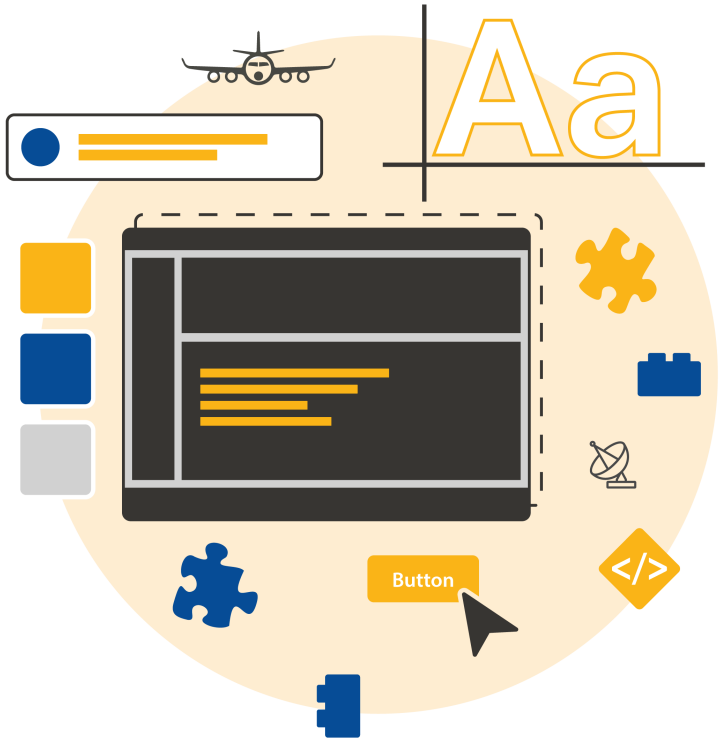
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Designing a Design System for Coherent Tactical GUIs

Leveraging systematic design to balance coherence and responsiveness in the design of complex applications

Master's thesis in Computer science and engineering

ANTON LINDER
MATTIAS NILSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Designing a Design System for Coherent Tactical GUIs

Leveraging systematic design to balance coherence and responsiveness in the design of complex applications

ANTON LINDER
MATTIAS NILSSON

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Designing a Design System for Coherent Tactical GUIs
Leveraging systematic design to bring coherence to tactical systems of complex applications
ANTON LINDER & MATTIAS NILSSON

© ANTON LINDER & MATTIAS NILSSON, 2022.

Supervisor: Pauline Belford, Department of Computer Science and Engineering
Advisor: Fredrika Lundkvist, Saab AB
Examiner: Staffan Björk, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Abstract

Today, the problem of incoherence exists within many organizations where separate teams oversee different products and services without utilizing central design principles or communication platforms. With increased scale and complexity, this problem tends to escalate to the point where resources can not be shared or reused across teams, limiting collaboration and symbiosis within the organization. Additionally, as different projects develop their own proprietary design language, the resulting applications tend to become incoherent and disassociate from the brand.

Front-end developers working in the domain of tactical applications constantly have to make difficult design decisions in their respective customer projects while trying to manage a balance between coherence and responsiveness to specific customer requirements. Additionally, thousands of hours have already been invested into each application and project. If design decisions are not kept documented and accessible by the whole organization, valuable knowledge and solutions easily get lost between teams and projects. As a consequence, resources are spent on reinventing new solutions to old problems resulting in incoherence between applications. Research suggests that one way of increasing communication and resource sharing between teams while simultaneously managing the balance between coherence and responsiveness is to design and utilize a design system.

As most contemporary design systems and previous research has been focused on web development, this master's thesis aimed to investigate this research gap and extend the body of knowledge of design systems into the domain of tactical graphical user interfaces (GUIs). This was done by addressing the research question *What are important requirements to consider when designing a design system for coherent tactical GUIs?*. The project followed an iterative design thinking process, using both ethnographic and participatory research methodologies, to research the field of tactical GUIs - specifically Airborne Early Warning & Control systems at Saab Surveillance in Gothenburg. This was combined with extensive literature studies and benchmarks of established design systems in order to understand best practices for designing design systems.

The outcome of the project is a requirements list that answers the research question. The list includes the most important requirements to consider when designing a design system for coherent tactical GUIs in terms of using, contributing, and minimizing the trade-off between coherence and responsiveness. The requirements and the insights gained from the design research has been used to prototype a foundation for a design system for Saab. The resulting design system, called *PhoeniX Design System*, is built on the FlatLaf Look & Feel and includes design tokens, components, and patterns that supports developers of AEW&C applications in building coherent, yet responsive, tactical GUIs. Finally, a reference product was prototyped to exemplify the use of the design system.

Keywords: tactical, GUI, UX, AEW&C, design system, style guide, pattern library, components, coherence, responsiveness

Acknowledgements

During this thesis project, we have encountered a lot of enthusiastic and helpful people that have supplied us with guidance, reflection, discussion, and knowledge. Therefore, we would like to direct a special thanks to our academic supervisor Pauline Belford for her support and her valuable input throughout this thesis. We also want to direct a special thanks to Fredrika Lundkvist, our supervisor at Saab, for guiding us through the complex domain of tactical applications. This project would not have been the same without both of you.

We would also like to direct a big thanks to everyone that we met at Saab, especially the ones who have contributed to this thesis project with their knowledge and reflection by being part of the design research. Thank you for taking the time to join us in interviews and workshops and for showing us the domain through your eyes and enthusiastically discussing the introduction of a design system at Saab.

Additionally, we want to thank each other for a well-executed thesis project, for deep conversations, and for carrying and supporting each other. Lastly, we want to direct a huge thanks to Chalmers University of Technology for these amazing years.

Anton Linder & Mattias Nilsson, Gothenburg, June 2022

Contents

1 Introduction **1**
- 1.1 Aim . . . 2
- 1.2 Goal . . . 2
- 1.3 Research Question . . . 2
- 1.4 Deliverables . . . 2
- 1.5 Research Contribution . . . 2
- 1.6 Industry Contribution . . . 3
- 1.7 Limitations . . . 3
- 1.8 Stakeholders . . . 4
  - 1.8.1 Thesis Duo . . . 4
  - 1.8.2 Chalmers University of Technology . . . 4
  - 1.8.3 Developers at Saab . . . 4
  - 1.8.4 Designers at Saab . . . 5
  - 1.8.5 Operators of AEW&C Applications . . . 5

2 Background **7**
- 2.1 Saab Airborne Surveillance . . . 7
  - 2.1.1 System Overview . . . 8
  - 2.1.2 Previous Work . . . 10
- 2.2 Related Work . . . 11
  - 2.2.1 Established Design Systems . . . 11
  - 2.2.2 Military Design Systems . . . 11
  - 2.2.3 Information Visualization in Complex Systems . . . 12

3 Theory **13**
- 3.1 Research Framework . . . 13
  - 3.1.1 Research for Design . . . 13
  - 3.1.2 Ethnographic Research . . . 14
- 3.2 Design Frameworks . . . 15
  - 3.2.1 Design Thinking . . . 15
  - 3.2.2 User-Centered Design . . . 16
  - 3.2.3 Co-Design . . . 16
- 3.3 Design Theory . . . 17
  - 3.3.1 Wicked Problems in Design Thinking . . . 17
  - 3.3.2 Human Factors . . . 17
  - 3.3.3 User Experience . . . 18

3.3.4 Usability . . . 18
3.3.5 Jordan's 10 Principles of Usable Design . . . 19
3.3.6 Norman's 7 Fundamental Principles of Interaction Design . . . 19
3.3.7 Consistency in Design . . . 20
3.3.8 Coherence in Design . . . 22
3.4 Design System . . . 25
3.4.1 Architecture . . . 25
3.4.2 Benefits of Design Systems . . . 29
3.4.2.1 Collaboration . . . 29
3.4.2.2 Efficiency . . . 30
3.4.2.3 Consistency & Coherence . . . 31
3.4.2.4 Usability & Accessibility . . . 31
3.4.3 Designing a Design System . . . 32
3.4.3.1 Approach . . . 32
3.4.3.2 Process . . . 33
3.4.3.3 Content . . . 34
3.4.3.4 Design System Repository . . . 36
3.4.3.5 Balancing Coherence and Responsiveness . . . 36

4 Methodology **39**
4.1 Empathize . . . 39
4.1.1 Literature study . . . 39
4.1.2 Interviews . . . 39
4.1.3 Contextual Inquiry . . . 40
4.1.4 Try it Yourself . . . 40
4.1.5 Benchmark . . . 40
4.2 Define . . . 41
4.2.1 Affinity Diagram . . . 41
4.2.2 Problem Statement . . . 41
4.2.3 Interface Audit . . . 41
4.2.4 Heuristic Evaluation . . . 43
4.2.5 Requirements . . . 43
4.3 Ideate . . . 43
4.3.1 Brainstorming . . . 43
4.3.2 Sketching . . . 44
4.3.3 Workshop . . . 44
4.4 Prototype . . . 45
4.4.1 Lo-fi Prototypes . . . 45
4.4.2 Hi-fi Prototypes . . . 45
4.4.2.1 Mockups . . . 45
4.5 Test . . . 46
4.5.1 Usability Testing . . . 46

5 Process & Execution **47**
5.1 Project Planning . . . 48
5.2 Empathize - Coherence between applications . . . 48
5.2.1 Literature Study on Consistency & Coherence . . . 49

5.2.2 Study of Company Research and Guidelines 49
5.2.3 Try it yourself 49
5.2.4 Stakeholder interviews 50
5.2.5 Takeaways 51
5.3 Define - Coherence between applications 51
5.3.1 Problem Statement 51
5.3.2 Research Scope 53
5.4 Empathize - Design System 53
5.4.1 Literature Study on Design Systems 53
5.4.2 Benchmarking 54
5.4.3 Developer Interviews 55
5.4.3.1 Work Process 56
5.4.3.2 Communication & Collaboration 57
5.4.3.3 Design System 57
5.4.4 Contextual Inquiry with Operators 58
5.4.5 Stakeholder Interviews 59
5.4.6 Takeaways 60
5.5 Define - Design System 61
5.5.1 Affinity Diagram 61
5.5.2 Interface Audit 61
5.5.3 Heuristic Evaluation 62
5.5.3.1 Internal Consistency 63
5.5.3.2 External Consistency 64
5.5.3.3 Analogic & Metaphoric Correspondence in the Real World 64
5.5.3.4 Use of Tokens & Elements to Form Components . . . 65
5.5.4 Project Scope 65
5.6 Ideate 66
5.6.1 Remote Workshop 66
5.6.2 Sketching 68
5.6.3 Takeaways 69
5.7 Prototype 69
5.7.1 Tokens 70
5.7.2 Components 72
5.7.3 Component Group 74
5.7.4 Mockup 74
5.7.5 Component Page 75
5.8 Test 76
5.8.1 Implementation Usability Test 76
5.8.2 Stakeholder Feedback Presentation 77
5.8.3 Takeaways 78

6 Results **79**
6.1 Design System Requirements 79
6.1.1 Usage Factors 80
6.1.2 Contribution Factors 81

6.1.3 Trade-off Factors . . . 81
6.1.4 Look & Feel Factors . . . 82
6.1.5 Organizational Factors . . . 83
6.2 Design System . . . 83
6.2.1 Tokens . . . 84
6.2.1.1 Sizing & Spacing . . . 84
6.2.1.2 Typography . . . 85
6.2.1.3 Colors . . . 86
6.2.2 Components . . . 88
6.2.3 Component Groups . . . 89
6.2.4 Mockup . . . 91

7 Discussion **93**
7.1 Process & Execution . . . 93
7.1.1 Confidentiality & Complexity . . . 93
7.1.2 Remote Work . . . 94
7.2 Result . . . 94
7.2.1 Design System Requirements . . . 94
7.2.2 PhoeniX Design System . . . 95
7.3 Stakeholder Implications . . . 97
7.3.1 Developers . . . 97
7.3.2 Designers . . . 98
7.3.3 Operators . . . 98
7.4 Ethical Considerations . . . 98
7.5 Future Work . . . 100

8 Conclusion **103**

References **105**

A Appendix **I**
Appendix A - Thesis Proposal . . . I
Appendix B - Gantt Chart . . . II
Appendix C - Benchmark Design Systems . . . III
Appendix D - Interview Script . . . VI
Appendix E - Contextual Inquiry Script . . . VIII
Appendix F - Interface Audit . . . IX
Appendix G - Workshop Structure . . . XI
Appendix H - Naming Components . . . XII
Appendix I - Implementation Test . . . XIII
Appendix J - PhoeniX's Components . . . XV
Appendix K - PhoeniX's Component Groups . . . XVII

Glossary

Base product - A code base of an application, including the UI and its standard functionality, from which the new versions of the application and project-specific applications are derived from.

Components - Components are the interface and GUI elements that make up a digital product, for example buttons, form fields, radio buttons, and checkboxes. Using reusable components allows designers and developers to quickly implement a commonly used function into an application without needing to redraw or code it every time.

Design debt - The implicit cost of skipping important steps of the design process in order to deliver quick results. Design debt introduces imperfections in terms of user experience and design processes that appear over time as a result of innovation, growth, and lack of design refactoring.

Design System - An overarching term for a library of experience design and development resources that is shared across an organization and teams. It typically consists of several parts.

Design Token - Design tokens are the lowest-level entities of a design system and are essentially name-value pairs that store UI attributes such as color, typography and sizing. The values can then be referenced and used as dependencies for higher-level entities of the design system such as components and, in turn, patterns as a systematic approach to design. Design tokens are often stored in framework files and can be documented and shared in nearly any format.

FlatLaf - A modern open-source cross-platform Look Feel for Java Swing desktop applications. It looks almost flat (with no shadows or gradients), clean, simple and elegant. It is able to scale on high pixel density displays and runs on Java 8 or newer.

Framework - A framework, or software framework, is a platform that provides a foundation for developing software applications. It can be thought of as a template of a working program that can be selectively modified by adding code.

GUI - Graphical user interface (GUI) is a digital user interface that is capable of displaying graphics. A GUI is the part of the application that allows the user and computer to interact and communicate with each other. The user is able to manipulate the GUI in various ways and control different functionalities within the application through, for example, menus, buttons, and dialog boxes.

Java - A high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java

without the need to recompile.

JComponent - The base class for all Java Swing interface components apart from top-level containers. Swing components whose names begin with "J" are descendants of the JComponent class: such as JButton, JScrollPane and JTable.

Look & Feel - In software design, the Look & Feel of a graphical user interface comprises aspects of its design, including elements such as colors, shapes, layout, and typefaces, as well as the behavior of dynamic elements such as buttons, boxes, and menus. A Look & Feel allows applications to have a specific style independent of the underlying platform. “Look” refers to the appearance of GUI widgets and “feel” refers to the way the widgets behave.

Pattern - A pattern, or component group, is comprised of several components that together has a specific use case and purpose. A reusable pattern allows developers to quickly implement a common combination of components into the application.

Swing - Swing is a GUI Widget Toolkit for Java and is an API for providing a GUI to Java Programs. Swing provides components written in Java and therefore are platform-independent.

Technical debt concerns the implicit technical cost of going to market without a fully finished or tested product. In software development, it is about prioritizing quick results over perfect code which causes imperfections that later need to be addressed and refactored.

1

Introduction

Problems with incoherence exist within many organizations as separate development teams oversee different products without utilizing central design principles or efficient communication between design and development departments. In addition, different products may have different functionalities or specific customer requirements to address, making them difficult to design in a coherent fashion. Therefore, it is important to balance flexibility in the design process while still maintaining coherence with internal and external applications. Research suggests that one way of mitigating the problem of incoherence while still allowing for freedom to adapt to local circumstances is by introducing a Design system (Harris & Henderson, 2012).

As a large and globally recognized company, Saab has built a thorough product portfolio of surveillance solutions over the years. In the domain of airborne surveillance, operators may work during long shifts in occasionally critical and stressful situations. Working in a critical situation, the operators must be able to make quick judgments based on large amounts of complex data with small room for error. Thus, it is important that the applications are designed to reduce all unnecessary disturbances to enable the operators to focus on their tasks. Being a large, multinational corporation with many different offices in different locations, different independent teams have developed the different applications and their graphical user interfaces for each radar system. As a result, there are some differences in the way some applications look and how functionally similar tasks are completed. The Human Factors Integration team at Saab Surveillance in Gothenburg has therefore expressed that it could be beneficial for the radar operators if the applications were more similar and coherent in the way they look and function.

By using a design system, it is possible to address internal and external coherence in a system of applications within an organization. The premise of building and using a design system is that it brings many benefits to organizations in terms of collaboration and efficiency, which ultimately allows teams to create more consistent and coherent experiences for end-users. As of today, most research on design systems is centered around websites and web applications, and not so much on complex applications (Frost, 2016; Suarez et al., n.d.; Vesselov & Davis, 2019). To be able to further look into the possibilities of using a design system as a potential solution for improving the coherence of Saab's airborne surveillance applications, research must be undertaken to find the best possible approach.

1.1 Aim

This thesis aims to research and analyze the incoherence between different Airborne Early Warning and Control (AEW&C) applications at Saab and identify opportunities for improvement based on principles of interaction design. Insights will be used to identify requirements, design a design system, and a reference product for coherent tactical GUIs.

1.2 Goal

The primary goal of the thesis is to supply developers of AEW&C applications with a library of easily accessible design resources and guidelines to help achieve coherence and compliance with user expectations and design principles.

1.3 Research Question

To help us reach our goal, the following research question is addressed:

What are important requirements to consider when designing a design system for coherent tactical GUIs?

In order to help answer this research question, two sub-questions are used:

1. What are important factors to enable developers to use and contribute to a design system for coherent tactical GUIs?
2. What are important factors to help minimize the trade-off between coherence and responsiveness in design systems?

1.4 Deliverables

The deliverables for this master's thesis are as follows:

- A foundation for a design system consisting of tokens, elements, components, and patterns.
- Requirements on how to design a design system for coherent tactical GUIs.
- A simple reference product in the form of a digital mockup and/or semi-functional prototype designed to embody the design system.

1.5 Research Contribution

This master's thesis contributes to the research field of design systems by addressing a gap when it comes to designing design systems for tactical GUIs. Today, most contemporary research focuses on how design systems should be designed to support developers and designers of websites (Frost, 2016; Suarez et al., n.d.; Vesselov

& Davis, 2019). As tactical applications tend to be an order of magnitude more complex and customized, the current research does not offer sufficient support in designing design systems for these kinds of applications.

Moreover, there is currently limited research on factors that help balance coherence and responsiveness in the development of tactical applications. Harris and Henderson (2012) emphasizes the importance of designing permeable and user-centered infrastructures to limit the trade-off between coherence and responsiveness, but does not suggest factors for how this can be achieved. This thesis employs ethnographic research methodologies through the design thinking and research for design frameworks to identify important factors for designing a design system to allow developers to manage this balance in the context of tactical GUIs. As a result, the research conducted in this master's thesis extends the current body of knowledge within the fields of tactical GUIs, coherence in design and design systems.

1.6 Industry Contribution

The problem of incoherence exists within many organizations as separate development teams oversee different products and services without utilizing central design principles or efficient communication between design and development departments. The research area bears particular significance when it comes to aiding the design and development process of complex and time-sensitive applications such that users can operate them intuitively and efficiently with minimal training and cognitive effort.

The benefit of the work includes improved communication infrastructure between designers and developers, improved implementation efficiency and cost-effectiveness along with improved brand identity and coherence between similar applications. As a result, the work accelerates and facilitates human factors integration (HFI) into the development process. As a secondary effect, improved coherence also benefits AEW&C operators working across multiple applications.

1.7 Limitations

Research and inquiries were to be limited to the context of tactical GUIs and the target users are developers of the AEW&C applications in the product portfolio of Saab AB Airborne Surveillance. Since Saab has a large product portfolio, this thesis project did not aim to bring coherence to all of Saab's tactical applications. Hence the project was limited to exploring only three of the airborne surveillance applications since neither developers nor operators work across more than one of the systems.

Due to the confidential nature of Saab's business area, there was limited access to direct communication with active operators of AEW&C applications. User studies and evaluations were therefore performed with instructors and past operators or

other stakeholders with advanced knowledge of the applications. Furthermore, due to confidentiality, the tactical GUIs could not, and are not disclosed in their full fidelity in this thesis. This also means that stakeholders outside of Saab, including the academic supervisor, were not able to fully appreciate the interfaces. Instead, the thesis duo has created mockups of the current interfaces to convey the essence without revealing sensitive information.

The project did not include the implementation of the final design system in terms of creating or utilizing an existing design system repository. Instead, the repository is left as future work. Regardless, the technical feasibility was still taken into consideration by continuously involving Saab's developers and stakeholders throughout the design process.

1.8 Stakeholders

The following stakeholders have been identified as having an interest in or being directly affected by the results of this thesis project.

1.8.1 Thesis Duo

The thesis duo consists of two students from the master's program Interaction Design & Technologies. This thesis constitutes the final step of the journey towards a master's degree for both students. This means that there is an inherent eagerness from both students to demonstrate theoretical and practical design proficiency while learning and applying new knowledge to produce an accomplished piece of research. Furthermore, since the duo strives to find employment after graduation, carrying out a well-received and impactful project at a big company like Saab could potentially benefit future career opportunities. It is worth mentioning that this thesis project is a continuation of a project that one of the students in the duo was a part of during a previous summer internship. The result of this project can be read in section 2.1.2 Previous Work.

1.8.2 Chalmers University of Technology

Chalmers and the faculty of Interaction Design & Technologies have an interest in further promoting their association with producing quality research and competent engineers. Hence, the name of the university is affected by the work performed by the duo during this thesis. The faculty provides an academic supervisor who advises the thesis duo in the writing of the thesis. Finally, the course examiner decides if the master's thesis report and presentation hold sufficient academic merit.

1.8.3 Developers at Saab

Developers at Saab are responsible for the coding and maintenance of different AEW&C applications. They carry the main responsibility of building and translating design decisions into concrete code. For this reason, developers are considered

primary users which means that it is key to understand and cater to their needs in order to achieve a successful result.

1.8.4 Designers at Saab

Designers at Saab Surveillance mainly consist of the Human Factors Integration-team (HFI-team) who are responsible for evaluating and suggesting general improvements to the different AEW&C applications. Traditionally, designers have not had any significant involvement in determining the design of the GUIs during development. Recently, some progress has been made to employ dedicated UX designers responsible for guidelines, user flows, and style for each application. As the thesis project aims to improve the way designers and developers collaborate. Designers, hence, are considered secondary users during this master's thesis.

1.8.5 Operators of AEW&C Applications

Operators working within and across multiple AEW&C applications are highly reliant on and affected by the structure, layout, visual appearance, and information displayed on the different GUIs. For this reason, it is important to understand operator needs and workflows to ensure that each application remains useful and understandable to current and new operators. As active operators are not accessible to the thesis duo, Saab's in-house instructors, who are former operators of AEW&C applications, represent the interests of this stakeholder group. Instructors, however, are not an ideal proxy for operators which needs to be taken into account during the research. Operators are considered side users in this master's thesis project.

Indirect stakeholders include Saab's customers who purchase the product and employees responsible for, for example, projects, IT, documentation, and marketing. However, this thesis mainly focuses on the implications for developers, designers, and operators at Saab.

2
Background

This chapter aims to give the reader a better understanding of the Saab domain and to clarify the need for this thesis and its connection to Saab. This is done by first presenting the field of Airborne surveillance at Saab and some of their previous work and what current tools are used for development. This is followed by an introduction to related work within the domain of design systems.

2.1 Saab Airborne Surveillance

Saab Airborne Surveillance, part of Saab AB, specializes in military and governmental surveillance solutions. The product portfolio includes GlobalEye (see Figure 2.1), an Airborne Early Warning and Control system (AEW&C), with an accompanying suite of applications. An AEW&C system is an airborne radar picket system designed to provide long-range air-, sea- and land surveillance and support government- and military forces with real time information.

Figure 2.1: GlobalEye, a multi-domain AEW&C solution from Saab AB

The main objective of the operators is to receive, interpret and distribute information and data provided by the built-in sensor suite of the AEW&C aircraft. In turn, the AEW&C applications allow the operators to monitor and label data in real-time to create a recognized air and sea picture. The nature of the live working context and complex, time sensitive data, means that the operators are faced with critical and stressful situations. In turn, this can result in the operators experiencing cognitive

strain which could affect the decision-making process. Therefore, operators need to undergo rigorous training in order to effectively operate the applications. Since the technical and operational needs differ between different AEW&C applications and between operators, each tactical GUI is designed for the intended use case and operator role.

The operators in an AEW&C aircraft are acting in a high risk domain, with potentially devastating impacts of faulty decisions. Therefore, it is crucial that the applications in which the operators are working are thoroughly developed to prevent user errors and present data so that it is understandable and interpretable by the operators.

2.1.1 System Overview

In this thesis project, three of Saab's applications associated with their AEW&C solutions have been studied. For reasons of confidentiality, the applications will not be mentioned by name but rather as Application 1 (A1), Application 2 (A2), and Application 3 (A3). Further, the GUI of the applications will be presented below, but for the same confidential reasons, only representative mockups will be presented. Together the applications account for three of the main applications that an AEW&C operator may encounter during a mission, both on board an aircraft and on the ground, see Figure 2.2.

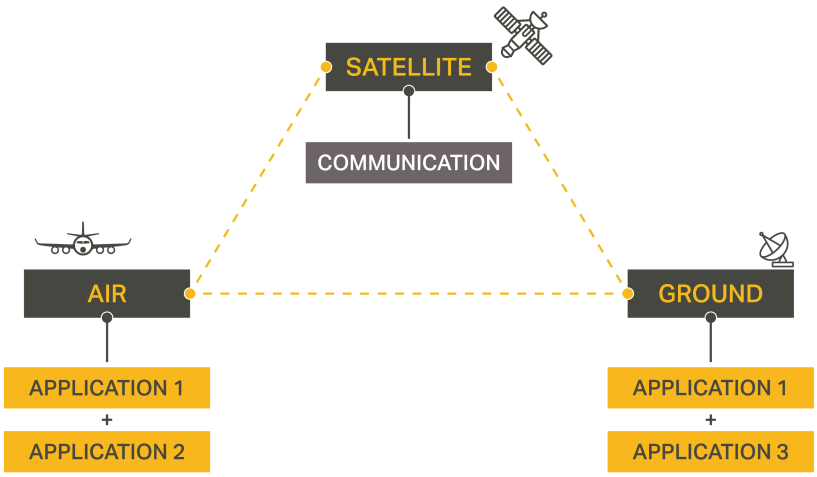


Figure 2.2: Schematic of the AEW&C system and its setting

In the AEW&C system, A1 can be seen as the primary application, used most frequently and by the majority of the operators at the same time. A1 and A2 are on board the aircraft, working with translating various radar data into visual information that can be presented to and interpreted by the operators. The applications in the aircraft continuously collect data from the radar sensors and communicate this down to the ground. On the ground, the operators can interact with A3, which collects the data from an aircraft and provides it to the operation center. In the

meanwhile, A3 receives data from the operation center, to be able to create a compilation and a more accurate air and surface situation picture. A3 also works as a remote controller for some of the functions in A1 on board the aircraft. In some cases, the operator working on the ground has two screens next to each other, one showing a replica of A1 from the aircraft and the other one presenting A3.

Since the systems have been developed separately over the years, similar or identical needs and functions have been addressed with different solutions resulting in visually and functionally inconsistent interfaces. For reasons of confidentiality, the applications can not be shown in full fidelity in this thesis. Instead, mockups of the three applications can be seen in Figure 2.3, 2.4 and 2.5.

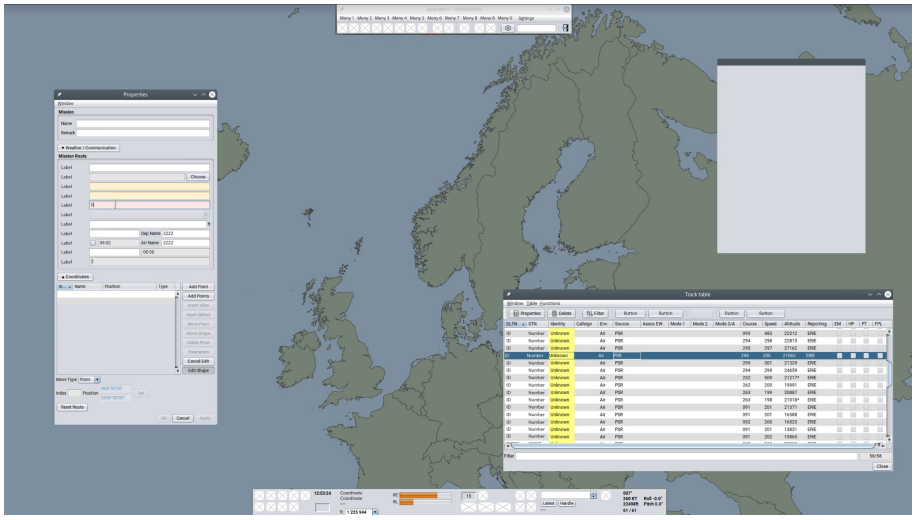


Figure 2.3: A mockup of Application 1

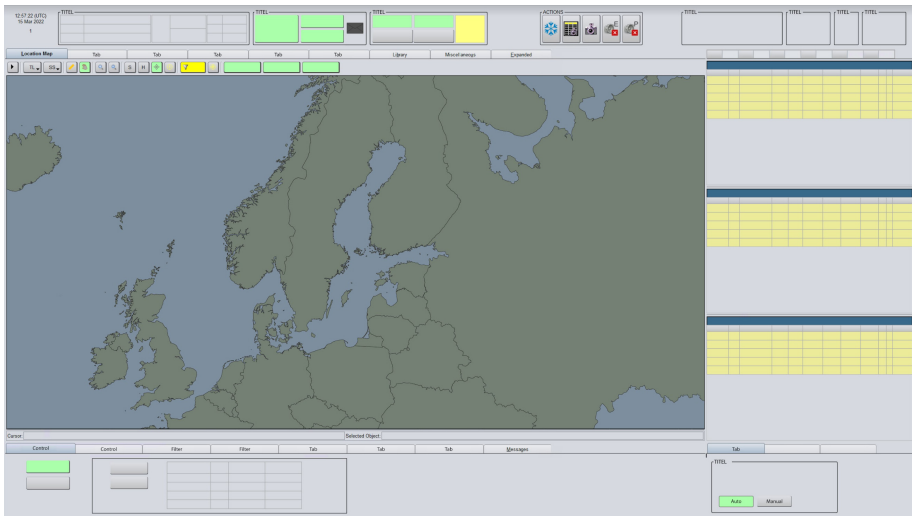


Figure 2.4: A mockup of Application 2

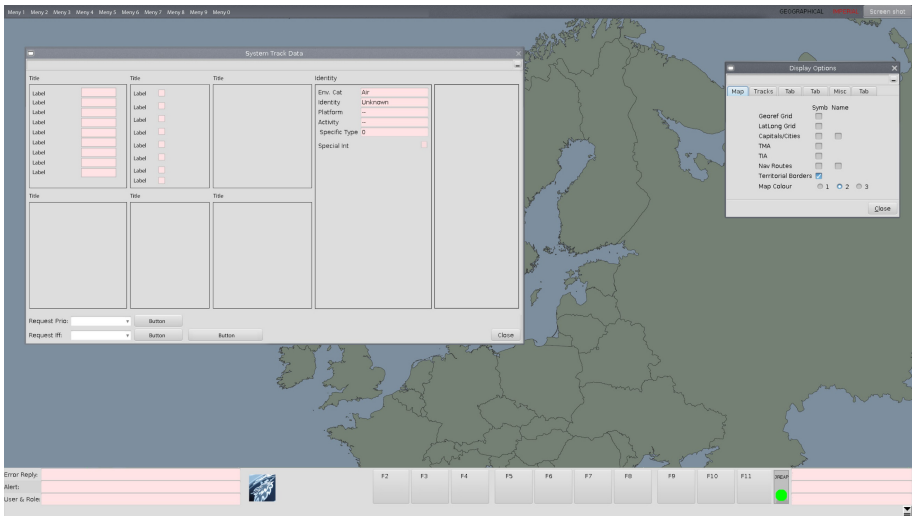


Figure 2.5: A mockup of Application 3

2.1.2 Previous Work

For a couple of years, Saab has had their own Brand portal (brand.saab.com) containing guidelines for how to communicate the brand on various platforms. Here, elements such as tone-of-voice, logotypes, colors, typography, and icons are documented, along with how these should be used in various situations. In addition to the brand elements, the brand portal also contains guidelines for how to name new products, how to style components for websites, and design guidelines for how to think when developing user-friendly products, such as usability rules.

In a recent project, the HFI-team reviewed the possibility of bringing the three AEW&C applications in a direction where they will look and behave more alike and be more coherent. Because Saab's product portfolio is so large, it is difficult to take on all applications at once and it is thus easier to look at one domain at a time. They chose to approach the problem by developing HMI-guidelines (human-machine-interface) for when designing tactical interfaces for Saab's ground and air surveillance systems. This was done as an attempt to provide developers with a tool to develop coherent applications even if they were developed by different teams and in different locations. Derived from literature and best practices in the field of desktop applications, over 200 guidelines were produced on how, why, and when to use different colors, components, and patterns in tactical desktop applications. However, the HFI-team realized that the HMI-guidelines document alone might not be enough and the thesis duo was asked to continue researching the domain where the HFI-team left off.

The nature of the problem falls distinctly inside the domain of interaction design as it treats the interface between different stakeholders within an organization. Moreover, the result of the project affects the interaction between operators and digital products.

2.2 Related Work

In this section, various work related to this thesis project is summarized. The related work listed in this section is meant to frame the scope and to show how others have approached similar challenges regarding consistency and coherence, design systems, and working with complex applications.

2.2.1 Established Design Systems

Many companies have developed proprietary design systems in order to provide their users with a coherent, branded experience throughout their product portfolio and third-party applications. In 2014, Google launched its *Material Design system* which was built on the metaphor of physical elements, known as "Quantum Paper" and mimicked physical textures, light reflections and cast shadows. The system was rapidly adopted by designers and developers who started to implement it in their applications and websites. Eventually, the use of the system started to diverge as individual developers made necessary adjustments to fit their brand and specific needs, leading Google to create *Material Theming*. Material Theming now offers tools and guidelines that let designers and developers customize the original Material design system to create their own design system within the bounds of Material Design's principles (material.io). This exemplifies an approach where responsiveness is balanced with coherence to create a unified yet customized experience for developers and, by extension, end users as well. Matias Duarte, Google's VP of Design, described Material Theming as "a design system for making design systems." (Bohn, 2018, para. 12).

Nowadays, many large companies have developed their own proprietary design systems to ensure consistent and coherent application and web development and resulting user experiences throughout their entire organization. Examples include Adobe Spectrum (spectrum.adobe.com), IBM Carbon (carbondesignsystem.com), Microsoft Fluent (microsoft.com/design/fluent) and Shopify Polaris (polaris.shopify.com).

2.2.2 Military Design Systems

Traditionally, GUIs in military systems tend to emerge as a consequence of hardware and software development. Therefore the design - in terms of visual appearance, layouts, and task execution - often results in a bi-product of the hardware- and software testing interfaces. (Hwang & Jo, 2008). As a result, research and utilization of design systems in military software development are close to nonexistent. However, the use of other development tools has been found to have major positive productivity implications within this context (Maxwell et al., 1996).

Since 2019, The MOD.UK Design System (design-system.digital.mod.uk) constitutes the foundation for all UK defense applications. It was developed in response to an abundance of legacy UI design patterns and external inconsistency among different user interfaces in the Royal Navy's product portfolio (Brantingham, 2019).

Inspired by the principles described in Atomic Design (Frost, 2016), MOD.UK Design System specifies and organizes UI elements and patterns into an interactive component library accessed through a web portal. The portal is built using a third-party open source tool for building UI components and pages in isolation (storybook.js.org). It is organized into several pages where each UI element, pattern, component, framework, and principle has its own page. Principles describe attributes such as colors, spacing, and typography. Components offer reusable UI elements like buttons, text inputs, and headers. Patterns are repeatable collections of components, provided with extra guidance. Frameworks provide helpful foundations for heavy, interaction-based interfaces (design-system.digital.mod.uk). The design process is described as experimental yet adhering to the Agile framework; involving scrum- and Kanban-methodology in combination with bi-weekly retrospectives (Brantingham, 2019).

2.2.3 Information Visualization in Complex Systems

In an AEW&C system, the primary users are operating in a complex environment characterized by huge amounts of data and the need to navigate through different sets of information. This makes visualizing the information in a way that is cognitively manageable to an operator a challenge (Colineau et al., 2004). It is common that the interface is cluttered with information which results in cognitive load for the operator, who must make decisions based on the presented information.

In their study, Livingston et al. (2011) presents an integrated set of functions for interactions with information for a mobile augmented reality application used in military contexts. They concluded the importance of simplifying tasks and filtering information to reduce cognitive load and reduce problems with information overload. This reiterates Oviatt (2006) who emphasized the importance of designing to minimize the user's cognitive load in order to effectively liberate mental resources and thereby increase performance and maintain situational awareness. This results in higher efficiency with fewer user errors. Furthermore, Luostarinen et al. (2010) also found that challenging environments, such as moving vehicles, demand further simplification and clarification of all visible parts of the UI. The authors recommend hiding complex parts of the system to a lower level in the system architecture.

Moreover, Colineau et al. emphasize the importance of being able to customize how information is presented to best support the operator in their tasks and decision-making. They developed a delivery engine called Virtual Document Planner (VDP) to solve this issue. VDP considers all the data processed by the system, organize it, and tailors the interface to present just the most vital information in order for the operator to complete a task, effectively minimizing the operator's cognitive load.

3

Theory

In this chapter, the theoretical background will be presented, containing relevant frameworks and design concepts used in the thesis project. Included in this chapter are sections on research frameworks, design frameworks, design theory, and theory on design systems. The section on design systems includes theory on what a design system is, the different terminologies used in the domain, the benefits of using a design system, and how to design a design system. The theories presented in this chapter will lay a foundation for the design work in this thesis project.

3.1 Research Framework

In this section, the different theoretical frameworks regarding research used for this research project are presented.

3.1.1 Research for Design

Forlizzi et al. (2009) defines research for design as "a theoretical outcome of many different activities that provide designers with theories they can apply to improve their practice of design" (p. 2892). In this thesis, the research for design framework will act to support the research and design by focusing on obtaining useful knowledge and understanding of the problem, thereby directly addressing the research question. This could be done by research to gather information and data needed to back up and conclude specific topics or questions (Downton, 2003). In his book, Downton (2003) describes research for design as "research to enable design"(p. 17). Compared to research through design, which is more focused on providing theories in a broader context to be used in future general projects, research for design is more practical and specific (Figure 3.1). The research will serve as a basis for decision-making within the thesis project about, for example, directions or specific design elements. However, even though the research is important, it will only work as one part of the decision-making. It is important to emphasize that this is a user-centered design (see User-Centered Design) thesis and decisions regarding the deliverables will also be based on insights derived from user studies.

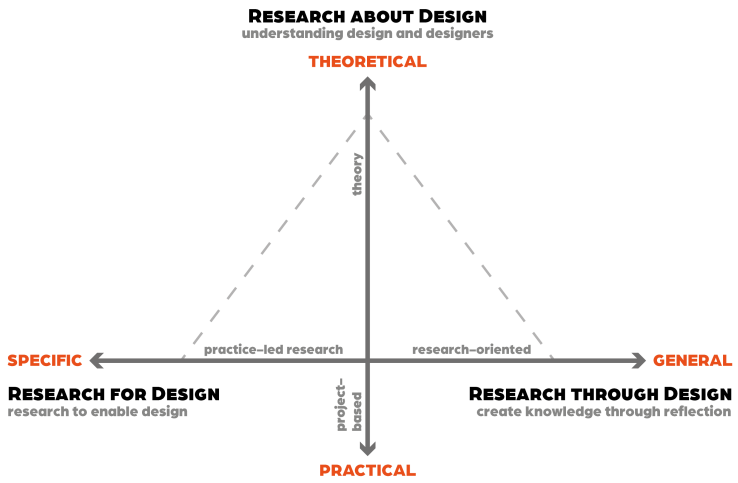


Figure 3.1: Schematic of design research approaches (inspired by Frankel & Racine, 2010)

3.1.2 Ethnographic Research

Ethnographic research derives from the field of social science with the goal of understanding human behaviors. Doing ethnographic research means getting involved in the everyday life activities of the users studied. It thus provides a unique perspective for understanding users' activities and to empathize with the user (Blomberg et al., 1993, p. 219-234). To obtain this behavioral understanding, ethnographic research typically includes field work involving some combination of observations, interviews, and participatory activities with the users.

Blomberg et al. (1993) mention four main principles to guide ethnographic work:

- **Natural setting** - Studying people and their activities in their everyday setting. This means that the research is to be conducted in the user's natural environment, rather than in a laboratory or experimental setting.
- **Holism** - Observing people's activities and behaviors should be done in the everyday context they occur in. Holism refers to how particular behaviors are influenced by and fit into a larger whole.
- **Descriptive** - Based on the fieldwork, ethnographers develop a descriptive understanding of the people studied and their everyday life. Ethnographers describe how people actually behave, not how they ought to behave.
- **Members' point-of-view** - To keep a non-judgemental stance, ethnographic research aims to always understand the world from the point-of-view of the people studied. By realizing that one can never really get into another person's head or see the world exactly as they do, ethnographic research methods aim to get as close to a user's view of the situation as possible. With such a focus, ethnographers are interested in describing behavior in terms that are relevant and meaningful to the participants.

In this thesis project, ethnographic research methodology will be used in combination with the Design Thinking framework (see 3.2.1 Design Thinking) in its first two phases, empathize and define, to gain a saturated understanding of the problem

from the developers' point-of-view.

3.2 Design Frameworks

There is a big variation of frameworks that can be used by researchers and designers to come up with great solutions to problems, those used for this project are described below.

3.2.1 Design Thinking

Design thinking is a design ideology that believes in a hands-on human-centric design process for solving wicked problems (Gibbons, 2016). Solving problems adhering to this approach, one believes that problem-solving can lead to innovation, and innovation can lead to differentiation and a competitive advantage. As a design framework, the design thinking process is an effective tool for solving complex problems by continuously iterating between five different phases: 1) Empathize, 2) Define, 3) Ideate, 4) Prototype, and 5) Test (Dam, 2021), as illustrated in Figure 3.2. The five steps are presented in detail in chapter 4 Methodology.

A design thinking approach should be applied in projects with an ill-defined, wicked problem (see Wicked Problems in Design Thinking) where it is not suitable to work in a linear process, such as the waterfall method (D. Norman, 2013, p. 219-234). This thesis project is researching the design of a design system aimed to balance coherence and responsiveness in a system of complex applications, used by operators working in challenging environments. In combination with the novelness of the research and limited time, the thesis is inherently tackling a wicked problem where it would be inappropriate to work in a linear process. Hence, the design thinking framework is deemed appropriate.

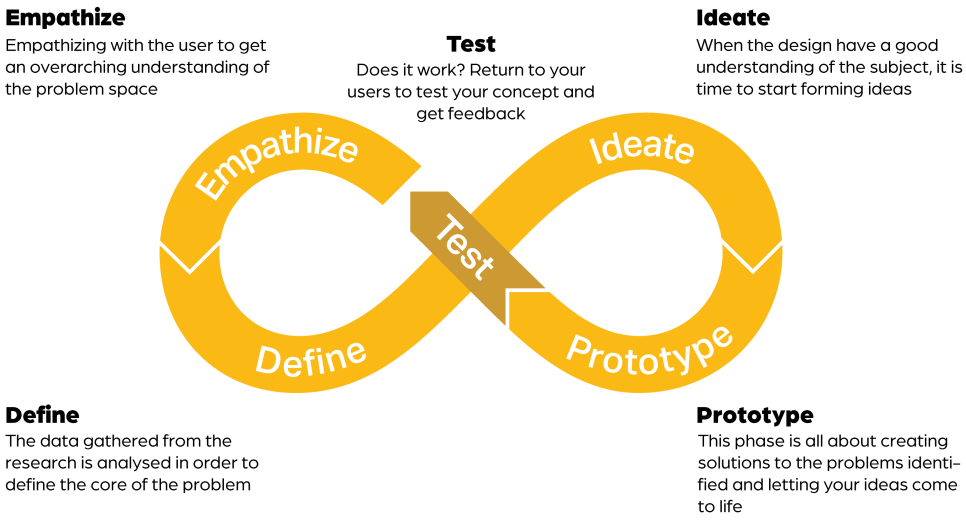


Figure 3.2: A schematic of the iterative design thinking process

3.2.2 User-Centered Design

User-centered design (UCD) is an approach to product development derived from a combination of engineering and psychology (D. Norman, 2013). The UCD framework focuses on the users, their needs, and surroundings, to make usable products that are actually useful. This is done by applying human factors (see Human Factors) and usability (see Usability) methods to improve the overall user experience with the product (International Organization for Standardization, 2019). The UCD approach aims to solve *the right problem*; to understand the origin of the problem and solve the problem based on what the real issues are (D. Norman, 2013, pp. 217-234). Norman stresses the importance of involving users in the development process since it is their needs and motivations that need to be reflected in the solution. Designers involve users in the development process to observe and analyze their habits and reactions during interaction with a product in order to better understand their needs. Most people do not appreciate their implicit needs and might even be unaware of the obstacles they are facing every day.

Recognizing the user, the product, and their surroundings as a system with interconnected parts it is easier to design for good communication between these parts. Excellent communication is an important factor, especially communication from computer to user (D. Norman, 2013, pp. 8-9). This means that the design should indicate what actions are possible to do, what is happening, and what is about to happen. This is especially important with regard to errors. Here, the communication between the system and the user is vital for both error prevention and recovery once they do occur. When users understand what has happened and how to recover from errors, the user experience tends to improve.

The UCD cycle is iterative and fits well within the design thinking framework. To avoid losing track and solving the wrong problem, the UCD approach instead iterates on an approximation. This iteration is done through rapid tests of ideas and the result is then used to modify the approach and problem definition. The UCD cycle iterates through four activities: 1) observation, 2) ideation, 3) prototyping, and 4) testing. The activities are repeatedly used over each cycle in the development process to get closer to a desired solution.

3.2.3 Co-Design

Co-design (collaborative design) is a collective creativity approach that actively engages a broader range of people and stakeholders beyond the design team (Sanders & Stappers, 2008). Involving different stakeholders in the design process increases the likelihood of achieving satisfactory products and services that comply with different kinds of users. It is about building a mutual understanding of the system by inviting participants with a big variation of skills and knowledge to better approach complex problems (Burkett, 2012). Sanders & Stappers argue that co-design should be applied throughout the whole development process since it generates a high value to the design. Since this thesis project is set out to create a foundation for a design system at Saab, it is important to involve the users who are going to use the prod-

uct; the developers and the designers at Saab. Therefore, the Co-design approach is suitable to make sure that the relevant stakeholders are included in the process and that the design system is created together with them.

3.3 Design Theory

In this section, relevant design theory and principles are presented.

3.3.1 Wicked Problems in Design Thinking

When dealing with ill-defined problems that cannot be solved by using traditional linear methods, designers most often deal with wicked problems. Wicked problems are often described as problems that seem impossible to solve. They do not have a true or false answer, but rather a good or bad solution (Rittel & Webber, 1973). Often, complex problems within healthcare, education, or government are seen as examples of wicked problems. The term was first coined by the mathematician and designer Horst Rittel in the 1960s. Rittel defined wicked problems as a “class of social system problems which are ill-formulated, where the information is confusing, where there are many clients and decision-makers with conflicting values, and where the ramifications in the whole system are thoroughly confusing." (Buchanan, 1992, p. 15).

It was Buchanan (1992), a professor of design, who first made the connection between wicked problems and the design thinking approach. Design thinking is an interactive process and it is extremely useful in tackling ill-defined problems. Buchanan saw the possibility of challenging the traditional linear design processes that were based on determinate problems with a definite outcome. In contrast, wicked problems are inherently more complex and indeterminate; there are no definitive conditions or limits to wicked design problems. It is this trait of wicked problems which may make them seem impossible to solve at first (Rittel & Webber, 1973).

In this thesis, the problem statement fulfills more than one criteria of wicked problems. The problem is ill-defined in terms of what requirements should be considered and there can not exist one indisputable solution, only better or worse ways to solve the problem.

3.3.2 Human Factors

Human factors mean designing and developing products that are adapted to the human body and cognitive abilities such that they are both efficient and productive (Bligård, 2015). Human factors “[...] applies information about human abilities, limitations, and other characteristics to the design of tools, machines, systems, tasks, jobs, and environments for safe, comfortable and effective human use.” (Chapanis, 1985, p. 2). According to Wickens et al. (1997), the goal of human factors is to reduce human error, increase productivity, and enhance safety and comfort with a specific focus on the interaction between the human and the engineering system.

They go on by writing "While the ultimate goal is to establish principles that reflect performance of people in real-world contexts, the underlying scientific principles are gained through research conducted in both laboratory and real-world environments." (Wickens et al., 1997, p. 11). In research, human factors engineering employs the scientific method to study human behavior so that it results in data that may be applied to our primary goals.

3.3.3 User Experience

User experience (UX) is a widely used term used by designers within GUIs with no real theoretical definition (Batterbee & Koskien, 2010; Hassenzahl & Tractinsky, 2006). The term UX is used to describe the usability of products, how they look, and the experiential aspects of technology use. But most importantly, UX describes the whole experience with the product or service; from talking about it, to the interface, to interact with it, to the context, to how you feel about it, and how it fits your everyday life (D. Norman & Nielsen, n.d.). In designing user experiences, it is important not only to focus on a specific task or a behavioral goal (Batterbee & Koskien, 2010). Even though UX is not as measurable as technical aspects, it is equally important to take into account as it covers the entire user journey and addresses the availability of a design. This will, in turn, change the focus away from only preventing problems in the design system (for example usability) towards a focus on creating quality experiences (Hassenzahl & Tractinsky, 2006).

3.3.4 Usability

Usability can be described as a quality attribute of a system that makes it possible to measure how easy-to-use a user interface is. The ISO definition of Usability is "Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" (International Organization for Standardization, 2013). Here, effectiveness is about how to achieve the intended goal as effectively as possible, whereas efficiency is the resources (e.g. time or effort) needed by the user in order to achieve the goal. In addition, the user must feel satisfaction with the interaction and experience. Nielsen (2012) divides usability into five main quality components:

- **Learnability** - How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency** - Once users have learned the design, how quickly can they perform tasks?
- **Memorability** - When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- **Errors** - How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction** - How pleasant is it to use the design?

3.3.5 Jordan's 10 Principles of Usable Design

Jordan (2002), introduces 10 principles of usable design that should be considered when developing new products and interfaces:

1. **Consistency** - Similar tasks should be performed in similar ways.
2. **Compatibility** - Tasks within the system should be performed in a way that fits with the user's expectations based on their previous knowledge.
3. **Consideration of User Resources** - Consider the user's physical and mental limitations.
4. **Feedback** - Give response to user actions, convey that they have been registered, and provide meaningful information about the result.
5. **Error Prevention and Recovery** - Minimize the possibility of user error and make recovery simple and fast when errors do occur.
6. **User Control** - Maximize the extent of which the user has control of a system's actions and states.
7. **Visual Clarity** - Information is presented in such a way that it is easily readable and understood without causing confusion.
8. **Prioritization of Functionality** - The most important functionality and information is easily accessible to the user.
9. **Appropriate Transfer of Technology** - Make appropriate use of technology developed in other contexts to enhance usability.
10. **Explicitness** - Shape the system to give explicit clues about functionality and method of operation.

3.3.6 Norman's 7 Fundamental Principles of Interaction Design

In his book *The Design of Everyday Things*, D. Norman (2013) outlines seven fundamental principles of interaction design:

1. **Discoverability** (Visibility) - The user should be able to determine what actions are possible and the state of the device.
2. **Feedback** - The system communicates the results of the user's actions.
3. **Constraints** - Limitations introduced to restrict certain actions and guide the user towards correct actions.
4. **Mapping** - Different interface elements should be placed in a logical manner in relation to each other
5. **Conceptual Model** - Refers to the way the artifact projects the information needed to create a good conceptual model of the artifact.
6. **Affordance** - The perceived and actual actions a user can perform on an artifact.
7. **Signifier** - Communicates to the user where the action should take place.

Cooper et al. (2014) suggests that we omit the term "and actual" from the definition of affordance to only describe the cognitive concept of an agent's perception of

possible actions on an artifact rather than what can actually be performed. Cooper et al. (2014) also introduces the term *pliant* to refer to objects or screen areas that do react to user input and that can be manipulated. Combining this with the concept of affordances and signifiers allows us to design artifacts that clearly communicate their purpose and method of operation to the user.

3.3.7 Consistency in Design

Consistency is a fundamental and established design principle that has been proven difficult to define within the context of Human-Computer Interfaces (Grudin, 1989). Jordan (2002) described consistency as a design principle in general terms of users being able to perform similar tasks in similar ways. More specifically, consistency refers to how similar parts look, feel and function throughout an entire system with the purpose of bringing uniformity to the interaction (Yllobre, 2020). In a GUI, each element can be considered its own stimuli, where consistency is achieved when similar sets of stimuli have the same usage, behavior, or result. Zuschlag (2010) identifies seven types of stimuli in GUIs:

1. **Symbols** - Imagery, icons that convey information.
2. **Codes** - Colors, sizes, weights, marks and other graphic dimensions that may represent data values or invoke a metaphor.
3. **Units of measurement** - Either for a particular numeric attribute or groups of related attributes.
4. **Data formats** - For data in the form of text.
5. **Terms** - Identifying the objects, classes, actions, and events in user interactions
6. **Abbreviations** - For terms, including the names of the function and control keys that are assigned to commands.
7. **Layouts** - UI elements' relative locations on a page or in a menu, window, or temporal sequence.

Inconsistency occurs when the meanings of these stimuli vary - when stimuli or usage pairs are not the same. This implies that there are two kinds of inconsistency. Zuschlag (2010) refers to these as *irregularity* and *contradiction*:

- **Irregularity** - When different sets of stimuli have the same usage, behavior, or result. For example, if some buttons are labeled "reset" and others "undo" although their functionality is the same. This type of inconsistency can lead to confusion and a lack of understanding.
- **Contradiction** - When the same sets of stimuli have different usages. For example, if a button labeled "undo" sometimes only reverts the last action and sometimes is used to reset everything. This can lead to misunderstandings and increased cognitive load.

Irregularities and contradictions can be more or less severe depending on the degree of similarity between two sets of stimuli or usages. Figure 3.3 illustrates a gradient

where darker areas represent the strength of inconsistency. Reversely, when we have similar stimuli for similar usages we achieve regularity and when we have different stimuli for different usages we achieve concordance. A weak irregularity would be designing a custom, larger checkbox rather than using the system standard whereas a strong irregularity would be to make the x in the checkbox indicating that the option is not selected.

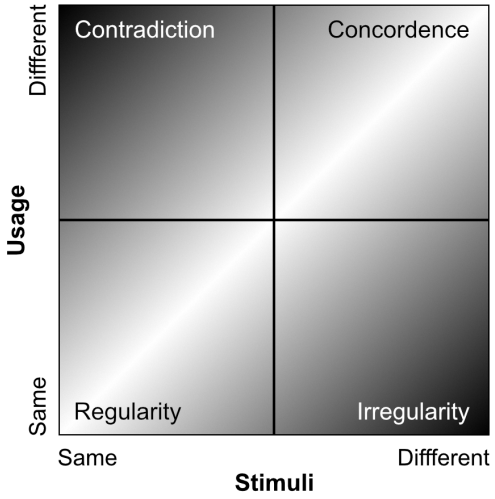


Figure 3.3: The strength of inconsistency

Furthermore, consistency in Human-Computer Interfaces is traditionally considered on three interrelated levels of an application (Grudin, 1989):

- **Internal consistency** - The degree to which a product is consistent with itself.
- **External consistency** - The degree to which a product is consistent with other interface designs familiar to the user.
- **External analogic or metaphoric correspondence** in the real world beyond the computer domain.

This was later simplified by Zuschlag (2010) to only consider internal- and external consistency where the latter describes the degree to which a product is consistent with any reference other than a part of itself. This includes similar products, design systems, standards, metaphors and analog equivalents in the real world.

In general, it is more crucial to consider internal consistency since parts of the same application exist within the same cognitive context and are used consecutively in near time and space. This creates a psychological proximity between interactions where users expect similar things to function in similar ways. The ability to differentiate and regard applications as their own cognitive context results from perceptual differences in visual appearance and the user's knowledge about the application (Zuschlag, 2010). However, although users are able to differentiate and regard ap-

plications as their own cognitive context, external consistency can still drastically improve learnability and performance, especially when working across functionally similar and/or contextually proximate applications.

There are certainly cases when inconsistency can be justified if it affords increased usability or is more contextually appropriate over using standardized solutions. For example, designing a larger checkbox to enable quicker selection in time-sensitive applications or matching it to the style of a themed game to increase immersion. This highlights the fact that increased consistency does not automatically result in improved user experience. Yllobre (2020) points to three fundamental problems with designing for consistency:

1. **Artificial** - Uniformity might look good on paper, but it is boring, disconnected from users and their needs, and is tedious for designers and developers to maintain. Thinking of consistency as the primary goal of your product or design system can result in clean libraries, tidy components, and many things feeling and looking the same, but this does not equal a good experience for your users.
2. **Rigid** - Consistency does not naturally scale to accommodate new needs and can not maximize the benefits of constant change.
3. **Inhibits innovation** - If consistency is the primary goal, we can end up rejecting everything that breaks that consistency, hence there is no room for innovation.

In reality, designers and developers need to make conscious and informed decisions about how and when to be consistent and when to design custom solutions. For instance, applications that are part of the system and used in conjunction might need to accommodate different functionality and prioritize different aspects of the interface.

Since the different AEW&C applications at Saab share a lot of functionality and are used in contextual proximity, a central focus of this thesis project has been to increase external consistency between different tactical AEW&C applications while identifying instances where customization is necessary. Specifically, how to accommodate inconsistency in a way that still makes custom solutions feel part of the same system and behave in accordance with users' expectations. Henceforward, we refer to this ambiguous problem as designing for coherence.

3.3.8 Coherence in Design

Coherence differs fundamentally from consistency in that the primary aim is to bring clarity rather than uniformity to the GUI. Consistency is a product-centered, top-down approach that only considers the relationship between stimuli and usage, which in itself does not require any knowledge about the user. In contrast, Spool (2018) describes coherence as a highly user-centered, bottom-up approach where, instead of asking whether a design is consistent with conventions, standards and similar

applications, the question is whether it complies with the user's *Current Knowledge*, also known as *Knowledge in the Head* (D. Norman, 2013). It is concerned with whether the user's previous experiences will help them understand how to use what is being designed. Answering this question requires in-depth understanding about users which ultimately allows designers to make more informed design decisions. By changing the conversation from consistency to coherence, Yllobre (2020) highlights three major advantages:

1. **Natural** - It focuses the attention on users and solving problems rather than conceiving and blindly following designated design systems.
2. **Flexible** - It is not about maintaining a rigid system of UI elements, instead it is about facilitating for everyone to contribute by enhancing or creating new elements if needed. Product teams go from consumers to contributors.
3. **Enabling innovation** - It helps developers stay agile and responsive to changing and new demands by enabling innovative solutions; considering inconsistency and changes to the current system when justified.

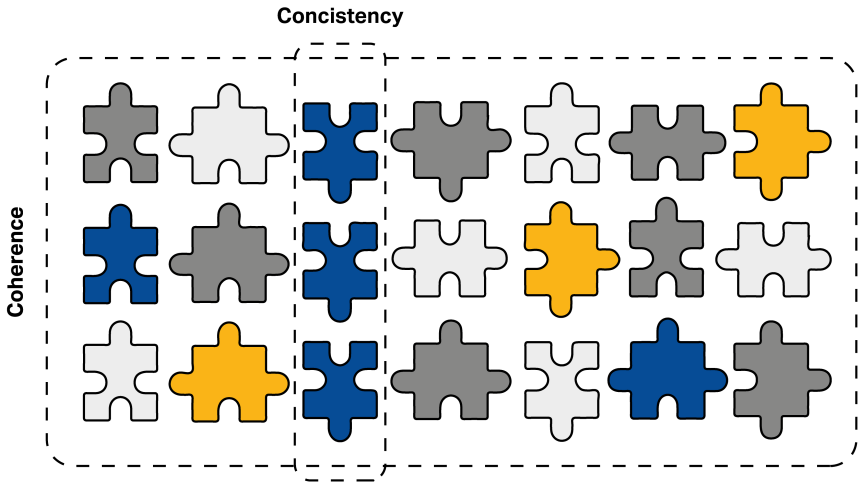


Figure 3.4: The difference between consistency and coherence. Consistency is concerned with uniformity while coherence is about everything fitting together and complying with users' current knowledge (Inspired by Yllbore, 2020)

.

As illustrated in Figure 3.4, coherence still embraces the principles described in consistency but expands the theory beyond uniformity to also consider the user's current knowledge. This allows designers to adjust parts of a system to specific contexts and usages yet still protect the integrity of the design. This quality of adapting to local circumstances is referred to as a system's responsiveness (Harris & Henderson, 2012). Traditionally, coherence and responsiveness are thought of as two opposite ends in one dimension, where the goal is to strike the right balance: "As designers and users, we would like each part of a system to be responsive to local circumstances and also the system as a whole to be coherent." (Harris & Henderson, 2012, p. 67). But Harris and Henderson also explain that coherence and

responsiveness can be studied as two separate dimensions, where a trade-off curve represents the antagonistic relationship between them. As long as we stay on the curve, the dimensions remain in an antagonistic relationship: if we do better on one dimension, we do worse on the other.

But it is also possible to leave the trade-off curve and increase or lower coherence and responsiveness simultaneously. This means that it's possible to jump to higher or lower trade-off curves, as illustrated in Figure 3.5. For example, one of the big pressures that tend to move systems toward lower trade-off curves is increasing scale and complexity, where both coherence and responsiveness tend to deteriorate due to inefficient communication infrastructure.

Another big pressure is sunk cost. Once investments have been made in development and training, it often becomes an anchor that discourages responsiveness. In the meanwhile, the context and usages of the system keep changing, necessitating patches and amendments to deal with unexpected circumstances. Consequently, as features are altered and added without considering the users or the integrity of the design, coherence is compromised in favor of responsiveness and at the expense of usability for end-users and maintainability for designers and developers. As put by Harris and Henderson (1999) in *A better Mythology for System Design*: "evolving systems need to maintain both agility [responsiveness] and coherence. If a system isn't agile [responsive], it cannot respond effectively to local variations or global change. If it loses its coherence, it cannot act effectively as a system at all." (p. 93).

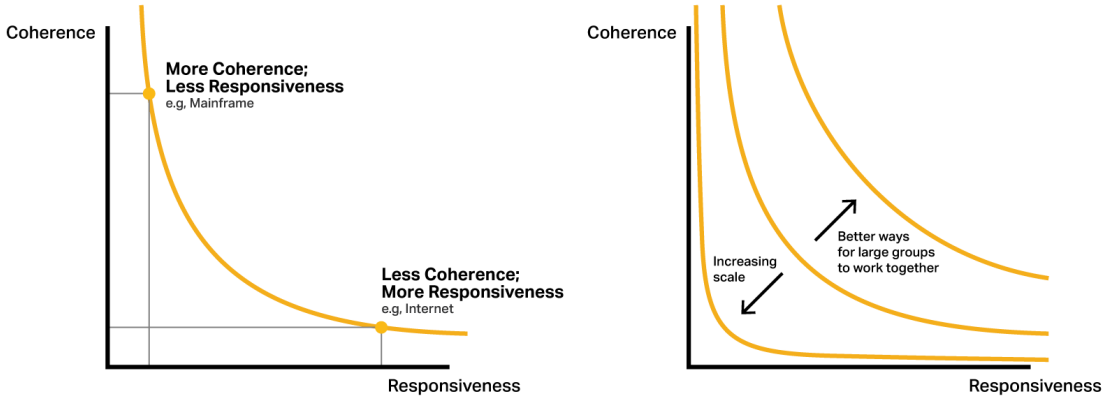


Figure 3.5: Tension between coherence and responsiveness in two dimensions (inspired by Harris & Henderson, 2012)

.

To counteract these negative effects in a scaling, complex business environment Harris and Henderson (2012) emphasizes the need for platform improvements and collaboration tools that aim to raise the system to higher trade-off curves, see Figure 3.5. This higher trade-off curve represents helping designers and developers manage very complex coordination and ultimately improving both coherence and responsiveness at the same time. Some of these collaboration mechanisms include company

wikis, version control systems, communication channels, and design systems. Finally, Harris and Henderson (2012) highlights that the infrastructure itself needs to be permeable to stakeholders' preferences and judgments. Here, designers and developers can be considered primary users, where the quality of the infrastructure design influences the ability to design for coherence and responsiveness.

3.4 Design System

Today, the concept of Design Systems is a growing and documented field within design and practiced within small and large organizations (see 5.4.2 Benchmarking) worldwide. There are, however, still no prevalently accepted single definition that encompasses everything a design system entails. At broad, a design system is commonly described as a set of principles and guidelines for managing application development at scale by reducing redundancy and creating a unified language and visual coherence across multiple applications in an organization. A few documented definitions are:

> *"A complete set of standards intended to manage design at scale using reusable components and patterns." (Fessenden, 2021, para. 2)*

> *"A collection of reusable components, guided by clear standards, that can be assembled together to build any number of applications." (Suarez et al., n.d., p. 7)*

> *"A series of documented elements, components, and regions that include both design and front-end guidelines." (Vesselov & Davis, 2019, p. 16)*

For the purposes of this project, where the main focus is centered around designing the system itself, including components and patterns for AEW&C applications, the thesis duo consider a design system to be:

> *A collection of reusable, coherent components and patterns built on fundamental rules and attributes.*

3.4.1 Architecture

According to Fessenden (2021) a design system can be architected in many different ways depending on different organizational needs. Commonly, however, design systems are comprised of a style guide, a component library, and a pattern library:

- **Style guide** - Contains high-level details about brand, logo, icons, colors, and typography along with implementation guidelines such as voice-and-tone, visual references, and design principles.

- **Component library (or UI element library (Suarez et al., n.d.))** - Contains core UI components that can be used and shared across teams. Each component should preferably have a name and description and provide guidelines, attributes, and states. Component libraries may or may not include static or interactive examples of backend- and frontend code (Fessenden, 2021; Vesselov & Davis, 2019).

- **Pattern library** - Specifies combinations of components (or UI elements (Suarez et al., n.d.)) into patterns with their own name, description, guidelines, and attributes.

Together, they form the design system to help create a systematic architecture and guide development (Edelberg & Kilrain, 2020).

Atomic Design (Frost, 2016) presents a concrete description and *mental model* (D. A. Norman, 2014) of design systems that aims to bring clarity to the different levels comprising the system. The book breaks down a design system based on chemical metaphors, where *atoms* refer to individual UI elements that constitute the fundamental building blocks of an interface. Combinations of atoms form *molecules*, which are simple component groups such as an input field with an accompanying action button.*Organisms* denote larger groups used extensively throughout an application, also known as patterns (Fessenden, 2021). This is commonly used for different panes, such as headers, navigation and footers. Finally, templates and pages are meant to demonstrate the pattern library in use, manifesting relationships between atoms, molecules and organisms and populated with placeholder content and real content respectively. This is illustrated in Figure 3.6.

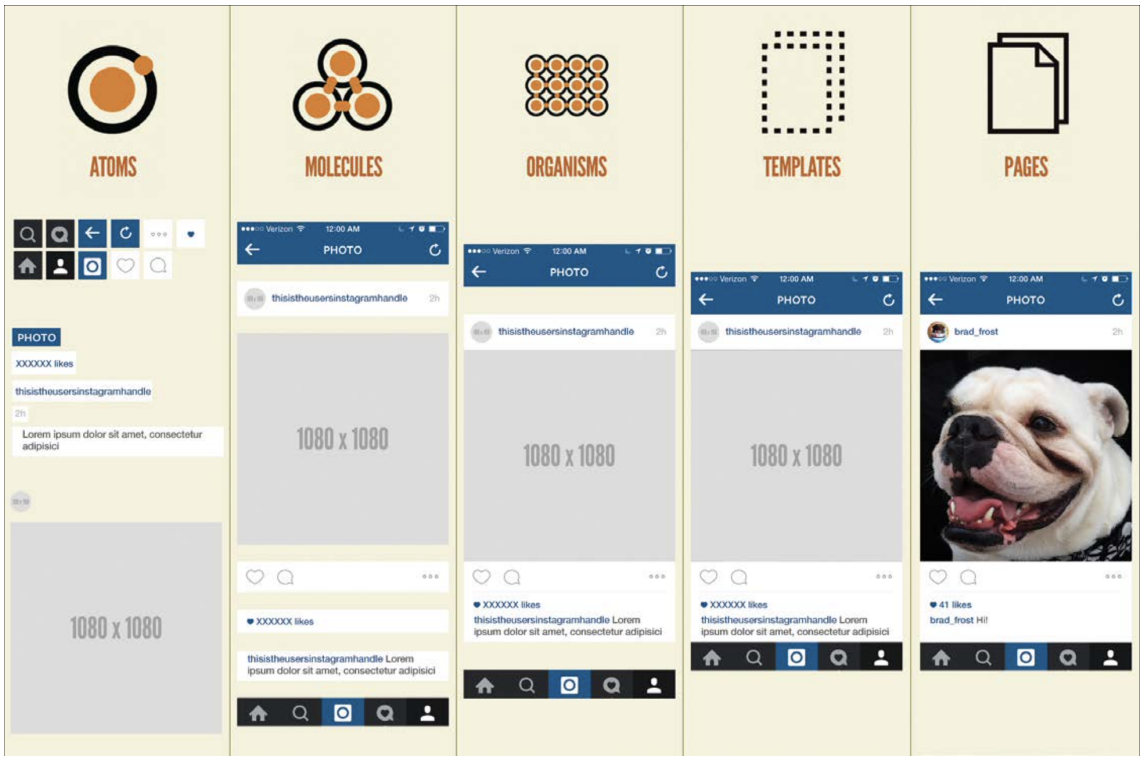


Figure 3.6: Breakdown of design systems according to the atomic mental model (Frost,2016)

.

- **Atoms** - UI elements that can not be broken down any further and serve as the fundamental building blocks of an interface.
- **Molecules** - Collections of atoms that form relatively simple UI components.
- **Organisms** - Relatively complex components that form discrete sections of an interface.
- **Templates** - Places components within a layout and demonstrates the design's underlying content structure.
- **Pages** - Applies real content to templates and articulate variations to demonstrate the final UI and test the resilience of the design system.

The atomic framework has a built-in hierarchy which helps to structure and educate stakeholders of where certain parts of the system belong. However, some organizations might need a different taxonomy to better fit their communication, mental model and internal structure. For instance, Vesselov and Davis (2019) has a similar approach to describing the basic architecture but instead of chemical metaphors, a more rudimentary terminology is used to describe the different parts:

- **Elements** - The lowest-level object. Elements cannot be broken down further.
- **Components** - A combination of one or more elements that function as a whole, such as a form.
- **Component groups** - A group of components that form a larger component

Here, a single component is comparable to an atom whereas component groups serve the same purpose as molecules and organisms. Furthermore, Vesselov and Davis (2019) coins the term *elements* to describe lower-level objects which can not be considered components by themselves. This can include labels, buttons, icons, and inputs which do not serve any function until combined into, for instance, a form or a dropdown menu.

Suarez et al. (n.d.) and Curtis (2016b) break down design systems yet further by introducing subatomic building blocks called *design tokens*. Design tokens are described as the foundation of a design system and an abstraction for everything impacting the visual design of an application. This includes fundamental attributes such as colors, typography, spacing, and sizing, but also animation durations and *visual form*. Visual form is the material properties of the UI, including "background images, gradients, and textures, shadows and elevation (z-indexes), rounded corners, and borders" (Suarez et al., n.d., p. 63).

Baldwin (2021) attempts to present a holistic model describing the anatomy and relationships between different parts constituting design systems, see Figure 3.7. In this model, a design system may cover everything from designer-facing aspects, such as the design process, design assets, and UI visual language system, to developer-facing aspects such as documentation, component libraries, and development processes. The model makes clear that a design system covers many overlapping areas from different parts of an organization, where resources are used and shared between different types of users.

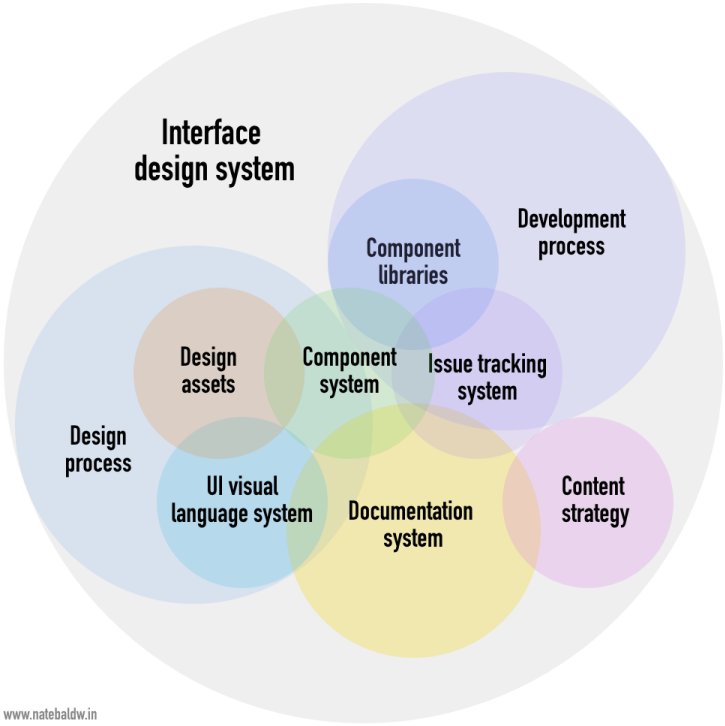


Figure 3.7: All the overlapping entities & resources that make up a design system (Baldwin, 2021)

.

3.4.2 Benefits of Design Systems

The premise of building and using a design system is that it brings many benefits to organizations in terms of collaboration and efficiency, which ultimately allows teams to create more consistent and coherent experiences for end-users. In this section, some of the most prominent advantages are presented.

3.4.2.1 Collaboration

A design system facilitates communication and collaboration between different teams by providing a common place for documenting and sharing resources continuously. Depending on the structure, a design system could include a glossary section with definitions of commonly used terms, including abbreviations, to promote consistency and, in turn, understanding between everyone in the organization (Frost, 2016). This effectively improves collaborative workflows across all disciplines (Suarez et al., n.d.) and helps educate and onboard new team members. A consistent and shared vocabulary also encourages teams to refer to components and patterns using correct GUI terminology (Suarez et al., n.d.), such as “toolbar” and “carousel”, instead of each team coining internal, proprietary names.

In addition to hosting vocabulary, a design system commonly also functions as a code repository. The collaborative advantage of including code alongside design re-

sources is twofold: it facilitates designer-developer hand-off by clarifying how design solutions should be implemented (Vesselov & Davis, 2019), and simultaneously allows developers to share code between multiple teams and codebases (Suarez et al., n.d.), resulting in improved coordination throughout the development process.

3.4.2.2 Efficiency

Design systems address a common bottleneck in development and design alike; the need to update documentation whenever a change is introduced (Vesselov & Davis, 2019). For example, when changes are made to a component, all screenshots for the manual must be updated to reflect the new appearance. Design systems can circumvent this entire process by using dependencies. Patterns can be generated by referencing the code tied to individual components, which in turn references elements and tokens. Screenshots, hence, can be automatically generated and updated to reflect even the smallest changes. This dependency phenomenon also helps to keep the code "DRY", meaning that there is no unnecessary duplication of code. (Frost, 2016)

Moreover, since the applications are powered by the reusable patterns and components of the design system, all applications automatically inherit all updates made to the design system, reducing the effort of maintaining the actual application. In effect, developers and designers can make changes to a component in the repository and see that pattern automatically updated wherever it is employed. (Frost, 2016)

Another benefit of design systems is that they allow teams to prototype much more efficiently and reliably. Collecting and organizing everything in one place increases the discoverability of existing resources, making them more likely to be found and utilized (Suarez et al., n.d.). Instead of constantly reinventing new solutions to old problems, developers can simply explore, modify and assemble existing components, patterns, and quality-approved code. Furthermore, since the design- and code resources exist in the same place, it is easy to keep them in sync with each other. Suarez et al. (n.d.) explains that "the closer the [resources] are to each other, the easier it is to trace and manage dependencies between bits of code, and the easier it will be to update and maintain" (p. 86). When major changes eventually are made to extend and improve upon the UI and UX flows, a design system keeps the code overhead low, while still allowing the application to grow and evolve, and reduces the effort from hundreds of lines of code to as little as a few characters (Suarez et al., n.d.). This effectively streamlines system development to become more efficient and responsive to new insights and requirements.
Readily available, standardized components allow designers to spend less time focused on style and more time developing a better user experience (Suarez et al., n.d.). Explicit rules and guidelines also help onboard new individual contributors who are new to the system or UI design in general, serving as an efficient and persistent tool for education and reference (Fessenden, 2021). This allows new members to reach proficiency faster while liberating senior members from time-consuming tutoring and supervision.

3.4.2.3 Consistency & Coherence

Without a systematic approach to design, individual contributors often end up with custom solutions to improve isolated areas of an application. The problem with this is that it causes visual and functional fragmentation, adding to design and technical debt. With a design system, new solutions can be shared across teams by implementing them right into the system. Since components are interdependent, a change in one location also propagates throughout the entire system. This makes style updates within a system trivial in effort but much greater in impact (Suarez et al., n.d.). Major visual rebrands and redesigns can be managed at scale through the design system (Suarez et al., n.d.) without causing inconsistency or incoherence between different applications. Moreover, design systems promote individual contributors to adopt a more empathetic workflow; considering existing tools, rules and best practices when approaching a new problem. "Designers and developers are forced to think about how their decisions affect the broader design system" (Frost, 2016, p. 32) which guides decisions to align with the rest of the system. This, in itself, acts as a powerful mechanism for maintaining and increasing consistency and cohesion. As the design system is adopted, made into a single source for design resources, and applied in different projects, all applications relying on the system will naturally become consistent and coherent with each other. Applications will automatically evolve to look and behave in similar ways and appear to be part of the same ecosystem.

3.4.2.4 Usability & Accessibility

The combination of using consistent, recurring components, patterns, and vocabulary in application development ensures that the relationship between stimuli and function remains consistent. Consistency, in turn, helps users learn and familiarize themselves with the relationship between stimuli and functionality (see 3.3.7). This accelerates the learning and mastering process for single and multiple applications, allowing users to achieve lower error rates and higher satisfaction. Ultimately, this increases the usability of the applications that rely on the design system (Frost, 2016).

In addition to usability, the same dependency phenomenon also helps developers build accessible applications. Many accessibility considerations can be implemented at token- and component level by optimizing fundamental attributes such as sizing, colors, durations, and language to accommodate for users with varying abilities and backgrounds. Furthermore, optimizing for accessibility often benefits regular, able-bodied users as well (Jordan, 2002, p. 10). Once these fundamental attributes have been considered, developers can simply deploy the patterns and reap the rewards (Frost, 2016). Frost (2016) highlights the ability to leverage these best practices at scale as being one of the most prominent advantages of establishing a design system. It means that users, developers, and designers do not have to be senior-

level to produce good work:

> *"The design system serves as a quality control vehicle that helps users apply best practices regardless of each individual's skill level." Frost (2016, p. 154)*

3.4.3 Designing a Design System

Designing and building a comprehensive design system is a monumental task as there are a myriad of factors to consider at many different levels: From defining a visual style and how to design for modularity and scalability, to how it will be used by other teams and how the idea should be presented to decision-makers at the company (Suarez et al., n.d.). A design system can be described as a "living, breathing entity" (Frost, 2016, p. 143), meaning that it is not a matter of delivering a finished product at the end of a project. A design system needs to be iterated continuously to maintain relevance in the dynamic landscape of software development.

> *"A sound design system doesn't roll off an assembly line, but is rather sculpted in iterative loops, building up fidelity as the project progresses." (Frost, 2016, p. 121)*

Therefore, Suarez et al. (n.d.) emphasizes that design systems should be designed and maintained by a large, multidisciplinary team. The team should preferably consist of designers, developers, accessibility experts, brand specialists, user researchers, product managers, and more in order to ensure that the system caters to all relevant stakeholders.

> *"Before diving into the design process, start by considering who needs to be involved in the creation of your design system and how the team will work together" (Suarez et al., n.d., p. 25)*

3.4.3.1 Approach

Generally, there are three ways for organizations to approach design systems: adopting, adapting, or creating their own (Fessenden, 2021).

Adopting an existing design system is the lowest-cost alternative as it circumvents the time-consuming process of building and maintaining the system. However, this approach allows for less customization and brand differentiation. Adapting an existing design system can significantly accelerate the first steps of the process by building on a foundation on which the company can alter the style guide and components to their needs. Still, there are strong reasons why a company might want to develop their own system from scratch.

Firstly, initial cost-savings gained from adopting or adapting an existing system will diminish as more customizations and adjustments are made over time (Fessenden, 2021). Open-source design systems, such as Google's Material Design

(https://material.io), are often associated with the identity of the company that produced them (Marchi, 2021). A proprietary design system, instead, allows the company to incorporate their own brand identity from the bottom up. This means that the company can tailor the style guide, the component library, and the structure of the system specific to their own needs from the start. Secondly, if a company has particular needs; including highly customized components, special coding frameworks, or tight security protocols, that can not be met by open-source design systems, it would be a better long-term investment to build a proprietary system either way (Fessenden, 2021).

3.4.3.2 Process

The process of designing a design system should be highly structured and collaborative and should involve cross-disciplinary stakeholders from the start (Frost, 2016). The beginning of the process naturally involves UX design to a large extent since the bulk of the elemental work involves researching organizational culture and workflows in order to elicit stakeholder needs and requirements (Vesselov & Davis, 2019). In order to do so, developers and other stakeholders must be consulted to ensure the resulting system aligns with and supports their work. As the project progresses, the collaborative process and workload between stakeholders should iterate between UX research, visual design and development roughly according to Figure 3.8 (Frost, 2016).

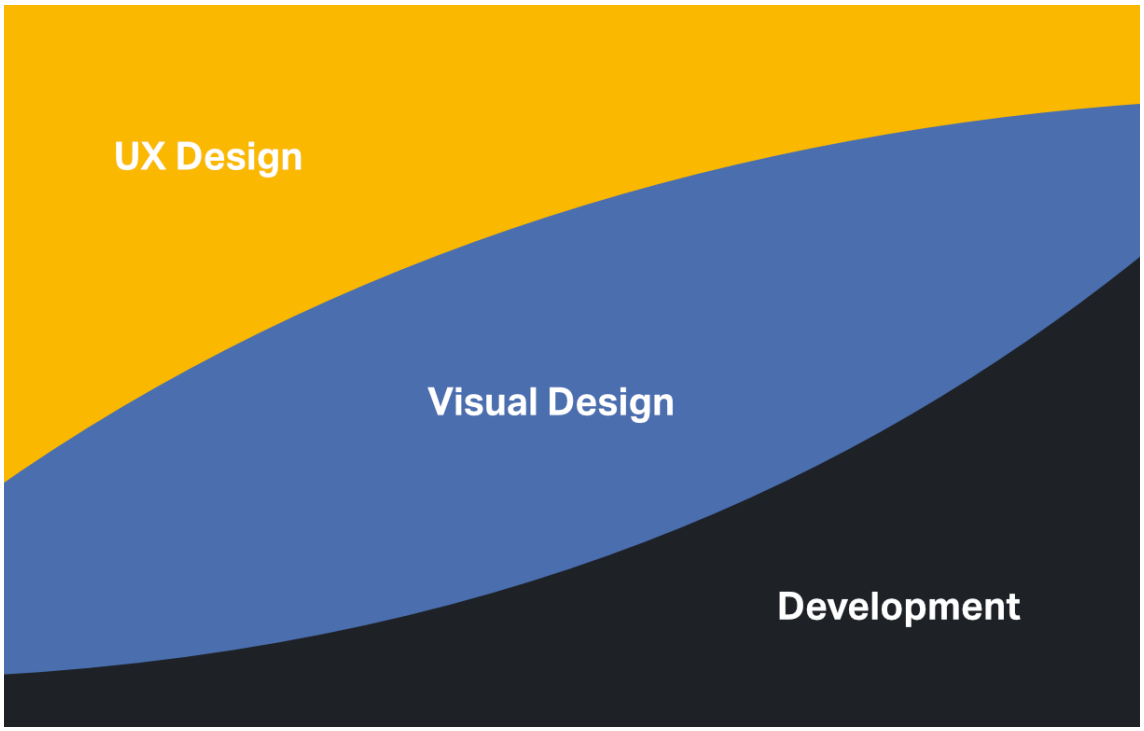


Figure 3.8: The iterative and collaborative process of designing a design system (adapted from (Frost, 2016, p. 118))

.

3.4.3.3 Content

According to Vesselov and Davis (2019, p. 23) a robust design system contains "guiding design principles, guidelines for use and implementation, as well as a component library that includes front-end code." Furthermore, Vesselov and Davis (2019) presents this approach as considering design systems as a language system, where the content of a design system can be divided into a *component library* (lexicon) governed by *guidelines* (grammatical rules). Here, the content of the design system should also, according to Suarez et al. (n.d.), follow a set of guiding principles:

- **It's consistent** - The way components are built and managed follows a predictable pattern
- **It's self-contained** - Your design system is treated as a standalone dependency
- **It's reusable** - You've built components so they can be reused in many contexts
- **It's accessible** - Applications built with your design system are usable by as many people as possible.
- **It's robust** - No matter the product or platform to which your design system is applied, it should perform with grace and minimal bugs.

Component library Depending on the state of the organization the component library can be built using either a bottom-up or top-down approach, or a combination of the two (Vesselov & Davis, 2019).

The bottom-up approach involves starting from the simplest tokens and elements and defining fundamental attributes from which the entire system can be built: a spacing system, grid system, color system, typography and iconography. Here, it is recommended to create a spatial system derived from increments of a base number. Using a base number of 4 pixels is currently growing in popularity as the recommended scale for many reasons. The most compelling reason being that operating systems use icons based on multiples of 4 pixels (such as 16, 24, 32, etc.) which ensures proper scaling (Suarez et al., n.d.). Having defined a base number, this unit of measure can then be multiplied by a factor and applied to attributes such as component paddings and margins (Curtis, 2016a; Vesselov & Davis, 2019). All design decisions involving colors, typography and iconography should be motivated by principles based on either usability or accessibility, or align with the organization's preexisting design language Furthermore, iconography should be based on an established icon grid and core shapes in order to maintain proportions and visual consistency. The color system should also preferably employ priorities to reduce decision making during the design process. Color priorities can be achieved by creating a system similar to font weights and assigning colors a numbered value that corresponds to the hierarchy in which they should be used. (Vesselov & Davis, 2019)

The top-down approach to building out a component library involves breaking down and creating an inventory of the current interfaces found within an organization (see 4.2.3) and analyzing the individual components to find similarities that can be used

as foundation for turning design decisions into a system (see 4.2.4) (Vesselov & Davis, 2019). Here, fundamental attributes such as colors, typography, spacing, visual form and icons should be documented to identify and define the visual language of the system (Suarez et al., n.d.).

Having defined fundamental attributes, components and patterns can then be built as they are needed for different products and interfaces. Components can be built using either a UI development environment, such as Storybook, design software, such as Figma or a front-end framework, such as Bootstrap. Regardless of method, it is crucial to keep reusability and scalability in mind to ensure components can be used in many different contexts without feeling out of place. In order to achieve this, Suarez et al. (n.d.) suggests that designs should be quickly translated into the actual context of use and reiterated from there. This means that designers and developers should collaborate closely in iterative loops of designing components and building GUIs. As each lower-level part of the design system becomes more fully baked, any application GUI that includes the components and patterns will become more fully baked as well (Suarez et al., n.d.). Adhering to **Frost** mental model of atomic design, designers can "concurrently create the final UIs and their underlying design systems" (p.57).
Guidelines In addition to providing reusable components and patterns, explicit guidelines need to be specified and documented in a design system in order to make it useful in practice. Here, it is important to consider that there are several kinds of users within different disciplines that are in need of different kinds of documentation. For instance, designers might need to know how a certain pattern should be handled in order to achieve a consistent and user-friendly interface, while a developer might need to know how to reference a component or pattern directly in code. Here, it is useful to consult each relevant discipline of the organization and benchmark how other design systems are documented for inspiration and guidance (Vesselov & Davis, 2019).

(Vesselov & Davis, 2019) recommends starting off by defining 3-5 solid, design principles; short and descriptive sentences that combine organizational goals with user needs. These design principles should aim to aid users of the design system in making crucial design decisions. Effective design principles will align the organization and create coherence within the different products and across teams (Vesselov & Davis, 2019). For sake of efficiency and clarity, it is preferable to use distinct descriptions, such as "Bring sharp focus by helping customers know what matters now" as opposed to single words, such as "minimal and efficient" (Vesselov & Davis, 2019, p. 59).

The documentation should also include clear guidelines associated specifically with each component and pattern in the design system (Frost, 2016; Suarez et al., n.d.; Vesselov & Davis, 2019), importantly:

- **A formal definition** - An explicit and clear description of each component and pattern in the library

- **Usage guidelines** - How to utilize each component and pattern when building interfaces, including do's and don'ts, behavioral rules and variations. Here, it may also be helpful to define contextual rules that describe how a component and pattern can, and should be modified or customized when local circumstances necessitates deviation from the typical way of implementation.
- **Technical guidelines** - Specification of code, class names and data attributes that help developers reference and properly implement the component or pattern when building applications. Here, it may also be helpful to include live code examples that are tied to the graphical representation of the component or pattern. This way, as attributes are programmatically modified, the graphical asset also changes to reflect the code.

3.4.3.4 Design System Repository

A design system repository (DSR) hosts and visualizes the design system and its documentation in its entirety. A standalone DSR functions as a single source of truth and it collects and structures the design system into pages and packages that are navigable and searchable (Suarez et al., n.d.). Each page can contain text descriptions, guidelines, graphical assets, and code examples that help developers fetch resources and understand how to implement reusable components such as buttons, alerts, and navigation in their applications. Furthermore, the DSR can also include hyperlinks to related components to facilitate navigation and discoverability across the system (Vesselov & Davis, 2019). Modern tools also support continuous and collaborative contribution in real-time, helping teams to communicate and produce coherent yet responsive designs. DSRs are usually implemented and accessed through either public or private websites, or desktop applications. Companies can develop their own DSRs from scratch or use existing tools. Some of the most popular DSRs currently on the market are:

- Storybook
- Backlight
- InVision
- Figma

Using a separate DSR to host the design system and making it accessible across the organization brings many long-term benefits, including the ability to manage versioning, code sharing across multiple teams and infrastructure for contributing to the system (Suarez et al., n.d.). Importantly, it also forces individual contributors to develop components in isolation so they are not tied to a single use case (Suarez et al., n.d.).

3.4.3.5 Balancing Coherence and Responsiveness

In order to achieve an effective and useful design system, it needs to be designed such as it balances coherence with responsiveness. This means that the design system should strike a balance between acting as a rigid system and being flexible to accommodate for contextual variations needed within different applications. If

a design system does not provide enough flexibility, it cannot effectively support developers and designers in building custom applications - leading to general, impercise solutions to local design problems However, if it loses its coherence with itself, it also surrenders its ability to act as a system at all - leaving design decisions open for individual taste and interpretation (Suarez et al., n.d.). Returning to Harris and Henderson (2012), this means that the design system can be seen as an infrastructure that needs to be permeable to stakeholders and contextual requirements in order to minimize the trade-off between coherence and responsiveness.

> *"A design system is not meant to be a rigid set of rules that remove creativity from the process. You are allowed to create deviations in your guidelines so long as you are intentional in your choices and document your decisions. Be strict but understand that outliers are sometimes necessary to improve your interface and user experience." (Vesselov & Davis, 2019, p. 58)*

There are several ways to manage this trade-off between the extremities. Firstly, coherence is inherently achieved by adhering to rules and building components from the same set of fundamental tokens. On one hand, this allows different applications to rely on shared design tokens which ensures consistency, but on the other hand, each application can override certain tokens to create their own unique theme that propagates throughout the interface (Suarez et al., n.d.). This means that designers are given very deliberate control over when to adhere to the default, systemized solution and when to deviate. Backus (2021) refers to this phenomenon as *customization* and highlights that it can provide more freedom and flexibility that may lead to more innovative approaches. Combining this with contextual rules; explicitly documenting when and how a certain component or pattern can, and should deviate, both responsiveness and coherence can be efficiently balanced (Vesselov & Davis, 2019).

> *"Flexible styles can transform the same component, giving it a completely different visual language while maintaining the same foundation" (Vesselov & Davis, 2019, p. 65)*

In addition to relying on tokens, Suarez et al. (n.d.) defines a set of guiding principles that components should follow in order to make them more flexible and reusable in different contexts:

- **Modular** - Modular components are self-contained with no dependencies
- **Composable** - Composable components can be combined to create new patterns
- **Generic** - Generic components can handle multiple use cases
- **Flexible** - Flexible components can be tweaked and extended to work in a variety of contexts

Flexible components introduce additional parameters tied to its visual and func-

tional properties that allow for more varied component behaviors (Backus, 2021; Suarez et al., n.d.). This phenomenon is also referred to as *configuration* (Backus, 2021) and allows individual components to be configured from a set of default options that helps designers and developers better cope with contextual variations in a coherent manner.

Customization and configuration can be combined and used for different components and patterns depending on their maturity and need for flexibility. As a rule of thumb, configurability is generally preferable when working with established, mature components that are used extensively, whereas customization can offer more flexibility and be more beneficial when prototyping new or highly bespoke features. (Backus, 2021)

Another important aspect in maintaining the balance between coherence and responsiveness is to allow frequent and collaborative reiteration, modification and contribution to the design system by all stakeholders in the organization. Vesselov and Davis (2019) points out that "the more team members you have using and contributing to it [the design system], the more cohesion your organization will experience". Finally, guidelines should be kept as rigid as possible but stay open to change throughout the lifetime of the system.

> *"Keep in mind that a design system, just like a language, can and should evolve over time. Change is a natural part of a language that allows new functions and concepts to form. As you begin to craft your own design language, be rigid with your guidelines but be open to change as your system adapts and grows." (Vesselov & Davis, 2019, p. 62)*

4
Methodology

In this chapter, the methods employed in the thesis project are described. The methods have been selected based on their appropriateness to answering the research question and are organized by the stage in which they are used in the design thinking framework.

4.1 Empathize

The first stage of the design thinking process is about empathizing with the users and the problem space. Here, research is conducted to get a deeper understanding of what the users do, say, think, and feel (Gibbons, 2016). Empathy is a crucial part of a human-centered design process since it allows the designer to disregard their own assumptions and opinions about the subject in order to gain insight into users and their needs (Dam, 2021).

4.1.1 Literature study

Conducting a literature review has shown to have several benefits in a project. It is a tool to use to gain a good overview of a field where you are less familiar or to reveal what has already been done. A literature review can also help with ideation and creating new ideas by getting a deeper understanding of the design space (Knopf, 2006).

4.1.2 Interviews

Interviews are used for gathering qualitative, firsthand personal, data from users by initializing conversations with them. The purpose of an interview is to get a deeper understanding of the user's experiences, opinions, and perceptions about the product, service, or a specific task. The structure of an interview can be different depending on your intended goal with the interview. Interviews can be either structured, following a script of questions, or unstructured, carried out in a more conversational manner. Further, semi-structured interviews can use both scripted and unstructured approaches during the interviews (Hanington & Martin, 2012, p. 102).

4.1.3 Contextual Inquiry

To study user situations and identify user needs in their context, a contextual inquiry can be performed. This is a method that fosters participatory design by inviting the user to show and describe their current work practices and associated experiences (Holtzblatt & Jones, 1993). The inquiry typically involves in-depth observations and interviews with users in their natural environment to gain a greater understanding of behaviors and their way of working (Raven & Flanders, 1996). According to Raven & Flanders, a contextual inquiry is based on three principles, namely:

1. **Context** - Data gathering must take place in the context of the users' work.
2. **Partnership** - The designer and the user form a partnership to explore issues together.
3. **Focus** - The inquiry is centered around a specific focus area; it is based on a clearly defined set of concerns, rather than on a list of specific questions (as in a structured interview).

4.1.4 Try it Yourself

Try it yourself is a method that allows the designer to try out a product or system by acting as the user. This is done to gain a sense of empathy for the user, the usage, and the context. By trying a product or service for oneself, the designer acquires first-hand experience and understanding that helps mitigate assumptions (Think Design, 2020). In this thesis, the try it yourself method will be used to get hands-on experience with the tactical applications to get an in-depth understanding of how the GUIs are structured, how to navigate the GUIs, and how specific components are used.

4.1.5 Benchmark

Benchmarking is a method used to get a broader understanding of the product market and the competition. It is a process of identifying critical development areas and opportunities to help an organization improve their performance (Anand & Kodali, 2008). In their paper, Anad & Kodali define benchmarking as "the search for the best industry practices which will lead to exceptional performance through the implementation of these best practices" (p.258). Thus, benchmarking is an activity where you look outwards from your organization to see what others have done and how they have solved problems. Benchmarking can be categorized into two classes, namely, internal and external benchmarking. In this thesis, external benchmarking is used to gain insight in how other organizations have organized their design systems, what elements they have chosen to include, and the structure of their design system repository.

4.2 Define

In the define phase data gathered in the empathize phase is compiled and analyzed in order to get a more coherent understanding of the problem and to pinpoint opportunities for innovation (Gibbons, 2016). The phase aims to synthesize findings to define the core problems that have been identified. The aim should be to define the problem as a problem statement in a human-centered manner (Dam, 2021). The define phase builds the foundation for the next phase - the ideation phase - by gathering great ideas to establish features, functions, and other elements that will help solve the problem.

4.2.1 Affinity Diagram

Affinity diagramming is a method used to externalize and cluster insights and observations from the user research (Hanington & Martin, 2012). Instead of relying on implicit information and knowledge acquired through research methods, affinity diagrams aim to synthesize what has been captured in terms of insights by documenting it on individual sticky notes. By clustering the sticky notes based on themes, or affinity, it is possible to observe emerging trends and patterns in the data. Hanington and Martin (2012) further emphasizes that affinity diagramming is an inductive exercise; instead of grouping the notes based on predefined themes, the work is done in a bottom-up manner by letting the themes emerge as notes of similar meaning are clustered together. Performing an analysis through an affinity diagram in an early stage of the process helps design teams to categorize large sets of data and gain a common understanding of the problem space (Hanington & Martin, 2012).

4.2.2 Problem Statement

A problem statement is an actionable formulation of a wicked problem, summarizing the user, their needs, and goals. The user in this case should be based on a real user segment identified through research while the needs and goals should be focused on what the user is trying to accomplish. Here, it is important to avoid solution-oriented terminology such as specific features and technologies. A problem statement is commonly formulated in the define phase in order to specify the scope of the design problem and provide a measurable goal for the design process. It may also be reiterated at multiple stages of a project in order to better reflect new insights and decisions made by the design team. It should act as a concise problem description that helps align the design team towards a common goal (Gibbons, 2019).

4.2.3 Interface Audit

An interface audit is conducted to create a comprehensive collection of all the elements that make up a user interface and can be used as a first step in the creation of a design system. The goal is to create an interface inventory that includes a categorization of all the elements in your user interface. The easiest way of doing

this is by simply taking screenshots of the UI, including all the different UI elements (Frost, 2016). According to Frost, there are five steps in the process of conducting an interface audit:

1. Open the project - To make an effective interface audit, representatives from all disciplines responsible for UI should be gathered for the exercise; UX designers, developers, copywriters, content strategists, project managers, and other stakeholders.
2. Prepare for screenshotting - Decide what tool you will use to gather all the screenshots and categories of the UI elements. To make it easier in the end, make sure to use the same tool so that you are able to combine the files.
3. Start screenshotting - Screenshot all the different UI elements and patterns in the program. These elements could be everything from menus and icons, to buttons and colors. Rather capture one instance of each unique UI element than capturing every instance of that particular UI element. To make the exercise as efficient as possible, assign each participant with a specific area of the interface or category of UI elements.
4. Categorize screenshots - Categorize all the different UI elements and patterns in the program. You can do this as you go or after your screenshotting session, but the goal is to be able to view all the different types of a particular UI element side by side.
5. Present findings - Time to share the findings with the rest of the team and gather all the findings in a large document. It is not that important what you choose to call each element at this point, it is better done at a later stage. Next up is to decide on what patterns to keep, what to name them, and what are the next steps in transforming the interface inventory into a living pattern library.

As put by Frost (2016): "By taking the time to organize the parts, you can now create the whole in a more realistic, deliberate, and efficient manner" (p. 108). Instead of sifting through a large, random pile of building bricks (components), an organized inventory of components (a design system) helps achieve a more deliberate and time-efficient design process, as illustrated in Figure 4.1.

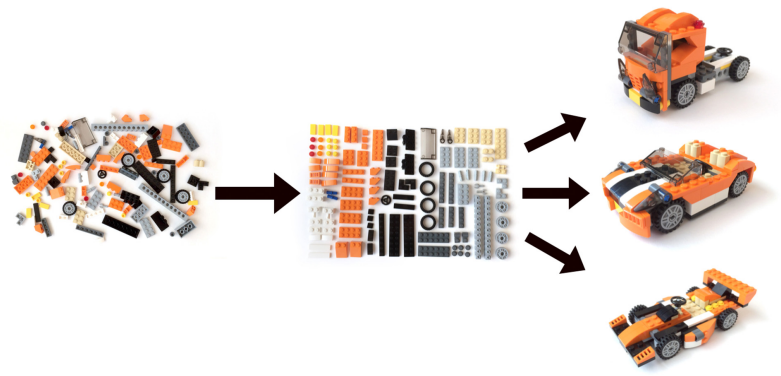


Figure 4.1: An organized inventory of components (Frost, 2016)

4.2.4 Heuristic Evaluation

Heuristic evaluation is a research method used to systematically assess interfaces against a defined set of usability principles. It is traditionally performed by using established heuristics such as Jordan's Principles of Usable Design (Jordan, 2002) or The Nielsen-Molich (1990) heuristics but researchers can also define their own sets of heuristics that better serve their study. The heuristic evaluation method allows researchers to iterate through and focus directly on individual elements of an interface, pinpoint usability issues, and determine their impact on the overall user experience. As the choice of heuristics influences the outcome of the evaluation, it's important to carefully consider the choice and number of heuristics. It is recommended to use 5-10 heuristics for a thorough yet contained evaluation ("What is Heuristic Evaluation?", n.d.). Furthermore, to make the evaluation exhaustive, it's important that the evaluation is performed individually and by several researchers before combining the findings (Nielsen, 1994).

4.2.5 Requirements

By effectively and systematically analyzing the data found during the empathize phase, it can be translated into a detailed design specification, so-called requirements. Sharp et al. (2019) defines a requirement as "a statement about an intended product that specifies what it is expected to do or how it will perform." (p. 378). Hence, requirements are qualifiers that something is desirable, or undesirable, and its purpose is to control and set the direction for the design work. Further, requirements should preferably be unambiguous and solution-independent, meaning that all possible solutions, regardless of the design, should meet the requirements (Bligård, 2015). The requirements should be created iteratively during the design phases and could be derived from various design methods and activities (Sharp et al., 2019). This is to allow the designer to better understand the requirements as they evolve during the process. By identifying user needs, they can be translated into product-specific requirements to help design a product that fulfills the users' needs. The requirement can then be captured and visualized in, for example, a prototype of the intended product concept.

4.3 Ideate

Ideation methodologies aim to generate innovative and creative ideas and diverge the solution space for a given problem statement (Dam, 2021). This section describes methods that are used to address user needs and explore the solution space in a broader sense; no idea is too far fetched and quantity supersedes quality (Gibbons, 2016).

4.3.1 Brainstorming

Brainstorming is a group activity used to generate a large number of quick ideas for solving a stated goal; quantity over quality (Hanington & Martin, 2012). It has

the advantage of one being able to generate wild ideas without being criticized or judged. Harley (2017) refers to brainstorming as a technique to "go for quantity, there are no bad ideas, crazy is welcomed, and encouraging building off ideas". By taking advantage of the cohesion of a group, and without having to worry about judgment and criticism, past ideas can inspire to create new ones (Dam & Siang, 2020a).

4.3.2 Sketching

Sketching is a tool to visualize and create preliminary ideas and solutions, and to build on the ideas of others. It is a process to explore the design space and communicate ideas through rapid sketches, instead of just using words (Dam & Siang, 2020a). Here, the focus lies on rough sketching and quick ideas and less on creating a beautiful drawing; the sketches should be as simple and rough as possible with just enough details to give meaning to the observer. Using sketches as a tool for communication will invite discussions and new insights within the design team and to help with decision-making.

4.3.3 Workshop

Conducting a workshop is a way to engage stakeholders and solve problems by enabling progress on a specific subject in a collaborative session (Kaplan, 2019). The purpose of a workshop is to make use of the creativity of others and together explore the problem space, possible solutions, or desirable future scenarios on a given theme. What distinguishes a workshop from a traditional meeting is its characteristic of being an activity including a purpose, scope, preparations, and structure, and they are usually longer than a regular meeting. When a design team needs in-depth input or ideate on, for example, a concept idea, conducting a workshop with relevant stakeholders is more appropriate than a meeting. Kaplan (2019) presents five of the most common types of workshops as being:

- **Discovery workshops** - Communicate the current state and create consensus for milestones and plans.
- **Empathy workshops** - Help a broad team or stakeholders understand and prioritize user needs before designing a solution.
- **Design workshops** - Rapidly generate and discuss a wide set of ideas with a diverse group of attendees.
- **Prioritization workshops** - Build consensus on which features customers (or other stakeholders) value most and prioritize them.
- **Critique workshops** - Ensure that design decisions align to user needs.

However, she does emphasize that there are a lot of other types of workshops and that the workshops presented can be both combined and adapted for the specific needs. The most important part is to set out a purpose and a scope for the workshop and its outcome.

4.4 Prototype

The fourth phase of the Design thinking process is about bringing ideas to life. Prototypes aim to produce early, inexpensive, and scaled-down versions of a product in order to test and identify any problems with the current design (Dam & Siang, 2020b). Making physical and tangible prototypes of a product concept also enables designers to think and talk about the solutions in new ways, instead of just visualizing abstract ideas in their minds. By the end of this phase, designers should have a better understanding of the constraints and how real users will use and behave while interacting with the product and what problems might occur (Dam, 2021). There are different kinds of prototypes to create depending on the design goal and what needs to be communicated or tested. The different prototypes are often categorized by level of detail and functionality, ranging from low- to high fidelity (Dam & Siang, 2020b).

4.4.1 Lo-fi Prototypes

Low-fidelity (lo-fi) prototypes usually take the form of basic models, sketches, and/or rough paper prototypes of digital interfaces. They are quick and inexpensive and leave room for exploration and iterations between designs (Walker et al., 2002). Lo-fi prototypes are ideal in the early stages of testing when details have not been finalized and things are likely to change. Further, lo-fi prototypes do well in the ideation phase where there is freedom to explore and generate new ideas and potential solutions (Dam & Siang, 2020c). Lo-fi prototypes differ from the more high-fidelity prototypes in how detailed they are; interactions, visual appearance, and/or level of detail.

4.4.2 Hi-fi Prototypes

High-fidelity (hi-fi) prototypes are prototypes that have a more elaborate look and operate closer to the finished product compared to a lo-fi prototype. They are used to refine and decide the final design decisions before they go into implementation. Thus, they are often more interactive and allow for in-depth testing with users to test how they will respond to, interact with, and perceive the design (Dam & Siang, 2020b). Despite being more expensive to invest in time-wise than lo-fi prototypes, it is important to note they are faster and less expensive to produce than coding the final product (Dam & Siang, 2020c).

4.4.2.1 Mockups

A mockup is a hi-fi prototype that takes the form of a visual representation of a product concept, created using digital tools like Adobe Xd or Figma. A mockup is a representation of what the finished product may look like; color schemes, visual style, typography, and other visual attributes (Mkrtchyan, 2018).

4.5 Test

The test phase is the final step of the Design thinking cycle. Being part of an iterative process, the insights from the tests are often used to redefine or add to the problem statements (Dam, 2021). The ultimate goal is to get as deep an understanding of the product and its users as possible. In turn, help you understand how people think, behave and feel towards the product, and/or even lead the process to loop back to a previous stage in the design thinking cycle.

4.5.1 Usability Testing

Usability is the extent to which a product can be used by a specified user to achieve a specified goal in a specified context of use (see Usability). It is better to test your concept iteratively throughout the process, independent of the fidelity, rather than test it when it is done (Nielsen, 2012). When it comes to usability testing, there are a lot of different things to test and a lot of different approaches to achieve the goal of the test. According to Nielsen, the most common and useful way of testing usability is by conducting a user test. There are three different components to keep in mind when setting up a user test:

- **Representative users** - Get hold of users that are representative for the intended assignment.
- **Representative tasks** - Make sure the to think through what is the intended goal with the test and try to make a suitable test to reach that goal.
- **Observation** - Observe the users as they perform the test, take notes on where they succeed and where they have difficulties. Do not interrupt the users, but let them do the talking. It is encouraged to ask them to speak out loud as they go along with the test to capture their instant thoughts as well.

The result is then analyzed and the insights can generate a better understanding of how to improve the product and/or the design requirements.

5
Process & Execution

In this chapter, the thesis project process and execution is described in chronological order as iterated through phases of the design thinking framework (see Design Thinking). Each phase has employed a human-centered co-design approach by continuously consulting developers and a variety of other stakeholders throughout the entire design process.

This master's thesis project stemmed from a proposal commissioned by the HFI-team at Saab Surveillance (see Appendix A - Thesis Proposal). The proposal outlined the need for a "base product" for different AEW&C and surface radar systems (not including naval solutions) as a means to streamline the development process and bring consistency to the operation of different tactical applications. Hence, a preliminary research question was formulated as a starting point for the project:

What are important factors to consider when developing coherent tactical GUIs?

After an initial empathize and define iteration on the topic of achieving consistency in applications, several factors pointed towards designing a design system as a means to more effectively address the original proposal. This led to a change of research question and a second iteration of the empathize and define phases, focusing specifically on designing a design system and how it may be designed to minimize the trade-off between coherence and responsiveness, as illustrated in Figure 5.1.

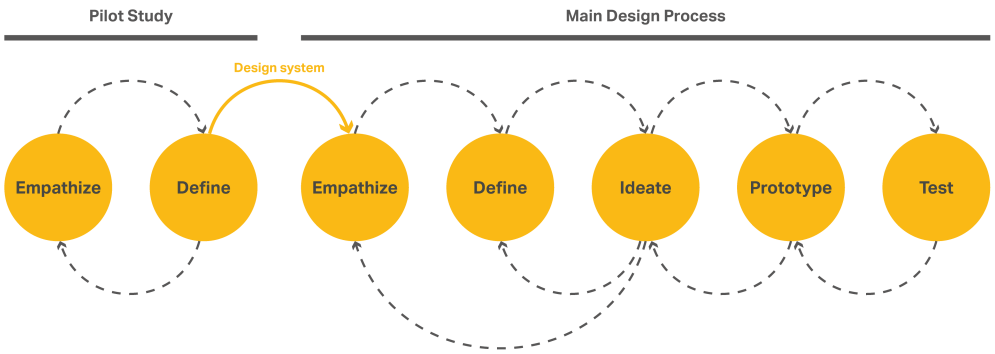


Figure 5.1: Design Thinking process: From Base product to Design system

5.1 Project Planning

This master's thesis was a 30 credits project within the field of interaction design. The thesis was carried out over 20 weeks during the spring of 2022. Throughout this period, supervision meetings with the academic supervisor from Chalmers were arranged once a week to provide feedback and guidance and help the thesis duo to stay on track. Likewise, supervision with the project mentor at Saab was scheduled once a month to ensure all the right support from the company was given in order to proceed with the project.

The first step of executing this thesis was to plan out all the different stages in the process. Since one member of the thesis duo had prior knowledge about the applications, it was important to create a mutual foundation to start from. To do so, initial meetings with various stakeholders were arranged to empathize with the systems, users, and Saab as an organization. Afterward, a plan was made with the goal to execute at least one full design thinking cycle with its five inherent phases; define, empathize, ideate, prototype, and test, with leeway for iterations between and within each step. A simplified version of the project plan can be seen in Figure 5.2 and a more detailed Gantt chart can be found in Appendix B - Gantt Chart.

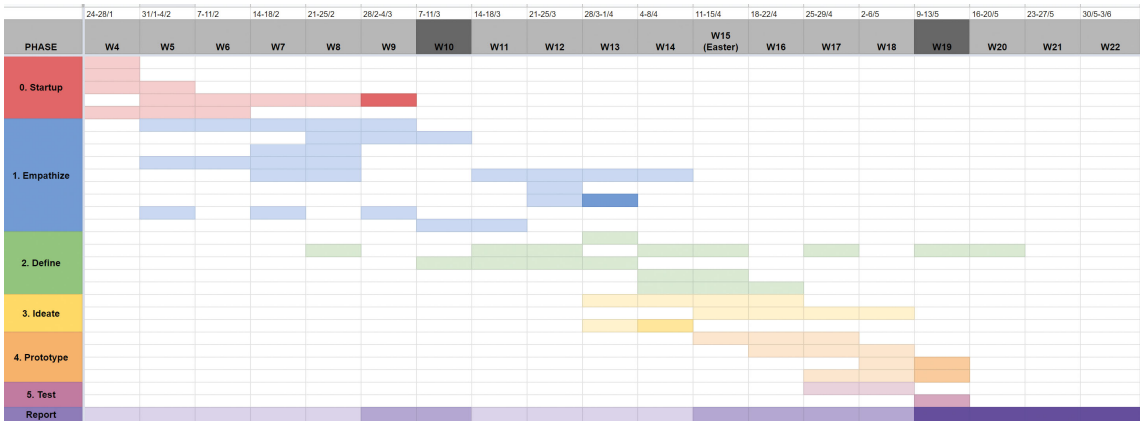


Figure 5.2: The simplified overview of the plan for the project

5.2 Empathize - Coherence between applications

In order to properly approach the problem of visually and functionally divergent radar applications at Saab, the thesis duo identified a need to better understand the theoretical principles of coherence in application design and pinpoint how and where problems currently arise within the design- and development processes. Hence, the first part of the project focused on researching consistency and coherence, familiarizing with the different applications and conducting stakeholder interviews.

5.2.1 Literature Study on Consistency & Coherence

To get a better picture of the current body of knowledge regarding consistency and coherence, an extensive literature study was performed covering best practices and theoretical frameworks for achieving and managing the balance between consistency, coherence, and responsiveness in application design (see Consistency in Design and Coherence in Design).

The literature study provided valuable theoretical insights into the field of consistency and coherence and helped clarify ways to define, measure, and achieve it in application design. Moreover, it provided the duo with a framework for thinking about external and internal consistency as well as the relationship between coherence and responsiveness.

5.2.2 Study of Company Research and Guidelines

Previous efforts have been made to implement a more coherent and user-centered approach to application design at Saab. In order to assess the current progress and avoid unnecessary duplication of work, the thesis duo performed a study of internal documentation produced by the HFI-team and other employees at Saab. The study covered HMI-guidelines for developers, user research on radar operators, and Saab's Brand Portal (brand.saab.com) in order to familiarize with Saab's values, guidelines, and best practices when it comes to design at large.

The HMI-guidelines were found to focus on how and where to use generic components and patterns such as buttons, checkboxes, comboboxes, and combinations of these. The Brand Portal, on the other hand, mainly catered to web-design as it only described general, overarching design principles for Saab as a whole, focusing on keeping digital experiences on-brand. In the digital design section, a limited set of interaction design guidelines were specified, including layout structure, components, colors, and an icon library.

5.2.3 Try it yourself

To get familiarized with the applications, the thesis duo had the chance to observe and try AEW&C applications in a simulator environment. Two experienced employees at Saab, each responsible for different simulators, assisted the duo in setting up scenarios, demonstrating different functionalities, and answering questions while the thesis duo tried the applications themselves. As this was the first interaction with the applications, the most important part was to get a fundamental understanding of the system and how the applications function and communicate with each other. Here, the different interfaces were examined by testing various functionalities and user flows.

From observations and interacting first-hand with the applications, the thesis duo gained an initial, basic understanding of each application's purpose, functionality and how they function together. More importantly, differences in operation, struc-

ture and visual appearance could be identified at an early stage. This helped to clarify the problem and eliminate assumptions.

5.2.4 Stakeholder interviews

Three unstructured interviews were conducted with employees at Saab in order to understand how applications are currently being developed and where in the process incoherence might be introduced. To get a holistic view of the process and potential problems, it was important to gather different perspectives from different departments involved in the design of the applications. Hence, a mix of remote and physical interviews was held with a developer and a project owner working with airborne radar applications, a UX-designer working with the surface radar systems, and a designer from the HFI-team. Each interviewee was asked to describe their domain, working process, and rationale behind some of the design and layout decisions made in the applications. Remote interviewees were able to screen share and offer a thorough walk-through of the applications and their functionalities.

Through the interviews, it was found that the surface- and airborne radar applications have been developed in different ways. The airborne radar applications, part of the AEW&C system, have been developed over many years, in different teams, in different programming languages, and at different geographical locations. This has led to insufficient cross-team communication and a lack of resource sharing, ultimately leading to diversity in the interface design. Furthermore, within each application, developers often reuse pre-specified components from existing code frameworks. This has proven to cause inconsistent use of components and patterns as new features are implemented without consulting designers or looking at HMI-guidelines.

Meanwhile, the surface radar applications have been developed from a newer, shared, base product and most teams, including designers, are working at the same geographical location. Consequently, there is a high level of communication and coordination between projects and disciplines. Furthermore, contrary to the airborne radar applications, the surface radar applications are developed based on a shared code framework, allowing teams to share and utilize resources in a much more efficient and consistent manner.

Both developers and designers expressed a need for better communication and collaboration platforms that could serve to improve the integration of human factors in application design. The member from the HFI-team specifically pointed out a need for a shared base product, similar to surface-based radar systems, serving as a foundation for developers to achieve improved consistency and coherence between different AEW&C applications. However, it was also found that certain differences in layout and function priority could be justified since the applications have, after all, different purposes.

5.2.5 Takeaways

The literature study helped inform the thesis duo about the weaknesses of focusing strictly on consistency and the importance of allowing for custom, flexible designs that still feel part of the same ecosystem. This led the duo to aim for coherence and strategies for how to achieve and balance coherence with responsiveness. Notably, strategies for managing this balance focused on designing quality platforms and tools that facilitate the collaboration between designers and developers. One such tool, promising to minimize the trade-off between coherence and responsiveness, was design systems.

Based on observations and interviews, the Brand Portal and HMI-guidelines were deemed incomplete in terms of included components, patterns, and guidelines. Currently, there is an absence of specified visual and functional attributes for each component and a lack of associated code and guidelines. This, in turn, makes them difficult for designers and developers to translate into practice and apply within the development process of complex applications. Furthermore, having the guidelines in a static document does not provide sufficient agility when it comes to keeping all the documentation and its dependencies up to date.

From the stakeholder interviews combined with hands-on experience with the applications, it became evident that the AEW&C applications featured a higher degree of functional and visual incoherence in comparison to the surface-based applications. Moreover, the developers and designers have limited or no direct contact with active operators of the applications. HMI-guidelines, hence, are seldom based on user requirements but rather on generic design principles.

Furthermore, the current practice of constantly complying with customer requirements often causes a direct tension between coherence (adhering to Saab's design language) and responsiveness (customizing according to customer's wishes). Finally, all interviewed developers and designers expressed a need for better communication- and resource-sharing platforms to improve collaboration between teams and disciplines.

5.3 Define - Coherence between applications

In order to make sense of the collected data and define the problem, the takeaways were discussed between the thesis duo and the academic supervisor. The goal of this phase was to synthesize the insights and formulate an actionable problem statement that could serve to determine the scope of the project going forward.

5.3.1 Problem Statement

Based on the insights gathered through interviews, observations, and literature studies, the thesis duo brainstormed possible directions for the project going forward.

Firstly, having established that the surface radar solutions (not including naval solutions) all share the same base product, and thus already are rather coherent, it was decided to narrow the scope to focus only on the AEW&C system. Secondly, as the HMI-guidelines and Brand Portal were found to be insufficient in their current form, the thesis duo reasoned that developers need a more straightforward and dynamic approach to application design. Further, by abstracting the HFI-team's suggestion of a base product into a more open solution space, the intrinsic need was also found to be centered around helping developers achieve coherence through coordinating and systemizing design decisions. Enabling developers to design more coherent applications would also, by extension, result in more coherent interfaces for AEW&C operators as well. Hence, going forward, developers were considered primary users, designers secondary users and operators side users, as they are indirectly influenced by the solution.

Another consideration was the importance for projects to accommodate customer requirements. As both designers and developers strive to cater to customer requirements, the solution would also need to be responsive enough to support new and unforeseen demands in the future. Customers were thus also considered an important stakeholder going forward. Hence, the stakeholder needs were defined as follows:

- **Developers** - Need a systematic, coordinated approach to application development
- **Designers** - Need more granular control and influence over design decisions and how they are implemented.
- **Users** - Need a more coherent system of applications that supports their knowledge.
- **Customers** - Need customized applications that are tailored to their specific needs.

Mapping out stakeholder needs, it became evident that the main challenge for developers was the tension in constantly having to balance coherence demands with responsiveness demands. Hence, the problem statement was formulated as follows:

Developers need a systematic, coordinated and responsive approach to application design in order to achieve coherence between different AEW&C applications.

Comparing the problem with the findings from the literature study, the thesis duo arrived at the conclusion that a design system could serve to address Saab's needs. Hence, the project pivoted towards researching and designing a design system that could serve to balance coherence and responsiveness in a systematic and coordinated manner.

5.3.2 Research Scope

In order to reflect and support the new focus of the project, the research scope was revised and formulated as follows:

What are important requirements to consider when designing a design system for coherent tactical GUIs?

Aiding in this research, two sub-questions were considered important for identifying relevant requirements:

1. What are important factors to enable developers to use and contribute to a design system for coherent tactical GUIs?
2. What are important factors to help minimize the trade-off between coherence and responsiveness in design systems?

5.4 Empathize - Design System

In order to elicit requirements to consider when designing design systems that can support coherent development in the context of tactical applications, a second iteration of the empathize phase was initiated. The thesis duo performed extensive literature studies on how to build and structure design systems while also studying a variety of existing solutions. This was combined with ethnographic research at Saab, focusing on understanding developer needs and requirements. Due to the complexity and critical nature of the AEW&C applications, a contextual inquiry with operators was also an important part of understanding the specific characteristics of the current interfaces. Finally, in order to align the design system with Saab as an organization, interviews were carried out with other stakeholders, including designers and executive staff.

5.4.1 Literature Study on Design Systems

Guided by the new research question, the aim of the second iteration of literature studies was to establish an understanding for the current body of knowledge on design systems in general as well as factors for enabling usage, contribution, and minimizing the trade-off between coherence and responsiveness. More specifically, the questions guiding the research were:

- What are some of the current definitions of design systems and their parts?
- What are some benefits and challenges of developing design systems within (complex) organizations?
- What are some good practices for building and structuring a design system with a focus on the balance between coherence and responsiveness?

The study covered academic articles, websites, and three books, including *Atomic Design* (Frost, 2016), *Design System Handbook* (Suarez et al., n.d.) and *Building*

Design Systems (Vesselov & Davis, 2019).

Altogether, the study provided a deep and nuanced understanding of how design systems may be built, hosted and structured to support developers and designers in achieving efficient workflows and coherent designs (see Design System). The different ways of describing the constituent parts of design systems were analyzed and compiled into a model summarizing the findings, as illustrated in Figure 5.3.

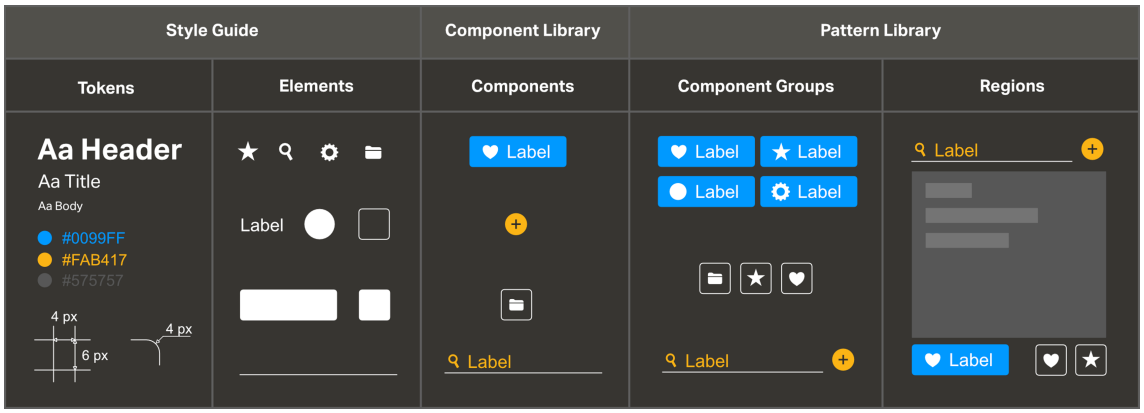


Figure 5.3: The thesis duo's model summarizing the constituent parts of design systems

5.4.2 Benchmarking

In order to understand what constitutes different design systems and identify best practices regarding content, organization and structure, numerous extensive and simple design systems were studied. Five design systems were selected for deeper analysis in the benchmarking. These design systems were Adobe's Spectrum (spectrum.adobe.com), IBM's Carbon (carbondesignsystem.com), Google's Material Design (material.io), Microsoft's Fluent (microsoft.com/design/fluent), and Shopify's Polaris (polaris.shopify.com). These design systems were chosen for two reasons. Firstly, they have open source repositories that can be viewed and used by people outside of the company. Secondly, they have different structures and target different types of markets and users. The latter was preferable to get a broader understanding of how design systems are designed and structured depending on the company and their product platform.

From the benchmarking, it was found that most major design systems primarily cater to web development. Only a few design systems, such as Material Design and Spectrum, also cater to simple application development. Hence, the design systems generally include components and patterns found within the web domain. For the same reason, code specifying visual attributes is rarely provided beyond the realm of the most popular web frameworks such as Angular, React, and Vue. The exception is Material Design which does specify code implementation for Android and iOS as well. Nevertheless, none of the benchmarked design systems appear to cater well to complex application development.

Regarding the repositories, it was found that the examined DSRs shared a very similar content structure. Most DSRs contain design guidelines, a component library, branding, and resources. Common for all component pages is that they include component name, description, example images, formatting guidelines, and behaviors, see Appendix C - Benchmark Design Systems for full inventory. Meanwhile, Carbon and Material Design are the only ones offering component-specific tokens and Polaris is the only one offering the ability to configure components in code. Finally, analyzing the structure of component pages, the included content commonly adheres to the following structure:

- Table of contents
- Component name
- Component description
- Component image
- Component guidelines
  - When and how to use
  - Dos and Don'ts
- Component variants
- Component behavior
- Associated code

5.4.3 Developer Interviews

When narrowing the scope from researching coherent GUIs to designing a design system, the primary user shifted from the AEW&C operators to the AEW&C developers at Saab. Building on the insights from the first iteration of interviews, a script for semi-structured interviews with developers and their work domain was put together and can be seen in Appendix D - Interview Script. Here, the focus was centered around getting to know how developers work on a daily basis; their work processes, communication platforms, collaboration tools, and whether they identify any potential improvements within these areas to facilitate the process of developing coherent applications. The interviews covered everything from how tasks are assigned, to how the designer-developer hand-off works and how design specifications are implemented in code. Finally, the thesis duo introduced the concept of design systems, and the interviewees were asked to give their initial thoughts on whether it would facilitate their work.

Five semi-structured interviews with a total of six developers were conducted. Since all AEW&C applications are developed in different locations four out of five interviews took place remotely. The sample group consisted of more and less experienced full-stack and front-end developers working strictly on one of the applications in question. One of the participants currently worked as a UX-designer but had a background as a developer.

5.4.3.1 Work Process

Through the interviews, it was found that the applications have been developed in parallel to each other, with no real cross-application communication. Furthermore, apart from incoherent GUIs, the applications are also written in different programming languages using different styling approaches. For example, A2 is developed in C++ and styled in Qt whereas A1 and A3 are built in Java, yet utilize different Look & Feels (see Figure 5.4) and are developed on different operating systems. As a result, it is often impossible to share code between teams and applications. Furthermore, all applications have a base product, each consisting of a framework with the most basic version of the application, including functions, visual styling, and components. The base product is then modified into a customer specific project, with a dedicated team of developers implementing adjustments on behalf of the customer. However, currently, there are inadequate platforms, documentation, and tools that guide how new features and adjustments should be designed and implemented in order to maintain coherence across versions of the same application and AEW&C applications in general.

"There are not really any guidelines right now, that's the problem right now" - A2 Developer

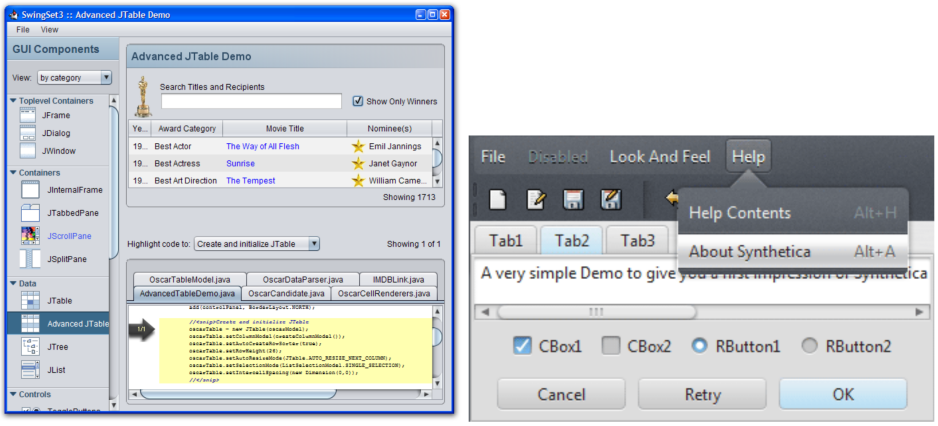


Figure 5.4: Java's Look & Feel Nimbus (left) and Synthetica (right) used in A1 and A3 respectively

The developers' tasks can be related to either the application's base product (e.g. fixing bugs) or a specific customer project (e.g. make changes according to customer requirements). Working in a project will naturally also involve implementing project-specific customer requirements and this demands flexibility in development.

"9 times out of 10, the functions should work in a certain way, but that 1 time out of 10 when it should work in a different way. Then you have to be able to accomplish what you want, even in that exceptional case" - A1 Developer

5.4.3.2 Communication & Collaboration

Until recently, there has been little, or no communication, between teams in different projects or working with different applications. Oftentimes developers find themselves lacking proper design documentation to guide design decisions when assigned with a task of implementing a new feature. Combined with the fact that teams have lacked a dedicated designer, this has led to many decisions being made by individual developers. As a result, the systems have grown further apart and become more incoherent. There is an explicit need from developers to be able to know where to turn when struggling, to get help from others, and/or to look up guidelines for development. But there is also a willingness to share knowledge and resources across teams.

> *"We have put in thousands of hours in our projects. It would be great if we could share our learnings and knowledge with others at Saab as well"* - A1 Developer

The problem is that there is currently no central platform for coordinating decisions and sharing insights across all AEW&C applications.

5.4.3.3 Design System

After explaining to the developers the concept of a design system and asking them about their thoughts regarding implementing a design system in their working progress, the overall reaction was welcoming. Even though they saw difficulties in the implementation in terms of budget and differences in work processes, they were positive about the concept. However, they stressed the importance that the decision of using a design system must be enforced at an executive level; it must be mandatory to use the design system as part of the working process, or else it would not work and they would end up back where they are today.

When discussing the content of the design system and its structure, the developers emphasized that the design system must be accessible and easy to use, in order to avoid annoyance that could lead to abandoning the platform. Further, if the design system should include code, it must cater to both Java and C++/Qt and be of a basic character to allow for flexibility when working with specific customer requirements and customizations.

> *"We have put in thousands of hours in our projects. It would be great if we could share our learnings and knowledge with others at Saab as well"* - A1 Developer

> *"What is important is that it must be mandatory to use it [the design system], because unless everyone uses it, I do not see the benefit of it. It must come from executive level"* - A1 UX Designer

> *"If this was achieved [implementing a design system], it would make all*

the applications uniform and.. Yeah, it would have been just great"
- A1 Developer

5.4.4 Contextual Inquiry with Operators

In order to understand how the applications are used in practice by operators, a contextual inquiry was performed in a simulator designed for A1 training. A scenario was selected and simulated by two operators, each with over 20 years of total experience with the A1 application as former operators and current instructors. The operators were observed from over-the-shoulder by the thesis duo and two assisting members of the HFI-team. The operators were encouraged to speak aloud and explain their actions and the rationale behind their preferred window layout while the thesis duo inquired about their behaviors. Two video cameras were set up to record the operator's screens and audio. Following a 15 minute break, an in-depth semi-structured interview was conducted where the operators were probed about their experience with the work context, the operator role, and how well the GUI supports their workflow (see Appendix E - Contextual Inquiry Script). Altogether, the observation and interview lasted for 2 hours and 30 minutes. Detailed notes were taken throughout and later complemented by reviewing the footage from the video recordings.

Figure 5.5: The thesis duo in the simulator lab with the two operators

From observing the operators, it was found that the primary window containing the map was the main area of interaction. Both operators strived to keep this area uncluttered by actively resizing and moving secondary windows to the left and right of the center and only keeping the most critical secondary windows open. Both operators also utilized a "tabs-only" viewmode for secondary windows to save additional space. However, it was observed that the operators occasionally still needed to scroll the map or move secondary windows in order to see what was underneath. One of

the operators configured and utilized Quick Filters to specify and toggle visible map data and overlays. A secondary map window was used with different zoom levels and filters applied. This was discussed as an important feature for surveilling multiple areas at once, decluttering the main map from data, and to increase situational awareness around the aircraft. One of the operators stated that new operators often struggle with keeping the GUI uncluttered and that some users prefer to utilize multiple virtual desktops for different secondary maps with different filters.

From the in-depth interview, it was found that A2 operators follow a completely parallel training program and do not require previous experience with A1 or A3 applications. With the exception of one of Saab's customers, the same individual is rarely trained to operate all three applications. Regarding the context, it was found that lighting, noise, and vibrations from the aircraft do not interfere with the workflow or perceivability of the GUI. The passenger windows are covered at all times and the artificial lighting is kept dimmed and consistent. Operators wear noise-canceling headsets and the seats are perceived as comfortable. The operators rely heavily on radio communication between each other, the pilots, and ground-based application operators. The work is performed according to mission protocol but can be changed in-flight due to different circumstances. Stress is not a prevalent part of the role as long as everything functions correctly but operators need to be able to handle errors at times. The main stressor of the work is the amount of data on the screen, but users do not necessarily need to adapt the secondary window layout or use of features depending on the amount of data or intensity of the situation.

The operators clearly expressed that they enjoy the current GUI in terms of features and ease of use. The fact that the three applications differ in terms of Look & Feel, and method of operation is bothersome but does not affect the workflow negatively according to the operators. Finally, when asked about preference they did not have any preferred GUI but stated that Saab should choose one and use it consistently.

5.4.5 Stakeholder Interviews

As the current Brand Portal attempts to provide some guidelines for digital applications, the thesis duo saw value in contacting Saab's brand design manager, who is currently responsible for it. After conducting an unstructured remote interview, it became clear that the vision had long been to create a design system and implement a way of making all the digital products reflect the Saab brand. The brand design manager explained that any user should be able to recognize that they are using a Saab product and expressed that a design system would be an important step in the right direction. The only apparent reason why the Brand Portal had not yet been developed into a complete design system was due to low explicit demand, which in turn had led to low priority.

Moreover, continuous formal and informal meetings with the HFI-team, including the project supervisor, were held to ensure that the project stayed on track.

5.4.6 Takeaways

The literature study on design systems helped the thesis duo acquire the knowledge and tools needed to approach building a design system based on user needs and best practices. The thesis duo came to understand a design system as an interface between different disciplines of an organization, see Figure 5.6. Hence, like any interface, a design system can be designed to provide a more efficient and satisfactory developer experience by acting as a platform for facilitating collaboration, communication, and resource sharing.

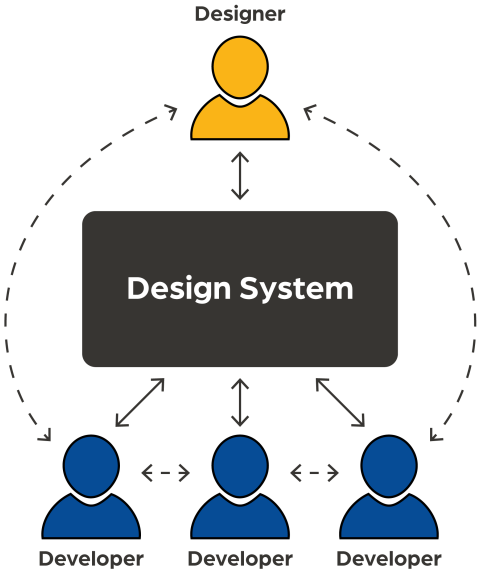


Figure 5.6: A design system as an interface between disciplines

The theoretical knowledge was complemented by benchmarking established design systems as a way to draw inspiration and understand how it may manifest within different organizations. Furthermore, provided that most contemporary design systems are not designed with complex application development in mind, the thesis duo realized that they would need to design a proprietary design system specifically for Saab and tactical GUIs. Importantly, greater emphasis would have to be placed on allowing for customizing and configuring components to specific use cases and applications in order to maintain the balance between coherence and responsiveness.

Combining the findings from the ethnographic research on developers, operators, and other stakeholders, several key findings were concluded. The current incoherence can be traced back to differences in programming languages, limited collaboration, and lack of clear guidelines supporting application design. There is an explicit appeal from several stakeholders for improved infrastructure, and a number of associated factors for such a platform to function within Saab as a company. Among the most important factors were the importance of enabling responsiveness to customer requirements and mandating the use of the platform across the company.

5.5 Define - Design System

Through the activities of the define phase, a preliminary list of important requirements for designing a design system for Saab's needs was conceived. Following best practices found in the literature, an interface audit and heuristic evaluation were conducted to systematically map and assess current solutions in terms of coherence. This ultimately led to clarifying and determining the scope of the project and subsequent ideation.

5.5.1 Affinity Diagram

In order to consolidate insights and identify needs from the interviews, an affinity diagram analysis was carried out. Quotes, paraphrases, and notes from the interviews were analyzed and iteratively clustered and reclustered by similar, overarching themes. The analysis resulted in a digital diagram containing eight themes; process & collaboration, resource sharing, technically possible, bottlenecks, users, current HMI, explicit needs, and design system. The cells of each theme were then analyzed for needs and translated into explicit requirements organized by different categories. Here, categories were based on the original sub-questions aiming to help answer the research question, including usage factors, contribution factors, and factors for minimizing the trade-off between coherence and responsiveness. In addition, a fourth category was identified: organizational factors, where specific requirements related to Saab as an organization were listed.

5.5.2 Interface Audit

Having familiarized with the applications through various observations and first-hand experiences, an interface audit was performed in order to systematically gather and categorize the different components in the AEW&C applications. Based on the results from the benchmarking, the different component categories identified in other design systems were used as a basis for the audit. The interface audit started off by screenshotting all the different components in the applications and pasting them into an Adobe Illustrator document, structured by application and component category. Here, the components were saved not only in their original form, but also in different interactive states, such as hover states, selection states as well as active and inactive states. If there were any components that did not fit any of the categories, these were set aside and categorized into new categories.

The interface audit generated a comprehensive, organized library offering an overview of the components (see Figure 5.7) and patterns (see Figure 5.8) currently employed in the different tactical GUIs. The resulting document paved the way for the following Heuristic Evaluation (see Heuristic Evaluation) and constituted the basis for the Remote Workshop (see Remote Workshop). Furthermore, current occurrences of tokens and components in terms of styling, sizing, and semantic colors could be used to inspire ideation and prototyping in the subsequent phases of the project.

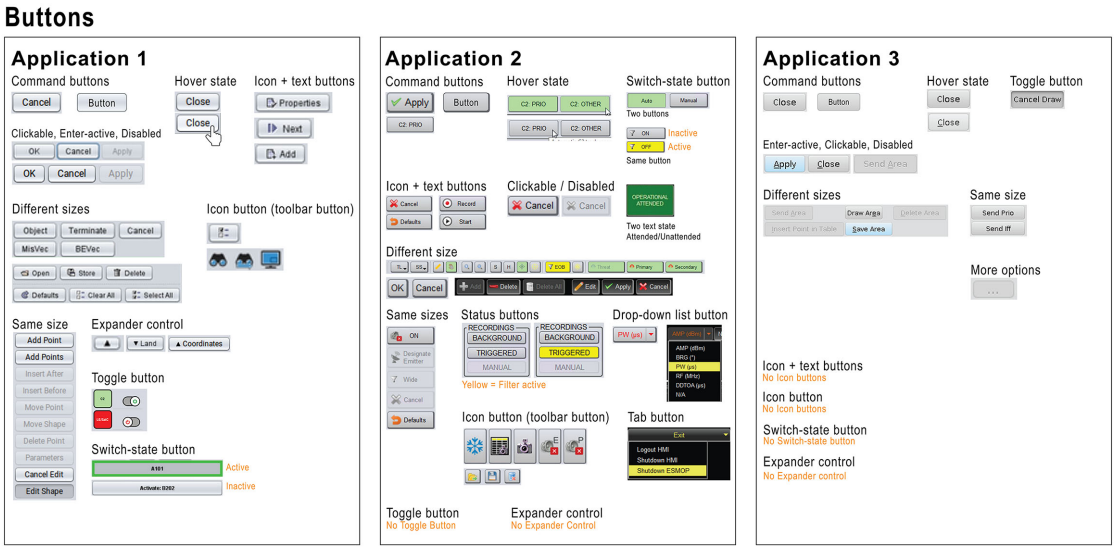


Figure 5.7: Button components found in the interface audit (all components can be found in Appendix F - Interface Audit)

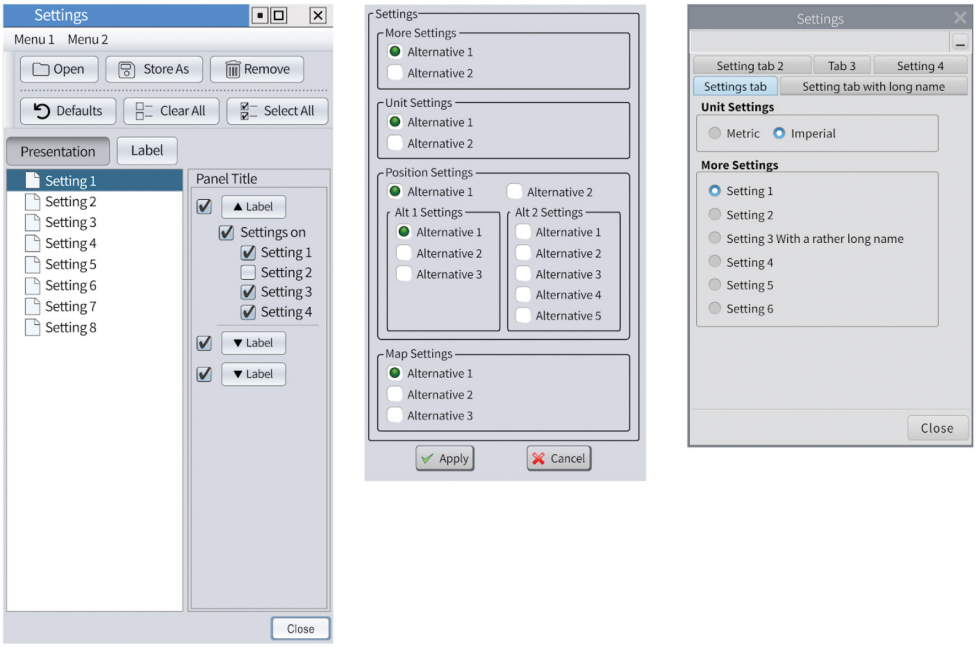


Figure 5.8: From left to right: Mockups of A1, A2, A3 settings dialogue patterns found in the different applications

5.5.3 Heuristic Evaluation

In order to understand and define current differences between A1, A2, and A3, a heuristic evaluation was performed on the categorized components from the interface audit. The heuristics focused on identifying and classifying internal and external

inconsistencies based on principles derived from the literature study on consistency and coherence (see Consistency in Design and Coherence in Design). Four main heuristics were selected for the evaluation, two of which had sub-heuristics, resulting in a total of six heuristics:

1. Are components internally consistent within each AEW&C application?
   (a) Do similar sets of stimuli have the same usage, behavior, or result?
   (b) Are similar usages, behaviors, and results represented with similar stimuli?
2. Are components externally consistent between Saab's AEW&C applications?
   (a) Do similar sets of stimuli have the same usage, behavior, or result?
   (b) Are similar usages, behaviors, and results represented with similar stimuli?
3. Do components have external analogic or metaphoric correspondence in the real world beyond the computer domain?
4. Are tokens & elements used consistently to form components?

Heuristics 1a, 1b, 2a and 2b aimed to help the thesis duo find and analyze the interfaces with regard to irregularities and contradictions (see Consistency in Design) whereas heuristic 3 aimed to evaluate whether the applications employed appropriate use of attributes to communicate different functions, usages, states, and results. Finally, the purpose of heuristic 4 was to identify whether there was any current system of building components based on tokens and elements.

The thesis duo systematically analyzed categories of components, including Buttons, Selection Controls, Navigations, User Inputs, Dialogues, System Statuses, and Tables. Consistency with regards to the use of colors, shadows, icons, sizing, layout, and more were documented in text with screenshots of the components. In order to achieve a more nuanced and exhaustive evaluation, the thesis duo performed the heuristic evaluation individually before comparing and summarizing the results.

The heuristic evaluation resulted in a large set of documented consistencies and inconsistencies, both internal and external between different applications. A summary of the key findings is presented in the following sections.

5.5.3.1 Internal Consistency

The internal consistency across the interfaces was found to vary by application. In general, A3 had the most internally consistent use of components. A1 and A2 were found to have both irregularities and contradictions. For example, A1 was found to utilize three different patterns for checkbox nesting, as illustrated in Figure 5.9. Furthermore, expander buttons appear similar to action buttons, causing a contradiction between stimuli and usage. Both A1 and A2 were found to have irregular use of iconography, both in terms of where icons are used and the styling of the icons themselves.

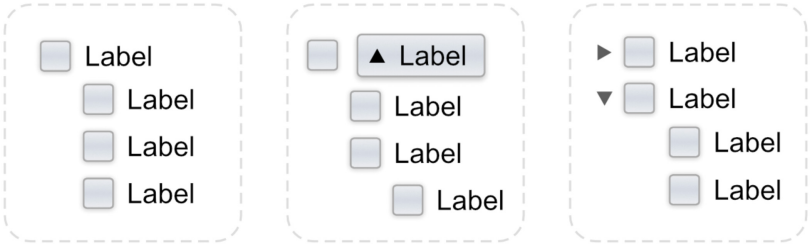


Figure 5.9: Examples of internal irregularity in different checkbox patterns found in A1

5.5.3.2 External Consistency

Some components do not exist across all of the interfaces, for example, toggle buttons, switch-state buttons, and icon buttons. A1, A2, and A3 all have externally irregular styling for components, including the use of colors, gradients, and drop shadows, see Figure 5.10. For example, A2 uses a heavy, 45-degree angled drop shadow while A1 has a more shadow directly underneath components. A3 on the other hand has no drop shadow, but uses inner shadows and gradients to convey a sense of depth. Furthermore, A2 utilizes more extensive use of icons on buttons in comparison to A1 and A3. Sometimes, inconsistencies were found to be justified.

5.5.3.3 Analogic & Metaphoric Correspondence in the Real World

Buttons are analogic to the real world, using drop shadows and gradients to indicate an elevated position in the interfaces. Once pressed, they retract into the interface, mimicking a physical button. However, contradiction occurs in places where status indicators are made to look exactly like buttons, when in fact they are not, as in Figure 5.10.

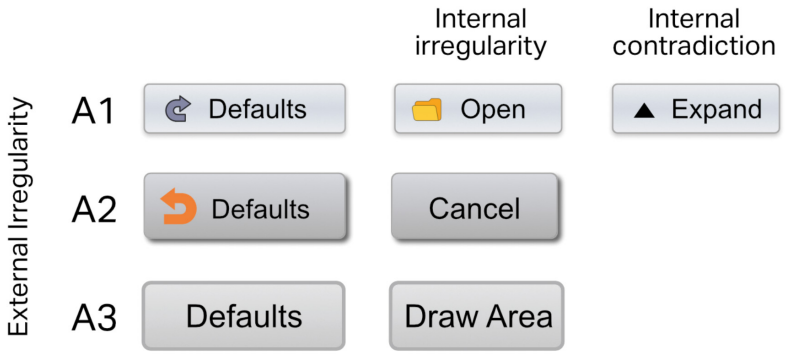


Figure 5.10: Examples of external and internal irregularity and contradiction found in buttons of the different applications

5.5.3.4 Use of Tokens & Elements to Form Components

Taken together, A1, A2, and A3 all use different color palettes causing an external irregularity. A1 and A3 have a more restricted color palette in comparison to A2, which relies on a number of colors to convey meaning and bring clarity to different components. Labels and icons, however, are generally used consistently across all applications to form components and padding is used consistently for the most part. From an internal perspective, however, A3 was found to utilize irregular padding on buttons, see Figure 5.10.

Altogether, the results from the heuristic analysis displayed a lack of consistency and coherence within and between the different applications. Underlying all of the inconsistencies was the fact that tokens and elements were not found to adhere to a system, meaning that components were rather based on individual intuition and taste. This is a problem from many perspectives. From a developer's point of view, components and patterns can not be shared and reused across applications. From a designer's perspective, the current differences in patterns complicate UX-flows. From the angle of Saab as a brand, the visual styling does not adhere to Saab's branding as a whole. Finally, from an operator's point of view, the internal and external inconsistencies may lead to confusion and errors in cases where stimuli are not clear or have unexpected effects.

5.5.4 Project Scope

Based on the insights and requirements derived from the second iteration of empathizing with and defining the problem, it was confirmed that a design system would be a welcome, needed, and adequate approach to support building coherent tactical GUIs.

As design systems are incredibly complex and demand a lot of resources, the thesis duo decided to limit the scope of the project. Following discussions with the HFI-team and academic supervisor, three main deliverables were agreed upon:

- A foundation for a design system consisting of tokens, elements, components, and patterns.
- Requirements/Guidelines for how to use and contribute to the design system.
- A simple reference product in the form of a digital mockup and/or semi-functional prototype designed to embody the design system.

Consequently, the project actively excluded research and design of the DSR; including the structure and organization of the system, extension of HMI-guidelines, as well as any UI for contributing and managing the design system. It was also decided to exclude any code-related parts of the design system. Finally, the research would not focus on any specific user flows for any of the AEW&C applications.

5.6 Ideate

Having defined the problem and the scope of the project, the ideation phase began exploring possible ideas and solutions for how a design system may be designed to support coherent application development. Here, future users of the design system were engaged through participatory design in the form of a remote workshop. Preliminary ideas were then elaborated and refined by the thesis duo in preparation for prototyping the design system.

5.6.1 Remote Workshop

In order to establish a common understanding and foundation for a design system for AEW&C applications, all interviewed developers were invited to partake in a remote workshop. The purpose of the workshop was to initiate a shared conversation and gain a better understanding of what AEW&C developers need in a design system for it to be useful and implementable in their workflows. The goal of the workshop was to achieve a shared terminology, brainstorm the content of the component library, and how coherence and responsiveness may be balanced to accommodate customer requirements and utilization in different parts of different applications. More specifically, the goals were to:

- Generate a common list of component names.
- Brainstorm what developers need to be able to use and contribute to a design system.
- Brainstorm what components should and should not be included in a design system.
- Brainstorm when, how and why components need to be responsive and how this can be achieved in a design system.

The workshop was generally built around "How Might We"-questions such as *How might we enable addition, modification, and development of the design system continuously?* and *How might we handle special configurations of components systematically?*

In order to sensitize participants to the basic concepts of design systems before the workshop, an invitation email was sent with an encouragement to familiarize themselves with the Carbon Design System (carbondesignsystem.com). Attached in the email was also a document listing a selection of similar basic components from the different AEW&C applications, where each participant was asked to provide their name for each component ahead of the workshop. The replies were aggregated by the thesis duo and used as a foundation for discussion during the workshop.

The workshop took place remotely over Skype and lasted for a total of two hours including a 15-minute break. Out of five invited developers, two canceled due to work commitments which left the workshop with three participants; two frontend developers and one UX-designer with a developer background, each representing

A1, A2, and A3 respectively. The workshop was initiated with a brief on design systems and a short warm-up exercise since the participants did not know each other beforehand. This was followed by three sections for discussing the different topics: Foundational Structure, Component Library, and Coherence & Flexibility (see Appendix G - Workshop Structure). The thesis duo facilitated and participated actively in the workshop while also taking detailed notes. The Skype meeting was recorded and transcribed with verbal consent from all participants. Following the workshop, notes were summarized and articulated into requirements as a basis for further ideation.

The workshop concluded that all of the components included in the Component Names Document were important, but some could be simplified and merged into more generic components that allow them to be employed more universally. The same document also highlighted the differences in how teams refer to components. Some names were completely different while some were similar. The developers pointed out that the naming often originated from the class- or library of the specific programming language, and agreed that a design system should employ a generic terminology when referencing components in the component library. To clarify the relationship between the component name and the programming language, it was suggested that each component page could reference the corresponding class- or library name in addition to the generic name to facilitate identification and discoverability across applications.

Regarding the rest of the content on the component page, the participants expressed that they liked a lot of the information provided in Carbon: a generic component name, an image of the component, variants, and a description with guidelines explaining how and where to properly utilize it. The participants added that the general information should be application-independent but that code needs to be specified for each programming language. The code should specify a core-component that can be modified to different needs depending on the use case and application.

When it comes to patterns, the developers pointed out that the three applications are too dissimilar for any elaborate, or specific component groups or regions to be included in a design system covering all three. If anything, it would only be useful to specify basic, recurring component groups like tabs and context menus, and regions such as dialogues, menu bars, and toolbars.

The participants noted that it is possible to style components to look graphically identical regardless of the differences in programming language and that this is more time-efficient than switching the entire application to be built in another language. However, it would still be a very time-consuming task to build and maintain. Programming a Saab Look & Feel in Java Swing would, for instance, take an experienced developer about 6 months according to one participant.

The developers expressed that frequent meetings, a dedicated design system team, and rapid review processes are important factors for enabling efficient contribution

and maintenance of the design system. From their experience, new components are rarely added but once they are, it is important that components are made available to everyone quickly. Otherwise, different developers will invent their own, proprietary components for the same use case, causing fragmented code and GUIs. Nonetheless, small changes to components could be updated automatically by anyone; "similarly to Wikipedia" or "through a form", while larger modifications should necessitate a collective review process before being granted.

Finally, as Saab deals with a lot of confidential data, all code that constitutes the applications, including code for components, can only be shared via protected channels. Hence, one developer suggested that the design system and associated code could be hosted on one of Saab's closed platforms accessible only by authorized personnel.

5.6.2 Sketching

Throughout different parts of the project, sketches facilitated communication between the members of the thesis duo whilst discussing approaches for building the design system. By visualizing ideas on paper and whiteboard, they became easier to appreciate and compare. In the ideation phase, sketches were mainly used as a tool for discussing the following points:

- How might we architect a design system for tactical GUIs?
- How might we define and present tokens and components?
- How might we define and present states and variants?
- How might we prototype a component page for testing?

The result from the sketch sessions can be seen in Figure 5.11. The rough sketches produced in the sketching sessions could then be used as a foundation for building out a prototype.

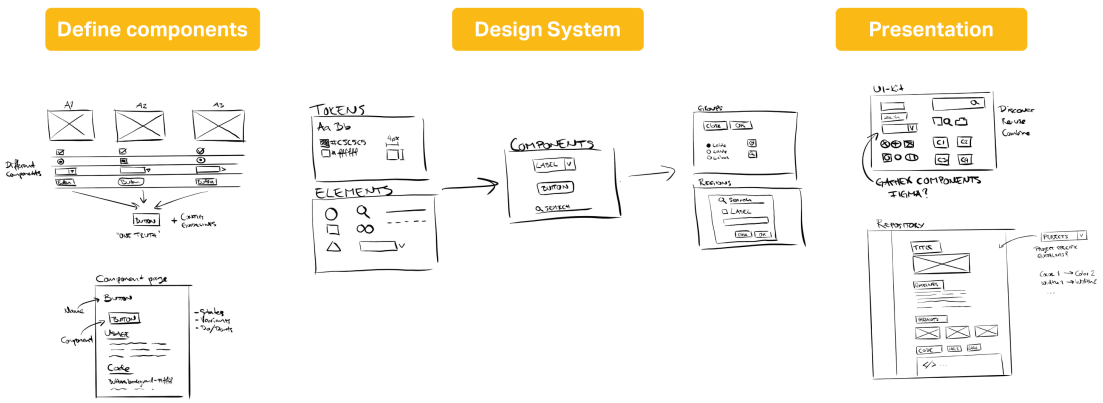


Figure 5.11: Sketches of what the design system should include

5.6.3 Takeaways

The workshop with developers offered a few new insights and perspectives on how design systems may be designed. New needs were analyzed and translated into additional requirements, some of which related to the design of the DSR. Although the project had already chosen to exclude the research and design of a DSR and the structure of component pages, insights regarding these were also compiled into requirements as a basis for future work. Additionally, from the concrete ideas regarding the design system generated during the workshop and sketching, the thesis duo was able to define a common terminology for components (see Appendix H - Naming Components) that would act as a basis for the design system going into the prototyping phase.

5.7 Prototype

Following the results of the ideation phase, the project started to transition into the prototyping phase. Here, the project continued to iterate back and forth between ideation and prototypes in order to quickly try out different ideas. This process consisted of designing a range of Figma (www.figma.com) prototypes of the design system in various fidelities; from tokens and elements to components and mockups. Using Figma as the main prototyping tool (see Figure 5.13) allowed the thesis duo to work collectively in real-time and utilize native support for components and variants. This way, all levels of the design system could be prototyped concurrently, adhering to the atomic mental model used by Frost (2016), in order to continuously ensure that tokens could be combined into components, which in turn could form coherent interfaces (see Figure 5.12).

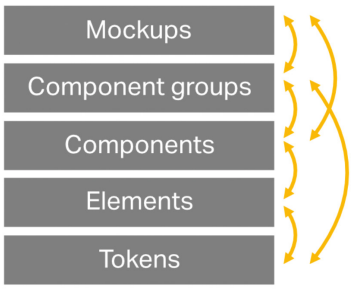


Figure 5.12: Iteration between levels of the design system prototype

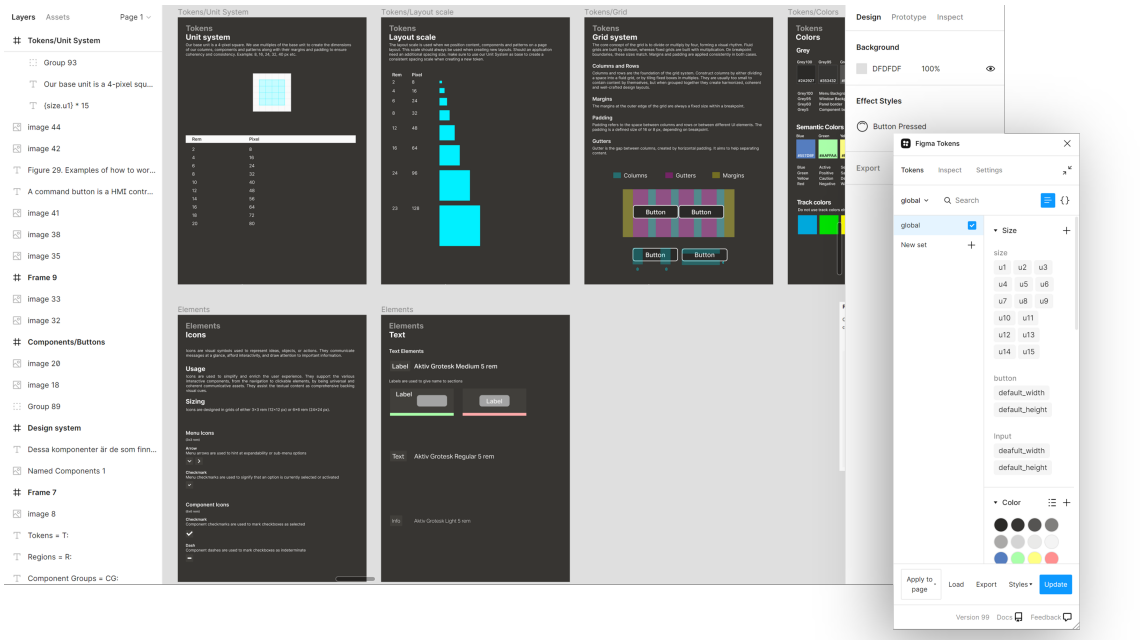


Figure 5.13: Figma Prototyping workspace and Figma Tokens plugin

The thesis duo based the ideas and prototypes on a number of insights derived from literature studies, Benchmarking design systems, the Interface Audit, and Heuristic Analysis combined with requirements identified through the contextual inquiry, interviews, and the remote workshop.

5.7.1 Tokens

As tokens are the foundation of a design system (see Architecture) they constitute the abstraction of everything impacting the visual design of an application, including fundamental attributes such as colors, typography, and spacing. The Figma-plugin *Figma Tokens* by Wahid et al. (n.d.) was used to define all the fundamental design tokens, as can be seen i Figure 5.13. This tool greatly facilitated working with a lot of dependencies across different levels of the design system.

Starting off at pixel-level, the first thing to decide was the unit system. The base unit stands as the foundation for how geometries and layouts are formed, how to scale them, and how they relate to each other. Here, the duo chose to adhere to the literature, suggesting a base unit of 4 px (see Figure 5.14) due to its appropriateness to scaling (see Designing a Design System).

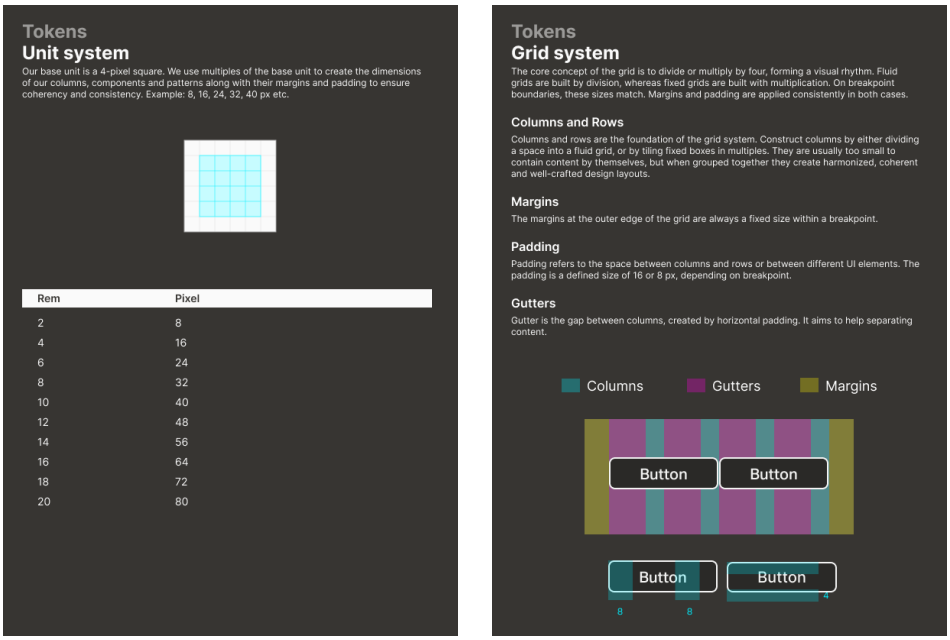


Figure 5.14: Unit system (left) and Grid system (right)

The color scheme for the tactical applications must be rather subdued and cannot contain a lot of colors and gradients that can interfere with what is occurring on the map. Since it was found in the Contextual inquiry (see Contextual Inquiry with Operators) that the operators work in dimmed lights for a long period of time, it was decided to make a darker color scheme for the foundation of the GUI. The colors picked for the design system were based on the colors from Saab's Brand portal, but additional colors were added to allow for a greater degree of freedom in the development, see Figure 5.15.

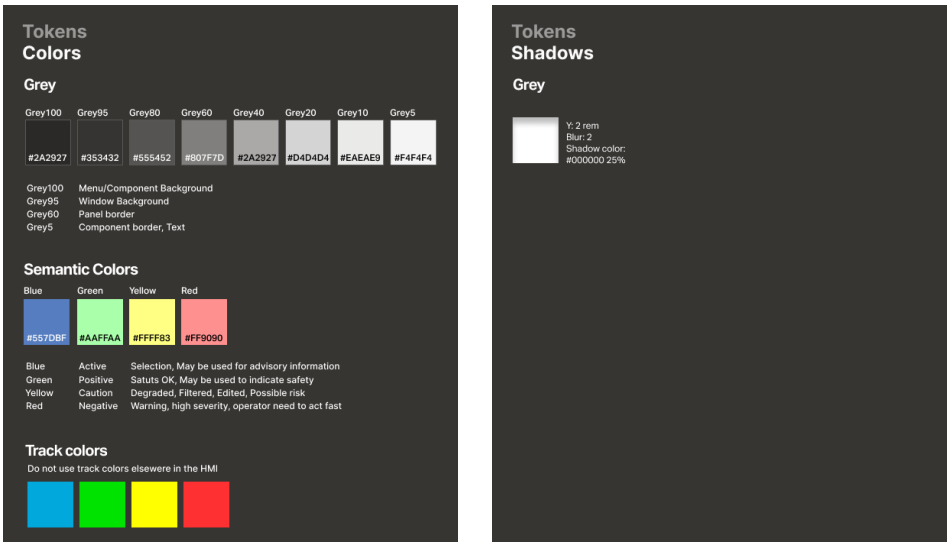


Figure 5.15: Colors (left) and Shadows (right)

Derived from the base unit, shapes such as radii and border widths could be defined and added to the design system, see Figure 5.16. The default typography defined in

the design system was retrieved from Saab's Brand portal.

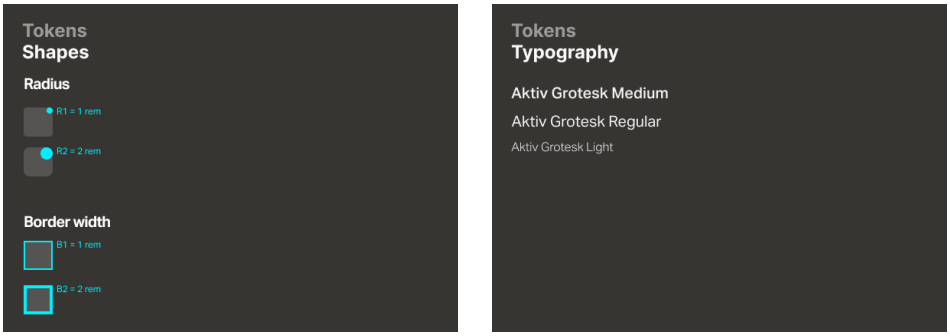


Figure 5.16: Shapes (left) and Typography (right)

5.7.2 Components

Having specified tokens, it was possible to create components based on these fundamental building stones. Combining the defined colors, geometry, radii, border widths, and typography into simple components such as input fields, combo boxes, and buttons, as illustrated in Figure 5.17.

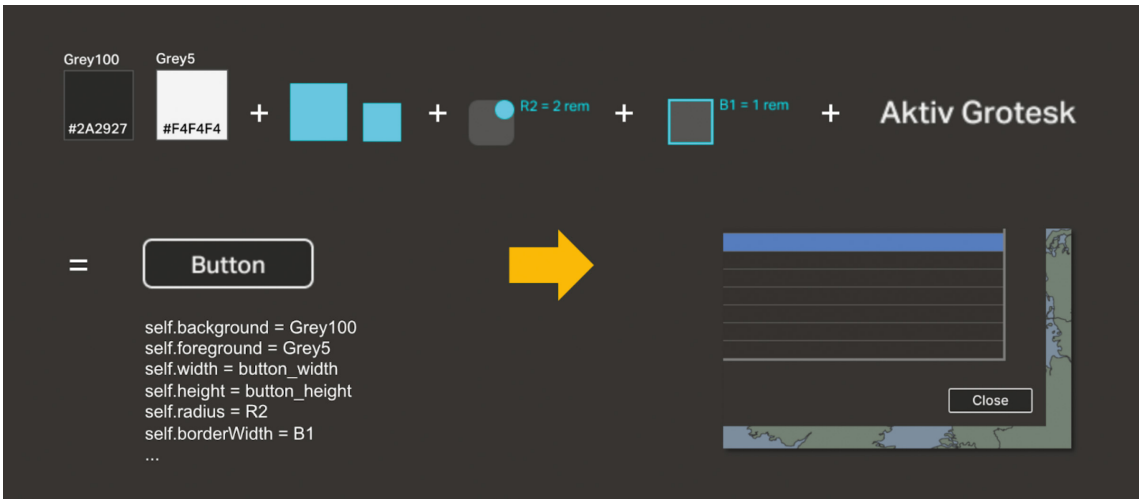


Figure 5.17: Example of how design tokens can be combined to build a button component

Following this principle, the following components in Figure 5.18-5.20 could be prototyped:

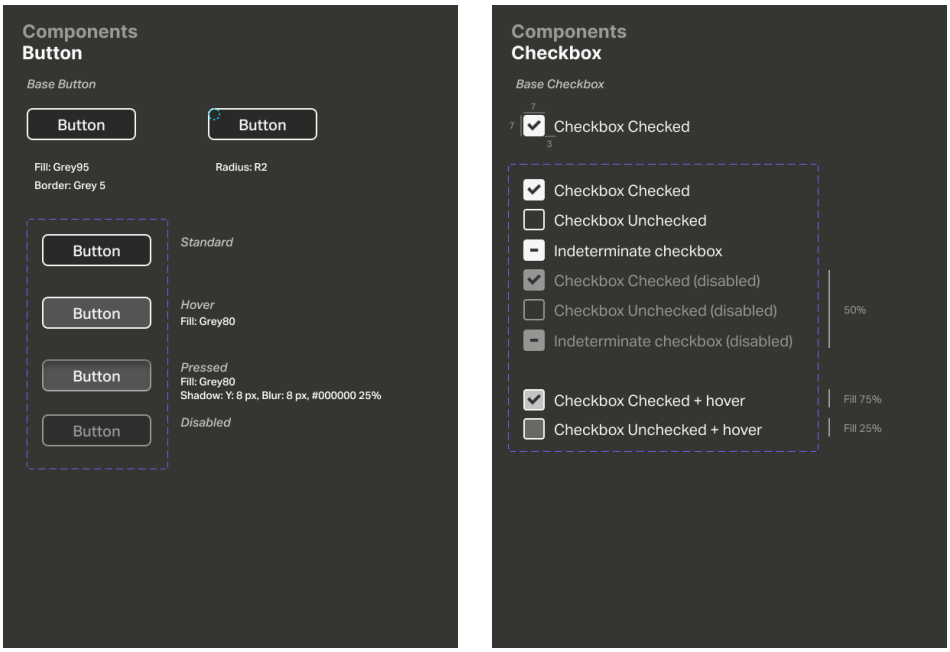


Figure 5.18: Button (left) and Checkboxes (right)

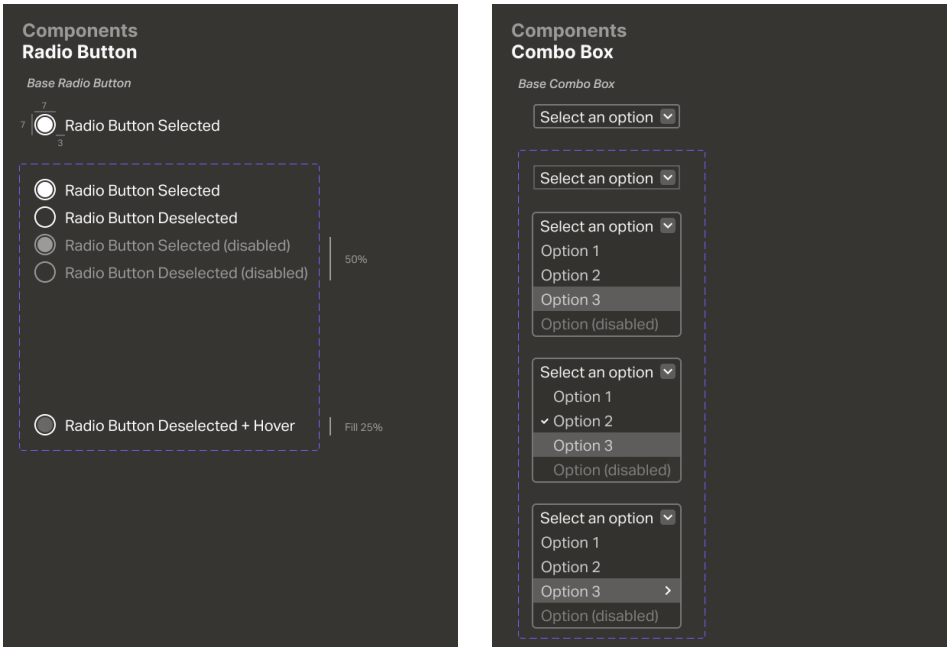


Figure 5.19: Radio button (left) and Combo box (right)

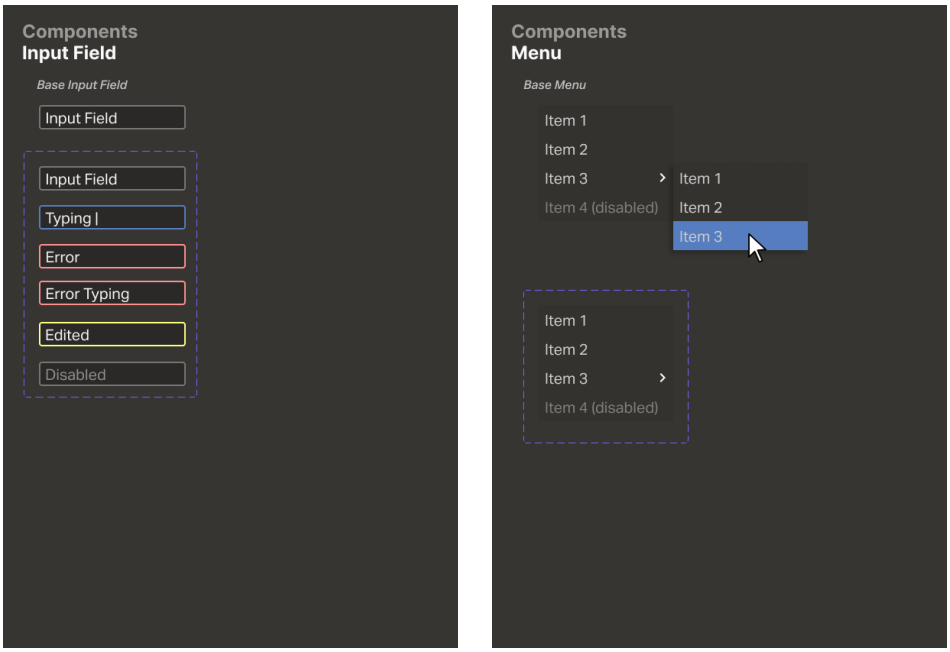


Figure 5.20: Input field (left) and Menu (right)

5.7.3 Component Group

Combining two or more components will create component groups, as illustrated in Figure 5.21.

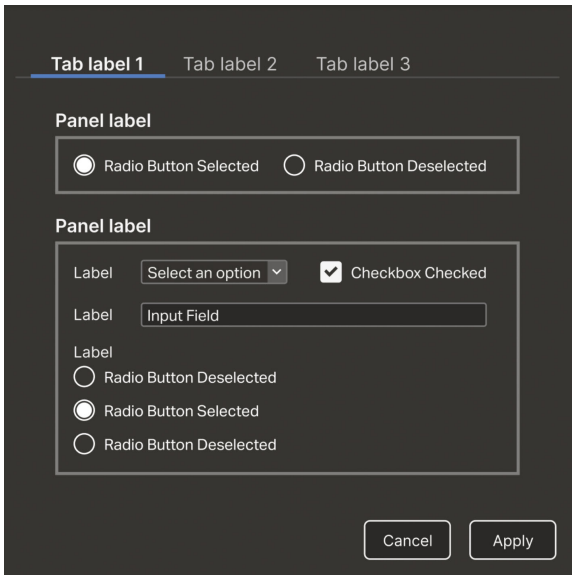


Figure 5.21: Example of how to combine components to component groups

5.7.4 Mockup

To continuously ensure that tokens and components could be used to build coherent interfaces within the context of tactical applications, a simple non-interactive

mockup was used to visualize how the resulting GUI would look, see Figure 5.22. The mockup was assembled utilizing Figma Components and defining pane sizes and colors based on tokens. This offered immediate and continuous feedback throughout the design process and enabled quick, iterative prototyping in high visual fidelity.

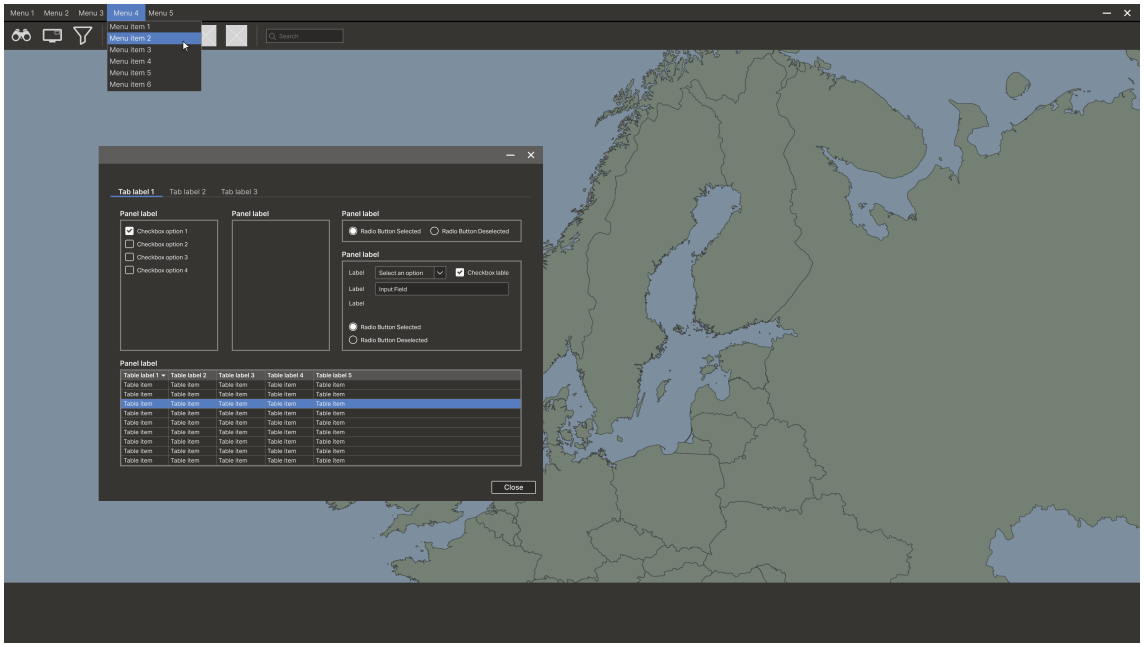


Figure 5.22: Example of how to combine components to component groups

5.7.5 Component Page

To test whether the tokens and components were able to be put into practice, a prototype of a component page was created containing guidelines, tokens, and graphics, as in Figure 5.23. The guidelines were assembled through a combination of theory, best practices, and the design guidelines created by the HFI-team at Saab. The component page included the following elements:

- Component name
- Usage principles
- Sizing
- Different states
- Colors
- Typography
- Dos & Don'ts
- Design guidelines

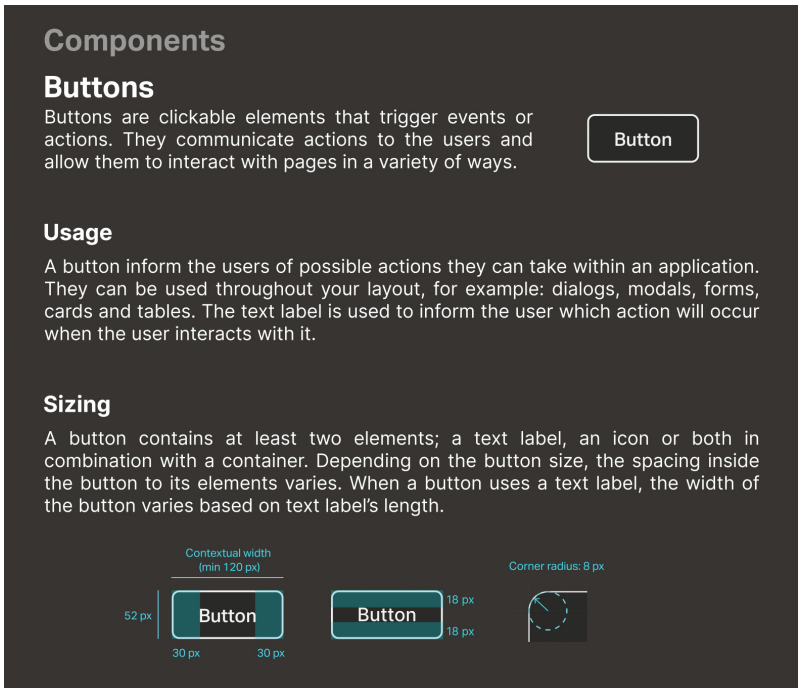


Figure 5.23: The component page of a Button. (see Appendix I - Implementation Test, for the complete component page)

5.8 Test

The test phase included an implementation usability test with five developers and a stakeholder feedback session with 54 attendees. The implementation usability test was carried out in order to evaluate the implementability of the components based on the visual attributes and guidelines provided. The purpose of the stakeholder feedback session was to present the findings from the project, gather initial thoughts, and get feedback from different perspectives on how the design system may be improved to better support Saab as an organization.

5.8.1 Implementation Usability Test

Inspired by usability testing, the implementation test was set up with the goal of testing whether it was possible for developers to follow the component page (see Figure 5.23) and its guidelines derived from the prototyping phase to implement graphical components in code. Here, it was important to find out what improvements could be made to the design system, its architecture, and intelligibility, following the question:

How well do the tokens and guidelines work in practice?

To test the structure of the implementation test, a pilot study was conducted with a developer working on site with the surface radar applications at the Gothenburg office. With the feedback from the pilot study, the implementation test was changed

slightly before the real test took place. Four AEW&C developers were invited to join the implementation test which took place remotely. Prior to the test, the button component prototype with its guidelines was emailed to developers in order to familiarize them with its content. The developers were also asked to prepare their integrated development environments (IDE) for coding interfaces. At the start of the test, the developers were asked to discuss the content of the prototype and the component documentation and if they saw any possible improvements. After the discussion, it was time to actually start the test session. Using the prototype as a foundation, the developers were asked to program and style a new button component using code. Setting a time limit at 30 minutes, the thesis duo wanted to see how the different developers took on the assignment and how they chose to solve it, how far they would come, and what challenges they would come across. The last ten minutes of the session were finished with discussing the assignment and their experience with the prototype.

The initial discussion revealed that the prototype's structure and content were understandable and relevant. However, there were some guidelines that were in conflict with how the developers work today to make the application more open for customization and customer requirements. The developers also expressed a need for more guidelines, such as how to use tooltips and the sizing of border widths. After testing the prototype and trying to implement the new component, the second discussion took place. Here it stood clear that 30 minutes was insufficient in terms of time for creating a new component from the guidelines. Developers expressed that it was quite difficult building components from scratch, and that there is a high risk of scaling errors and other bugs. The thesis duo noticed that attempts were made to style existing components in accordance with the guidelines as a way to save time. It was also explicitly suggested that it would be easier and more reliable to style existing components from Java Swing and C++/Qt in order to increase the speed and agility of implementation.

"Feels a bit too advanced to develop new components in such a short time period" - A2 Developer

"The prototype itself is good, but it would have been better if there were some information on how to specify the code" - A3 Developer

5.8.2 Stakeholder Feedback Presentation

As a design system impacts a large number of different stakeholders at Saab, it was deemed important to gather as many perspectives as possible on the project. Therefore, the thesis duo invited 300+ employees from different disciplines of the organization to a feedback presentation. This was seen as an excellent opportunity to educate and inspire people at Saab to recognize the benefits of a design system and align the company behind a common vision. Furthermore, the goal was to initiate a shared discussion among different teams that would help the project gain momentum within Saab.

The presentation itself lasted for approximately 30 minutes and took place in a hybrid format with 54 people from Saab's different offices. Attendees included members of the HFI-team as well as developers, designers, product owners, managers, system engineers, and executive staff. The presentation was positively received as a whole and sparked a fruitful conversation between the thesis duo and the stakeholders, as well as between the stakeholders themselves. Topics ranged from how Saab should manage ownership of design decisions at different levels to more technical questions about concrete ways for developers to contribute to the system. Furthermore, specific questions regarding how to manage the right balance between coherence and responsiveness were discussed in more detail.

A member from the HFi-team pointed out that the surface-based radar applications, due to their relatively high level of coherence, might provide a better platform for a design system pilot project in the future. Ultimately, Saab's brand design manager praised the initiative and called for a global forum with representatives from all of Saab's domains, in order to coordinate the continued development of the system.

5.8.3 Takeaways

From the implementation usability test it stood clear that it is difficult and time-consuming to create new components from scratch. Hence, it was decided, in consultation with stakeholders, to rely on an existing Look & Feel as the basis for the design system. This means that coherence between components can be managed through attributes and dependencies specified directly in the Look & Feel. As both A1 and A2 are styled using Java Swing, this was chosen as the primary focus while A3, being the only application using C++/Qt, would have to be styled to match the Java Swing Look & Feel. Still, the design system would need to include relevant code for both languages. Research revealed that there are newer, more modern Look & Feels offering more control over visual attributes in comparison to Nimbus (A1) and Synthetica (A3). The thesis duo benchmarked a few alternatives before selecting the *FlatLaf* Look & Feel, due to its customizability, native support for different color schemes, high-resolution scaling, and a flat design style which concurs with Saab's existing brand design language to great extent.

As the developers participating in the implementation usability test requested more guidelines, this need was further emphasized in the requirements list considering that it relates to the content and structure of the DSR and its component pages.

6
Results

In this chapter the result of the thesis project is presented in two parts. First, the final set of requirements to consider when designing a coherent design system for tactical GUIs are presented. Secondly, the design system is presented, including design tokens and the component library. Together, these results answer the research question of the thesis:

What are important requirements to consider when designing a design system for coherent tactical GUIs?

6.1 Design System Requirements

From each phase, requirements on how to design a design system for coherent tactical GUIs were produced. These requirements are divided into five factors: usage factors, contribution factors, factors for minimizing trade-off between coherence and responsiveness, Look & Feel factors, and organizational factors. The last two categories were not originally in the scope but were considered important aspects from the findings of the research. These requirements answer the two sub-questions:

1. What are important factors to enable developers to use and contribute to a design system for coherent tactical GUIs?
2. What are important factors to help minimize the trade-off between coherence and responsiveness in design systems?

These requirements aim to support development of design systems for tactical applications, focusing on accessibility and usability. Since the requirements are divided into factors, they can be used in multiple situations and in different combinations depending on a specific use case. In the case of this thesis project in collaboration with Saab, all the requirements should be considered when developing their design system. Therefore, after the requirements for each section of factor, it is described how the requirements should be applied to Saab's design system.

In the list of requirements presented below, there is an additional column showing from what part of the research the requirement was derived. This is presented as: LR=Literature review, I=Interview, CI=Contextual inquiry, and W=Workshop, T=Test.

6.1.1 Usage Factors

To allow the developer to use the design system as part of their work process it must be optimized for usability and accessibility. To do so, the design system must be usable and give value to the developers, in terms of workflow and collaboration.

To allow the developer to use the design system...

No.	Requirement	From
1.1	...it must be gathered and structured in a repository for easy access	LR, W
1.2	...it shall include core components for tactical applications	W
1.3	...it should include basic recurring component groups and patterns	W
1.4	...each component page must include application-independent information and guidelines for when and how to use each component	W, T
1.5	...it must include guidelines on how to modify the components and/or the patterns for specific customer requirements	W, I, T
1.6	...components shall be used and shared consistently across all tactical GUIs	Ci
1.7	...it must allow for easy identification and discoverability of components by, for example, include a logical component name in the component pages	W
1.8	...the components must be translated into usable code for each programming language used within the organization	W, I
1.9	...shall include guidelines for how to write and structure code for specific components or patterns	I, T
1.10	...the style guide must be visually and functionally compatible with the programming languages used within the organization	I
1.11	...the component page should include, at least, a logical component name, tokens, visual style, code, and guidelines for usage	I

To enable developers at Saab to use the design system, it is important that it is hosted in a repository for where it can be accessed and easily navigated. Furthermore, the repository must include all the components needed for developing the applications and explicit, clear guidelines explaining how and when to use them. To achieve a higher degree of coherence between the tactical applications, the design system should be used as a part of the development process. To make the development process more efficient, code should be included in the design system for each

programming language used at Saab.

6.1.2 Contribution Factors

Since a design system is an ever-growing system, it is important to keep it up to date. By letting the developers contribute to the design system will allow it to be updated in regards to what is new in the domain of the developers.

To allow the developers to contribute to the design system...

No.	Requirement	From
2.1	...there must be a way to add new components quickly	W
2.2	...it should allow for small modifications to be made by anyone	W
2.3	...it should include a way for developers to update the design system if, for example, there are any new functions or detected bugs	I
2.4	...it shall include a contribution section with guidelines for how to contribute	W

To allow developers at Saab to contribute to the design system, there must be a way to communicate with the design system team. It must be easy to contribute to the design system and to make modifications if there are any new functions or detected bugs. To avoid confusion and/or frustration, there must be guidelines on how to contribute to the design system.

6.1.3 Trade-off Factors

To create tactical applications that have internal and external coherence, yet are responsive enough to adjust for specific requirements, the design system must be developed to minimize the trade-off between the two of them.

To minimize the trade-off between coherence and responsiveness...

No.	Requirement	From
3.1	...the design system must allow developers to easily change attributes in the design to match customer requirements	I
3.2	...the design system must include guidelines on how to change attributes in the design without losing the company's brand identity	I
3.3	...the design system must include guidelines on how to design and contribute with new components and patterns	I
3.4	...the included code must be of such a basic character as to allow for modifications based on customer requirements	I

Since Saab receives a lot of project-specific customer requirements, it is important that the design system allows for some degree of freedom to make changes, without losing the brand identity. It is therefore important that there are guidelines on how to make changes and that the code included in the design system is of such a basic character that it allows for modifications.

6.1.4 Look & Feel Factors

To create applications that mirror the brand language of the organization and what it stands for, it is important that this can be communicated through the design system.

The design system's style sheet...

No.	Requirement	From
4.1	...shall include colors that reflect the company's brand identity	LR, I
4.2	...shall include a color scheme that makes the GUI clearly visible in the context where the operators operate	LR
4.3	...shall include a color scheme that makes the GUI comfortable to use and look at during extensive periods of time	LR

To achieve tactical applications that reflect Saab's quality and strong brand identity, the colors used must be representative. The colors should therefore be derived from and based on the colors specified in Saab's brand portal. Further, since the operators who operate the applications work in different lighting for many hours, the color scheme of the applications must be adapted to the context in which it is used.

6.1.5 Organizational Factors

To have a highly functional and up-to-date design system, some requirements must be handled on an organizational level.

From an organizational point of view, the design system...

No.	Requirement	From
5.1	...must be mandatory to use as part of the development process	W, I
5.2	...must be maintained by a dedicated team to handle changes and keep it up to date	LR, I, W
5.3	...should be accessible only to authorized personnel	W
5.4	...must be prioritized in the budget	I
5.5	...should allow for a way of continuously updating the images in the user manuals, to also accommodate for small updates	I

To maintain a design system and keep it up to date, it must be prioritized in the budget. It is also important that it is governed and maintained by a dedicated team to handle all the changes and contributions coming in. In Saab's case, it was shown from the user research that it is preferable to use a Hybrid team model; a dedicated design system team as well as members of different product teams. To make sure that the design system is used in the development process at Saab, it must be mandatory to use it. This, in turn, must be a demand coming from an executive level.

6.2 Design System

This section presents a foundation for a design system that has been designed for Saab's AEW&C applications. The design system, called PhoeniX Design System (see Figure 6.1), includes tokens, components, and patterns. Together, they are combined into mockups demonstrating how the system can be used to build coherent applications. Following decisions from the tests and input from stakeholders, another iteration of tokens and components was designed and built as a final concept for the system. Using FlatLaf Theme Editor (formdev.com/flatlaf/theme-editor), the thesis duo was able to design components based on FlatLaf's style foundation and Java Swing's default *JComponents*. This helped in achieving a higher-fidelity prototype of the design system, closer to how the components would appear when implemented in a developer environment.

Figure 6.1: Overview of the PhoeniX Design System comprised of tokens, components, patterns and a mockup exemplifying the use of the design system (top right)

6.2.1 Tokens

The final concept of the design system utilizes several fundamental tokens comprising the system. The purpose of the tokens is to define basic principles that guide design decisions concerning the use of colors, sizing, typography, and more when building components and patterns for Saab's AEW&C applications. Adhering to existing tokens ensures that components are designed in a purposeful, consistent, and coherent manner, in turn resulting in coherent GUIs. The tokens are based on ethnographic- and literature research and inspired by benchmarking existing systems and Saab's design language. The concept presents a foundation that should be expanded upon with additional tokens, guidelines, and do's & don'ts in the future.

6.2.1.1 Sizing & Spacing

The base unit for the system is 4x4 pixels. This aligns with the research as it allows for flexible designs that scale well on screens of various resolutions (see 3.4.3.3). The base unit can be multiplied by a factor to produce a range of standard sizes and spaces: 8, 12, 16, 20, 24, 28, and so on. For measurements smaller than 4, such as border widths, the base unit can be divided by 2 or 4 resulting in two additional units: 1 and 2. Figure 6.2 illustrates how the base unit is used to define a number of standard measurements and visual properties that the rest of the design system is based on. Adhering to this rule ensures that interfaces are built in a coherent manner that still allows for flexibility.

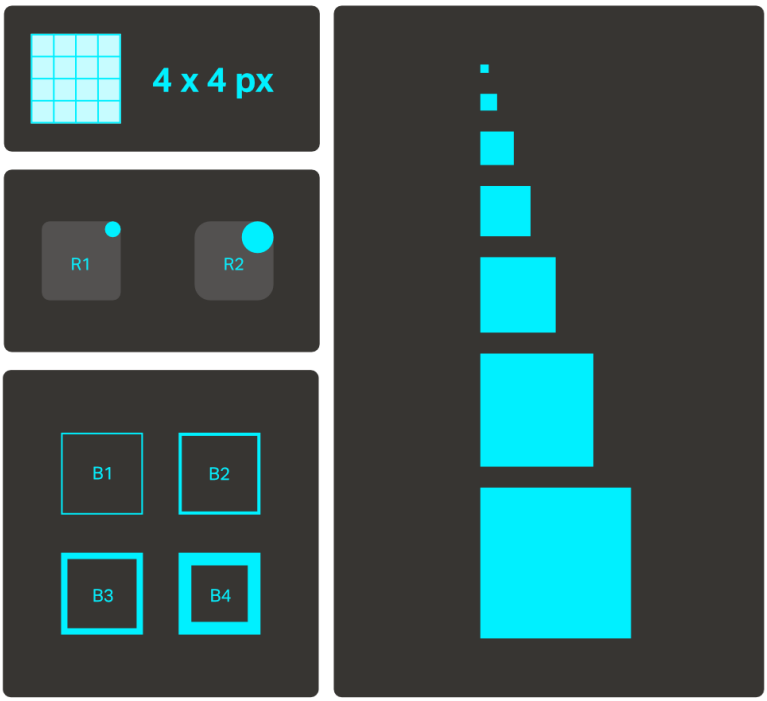


Figure 6.2: PhoeniX Design System uses a base unit of 4x4 pixels which can be multiplied by a factor to create a number of standard measurements and visual properties

PhoeniX Design System includes a few recurring size and spacing attributes that are either defined based on experimentation during prototyping or inspired by other design systems. Here, minimumWidth is based directly on FlatLaf's default minimumWidth for components in order to adhere to best practices. This means that standard components, such as input fields and buttons cannot be less than 64 pixels wide unless specifically overwritten to accomodate for contextual requirements. When deviating from the default sizing tokens, it is important to be purposeful and still adhere to multiples of 4 pixels in order to maintain the systematic approach.

```
base unit = 4
borderWidth = 1
cornerRadius = 4 or 8
minimumWidth = 64 or 72
Component.margin = 4, 8, 4, 8
Component.icon = 8
```

6.2.1.2 Typography

The typography of the design system mirrors that of the brand portal by using the Aktiv Grotesk font (see Figure 6.3). The font can be described as taking an authoritative but neutral position, supporting any message without overpowering it. It is a highly versatile font with 24 different styles including italics, bold and

black. The design system defines the default font for tactical GUIs as Aktiv Grotesk Regular. This is used for both headings and text, including component labels. The exception is default buttons which should use the Medium font variation to clearly signify the selected state. When needed, the tactical GUIs can utilize the light font, defined as Aktiv Grotesk Light and the semibold font, defined as Aktiv Grotesk Medium.

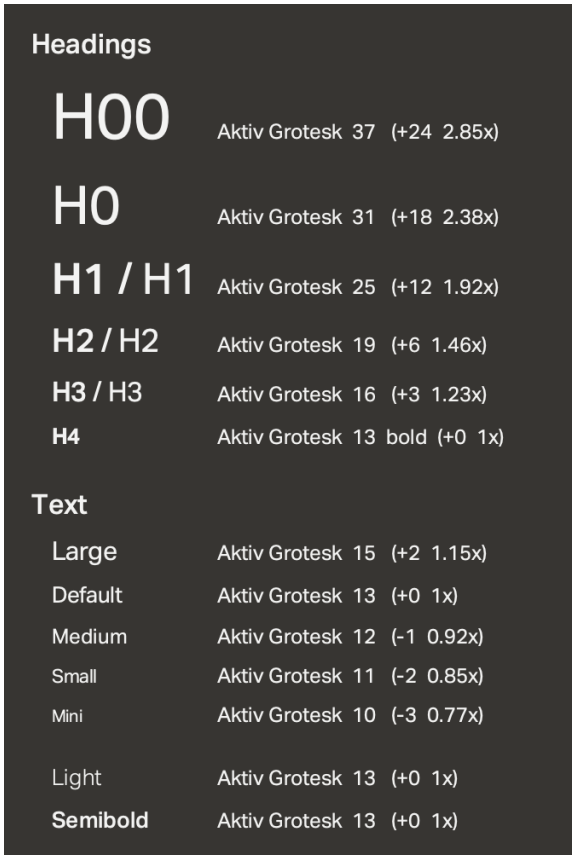


Figure 6.3: Typography of PhoeniX Design System

6.2.1.3 Colors

Background and foreground colors are chosen directly from the Brand Portal's dark color scheme for sake of consistency with Saab's existing design language. The colors work well for tactical applications as they are unobtrusive to the user and easily discernible from any semantic colors indicating statuses. Components have a background color 5% darker than the background of the interface to stand out to the user. The foreground color is used for text, labels, and borders of enabled buttons. FlatLaf automatically derives component colors from the background- and foreground colors through a variety of modifications and dependencies. This includes modifying the tint, lightness, and opacity of these colors to achieve a complementary color palette. In effect, large visual changes can be managed with ease to suit different company requirements and preferences whilst maintaining a coherent interface overall. This helps balance the trade-off between coherence and responsiveness.

```
## General colors ##
background = #373532
foreground = #f3f3f2
componentBackground = darken(@background, 5%) = #2a2826
```

Semantic colors are colors with an attributed meaning, such as negative (warning/error/high severity), positive (OK/safety), or neutral (selection). Semantic colors in PhoeniX Design System are partly derived from the Brand Portal but modifies and extends a few to accommodate for needs identified in tactical GUIs. Specifically, PhoeniX Design System introduces a systematic approach of achieving color variations to aid developers in responding to project or customer requirements regarding colors in a coherent, predictable manner. Lighter colors are achieved by increasing the lightness and decreasing the opacity of the original semantic color. Meanwhile, darker color variations are achieved by decreasing both lightness and opacity of the original semantic color. Lightness is increased or decreased in increments of 6 and opacity is changed in increments of 10% (see Figure 6.4).

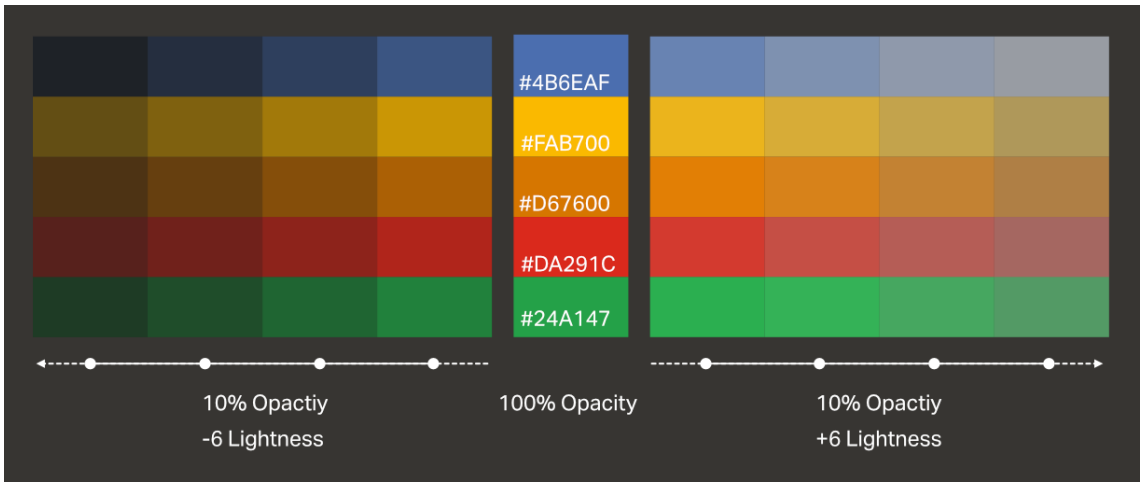


Figure 6.4: Semantic color palette and systematically generated variations

PhoeniX Design System also introduces an orange color with an attributed semantic meaning of "modified/unsaved" which separates it from the yellow (caution/possible risk). This is thought to provide more clarity to the operator.

```
## Semantic colors ##
blue = #4B6EAF
yellow = #FAB700
orange = #D67600
red = #DA291C
green = #24A147
```

Each semantic color and its variations can then be assigned a designated use case, such as:

```
selectionColor = blue = #4B6EAF
errorColor = changeAlpha(changeLightness($red, 24), 40%)
modifiedColor = changeAlpha(changeLightness($orange, 24), 40%)
```

Here, errorColor (used for indicating an invalid input in a field) is achieved by 4*6 steps of change in lightness and 4*10% change of opacity. By using the same rule for all colors and their respective variations, the system can be easily managed and expanded to eventually include more colors as well. This ultimately helps the system maintain a balance between coherence and responsiveness. If Saab would like to change any of the defined colors, the system itself can also be adjusted in order to produce a new color palette based on other rules.

6.2.2 Components

Components presented in the design system rely on Java Swing's JComponents, styled in FlatLaf with attributes customized according to defined tokens, see Figure 6.5. Component dependencies can easily be customized in code by changing what tokens constitute different attributes for each component (Baldwin, 2021). By consistently applying systematic sizing and color rules, components are allowed to be styled differently, without sacrificing coherence with the rest of the system. For example, design decisions regarding individual components, such as buttons and combo boxes, are flexible in terms of using different sizes, border widths, corner radii, border colors, and so forth without compromising the integrity of the system, see Figure 6.5.

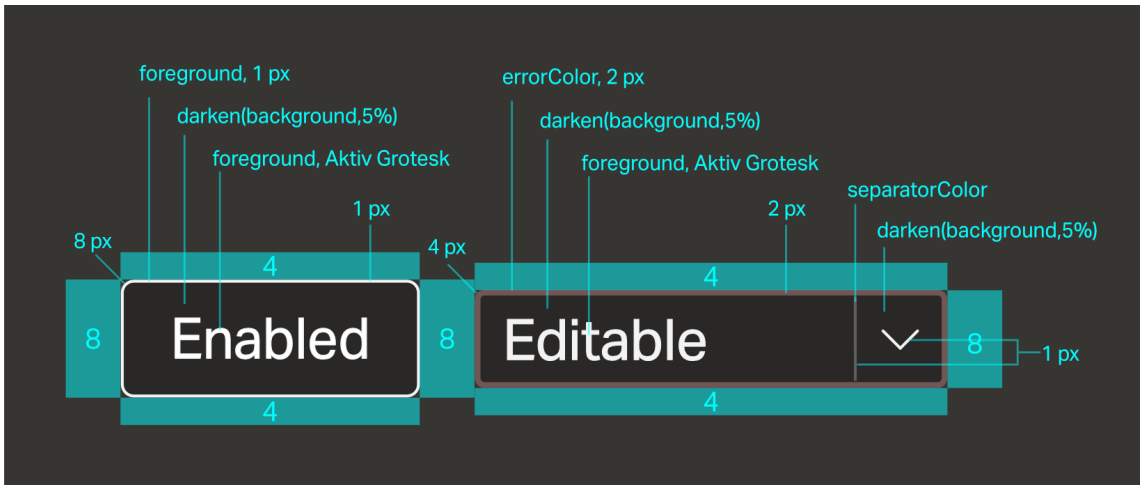


Figure 6.5: Button and Combo box as defined by different tokens

The component library also includes variants for some components, such as buttons with square or rounded edges and input field with and without icons. This enables developers to configure components to better suit contextual circumstances (Baldwin, 2021). Here, guidelines should be developed to help developers decide when to use which variant. As a result of this approach several, components have been systematically generated in this using FlatLaf Theme Editor, see Figure 6.6.

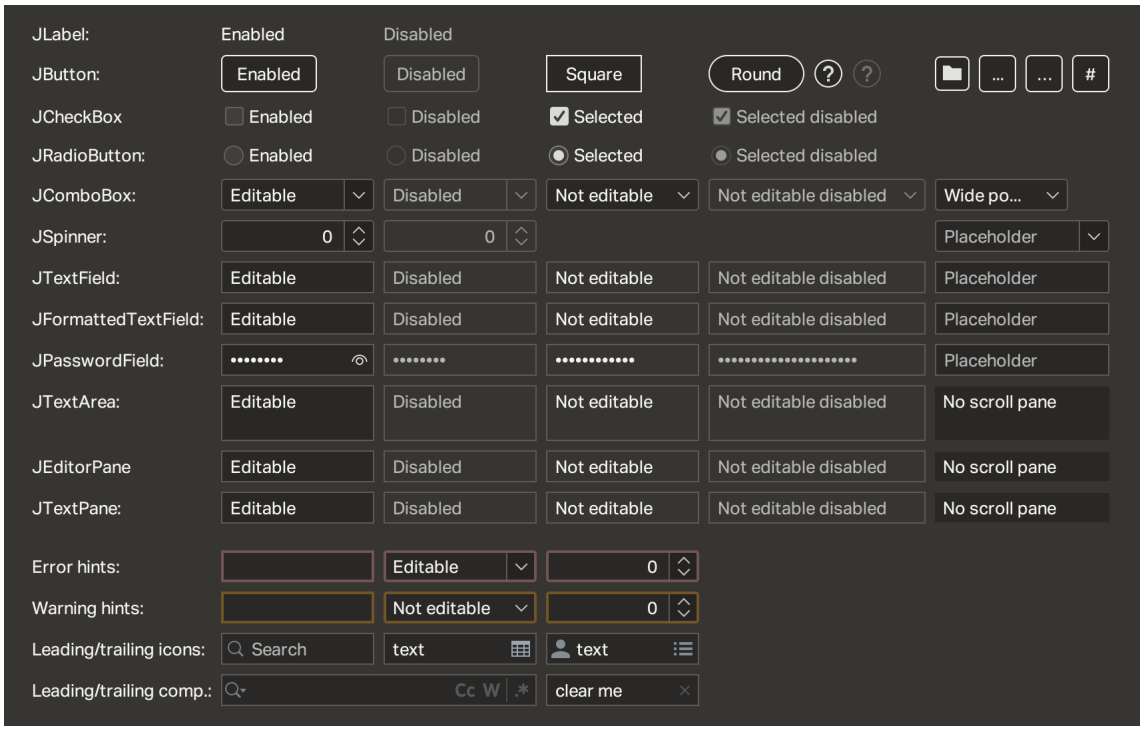


Figure 6.6: Part of PhoeniX Design System Component Library (see Appendix J - PhoeniX's Components for complete library)

6.2.3 Component Groups

In Saab's AEW&C applications, Tabs is a recurring and important pattern found across all of the interfaces (see 4.2.3). In FlatLaf, this basic pattern has been designed to encompass a large number of variations, offering responsiveness to a variety of different contextual circumstances, see Figure 6.7. Furthermore, the tabs are designed for optimal space efficiency in the GUIs through the use of scrolling behavior. This means that tabs can be kept on a single row and expand responsively to the size of the container window.

Another recurring pattern in Saab's AEW&C applications is groups of inputs organized in different panels. Each panel constitutes its own category of options that can include components such as radio buttons, checkboxes, sliders, input fields, and combo boxes. FlatLaf Theme Editor does not automatically generate component groups of inputs, but a mockup example is presented to demonstrate how these groups would appear using the PhoeniX Design System, see Figure 6.8.

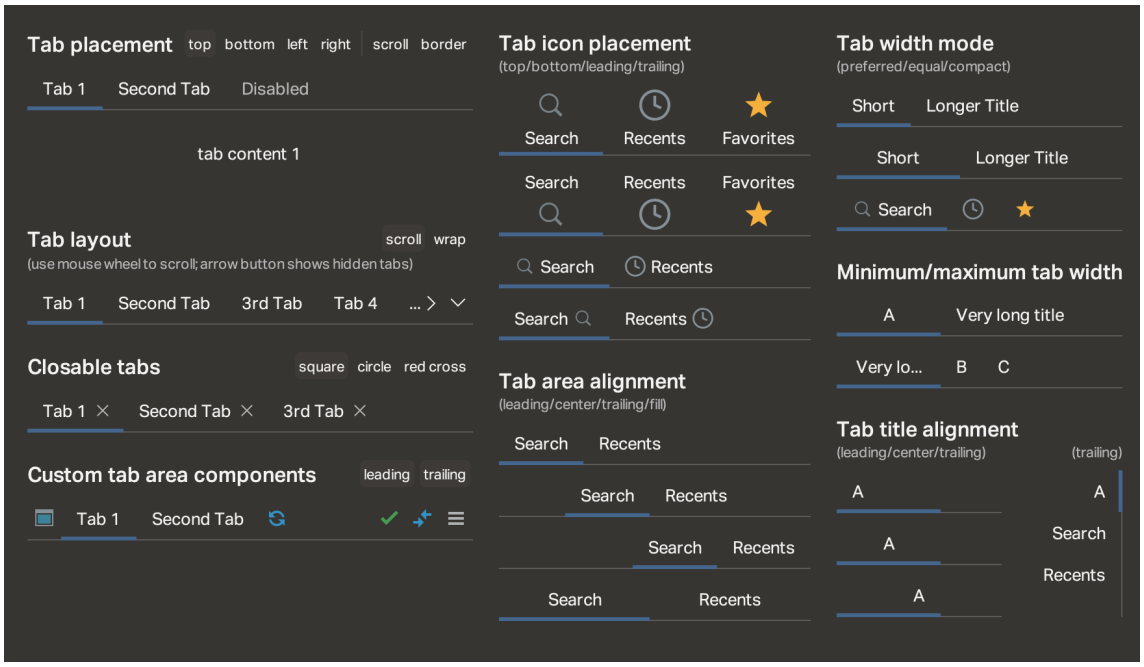


Figure 6.7: PhoeniX Design System component group: Tabs (see Appendix K - PhoeniX's Component Groups for complete library)

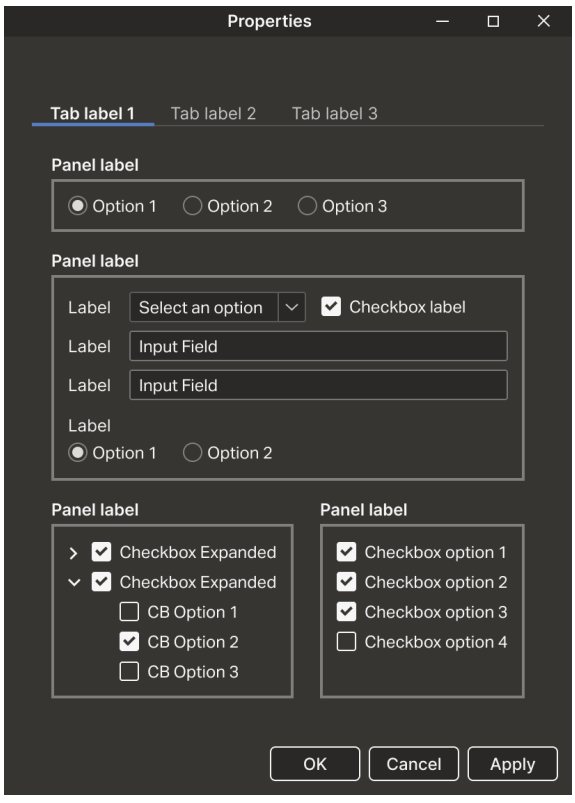


Figure 6.8: PhoeniX Design System component group: Properties

6.2.4 Mockup

Bringing together the tokens, components and component groups, a final proposed concept for the AEW&C application suite is presented in Figure 6.9. The interface demonstrates how adhering to the PhoeniX Design System can result in more visually and functionally coherent tactical GUIs.

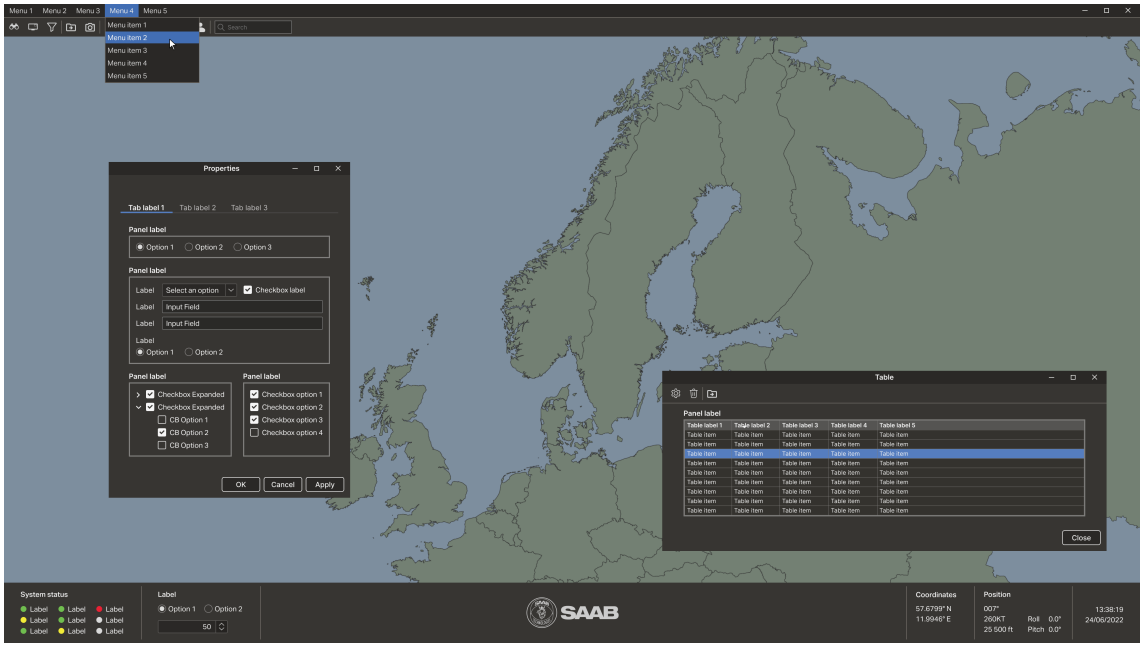


Figure 6.9: Mockup of a tactical interface based on the PhoeniX Design System

7
Discussion

This chapter discusses the design process and execution of the project as well as some of its implications on the results. This is followed by a discussion on the results themselves and the implications they might have for Saab and its stakeholders. Furthermore, ethical issues concerning the project and its results are identified and discussed. Finally, the chapter presents some suggested areas of future research and development regarding the design system.

7.1 Process & Execution

In this section, limitations and other potential factors that have and might have affected the outcome of the result of the thesis project are discussed.

7.1.1 Confidentiality & Complexity

Working within a company in the domain of governmental and military surveillance, there is naturally a lot of confidentiality and complexity to consider. The thesis duo has operated under a non-disclosure agreement prohibiting the sharing of certain information outside of the company. This has led to limitations in terms of academic supervision.

Furthermore, due to the confidentiality and complexity of Saab as a company, many employees have limited apprehension of the complete picture regarding the different systems. Most stakeholders interviewed have been experts within their specific application and customer project. Hence, developers and designers have only been able to provide their own perspectives as individual contributors. As a result, the thesis duo has needed to work with incomplete information and continuously reach out to stakeholders with more high-level knowledge in order to better understand the big picture. At times, this has led to conclusions and decisions being based on information that later was found faulty or very specific to an instance of a project. This has affected both the speed of progress and the validity of the ethnographic research leading up to the requirements list and the design of the design system. For example, to some of the stakeholders included in the research, A3 was considered one of the primary applications in the AEW&C system, however, this turned out to be the case in only one specific customer project, and not a part of Saab's regular product portfolio. If this information would have been discovered in an earlier stage in the thesis, A3 might have been excluded from the scope and thus making the

project more narrow. With that said, A3 is still a relatively new application, and might be included in Saab's regular product portfolio in the future, and therefore the thesis duo reasoned that it was a good choice including it in the project.

7.1.2 Remote Work

As the AEW&C application suite is developed in different locations, the majority of the ethnographic research has been conducted in a remote setting. Although stakeholders have been keen to spare time and help out throughout the project, all meetings needed to be formally scheduled several days in advance. This has impeded the possibility of getting quick answers whenever a question has arisen and also the risk of losing several days of work if a meeting would be canceled. Furthermore, as with any company dealing with sensitive information, there are certain rules for what online tools to use. In terms of online conversations and online workshops, the project has been limited to using Skype (skype.com) and its associated tools. Unfortunately, Skype's tools are limited in terms of collaborative functionality when compared to other online collaboration platforms, such as Miro (miro.com). This limitation has possibly affected the level of engagement and contribution during the workshop and interviews to a considerable extent. In order to obtain more rapid, hands-on knowledge of technical inquiries, the thesis duo has complemented AEW&C-specific developer interviews with in-house developers working with various surface radar applications.

7.2 Result

In this section, the result of the project, the requirements list and the design system itself, is critically analyzed and discussed.

7.2.1 Design System Requirements

The requirements list provided in this thesis project are important requirements to consider when developing a design system for coherent tactical GUIs. Since these requirements are derived from research at an organization, Saab in this case, it must be emphasized that these requirements do not account for the whole truth. A design system is very complex and has to be designed and built differently for each company or project to cater to specific needs. Each organization has different users, aims, and requirements and, hence, following these requirements will not necessarily result in a similar design system as the one presented in this thesis. Instead, the requirements list provided can be used as a foundation, providing some factors to consider when starting the development of a new design system in the domain of tactical GUIs.

In the case of this thesis project and its scope, not all of the requirements created have been considered when developing the foundation for Saab's design system. This is because the list of requirements also brings up factors that relate to the design system repository and the organizational implementation - both of which were not

part of this thesis project. These factors should be considered at a later point or in parallel with the design of the design system itself.

Finally, the requirements list might have turned out differently if more designers were included in the process. Since developers at Saab were set to be the primary user, most of the research has been centered around them and their needs. However, the result from this thesis project will still constitute a solid foundation and a starting point for the continued development of a design system at Saab.

7.2.2 PhoeniX Design System

The design system proposed in this thesis project provides a foundation for future work and the development of a design system at Saab. This means that the design system presented is by no means a complete design system, but has to be reiterated, extended and refined by Saab in the future. Considering that this thesis project has been limited in terms of time and scope, many of the applications in Saab's product portfolio have been excluded from the research. Therefore the design system must be further developed and tested in a real project with developers to get more insights on how it could be improved to better fit Saab as a whole. As Saab's applications are of a considerably more complex nature than a website, Saab's design system will differ from an "ordinary" website design system in terms of the need of responsiveness to meet specific customer requirements.

Further, finding the right balance between coherence and responsiveness is ultimately a philosophical question that Saab would need to experiment with. Some companies lean more towards customization and being dynamic to changing circumstances while others are more restricted and conservative, and both approaches are valid in their own respect. In the past, Saab has been leaning more towards being responsive to customer requirements at the expense of coherence (see Figure 7.1).

In PhoeniX Design System, a balance is achieved by enhancing both coherence and responsiveness at the same time, essentially moving the trade-off curve, as described by Harris and Henderson (2012), to minimize the compromise of one when pursuing the other (see Figure 7.2).

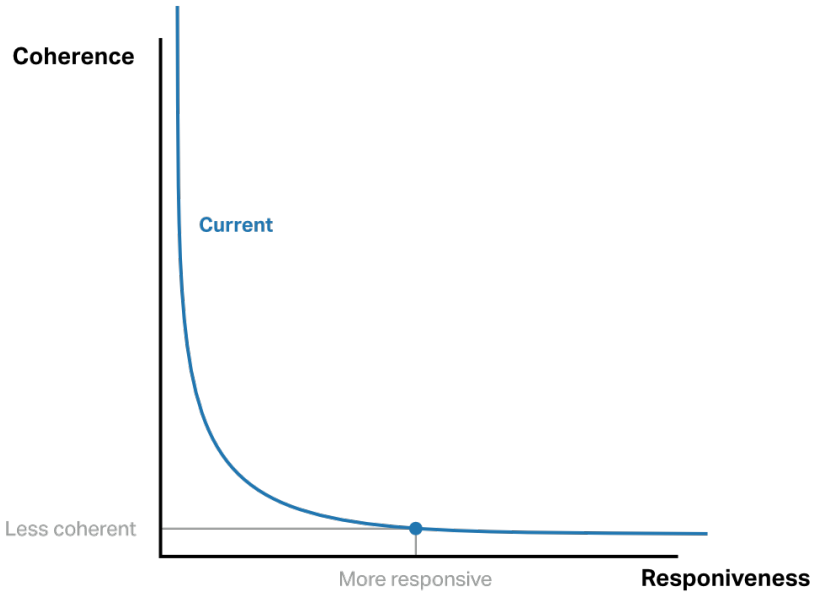


Figure 7.1: The current trade-off curve favoring responsiveness at the expense of coherence

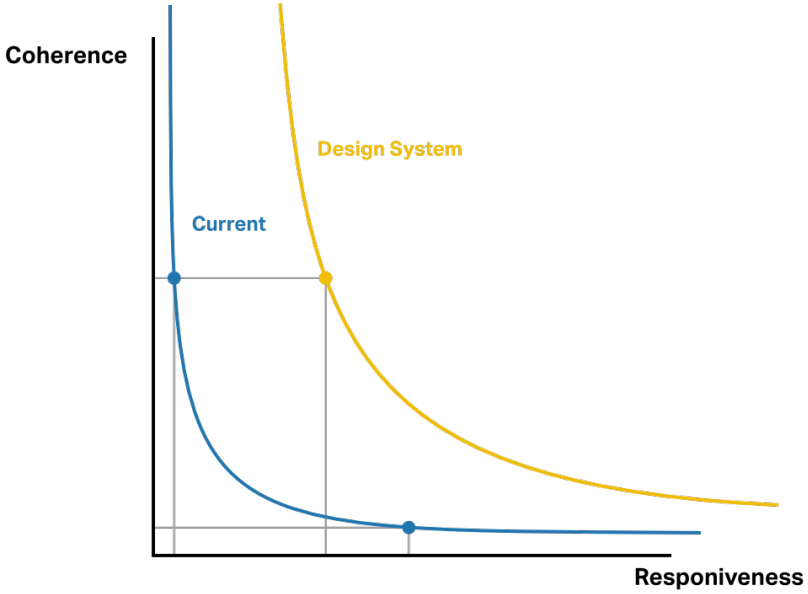


Figure 7.2: The PhoeniX Design System minimizes the trade-off between coherence and responsiveness by improving both at the same time

The design system affords increased coherence through the introduction of application-independent reusable design tokens, components and patterns that allows individual contributors to use and share resources across teams. By limiting the choices and possibilities to a discrete set of systematically defined choices, combined with guidelines for where and how to use assets, developers at Saab should be able to achieve

a higher level of coherence in their tactical GUIs going forward. Simultaneously, by designing the design system based on tokens that can easily be modified, and components with variants, developers can still stay responsive to customer requirements. The new set of systemized sizing, colors and fonts enables freedom of configuration without sacrificing coherence with the rest of the system. Moreover, by providing guidelines for how to contribute and make modifications to existing components and patterns, the infrastructure itself can be kept permeable to stakeholder needs as the system evolves, while also maintaining coherence with itself.

Since designing a design system is a large and complex endeavor, it has continuously been difficult to maintain a narrow scope for the project. Some parts of the system have ended up in a gray zone, leading to additional work that was originally out of the scope of this thesis. For example, at first, the project went towards a design system where components were created from scratch based on the tokens. However, the user research showed that it is quite difficult building components from scratch and that it might result in bugs, such as errors in scaling. Since Saab has no room for errors in their systems, it was decided to use existing components and then style these according to the design system using the FlatLaf Look & Feel and/or a stylesheet. Therefore, the thesis duo decided to code the FlatLaf editor to exemplify the use of the Look & Feel in combination with the design system, even if it was stated in the beginning of the project, that coding would be excluded. Consequently, time which could have been put on iterating the process were put on learning to control and program the FlatLaf components editor. However, the thesis duo reasoned that this was an important step to include in the project to be able to provide a solid hand-off for Saab.

The literature on how to design a design system highlights the importance of a multidisciplinary team of experts and the involvement of various stakeholders in order to produce an adept design system for the organization. Considering that this project was carried out by a small duo of two designers with no previous experience with the AEW&C domain, software development, or design systems, the outcome is expectedly not on par with what a multidisciplinary, expert team would be able to produce. However, throughout the project, the thesis duo has been able to counter some of these limitations by continuously employing an ethnographic, user-centered research approach to understand and deliver on the problem.

7.3 Stakeholder Implications

The introduction of a design system at Saab will have a number of implications for all kinds of stakeholders involved in this project. This section discusses some of these implications.

7.3.1 Developers

Developers are affected by the result of this thesis project since the design system directly influences individual, team, and cross-team workflows in terms of collaboration

and resource management. Specifically, the design system constitutes an additional repository for resources that need to be supported and maintained separately from the developer environment. By adding an additional source of information to consult, this could be seen as an inconvenience for developers who are used to working in frameworks. However, during the user research, developers indicated the need of having some sort of guidelines when developing new components or functions. Hence, the design system would be a tool to help developers make design decisions that are in line with Saab's brand identity. Further, by allowing developers to share knowledge and contribute to the design system, resources can be shared between teams and thereby make the workflow more efficient.

7.3.2 Designers

By introducing a design system, different designers at Saab will gain increased influence over the general design direction of AEW&C applications and be able to control specific design decisions on a system level. As a consequence, designers are affected in terms of responsibility, workflow, and workload which need to be taken into account when designing an effective design system.

7.3.3 Operators

Since high-level decisions are propagated throughout the entire AEW&C product portfolio, the final design of each GUI will be directly influenced by decisions made in a design system.

7.4 Ethical Considerations

Developing products for governmental and military purposes should be considered an ethical dilemma since the technology can be used for constructive and defensive functions but also abused and used in destructive and invading purposes. Consequently, the result of enhancing developer efficiency and user experience in the applications has the potential to benefit both sides of operations.

There is also an ethical consideration in how different types of functionality and information is visualized and presented to the users. Assuming that operators are influenced by the information and options available in the interface at any given moment, design decisions regarding the GUI have the potential to cause behavioral changes that cause errors and poor decisions in critical situations. Thus, it is important to carefully consider the implications of the design and evaluate prototypes to ensure safe, efficient, and satisfactory operation in complex and time-sensitive situations.

A third consideration of this project relates to the ethical dilemma introduced in changing applications that have already been developed, documented, and learned. The current interfaces are supported by countless hours of work, money, and time wherefore making drastic changes inevitably leads to a domino effect of additional

work. This ultimately introduces a lot of cost and time for multiple stakeholders as code has to be revised, processes need to change, documentation needs updating and operators, as well as instructors, might need retraining. The cascading implications of decisions taken on a design system-level must therefore be taken into account and involve stakeholders throughout the entire process to help maintain a realistic, feasible, and ethical approach.

7.5 Future Work

The results presented in this thesis should not be seen as the finished work of Saab's design system. Instead, the design system and requirements provided should be seen as a first iteration constituting the foundation for a design system and future iterations to come, see Figure 7.3. This section outlines suggestions for future work within the domain.

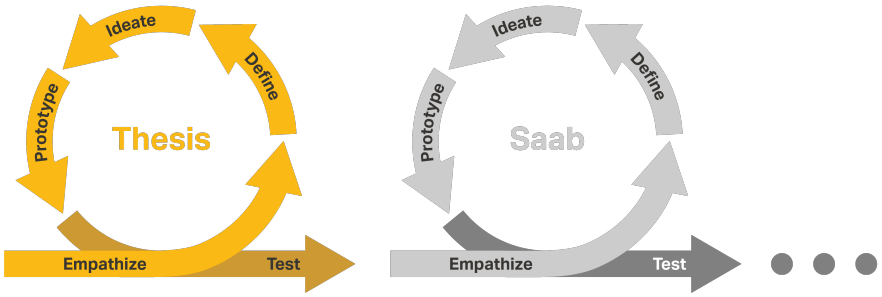


Figure 7.3: The thesis project is the first iteration of the design system and subsequent iterations need to be carried out by Saab in order to develop, integrate, and maintain the system over time.

This thesis project has designed and presented a foundation of a design system for Saab and the domain of tactical applications. Due to the limited time of this project, the scope was set to exclude corresponding code, development for the design system repository, and the implementation of the design system at Saab. Continuing working on the design system, Saab needs to set up a dedicated design system team who can manage the design system continuously and coherently in order to keep it usable and up to date. The thesis duo recommends employing a hybrid team model at Saab, to further include the users of the design system and thus develop it in a user-centered manner. Since the design system and its team is in need of their own budget, it is recommended to start off on a smaller scale, for example in a specific domain or project, rather than implement it on the whole organization at once.

The result presented from this thesis project includes design tokens and how they can be translated into a Saab Look & Feel, based on existing components within Java and their Look & Feel called FlatLaf. As of today, Saab either uses the Look & Feel called Synthetica or Nimbus to style their applications. A first step in implementing the new Saab Look & Feel would be for Saab to look into the possibility of starting using FlatLaf as their primary Look & Feel, instead of the two existing ones. The components should then be styled using the FlatLaf editor in combination with the design tokens presented. Using an existing Look & Feel as a base, will allow Saab to create their own Saab Look & Feel and save both money and time, and reduce the risk of errors that might occur when developing new components from scratch. These components must then be translated into code and implemented in the framework for the developers to use across the different platforms. To be able to share resources and contribute to a dynamic design system, it is also recommended

to use the same programming language and version across all platforms.

To make the design system accessible and usable to the organization, Saab needs to develop a repository or web-portal to gather all the components and documentation. Designing and developing a design system repository is a whole project in itself. It should be designed together with various users to make sure all important elements are included and that it is created in a user-friendly manner. To help developers in when, why, and how to use specific components and patterns, guidelines for usage should be included for each component and pattern. The thesis duo therefore recommend Saab to extend the existing HMI-guidelines and complement them with component specific guidelines and include them in the design system.

8

Conclusion

The purpose of the thesis project has been to research what factors influence why incoherence emerges in functionally similar AEW&C applications at Saab and to identify opportunities for improvement. What started as a project focused on designing a unified base product to create increased consistency between three AEW&C applications in Saab's product portfolio, eventually pivoted into designing a design system for Saab's airborne surveillance solutions. The goal has been to supply developers of AEW&C applications with a library of easily accessible design resources and guidelines to help achieve coherence and compliance with user expectations and design principles. More specifically, the aim has been to answer the following research question:

What are important requirements to consider when designing a design system for coherent tactical GUIs?

The research question has been addressed by carrying out an extensive research for design process, involving both ethnographic and participatory research methodologies, to co-design a foundation for a design system together with developers and other stakeholders at Saab. Further, to create a genuine and thought-through foundation for the design system, a requirements list of the most important requirements to consider when designing a design system for coherent tactical GUIs was derived from the design research. By using an iterative design thinking process, both the requirements and the design system have been considered from different point-of-views and by a variety of stakeholders to build a solid foundation.

The project has resulted in a list of requirements that should be considered when designing and developing a design system in the domain of tactical applications, divided into five factors:

- Usage factors - Requirements to make the design system usable for the developers.
- Contribution factors - Requirements to allow developers to contribute to a dynamic design system.
- Trade-off factors - Requirements for minimizing the trade-off between coherence and responsiveness.
- Look & Feel factors - Requirements for the styling and experience of the GUI.
- Organizational factors - Requirements that must be in place to enable and maintain a design system.

The requirements presented in this thesis covers the whole development process of a design system for tactical GUIs. Therefore, not all requirements have been considered in this project when designing the foundation for Saab's design system since they cover areas that are outside of the project scope, like implementation and the design system repository. The design research with the employees at Saab showed that they are okay with their way of working today, but that they see potential areas of improvement. For example, it would be very beneficial for all teams to be able to share resources and to communicate better between teams and projects since they put so much time and money into each project. The developers liked the idea of implementing a design system in their work process but emphasized the importance of making it responsive to customer requirements.

Prior to this project, Saab had already made some design decisions for their GUIs, like what colors or fonts to use, that were used as a basis for the design system. The design research showed that it could be advantageous if using existing components, instead of creating new ones from scratch, in terms of money and time saved, and to reduce the risk of bugs, such as scaling errors. In the domain of tactical applications, there is little or no room for errors, and it was therefore decided to use the existing components from each programming language and style them according to the design system. For the Java-based applications, it was recommended to make a switch from the existing Look & Feel to a more modern Look & Feel called FlatLaf since it supports a higher degree of customization. The relevant requirements from the requirement list and the insights gained from the design research were then used to prototype, design, and evaluate a foundation to a design system for Saab, including design tokens, components, and patterns to develop coherent, yet responsive, tactical GUIs. In addition, a reference product was prototyped to exemplify the use of the design system.

This thesis project contributes to the research field of design systems by addressing the gap within the domain of design systems for tactical GUIs. By developing a foundation for a design system for Saab, together with important requirements for using, contributing, and minimizing the trade-off between coherence and responsiveness when designing a design system, the research question has been successfully addressed.

Bibliography

Anand, G., & Kodali, R. (2008). Benchmarking the benchmarking models. *Benchmarking: An International Journal*, *15*(3), 257–291. https://doi.org/10.1108/14635770810876593

Backus, C. (2021). Customization vs. Configuration in Evolving Design Systems - Spotify Engineering : Spotify Engineering. https://engineering.atspotify.com/2021/04/customization-vs-configuration-in-evolving-design-systems/

Baldwin, N. (2021). Revisiting the anatomy of a design system.

Batterbee, K., & Koskien, I. (2010). Co-experience: user experience as interaction. *http://dx.doi.org/10.1080/15710880412331289917*, *1*(1), 5–18. https://doi.org/10.1080/15710880412331289917

Bligård, L.-O. (2015). *ACD3 - Utvecklingsprocessen ur ett människa-maskinperspektiv* (2nd ed.). Chalmers University of Technology. https://doi.org/10.13140/RG.2.1.1954.4400

Blomberg, J., Giacomi, J., Mosher, A., & Swenton-Wall, P. (1993). Ethnographic field methods and their relation to design. *Participatory design: Principles and practices* (pp. 123–155). https://doi.org/10.1201/9780203744338

Bohn, D. (2018). Google's software design is having a reformation. https://www.theverge.com/2018/5/10/17339230/google-material-design-theme-update-new-tools-matias-duarte

Brantingham, R. (2019). The NELSON Standards - A design system for the Royal Navy - Defence Digital. https://defencedigital.blog.gov.uk/2019/12/18/nelson-standards-creating-royal-navy-apps-with-a-consistent-look-and-feel/

Buchanan, R. (1992). Wicked Problems in Design Thinking. *Design Issues*, *8*(2). https://doi.org/10.2307/1511637

Burkett, I. (2012). An Introduction to Co-Design. *Sydney: Knode*.

Chapanis, A. (1985). Some Reflections on Progress. *Proceedings of the Human Factors Society Annual Meeting*, *29*(1), 1–8. https://doi.org/10.1177/154193128502900102

Colineau, N., Lampert, A., & Paris, C. (2004). Task-sensitive user interfaces. *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, 218–225. https://doi.org/10.1145/989863.989899

Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). *About face: the essentials of interaction design* (4th ed.). John Wiley & Sons.

Curtis, N. (2016a). Space in Design Systems. From Basics to Expanded Concepts to... | by Nathan Curtis | EightShapes | Medium. https://medium.com/eightshapes-llc/space-in-design-systems-188bcbae0d62

Curtis, N. (2016b). Tokens in Design Systems. 10 Tips to Architect & Implement Design... | by Nathan Curtis | EightShapes | Medium. https://medium.com/eightshapes-llc/tokens-in-design-systems-25dd82d58421

Dam, R. F. (2021). 5 Stages in the Design Thinking Process. https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process

Dam, R. F., & Siang, T. Y. (2020a). Introduction to the Essential Ideation Techniques which are the Heart of Design Thinking. https://www.interaction-design.org/literature/article/introduction-to-the-essential-ideation-techniques-which-are-the-heart-of-design-thinking

Dam, R. F., & Siang, T. Y. (2020b). Stage 4 in the Design Thinking Process: Prototype. https://www.interaction-design.org/literature/article/stage-4-in-the-design-thinking-process-prototype

Dam, R. F., & Siang, T. Y. (2020c). What Kind of Prototype Should You Create? https://www.interaction-design.org/literature/article/what-kind-of-prototype-should-you-create

Downton, P. (2003). *Design research*. RMIT Publishing.

Edelberg, J., & Kilrain, J. (2020). Design Systems. *Proceedings of the 38th ACM International Conference on Design of Communication*, 1–3. https://doi.org/10.1145/3380851.3416743

Fessenden, T. (2021). Design Systems 101. https://www.nngroup.com/articles/design-systems-101/

Forlizzi, J., Stolterman, E., & Zimmerman, J. (2009). From Design Research to Theory - Evidence of a Maturing Field. *Proceedings of the 3rd Conference of the International Association of Societies of Design Research (IASDR'09): Rigor and Relevance in Design*, (October 2015).

Frost, B. (2016). *Atomic Design*.

Gibbons, S. (2016). Design Thinking 101.

Gibbons, S. (2019). User Need Statements. https://www.nngroup.com/articles/user-need-statements/

Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, *32*(10), 1164–1173. https://doi.org/10.1145/67933.67934

Hanington, B., & Martin, B. (2012). *Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions*. Quarto Publishing Group USA. https://doi.org/10.5860/choice.49-5403

Harley, A. (2017). Ideation for Everyday Design Challenges. https://www.nngroup.com/articles/ux-ideation/

Harris, J., & Henderson, A. (2012). Coherence and responsiveness. *Interactions*, *19*(5), 67–71. https://doi.org/10.1145/2334184.2334199

Harris, J., & Henderson, A. (1999). A better mythology for system design. *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, 88–95. https://doi.org/10.1145/302979.303003

Hassenzahl, M., & Tractinsky, N. (2006). User experience - a research agenda. *Behaviour & Information Technology*, *25*(2), 91–97. https://doi.org/10.1080/01449290500330331

Holtzblatt, K., & Jones, S. (1993). Contextual Inquiry: A Participatory Technique for System Design. *Participatory design* (pp. 177–210). CRC Press. https://doi.org/10.1201/9780203744338-9

Hwang, S. M., & Jo, J. Y. (2008). A development method of GUI in military system software. *Proceedings of the 2008 Advanced Software Engineering and its Applications, ASEA 2008*. https://doi.org/10.1109/ASEA.2008.48

International Organization for Standardization. (2013). Usability of consumer products and products for public use — Part 2: Summative test method (ISO Standard No. 20282-2:2013). https://www.iso.org/obp/ui/#iso:std:iso:ts:20282:-2:ed-2:v1:en

International Organization for Standardization. (2019). Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems (ISO Standard No. 9241-210:2019). https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-2:v1:en

Jordan, P. W. (2002). An Introduction to Usability. *An Introduction to Usability*. https://doi.org/10.1201/9781003062769

Kaplan, K. (2019). 5 UX Workshops and When to Use Them: A Cheat Sheet. https://www.nngroup.com/articles/5-ux-workshops/

Knopf, J. W. (2006). Doing a Literature Review. *PS: Political Science & Politics*, *39*(01), 127–132. https://doi.org/10.1017/S1049096506060264

Livingston, M. A., Ai, Z., Karsch, K., & Gibson, G. O. (2011). User interface design for military AR applications. *Virtual Reality*, *15*(2-3). https://doi.org/10.1007/s10055-010-0179-1

Luostarinen, R., Manner, J., Määttä, J., & Järvinen, R. (2010). User-centered design of graphical user interfaces. *Proceedings - IEEE Military Communications Conference MILCOM*. https://doi.org/10.1109/MILCOM.2010.5680400

Marchi, S. (2021). Using open source Design Systems: pros and cons. https://bootcamp.uxdesign.cc/using-open-source-design-systems-pros-and-cons-a80eef3f95c3

Maxwell, K. D., Van Wassenhove, L., & Dutta, S. (1996). Software development productivity of European space, military, and industrial applications. *IEEE Transactions on Software Engineering*, *22*(10), 706–718. https://doi.org/10.1109/32.544349

Mkrtchyan, R. (2018). Wireframe, Mockup, Prototype: What is What? https://uxplanet.org/wireframe-mockup-prototype-what-is-what-8cf2966e5a8b

Nielsen, J. (1994). How to Conduct a Heuristic Evaluation. https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

Nielsen, J. (2012). Usability 101: Introduction to Usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Norman, D. (2013). *The Design of Everyday Things* (Rev.ed.). Basic books. https://doi.org/10.15358/9783800648108

Norman, D., & Nielsen, J. (n.d.). The Definition of User Experience (UX). https://www.nngroup.com/articles/definition-user-experience/

Norman, D. A. (2014). Some Observations on Mental Models. *Mental models* (pp. 15–22). Psychology Press. https://doi.org/10.4324/9781315802725-5

Oviatt, S. (2006). Human-centered design meets cognitive load theory: Designing interfaces that help people think. *Proceedings of the 14th Annual ACM International Conference on Multimedia, MM 2006*, 871–880. https://doi.org/10.1145/1180639.1180831

Raven, M. E., & Flanders, A. (1996). Using contextual inquiry to learn about your audiences. *ACM SIGDOC Asterisk Journal of Computer Documentation*, *20*(1), 1–13. https://doi.org/10.1145/227614.227615

Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, *4*(2), 155–169. https://doi.org/10.1007/BF01405730

Sanders, E. B.-N., & Stappers, P. J. (2008). Co-creation and the new landscapes of design. *CoDesign*, *4*(1), 5–18. https://doi.org/10.1080/15710880701875068

Sharp, H., Preece, J., & Rogers, Y. (2019). *Interaction Design: Beyond Human-Computer Interaction* (5th ed.). John Wiley & Sons, Incorporated.

Spool, J. M. (2018). Consistency in Design is the Wrong Approach.

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

Think Design. (2020). Try it yourself. https://think.design/user-design-research/try-it-yourself/

Vesselov, S., & Davis, T. (2019). *Building Design Systems*. Apress. https://doi.org/10.1007/978-1-4842-4514-9

Wahid, A., Cheran, E., & Six, J. (n.d.). Figma Tokens. https://www.figma.com/community/plugin/843461159747178978/Figma-Tokens

Walker, M., Takayama, L., & Landay, J. A. (2002). High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes when Testing Web Prototypes. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *46*(5), 661–665. https://doi.org/10.1177/154193120204600513

What is Heuristic Evaluation? (n.d.). https://www.interaction-design.org/literature/topics/heuristic-evaluation

Wickens, C., Gordon, S., & Liu, Y. (1997). *An introduction to human factors engineering*. Addison Wesley Longman, inc. https://doi.org/10.1177/106480469900700209

Yllobre, C. (2020). Design for Coherence, not Consistency.

Zuschlag, M. (2010). Achieving and Balancing Consistency in User Interface Design. https://www.uxmatters.com/mt/archives/2010/07/achieving-and-balancing-consistency-in-user-interface-design.php#

A

Appendix

Appendix A - Thesis Proposal

Design of a Base Product for Tactical Human Machine Interfaces

Background

The product catalogue at Saab Surveillance includes several systems that are operated through different Human-Machine-Interfaces (HMI), many of which are tactical software applications that are operated via a digital display.

The technical and operational needs differ for different Airborne Early Warning (AEW) and surface based radar systems and for different operator groups, so each tactical HMI is adapted for the intended use and users. However, many needs are similar or identical but are met with different solutions.

We believe that both the development and the use of tactical HMIs could be more effective and efficient with a common base product for tactical HMIs.

Thesis Description

In this Thesis, you will study the operational and tactical needs for radar operators and you will design a base product for tactical HMIs. The research should focus on finding logical workflows and layouts to make operators work more effective, efficient and satisfactory. Also, the technical possibilities and limitations should be identified and considered in the design.

The final concept of the base HMI must be flexible, to suit different Saab HMI applications, with no or few adjustments, and different operator roles. Recommendations for how to use and adjust the base HMI for different needs should be included in the deliverables.

Recommended activities:

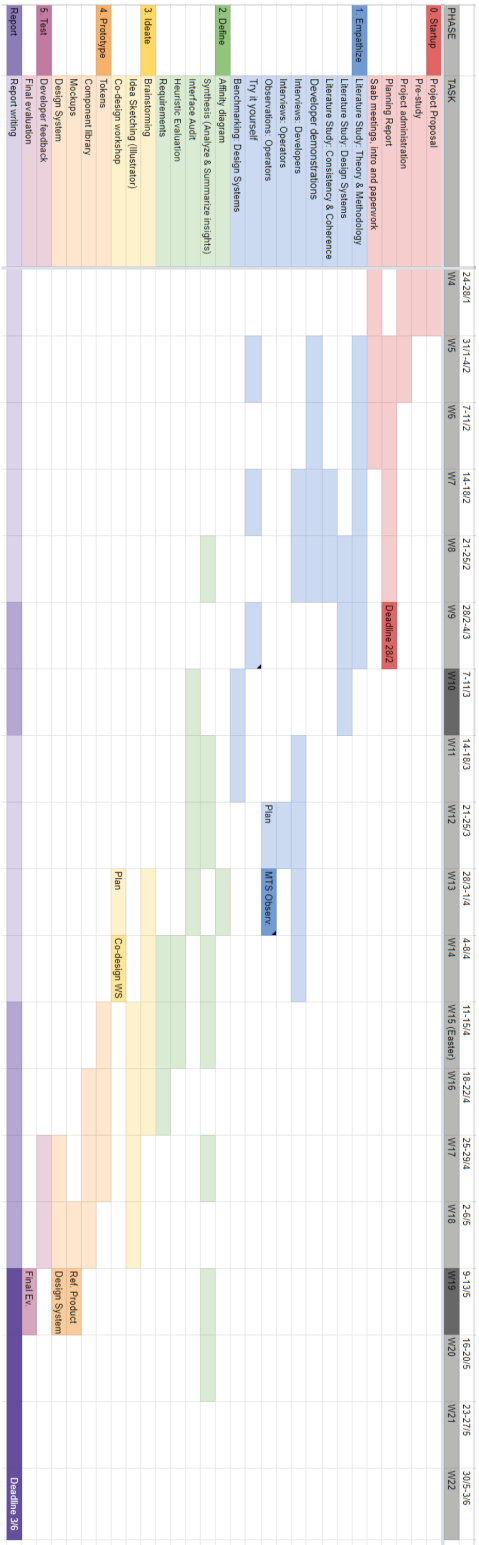
- User research
- Task analysis
- Design, concept and prototyping (e.g. Adobe XD or programming)
- Usability testing

Student Profile

We are looking for one or two students with an educational background within human factors, interaction design and user experience.

We expect you to be at the end of your master's degree in Interaction Design and Technologies, Industrial Design Engineering, Cognitive Science or equivalent, and is eligible for your 30 HP degree project.

Appendix B - Gantt Chart



Appendix C - Benchmark Design Systems

What to include in a design system repository

	Microsoft	IBM	Adobe	Google	Shopify
Open scorce					
Public-facing	✓	✓	✓	✓	✓
Repository	✓	✓	✓	✓	✓
Layout					
Grid/Spacing	✓	✓	✓	✓	✓
Patterns	✓	✓	✓	✓	✓
Design styles					
Typography	✓	✓	✓	✓	✓
Text/writing style	✓	✓	✓	✓	✓
Color	✓	✓	✓	✓	✓
Object style (lines, rounding, etc)	✓	✓	✓	✓	
Iconography	✓	✓	✓	✓	✓
Sound	✓			✓	✓
Motion	✓	✓	✓	✓	
Interaction states / Status indicators	✓	✓	✓	✓	✓
Images/Photos	✓	✓	✓	✓	
Illustrations / Animations		✓	✓	✓	✓
Data visualization		✓	✓	✓	✓
Content					
Voice & tone		✓	✓	✓	✓
Vocabulary & Grammar		✓	✓		✓
Naming					✓
Design principles	✓	✓	✓	✓	

Resources					
Component library	✓	✓	✓	✓	
UI kit	✓	✓	✓	✓	
Tokens	✓	✓	✓		✓
Theming		✓		✓	✓
Font-kits	✓	✓	✓	✓	✓
Icons (SVG)		✓	✓	✓	✓
Logos		✓			✓
Components					
with code	✓	✓	✓	✓	✓
without code					
Different frameworks		✓	✓	✓	
Live code / Demo		✓			✓
Usability					
Accessibility	✓	✓		✓	✓
Internationalization / Inclusion			✓		✓
Information Architecture	✓			✓	✓
Miscellaneous					
Markerting		✓			
What's new / Versioning	✓	✓	✓	✓	
Blog				✓	
Help / Reporting issue		✓	✓		
Contribution		✓			

Component library content

New table

	Fluent	Carbon	Spectrum	Material Design	Polaris
Namn	✓	✓	✓	✓	✓
Description	✓	✓	✓	✓	✓
Table of Content / Page navigator	✓	✓	✓	✓	✓
Overview / When to use / purpose	✓	✓	✓	✓	✓
Tokens		✓		✓	
Component examples	✓	✓	✓	✓	✓
Code	✓	✓	✓	✓	✓
Different framework		✓		✓	✓
Different platforms	✓		✓	✓	✓
Variants	✓	✓	✓	✓	✓
Component code configuration					✓
Formatting guidelines	✓	✓	✓	✓	✓
Dos and Don'ts		✓	✓	✓	✓
Grouping		✓	✓	✓	
Behavior	✓	✓	✓	✓	✓
Accessibility		✓	✓	✓	✓
Modifiers		✓			
Related components	✓	✓		✓	✓
Feedback	✓	✓		✓	
References		✓			

Appendix D - Interview Script

Developer interview

Om oss

Vi är…
Bakgrund inom användarcentrerad design och interaktionsdesign…
Vårt arbete handlar om att förbättra samarbetet mellan utvecklare och designers för att göra applikationerna mer konsekventa och sammanhängande.

Är det okej om vi spelar in samtalet?

Frågor

Berätta lite om dig själv och vad du gör på Saab

- Vad gör ditt team?
- Hur många är ni?

Arbetsprocess

1. Berätta om din arbetsprocess:
    - Hur får man arbetsuppgifter? Ansvarsfördelning?
    - Får du en brief och sen bara kodar iväg?
    - Har du guidelines att gå efter?
    - Om en ny funktion ska in, hur tänker man då?
2. Hur fungerar samarbetet mellan utvecklare?
    - Inom samma team och mellan team
    - Finns det plattformar för att dela och återanvända kod?
        - Repositories? Git?
        - Används någon typ av template/placeholder kod?
        - *Kan du visa / skärmdela hur det ser ut?*
3. Hur fungerar samarbetet mellan utvecklare och designers i dagsläget?
    - Vad funkar bra och mindre bra? Varför?
    - Arbetar ni med användarbehov / human factors?
4. Sitter man med ett kundprojekt, en version av programmet, eller arbetar man med alla?
    - T.ex. om en sak fixas i ett kundprojekt, delas lösningen centralt för hela systemet?
5. Idéer på hur man skulle kunna förbättra samarbetet?

Systemarkitektur

1. Programmeringsspråk?
    - Frameworks
    - Hur ofta uppdateras komponenter och kod?
        - Hur går det till?
        - Hur jobbar ni med versionshantering?
2. Hur är Backend och frontend uppdelat?
    - View-controller arkitektur?
3. Varför ser programmet ut som det gör?
    - Layout, grafisk stil
    - Är ex. en action button bara en standardknapp från ett bibliotek? Har man lagt på kod i stylesheet för att ändra dess utseende?
    - Anpassas systemens språk efter kundens inställningar/önskemål?

Brief Design System

En potentiell lösning är att implementera ett designsystem som samlar komponenter och guidelines på ett lättåtkomligt sätt, tänk Google's Material Design. Kan vara en portal där man kan kolla upp hur en knapp ska utformas, hur koden ska skrivas och hur man kan göra förändringar efter behov.

- Spontana tankar?

Co-design signup

Vill du vara med och ta fram grunden för ett designsystem på en workshop den 7:e eller 8:e april?

Appendix E - Contextual Inquiry Script

Contextual Inquiry w. Operators - Simulator

Part 1 - Observations ~ 1 hour

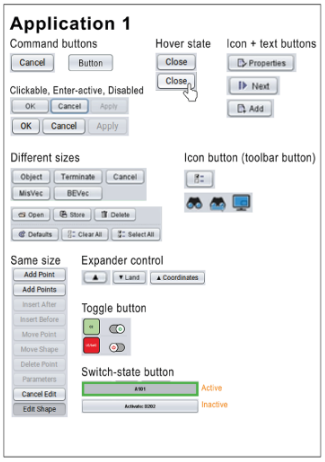
- Window Layout
- Functions
    - How often?
    - For what?
- Interaction
    - Key commands or cursor input?
    - Miss-clicks?
    - Unnatural cursor movement?
- Cooperation
    - How are they communicating with each other?
    - How are they coordinating different parts?

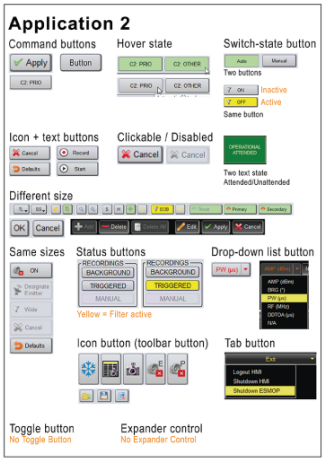
Part 2 - Interview ~ 1 hour

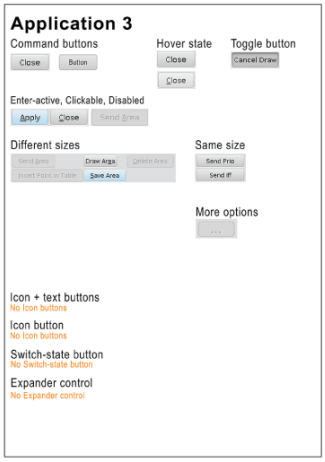
- Context
    - Lighting?
    - Noise?
    - Vibrations?
- The operator role
    - What different roles do you have as an operator?
    - What tends to be the most difficult part of the job?
    - Do you need to operate in stressful situations? Describe.
- GUI
    - Are features well prioritized based on how you use it?
    - Do you usually put windows like this? Why?
    - Is it important that windows can be moved to different locations depending on the task?
    - Bright / Dark mode or gradient adjustment?
    - What tends to be the most difficult part of the GUI to master? Why?
    - What is your favourite part of the GUI? Why?
    - What is the thing that bothers you the most? Why?
    - Comparing the interfaces:
        - Why are they different in appearance and layout?
        - Are there some things that are done better in one of them?
- Change
    - What is the sentiment around change?
    - Where could we bring the most value?

Appendix F - Interface Audit

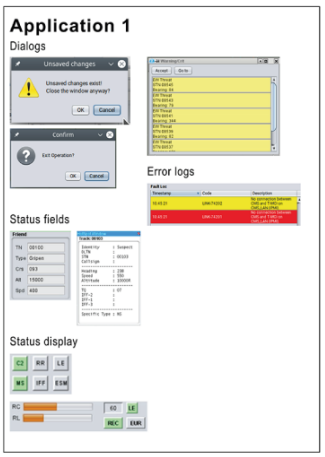
Buttons

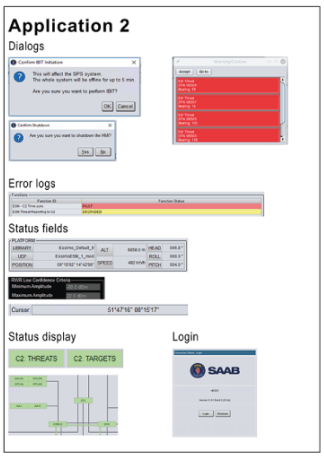


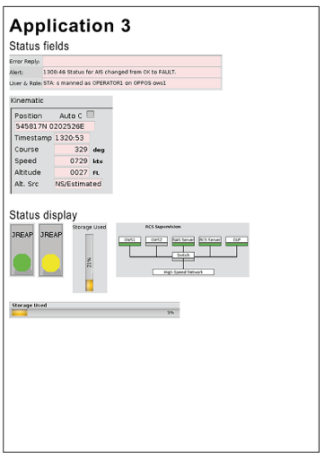




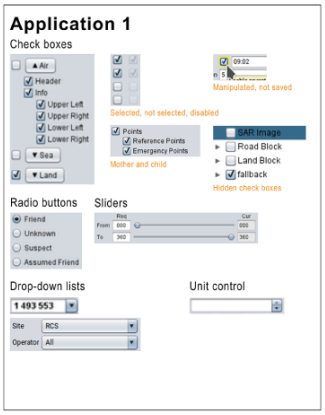
Dialogs & System status

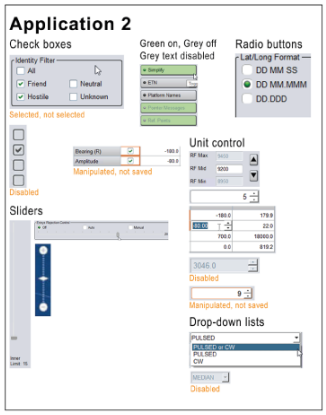




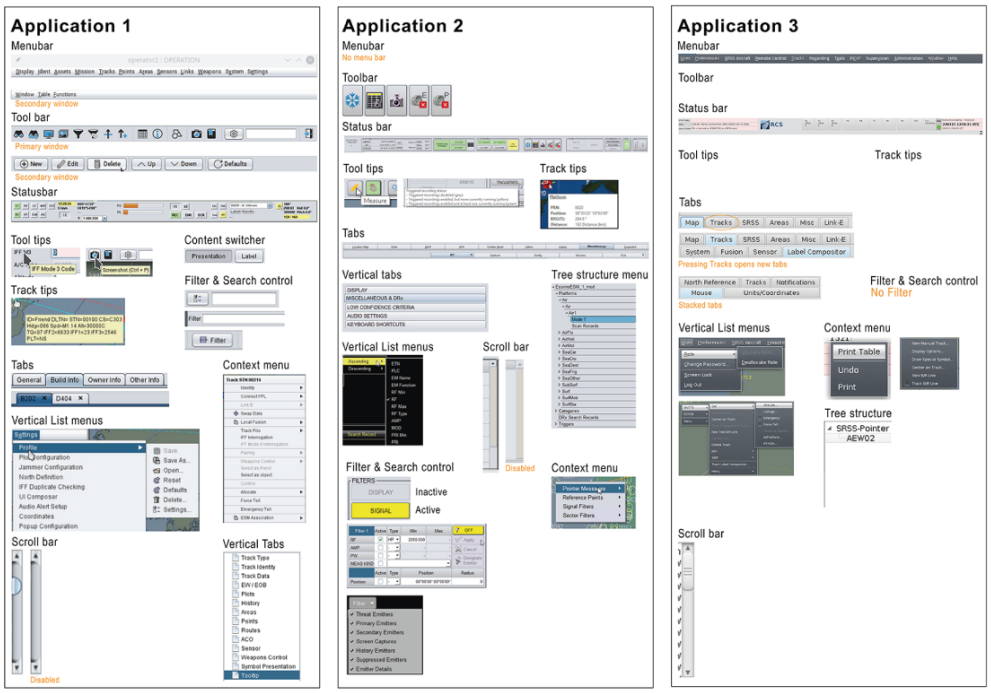


Selection control

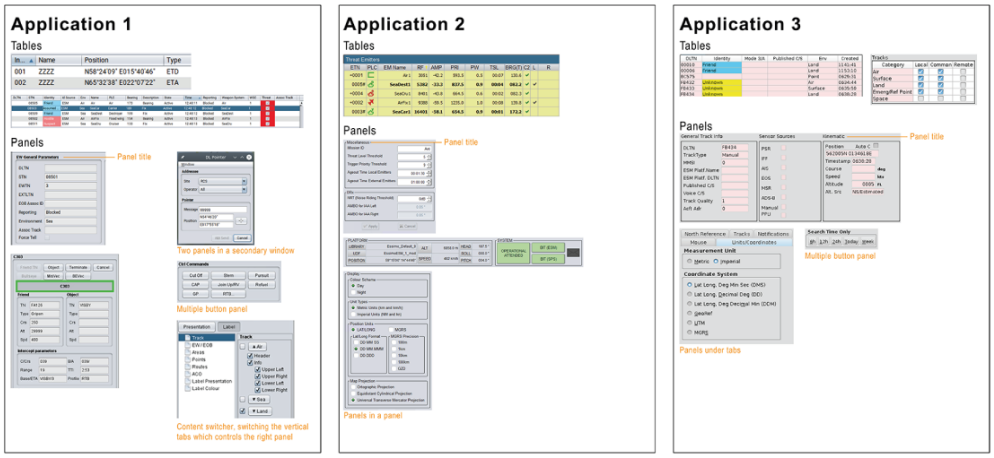




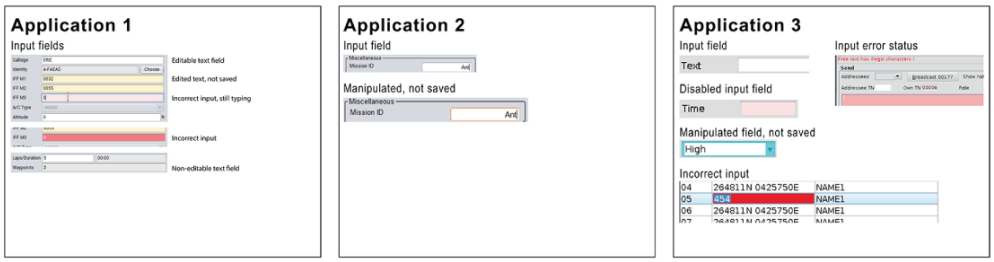
Navigation



Tables & Charts



User input



Appendix G - Workshop Structure

Workshop

with Developers

Skicka ut sensitizer:
- Skicka ut info om designsystem och länk till IBM Carbon
- Formulär där deltagarna får fylla i vad de kallar varje komponent

Syfte:
Få en bättre inblick i vad det är som behövs från oss för att skapa ett användbart designsystem och vad som krävs för att det ska gå att vidareutveckla.

Mål
- Skapa en gemensam lista över komponenternas namn
- Ta reda på vad utvecklare behöver för att kunna använda och bidra till ett designsystem.
- Ta reda på när, hur och varför ett designsystem måste tillåta flexibilitet och på vilket sätt vi kan möjliggöra det.
- Vad bör ingå i ett designsystem på SAAB?
- Kravlista för designsystem

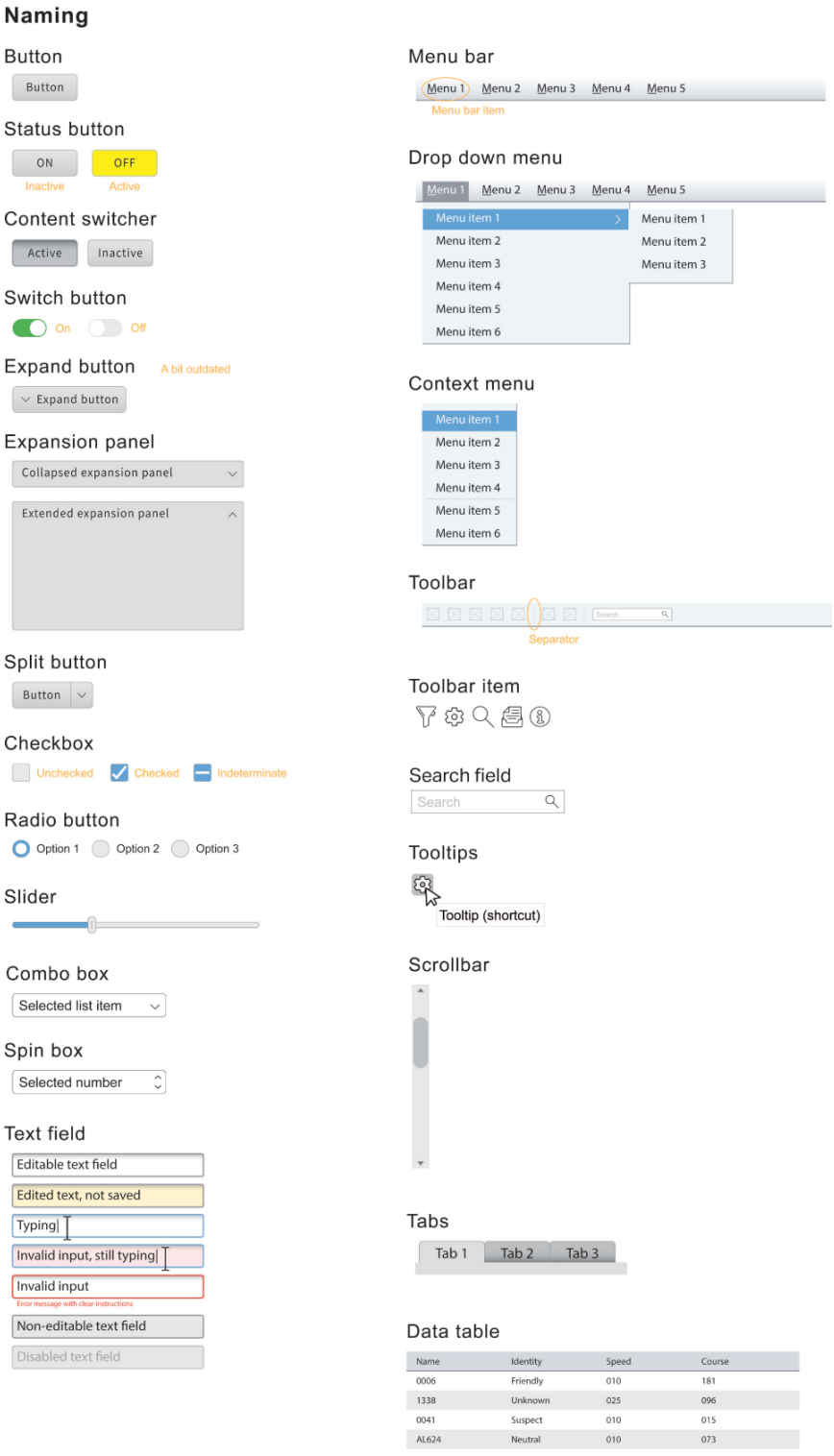
Upplägg:
1. Förklara vad ett DS är; crash course 101
2. Grundläggande struktur
3. Komponentsidans innehåll
4. Koherens och flexibilitet

Agenda:
- Intro
- Vad är ett designsystem och varför är det viktigt?
- Uppvärmning / Ice-breaker :)
- Övning 1 - Grundläggande struktur
- Paus
- Övning 2 - Komponentsida
- Övning 3 - Koherens och flexibilitet
- Sammanfattning

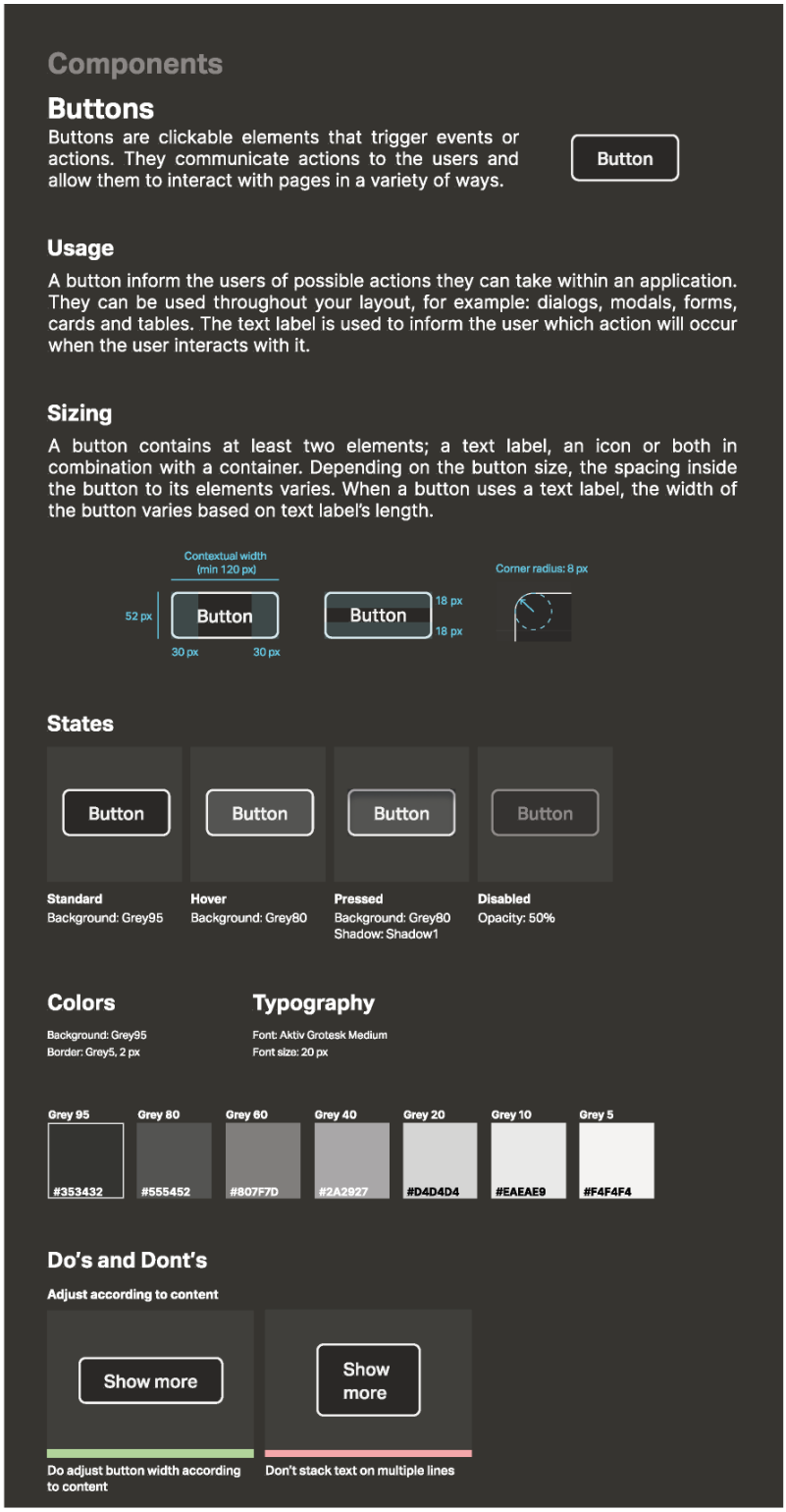
Intro och presentation:

- Vi är Mattias & Anton…
- Syftet med denna workshop…

Appendix H - Naming Components



Appendix I - Implementation Test



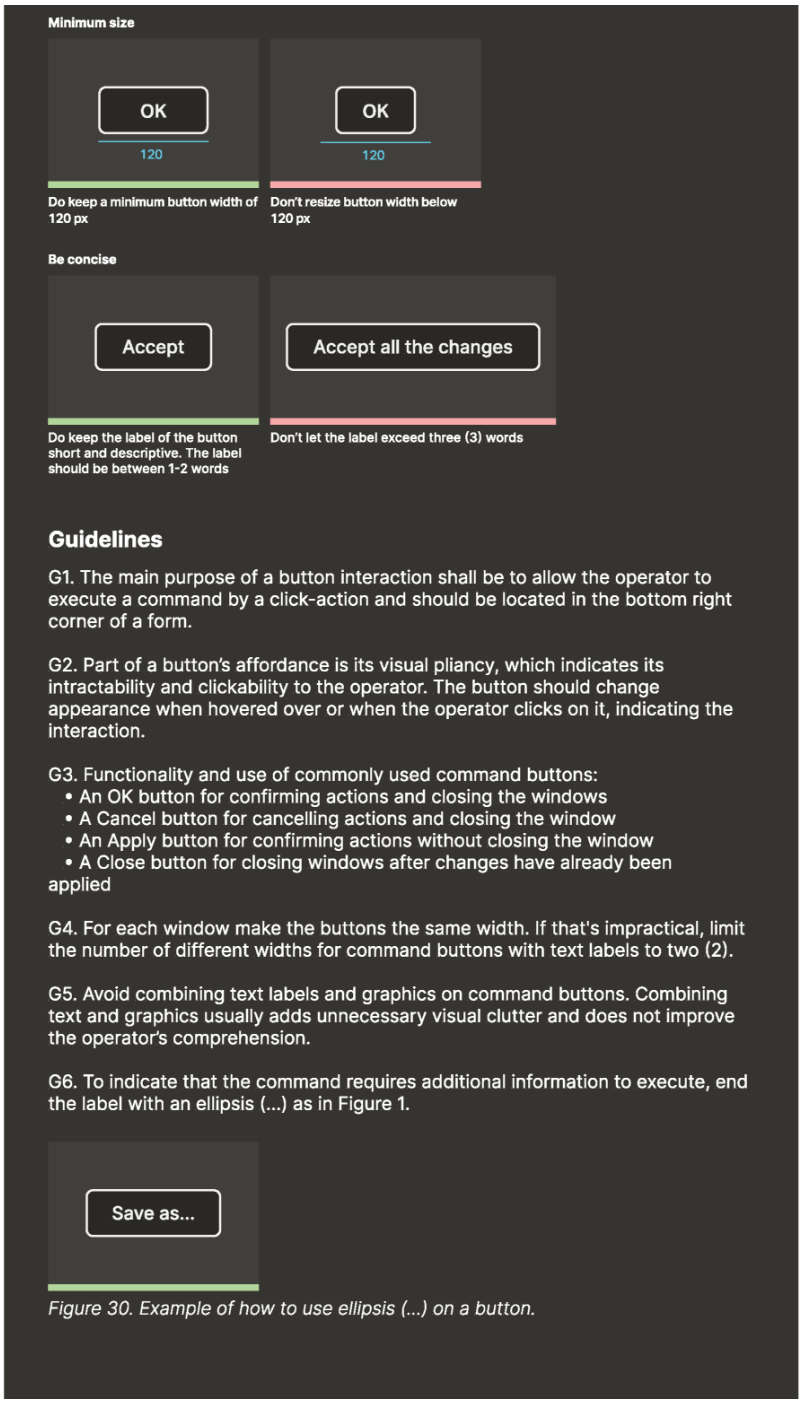
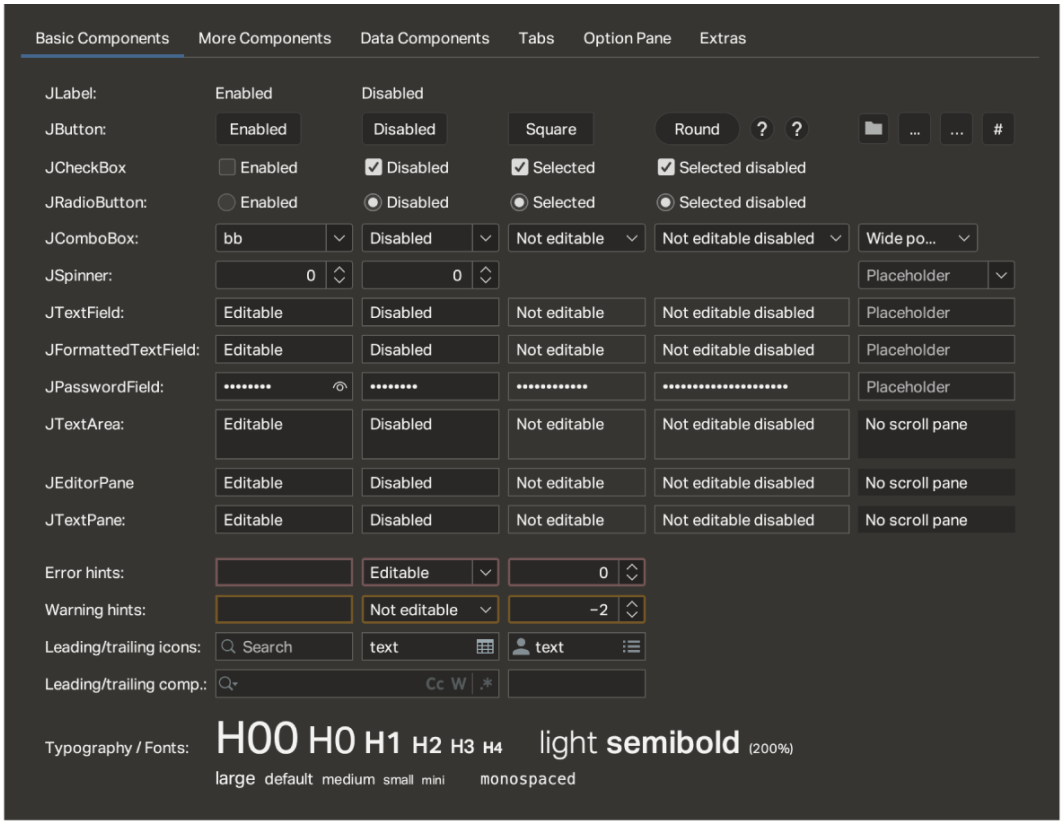
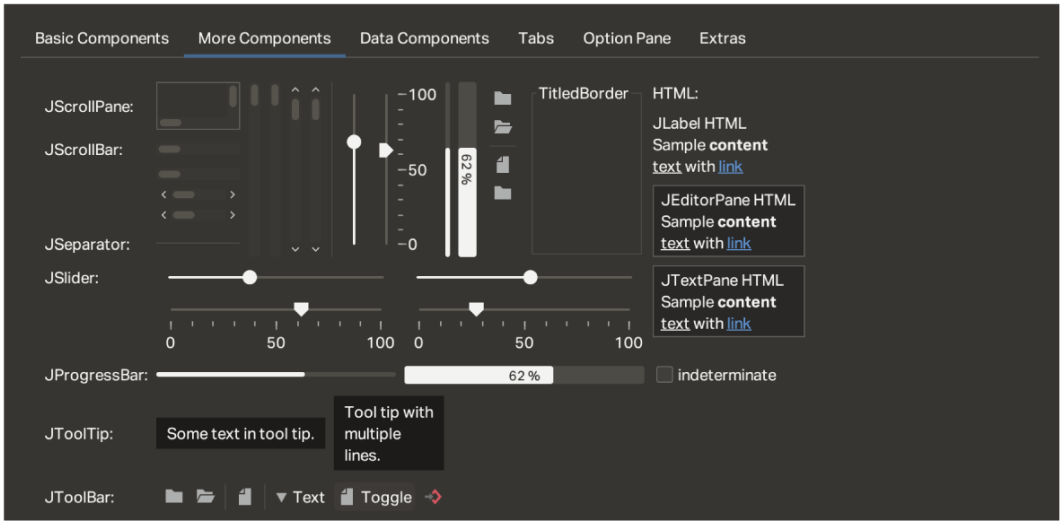
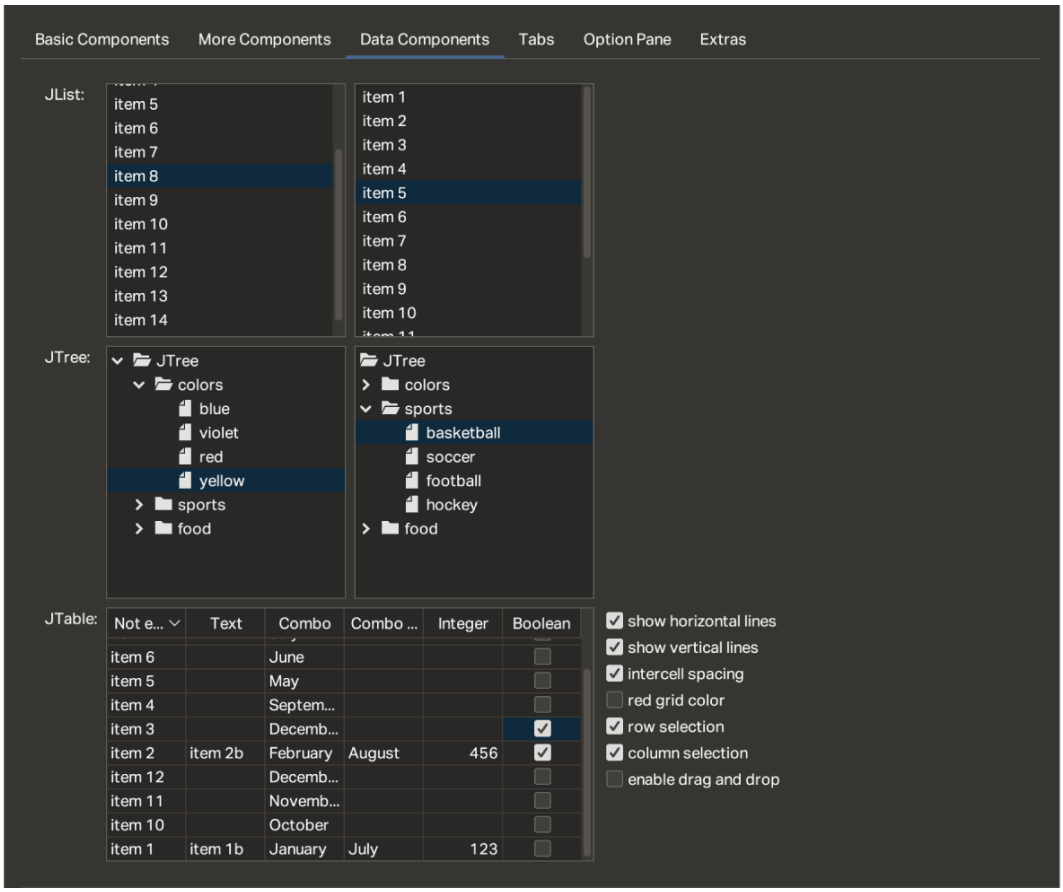


Figure 30. Example of how to use ellipsis (...) on a button.

Appendix J - PhoeniX's Components






Basic Components
More Components
Data Components
Tabs
Option Pane
Extras
JList:
item 5
item 6
item 7
item 8
item 9
item 10
item 11
item 12
item 13
item 14
item 1
item 2
item 3
item 4
item 5
item 6
item 7
item 8
item 9
item 10
JTree:
JTree
colors
blue
violet
red
yellow
sports
food
JTree
colors
sports
basketball
soccer
football
hockey
food
JTable:
Not e...
Text
Combo
Combo ...
Integer
Boolean
item 6
June
item 5
May
item 4
Septem...
item 3
Decemb...
item 2
item 2b
February
August
456
item 12
Decemb...
item 11
Novemb...
item 10
October
item 1
item 1b
January
July
123
show horizontal lines
show vertical lines
intercell spacing
red grid color
row selection
column selection
enable drag and drop

Appendix K - PhoeniX's Component Groups