at the company. The problem areas regarding the designer's and developer's current working process and the general structure of the current design system at the company have been summarized in the list below;

1. No universal system for development and design
2. Lacking collaboration between the teams
3. Low reusability of components
4. Not enough time for maintenance
5. Lacking feedback from, and information about the users

The current design system at Universal Avenue lacks a clear process for product development. It currently contains two sketch libraries and two component libraries. Having two libraries of each type is not a problem per se but it becomes a problem when there is not a process in place for how to manage them. After the interviews, it became clear that neither designers or developers understood the differences between the libraries completely, and in many cases, they only used one of the libraries making the other one obsolete. The lack of a clear system also amplifies the problem of not having a single source of truth, as much time is spent on discussions that would not be necessary if a defined process was in place.

Both designers and developers wished for more communication both within and between the teams. The lack of communication lowered the amount of collaboration that was possible as each team acted rather independently from one another. The biggest problem with this way of working was that the team members from one team did not know what the other teams were doing. This resulted in the developers being too dependent on reviews from their own team. If the majority of developers were busy it could take considerable time before a review was made, which could block the developers from continuing with the next task. Designers had no clear process for when and how to share design-ideas resulting in some decisions and ideas not being shared with all designers. Worth noting is that the cross-disciplinary collaboration between designers and developers in each team was working great and something both parties wanted to keep.

Somewhat connected to the lack of communication is the low reusability of components. As each team was working independently it was hard to create components usable for all teams. There was also a lack of structure when it came to how the components were made and no clear guidelines defining how strict the components should be.

The company had a high focus on creating new features and components. The pressure to always create new components leaves little to no time for maintaining the already written code. The existing components might therefore be changed locally to remove the need for extensive quality assurance (QA) testing. By breaking the connection to the main component, the changes introduced do not affect the other uses of the same component and a lot of testing can be avoided, speeding up the delivery in the short term. The constant creation of components also creates a need for fast code reviews. Every new change needs to be reviewed by another designer or developer before it can

be merged into the product. This way of working has a tendency to create legacy code that is harder to maintain and use.

4.4 Current Design System at Universal Avenue

Through the user studies, five different building blocks of the current design system at the company could be identified, see figure 12. Each of them will be described below.

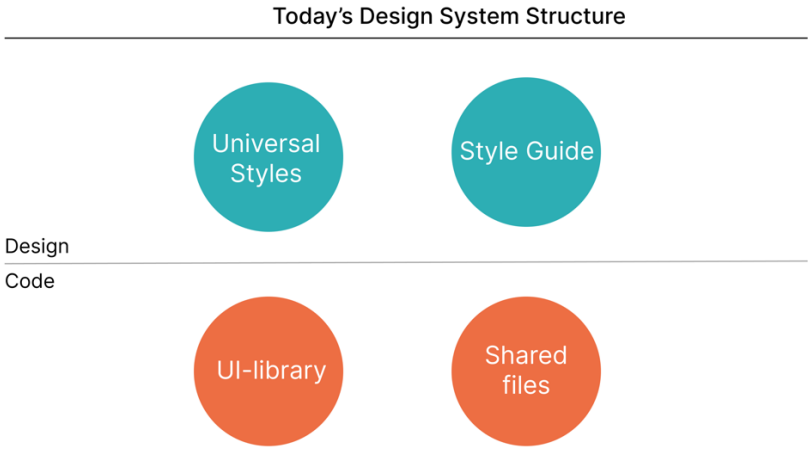


Figure 12. Current design system at Universal Avenue.

4.4.1 Universal Styles

Universal styles is a Sketch library that contains a Style guide with information about color, typography, buttons, and some simple forms. It also contains components for the old Universal Avenue platform that is no longer in use. The styles and components found in the library are outdated and the library is currently being phased out of the design process and is to be replaced with the new design system.

4.4.2 Style Guide

The Style guide has two different meanings at Universal Avenue. By the designers, the Style guide is referred to as a project on Abstract called Style guide. This project has a sketch-file called Style guide which contains descriptions and best practices for how to use; the logotype, typography, colors and, spacing. As another part of this Style guide project, there is a sketch-library called UAUI. This library contains; fonts, typography, colors, buttons, tables, and symbols. UAUI is used as a dependency and library for other projects on Abstract where the designers are able to reuse the elements and components when designing new pages. Thus, the Style guide is considered as today's design system by some of the designers. However, the Style guide is used differently by the teams and some rely more on the Style guide when creating new designs than others. Designers are the only actor using and updating this Style guide and it is currently out of sync with the code.

By the developers, the Style guide could be referred to as the place where they store and display the coded components, i.e. Storybook. It could also be referred to as pure documentation for how to write code. The latter is, however, team-specific documents created by each team individually.

4.4.3 UI Library

The UI library at Universal Avenue is a repository located on GitHub that consists of coded components used in production. The library is used to varying degrees by the different teams but is supposed to work as a source of truth for all teams that use react as the frontend framework.

The current implementation of the library contains too many outdated components and the ones that are not outdated are in need of rewriting so that they follow the best practices of today. When components do not follow the best practices, the logic has to be rewritten for the components to work together with the code used in production. The UI library was supposed to work as a base for Storybook so that all components in the library had a visual representation. Unfortunately, this was not the case even though the goal was to have a complete correlation between the UI library and Storybook.

4.4.4 Storybook

Storybook is an open-source service that provides a platform to develop, organize and display UI-components. React is one of the frameworks that are supported by Storybook (Storybook, 2020). At Universal Avenue, Storybook is used as a place to display UI-components. The aim was to use Storybook as the source of truth when developing new features and that all components in the UI library should be displayed through this platform. However, none of the investigated teams are using Storybook as intended. The reason for this is the lack of maintenance to keep it up to date with the UI library. Another reason why Storybook is not used by all teams at the company is due to the lack of support for Rails, which is the code base used by one of the teams. Further, the diversity of components needed to support the different functions of the products is making it hard for the teams to reuse each other's components as they currently are too specific.

4.4.5 Shared Files

Shared files are used by the local teams and were originally a copy of the UI library but has with time developed to be its own entity. Many of the original components have been reworked and plenty of new ones have been added. There are no dependencies between the UI library and the Shared files so a change in one of the repositories does not affect the other in any way.

4.5 Relation Map of the Current Design System

The result from the card sorting method has been compiled into a relation map, see figure 13, which illustrates the correlation between the different cards, i.e. the elements of the current design system at the company. The compilation has been done

statistically by counting the number of times two words were grouped together by the participants. If all participants had grouped two words together the words were linked with the number 6 in the relation map. If five participants had grouped the words the link was marked with a 5 and so on all the way down to three, which corresponds to half of the participants. Words that were grouped together less than three times has not been considered in the relation map.

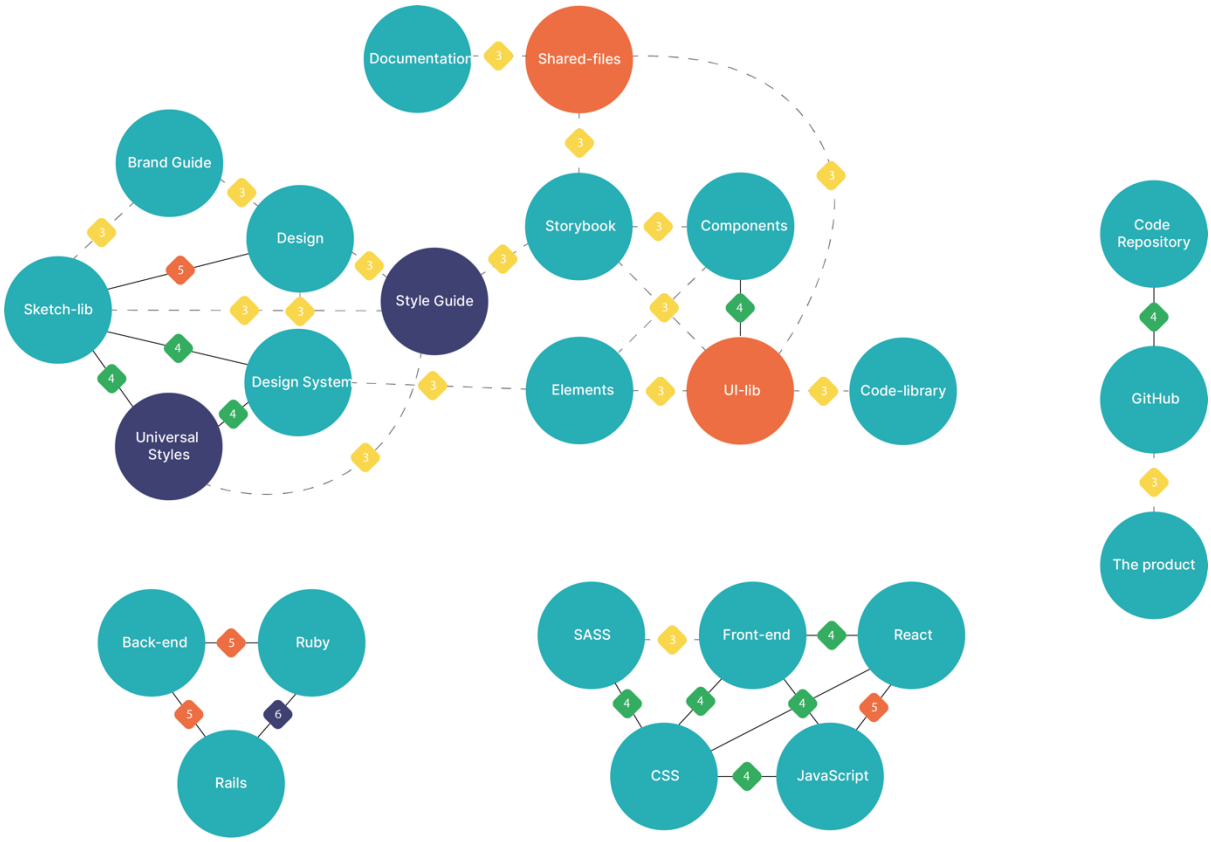


Figure 13. Compilation of the current design system based on the card sorting method.

Based on the relation map, some main conclusions could be made:

- Backend is separated from both frontend and design - nothing in the current design system is shared with backend.
- The current design system is mainly connected to the Sketch libraries which is maintained by the design team.
- SASS and CSS styling were connected to the frontend developers.
- There is no established connection between the sketch libraries and CSS or SASS in the code libraries.
- The Style guide was referred to by both designers and developers and that is why it has a central part in the system map. However, the Style guide could either be described as a guide for how to design or as a guide for how to write code. Some of the developers mentioned that they had internal guidelines for how to write code. A common Style guide for all teams is missing, both on the design side and on the code side.
- There is no Brand guide today, it is requested by the designers and would be connected to the design team and the Sketch libraries.

- UI-lib and Shared files are both versions of Storybook but used differently by the frontend teams. Hence, there is not a single way to connect the code-libraries in the current system.

4.6 User Profiles

The result from the user research has been summarized into two user profiles, which are representing the stakeholders of the design system at Universal Avenue. The two main users of the design system are designers and developers. Firstly, the problems and concerns of each user group will be described. Then, guidelines of what the design system should fulfill, to meet the needs of each user group respectively, will be presented.

4.6.1 Designers

The designers are the primary user of the current design system and were mainly consuming and contributing to the Style guide and Sketch libraries described in 5.4. The problems and needs described in the following sections are based on insights received through observations and interviews with all designers, one from each of the three teams.

What is rather unique among the designers at Universal Avenue is that they can write *HTML* and CSS code, which most designers cannot. This makes the development process a lot easier since the designers can style the web pages directly with code, instead of just handing over a design sketch to a developer. But since they are only one designer per team, they cannot keep up with the developers' speed. Hence the designers have to do quick mockups of the design in Sketch which the developers can have as their reference point when coding. The styling made by the developers then has to be reviewed by the designer, taking away even more time from the designers to actually write the styling themselves.

As a designer, it is important to not have too many restrictions. In the current situation, the designers express their concerns about the code structure being written in a way that hinders them from implementing their design. Some components in the products are written locally without dependency on the main library. This makes it harder to introduce new designs to the whole product at once and the designers have to change the design of a component in every place it is used. If the components still had a dependency on the main library a change in design would update throughout the product and much time could be saved. This problem leaves the designers with little freedom and time to design, test and, optimize new user interfaces.

The current design library is stored and updated in two different Abstract projects; Universal styles and Style guide. The problem with those Abstract projects is the lack of guidelines and descriptions for how and when to use the components in the product. Due to this, the designers have to make their own decisions regarding different component styles based on the functionality required by their specific part of the product. This is causing diversity in the overall product design. In relation to this, the

designers' ability to see what components already existing in code is lacking. There is currently no platform where designers and developers can sync the design sketches with the code. There is also a lack of coded equivalents to the components that exist in Abstract. This means that both designers and developers have to reinvent the wheel for almost every new page that is developed.

Guidelines when developing a design system for designers at Universal Avenue

The implementation of a design system should help the designers to work more effectively towards a uniform and user-friendly product. To do so, the design system must contribute to decreasing the current gap between the different teams, regarding both design and code. Instead of spending time on design-reviews, the designers should be able to focus more on the creation and testing of new designs. The design system should further contribute to a shared view and a clear structure of the design process that will serve all product teams in the most suitable way. To solve the problems described in this chapter, a design system should:

- Be up to date and display which elements, components, and patterns that are used.
- Split up the code into more manageable chunks with less nested dependency and allow both designers and developers to access the code.
- Include guidelines for how to address design problems and what the overall design of the product should express.
- Be shareable between the different product teams.
- Have one person or team in charge of the maintenance of the design system.
- Allow notification for all stakeholders about updates or changes in the design system.
- Allow comparison of the current design in Sketch with the corresponding styling in code.
- Include instructions for how to write the code in a correct hierarchical structure that allows global changes in the design of the products.

4.6.2 Frontend Developers

The frontend developers are the second primary user of the current design system and they are consuming components from the different code libraries on a daily basis. The problems and needs described in the following sections are based on insights received through observations and interviews with a total of six frontend developers from Universal Avenue.

Storage and maintenance of components in libraries

Frontend developers work with the goal of delivering well-written code that supports needed functionality. To make their work more efficient, code libraries are used to provide components whose functionality and design have been tested and approved. Even though the components in the current libraries have been tested, they are often not fully compatible with the code structure of each team. This results in developers having to rewrite the components to varying degrees. The process of rewriting code is both annoying and time-consuming, so the developers prefer to reuse their local components. This will lead to diversity in the product's code structure. One developer even expressed the problem as; "If the UI library would have been an open-source

package we would have decided to abandon it since it is not stable enough and there are conflicts between the components."

There are two internal libraries used to manage, store and display code at the company; UI-lib, and Shared files. In the current situation, these libraries can be considered as three separate pillars but with some content overlapping. The problem is that product teams are using the libraries with different approaches which are causing local proliferation and diversification of the components. This complicates the maintenance of a global library that works for all teams since local dependencies have been created when components are adapted to a specific context. Another problem that stands in the way for a global library is that the current libraries, are all written with React as the codebase. This means that the team, which is based on Rails, cannot use any of the components that exist in the global libraries except from CSS code for styling.

Collaboration and development

The collaborations *within* each team are great and a lot of discussions are held regarding the code structure and how to solve complex problems. On the other hand, the collaboration *between* the teams is lacking. This has resulted in a big code repository that few developers understand completely. There is no efficient way to derive the components in the libraries to their presence in the overall product which have created hidden dependencies that complicate the structure of the library. The lack of traceability has created a dilemma that developers cannot change components in the library in case it ruins the functionality of a component in another part of the product.

The developers want to put their focus on building new features that can be added to the product, rather than spending time on bug fixes and administrative tasks, like documentation. They also express the importance of working close to the designers and they enjoy having discussions about UI and UX problems instead of receiving a design "in their lap" once it is done by the designer. If the developer and designer manage to have more discussions about changes in the product, they would both save time that they usually spend on design reviews. Also, if the designer considers already existing components when creating their designs, the developer can put more effort into creating great functionality instead of creating new components.

Code structure

An ongoing discussion at the company is in what level of detail the coded components should be written. If a component is strict and well defined it can be implemented without any major changes in the code, however, the number of locations where this component can be used will be very limited, and hence a lot of components will be needed to cover all use-cases. If a component, on the other hand, is more flexible and less defined it can be applied in more use-cases, but the developer will then have to add more code to contextualize the component. When discussing this with the developers they express the importance of having a balance between control mechanisms and complete freedom. Hence, perceived freedom can be considered as an important factor to keep in mind while designing a new design system.

Another problem that is related to the structure of the code is whether the styling should be separated from the rest of the frontend code or not. Currently, some of the CSS styling is written in the JavaScript files instead of in the CSS files. This not only

excludes the designers from the styling process but also creates unnecessary dependencies and limits the global reusability of components. An exemplification of this problem is that the typeface used in the product cannot be changed globally due to local CSS-styling. This means that if they what to change the typeface, all locations where the typeface is defined have to be localized in the code and changed manually.

Guidelines when developing a design system for developers at Universal Avenue
By introducing a design system, the developers could get more time to create new functionality instead of spending time on bug fixes or building new components. To achieve this, the developers should not have to worry about a component from the global library breaking any existing code. If the same components are used throughout all products the codebases are going to look more and more the same with the logic being structured in similar ways for all teams. By creating this similarity, developers would have an easier time keeping up to date with the other projects and even be able to collaborate more easily across the teams. To solve the problems described in this chapter, a design system also should:

- Unify the code to follow the best practices when it comes to how to code a component.
- Unify the teams to follow common guidelines for how to code a component.
- Make it easier for the developer to understand what impact a change in the code will have on the overall product.
- Make it easier for designers to use components that have a coded counterpart.
- Allow developers to spend most of their time on development and minimize the time spent on administrative work.
- Have a global dependency on elements.
- Include a global code library.
- Support a code structure that separates the styling of a component from its functionality.
- Make it easier to see what components are used in the live product.
- Provide “ready to use” components.
- Be easy to maintain and update.
- Provide guidelines for how developers can “consume” design.
- Allow developers to focus more on implementation and the designers to focus on the styling and UI parts.

4.7 Defining a Design System Concept for Universal Avenue

The literature review, benchmarking, analysis of the user studies, and co-creation sessions have together resulted in a concept of a design system for Universal Avenue. The concept is divided into seven different parts and will be defining;

1. The defined problem
2. The goal of the design system
3. The content of the design system
4. Maintenance and documentation of the design system

5. The implementation of the design system
6. The process of building global components

4.7.1 The Defined Problem

The main problems identified during the user studies at the company was related to the structure of the codebase. The current code structure contains a lot of local dependencies which hinders global changes and the implementation of new designs. Many components used in the current products are also developed and only implemented on a local team basis which complicates the maintenance of a global component library. This together with the lack of common design guidelines for how to create new components are causing diversity in the overall product design.

4.7.2 The Goal of the Design System

The goal of the design system at Universal Avenue is to make the development process more efficient and predictable by providing a set of guidelines and components that can be reused during development. By streamlining the development process, both designers and developers can minimize the effort spent on rudimentary tasks and instead shift their focus towards creating more value for the customers and company. The new design system will be catered towards the users at Universal Avenue by fulfilling their most important needs gathered during the user studies.

4.7.3 The Content of the Design System

Insights from analyzing the user studies together with foundations from the literature have resulted in three building blocks being defined as the cornerstones of Universal Avenue's design system. The three building blocks are introduced in the following sections.

Design Elements
Design elements have been defined as the smallest building block of a component. It is hence reasonable to start defining the elements when building a new design system. Based on the data collected from the co-creation session the main design elements for Universal Avenue are; color, typography, icons, and spacing.

Color elements are used both for branding and for functional indicators, such as warnings or alert messages. The colors should be defined with color-codes to secure that the same colors are used all over the product. Since the colors have to be defined in both design-mockups and code, consistent naming of the colors is crucial to foster cross-disciplinary collaboration.

The typography element defines which typeface, font-sizes, font-weights, line heights and bottom margin to use when using text in the products. The typography should be divided into different, so-called, typestyles which indicate different typography use-cases, e.g. which size, weight, and line-height that should be applied on a heading or body text.

The icons are often used as compliments to text e.g. when building a header menu. The icons should follow common principles to work well together and to create a user-friendly experience for the end-user.

The last element to include in a design system at Universal Avenue is spacing which defines the distance between elements and components, e.g. the distance between the text and the box element of a button. Spacing elements will require guidelines and examples illustrating how it should be applied in the product.

Design Guidelines
Design guidelines have been defined as documentation for how to use the elements and components to design and build parts of the product, or even whole pages. The general structure and content of design guidelines at Universal Avenue was defined during the co-creation workshop and will be further elaborated and described below;

- Examples - The design guidelines should include relevant examples of components as well as information about how they should be used and why. The examples should also show different states of components, e.g. the hover state of a button. The examples should include both graphical elements, styling, and code.

- Do's and don'ts - The design guidelines should include do's and don'ts which will describe how to or how not to design pages. An example of a "don't" would be to use black text on a dark background. The designers considered the don'ts as most important to display in a design system since it would allow for everything else to be explored except the don'ts.

- Typography - The design guidelines should include instructions for how to apply the typography elements.

- Spacing - The design guidelines should include instructions for how to apply the spacing elements.

- Patterns - The design guidelines should include instructions for how to use patterns when considering user-flow and the structure of pages in the product.

- Accessibility - The design guidelines should include instructions for how to consider user accessibility.
- User behavior - The design guidelines should include documentation about what user behaviors that should be encouraged by the design of the product.

- Code structure - The design guidelines should include documentation about how to code a component so it can be reused in all teams and included in the global code library.

The users stressed the importance of consistent guidelines. To achieve that, some control mechanisms will be needed for the maintenance of the guidelines, preferably by a dedicated team or person. Another aspect to consider when designing guidelines are the degree of freedom, the guidelines should be applicable in all product teams at the

company, and hence there must be some freedom to adapt the components to different contexts. Also, too many strict rules can inhibit the creativity of improving the design of the product. On the other hand, too little regulation could result in inconsistent product design followed by poor end-user experience.

Design guidelines should not only include guidance for designers, but the documentation should also include information about how to structure and build the components in code. During the user studies, it was found that a framework for how to write the CSS was especially important to include in the guidelines since CSS can be written by both designers and developers.

In the long run, each component of the design system should have its own guidelines including the relevant content from the list above. This structure could be found among the mature design systems included in the benchmark. However, when a design system is in its early stages of development, the components are likely to change more frequently making component-specific guidelines hard to maintain. Therefore, general design guidelines without too many details and constraints are preferred at this time.

Component Libraries

The third building block of the design system is component libraries. At Universal Avenue, there will be three different types of global component libraries; one Sketch library, one React library, and lastly one Rails library. The Sketch library will contain graphical representations of the components to be used when building new design mockups. The code libraries will be used mainly by the developers when building the functionality of the product. The reason for having two code libraries, one in React and one in Rails, is because the product teams are using different code platforms. Each component in the React library will include a separate CSS-file which would allow for both designers and developers to more easily change the styling. The styling is what unifies all product teams at the company since they all use CSS to style their products. Based on insights from the user studies the following will also apply to the styling of components;

- The CSS-code should be decoupled from the functional code but still be located in the same repository.
- The separation should be done early in the process of building new components for the design system. Decoupling CSS from old components should not be prioritized.
- The team using Rails will have to build their own functional code library but will be able to use the styling from the global CSS-files.

The component libraries can, in turn, be divided into global and local libraries, where global libraries are defined as libraries that can be used by all product teams at the company. Local libraries, on the other hand, are libraries that only exist in a local product team. The long-term goal with the design system is to have one global Sketch library and one global code library to be used by all teams at Universal Avenue. Having global libraries from which components can be consumed by the product teams will decrease the diversifications in the products, which earlier was caused by different component-designs emerging from the local product teams.

A solid global library does not occur overnight, instead, components will have to be added over time. Building a global library can initially be described as a bottom-up process that must allow local components to emerge. Those local components will be pushed into the global libraries when tested and adapted to the needs of all product teams, this process will be described in more detail later in this chapter. The long-term goal with the global libraries is to establish a top-down process where components can be pulled from the global libraries into the local production. This pull-process aims to imitate the pull-system advocated by lean management principles, to create a more efficient production flow. The bottom-up and top-down processes are described in figure 14.

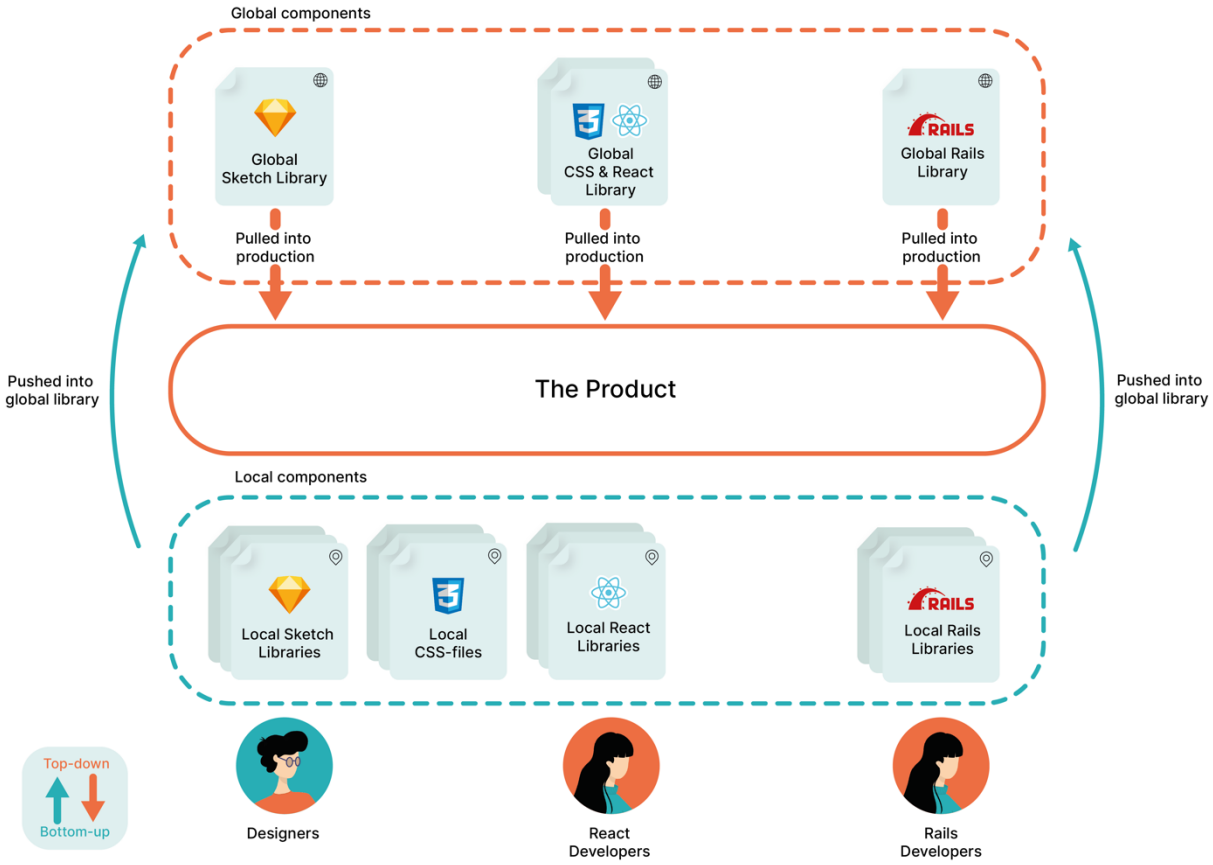


Figure 14. The bottom-up and top-down process concept for structuring component libraries.

Important to consider when developing coded components is that they should be reusable by several product-teams and contextual dependencies should therefore be taken into account. Examples of contextual dependency can be font sizes in buttons. The contextual dependency was especially debated by the developers during the user studies and the co-creation session. According to the developers, the functional components should be bare-boned and have few contextual dependencies so that they can be applied all over the product. This means that some modifications always will have to be made to a component when pulled from the global library into production. However, this alternative was preferred over having too specific components that would require too many variations. The balance of having context-specific and bare bone components are delicate. The goal with a global code library should always be to

decrease the developer's time spent on reinventing components. If the use of global components requires too much extra work the developers are likely to develop their own components on a local level, and the point of having a global library will be lost.

4.7.4 Maintenance and Documentation of the Design System

The maintenance of the design system was discussed with the designers and developers during the user studies. They expressed the concern of not having the time to both develop and maintain the design system and to perform their regular jobs. They suggested that a design system team should be appointed. To define some control mechanisms for maintaining the design system was also advocated in the literature. If putting together a dedicated team is not an option, one or a couple of individuals could be appointed to take some extra responsibility for the development of the design system at Universal Avenue.

A dedicated design system team should preferably consist of both designers and developers working at a global company level to moderate the design system. Their responsibilities would be to define and maintain the building blocks of the design system i.e. the design elements, design guidelines, and component libraries. They should also facilitate the process of transferring local components into global components and continuously release new versions of the design system.

Documentation of the design system has also been discussed with the designers and developers at Universal Avenue. Designers prefer to have the design elements and guidelines accessible directly in Sketch to speed up their design process. Developers seem to be more skeptical about having documentation in Sketch since it would imply an extra step in their process and would require a license before they can make changes in the Sketch-file. The ultimate solution would be to store the documentation related to the design system at a place accessible for both designers and developers. Online websites were the dominant platform for documenting design systems found during the benchmarking. Based on the discussions with the users at Universal Avenue, a website would be the optimal long-term solution. The downside with a website is that it requires effort to build and can easily take away the focus from actually developing the design system.

At Universal Avenue, Storybook could be an alternative solution to document the components in the early phase of the design system. Using Storybook would give both designers and developers access to the documentation. To minimize the hassle for the designers, a copy of the design elements and guidelines could be implemented in the design system Sketch library. To minimize the hassle for designers and developers, a copy of the documentation in Sketch can be imported into Storybook. However, this solution provides that someone is responsible for synchronizing the documentation in Sketch with Storybook and vice versa. This is crucial for Universal Avenue, to not end up in the same situation as before, where the components in Sketch and Storybook had diverged from each other.

Regardless of how the documentation is stored, it is important that elements and components have consistent naming. This would help designers and developers collaborate and lower the risk of miscommunication between the two. Consistent

naming would also allow for faster workflow. For example, during handoff when the developer starts implementing the design, they can look up the names of the components and find them directly in code.

4.7.5 Implementation of the Design System

The implementation of a design system is a crucial step since it forms the basis for how it is received by the people in the organization. Hence, much effort has been put into defining the process of implementing a design system at Universal Avenue. Inspiration gained from lean management together with the analysis of the benchmarking, user studies, and the co-creation sessions has contributed to a final concept of the process, see figure 15, which will be presented in more detail in the sections below.

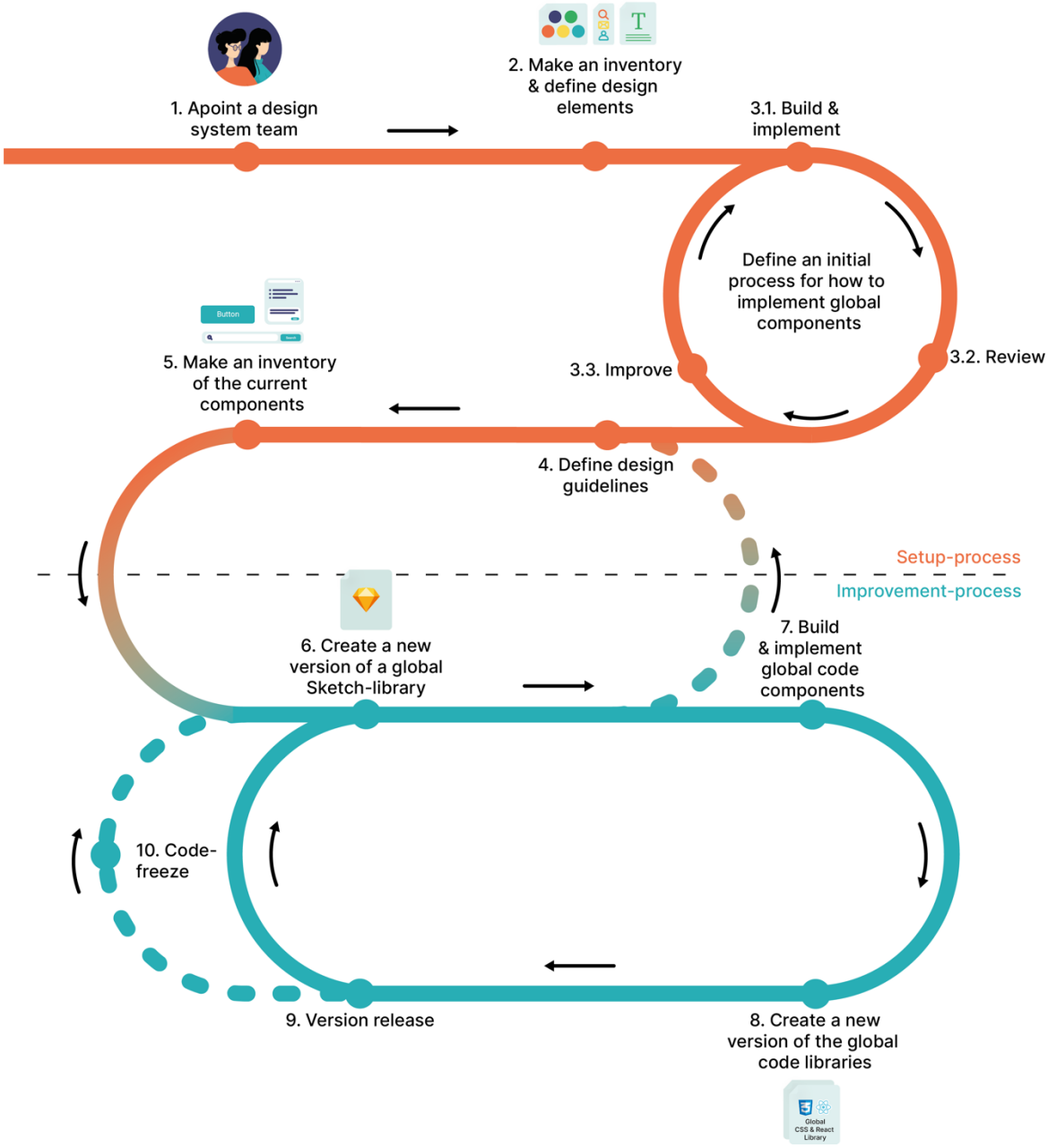


Figure 15. The final concept of the design system implementation process.

The implementation process contains a total of ten primary steps where some of them are defined with sub-steps and sub-processes, see figure 14. Step one to five represents a one-time set-up process that will be performed when first introducing the design system. Step six to ten represents an ongoing process that aims to describe how to build new versions of the design system. At an early stage, the focus will be to frequently add components to the global libraries to build the design system. In the long term, a new version of the design system will be released when there is a need for new or updated components. This process (step 6-10) will be repeated as a process for improving the design system along with the growth of the product and the company.

Step 1 - Appoint a team responsible for the design system
For a design system to not only be implemented but also maintained and used over time it needs a governing entity to ensure that it will stay relevant. As Universal Avenue is a rather small company it is suggested that they appoint a design system team of one or two persons that will take extra responsibility for the design system. These persons would then be responsible for the implementation process, which does not only include the creation of the system but also communication with all relevant stakeholders.

Step 2 - Make an inventory and define design elements
The second step in the process of initiating a design system is to make an inventory of the existing elements in the product. After the inventory, it is time to define the elements to be used in the design system. At Universal Avenue, this process has already started and the elements for the new design system have been defined as colors, typography, spacing, and icons. Colors and typography were elements brought over from their old design system and this study added spacing and icons. Defining the elements are made with input from designers and developers. The documentation of the elements should be stored at a place accessible by all users.

Step 3 - Define an initial process for how to implement global components
The process of implementing a new global component has been inspired by the lean approach Build-Measure-Learn described in chapter 3.5. In this case, the first step, (3.1) in figure 14, would be to decide upon one component to build and implement across all teams. These components should be small and rather simple so that iterations can be made quickly. This would be followed by a review (step 3.2) of how the implementation went out in all teams. Here it is especially important to document and evaluated how the implementation went in the teams with different code bases to make sure that the process being developed works for the whole company. Since all teams currently have their own way of writing and structuring the code it is likely that modifications will be needed for the component to be applicable in all product teams. Based on the review, the process will be improved to fit the needs of all product teams (step 3.3). This process (step 3.1-3.3) will have to be repeated until an initial process for how to implement global components have been defined.

Step 4 - Define design guidelines
The fourth step is to start defining the design guidelines. Important to note is that defining design guidelines have to be an ongoing process but can preferably be initiated after step 3 when the needs of the different product teams have been exposed. The guidelines are likely to change as the design system and the product evolves but to start

documenting the guidelines early in the process will help Universal Avenue to unify their designs. This will help in creating a consistent design language and end-user experience across all their products.

Step 5 - Make an inventory of the components used in today's product
The next step is to make an inventory of the components used in the current products. When the inventory is done for the first time, it can preferably be a workshop where designers and developers from the different product teams can discuss the components and how they might differ between the teams. If there is a need for continuous inventories, these could be managed by the design system team.

The inventory creates a good overview of where components are used and wherein the product the designs deviates the most between the teams and the code. For Universal Avenue to have documented which components that are currently used in the product also creates a good starting point for further development. The inventory can also indicate which components that should be prioritized when building the first versions of the design system based on number of uses across all teams.

Step 6 - Create a new version of the global Sketch library
After the inventory is made the new components need to be designed. The new designs are made in Sketch and can be worked on by designers from all teams, not necessarily the ones responsible for the design system. The process of designing global components will be described in more detail in chapter 4.7.6.

Step 7 - Build and implement global code components
When a new suggestion of a component has been designed in the global Sketch-library, the next step is to build the component in code. This process will be a cooperation between the local developers and the design system team. The design system team's responsibility will be to consider the local team's different needs and to make sure that the code can be used in all product teams. The process of building global code components will be described in more detail in chapter 4.7.6.

Step 8 - Create a new version of the global code library
When all components are reviewed and approved by the different teams it will be added to a new version of the global component libraries by the design system team.

Step 9 - A new version of the design system is released and ready to use by all product teams
The global component libraries will be released in versions, which means that a number of new components would be collected before being released in the next version of the design system. The aim is to allow each team to independently choose when to upgrade their codebase to the latest version. By deploying the design system in versions each team has higher control over their own project and the risk of introducing breaking changes are lower.

Step 10 - Clean up old components or code-freeze
This step is optional and might not be relevant for all teams or for each release of a new version of the system. However, the users of the design system at Universal Avenue did express the need for a so-called code-freeze to clean up old components and legacy

code. This can be done to create a better environment for development and to avoid conflicts that might appear between old components and the new design system components. After finalizing either step 9 or 10, the process starts over from step 6.

When developing new components, conflicts might appear with the current design guidelines. If there is a need for adding new or updating the old guidelines it should be done before finalizing a new version of the design system. When changing the guidelines, it is important to make a new inventory of the components already implemented in the products to secure that those are following the new guidelines. If updates of old components must be made, those should preferably be included in the release of the new version to avoid inconsistency in the product. This process has been illustrated with the dotted arrow between step 6 and 4 in figure 15.

4.7.6 The Process of Building Global Components

Found during the user studies was that the designers experienced a lack of connection between the graphical components and the coded component that is stored in the code libraries. If a designer made a new component that did not exist in code, the design would take a longer time to implement since it has to be built from scratch by the developer as well. This problem was a result of having low control over the components that currently existed in their design system. As a response to this problem, a process for building global components have been suggested. In the implementation process described in the previous section, step 6 and 7 briefly described how global components were created. The process of building global components will now be described in more detail and the process is illustrated in Figure 16.

The process starts with a designer beginning on a new component in Sketch (6.1). This can either be done by one of the product designers or a designer in the design system team depending on where the need for the component appears. Preferably, some developers will be included in the process of defining a new component as well. The design is then reviewed by both the design system team and designers in the local teams (6.2).

As soon as the foundation of the new component is defined, a developer can start creating the functional part of the component in parallel with the design process (7.1). Step 7.1 also includes creating a CSS template with minimal CSS used for testing purposes. When the design has been set by the designer, the styling of the component (7.2) can be finalized by either a local designer, local developer, or someone in the design system team. The following step (7.3) is to review the styling so that it looks the same as the design and that the code follows best practices and are structured in a sensible way. The review will follow the usual review process currently performed in the product teams and aims to secure that the quality of the code meets the guidelines.

After the component is reviewed it is ready to use by the local team that has been part of the development process (7.4). At this stage, the design system team takes over again and is responsible for making the new component compatible with the rest of the product teams (7.5). When those steps have been passed, the component is ready to be released in the global libraries (7.6).

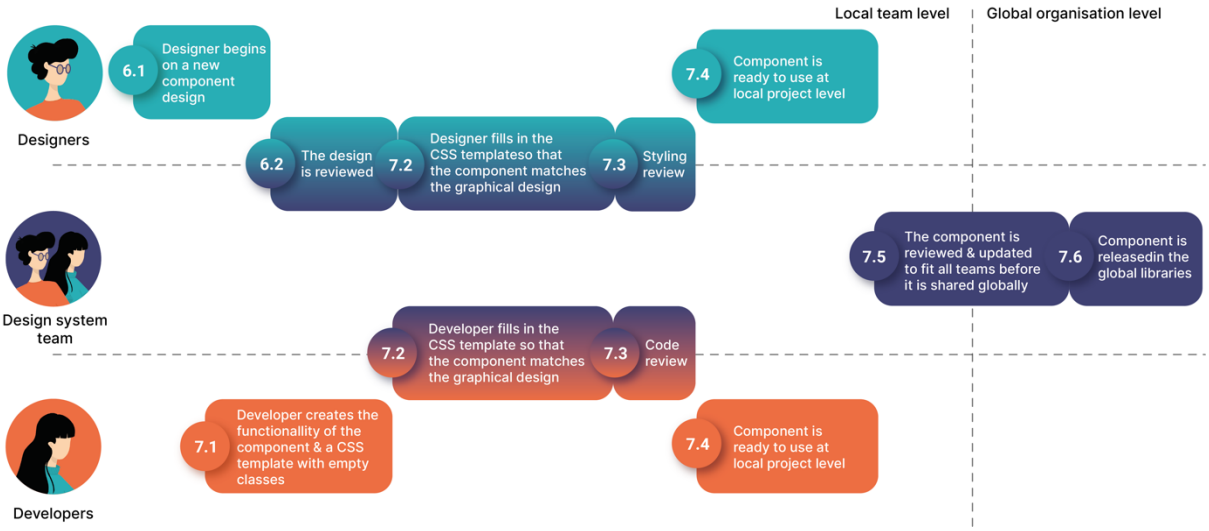


Figure 16. Process for implementing a component in the global code library.

4.7.7 Summary of the Design System Concept for Universal Avenue

The following list is a compilation of the design system concept presented in chapters 4.7.1 - 4.7.6.

- The goal of the design system is to make the development process more efficient by providing global component libraries to be used by all product teams at the company.
- The main building blocks of the Universal Avenue design system are; design elements, design guidelines, and component libraries.
- The design elements include; color, typography, icons, and spacing.
- The design guidelines include; examples, do's and don'ts, typography, spacing, patterns, accessibility, user behavior, and code structure.
- The component libraries include a Sketch library and two code libraries with separate CSS files.
- A dedicated design system team should be appointed and be responsible for the maintenance of the design system.
- The design system needs to be documented and the documentation should be accessible by both designers and developers.
- The implementation process includes a one-time setup process and a process for continuous improvement.

In chapter 4.6 User profiles and guidelines for how to design a design system for Universal Avenue were defined. The guidelines were compared to the concept of the design system that has been presented in chapter 4.7.3 - 4.7.6. The designer and developer guidelines are presented in table 4 and 5 respectively. Accompanying each guideline are comments on how they have been considered in the concept of the design system.

Table 4. Designer guidelines for how to design the design system for Universal Avenue and how they have been considered in the concept.

Designer Guidelines	Fulfilled	Comment
Be up to date and display which elements, components, and patterns that are used.	Yes	A process for maintaining the design system has been suggested. Displaying the used components in one place is a long term goal of the design system and something that will be implemented on a future website.
Split up the code into more manageable chunks with less nested dependency and allow both designers and developers to access the code.	Yes	This need will be met by separating the CSS styling from the functional code by introducing a separate CSS file for each component located in the repository.
Include guidelines for how to address design problems and what the overall design of the product should express.	Yes	In the concept, a framework for what the design guidelines should cover has been presented. However, the design guidelines have to be defined in more detail and this should be done while the design system is developed.
Be shareable between the different product teams.	Yes	The implementation process and the process of building global components have been designed to support the development of sharable, or global, components that can be used by all product teams.
Have one person or team in charge of the maintenance of the design system.	Yes	A design system team responsible for the maintenance has been suggested in the concept.
Allow notification for all stakeholders about updates or changes in the design system.	Yes	By introducing version releases of the design system, notifications for all stakeholders will be possible.

Defining a Design System Concept for Universal Avenue	Partly	The suggested process for implementing global components will ease the synchronization of graphical and coded components. Hence, the need for comparing the components will be lowered.
Include instructions for how to write the code in a correct hierarchical structure that allows global changes in the design of the product.	No	Not considered in the concept.

Table 5. Developer guidelines for how to design the design system for Universal Avenue and how they have been considered in the concept.

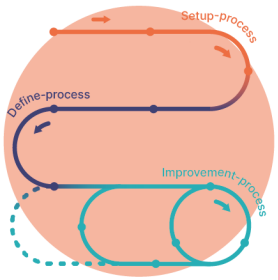
Developer Guidelines	Fulfilled	Comment
Unify the teams to follow common guidelines for how to code a component.	Partly	In the concept, a framework for what the design guidelines, including code guidelines, should cover has been presented. However, defined guidelines for how to code the components have not been included in the concept.
Make it easier for the developer to understand what impact a change in the code will have on the overall product.	No	Not considered in the concept.
Make it easier for designers to use components that have a coded counterpart.	Yes	In the concept, all components should have a coded counterpart to be considered global.
Allow developers to spend most of their time on development and minimize the time spent on administrative work.	No	Hard to evaluate, still a goal with the design system.
Lower the dependency between styling and functionality.	Yes	Achieved through separating the CSS in its own file in the repository.

Have a global dependency on elements.	Yes	Global dependence on elements has been suggested in the concept.
Include a global code library.	Yes	A global code library has been suggested in the concept.
Support a code structure that separates the styling of a component from its functionality.	Yes	To separate styling and functionality has been suggested in the concept.
Allow creative freedom.	No	Not considered in the concept. Hard to measure.
Make it easier to see what components are used in the live product.	Yes	All components in the global libraries will be represented in the product.
Provide “ready to use” components.	Partly	The concept is not providing ready to use components, but the finished design system will.
Be easy to maintain and update.	Yes	A process for how to maintain and update the design system has been suggested in the concept.
Provide guidelines for how developers can “consume” design.	No	Not considered in the concept.
Allow developers to focus more on implementation and the designers to focus on the styling and UI parts.	Partly	Have to be evaluated over time but the conditions for that change to happen have been met with the inclusion of a separate styling file.

05.

Implementation Support Framework

The study and design process performed at Universal Avenue have generated knowledge and insights on how to design and implement a design system in a particular context. From the context-specific design system and implementation process at Universal Avenue, it is believed that a more general concept can be developed.



The insights from the study, together with a compilation of the literature about design systems, have resulted in a concept of an implementation support framework (ISF). The ISF is intended to work as guidance for other companies that want to implement a design system in their organizations. In this concept, much of the company-specific requirements are removed in favor of applying more general requirements to widen the use cases of the concept. The ISF is a summary of the methods and design process performed in this constructive design research. In addition, insights from the literature review and the benchmark will complement the research and add additional support to create a generalized concept for how to implement a design system.

The ISF (figure 17) contains three main activities; Setup process, Define the content and start building the design system, and Improve and maintain the design system. In the following sections, the ISF will be described in more detail.

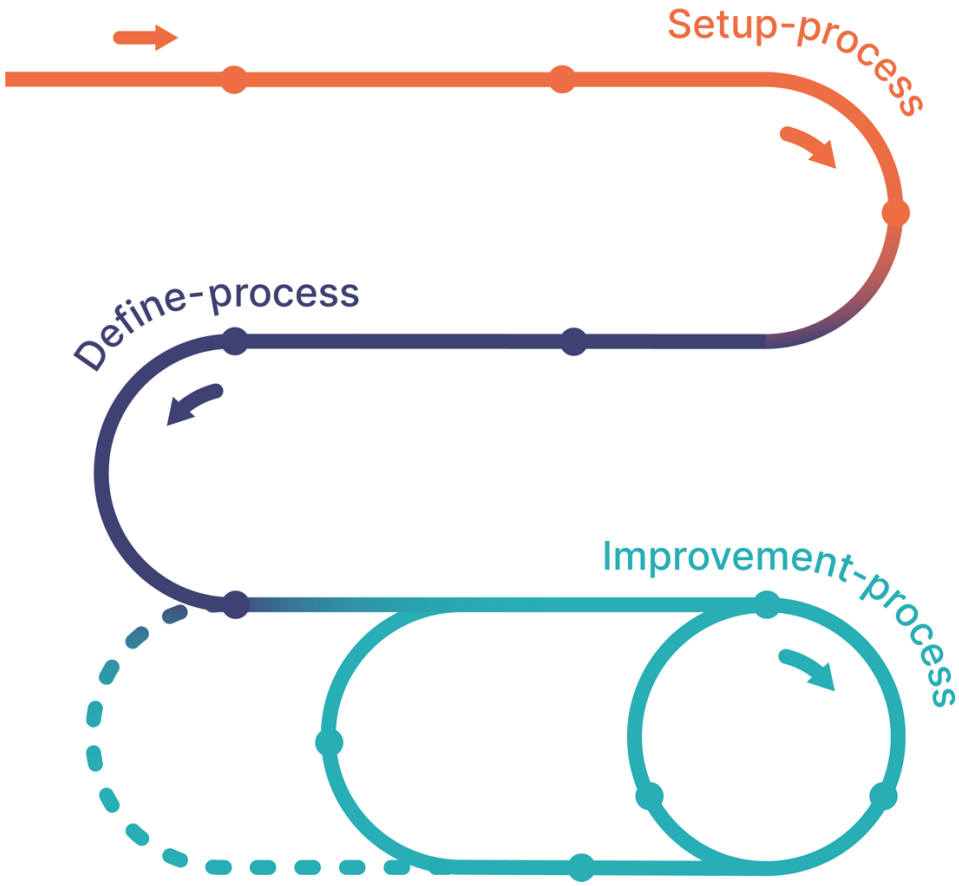


Figure 17. Visualization of the implementation support framework.

5.1 Setup Process

The first activity in the ISF is called the setup process. It contains three main steps, see figure 18, that will help the organization getting started with the design system. The three steps will be described in more detail below.

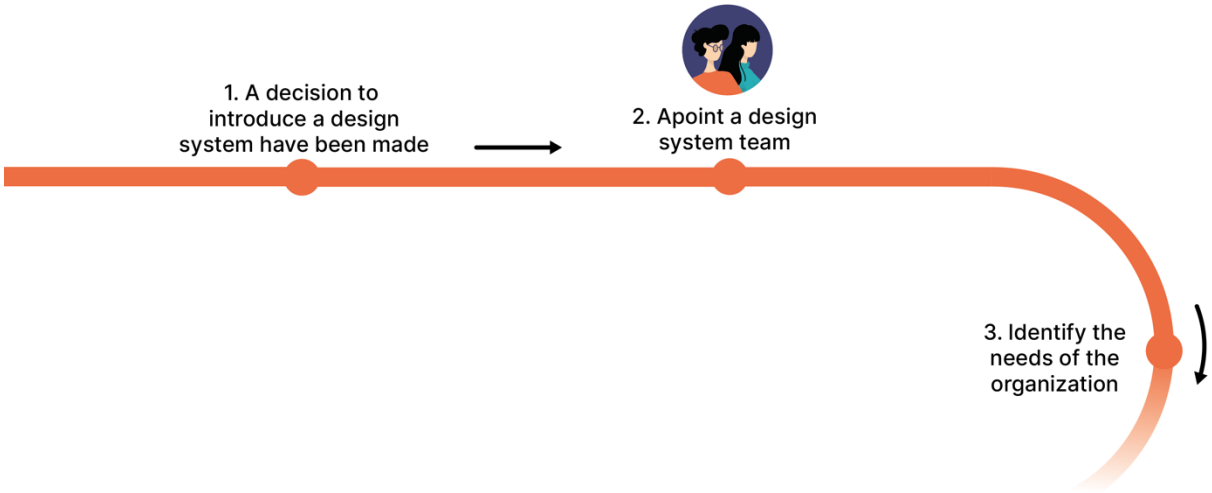


Figure 18. The setup processes.

5.1.1 Decision to Implement a Design System

Before the implementation is started, the company must commit to the building of a design system. This is an important step to think through and the company must be aware that the implementation of a design system will require some effort to both introduce and to maintain. By committing to the implementation, the company must make sure that the necessary resources can be allocated. Chapter 3.4 provides examples of when a design system could be especially beneficial to introduce in an organization.

5.1.2 Appoint a Design System Team

A design system could be compared to a product for internal use and would benefit from having a dedicated team overseeing its development. Also, according to the definition, which was described in the literature review, a system will need control mechanisms to stay relevant and up to date. Hence, responsibility for the design system should be appointed early in the implementation process.

5.1.3 Define the Needs of the Organization

During the creation of the design system framework at Universal Avenue, it became apparent that there are some specific insights, related to the context in which the design system will be implemented, that would increase the chances for a successful implementation. Those insights were related to the current working process and culture at the company e.g. to have great insights into the tools and platforms that are currently in use and which attitude the users have could help when defining the new design system.

The literature advocates some activities that could lay the groundwork for a successful implementation and those activities are; to map the current working process (Devanney, 2017), identify the needs of the users of the design system (MacDonald, 2019), identify the problems related to the current working process, and define the goals of implementing the new design system (MacDonald, 2019). By examining those areas, a foundation for building the new design system can be achieved.

In table 6, the four suggested steps for defining the organizational needs are presented. The methods presented in the table are a selection of methods that can be used as inspiration for collecting data about the processes and the users.

Table 6. Steps in defining the organization's need for a design system.

What?	How?	Why?
Map the current working process.	● Interviews ● Observations ● Documentation ● Questionnaires	To expose the problems and identify potential improvements. This step is especially important if the design system is developed by someone with an external background, for example, consultants.
Identify the needs of the design system users.	● Interviews ● Observations ● Questionnaires	To gain insights about what to consider when designing the design system. Defining the user's needs will help to prioritize what to include in the design system.
Identify the problems related to the current working process.	Analyze the data collected in the previous steps.	Defining the problems of the current working process will help to prioritize the building blocks of the design system.
Define the goals of implementing a design system.	Look at the most distinctive problems and define the goals to solve those problems. Try to be as specific as possible.	Defining the goals with the design system will help to stay on track and will make it possible to measure the progress.

5.2 Define the Content and Start Building the Design System

When the needs of the organization have been identified, the next step is to define the content of the design system. During the literature review, it was found that the most common building blocks of a design system are; elements, components, style guides,

libraries, and guidelines. The benchmarking also generated a complementary understanding of the most common building blocks of design systems.

Based on the result from the literature review and the benchmarking performed in this study, the fundamental building blocks of a design system have been summarized into design elements, design guidelines, and component libraries. The structure of a design system, at least in an early stage, does not need all the different building blocks brought up in literature. Instead, it is better to focus on a few and get them right before potential expansion can take place. Important to note is that there might also be other potential building blocks to consider than the three that have been suggested, depending on the needs of the organization.

The process of defining the content and start the building of the design system consists of four different steps presented in figure 19 and elaborated upon in the following sections.

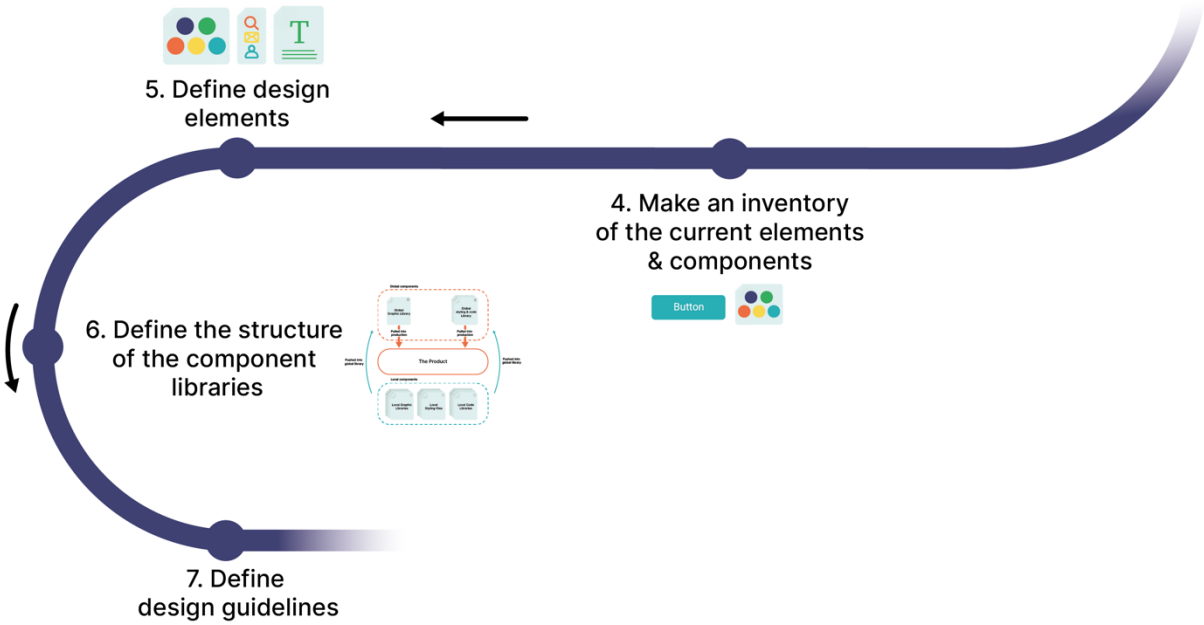


Figure 19. Define content and start building the design system.

5.2.1 Inventory of Current Elements and Components

A great way to start the process of defining and building content for the design systems is to perform an inventory to identify which elements and components that are currently in use (Trender, 2017; Vesselov & Davis, 2019; & Hacq, 2019). This activity can help to decide which elements and components to prioritize in the first iterations of the design system. An inventory can also help in evaluating if the current elements contribute to a unified product, or if they need to be improved to live up to the goals of the new design system.

5.2.2 Define Design Elements

In the literature review, design elements were defined as the smallest piece that builds up a component. Defining the elements is a relatively easy way to create variables that can be shared between all teams in the organization. The elements will also work as a great starting point for designing global components. Data from the benchmarking shows that the most common design elements used in the open-source design systems were; color, typography, spacing, and icons. Those elements will serve as a great starting point, and many components can be created by only using those four elements. As the design system grows, more elements can be added to meet the needs of the organization. See figure 8 for more inspiration.

5.2.3 Define the Structure of the Component Libraries

The literature review described two types of component libraries, graphical libraries, and code libraries. Those are used to store either graphical or coded components so that they easily can be reused during the design and development process. However, the problem of having both graphic and code libraries is that there is currently no solid solution developed to facilitate the synchronization between the graphically represented components and the coded ones. To avoid having divergent libraries, this chapter will suggest a way to structure the libraries so they will be easy to maintain and synchronize.

At Universal Avenue, three global component libraries were presented in the design system concept. One library for graphical components created in Sketch and two code libraries because they use two different codebases to build their product. During the design phase of this project, much emphasis was put on defining the process of adding components to the global libraries to allow its users to control its content and to secure that all three libraries remained consistent. This resulted in a bottom-up and top-down process that has been described in chapter 4.7.4. This process has been generalized and presented in figure 20. During the study, it became apparent that local component libraries must exist to avoid prototyping in the global libraries, which would affect the whole product. Instead, the design system must allow local prototyping, which can push new components into the global libraries when so is needed. Hence, it will only exist components that have been tested and approved by all teams in the global libraries. Those global components are then ready to be pulled directly into the product by the different product teams.

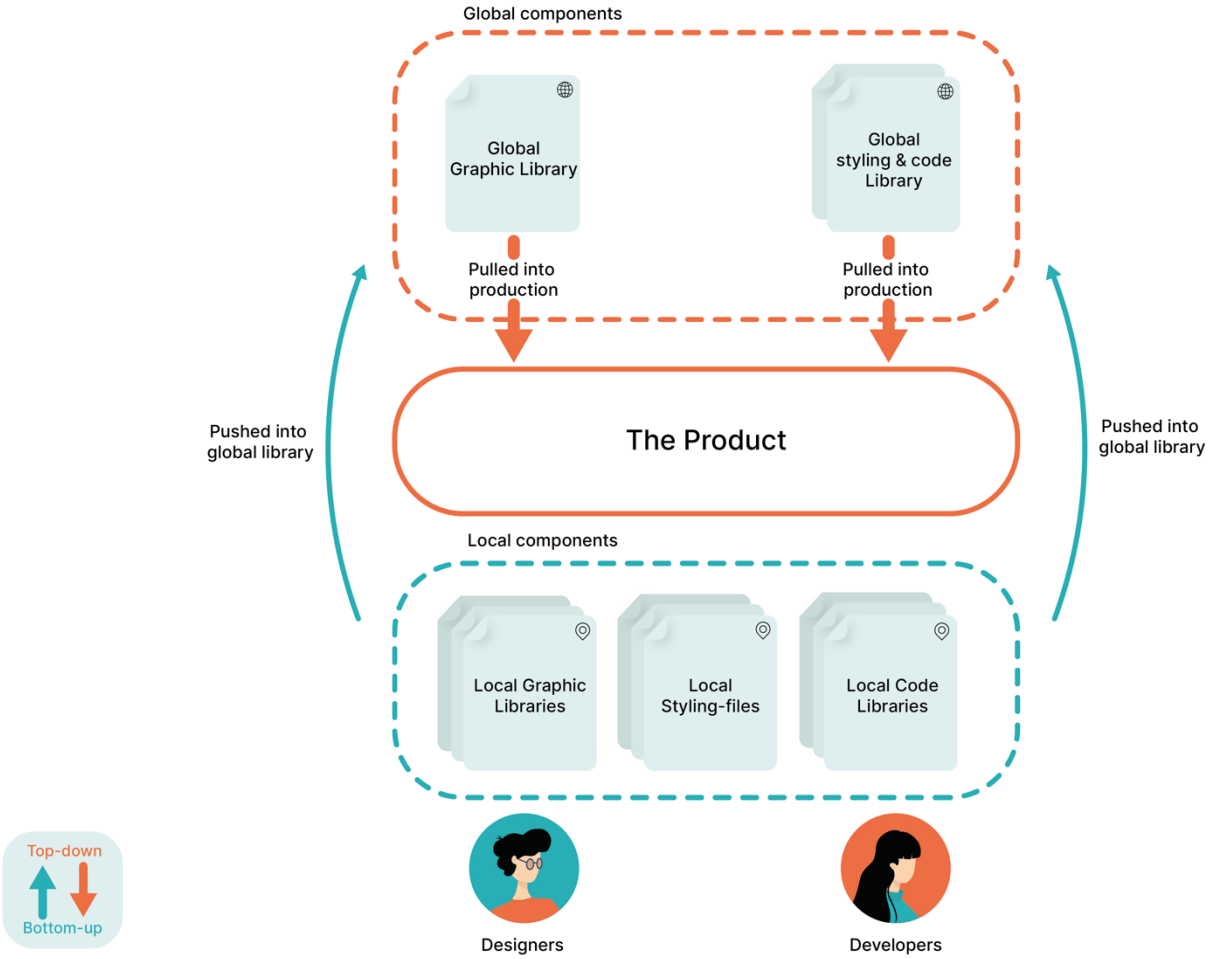


Figure 20. A generalized representation of the bottom-up and top-down process for building global component libraries.

Based on the insights gained during the study, the following list presents some things that can be worth to consider when adapting the bottom-up and top-down process in other organizations:

- Examine how the current components are being built today. Are they represented both graphically and in code?
- Is there any existing mechanism for how to keep the graphical library in sync with the code-library?
- Are the current libraries global and can they be accessed by everyone in the organization? Or do local libraries exist?
- If local libraries exist, is there any mechanism in place that facilitates the transfer of local components to the global libraries?

5.2.4 Define Guidelines

In the literature review, guidelines were described as instructions or guidance for how to design and code. The benchmarking investigated which design guidelines that were used most frequently, and those were;

- Code examples
- Examples
- Typography guidelines
- Accessibility guidelines
- Icon guidelines
- Voice and tone
- Do’s and don’ts
- Best practices
- Spacing guidelines

In the literature review, guidelines were divided into three subcategories; design, pattern, and code. Depending on the needs of the company, those guidelines could be divided and extended into different parts of the design system to be relevant for that specific organization. The open-source design systems that were investigated in the benchmarking did have different approaches for how to write and categorize their guidelines. Some of the design systems had separate documentation for the guidelines, while some had written the guidelines in relation to each component.

In the study at Universal Avenue, the guideline categories were defined through the co-creation sessions that were held with two designers and two developers at the company. Regardless of which approach that is used to formulate and define the design guidelines, it is important that all stakeholders of the design system are involved in the process (Lopes, 2017). If the designers and developers are included in the process of defining the guidelines, they will be more prone to follow the guidelines in their daily work.

5.3 Improve and Maintain the Design System

In the third part of the ISF, the process for improvement and maintenance of the design systems will be defined. The literature on lean management was used as inspiration when developing the implementation process. Lean principles advocate standardized processes to ensure high-quality with minimal waste (Liker & Meier, 2006). The aim of the process suggested in this project was hence to create a standardized approach to create new versions of the design system, represented by step 4-10 in figure 11. The iterative part of the process designed for Universal Avenue has been working as inspiration for the general process described in this chapter.

In the literature, MacDonald (2019) and Hacq (2019) also stressed the importance of implementing the design system as an iterative process, where the most-used components should be prioritized in the first iterations of the design system. Those ideas have been brought into this third part of the ISF as well, and this chapter will describe the three steps that will be included in the process of improving and maintaining a design system, see figure 21.

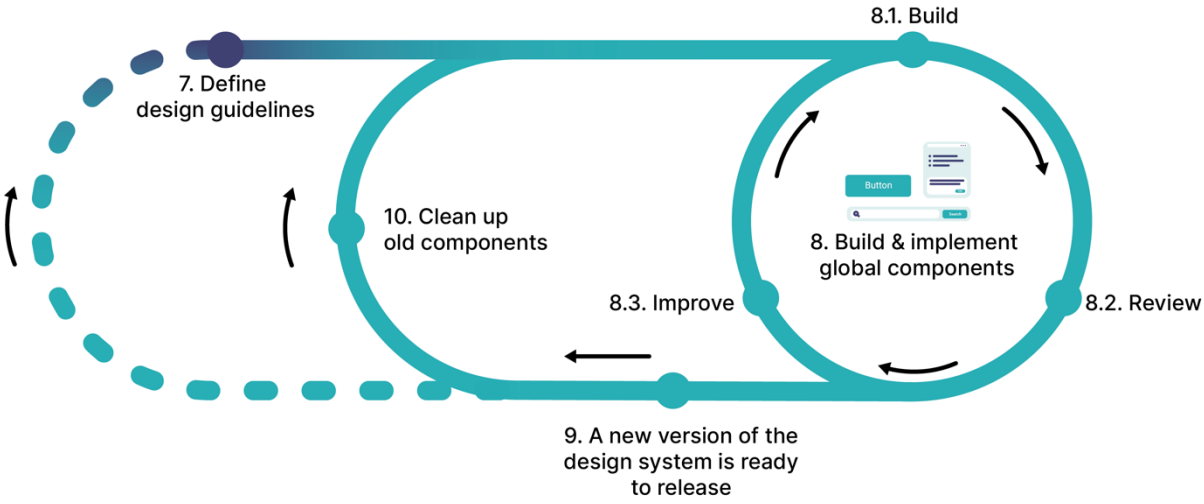


Figure 21. Improve and maintain the design system.

5.3.1 Build and Implement Components to Global Libraries

The first step of improving and maintaining the design system is described by a three-step process illustrated as step 8 in figure 20. This process has been inspired by the Build-Measure-Learn feedback-loop that have been described in chapter 2.5 in the literature review.

In the ISF, the loop starts with an MVP of a component being built and implemented into the global library. This can be done either by the design system team or a local team. Then, the component will have to be reviewed by designers and developers according to how well the component suits their needs when developing the product. Based on the feedback from the users, the component will be improved to fulfill their needs. This loop will continue until the component is approved by all and ready to be implemented in the global library.

5.3.2 Release a New Version of the Design System

When components have been built and implemented in the global libraries through the feedback-loop, a new version of the design system can be released. The benefit of implementing the design system in different versions is that it gives each team more control. If the new version of the library contains components that are not relevant for a specific team, they can wait with updating until the next version is released. By being able to either wait or skip new releases they are not forced to adapt to new components that might break components that are already in the production.

The initial versions of the design system that is built through the suggested process should focus on the elements and components that were identified as most important during the inventory. In the long term, when the cornerstones of the design system are in place, the new versions of the design system that are created through the process will rather focus on minor changes and updates to maintain and support the constant improvement of the design system.

5.3.3 Clean Up Legacy Code and Repeat

As advocated by lean principles, the amount of waste in the process should be minimized and having a lot of legacy code in libraries has a tendency to slow down the development process. The last step in the process would hence be to clean up the old components in the local product teams as some code will be out of date as the design system evolves. Even if the clean-up is a strong recommendation, it can be considered as an optional step that might not be necessary after every new release of the design system. It will have to be up to each organization to decide where to direct its focus.

After finishing step 10 in the ISF and a new version of the design system have been released, the process will repeat itself from step 8 where a new MVP of a component is built, reviewed, and improved. Depending on the feedback received in this loop, it might be necessary to go back and add additional information or update the guidelines to keep them up to date with the design system. This step has been illustrated with a dotted line in figure 22, which is a composition of all the steps in the ISF.

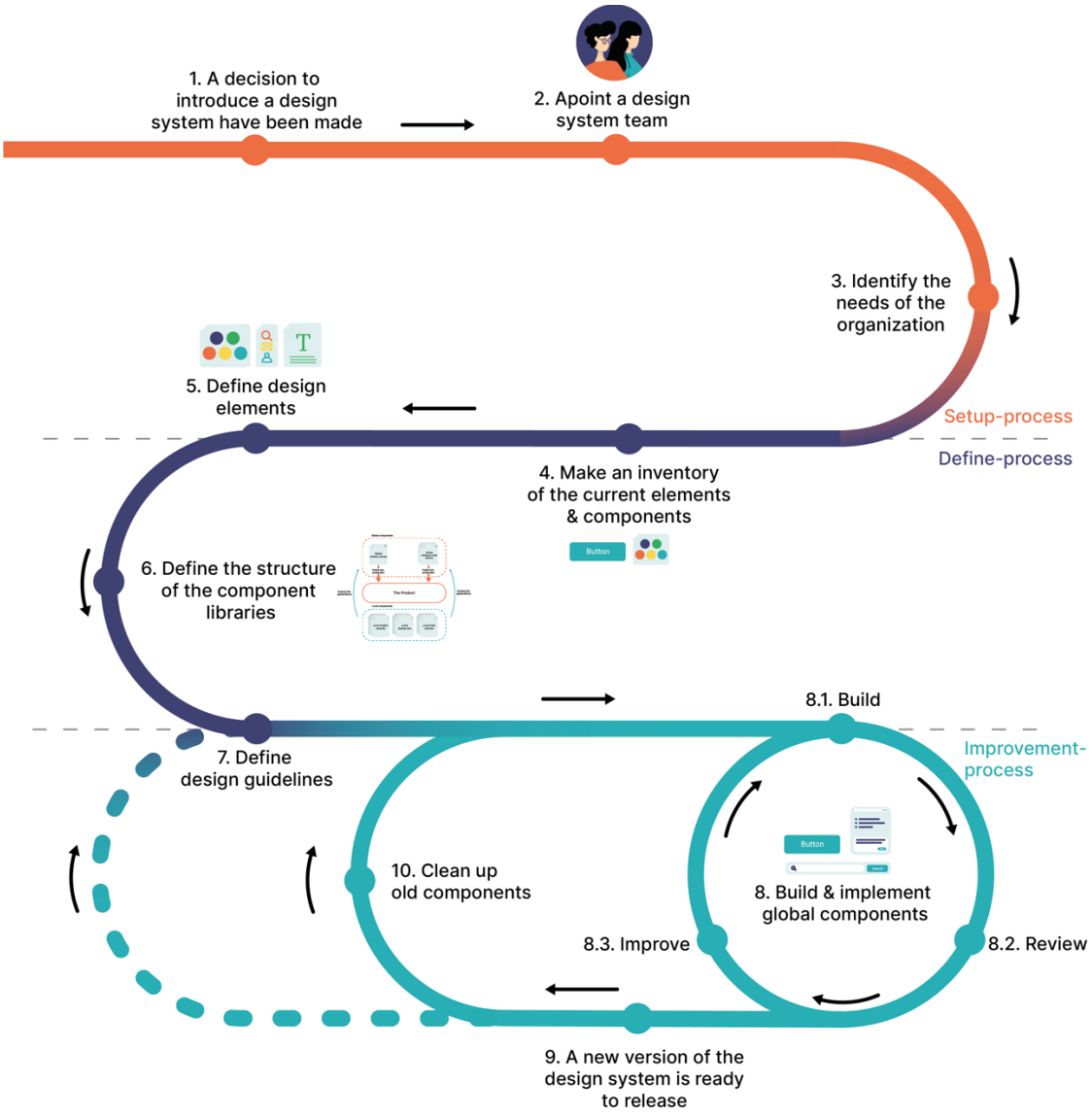


Figure 22. Illustration of all the steps in the implementation support framework.

06.
Discussion

The discussion brings up the constructive design research as an approach for researching design systems, together with a review of the methods that have been used. This is followed by a discussion regarding the context-specific concept designed for Universal Avenue and its generic counterpart, the ISF. Lastly, ethics and sustainability in relation to design systems will be discussed.

6.1 Constructive Design Research on Design Systems

The benefits of constructive design research is that it facilitates the collection of knowledge and insights at the same time as new concepts and designs are developed. This approach allow freedom in the exploration, which has been highly valued during this study since the topic of design systems is very broad. The sub-approach of the constructive design was field studies, which made it possible to study design systems in a relevant context and with real users. The study generated insights that could not have been gained through e.g. just benchmarking and literature studies. This allowed a much deeper understanding of the users and their needs, as well as enabling theories from the literature to be tested in reality.

This project has been focusing on how design systems should be implemented to be beneficial for its users, in contrast to what could be found in the literature, where the focus was on the content of the design system. Design systems found in literature was often explained insufficient and at a high level and did not cover much of the details regarding its implementation.

In the literature review, it also became clear that the current research on design systems for web development was lacking clear definitions and standards for both naming and content. The dominant argument was that as long as the naming was consistent in the company it did not matter what the building blocks was called. This study shows the same, that the most benefit can be achieved by applying the same naming throughout the company. However, by discussing the naming with the designers and developers it became clear that choosing names based on their mental model would lower the learning curve. If the naming problem is viewed from a broader perspective consistent naming introduced by design systems could be advantageous for the whole industry of web development. It would hence be much easier to learn and adapt from each other if naming and conventions were the same throughout the industry. Although, it might be hard to achieve a standard as design systems are highly context reliant and varies greatly between companies.

6.2 Adapting Existing Tools to Design Systems

The card sorting method resulted in many insights regarding the actual working process at Universal Avenue. Especially after the data was analyzed and compiled into a relation map that highlighted connections not found by plain observation. The reliability of the results from the card sorting could be considered as rather low as the results from a participant probably would not be the same two times in a row. In hindsight it might also have been beneficial to add axes to the card sorting making it easier to compare the scores between all participants by giving them a frame of reference while doing the sorting. However, the main purpose of the card sorting method was now to map the exact relation between the cards, but rather to facilitate the discussion about the current working process. When performing the card sorting a number of key areas were prepared beforehand and printed on notes. These areas, although based on the interviews, might have affected the outcome of the study by

making it harder for the participants to think outside the box. Each key area was only represented once on the notes, which made the participants adapt their sorting to the available notes instead of creating new ones. A number of notions were not represented among the cards, which also might have affected the result as the participant would have had to add them themselves.

Benchmarking was a good method to gain a general understanding of current best practices in the industry. Unfortunately, the benchmarking did not allow for any deeper understanding regarding why the systems had a particular structure and how the systems were implemented in the organizations over time. Some insights could be gained by comparing the more mature design systems with the newly created ones. For example, the more mature design systems often had more examples and more detailed documentation of their components than the less mature ones. But the data collected was from a small number of design systems and might not represent the whole, and hence general assumptions should be avoided.

The co-creation sessions were held remotely and needed a lot of material to be prepared beforehand to proceed smoothly. The participants were free to make their own material during the session, but it was a strong overrepresentation of the prepared material in the final creations. The solutions ideated during the session, even if the session was somewhat limiting, was of great importance for the development of the context-specific design system framework and by extension also for the more general ISF. The co-creations were held with two pairs, consisting of one designer and one developer. This constellation generated interesting discussions as they sometimes viewed the same problem from two different perspectives. This forced them to argue more about their role-specific needs during the ideation of the concepts that they developed.

The validity of the performed user studies is rather low as the number of participants for each study was a maximum of six. How much this affected the results is hard to say, but it can be assumed that the studies do not perfectly reflect reality.

6.3 A Design System Proposition for Universal Avenue

The design system framework that has been designed for Universal Avenue is a proposition for how the design system should be implemented to be beneficial for the designers and developers in their particular context. The concept has been presented and discussed with both designers and developers, however, the concept has not been tested in practice due to the limitations of this project. Further testing and evaluation of the concept will hence be needed to validate the applicability and it is likely that new iterations of the concept will be needed. Even though the concept has not been tested in practice, it is still based on the insights and needs of both the designers and developers at the company. Hence, it will still be able to serve as great guidance for Universal Avenue when implementing their design system.

What also needs to be investigated further is how to measure improvement in the product development process as a result of the new design system. As said, the aim of a

design system is to speed up the development process, make it more efficient, and make the product design more consistent. It would hence be interesting to measure those parameters in the process to be able to compare the effect of the design system. One way to measure the speed of the process could, for example, be to calculate the average time it takes for a designer or developer to finish a story, before and after the implementation of the design system. Hopefully, that time would decrease after the design system has been implemented. Something that is harder to measure but still one of the more valuable outcomes of a design system is the design consistency. Generally, design systems help companies and organizations unify their design language and allow them to focus more on the end-user experience. User experience is possible to measure before and after the introduction of the system, but it will be harder to show that the change is a result from its creation and not just a result of a more collaborative workforce.

In the literature review, lean management was brought up as an inspiration for the development of the design system. The lean principles that were presented have been used when defining the process of building global components at the company. The aim was to imitate a pull-process of global components to avoid an overproduction of components. Another lean principle that was applied more generally was the idea of standardizing best practices to advocate the constant improvement of the design system. This principle was applied in the implementation process where a standardized and circular process was suggested for building new versions of the design system. One part of the hypothesis is that a standardization of the process for building global components will speed up the building of the whole design system, and also that the process could be able to serve the constant improvement of the design system over time.

The lean principles also stress the importance of having a human-centered approach when developing production processes. It is reasonable that this approach can be applied to the implementation of design systems as well. In this project, much emphasis has been put on understanding the organization and its needs, especially the needs of the designers and developers who have been considered as the primary users of the design system. Insights from this study have resulted in an even greater understanding of the importance of involving the users when developing new processes. Someone leading the implementation of the design system is, of course, a reasonable assumption, but the potential risk of applying a traditional hierarchical approach is that the designers and developers would feel less ownership of the design system. If, on the contrary, the designers and developers have an active role and much influence on the development of the design system it is reasonable to think that they are more likely to accept it and use it in their daily work. Therefore, this project advocates a more collaborative approach when building the design system.

The discussion regarding the design system proposition for Universal Avenue is summarized below;

- The concept developed for Universal avenue will need further testing, but it could still serve as guidance for their design system implementation.
- The concept has been evaluated together with designers and developers.

- The proposition does not include any approach to measure the potential improvement generated by the design system, this should be investigated in the future iterations of the concept.
- The process of implementing the design system has been inspired by the push and pull strategy, together with the standardization and human-centered approach for how to design processes, which all can be found in lean management.

6.4 The Implementation Support Framework

One of the primary focuses of this study was to use constructive design research to investigate how design systems could be implemented in a specific context, and through those insights suggest a general framework for how to perform the implementation in other companies. As in the case with the concept for Universal Avenue, the ISF has not been tested in practice. This must be done before any conclusions about its accuracy could be made. Similar studies will need to be performed at other companies to collect data and insights that could be compared to what was found at Universal Avenue. Insights from other contexts could strengthen the generalizability and applicability of the proposed ISF.

Universal Avenue is a relatively small enterprise with few employees in relation to medium and large enterprises. It is hence reasonable that those larger enterprises will have different needs of a design system, and that it will have to be designed and implemented differently from what has been proposed in the ISF. Also, the users of the design system might have different needs as the organization gets larger and communication between different teams becomes more challenging. Literature suggests some methods that have been included in the ISF that can be performed regardless of the size of the company, e.g. doing an inventory of the current product and identifying the needs of the organization. This speaks for that at least some parts of the ISF could be applicable in different types of organizations.

6.5 Ethical Considerations of Design Systems

This chapter will discuss some of the ethical issues that have been identified in relation to the implementation of the design system. The discussion focus on what effects this study has on the primary and secondary users of a design system and what ethical issues it might entail.

In this study, designers and developers have been considered as the primary users of the design system. However, they are not the only users that will be affected by a design system. The management and sales departments at Universal Avenue will also be affected since it will impact the product development process and imply changes in the product that is delivered to the end-user. The management department is controlling the product development process and that the development follows the objectives and key results (OKR). The implementation of a design system is a long-term investment that will require some effort and sacrifice in the near time, which might imply that the OKRs have to be changed in favor of the design system. One ethical issue with this study is hence that the need of the management and sales department have not been

considered when developing the design system framework for Universal Avenue. On the other hand, this study is just presenting a hypothesis for how a design system could be implemented at the company and it could advantageously be used as a basis when making the decision of implementing a design system or not.

The implementation of a design system will also have an impact on the primary users even if their opinions have been considered in this study. Some of the participants in the user studies stressed the importance of creative freedom in their work. One of the challenges with design systems is that they will imply more structure and rules which could intrude on the experience of freedom among the designers and developers. When developing the concept, attempts have been made to increase the experienced freedom by giving the designers and the developers an active role in the design system. The bottom-up process that was included in the concept will give the developers and designers a chance to impact the components transferred into the design system. By increasing the primary user's influence over the design system, they will hopefully feel less limited by its structure. There is also the argument that the creativeness is not lowered by introducing a design system, only that the outlet changes from the micro to the macro perspective.

The ethical issues discussed above have primarily been focusing on the dilemmas for the users. Due to time constraint and the low severity of the ethical aspects the impact on secondary users, for example, managers and salespersons, have been left for future projects to explore. Even though the ethical repercussion was evaluated as low during this project it does not mean that it will be the case when other design systems are implemented. Hence ethical aspects should always be evaluated on a case to case basis.

07.
Conclusions

This chapter will provide the main conclusions of this study as well as the answers to the research questions.

This study aimed to answer the following research questions;

- What are the designers' and developers' needs regarding a design system?
- What should the design system contain, and how should it be implemented to meet the needs of, and be advantageous for, both designers and developers in the development process?

Based on the results of this constructive design research, conclusions could be made regarding the designers' and developers' needs of a design system, and the most important needs are listed below.

- Both designers and developers wanted the code to follow the structure of a single source of truth, i.e. local diversifications of components should be avoided.
- The designers wanted to have greater access to the code, e.g. to be able to make global changes in the styling of components.
- Both designers and developers requested guidelines to support the process of building and implementing global components.
- The designers needed to be able to change design elements in the product all at once, i.e. the styling should not be defined in local files.
- Both designers and developers requested a design system which facilitates the synchronization of graphical and coded components.
- Developers wanted the design system to provide coded components were both the functionality and styling have been tested and approved, i.e. the design system should provide components that are easy to implement directly into the product.
- Both designers and developers requested principles or for the maintenance of the design system.
- Both designers and developers wanted a design system which would advocate and allow cross-disciplinary collaboration.

The second research question focused on the content of the design system and how it should be implemented to be advantageous for its users. The conclusions made through this study are described below.

A design system should contain three main building blocks; design elements, guidelines and components libraries to facilitate the construction of digital products. Depending on the needs of the organization, a design system could expand with more building blocks if necessary.

A design system will need control mechanisms to stay relevant and useful. The following list summarizes the main findings regarding the maintenance and improvement of design systems over time;

- A design system's implementation and maintenance must allow improvements through iterations.
- There should be dedicated personnel taking responsibility of the design systems maintenance and implementation.

- The users of the design system should be involved in its implementation and maintenance to secure that the design system supports their working process in the best possible way.

The code architecture plays an important role for a successful implementation of a design system, the following list summarizes the conclusions made in relation to the structure of the code;

- The structure of the code should allow global components to be implemented all over the organization.
- The building of global components should allow local components to emerge, but the development must be controlled to avoid local proliferation.
- The design system should advocate a process for how to build and improve new and existing components.
- The dependency between styling and functionality should be kept low to allow for independent changes in one or the other.

08.
Future Work

This chapter will provide some recommendations for future work regarding design systems and the implementation support framework.

Naming do matter when implementing design systems and can both be affected by and affect mental models of designers and developers. The mental models' effect on the learning curve of the design system's users is something that this study has not focused on but instead left for future research to explore.

Future work must be done before any further conclusion could be made regarding the ISF and its applicability. The framework designed for Universal Avenue is a proposition that needs to be tested in practice to see if, and how well, the needs of the primary users are fulfilled by the concept. Also, the needs of the secondary users of a design system, such as the management and sales departments, should be investigated and included in the next concept iteration.

Further testing and iterations of the ISF will also be needed, preferably by performing constructive design research similar to this study, but in other organizations. Such studies could elicit needs and requirements from other users that could be compared to the ones summarized in this study. Furthermore, future work should consider organizations of various sizes to be able to verify the ISF's applicability or if changes need to be made to fit all use cases.

09.
Bibliography

Atlassian. (2020). *The power of teamwork.* Retrieved from https://www.atlassian.com/company

Bhatia, M. (2018). *Your guide to qualitative and quantitative data analysis methods*. Retrieved from https://humansofdata.atlan.com/2018/09/qualitative-quantitative-data- analysis-methods/

Bohgard, M. & Karlsson, S. (Eds.). (2015). A*rbete och teknik på människans villkor. Prevent*. ISBN: 9789173652636

Churchill, E. (2019). Scaling UX with Design Systems. *Interaction, 26* (5), 22-23. dio: 10.1145/3352681.

Curtis, N. (2016, February 26). A Design System isn't a project. It's a product serving other products. *Medium.* Retrieved from https://medium.com/eightshapes-llc/a-design-system-isn-t-a-project-it-s-a-product-serving-products-74dcfffef935

DelVecchio, J., White, F., & Phelan, S. (2013). *Tools for Innovation Management: A Comparison of Lean Startup and the Stage Gate System.* Retrieved from https://ssrn.com/abstract=2534138.

Design Council. (2020). *What is the Framework for Innovation? Design Council's evolved Double Diamond.* Retrieved from https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond

Devanney, J. (2017). *The Design Management Office: A Guidebook for delivering Design at Scale*. Retrieved from http://www.momentdesign.com/wp-content/uploads/2017/10/DMO-Guidebook-20171019.pdf

Flood, R.L. & Carson, E.R. (1993). *Dealing With Complexity: An introduction to the Theory and Application of Systems Science.* New York: Springer Science and Business Media.

Frost, B. (2016). *Atomic Design*. Retrieved from https://atomicdesign.bradfrost.com/table-of-contents/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Retrieved from https://books.google.se/books/about/Design_Patterns.html?id=6oHuKQe3TjQC&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false

Ghezzi, A. (2018). Digital startups and the adoption and implementation of Lean Startup Approaches: Effectuation, Bricolage and Opportunity Creation in practice. *Technological Forecasting & Social Change, 146,* 945-960. dio: https://doi.org/10.1016/j.techfore.2018.09.017

Gothelf, J. & Seiden, J. (2016). *Lead UX: Designing Great Products with Agile Teams.* Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc.

Hacq, A. (2019, October 1). Workshop: How to prioritize your design sytem components? *Medium.* Retrieved from https://uxdesign.cc/workshop-how-to-prioritize-your-design-system-components-744aa99f07d7

Heat, C. (2020). *How Wonderbly sold 3 million books worldwide with no publishing connections.* Retrieved from https://www.canva.com/learn/how-wonderbly-sold-3-million-books/

IDEO. (2015). *The Field Guide to Human-Centered Design (first edition).* Retrieved from https://www.designkit.org/resources/1?utm_medium=ApproachPage&utm_source=www.ideo.org&utm_campaign=FGButton

Koskinen, I., Zimmerman, J., Binder, T., Redstrom, J. & Wensveen, S. (2011). *Design Research Through Practice: From the Lab, Field and Showroom*. Waltham, USA: Elsevier Science & Technology.

Liker, J (2004). *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer.* Retrieved from: https://www-accessengineeringlibrary-com.proxy.lib.chalmers.se/content/book/9780071392310

Liker, J. & Hoseus, M. (2008). *Toyota Culture: The Heart and Soul of the Toyota Way.* Retrieved from: https://mhebooklibrary-com.proxy.lib.chalmers.se/doi/book/10.1036/9780071712576

Liker, J. & Meier, D. (2006). *The Toyota Way Fieldbook.* Retrieved from: https://www-dawsonera-com.proxy.lib.chalmers.se/readonline/9780071502115

Lopes, M. (2017, Mars 6). Setup a design system. *Medium.* Retrieved from https://blog.prototypr.io/design-system-ac88c6740f53

MacDonald, D. (2019). *Practical UI Patterns for Design Systems*. Victoria, Australia: Apress.

Macrotrends. (2020). *Salesforce, Inc: Number of Employees 2006-2020.* Retrieved from https://www.macrotrends.net/stocks/charts/CRM/salesforce,-inc/number-of-employees

Maylor, H. (2010). Improving project performance. In *Project Management.* (pp. 384-406). England: Pearson.

Meetup. (2020). *Media.* Retrieved from https://www.meetup.com/media

Pakdil, F. & Leonard, K. M. (2017). Implementing and sustaining lean processes: the dilemma of societal culture effects. *International Journal of Production Research, 55* (3), 700-717. doi: 10.1080/00207543.2016.1200761

Radford, R. (2018). *Balancing creativity with Design Systems.* Retrieved from

https://medium.com/default-to-open/balancing-creativity-with-design-systems-c31b083258ac

Salesforce. (2020). *Lightning Design System*. Retrieved from https://www.lightningdesignsystem.com/

Scupin, R. (1997) *The KJ Method: A technique for Analyzing Data Derived from Japanese Ethnology.* Retrieved from https://search-proquest-com.proxy.lib.chalmers.se/docview/201150881?accountid=10041

Shopify, (2020). *About.* Retrieved from https://www.shopify.com/about

Shopify. (2020). *Company info.* Retrieved from https://news.shopify.com/company-info#

Spencer, D. (2009). *Card Sorting: Designing Usable Categories.* Brooklyn: Rosenfeld Media.

Storybook. (2020). *Build Bulletproof UI components faster*. Retrieved from https://storybook.js.org

Trender, M. (2017, April 7). Design systems sprint 1: The interface inventory. *Medium.* Retrieved from https://medium.com/@marcintreder/design-systems-sprint-1-the-interface-inventory-1f78d376e49a

Universal Avenue. (2020). *About us.* Retrieved from https://www.universalavenue.com/about

UXPin. (2020). *Adele The repository of publicly available design systems and pattern libraries.* Retrieved from https://adele.uxpin.com/

Vesselov, S. and Davis, T. (2019). *Building Design Systems: Unify User Experience Through a Shared Design Language*. USA: Apress.

Wensveen, S. A. G. (2018). *Constructive design research*. Eindhoven: Technische Universiteit Eindhoven.

Wilson, C. (2013). *Brainstorming and Beyond A User-Centered Design Method.* Retrieved from *https://doi.org/10.1016/B978-0-12-407157-5.00001-4*

Wonderbly, (2020). *Our story.* Retrieved from https://www.wonderbly.com

Yarkov, A. (2019). *Symbol Design System Styles & Symbols based on Sketch 54.1.* Retrieved from https://medium.com/sketch-app-sources/symbol-design-system-5f97f23ff217

Appendix I - Detailed Results from the Benchmarking of Open-Source Design systems

Lightning by Salesforce
Salesforce is a company that provides one of the most used customer relationship management services and was founded in 1999. The company is classified as a large enterprise with over 49´000 employees in year 2020 (Macrotrends, 2020).

The design system used and developed by salesforce is called Lightning which is a comprehensive design system containing a lot of information for both designers and developers. Lightning provides information for many use cases to ensure that all potential developers have access to the information they need. It has for example defined 48 different colors to be used throughout the application.

Lightning is divided into several smaller building blocks. It starts with a “What’s new” tab that presents a changelog over the most recent changes. This is followed by an introduction about how to use the system and how the websites using the system should be designed. The design guidelines presented in Lightning have a high abstraction level explaining how patterns should be used without discussing the actual design on a micro-level. The components in Lightning are marked with responsive, adaptive, or prototype depending on how far in development they are with the goal that all components should reach the responsive category. A picture of the overall structure of the design system can be seen in figure I.

Figure I. Picture of the structure of Salesforce's design system.

Each of the components is well documented with a lots of code examples that show how it can be used. All components in Lightning also have a "What's new" tab that contains the changelog for the component that is more precise than the changelog under the general "What's new" tab.

Salesforce has grouped all information about elements in a tokens tab. All elements in the tokens tab are cross-platform reusable (web, ios, android) to aid in the creation of a single source of truth.

Design System by Atlassian

Atlassian is a software development company providing different tools to support teamwork and project management including applications such as Trello, Jira, and Confluence (Atlassian, 2020). The company was founded in 2002 and has over 3000 employees globally and will hence be considered as a large enterprise. Atlassian's design system consists of six main building blocks; design principles, brand, marketing, product, voice and tone, and lastly Atlaskit, which is their official UI library.

Atlassian's design principles include a description of their design philosophy and what to strive for when developing their products. The branding section includes Atlassian's personality, writing style, color, illustrations, logos, and typography. What Atlassian call brand guide, are in this report defined as a style guide. The marketing section includes both a comprehensive description and best practices for how to use the elements included in the branding section, together with links to their resource bank containing

e.g. logos and illustrations files. The resource bank is however not open to the public and can only be reached by Atlassian employees.

The product page contains foundations, components, and patterns, where the foundations describe e.g. the grid spacing and iconography. The component section provides a description of all the main components used by Atlassian, including examples of how to use them. The voice and tone section contains personas, language and grammar guides, and a glossary. The last building block of Atlassian's design system is Atlaskit which is their UI library. Atlaskit has been built based on its design principles and contains documentation and packages. The documentation describes the developer workflow, product integration and releases workflow to name a few. There is also a list of different guides for contribution, mainly intended for developers. A picture of the overall structure of Atlassian´s design system can be seen in figure II.

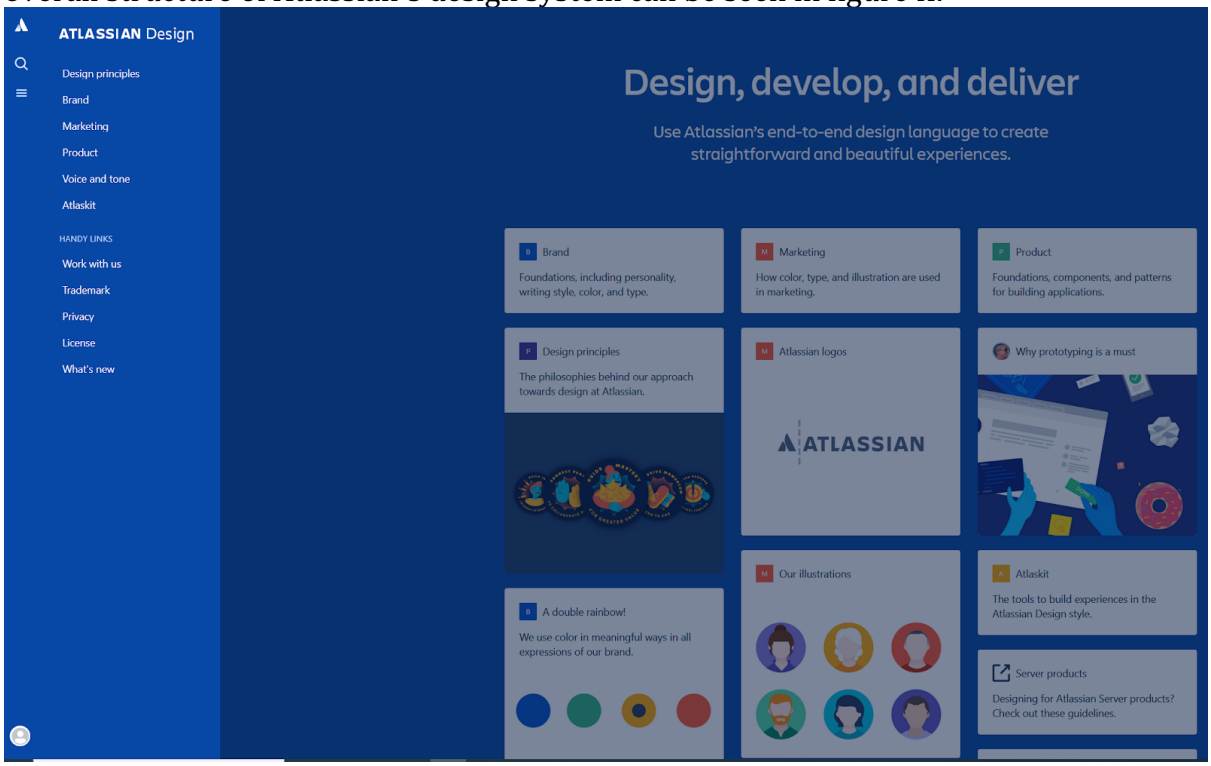


Figure II. The menu structure of Atlassian's design system.

Design System by Wonderbly

Wonderbly is a publishing company providing personalized children's books founded in 2012 (Wonderbly, 2020). The company currently has around 100 employees, and can hence be classified as an SME (Heat, 2020). Wonderbly's design system is said to be following the Atomic approach by Brad Frost and the system contains four main parts; goals, design principles, rules, and components.

In the goal section, Wonderbly describes the aim with their design systems which they divide into four parts; cohesiveness, efficiency, trustworthiness, and onboarding. The design principles are representing the mindset one should have when designing and developing new products at Wonderbly, the principles include accessibility and how to approach their customers. The rule section contains more static information about elements, called style guide in this report. The components are organized in a list of Wonderbly's most common components. The description of each component contains

information regarding context and usage, together with an interactive display of the component which allows the user of the system to see which states that are available for each component. A picture of the overall structure of Wonderbly´s design system can be seen in figure III.

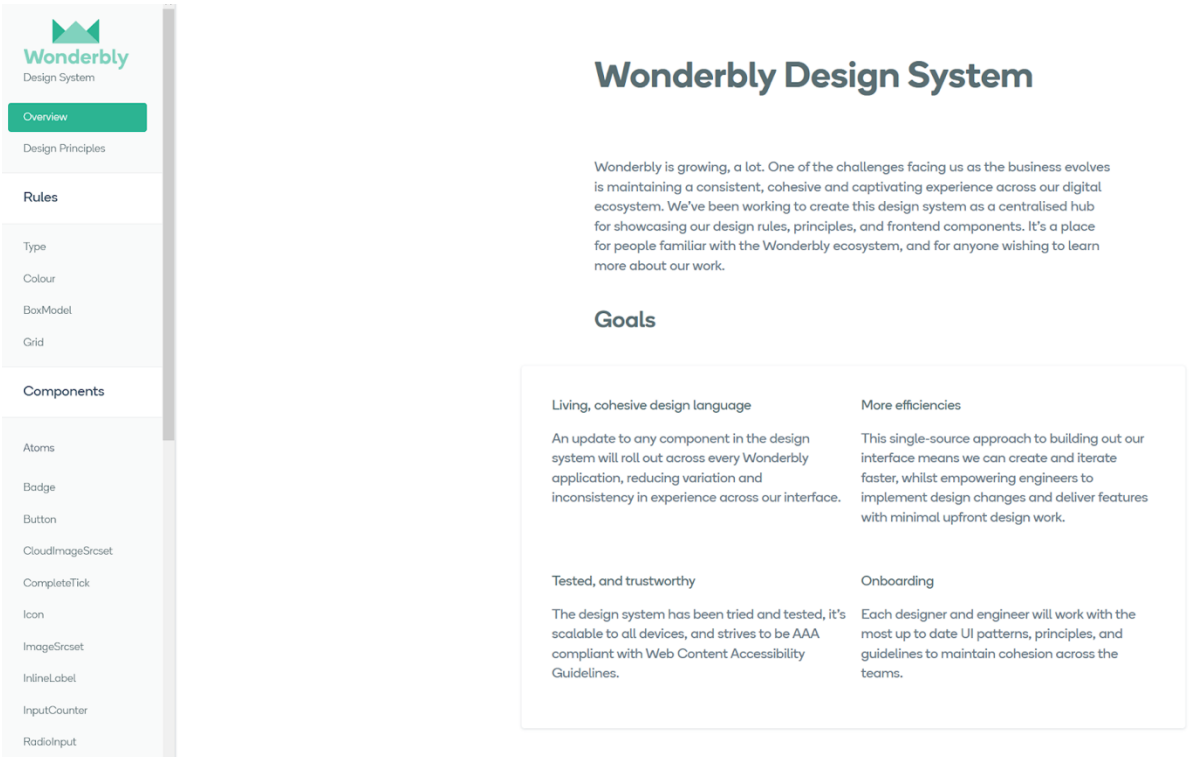


Figure III. The menu structure of Wonderbly´s design system.

Swarm by Meetup

Meetup is a community platform that provides different activities where people with the same interests can meet. The company was founded in 2002 and has approximately 300 employees (Meetup, 2020). Meetup's design system is called Swarm and contains four main parts; design, brand, content and web components.

The design part of Swarm contains general guidelines regarding color, buttons, icons, typography, and forms but also guides regarding layout and usability. Some of the guides include best practices and illustrative examples of usage but there is no consistent way of structuring the information. The brand section includes brand assets such as logos and images together with documentation regarding the brand voice. The content part of the design system provides general writing guides e.g. capitalization, punctuation and, spelling. Lastly, the web components are displayed in Storybook and provides an interactive view of the components together with the code. A picture of the overall structure of Swarm can be seen in figure IV.

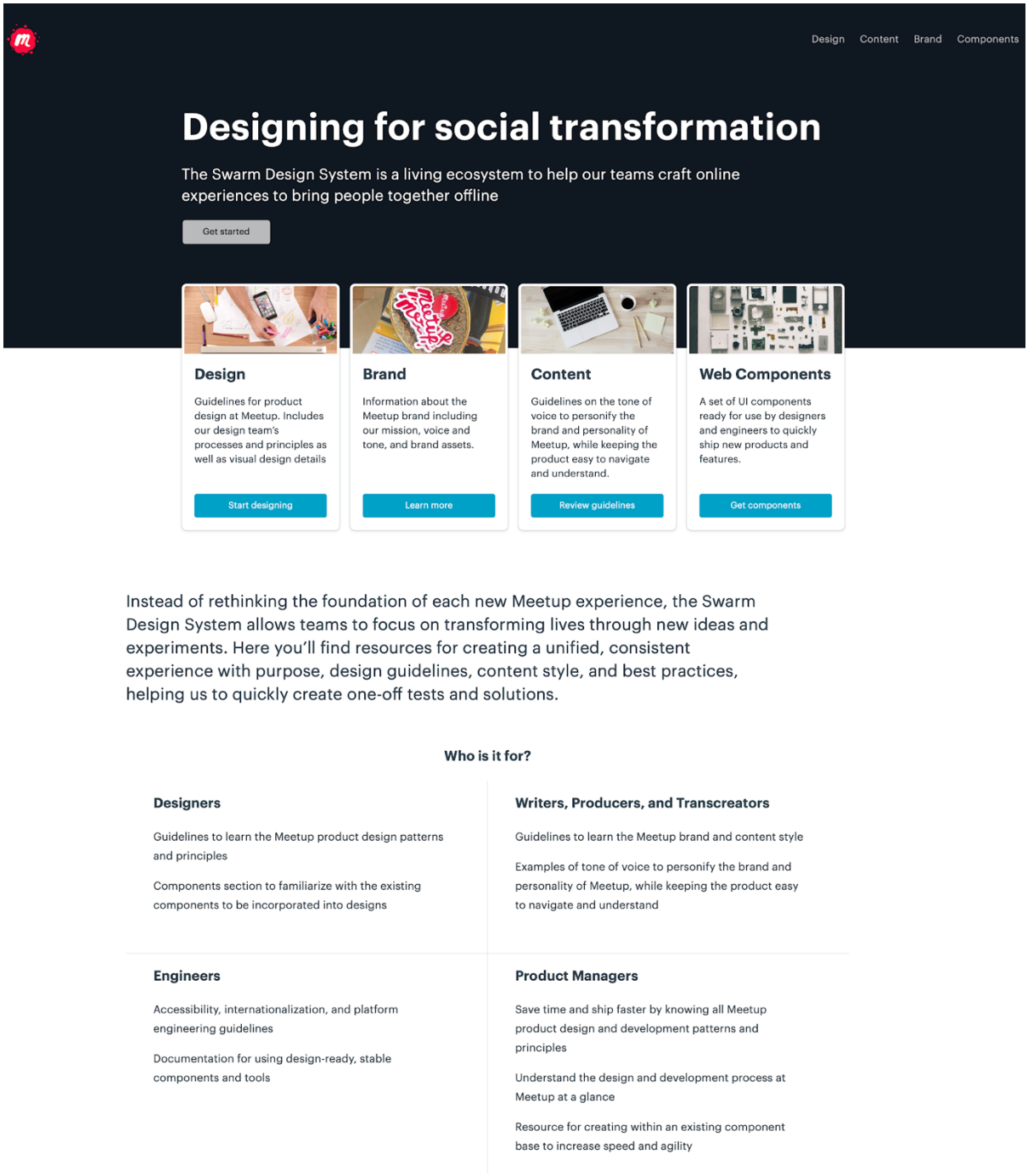


Figure IV. The menu structure of Swarm, Meetup´s design system.

Polaris by Shopify

Polaris is the design system created and curated by Shopify. Shopify provides retailers with a suite of services that helps them run an online store. The company was founded in 2004 and has over 5000 employees (Shopify, 2020), classifying it as a large enterprise.

Polaris is divided into 5 major sections; foundations, content, design, components and experiences. Foundations contains general guidelines for accessibility and how apps should be designed for the shopify platform. Content handles most of the text guidelines

and brings up both voice and tone and grammar. The design tab primarily contains guidelines for design elements. Under the components tab all components are ordered in a list. Each component is displayed with active code examples and a system for changing the styling. The last tab about experiences brings up different patterns and good thing to have in mind regarding the overall design and layout. A picture of Polaris can be seen in figure V.

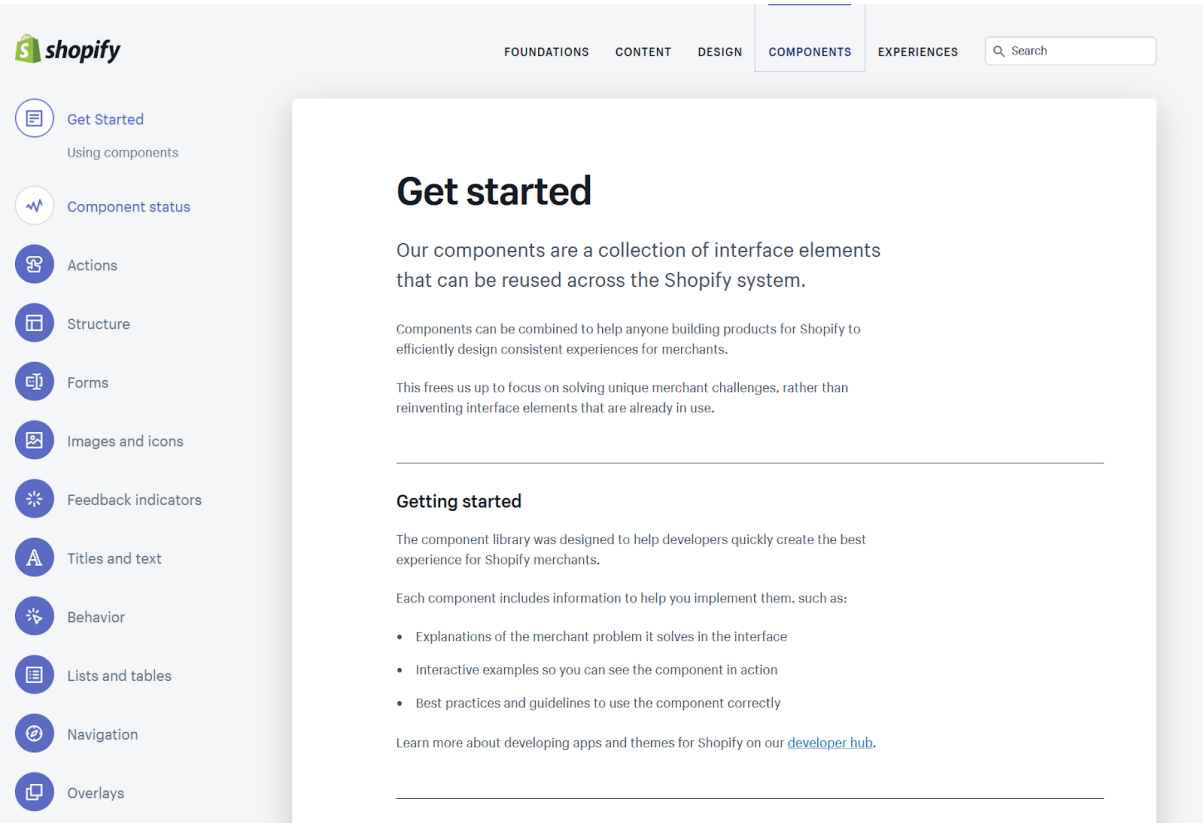


Figure V. The menu structure of Polaris, Shopify's design system.

Appendix II – Early Concept Iterations

This chapter provides a compilation of the early concept iterations that was performed to define the final design system framework for Universal Avenue.

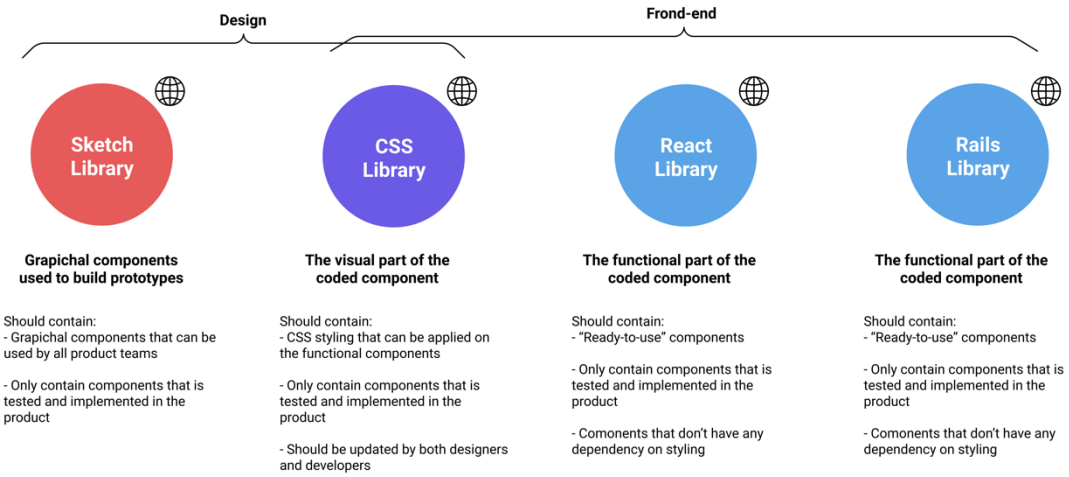


Figure VI. A first composition of the main component libraries to be used in the new design system.

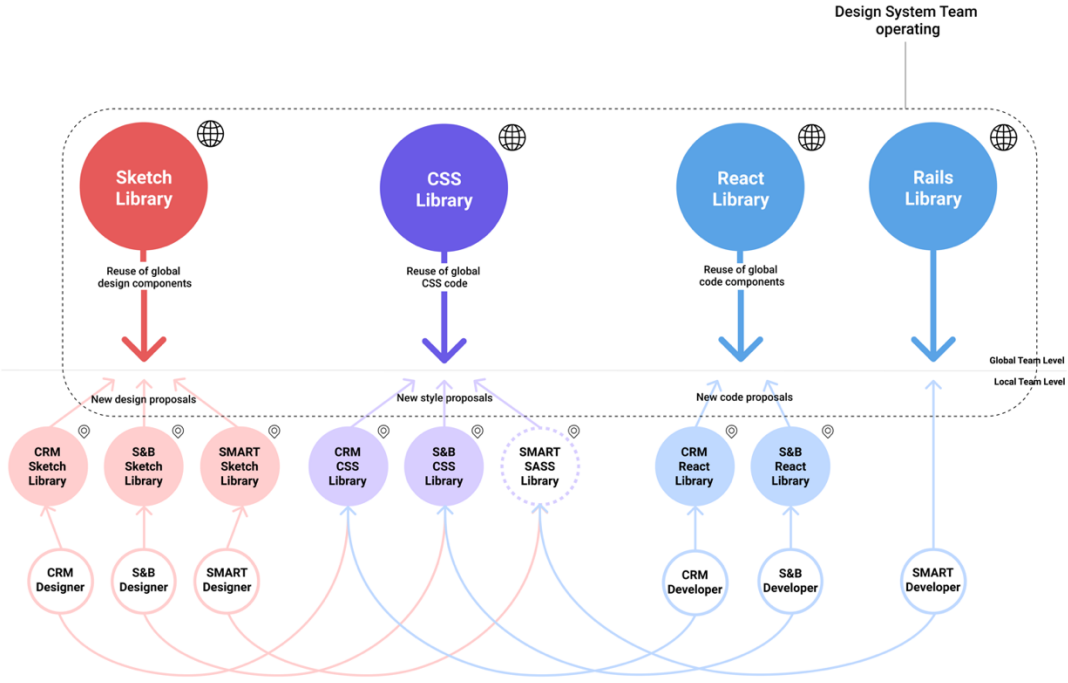


Figure VII. First concept of the top-down and bottom-up process for structuring component libraries.

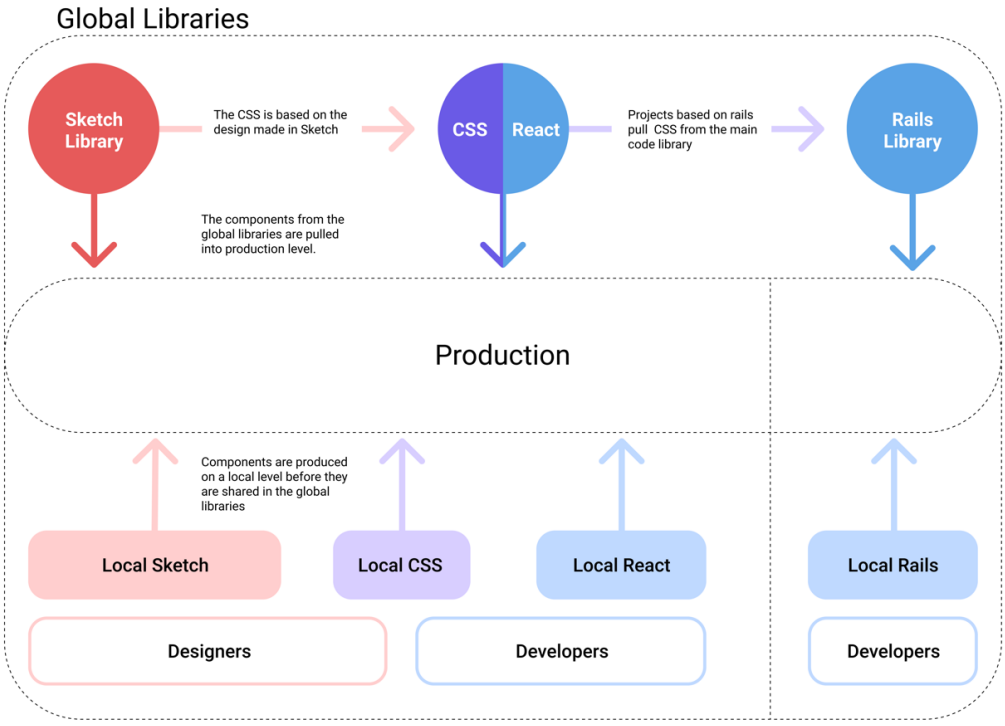


Figure VIII. Second concept of the top-down and bottom-up process for structuring component libraries.

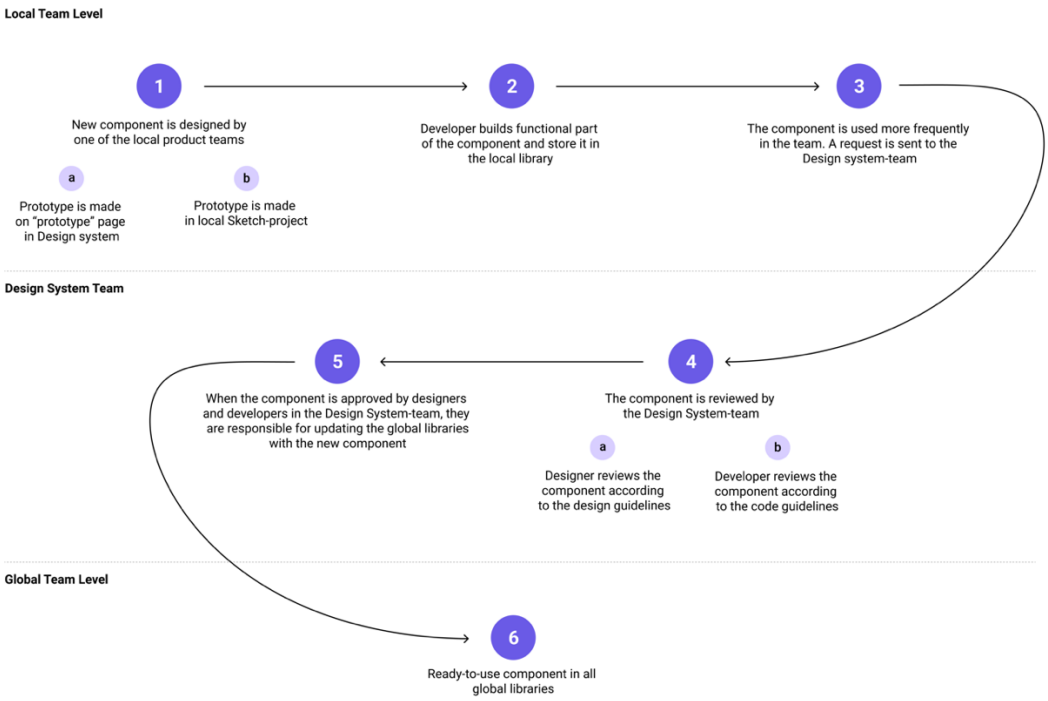


Figure IX. First concept on the process of building global components.

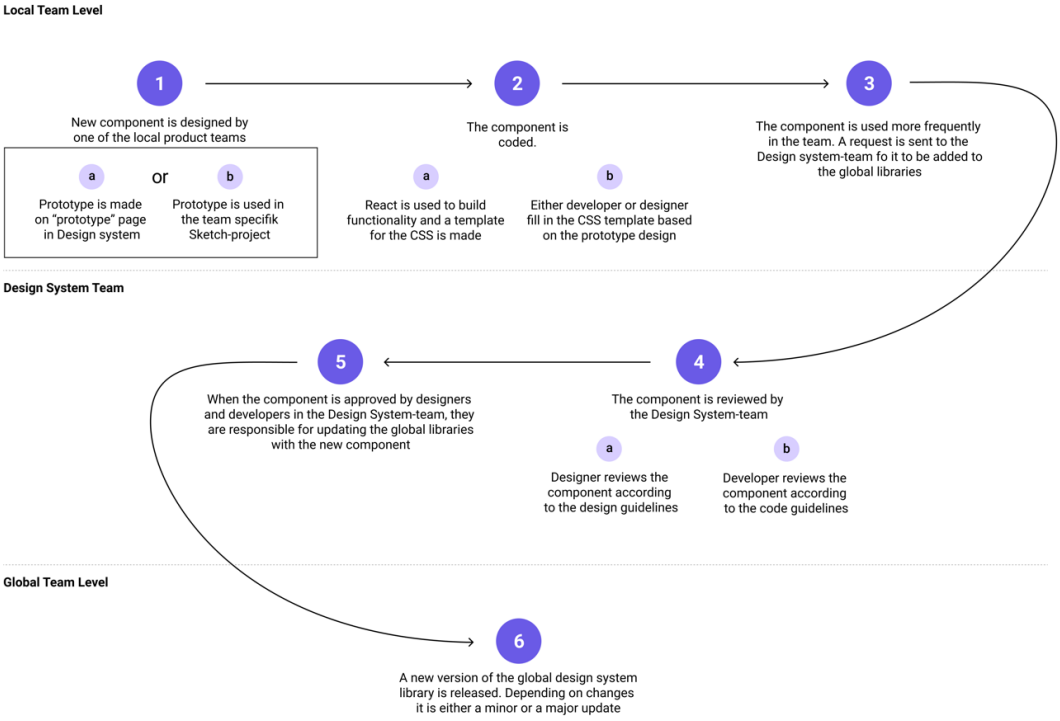


Figure X. Second concept on the process of building global components.

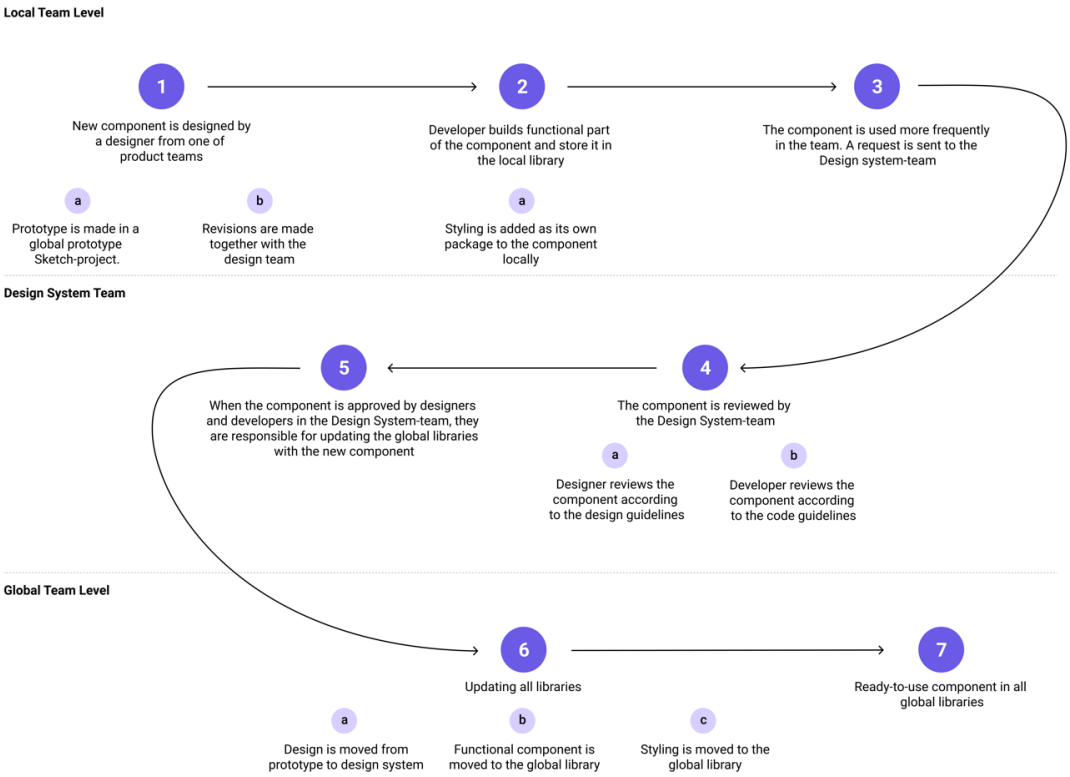


Figure XI. Third concept on the process of building global components.

Appendix III - Problem areas defied at Universal Avenue

The complete list of problem areas found during the first round of interviews at Universal Avenue.

No universal system for development and design

- Quick reviews are crucial for the experience of a good working process
- Both designers and developer express that they want to collaborate more, both within and in between the teams
- If the designers take the already existing components into consideration when designing new things, the developers work will be smoother
- Variation in the development process is important for high motivational levels
- The distance between designers, developers and stakeholders are creating a lack of understanding and underestimation of each other's processes and how long things take to create/do. This is a source of frustration.
- There is an interest in creating a common ground for how development should be done, both from designers and developers.
- Several attempts have been made to create a common ground for development, however none have been a breakthrough and they are used differently by the different teams.

Lacking collaboration between the teams

- There is a wish of having more regular collaboration between the different teams.
- To regularly have discussions with other developers and designers are important, preferably face to face
- Generally, the designers have more regular communication with each other, however they wish that they had even more frequent communication
- The designers express a need for having more discussions and collaborations regarding the overall design of the products

Low reusability of components

- The created components are to specific and are only used in a couple of places. Components taken from the global library are often in need of rewriting to be usable.
- The general usability of components are low and the existing components are often too specific to reuse, and rewriting is required.
- Difficult to make changes to the CSS in today´s codebase

Not enough time for maintenance

- There is no, or little time spent on maintenance of the existing codebase.
- High focus on creating features instead of focusing on meeting user needs.

Lacking feedback from, and information about the users

- It is easier to design if there is a well-defined problem and a clear context.

- The teams would generally like to have more contact with or information about the users.
- The developers appreciate to get a context for what they are developing, the “why” is important.

DEPARTMENT OF INDUSTRIAL AND MATERIAL SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY