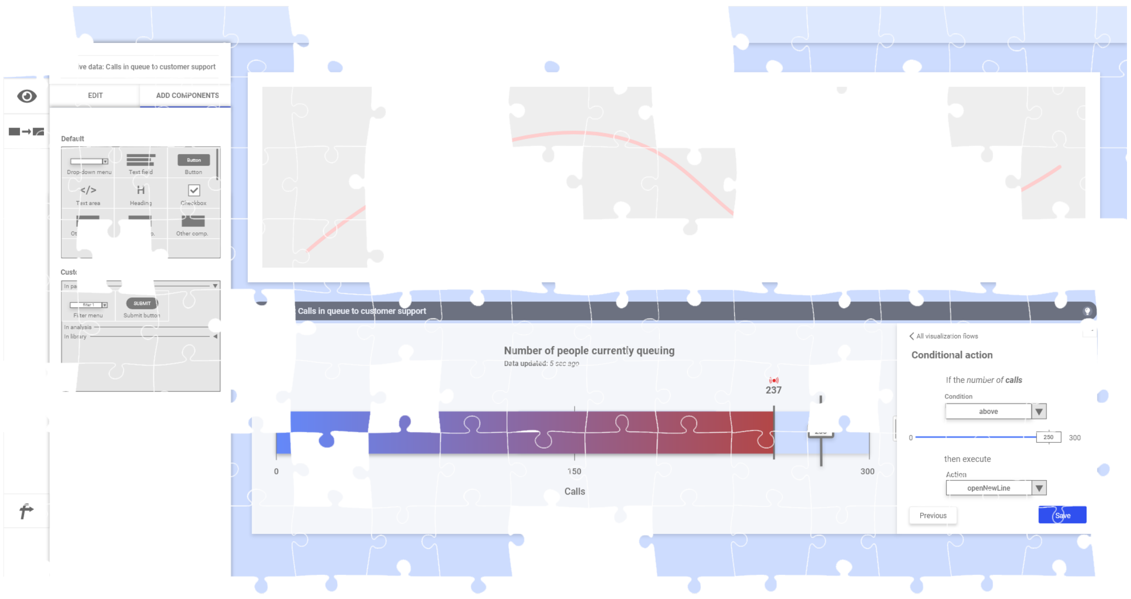


Towards the Desired Experience for Custom Applications in Analytical Software

Supporting application builders who use analytical software to create custom applications

Master's thesis in Interaction Design and Technologies

Matilda Sjöblom and Kevin Brunström

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Towards the Desired Experience for Custom Applications in Analytical Software

Supporting application builders who use analytical software to create custom applications

Matilda Sjöblom and Kevin Brunström

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Interaction Design and Technologies
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Towards the Desired Experience for Custom Applications in Analytical Software

Supporting application builders who use analytical software to create custom applications

MATILDA SJÖBLOM, KEVIN BRUNSTRÖM

Supervisor: Staffan Björk, Department of Computer Science and Engineering
Advisor: Maria Redström, The Company
Examiner: Josef Wideström , Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Division of Interaction Design and Technologies
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A print screen from the prototype styled as a puzzle.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Towards the Desired Experience for Custom Applications in Analytical Software
Supporting application builders who use analytical software to create custom applications

MATILDA SJÖBLOM, KEVIN BRUNSTRÖM

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Today, virtually all fields of industry and research use data analysis: Formula One teams, clinical research studies and many more. The ability to create customised analytical applications is an approach analytical software can employ in an attempt to met the needs of all these groups. This thesis examined how analytical software can support analytical application builders. And how the software can help them accomplish the desired user experience in an application. The research resulted in twelve formative guidelines and an interactive prototype; demonstrating the concept of analytical workflows. Theory about end user development influenced the design process. In total, the process had three iterations. During each, the prototype was used to evaluate the guidelines and vice versa. User research was mainly done through interviews, a design workshop and affinity diagrams. The intention with the guidelines and the prototype is to support future development of analytical software.

Keywords: analytical software, application building, information visualisation, end-user development

Acknowledgements

We would like to thank our supervisor, Staffan Björk, for all guidance and support throughout this thesis. We would also like to express our appreciation to our Company mentors Maria and Gustaf.

Contents

List of Figures

Glossary

Analysis A file that is created by The Product. An analysis may consist of one or more pages. 49

Application An application that is developed and run inside The Product. xvii, 43, 44, 48, 56, 65, 68, 73, 75, 86

BI Business Intelligence. 12

EUD End-User Development. 16, 41, 44, 56, 64

Page A page inside an analysis. A page may consist of zero or more visualisations, texts, components, images. 39, 43, 52, 65, 68

The Company A Gothenburg based company where this thesis is conducted. 5, 6, 39, 44, 52, 56, 59, 65, 72, 74, 84, 124, VII

The Product The analysis platform developed and owned by The Company. xiii, xv, 5, 6, 7, 37, 39, 41, 44, 45, 48, 52, 53, 56, 57, 58, 60, 65, 69, 70, 72, 74, 75, 83, 84, 86, 91, 108, 122, 123, 124

Workflow A task in an application, performed by the user. 39, 56

WYSIWYG A What You See Is What You Get interface style. 63, 76, 77, 81, 85

1

Introduction

In 2017, The Economist stated that data replaced oil as the planet's most valuable resource [1]. While oil and data's inherent properties do not have much in common, they share the fact that there is value in simply owning and analysing them. Still, the availability of oil is decreasing, and organisations have more data than ever before. To find valuable information in these vast quantities of digital information, the usage of analytical software is a must.

The exponential growth in data analysis [1], particularly in big data analysis, echoes the consensus about data analysis as valuable activity. In essence, Big Data is both a research field - and a general concept - about the organisation, handling and sense-making of the large data sets many organisations accumulate nowadays [2]. A survey by IDG (n=800) showed that 78 % of IT decision-makers, i.e. project managers and similar, strongly agree that big data analysis will be beneficial for their organisations [3]. Further, a qualified majority of the respondents believed findings from data analysis could fundamentally change how their companies operate in the coming years.

Modern data analytics does not have a single point of origin, it builds upon knowledge from several domains. The usage of staticstics can be traced all the way back to the construction of the pyramids [4]. Analytical then continued to evolve throughout history. A giant leap forward came in the middle of the 20th century with the advent of computers systems. This innovation prompted pushed the development development of many modern analytical domains. As computers gradually became more available, a market for commercial analytical software spawned. Though developing analytical software is no easy feat. First of all, analytics is by itself a complex and diverse field [5]. Moreover, the data analysts who use analytical software are present in all major industry sectors and fields of research. And while there is universal analytical knowledge, each domain also comes with its niches. Analysts also have diverse skills and interests, some can have extensive programming knowledge, while others are novices. Thus, designers of analytical software face a diverse set of users. At a glance, 'the user' is someone who wants to modify, understand and visualise data of any nature, and in any quantity. Catering to such a user might be a slightly complex design challenge.

One way vendors of analytical software can support the users' many needs is by offering software that supports the creation of custom analysis applications. This approach allows users to create an environment where they can uncover insights from data analyses and use them when making decisions through the applications GUI. External systems can also be connected to an application, giving it the possibility to become a digital hub within an organisation. In a sense, the analytical software provides a sandbox. A place where the user is responsible for building what she desires. From the user's perspective, the vendor becomes someone who primarily supplies the required building blocks and tools. Other sectors catering to a diverse crowd also use this approach. Two examples are: the game industry, with game engines like Unity [6], and web development with applications like Wix [7].

With the shift in construction responsibility, the user experience during application construction becomes key. Aiming to give application builders the experience they desire comes with many new challenges. Firstly, as they are not the equivalence of pure end-users, they might not get as much attention as the pure-end users do. Secondly, most data analysts are not professional software developers. How can analytical software help them construct the applications they desire? Besides them, some analysts use programming and others get help from professional developers. These two groups also desire a certain user experience while building custom applications. And who to consider does not end there: an application also has users. Giving them useful applications is in the interest of platform providers and application developers.

This thesis is carried out in cooperation with a company that provides data analysis software with the ability to create custom applications. The Company will be anonymous in this thesis, they will simply be referred to as The Company. Hence, references regarding them or their product will not be included. The background chapter provides more information about them and their product. The authors of this thesis are two students attending the master's program Interaction Design and Technologies at Chalmers University of Technology. Both are interested in user experience and data science, which this thesis combines.

1.1 Purpose

This thesis mainly considers three parties: *(The) Product developers*, *application builders* who work at The Company's customers, and *application users*, who also are known as end-users. Their relationship can be regarded as a hierarchy, where the work an upper level does affect lower level (s). At the top are Product developers who develop the software used by Application builders. As a group, the application builders have a varying professional software-development and user-experience knowledge. Some could be complete novices within both fields. Still, the application's users and its stakeholders both value usability. Hence, the goal is to understand what support each developer group needs to support its target audience. Ideally, the outcome of this thesis increases the satisfaction for everyone in the hierarchy. In conclusion, the thesis aims to answer the following research question:

What should be considered when supporting application builders who aim to fulfil their end-users' desired experience in analytic software?

To answer this research question, a suggested set of design guidelines will be created. These guidelines will be based on adequate research in the area, best practices in the field, and our own experiences when developing a prototype for application builders.

1.2 Delimitations

As mentioned, this thesis focuses on the user experience during application building in analytical software. The aim is to provide knowledge that can be of use for anyone who develops analytical software with this feature. Company competitors will be examined, but The Product will be the only software we use to evaluate our design. Working with multiple environments is beyond the scope of this thesis.

Security, software integrity, application programming interface (API) design or other software properties are excluded from the scope of this thesis. Furthermore, issues may arise when vendors enable custom ad-hoc applications. A possible issue is compromised security, since the platform owner cannot guarantee the developers' experience regarding software integrity. Even if 'to feel safe' is a fundamental human need, and the lack of it may damage the user experience, security in applications does not lie within the scope of this thesis.

Developing advanced graphical visualisations may require a significant amount of hardware resources, and an unresponsive program with a low frame rate may negatively inflict the user experience. This could be a problem when developing applications on top of an analytical platform, but managing hardware resources will not be covered in this thesis.

1.3 Ethical Issues

As mentioned, this thesis and The Company's platform connect several parties. The hierarchy and relation between these groups have to be considered to uncover possible ethical issues. To start with, application builders at customer companies can become information gatekeepers, like newspaper editors. Both parties in a sense have a medium and a 'story' to tell, and it is always possible to tell a story in a thousand different ways. Furthermore, seemingly trivial design decisions by application builders, such as colour and placement of elements, could have a profound effect on what conclusions application users draw.

In close relationship with application design is data visualisation. The choice of visualisation can affect application users due to how they decode the visual structure. Even the tweaking of individual diagrams components, such as axis values, can affect.

2 Background

This chapter introduces related work. It also contains an overview of the company where this thesis will be carried out, and the product of this company. The Company has requested to be anonymous, and are referred to as The Company. Their product is referred to as The Product. Due to the anonymity, no reference regarding The Company will be included. Competing companies whose software is useful for this thesis will also be presented.

2.1 The Company

The Company where this thesis will be conducted is a software company founded in 1997. They offers a range of specialised software products data analysis and its related fields. They have offices in North America, Europe, Asia, Africa and South America, and have approximately 4,200 employees. This thesis is done with support from their office in Gothenburg.

2.2 The Product

In the mid-1990s, The Company launched The Product, and it has been available on the market since then. It is one of the products offered by The Company for data analysis. There are primarily two types of customers who use the product, those who use it only within the organisation, and those who use it for business with external customers (but they could also use it internally for other tasks). Some use it for both purposes. In essence, The Product is used for data analytics, and it has also built-in support for data visualisations. The last major update of The Product was in 2018, and included natural language query-based searches, AI-driven recommendations, and model-based processing. Another key feature is The Product's extensive support for the creation of custom applications. Users of the software are present in many data-processing fields of work and research: medicine, trade and transport are some examples of these sectors. In addition, even customers in the same sector use it differently due to the often complex use cases.

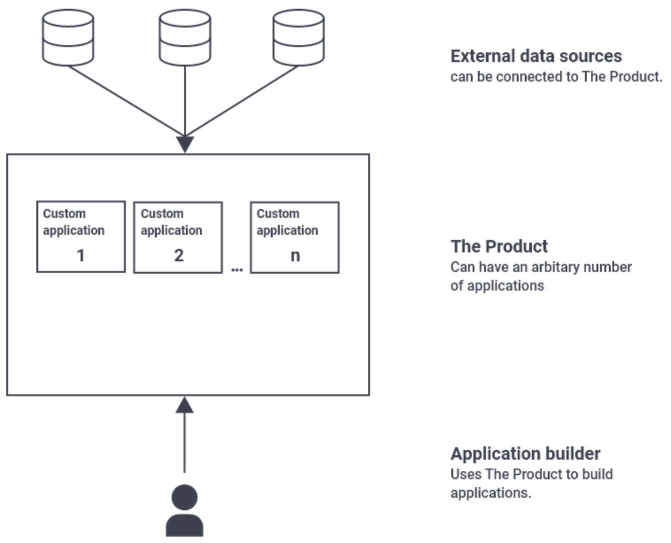

External data sources
can be connected to The Product.
Custom application 1
Custom application 2
...
Custom application n
The Product
Can have an arbitary number of applications
Application builder
Uses The Product to build applications.

Figure 2.1: A simplified visualisation of the relationship applications have with The Product.

2.2.1 Developing Custom Applications in The Product

The Company's product is, as mentioned, designed to support extensive customisation by users. One way The Product supports customisation is by offering built-in functionality for creating custom applications. Developers can build custom applications for niched requirements. These applications exist within The Product environment (2.1), it is still possible to connect them with external systems. An application could also be integrated within a separate system. For example, an application can be embedded in a web page. The Product would then power the application 'under the hood', even though it appears powered by the web page.

Another way of customising The Product is to utilise the ''Visualization Mods" framework. Visualization mods enable users of The Product to create their own visualisation using web technologies like JavaScript or TypeScript. The newly created visualisation behaves like a built-in visualisation of The Product, as it functions within it and is powered by its framework. These visualisation mods can later be shared and distributed via cloud-driven channels provided by The Company.

2.3 The Product's Users

Knowing who the user is and what she needs can be a challenge for any organisation, and it is particularly challenging for The Company. They often have many indirect customers. It could be people who come into contact with the software, as colleagues use it or similar. Users at each layer are also likely to have different unique needs. Figure 2.2 shows the most common user groups and their relationship with The Product, according to The Company. This representation also excludes many relationships with The Product, e.g. The Company's employees who use The Product when they help customers, or the external auditors who inspect some customer applications. An individual could also be both a user and a builder, like a smartphone application developer who builds an application for personal usage.

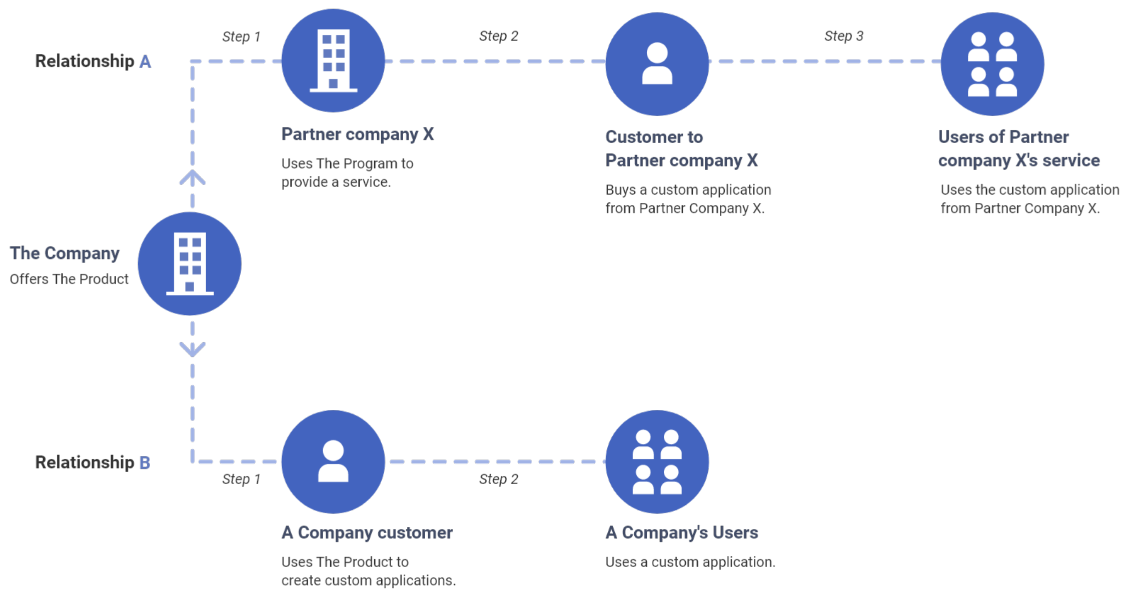

Relationship A
Step 1
Step 2
Step 3
Partner company X
Uses The Program to provide a service.
Customer to Partner company X
Buys a custom application from Partner Company X.
Users of Partner company X's service
Uses the custom application from Partner Company X.
The Company
Offers The Product
Relationship B
Step 1
Step 2
A Company customer
Uses The Product to create custom applications.
A Company's Users
Uses a custom application.

Figure 2.2: An overview of common user groups and their relationship with The Product.

In **relationship A** a partner company uses The Product to sell their customers a custom application. Usually the customer is an organisation that eventually has an intended group of users.

In **relationship B** a customer uses The Product to build custom applications. The customer could be a single individual or an organisation that plans to let multiple employees use it.

The relationship in focus of this thesis is relationship **B**.

2.4 Competitors

Numerous companies compete with The Company, below are those who offer their users some kind of extension development. Besides this selection, other competitors include: Birst [8] and Microsoft BI [9].

2.4.1 Tableau

Tableau is a software company which develops products that support data analytics and visualisations [10]. Tableau offers a service they call Developer Tools which, among other several capabilities, enables the user to extend the existing platform by adding third-party functionalities [11]. This is available via the "Extensions API", which makes it possible for the user to, for example, modify the source data directly in a visualisation. The dashboard extensions are web applications that have two-way communication with the Tableau software [12]. To give users who create extensions with a pleasant user experience, Tableau provides design guidelines [13]. They are also offering samples of extensions and ”Tableau UI”, which is pre-made React components that have implemented the look and feel of Tableau [14]. React components can be described as independent, reusable pieces of code [15]. An overview of Tableau Desktop can be seen in **fig** 2.3

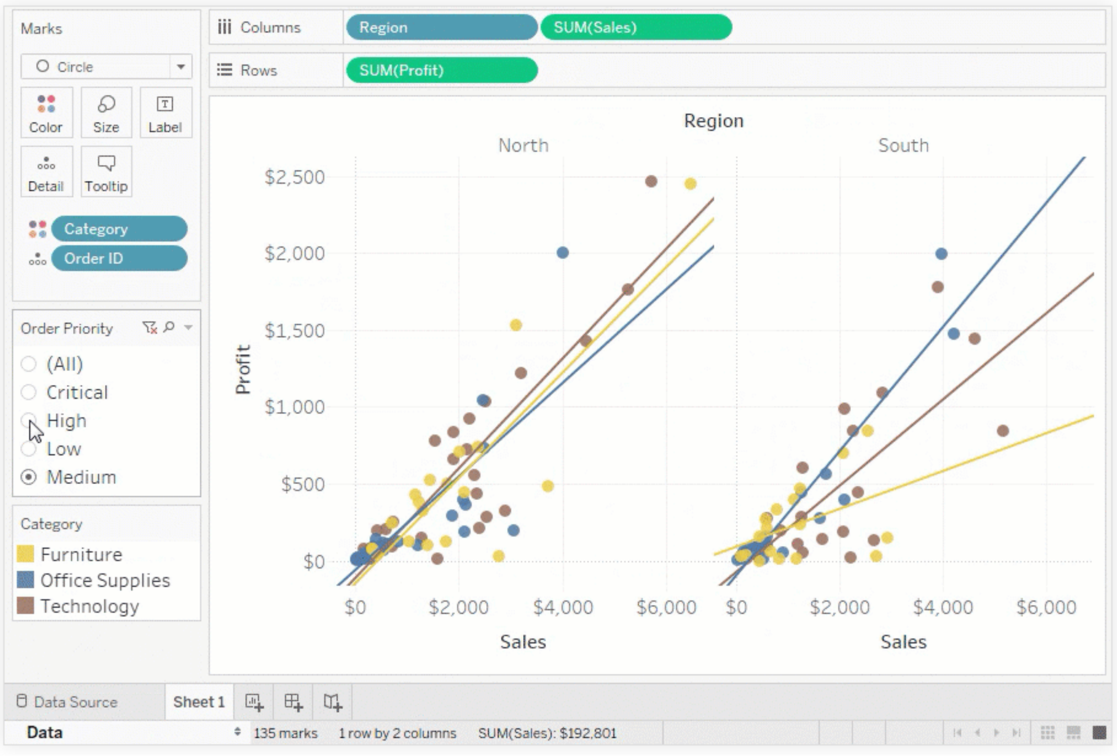


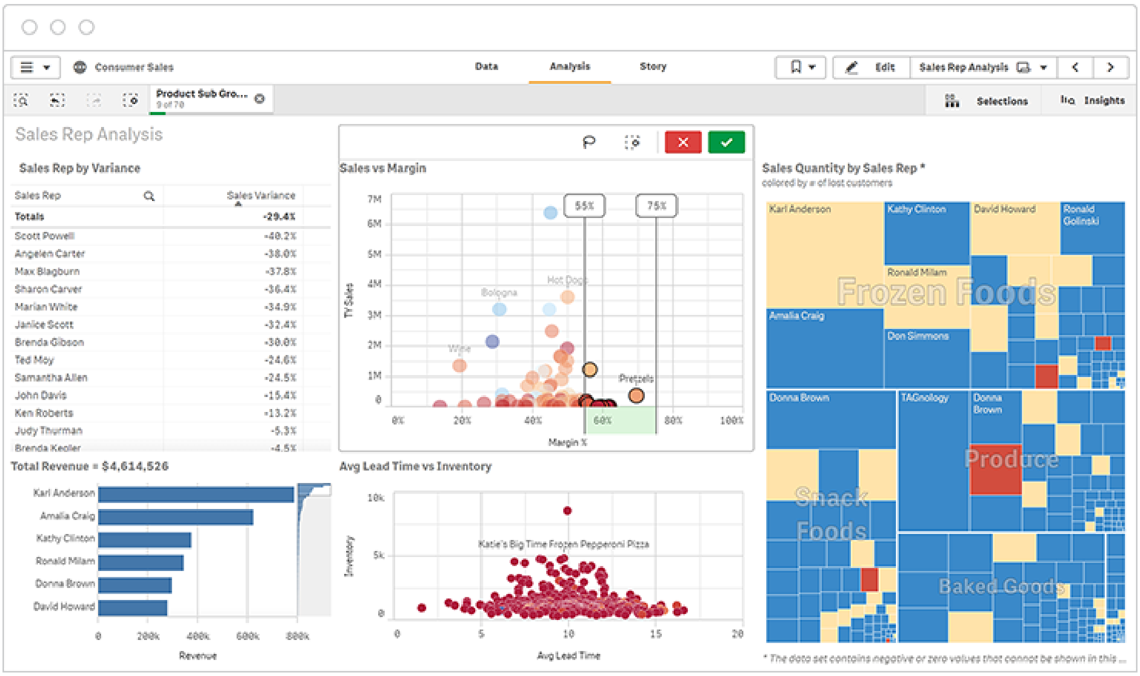
Figure 2.3: A printscreen of Tableau Desktop [10].

2.4.2 Qlik

A software company that provides different data analytics tools for business intelligence [16]. In their analytics platform Qlik Sense, the users can extend their dashboard applications via “Embedded analytics”, whose key concept is to access data from daily workflows without any disruptions. This is possible by providing an application programming interface, API, for their users [17]. An overview of Qlik Sense can be seen in **fig** 2.4.

2.4.3 Microstrategy

Business intelligence software suitable for both mobile and desktop platforms, is what MicroStragies provides its customers [18]. They allow customisation and extensions of their software through several different software development kits, SDKs, and REST APIs [19]. A printscreen of MicroStrategy’s software MicroStrategy Web can be seen in **fig** 2.5.



Sales Rep by Variance

Sales Rep	Sales Variance
Totals	-29.4%
Scott Powell	-40.2%
Angelen Carter	-38.0%
Max Blagburn	-37.8%
Sharon Carver	-36.4%
Marian White	-34.9%
Janice Scott	-32.4%
Brenda Gibson	-30.0%
Ted Moy	-24.6%
Samantha Allen	-24.5%
John Davis	-15.4%
Ken Roberts	-13.2%
Judy Thurman	-5.3%
Brenda Keeler	-4.5%

Figure 2.4: A printscreen of Qlik Sense [16].

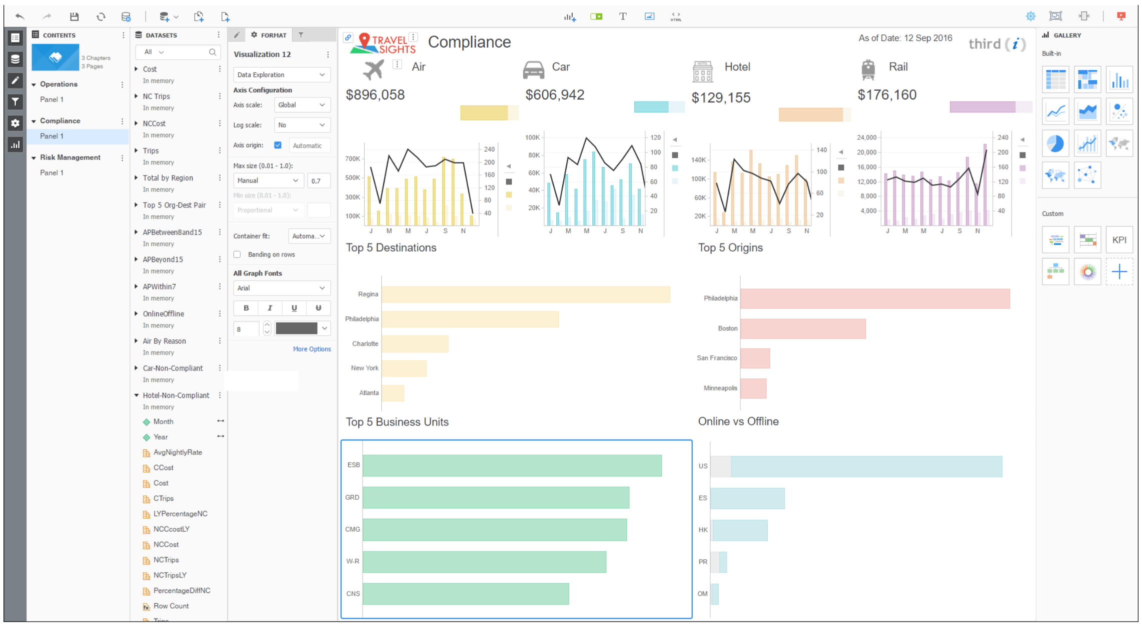


Figure 2.5: A printscreen of MicroStrategy Web [20].

3

Theory

This chapter covers theory about wicked problems, user experience, developer experience, end-user development, design systems, customizable software, and information architecture. Also included is an overview of data analytics and information visualisation, since The Product relates closely to those two fields.

3.1 Wicked Problems

Design is an ever expanding field, and it overlaps with several other domains. Design problems can be often unique and complex. The proposed solutions to such problems often combine expert knowledge from several fields. The multidisciplinary entanglement has led to difficulties in communication between designers and domain experts [21]. Designers are specialists in the area of design, whereas the scientists are experts in their domains and each domain has its separate vocabulary.

These tangled problems are commonly called Wicked problems, and they are described as "that 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" [22]. As Buchanan states, this description of wicked problems points out the relationship between determinacy and indeterminacy in design problems.

A linear model of problem solving, where the problem is first defined and then solved, can be proposed. This model implies that the problem, the design problem in this case, can be divided into phases. These problems are called determined problems. Another approach to solving design problems views all of them, except the most trivial ones, as wicked problems. This indicates that the problem has an underlying indeterminacy, which implies that "there are no definitive conditions or limits to design problems" [21].

3.2 Information Visualisation and Visual Analytics

The word visualisation has several definitions, and this thesis uses the following definition: “a graphical representation of data or concepts” [23]. As scientific field visualisation can be divided into two categories, *scientific visualisations* and *information visualisations* [5]. Scientific visualisations are “visual displays and realistic renderings of spatial data associated with scientific processes”. In other words, they are graphical representations of numerical data. The other group is information visualisations; visual representations of abstract, non-spatial data. This type of data does not define the relationship an entity has with a space, e.g. its position or shape. However, a spatial data entity, like a 2D-coordinate, acknowledges where it belongs within a coordinate system. Still, both groups have a common goal. They communicate meaning through a graphical interpretation of the underlying data [5].

Data analysis can be considered an activity, and when you look at the field from that perspective, it is about data modelling and exploration. The result from data analysis - new information - can be used in related decision-making processes. Furthermore, the two main sub-fields of data analysis are *confirmatory data analysis* (CDA) and exploratory *data analysis* (EDA) [5]. In CDA, analysts use statistical processes to evaluate hypotheses on existing data through a statistical hypothesis test. Generally, this field favours more traditional statistical tools as confidence and inference. In contrast, EDA is used when no hypothesis exists, to discover unknown patterns, and features in a data-set. This area is mostly taking advantage of visual methods to sum characteristics in data, and is also much more intractable than CDA [5]. Regarding their history, EDA is a newer field than CDA, and EDA also emerged from it. Before EDA was recognised in the world of statistics “data analysis techniques including statistics and data mining developed independently of visualisation and interaction techniques” [5].

The combination of CDA and EDA form the multidisciplinary research *visual analytics*. When researchers work with complex data sets, visual analytics offers knowledge about how visualisations and interactions can help humans understand data. Ultimately, it supports the researcher’s knowledge discovery and nurtures the possibility of data giving valuable insights [5]. In summary, visual analytics combines data analysis (visualisations) and interaction techniques. This combination together with a domain expert - who is creative - can promote a well-informed decision-making process.

Closely related to visual analysis is the area of *Business Intelligence* (BI). Business intelligence aims to support business decision through data, facts and statistics it harbours. Business intelligence is defined by Negash and Gray [24] as a system that combines:

- Data gathering

- Data storage
- Knowledge management

Gray also explains that the areas listed above are combined with analysis to evaluate complex corporate and competitive information. In the end, planners use the aggregated information as a means which support them and contribute towards the completion of their goals [24]. One could say that business intelligence is utilising visual analysis to accomplish its goals of aiding the decision process.

3.3 User Experience

In the white paper published after the Dagstuhl Seminar on Demarcating User Experience [25], the editors describe the outcome from a collected effort to define User Experience (UX). A glance at the attendance list reflects the degree of difficulty in this task. About 30 academic and industry experts from several continents were summoned. In the end, the gathering did not produce a universally accepted definition, but the paper still contains a concise description. This thesis will use it as the primary definition of User Experience:

> *"The field of UX deals with studying, designing for and evaluating the experiences that people have through the use of (or encounter with) a system." [25]*

Meanwhile, there exist an abundance of other definitions offered by organisations, researchers and industry experts [26]. Below are some additional ones:

> *UX is a momentary, primarily evaluative feeling (good-bad) while interacting with a product or service.*
>
> *– Hassenzahl (2008)*

> *The value derived from interaction(s) [or anticipated interaction(s)] with a product or service and the supporting cast in the context of use (e.g., time, location, and user disposition).*
>
> *– Sward & MacArthur (2007)*

> *All aspects of the end-user's interaction with the company, its services, and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer, without fuss or bother. Next comes simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. In order to achieve high-quality user experience in a company's offerings there must be a seamless merging of the services of multiple disciplines, including*

engineering, marketing, graphical and industrial design, and interface design.

– *Nielsen-Norman Group*

Dagstuhl concluded that the multidisciplinary usage of UX has spawned many viewpoints which all view UX differently from one another. Besides the wide-spread usage, UX knowledge often relates and overlaps, with other disciplines. In figure 3.1 Dan Saffer offers a visual representation of UX and it's 'relatives' [27]. Cooper expresses some criticism towards this "umbrella" state, as he frames it [Cooper, xxii]. In summary, he believes that the field of Interaction Design better captures the challenges of designing specific software behaviour. The effort to combine the design considerations for a physical vending machine, and a day-trading program, might fail to capture some unique aspects of each venture harbours, according to Cooper.

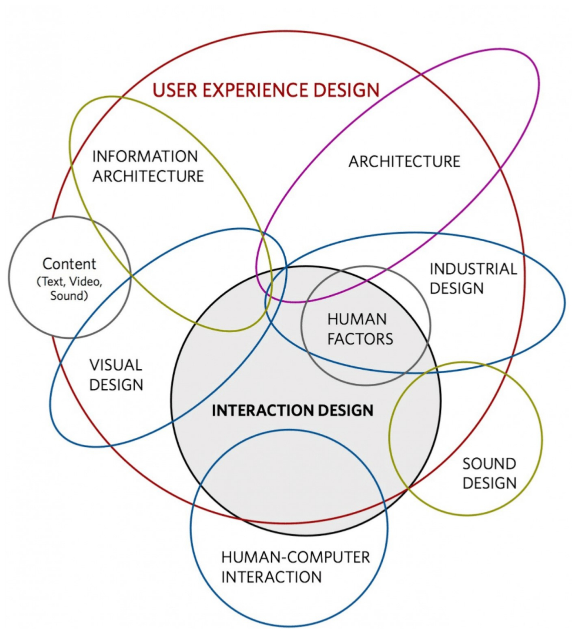


Figure 3.1: User Experience and related fields.

To create order, Dagstuhl lists three ways to relate to UX [25]. It can be viewed as a phenomenon, field of study or practice. The last perspective puts the user in the centre of a design and emphasises the need to consider user experience factors. Evaluation criteria, stakeholder valuation, representation of UX ideas are some of the many factors a designer might need to consider with-in a project, and in connection to it. As with multi-factor problems in general, a solution for one factor might negatively impact another. The situation is truly a ball of yarn that constantly seeks to entangle itself. To untangle it designers have to use their expertise and tools to identify the key criteria that - during that moment - contribute the most towards a successful (UX) design.

3.3.1 Developer Experience

Developer experience, DX, is highly influenced by the concepts of UX. According to Münch and Fagerholm, DX differs from UX in a way that it "abstracts from the broad properties of human characteristics without ignoring either" [28]. DX attempts to satisfy the needs of users that also act as developers. An important notice is that UX considers the context of use, while DX, however, considers the context of development. In this context, developers are considered people engaged in the activity of developing software [28].

The concept of the mind is often divided into three parts by psychologists; cognition, affect and conation. DX covers the affective factors of experiences, which are shown to be important, as the emotions of developers influence their programming tasks [29]. Moreover, the conative and cognitive elements of the mind are also covered. The cognitive part covers use of skill and certain techniques in the development process, while the conative part considers the intrinsic feeling about the task itself [28]. These three elements, combined with the social and technical environment, forms the framework of DX. This framework can be seen in fig 3.2.

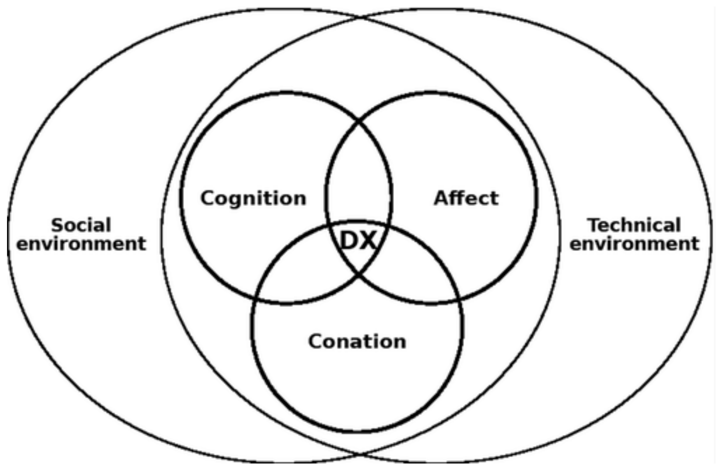


Figure 3.2: A concept figure of developer experience [29].

In Kuusinen's paper, she finds that DX regarding graphical user interface (GUI) development is highly influenced by the developer's intrinsic motivation and feeling of experienced flow [29]. Flow has been described by Csikszentmihalyi as a state where a person is completely absorbed and focused on its current task, a feeling of being unimpeded [30]. To reach a state of flow, it is important to understand intrinsic and extrinsic motivations. Extrinsic motivation is derived from external rewards that are the result of accomplishing the task. If this reward is removed, the task is likely to stop being executed. In contrast, intrinsic motivation is derived from the inherent satisfaction of performing the task, to just do the task for its own sake. The flow state is possible to reach if the task is performed with, at least partly, intrinsic motivations. With that said, a task can be performed with both extrinsic and intrinsic motivations simultaneously [30].

To reach the flow state, Mihaly Csikszentmihalyi means that the task must fulfil these three conditions [31]:

- The goals have to be clearly stated. If the goals are clear, the person performing the task can reach a structured consciousness and focus the psychic energy.
- There must be a balance of difficulty. The person has to feel capable of accomplish what is in front of them without it being too easy.
- Immediate feedback or response must be present. This could be intrinsic rewards as well as external feedback.

Kuusinen points out that the most commonly stated motivation of developers is “The work itself”, a seemingly extrinsic motivator. What is inadequately documented though, is what aspects of the work that are motivational. It can therefore not be completely discarded that there is something about the work that is intrinsically motivated. At last, she suggests for improving DX, the focus should lie on getting the developers interest, offering rewards throughout the process of creating GUIs and to get a balance in the perceived challenge [29].

3.4 End-User Development

End-user development (EUD) has a working definition as "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artefact" [32]. In this field, as the definition states, the "users of software systems" are assumed to have limited or no experience as software developers.

The idea behind EUD is that end-users know their domain best. This domain knowledge may be too difficult to acquire or to understand enough for a software developer to meet the requirements of the desired product. In the end, the final product may be either too general to fit the needs of all users, or too narrow to fit the needs of most users. Therefore, enabling users to extend their product themselves can be beneficial for everyone. Use of this can be seen in e-mail inboxes with filtering features, ad-blocks in web browsers and setting up spreadsheet calculations. It is also widely used in game development environments, with the emergence of game engines [33]. The spectrum of computer users in the context of EUD is broad. This is visualised in fig 3.3, to the right there are the users that use computers for developing software, and to the left the ones who are only using computers for running existing software.

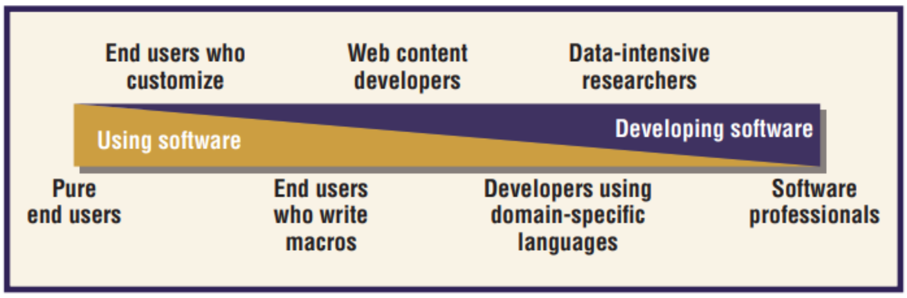


Figure 3.3: The spectrum of users from a EUD perspective. The line between "pure end-users" and users who develop own software has become blurred, which this figure illustrates. [34]

Designing and creating software is a complex task. In EUD, the required knowledge tend to be distributed among several specialised individuals in a team [34]. Some individuals may be more capable and interested in creating software, while others are more specialised in the domain. Hence, studies show that EUD as an activity is facilitated by allowing users to work together as a group.

Besides working together, EUD tools have to balance the appeared difficulty for novice users while still accommodating the more experienced ones [33]. Novice users have to be instantly gratified when completing a task, while experienced users can create the level of sophistication and completeness they require. A more elaborate description of this balance can be found in 3.4.1.

3.4.1 The Gentle Slope of Complexity

Any mechanism in a software system has an inherent degree of complexity that users must understand before they can use it. The gentle slope of complexity is a theory about how a system's total complexity can be distributed efficiently into different mechanisms or tools [35]. The definition of a tool depends on the nature of the system: it could be paintbrush in a drawing program or the possibility to attach custom scripts to components in a graphical user interface (GUI). The goal with the distribution is to have a system with tools that together strengthen each other's degree of adaptability by forming a 'gentle slope' as seen in figure 3.4.

Ludwig et al. wrote that when novice application builders create applications for the first time they "begin to climb the tailorability mountain" [36]. As seen in figure 3.4 the ability to tailor increases as builders climb higher and learn new tools. Meanwhile, it is possible that some builders have existing knowledge and therefore climb the first bit quickly - though it cannot be assumed. When the threshold is to step, builders could stop learning. As the left diagram in figure 3.4 shows, the threshold between drag-and-drop tools to scripting might introduce too much for some users to overcome.

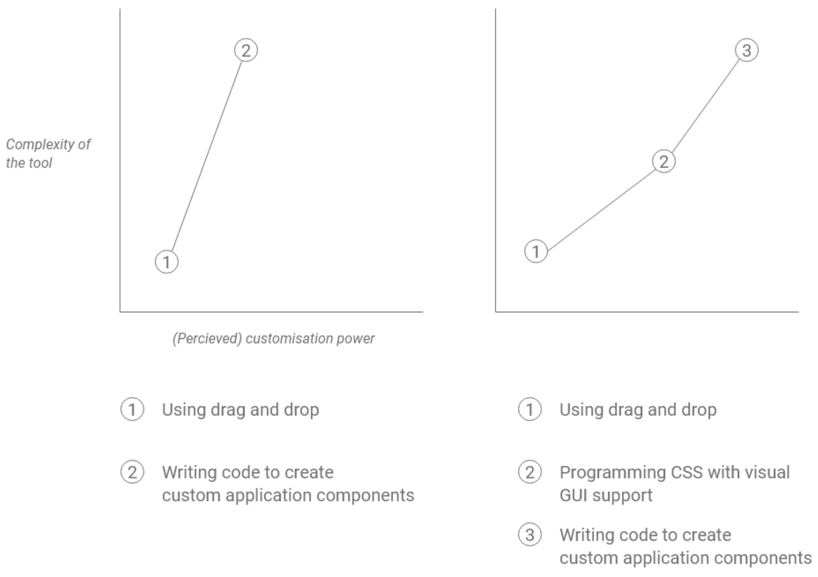


Figure 3.4: How partitioning of software functions can lower the slope inclination.

The transition can be eased by dividing complex functionality into tools. The right diagram in figure 3.4 shows a simplified example of how more tools can lower the steepness. Learning becomes even easier if knowledge from a previous tool can be used to understand the next one.

Any mechanism in a software system has an inherent degree of complexity that users must understand before they can use it. The gentle slope of complexity is a theory about how a system's total complexity can be distributed efficiently into different mechanisms or tools. The purpose with the distribution is to offer users understanding The definition of a tool depends on the nature of the system: it could be paintbrush in a drawing program or the possibility to attach custom scripts to components in a GUI.

3.5 Design Systems

New software development processes, primarily component-based architecture and agile development, have amplified the need for design systems [37] . Khalmatova defines them in the following way:

> *"...a set of interconnected patterns and shared practices coherently organised to serve the purpose of a digital product." [38]*

Vesselov and Davis have a similar definition, and they emphasise the "shared practices" and "principles" that the system builds upon [37]. Regardless of variations in how it is defined, design systems strive to bring order and consistency to software design.

Having “one source” for design information also benefits others than designers. Software developers, marketers, product owners and more all benefit from using a single source of truth, as Kholmatova frames it [38]. Having to constantly recreate material for others is time-consuming for designers and developers parties. From a designer’s perspective, it functions as a FAQ (frequently asked questions). In summary, the system contributes towards a better experience for the most important group - users. When a design system is woven into a program on all levels, users experience increased efficiency. Besides swiftness, users can also think of the software as harmonious and meaningful to use [38].

3.5.1 Designing a Design System

Another question is how a design system should be organised, and what should it should contain. This question does not have a single answer in the form of a checkbox list. The orchestration depends on who the intended users are, and the intended use is. Vesselov and Davis, each suggest a high-level categories for content structure. Kholmatova refrains from offering a similar suggestion. According to her, designers should instead consider three system characteristics, by considering where the system will position itself on the spectrum of each trait.

Kholmatova

Rules

Strict ——— Lose

Parts

Modular ——— Integrated

Organisation

Centralised ——— Distributed

Vesslov and Davis

Layout, Styles, Components, Region, Content and Usability

Vesselov and Davis write the categories are connected, and they should be used together. For example, All groups should consider Usability, and Components need to follow the guidelines Layout and Style stipulate.

3.5.2 Existing Design Systems

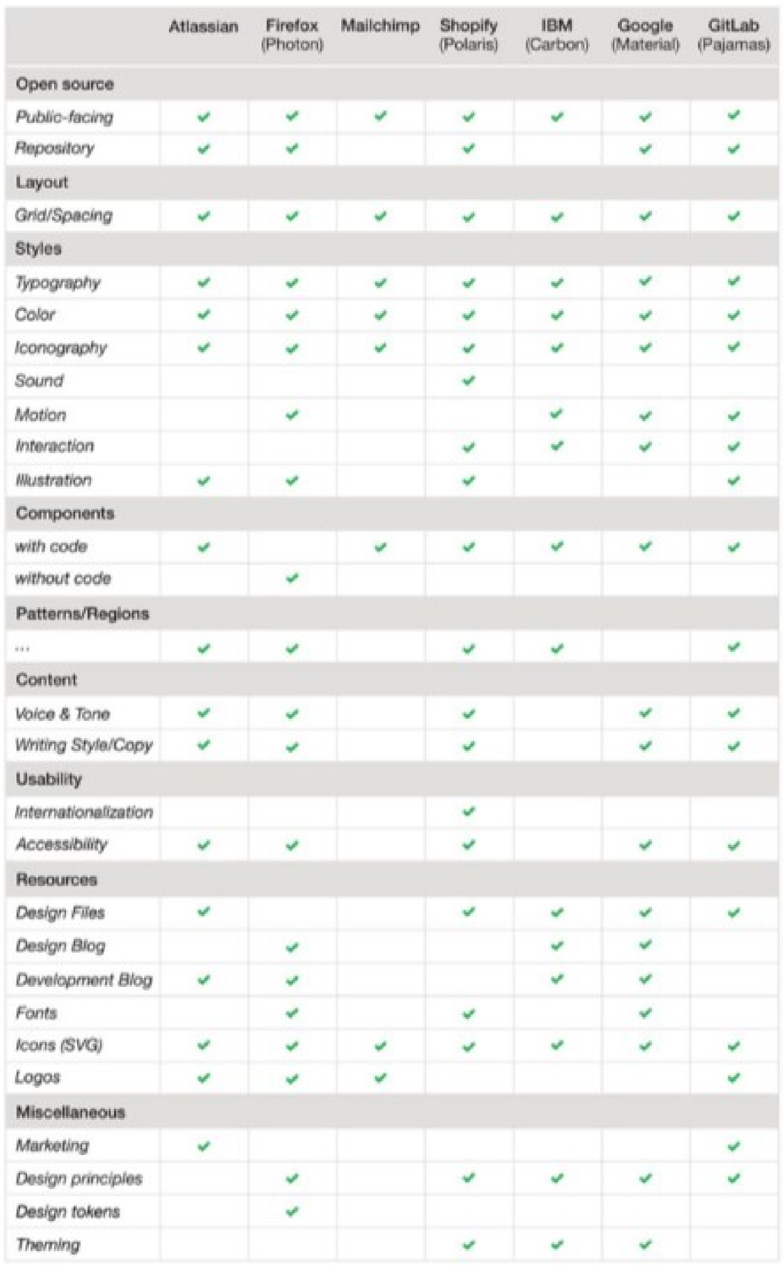
To understand how this translates into industry practice, the principles can be viewed in comparison with existing guidelines offered by software companies. Adobe and Microsoft are two prominent actors in the software industry, and each have publicly available a design system. Figure 3.5 provides an overview of content in some additional design systems, many more exist [39]. These corporations have the financial, intellectual and manpower needed for developing comprehensive guidelines. Their level of resources means that they are able to contribute with new design innovations that relate to Design Systems. New design patterns or recommendations for system language are some examples of new design content In summary, the development of design systems is thus shaped by both academia and industry.

Adobe Spectrum In 2018 Adobe launched their design system Spectrum [40], it is published as a website [41]. It has the following sections:

- **Spectrum (entry page)** Core principles and new content
- **Foundation** An extensive list that explains a range of principles and design elements. Color, typography, International design, to mention a few.
- **Content** Language recommendations
- **Components** Graphical elements and how they collaborate. Button design, systems status etc.
- **Patterns** Descriptions of Adobe's fundamental graphical components: cards, tables and headers, and their respective sub-components.
- **Tools and resources** Fonts and other premade design material.

Microsoft's Fluid Design Microsoft's Design System is called Fluid Design [42]. To cater for different software-related roles, Microsoft offers information about the system in many formats. The design system is presented as a resource for the construction of Microsoft applications. In contrast to Adobe, Microsoft also includes technical information with each design element. Readers are given code-snippets and brief technical information that helps them translate the principles into functional software. The main sections are:

What's new, Get started, Design and UI, Overview, Design basics, Layout, Style, Motion, Shell, Input and interactions, Devices, Usability



	Atlassian	Firefox (Photon)	Mailchimp	Shopify (Polaris)	IBM (Carbon)	Google (Material)	GitLab (Pajamas)
Open source							
Public-facing	✓	✓	✓	✓	✓	✓	✓
Repository	✓	✓		✓		✓	✓
Layout							
Grid/Spacing	✓	✓	✓	✓	✓	✓	✓
Styles							
Typography	✓	✓	✓	✓	✓	✓	✓
Color	✓	✓	✓	✓	✓	✓	✓
Iconography	✓	✓	✓	✓	✓	✓	✓
Sound				✓			
Motion		✓			✓	✓	✓
Interaction				✓	✓	✓	✓
Illustration	✓	✓		✓			✓
Components							
with code	✓		✓	✓	✓	✓	✓
without code		✓					
Patterns/Regions							
...	✓	✓		✓	✓		✓
Content							
Voice & Tone	✓	✓		✓		✓	✓
Writing Style/Copy	✓	✓		✓		✓	✓
Usability							
Internationalization				✓			
Accessibility	✓	✓		✓		✓	✓
Resources							
Design Files	✓			✓	✓	✓	✓
Design Blog		✓			✓	✓	
Development Blog	✓	✓			✓	✓	
Fonts		✓		✓		✓	
Icons (SVG)	✓	✓	✓	✓	✓	✓	✓
Logos	✓	✓	✓				✓
Miscellaneous							
Marketing	✓						✓
Design principles		✓		✓	✓	✓	✓
Design tokens		✓					
Theming				✓	✓	✓	

Figure 3.5: A compression between a couple of public design systems [37].

3.6 Customisable Software

For many years, the trend has been that companies require increasingly niched software for their business [43] [44]. One way software providers attempt to meet the demand, while maintaining a manageable business model, is by offering increasingly customisable software. To offer each customer a unique product, is no longer a feasible business model for many providers [43]. Still, as a term customisable software encompasses a spectrum of different solutions, which part of the spectrum usage refers to depends on context [43]. Also, a concept that sounds similar is custom software, and it often refers to when the vendors create bespoke software for a customer [43]. There is also a distinction between configuration and customisation, the former is when a user adjusts only a pre-existing program functionality, offered by the software as is [44]. It should also be noted: customisation is not about offering users the option to tweak and define every possible aspect of a program. This approach will likely fail [45]. When software users are given too much freedom they easily become disoriented and overwhelmed, integrating beneficial customisation capabilities requires extensive work [45]. How customisation and application building is supported in The Product is described in section 2.2.1.

3.7 Information Architecture

Information architecture (IA) is described by Rosenfeld et al. as a way of "making information findable and understandable". Moreover, they claim that there are many ways of attempting to define it, but that the following bullet points summarise the core of the topic [46]:

1. The structural design of shared information environments

2. The synthesis of organization, labelling, search, and navigation systems within digital, physical, and cross-channel ecosystems

3. The art and science of shaping information products and experiences to support usability, findability, and understanding

4. An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape

IA is concerned with the structure and relationship between software non-data objects [46]. Information granularity is one example of a factor it examines, i.e. what are the smallest possible 'atoms' in a system? Socks and jackets could be the smallest atom for an online fashion retailer. The items in the system have to be findable, and to understand what is findable, the information-seeking behaviour of the system's users must be discovered. Visitors to the fashion site would probably find "blue socks" or "blue socks for children" a more useful atom. In summary, IA covers everything from deciding what the information consists of to studying the behaviour of the information seekers.

Getting lost in an application can be both scary and frustrating for the users. In an attempt to solve this, the users are provided with a navigation system [46]. The basic parts of a navigation system are local, contextual, and global navigation. While global and local navigation is about learning the users where they are and how they can navigate, contextual navigation is about conveying how the current objects on the screen are related . Moreover, a digital system does not naturally provide the user with clues about where they are. In the real world, directions like north and south always exist, but in a digital system, the creators have to mediate contextual clues to the user by themselves. An overview of the navigational cornerstones can be seen in fig 3.6.

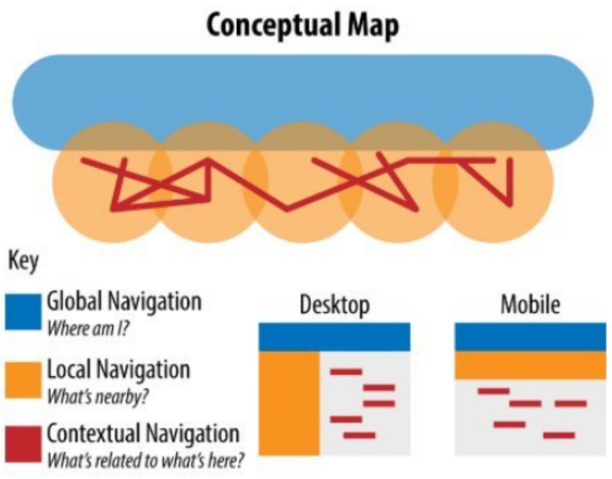


Figure 3.6: An overview of the global, local and contextual navigational in an application [47].

3.8 Desktop Applications

According to Cooper et al., all products have a posture: how the product represents itself for its users. When compared to humans, a generalised posture of a guard may be stiff and stern, while for a child carer it would be warm and loving. Desktop applications have three types of postures: sovereign, transient and daemonic.

Applications that do not occupy most of the user's workflow is a transient application [48]. This type of application 'comes and goes' and is usually specified on one certain task. An example of such an application is widgets in mobile phones. Daemonic applications are even more silent than the transient ones. They act in the background, and they do not need any human interventions. One example is printer drivers.

Sovereign applications, however, are applications that users spend a lot of continuous time in [48]. They are the main point of a user's daily workflow. As the users spend several hours in these applications, they tend to evolve from a novice user to an intermediate one quick. Compared to the total time the user spends in the application, the novice time is brief. Therefore, the target of such applications should be intermediate users.

A consideration when developing sovereign applications is the screen real estate. As these types of applications have their users' full attention, they are mostly used full screen [48]. Hence, making the amount of available screen real estate large. That allows providing rich visual feedback and lots of tools. Do not clutter the screen though, as the users stay in the application for lengthy periods, the application should have a conservative interface with subdued colours.

4

Methodology

In this chapter, the field of design research along with design processes is presented. This thesis is implementing an iterative design process, hence methods suitable for each step of the iterations are also described.

4.1 Design Research

Gustav Frayling wrote that research might be the most important nourishment for design. The remark came from his paper Research in Art and Design, where Frayling reasoned about how design research, as an activity, had evolved [49]. There is Picasso who describes himself studying photos, books and more before painting. On the other end we have a strict scientist who uses scientific methodology to improve a design process. Both are concerned with design, and use research to improve it. And while there are similarities in their approach, few would likely argue that they are totally equal. To create a distinction, while remembering that there is common ground, Frayling introduced three new kinds of design research: research into, for, and through design [49]. According to Frayling each branch is about the following:

Research into design

When we study design itself, it's history and the different theoretical perspectives associated with design.

Research for design

When we do research to inform a design.

Research through design

When we use design as a means to do research.

Introducing these new sub-domains helps us to evolve our understanding of design research. And in the end, we humans want to systematise our surroundings.

4.1.1 Research Through Design

Although it has been almost three decades since Freyling presented research through design (RtD) [49], there is still no exact definition of what it is. Today, the common understanding is that research through design refers to when design activities are used to create new scientific knowledge [26]. Moreover, although not a requirement, this research often includes prototypes. In a scientific context, RtD usually aims to apply design in such a way that any new knowledge also becomes sharable. Being sharable means that the knowledge is generalised enough to potentially be of value to other research. John Zimmerman is generally acknowledged for bringing RtD thinking to Human-Computer Interaction (HCI). Integrating an RtD into a traditional HCI process is by John considered an act with only benefits for everyone involved [50]. This thinking could enable teams to tackle wicked problems. A set of problems which traditional scientific or engineering methods cannot solve to a satisfactory degree - the problems are seldom possible to completely solve. It also strengthens designers' role in product development, since it brings out what designers do best - they reframe problems. Finally, the likelihood that design knowledge can successfully spread increases. The chance of spread is likely when prototypes, or design artefacts, are created, since they, as Zimmerman explains it, are the "currency" of design communication [50]. In other words, design artefacts allow designers to embody their ideas. When an idea can be represented by an artefact, it becomes easier for the designer, and his or her peers, to evaluate and refine the artefact, and the idea itself.

4.2 Design Frameworks

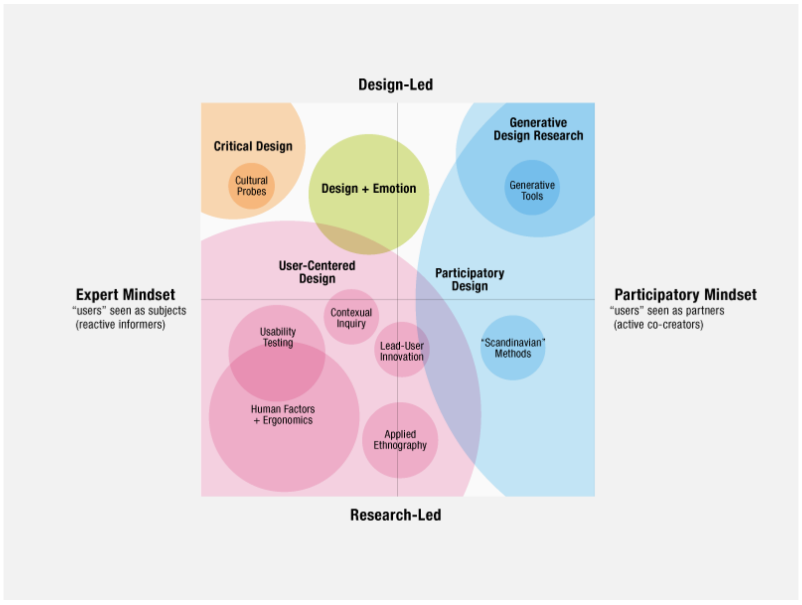


Figure 4.1: Sanders map of the relation between different design fields [51].

In 2008, as an attempt to understand the lay of the 'design land', Sanders published the Design Research map, shown in figure 4.1 [52]. The map was an attempt by her to understand how different approaches to design relate. The placement of each approach (bubble) depends on its relationship to the two axes. The horizontal axis represents mindset, and the vertical approach. According to Sanders, the two biggest groups with opposing views are expert mindset (left side) and participant mindset (right side). To the left, designers act in a culture where they view themselves as the experts who help users, they design for users. Participatory design, however, views users as equal partners, designers in this culture design with users. To clarify, the opposing views do not relate to organisational or work-life relationships between designers and users. Both sides do expect designers and users to treat each other with professional respect.

4.2.1 User-Centred Design and Participatory Design

Today, the design framework User-centred design (UCD) is characterised by its view of the user as the central subject, or participant, in connection to the development of a design [53]. Commenting on the beginning of philosophy, during the 1970s, Sanders wrote that designers were then considered experts who helped the users (subjects) [51]. Observation of the passive user was considered the best way for designers to gain information that could be valuable for the design process. Throughout the years, UCD evolved, and it came to value increased user inclusion during design processes, while simultaneously softening the belief that only designers have enough authority to actively shape a design. Still, far from everyone in the design community believed that the teachings of UCD embrace an optimal relationship between the design process, designers and users. For example, Participatory Design (PD) grew from the increasing wish, among some designers, to include users as active participants or co-creators during a design process. Followers of PD did not want to exclude users in the same way they believed UCD did. The main worry was that insufficient involvement could stop designers from fully understanding the users' needs. Ultimately and unknowingly leaving users with poorly informed designs [51].

Meanwhile, Professor Don Norman, who is credited with popularising UCD, has argued that being "a fly on the wall" during users' interaction with a system - can have tremendous value for designers [54]. The value in observation comes from the difficulty recalling past events, for users and humans in general. It is even more challenging for designers when users offer misleading information about the issues during an experience. For example, is it hard for a user to work with data, or does the import menu fail to support a correct import? It is not a guarantee that a user would notice this cause and effect, but an observant designer might catch it.

In the end, it is hard to tell exactly where the border between user-centred and participatory design lays - there is no strict line drawn in the sand. Despite the fussy borders, there still exists a distinction between the two. Nowadays there is also much prestige in being an organisation, or designer who works closely with its users. The general desire to have high user involvement might lead designers to overestimate the amount of knowledge they have. Finally, this error could lead to inferior designs and possibly hinder designers from reflecting clearly about how future processes can improve.

4.3 Design Process

A design process is the journey from start to finish for a particular design. Chris Jones said it consists of every action a designer takes during that time [55]. By this, he did not mean going for coffee or sharpening a pencil, rather the methods used, still being energised and having the correct tools, can of course be wise. In his book Design Methods, Jones discussed what designers should consider when they construct or choose their design process. In essence, designers should reflect on what, when and how methods are used. Continuous reflection and changing methods during the process is a positive sign. If the design process does not change during the process, designers could have stopped taking into account new information that emerges during the process. Unforeseeable insights arising during a project are also why designers should refrain from working towards an initially fixed goal. In the beginning of a process, the goal state is merely designers and stakeholders hypotheses, and could be proven false.

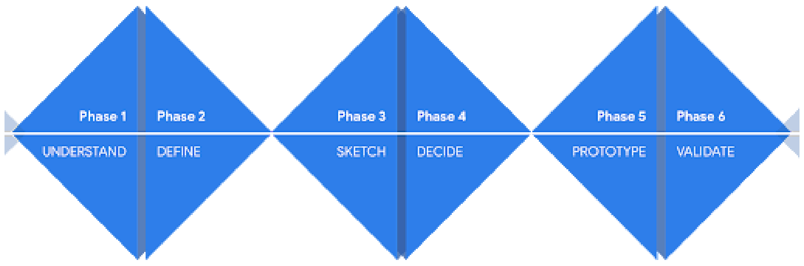


Figure 4.2: Google's *triple* diamond process, they have named it Design Sprint.

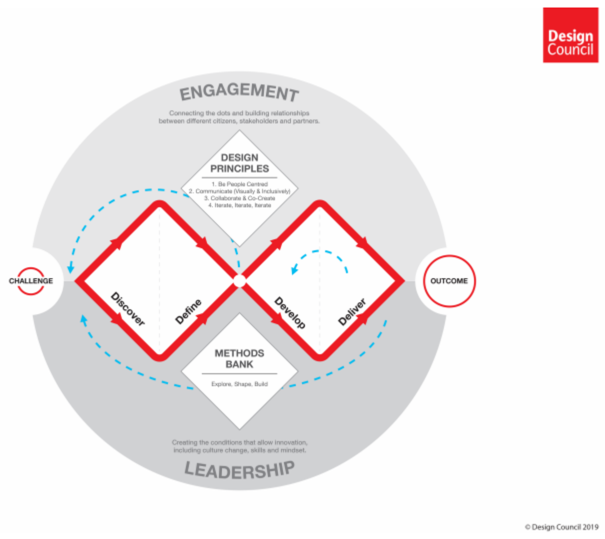

Design Council
ENGAGEMENT
Connecting the dots and building relationships between different citizens, stakeholders and partners.
DESIGN PRINCIPLES
1. Be People Centred
2. Communicate (Visually & Inclusively)
3. Collaborate & Co-Create
4. Iterate, Iterate, Iterate
CHALLENGE
Discover
Define
Develop
Deliver
OUTCOME
METHODS BANK
Explore, Shape, Build
Creating the conditions that allow innovation, including culture change, skills and mindset.
LEADERSHIP
© Design Council 2019

Figure 4.3: Design Council's *double diamond process*, called Design Sprint.

Today there exist a myriad of different design processes, and virtually all are iterative. A popular set are the diamond-processes, see figure 4.2 [56] for Google's version of it - a *triple diamond*. In general, scanning one of the shapes from left-to-right, the first half represents an exploratory phase, and the last half represents narrowing and decision-making. In its entirety, each diamond represents a phase and together they form the process. Google's UX team offer their process as a template for designers to customise according to their needs and timeframe [56]. The Design Council chain together two diamonds and suggest a *double diamond* process, see figure 4.3. The blue arrows in the figure also represent that a (digital) design "is never finished", i.e. many products or services require continuous improvement. Moreover, the arrows also echo Jones' view of the process and its ability to find information that fundamentally changes the goal and sends designers back to the drawing board.

In summary, when designers start constructing their process, they should consider the possible solutions, but there are other important factors to consider. One of them is change, with regard to the temporal dimension and new requests from stakeholders. Also, Jones emphasises that having a well-planned process demystifies the work of designers, and allows for better communication between designers and non-designers. A robust process can also act as a counterweight towards all the uncertainty that comes with design work. Since design work entails uncertainty, a clear process can act as a confidence booster.

The following sections describe one possible composition of phases and methods for a triple-diamond process. The usage of six phases is with inspiration from the previously mentioned Design Sprint by Google [56]. Although not explicitly shown in the sections structure below, we would like to stress that many methods are applicable during several phases. For example, Harrington and Martin suggest - when writing about their five phase process - that interviews give value during phase 2, 3 and 4. To provide a clear structure below each method described is under a phase were it commonly occurs.

4.3.1 Understand

As indicated by the shape of the diamond - expanding from the left towards the mid-line - this initial phase is categorised as being expansive or divergent [57]. Google wrote that the goal of the exploration should be to establish a common foundation of knowledge among process participants [57]. Knowledge acquisition is also intertwined with iterating the problem definition. In other words, besides giving designers contextual knowledge, this phase also informs designers, and help them to refine, or re-frame, their perspective on the design problem [55].

Methods used during this phase should help the designer gather relevant information. **User Interviews**, **User Journey Mapping**, **Affinity Clustering** are some examples of divergent methods designers could use [57]. Hanington and Martin call their equivalent of this phase *"Planning, Scoping and Definition"* [58]. Sometimes a design is iterated several times, then it could be that this phase is disregarded or given less time. Though, during a later phase, the perception of valuable information 'around the corner' may also motivate a return to this phase for more extensive probing.

4.3.2 Define

The define phase of the triple diamond design process is when the information gathered in the understand phase is summarised and analysed. Here, the problem's context is defined and the desired outcomes of potential solutions are set [59]. Several design processes include a define phase, such as in the earlier mentioned double diamond process [60], as well as in the “Design Process” [61]. In both of these, this phase serves the same purpose as in the triple diamond process.

One applicable method in this phase is **The Golden Path**, also referred to as **The Key User Journey**. This method's purpose is to define the ideal path that a user has when performing a task in the product. If a product not yet exists, the ideal path of a future product should be formulated [62]. During the creation of this path, errors and exceptions are meant to be ignored, since the purpose of the method is to focus on the main problem.

4.3.2.1 Personas

A way of summarising information gathered about users is to create personas. Cooper et al. describes personas as “user models that are represented as specific, individual human beings” [63]. Personas are intended to be the product of ethnographic studies, such as user interviews. The collected data is then interpreted by the designers and translated into personas. Personas encapsulate behavioural patterns of specific users of a specific product, and are not meant to be generalisations and stereotypes of the user group. One could say that a persona is a synthesis of a user group into a fictive person [63].

A design that attempts to fit every user's needs could, according to Cooper, increase the cognitive load and navigational overhead for the users. This is because accommodating some users' needs may interfere with other users' needs. To utilise personas can, as Cooper describes it, help designers avoid typical pitfalls like self-referential design, designing for edge cases and designing 'for everyone' [63].

4.3.3 Sketch

During this phase designers begin to shape the knowledge they gained from the precious stages. As the name of the phase indicates, designers begin to sketch, and doing it has two main benefits at this stage. First of all, transitioning the idea from mind to paper helps define the essence of an idea - what is this idea really about? [64] Also, the sketch itself communicates information that is hard, if not impossible, to convey only via talking. For example, to orally explain all the interface details of new a new program view to someone else, and expect to them to see the same vision, would be poor judgement.

As in other phases, the design work in this step can also be addressed using structured methods [65] [66]. Crazy 8 can be applied to ideate solutions for a theme [67]. A couple of team members, as suggested by the methods name, begins by drawing eight sketches. A common approach is to divide a paper into eight equal sections and sketch one solution in each section. A session usually lasts about an hour. After completion, participants can choose to develop a few of the ideas further. **Solution sketching** is one method that can be applied after period of quantitative sketching. If a sketch seems extra valuable, the method suggests some rules for how designers can develop it. **User sketching**, or stakeholders sketching is another participatory approach, suggested by IDEO [66].

4.3.3.1 Sketching

Sketching, or even doodling, is a fundamental design activity that can support designers throughout their work with, and communication of, everything from abstract process parts to concrete design elements. When is is done as a team activity cooperation between members can improve, by for example literally sketching a common image of the design processes and it's components [64]. As mentioned in the section Sketch, it can also support the communication of concrete design elements. It is also important to remember that the visual vocabulary of a sketch - as Bill Buxton calls it - affects the viewer. This vocabulary is the style or meta-design of a sketch: formed by the fidelity, composition, level of details and so forth. Buxton emphasises that sketches are not drawings, and that the distinction is important to remember. Sketches are incomplete and unrestrained in nature, and for the beholder these qualities evoke uncertainty - and that is almost always a positive state. Looking at it, spectators become naturally inclined to iron out the blank-spots: “Would you have a radio group here?”, “Is this where users a directed after login”. All this ’blank filling’ could be valuable knowledge for the person behind the sketch. Lastly, an important reminder: you do not need to be an artist to sketch - everyone can and should sketch their design ideas.

4.3.4 Decide

Sketching, or even doodling, is a fundamental design activity that can support designers throughout their work with, and communication of, everything from abstract process parts to concrete design elements. When it is done as a team activity cooperation between members can improve, by for example literally sketching a common image of the design processes and it's components [64]. As mentioned in the section Sketch, it can also support the communication of concrete design elements. It is also important to remember that the visual vocabulary of a sketch - as Bill Buxton calls it - affects the viewer. This vocabulary is the style or meta-design of a sketch: formed by the fidelity, composition, level of details and so forth. Buxton emphasised that sketches are not drawings, and that the distinction is important to remember. Sketches are incomplete and unrestrained in nature, and evoke uncertainty in the beholder. When the spectators look at them, they naturally try to fill in the blanks: “Would you have a radio group here?” “Is this where the users are directed after the login?”. All this ’blank filling’ could give the designer behind the sketch valuable knowledge. Finally, an important reminder: you do not have to be an artist to sketch. Everyone can and should sketch their design ideas.

Decisions have to be done continuously throughout the design process; which colours to use, where to place a menu or whom to interview. In the Decide phase of the triple diamond process, decisions regarding the direction or concept of the design is what the name refers to [68]. Here is where the team comes to consensus about a single idea to continue working on in the current sprint. This phase is, for example, also included as a part of the Define phase of the Double Diamond [60].

Some applicable methods for reaching a decision are **Present Solution Sketches** and **Dot Vote** [69] [70]. Present solution sketches is a method where each team member gets to present their sketch from the previous phase, to help the team gain understanding and discuss its features. The team can also discuss similarities from other ideas and make comparisons. To make the method more time efficient, it is recommended to give each team member a time limit on their presentation. This method is often combined with other ones that include voting.

A team can use the dot vote method to find an agreement regarding which idea to focus on during the design sprint [70]. By applying it, all participants see each other's ideas, each participant can pin their sketches on a wall or similar. Then, everyone gets to present and explain their ideas for a few minutes. During the presentation, the rest of the team can ask questions and discuss details. After that, it is recommended to review the criteria upon which an idea should be chosen. Each member is then given a couple of votes, dots to put on the ideas. The idea that has received the most votes will be further developed in the coming phases.

4.3.5 Prototype

In the design phase of a project, design and prototyping are often intertwined. To create a prototype as early as possible is often beneficial, as it can act as a proof-of-concept of the design idea [71]. It is often more manageable to adjust design flaws in an early prototype than it is in a programmatically implemented design. In this section, two levels of prototypes will be discussed, low-fidelity and high-fidelity.

Low-fidelity prototypes are prototypes which are intended to give an abstract idea about the intended design of the eventual product. These kinds of prototypes can, for example, be constructed on physical paper or cardboard [71]. They should not include a high level of detail, only convey a look and feel. Moreover, they should not include any navigation nor system functionalities.

Low-fidelity prototyping have several advantages, they are quick to create and therefore also cost-effective [72]. However, insufficient detail can be a disadvantage. It could be tough to use these prototypes in communication with non-designers. Software developers would have to make their own decisions about details [72].

High-fidelity prototypes are intended to give the user a detailed view of the design. These kinds of prototypes give the possibility to test and evaluate detailed interactions and design decisions before the final implementation. Even though the prototype may consume much more time and effort than a low-fidelity version, it is still beneficial to adjust design decisions at this phase rather than in a complete implementation [71]. Among these advantages, high fidelity prototypes can also be suitable for sales demos of a future product. One disadvantage of the method is that radical changes in the design become time-consuming [72].

4.3.6 Validate

In the validation phase, designers let users test the design. Feedback regarding interactions with the design is gathered to evaluate the concept. The data can be used to improve the design during future iteration [73]. A validation, or evaluation, phase is present in several design processes. In Hartson and Pyla's design process "The Wheel" it is called "Evaluation" [74] and in the double diamond process, it is called "Deliver" [60]. In both of these, it serves the same purpose as in the triple diamond process.

When discussing evaluation in design, commonly used terms are formative and summative evaluation. Formative evaluation is about "collecting qualitative data to identify and fix UX problems and their causes in the design". Summative evaluation, in contrast, is about "collecting quantitative data for assessing a level of quality due to a design, especially for assessing improvement in the user experience due to formative evaluation". [75]. The following are methods applicable in the validate phase.

4.3.6.1 Walkthroughs

One especially effective method in at an early stage of a design is **design walkthroughs** [76]. This method is quick, and even though it may be most effective in the early stages, it can be applied in any stage of the design. It works in a way that a group is collaborating with a leader that guides the group through the design. This group can consist of, for example, the design team, customer representatives, and UX analysts. Together, they strive to explore the design on behalf of users and try to anticipate what problems might arise.

A variant of design walkthroughs is **cognitive walkthroughs**, where the most frequent tasks made by new or infrequent users are explored [77]. Throughout every step of the task, the performers of the method ask themselves: "Will the user know what to do at this step?" and "If the user does the right thing, will they know they did the right thing and are they making progress toward their goal?" [77]. The purpose is to evaluate the workflow of the product on a detailed level.

Another popular walkthrough method is **heuristic evaluation** [76]. This method is an analytical inspection of the design, performed by an UX designer. Sometimes it can be beneficial if the performer of the evaluation is a person outside of the design team, to get a new perspective. Heuristic evaluations and cognitive walkthoughs have similarities, both require a specialist who tests the design and tries to predict problems that could affect users. The difference with heuristic walkthroughs is that the entire design is tested, and that generalised design guidelines drive the inspection. Originally, these generalised design guidelines are the ten heuristics, created by Jakob Nielsen [76]. This method is both cost effective and intuitive to perform.

4.3.6.2 Remote Usability Testing

Another way of validating the design is by conducting remote usability tests. These tests utilise an application to record the screen and perhaps the voice of the tester, as they interact with the product or prototype remotely [78]. To execute the tests remotely lowers the resource costs and enables the testers to perform the tasks in their natural environment. There are two types of usability tests, moderated and unmoderated.

Moderated usability tests have a moderator that observes a user who interacts with a design [79]. Often the user is asked to complete a set of predefined tasks. The moderator is supposed to provide guidance and answer any questions that may arise. The advantages of moderated tests are that the prototype or product tested may lack functionality and be complex, as a moderator guides the tester through the test. It is also possible to acquire a deeper understanding of the user journey, as the test is almost an interview between the tester and the moderator. The disadvantages are that, as they involve both a moderator as well as a tester, they require scheduling and coordination, and are more time-consuming. Therefore, they may be more expensive to conduct.

Unmoderated usability testing does not involve anyone except the person testing the product or prototype [79]. The tester completes all the tasks by themselves, and therefore this method may be faster than the moderated version. This method is feasible for validating concepts in the design and for observing the tester in their natural habitat, while not getting affected by another person. A benefit with this test is that it requires little time and thus enables a large sample size. One limitation is that the moderator cannot ask any questions, and therefore the tasks must be clearly articulated in advance.

4.4 Software Tools

In the following chapter, a description of each software tool used to develop the prototype is presented. Software used for organisation, communication or other activities is not included.

Miro is a collaborative, online, whiteboard [80]. It can be used for creating wire-frames, taking notes, conducting workshops and more. Users create and store their content on an expandable canvas.

Balsamiq is a tool for creating low-fidelity prototypes [81]. It mimics sketching on real paper by only offering black and white components for wireframing. This allows for rapid creation of wireframes with a focus on structure instead of details.

Adobe XD is a collaborative tool for high-fidelity prototyping [41]. The user can create wireframes, animations, and runnable prototypes without the need to code, which makes the prototypes behave similar to a regular program.

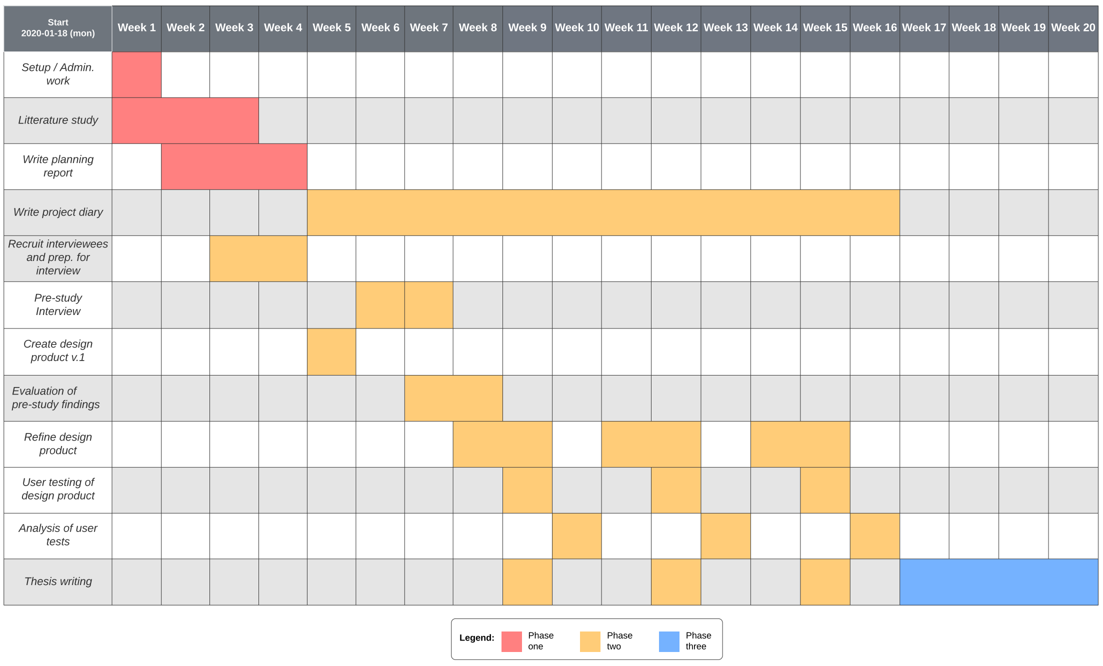
5
Process

This chapter describes the preparation and execution of the design process. A four-week literature study was conducted before the process began. We examined The Product and other Company resources in parallel with the study. The design process mainly consisted of three iterations. During all of them, both the prototype and the guidelines were iterated. All three iterations consisted of the following phases: understand, define, sketch, decide, prototype, validate, section 4.3.1 to 4.3.6 covers the theory behind them. Continuously collecting and analysing user data was a key motivation to this approach. The users this thesis concerned are described in section 2.3.

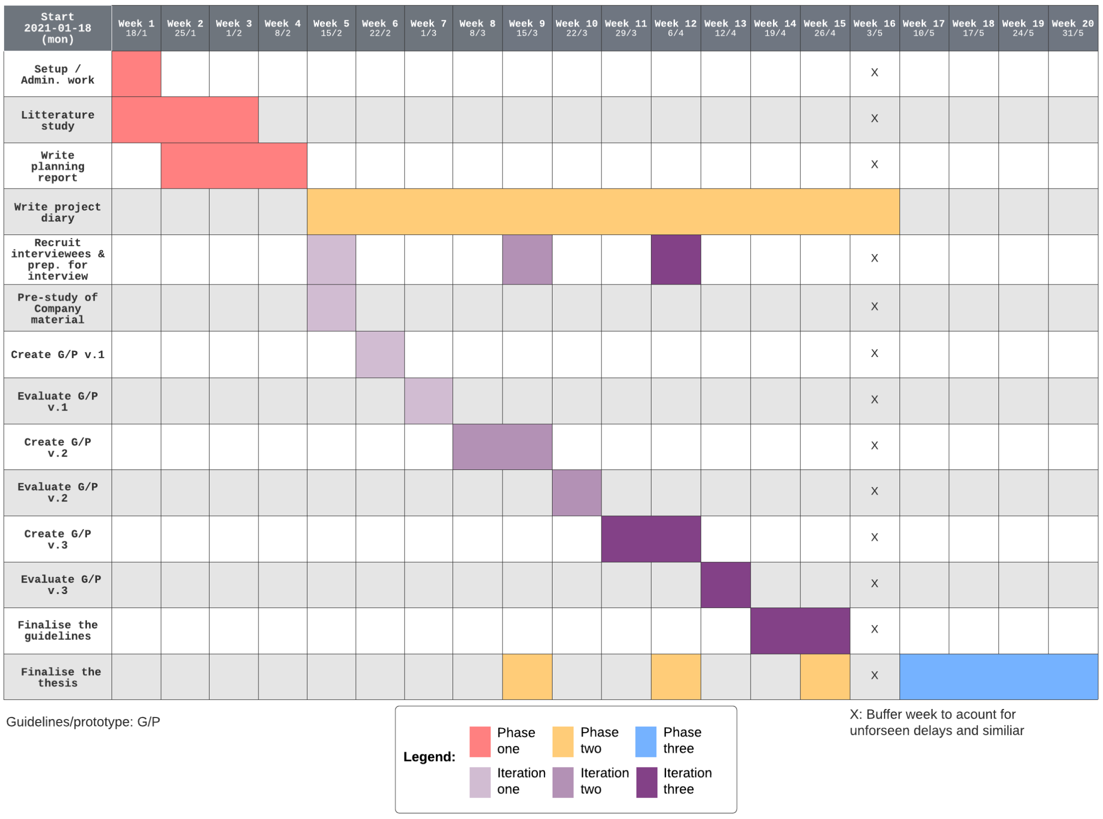
5.1 Planning and Preparation

Work with this thesis began 2021-01-08 and lasted about 20 weeks. It was carried out at the Chalmers University of Technology and at The Company during the spring of 2021. The thesis was created in three steps: preparation, execution, and writing.

A thesis proposal was created before work began in January. It included a first version of the design process and a week by week plan for the execution of this thesis. The plan was summarised in a Gantt chart, and can be seen in fig 5.1a. After the thesis proposal was approved, a planning report was started. This planning report included three weeks of literature study and a study of The Product. At the end of the preparation phase, the original plan in fig 5.1a was revised. This was done as a consequence of new findings about The Company and the work this thesis would include. Hence, the revised plan contains more details about each iteration of the design work. The revised plan can be seen in fig 5.1b. A more detailed description of the earlier mentioned literature review can be found in section 5.1.3, but to summarise, the outcomes became this thesis theoretical framework.



(a) The first version of the Gantt chart showing the execution plan of this thesis.



(b) The refined version of the Gantt chart showing the execution plan of this thesis.

Figure 5.1: Gantt charts showing the planned process, each square represents a week.

5.1.1 Understanding The Product and Its Users

Significant time was spent exploring the The Product and learning about its users. The company did two in-depth interviews with representatives from company customers to provide some initial data. Besides them, understanding how The Products works was a time-consuming process, data science and visualisations themselves are complex fields. Furthermore, there exist many kinds of user-company relationships, as written in 2.3. Another layer of complexity comes from the fact that users are found in all major industrial sectors. Information from The Company and research showed that every sector can apply data science differently. Hence, users often request functions that only address issues unique to the sector they operate in. Moreover, 'the user' can refer to a single individual, or a large team with professional software developers, data scientists and many more, whom all collaborate. The epithet Data Scientist also denotes individuals with diverse skills and goals. Some have extensive software development knowledge, and appreciate using it, others do not. In summary, there has been some research on data scientists, although much remains to be done [82].

Many of the initial guidelines are based on insights we got while familiarising ourselves with The Product and studying The Company's data about their users. Managing the navigation, especially in larger applications, emerged as a possible pain point. The Company also received much feedback regarding The Products support for scripting. Users complained about having to use it and the experience while using it. We explored what design patterns the GUI uses, and not. A tricky question, related to the GUI, is how much information the software itself should present automatically. It should show relevant information without drawing too much of the users' attention away from the visualisations. From our initial probe and the interviews, some information about system status was unnecessarily challenging to find.

5.1.2 Study of Interviews Conducted by The Company

During the literature review, we examined two interviews that The Company had conducted. The aim was to understand what applications are and how the process of creating them works. It was discovered applications can have over 50 different pages. Thus, be large and complex. In large applications, it can become challenging for application builders to manage the navigation structure, and the struggle can affect the application's user experience. It was also discovered that application users regularly perform various tasks that depend on the information conveyed by visualisations. We came to call these tasks *workflows*, further described in section 5.1.5. One important condition of workflows is the clear difference between business areas in terms of how strict and controlled they must be. Some areas value strict workflows in which the user can only view one page at a time. These processes could also be subject to audits by external organisations and government agencies. Other application builders needed to build a more exploratory workflow.

5.1.3 Literature Review

As mentioned in the beginning of this section, data from The Company about their users influenced the literature review. The company suggested some books regarding user experience and information visualisation from their private collection. By studying this literature and company data about users, a couple of initial keywords emerged. Some of them were Information visualisation, User Experience (UX) and User Research. In addition to these domains, theory about design systems and industry examples of it was examined. The main goal was to find UX research, or related knowledge, about data scientists. It was considered preferable if the research involved activities comparable to application building.

Google scholar [83] and the databases IEEE Xplore [84] and ACM Digital Library [85] were frequently used to find relevant research. Google's search engine [86] was used to find industry resources, new keywords and inspiration. We considered industry practice, as it has a considerable impact on design development.

Early in the research, it seemed like application building inside software was a rather unexplored field, particularly in the world of analytical software. Yet, the term End-User Development [32] was discovered, together with Developer Experience [28], and Customisable Software [44]. Section 3.3.1 describes Kari Kuusinen's work with developer experience and software developers. As part of her work, Kuusinen developed a set of categories to measure what software qualities the developers appreciate. From these categories, the first guidelines came to include a set of tags. The purpose with them was to achieve a mapping between different guidelines and users' needs. The needs of developers can still differ from the ones data scientists have. However, we deemed her work to be useful, since this thesis was concerned with data scientists who, by some means, create software.

While both user and developer experience are interested in users who create or extend software, neither appeared to have extensive information about the software's user experience. In conclusion, likely, there has not been much previous research closely related to this thesis.

In addition to the search for a theoretical framework, the literature review also influenced the methodology. Another source of influence was literature from previous courses that the authors of this thesis had attended. Some of these courses were Interaction Design and Methodology, Interaction Design and Prototyping, and Information Visualisation. As the planned result of this thesis were design guidelines and a high-fidelity prototype, the field of Research Through Design was explored. The outcome of the methodology research, the design process and chosen methods, are presented in section 4.

5.1.4 Reflections based on the Literature Review

A goal with the literature review was to narrow the scope of this thesis, to formulate guidelines for application building without any limitations, was not feasible. The intention was to review the discovered literature and select the most suitable theories. By making this choice, the scope would naturally become narrower. Attempts to find literature regarding application building inside an analytics platform were done, but nothing was found. In our opinion, end-user-development and developer experience was the most suitable literature. Our initial data about data scientists also supported this selection. As mentioned, many scientists disliked when they had to include custom code in their applications.

As mentioned, End-User Development theory was discovered, and it appeared valuable. Research within the domain was concerned with the user experience end-users have when they create software. In EUD literature, an end-user is someone who is a non-professional software developer and develops software, often by other means than traditional programming. End-users were an ambiguous name, since we viewed application users as the 'end user'. We made an attempt to find literature about those who use software built by end-users. Nothing of true relevance was found. But since there are other, more general methods to improve user experience in applications, they were instead explored. One example of a more common approach is to employ a design system. These help designers and developers achieve a consistent design, which many data scientists want to achieve.

In summary, about 70 papers and books were collected. The most related topics for this thesis appeared to be EUD and developer experience. We had also found many unconnected theories within these domains. In a sense, the review still managed to narrow our scope. However, our sources still lacked a clear connection to application building in analytical software. The current process of application building in The Product had to be explored thoroughly. To evaluate the potential value of our sources and to obtain more specific guidelines.

5.1.5 Workflows

Early in the process, the term *workflows* became part of discussions and appeared during interviews, see section 5.1.2, with The Company. What workflows refer to is highly dependent on domain and industry. Our interpretation of the term is: a set of tasks performed inside an analytical application. The tasks are performed either on data rows, visualisations, or in one or more pages. This interpretation has been developed throughout the process, but it was established at this point. Another, more general, definition is “a workflow is the organisation of steps into a sequential order” [87]. A step can be an activity or a resource. Many scientific disciplines have developed versions that are specific to their domain. We discovered, via the interviews described in section 5.1.2 and 5.2.3, that a flexible degree of strictness is important in the world of analytics. In a strict workflow, users can only visit predetermined locations in the application. During the opposite scenario, users explore contexts to complete a task. The level of strictness often depends on the business domain. For example, clinical studies require very strict workflows with external audits, due to regulations by law. Hence, the employee has limited access to pages at steps in the workflow. However, some business domains encourage exploratory workflows where the employees can wander more freely in the application.

Usually, analytical workflows refer to a set of instructions that analytical software performs [88]. The analyst or scientist creates the content of each sequence and defines conditions for when the software should execute it. And while a sequence in this context likely represents a mathematical model, it could represent something completely different in another domain. In summary, the core structure of workflows remain similar between different domains. It is primarily the content and operators that could differ.

The Company has another product for the construction of data science workflows, figure 5.2 shows its canvas. Among many features, program users can connect data sources and apply conditional step-by-step processes to them. It can also be used to create machine learning and data preparation pipelines. In general, people who work with data science often use workflow concepts to organise their work [88]. This tool was discovered during user interviews in iteration one, which can be read more about in 5.2.1.

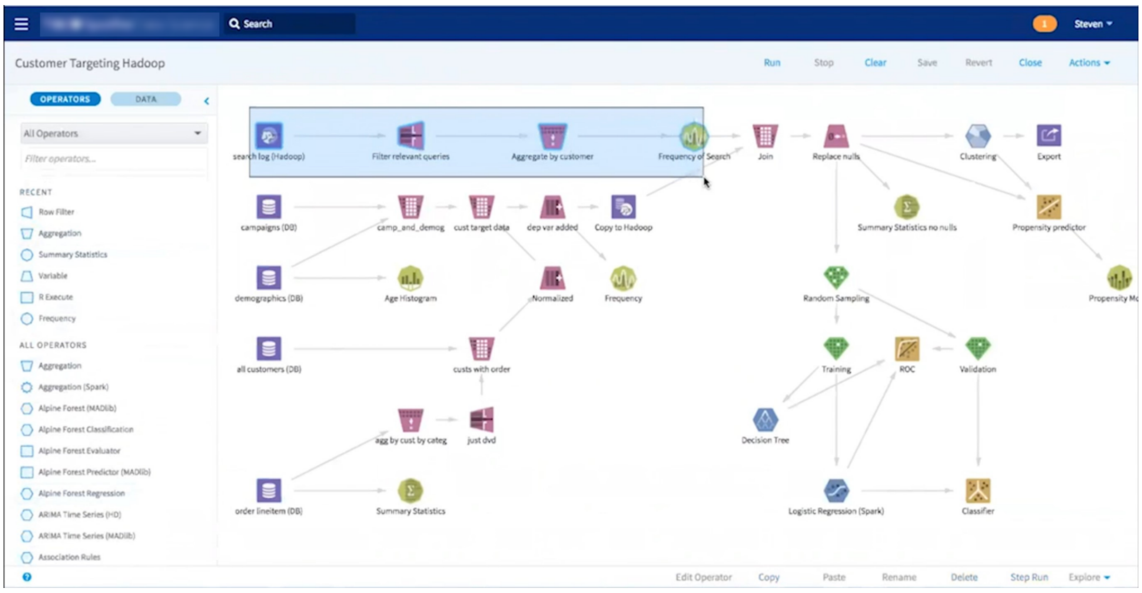


Figure 5.2: A print screen from The Company's data science product and its canvas.

5.1.6 Guidelines v.0

Multiple sources influenced the first set of guidelines. The interviews described in section 5.1.2 had a substantial impact. At this point, they hinted about the various struggles some applications builders experience with scripting. By studying the software, we also noted some questionable software behaviour, with respect to, e.g. how it conveys information about its current state. Still, most influence came from existing research and industry work described in 5.1.3. It should also be noted that all three sources affected each other since work with them was done in parallel.

Theory about design systems and industry examples of them were used as a source of inspiration for the guidelines format. Miro was used to create the guidelines and to store most process resources, a description about the program is found in section 4.4. The guidelines used a tag system, this can be seen in fig 5.3. The tags were based on Kuusinen's paper regarding developer experience, described in section 3.3.1, and were stated as follows:

Informative, Efficiency, Flexibility, Application Design, Reliability and Advanced Usage

The first set of guidelines contained eight guidelines, labelled G1 to G8, their tag system can be seen in fig 5.3):

G1 **Provide a curated set of default components for application building, and allow developers to create their own**

Default components should be customisable and help developers create the desired user experience.

From the interviews described in section 5.1.2, it was discovered that application development requires extensive use of code that is not related to the data and visualisations. According to end-user development theory, section 3.4, non-professional software developers are more likely to reach their development goals by using other methods than text-based programming.

G2 **Provide developers with flexible and informative functionality that supports the creation of the desired navigation structure within an application**

- Users should feel able to explore with ease.
- Developers should have extensive software support for the creation of predefined and restricted navigational paths.

During the interviews described in section 5.1.2, it discovered that applications can contain over 50 pages. Without sufficient navigational support, there is a risk that applications users become lost.

G3 **The Product and its applications should provide developers and users with modeless feedback.**

Follow best practice for different load times.

One topic in the interviews described in section 5.1.2 was that users of an application do not know the status of scripts with long loading times. To show system status is also a part of Nielsen's ten heuristics for a better user experience [89], and encouraged in desktop applications, see section 3.8.

G4 **Provide an approachable API of The Product**

Make on-boarding to the API smoother by providing examples of how parts of the API can be used, provide instructional material with how-to examples

The mentors of The Company mentioned that the API of The Product appeared to have a steep learning curve. Those who could use it believed it greatly improved their ability to customise. Therefore, a proposal is to offer new application builders a more approachable API. A paper on how to construct APIs more user-friendly was studied. It concluded that, for example, APIs with thorough examples and how-to:s were appreciated by users [90].

G5 **Ease the cognitive load when developers perform complex tasks**

– The Product should provide a visual preview of visible application components, e.g. for buttons, links, checkboxes, lists.

– The cognitive load for developers should be minimal. When possible it should be replaced with visual and motor load: especially during complex tasks.

– Follow developers and users mental models.

– (High-level) naming should capture the essence of what it represents.

Applications studied in the interviews in section 5.1.2 gave a messy impression. In the current version of The Product, there are no instant previews of components not related to visualisations. Therefore, cognitive load according to Lightbown was studied.

G6 **Customisation functionality should consider The Gentle Slope of Complexity.**

See section 3.4.1 for a description of The Gentle Slope of Complexity.

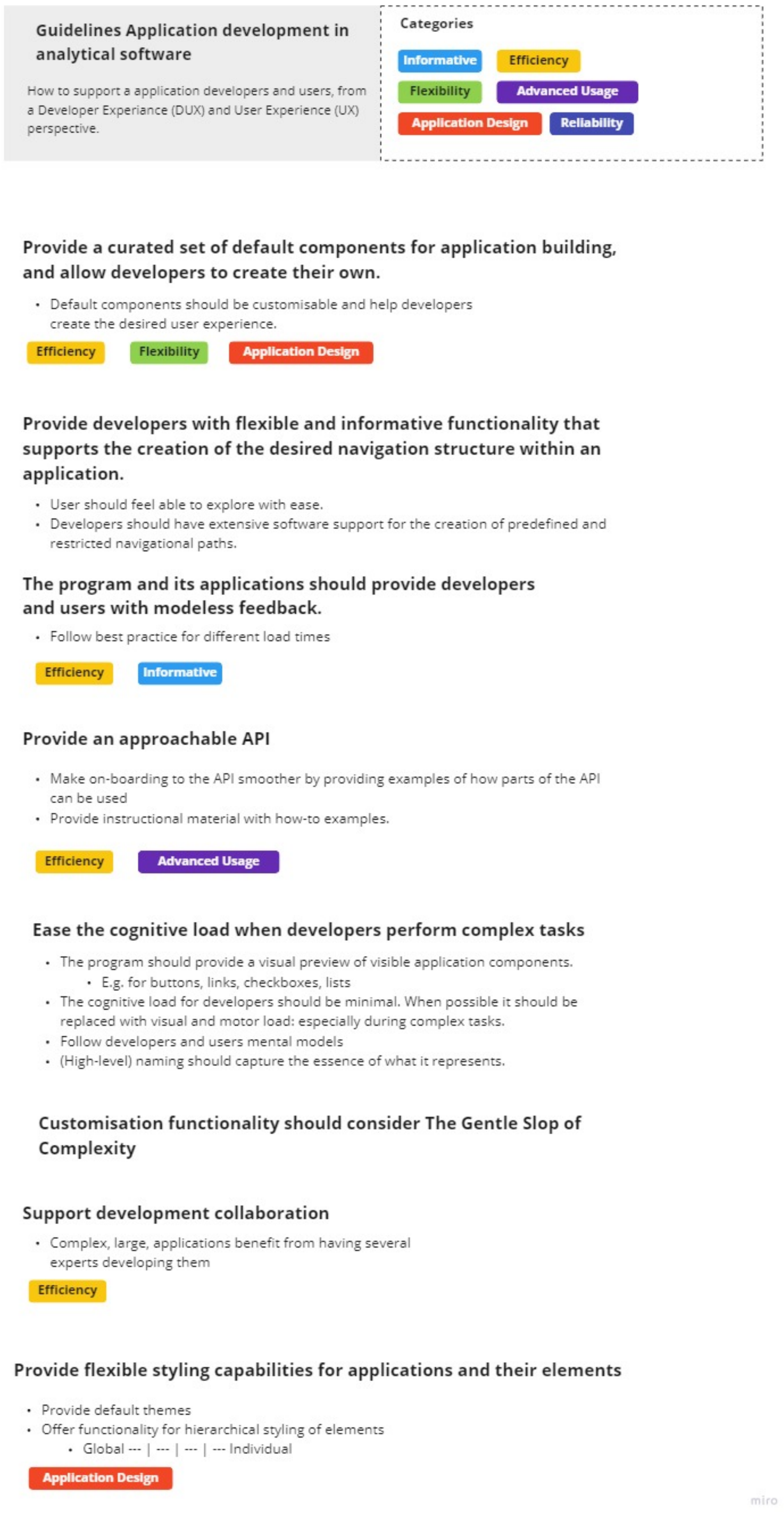
G7 **Support development collaboration**

Large applications, those with tens of pages, often have several developers. One point of discussion during the interviews, in section 5.1.2, was that application developers could not collaborate, as they wished, when creating an application. Support for collaboration should be considered according to EUD, section 3.4.

G8 **Provide flexible styling capabilities for applications and their elements**

- Provide default themes.
- Offer functionality for hierarchical styling of elements.

This was added as the users of The Product values a high level of customisation.



Guidelines Application development in analytical software

How to support a application developers and users, from a Developer Experiance (DUX) and User Experience (UX) perspective.

Categories

Informative · Efficiency · Flexibility · Advanced Usage · Application Design · Reliability

Provide a curated set of default components for application building, and allow developers to create their own.

- Default components should be customisable and help developers create the desired user experience.

Efficiency · Flexibility · Application Design

Provide developers with flexible and informative functionality that supports the creation of the desired navigation structure within an application.

- User should feel able to explore with ease.
- Developers should have extensive software support for the creation of predefined and restricted navigational paths.

The program and its applications should provide developers and users with modeless feedback.

- Follow best practice for different load times

Efficiency · Informative

Provide an approachable API

- Make on-boarding to the API smoother by providing examples of how parts of the API can be used
- Provide instructional material with how-to examples.

Efficiency · Advanced Usage

Ease the cognitive load when developers perform complex tasks

- The program should provide a visual preview of visible application components.
  - E.g. for buttons, links, checkboxes, lists
- The cognitive load for developers should be minimal. When possible it should be replaced with visual and motor load: especially during complex tasks.
- Follow developers and users mental models
- (High-level) naming should capture the essence of what it represents.

Customisation functionality should consider The Gentle Slop of Complexity

Support development collaboration

- Complex, large, applications benefit from having several experts developing them

Efficiency

Provide flexible styling capabilities for applications and their elements

- Provide default themes
- Offer functionality for hierarchical styling of elements
  - Global --- | --- | --- | --- Individual

Application Design

Figure 5.3: The first set of guidelines - with its tag system - during development in Miro [80].

5.2 Iteration One: User Research and Exploration

During iteration one, the goal was to deepen the understanding of The Product's user and the problem space. To continue having an informed prototyping process. Feedback from long-term users of The Product and product managers was collected by conducting semi-structured interviews, described in section 5.2.3. Videos of each interview were transcribed and later analysed by affinity mapping.

After the analysis of the interviews, several topics emerged. We designed and led an ideation workshop based on these topics, together with The Company's UX department. This workshop generated many possible solutions for issues that may appear during application building in The Product. Afterwards, we identified themes among the ideas and used them during the first sketch phase. Finally, the initial guidelines were evaluated against the sketches, and many were reformulated.

5.2.1 User Interviews

After studying the interviews in section 5.1.2, a deeper understanding of the general application creation process was needed. To achieve this, two user interviews were held. The interviewees were two data scientists who work at The Company. As part of their role, they build applications for marketing and information resources. Besides that, they also support company customers in building the applications they require. Prior to joining The Company, one of the scientists used the software on a daily basis as a customer. The interviews were held remotely via a video conference call and followed a semi-structured approach.

As mentioned, the goal of the interviews was to get a better understanding of the application building process in The Product. Questions regarding findings in the previously described interviews were asked, like how they view The Product's support for navigation in an application. We also asked what they view as the biggest challenges with application building. Another topic was examples of common workflow use cases they are asked to implement for customers. All questions can be found in Appendix A.

5.2.2 Internal Stakeholder Interviews

Two interviews with product managers of The Product were conducted to get a deeper understanding of The Company's customers. The questions were about what needs managers had identified, related to application development of The Product. The product managers had two different areas of responsibility; The Product, and its API. As the interviewees have different expertise, the questions asked were adjusted accordingly. Both interviews followed a semi-structured approach and were held remotely.

The product manager with a general product responsibility was asked to reflect on his application building experience. Furthermore, customers' needs and The Product's future were discussed. For the product manager with the API focus, questions were asked about feature requests and the most common complaints from customers. Lists containing all questions in both interviews can be found in the Appendix A.

5.2.3 Analysing Both of the Interviews

Notes were placed on a virtual whiteboard to summarise the interviews and their topics. These notes were later categorised using affinity mapping, which can be seen in fig 5.4. A deeper understanding of applications inside The Product was gained from the interviews. The main takeaways were that the application development process in The Product appeared to lack sufficient support from The Product. There are several community discussions and blog posts with guides and tips, but there is insufficient integrated support for common tasks, such as building a global navigation panel. The lack of support has led to applications with script-based 'hacks', along with the need to use duplicate code in many pages. Ultimately, applications can become difficult to maintain.

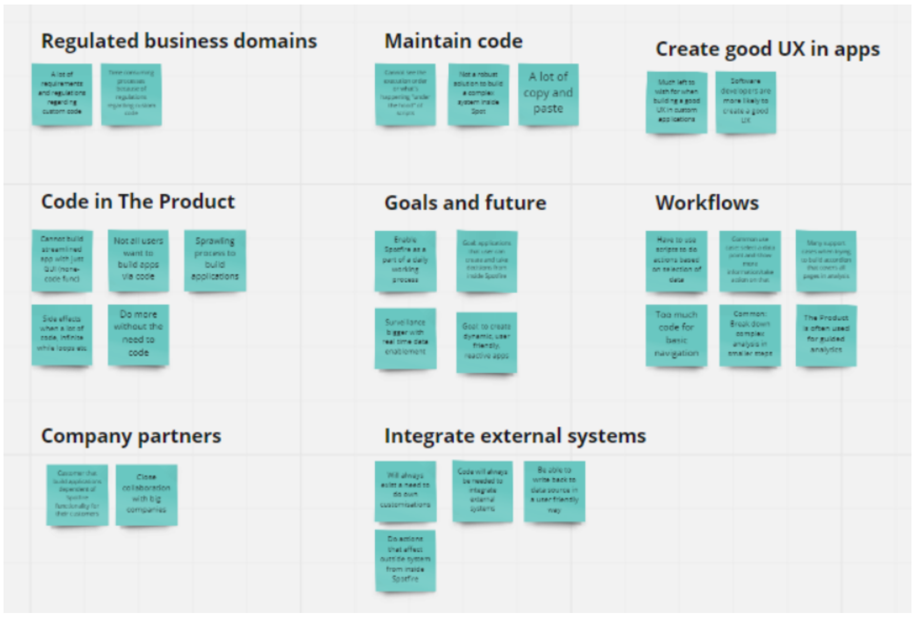

Regulated business domains
Maintain code
A lot of copy and paste
Create good UX in apps
Code in The Product
Not all users want to build apps via code
Sprawling process to build applications
Do more without the need to code
Goals and future
Workflows
Too much code for basic navigation
Company partners
Integrate external systems

Figure 5.4: Affinity mapping from one of the interviews.

Nearly all business areas use visual analytics. Consequently, The Product has users from different backgrounds and with different expertise. Users therefore need The Product to offer a high degree of customisation. Currently, almost any software behaviour can be achieved by using the ability to invoke external systems via user-created scripts, in combination with API calls to The Product. However, this approach requires substantial experience in programming.

Other conclusions from the interviews were:

- Applications with a guided workflow are common. In this context, a workflow refers to a complex analytics process that has been divided into several steps.

- Many users want default components, for example, accordions, global menus, fly-out menus, and ones that support. As of now, these components have to be manually built through HTML, CSS and JavaScript.

- Some sectors require strict workflows, others exploratory. This need was discovered during the interviews described in section 5.1.2, it was also a topic during the interviews in this iteration. Pharmaceutics is an industry that often requires strict flows to ensure that work is in accordance with all laws and regulations. Section 5.1.5 has more information about workflows.

- Improved collaboration functionality would ease work with applications that have tens of pages.

- On an application page, non-visualisation content should occupy as little screen space as possible. The purpose of an applications, and its pages, is usually to let the data (visualisations) speak.

5.2.4 Ideation Workshop

The ideation workshop was to be the last step before the sketch phase began. Its topics were based on the previous interviews, and the goal with them was to generate ideas for sketching. Twelve designers from The Companies UX department participated by remote. During an hour, participants switched between ideation in breakout rooms and brief presentations in the main room. For each topic, there were eight minutes to discuss, then each group presented for two minutes. Both of us acted as facilitators of the ideation workshop. We prepared a virtual mural board with sticky notes, for the attendants to use, and it can be seen in fig 5.5.

The workshop included a fictional data scientist named Bosse, and he used The Product on a daily basis. Bosse must - preferably with a minimum of custom scripts - build and later maintain a large application in The Product. Together, the topics' scope was from the smallest element, data rows in a visualisation, up to an entire analysis. A slide from the introduction of the workshop can be seen in fig 5.6.

Guided workflows were the first topic. Bosse wants to create an informative, guided workflow for application users, but information must not occlude page content. The entire description is shown in figure 5.7, slides for the other topics are found in the Appendix B. Every slide with a topic had the same layout. Most ideas generated in the breakout rooms were noted on sticky notes in the workshop environment. All of them, more or less incomplete, are listed below:

- To have a workflow wizard that guides the user through a workflow.

- To have a floating area on analysis level containing objects relevant for the current page, populated by the author of the application.

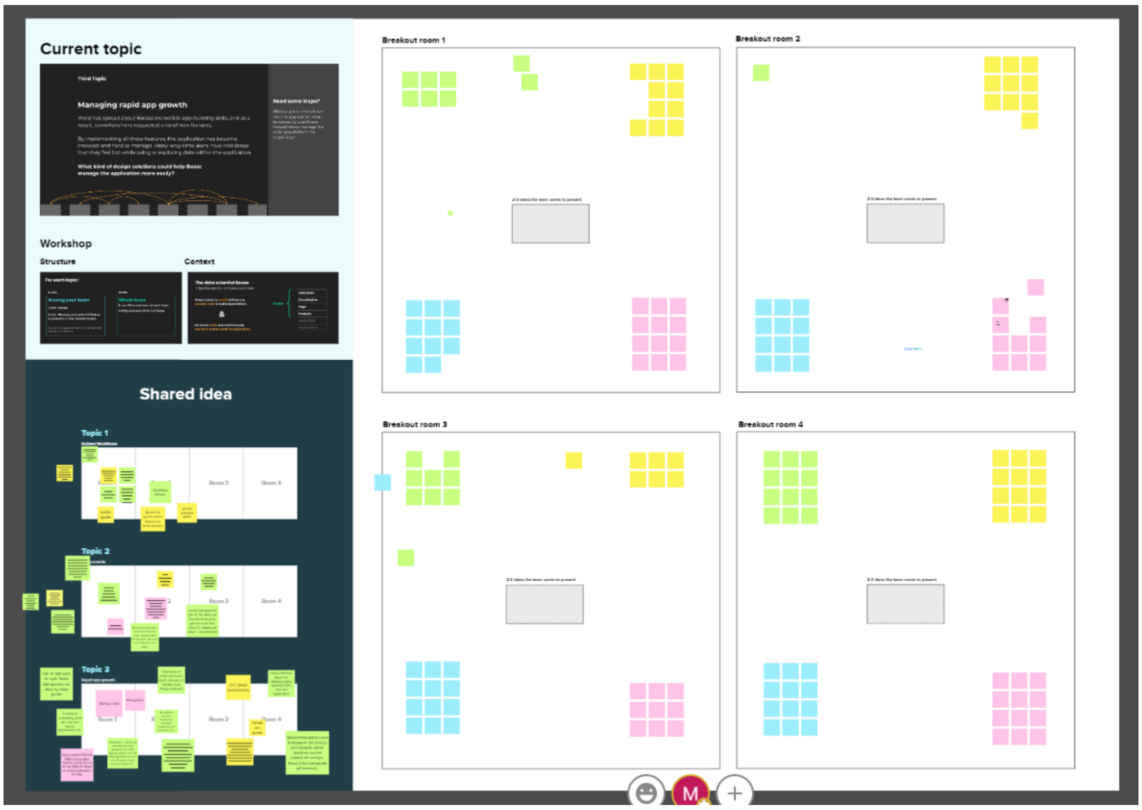
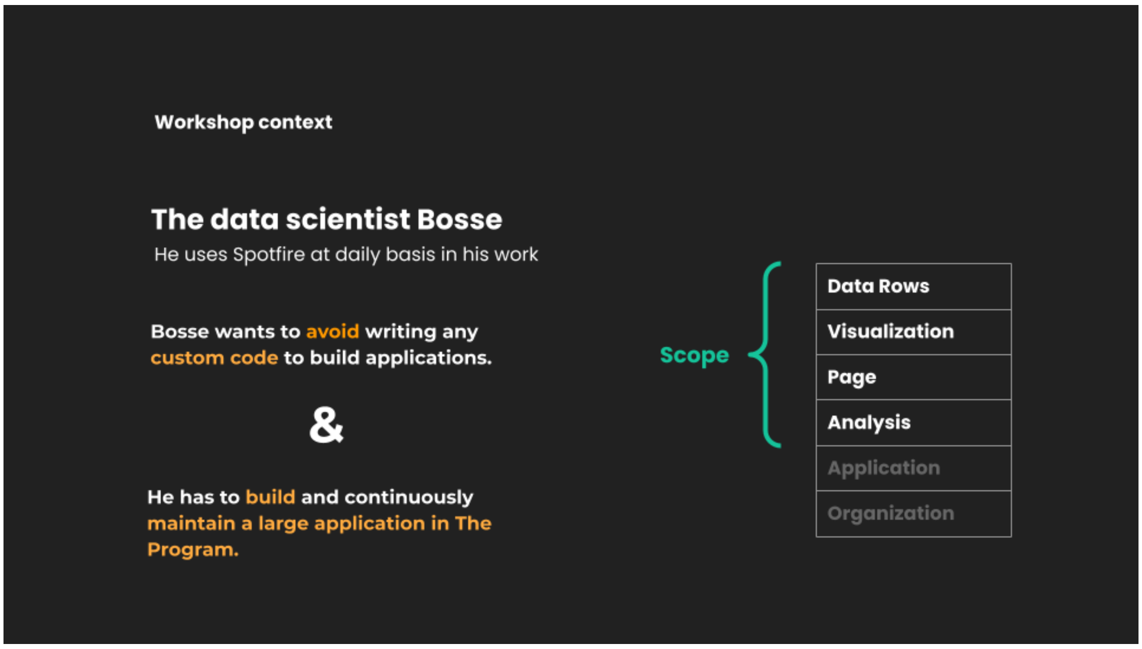


Figure 5.5: The virtual mural board after the workshop was done.

- An audio guide that guides the user through a workflow.
- To be able to test-run a workflow to see it from a user perspective.
- Select some data points in some visualisation and get a wizard that is related to the selection.
- A panel or flyout that is pre-populated with, for example, pages for navigation.

The second topic covered User Interface (UI) components. Bosse wanted to create highly customisable UI components, by other means than scripting. The slide presenting this topic can be found in Appendix B.

- Templates for certain types of dashboards, "if you want to build a finance dashboard, maybe use these visualisations and these components...".
- Reusable components.
- A standardised pattern in components that are encouraging the creator to be consistent.
- General navigational components, common for many types of applications, for example, a global menu.



Workshop context

The data scientist Bosse

He uses Spotfire at daily basis in his work

Bosse wants to avoid writing any custom code to build applications.

&

He has to build and continuously maintain a large application in The Program.

Scope

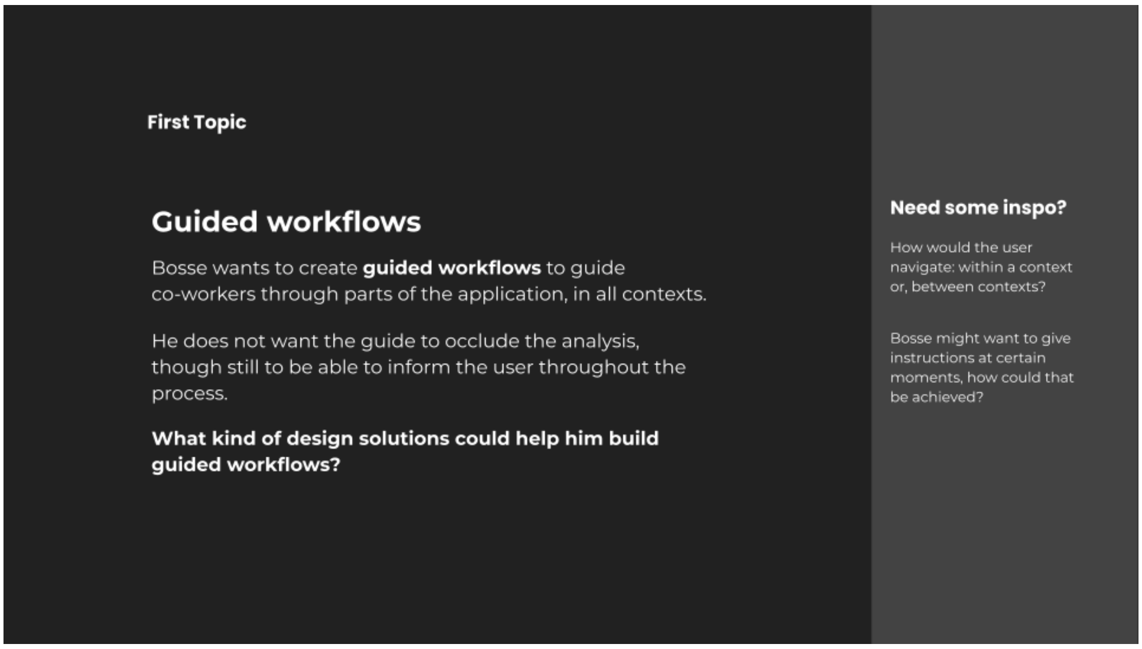
Data Rows
Visualization
Page
Analysis
Application
Organization

Figure 5.6: An introductory slide from the ideation workshop.

The final topic was about how rapid application growth could be easier to manage. One of Bosses' applications had experienced a surge in numbers of features to include. More and more applications users are becoming lost and confused while using it. The slide that presented this topic can be found in Appendix B. Some suggested ideas, more or less complete, were:

- Having an overview of what is built that shows how different pages interact.
- Suggestions and recommendations that analyses the application and suggests how it should be changed.
- Step by step guides on how to use an application.
- Show other large applications and explain how they solved similar issues.

In summary, the workshop served both a validating and ideating role. Many of the ideas brought to the table during the workshop shared similarities with ideas that we had considered before the workshop. This overlap might strengthen their significance. Still, many ideas from the workshop were new and would have been interesting to explore further, for example, to start a workflow from a selection of data. In general, many ideas described the need for a more zoomed-out perspective, mainly in terms of resources during application development. In conclusion, the workshop generated many ideas and significant inspiration for the sketch phase.


First Topic

Guided workflows

Bosse wants to create **guided workflows** to guide co-workers through parts of the application, in all contexts.

He does not want the guide to occlude the analysis, though still to be able to inform the user throughout the process.

What kind of design solutions could help him build guided workflows?

Need some inspo?

How would the user navigate: within a context or, between contexts?

Bosse might want to give instructions at certain moments, how could that be achieved?

Figure 5.7: The first topic of the workshop.

5.2.5 Sketching

Themes for the first sketch phase, built upon the literature, ideation workshop, and conducted interviews. Below is a summary of each theme:

Offer help/information closer to the workspace. Users sometimes request features that already are provided or possible to create in The Product, according to internal stakeholder interviews (section 5.2.2). The Companies community website already contains instructional material and how-to guides related to application building. Bringing this information into the workspace could help users become more aware of its existence.

Recommendations at context(s). One guideline of The Product written by The Company is to create context-aware recommendations and suggestions. Hence, several ideas in the ideation workshop included ideas related to it. This was something that had not been thought of previously, and an exciting area to explore.

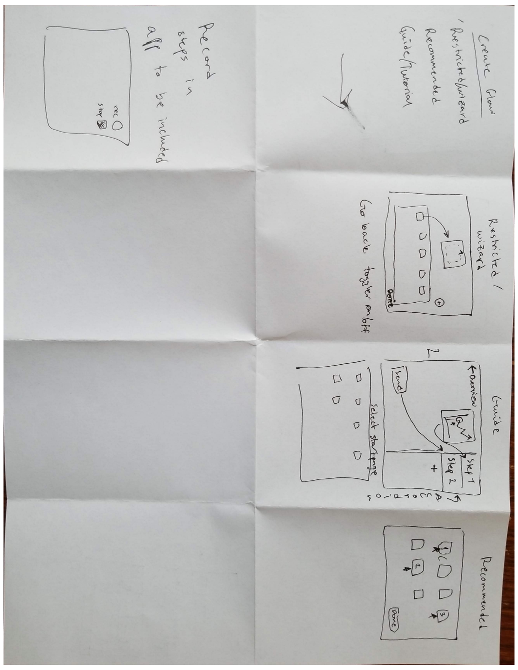
Overview of the relationship between objects. In the current version of The Product, there are no overviews of pages or all visualisations in an application. Several ideas from the workshop related to this topic.

Workflow construction. In the current version of The Product, there is no dedicated mode for creating workflows. This was something discussed during interviews as well as ideated during the workshop, and many of those ideas seemed to have the potential to explore further.

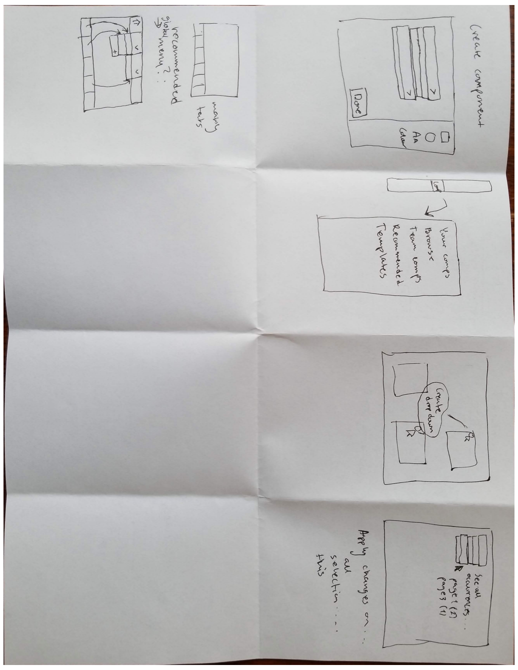
No-code components. An issue discovered during the interviews was that many application builders lack programming knowledge. This theme was tested during the ideation workshop as well, and as workflow construction, it needed more exploration.

Documentation/guidance about how the application works to end-users. This was highlighted as an issue during interviews. Sometimes it can be difficult to mediate what has been done in an application or to convey how it even works. To offer functionality for this in The Product could help solve that.

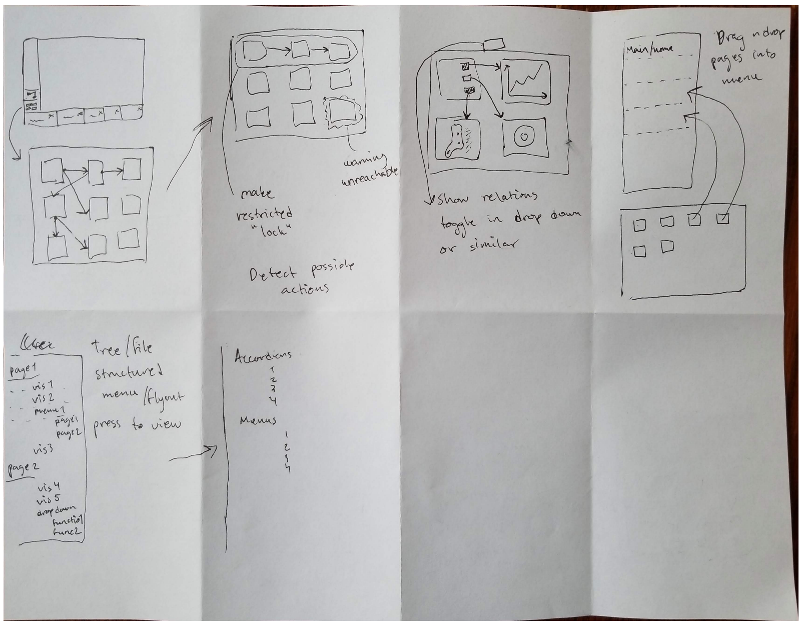
A crazy-8 session (described in section 4.3.3) about each topic was then conducted. Examples of results from such a session can be seen in fig 5.8.



(a) Workflow construction.



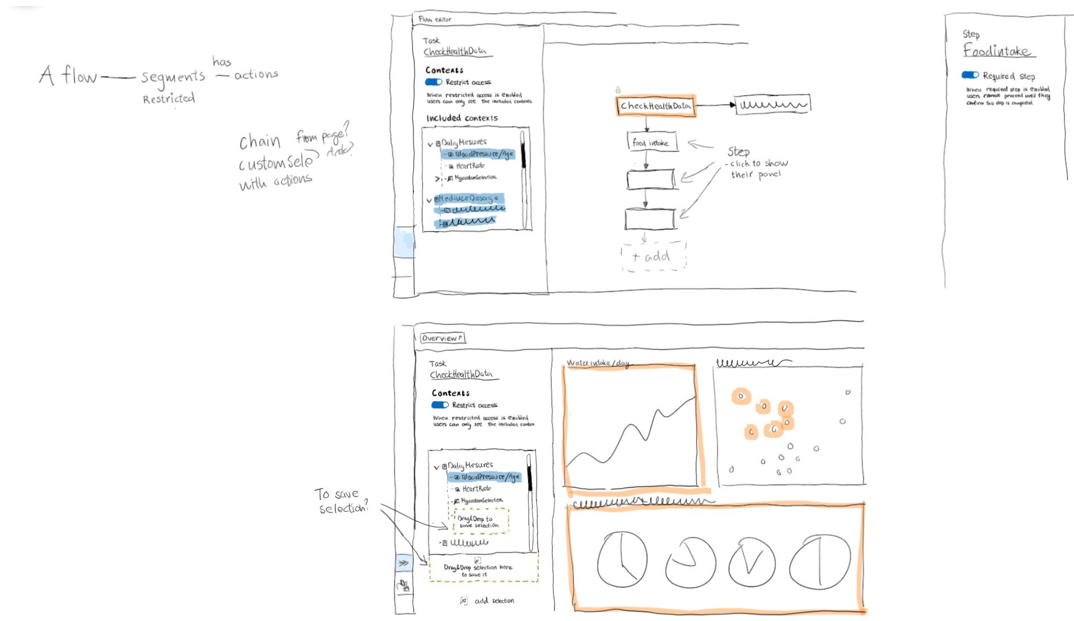
(b) No-code components.



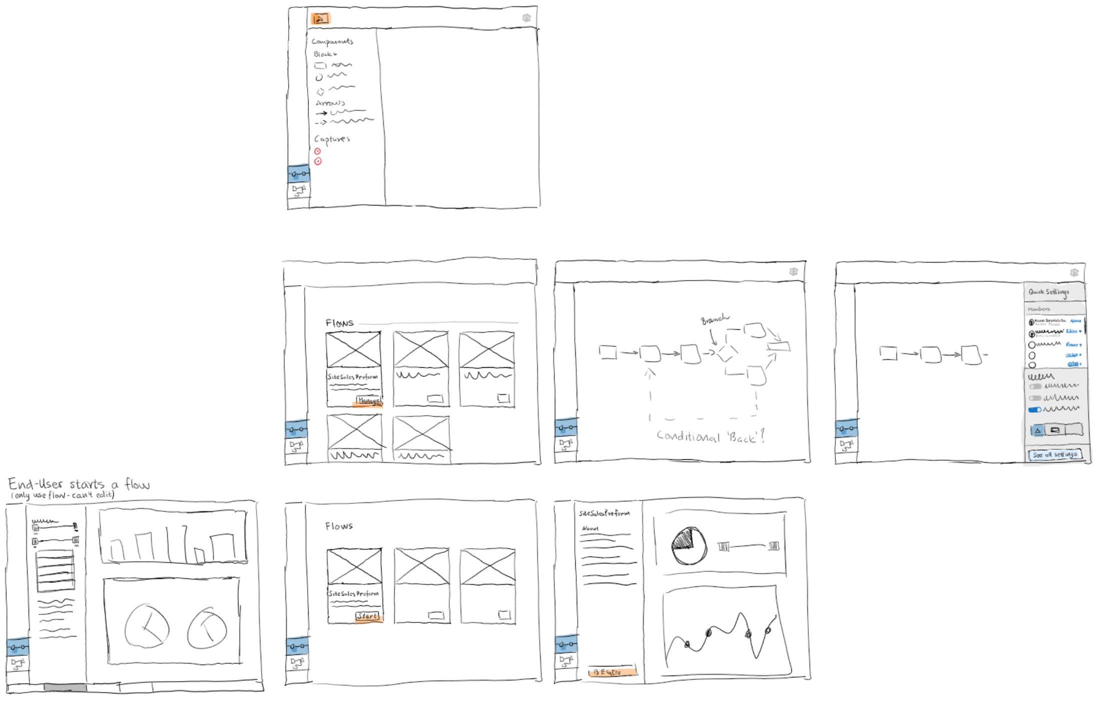
(c) Overview of the relationship between objects.

Figure 5.8: Sketches from different crazy-8 sessions.

After these crazy-8 sessions, the outcomes were discussed. It was decided that some topics may benefit from being combined, and that some topics already are implemented in other products, and would be alike if realised here. Hence, spending time exploring them further would not yield substantially different results. After crazy-8, solution sketching was performed. The topic chosen for solution sketching was "Workflow construction", which was later abstracted to be simply "Workflows". The term workflow can be read more about in section 5.1.5. An example of a solution sketch can be seen in fig 5.9.



(a) How to put actions related to visualisations in a ”workflow-panel”, which is placed at the left side of the screen.



(b) Overviews of construction and execution of workflows

Figure 5.9: Solution sketches for workflows.

5.2.6 Guidelines v.1

After iteration one and feedback from the mentors at The Company, the direction of the thesis had a slight change towards the workflows of an application, and how these can be constructed by an application builder. The purpose with the shift was to narrow the focus of the thesis. With this said, we still considered the entire application building process, but the prototypes and the guidelines relating to workflows would be tested more extensively. An important note is that in these early guidelines, “User” can refer to both application builders and the users of their applications. The new guidelines were stated as follows, with indifferent numbering from G1 to G11.

G1 **The user should not feel lost in an application**

The interviews studied in section 5.1.2, showed that applications can suffer from feature creep and contain more than 50 pages. As the application grows, it can become hard for application builders to maintain the desired navigation. This guideline aimed to provide recommendations that simply the creation and usage of navigational components. According to information architecture, for example, a ”You are here”-marker and letting the application contain an overview section can help the user navigate.

- Provide a ”Your are here”-marker.
- Provide an overview of the application.

G2 **The user should be able to achieve a desired look and feel**

It was discovered from the interviews that some design alterations require quite a bit of custom code. According to end-user development (EUD), section 3.4, developing an application as a user becomes more approachable when it does not involve direct programming. It was also discovered from the interviews conducted during iteration one (see section 5.2.3), that the users of The Product highly values customisation. To combine these aspects, the guideline emphasises design functionality that does not require a lot of custom code.

- Provide a curated set of default application (non-visualisation) components.
- Allow application builders to create their own components.
- Styling actions should be available at many levels.
- Offer previews for visual changes.
- Integrate Web Content Accessibility Guidelines (WCAG) for improved readability.

G3 **For the design of the applications structure: keep the required amount of non-domain coding to a minimum**

As mentioned in G2, the application building process becomes more approachable for some users when they have to write less code. It was also highlighted during one of the interviews conducted during iteration one, that companies want to hire data scientists to work with data science, not to program other artefacts.

G4 **The Product should support collaboration**

The examined literature shows that the degree of collaboration needed in data science work continues to rise [82]. Building and using custom applications can involve many different domain experts. Analytical software should have extensive support for collaboration since it is a key activity. Besides the need to share insights efficiently, possibly through recorded work sessions, this product is also concerned with how changes by others are conveyed. In the UI, a user should directly see what

Kept from the initial set of guidelines see ”**G7** Support development collaboration”. It was highlighted during one of the user interviews that there are several use cases where an application builder has built something and want to share the process to a colleague, which is not supported today.

G5 **The user should able to create custom workflows without the need to code**

The Product should have graphical user interface (GUI) functionality which supports the creation of workflows.

From the interviews, it was discovered that to construct a workflow is a common use case today. This functionality is not supported in the current version of The Product, but it can be made by different workarounds, often referred to as ’hacks’.

G6 **The Product’s ontology must be designed with consideration to the user**

The ontology describes the objects in a system, their ”size” and their relationship with each other. For example: for a clothing site the smallest object could be Women clothing, or it could be split into ”women’s blue socks, women’s red socks” and so forth. Consider the mapping between actions and objects, for example a button could have the following hierarchy: a button - all buttons on a page - all buttons in an application. Most likely the user wants to do certain actions at all the different levels.

This guideline was added as the course of the thesis shifted towards workflows. With that shift, information architecture as a topic was discovered and studied. Ontology is a part of information architecture, which can be read more about in section 3.7.

G7 **Identify the different scopes that users want to apply an action to and offer the opportunity to apply actions to each scope, including a single instance**

The user should be able to apply changes at group-level, for related objects.

This guideline was discussed during the interviews, see section 5.2.1. When it comes to application building, the current version of The Product lacks this kind of support.

G8 **The dashboard screen estate is precious, it should primarily contain visualisations**

Allow users to store information about the dashboard in collapsible panels beside the dashboard.

It was discovered during the interviews in section 5.2.1 that the screen real-estate of an application is precious. The application builder wants to emphasise the visualisations in the application the most. Therefore, functionality not directly related to the visualisations should be able place somewhere it can be hidden.

G9 **Some users will always want to code: follow best practices for code development environments**

Kept from the initial set of guidelines but reformulated, see "**G4** Provide an approachable API of The Product". It was reformulated to not only focus on the API. Creating a user-friendly API is a large topic that concerns software engineering more than interaction design, at least in the context of this thesis. Therefore, focusing on the API was put out of scope.

G10 **The distance between program functions that are often used together should be short**

- If the user clicks on an application object, styling objects should be in close proximity, floating beside the object.

- Avoid having users to navigate frequently between related workspaces.

G11 **The user should be able to navigate, with ease, within and between program contexts and in The Product**

- Provide the ability to search globally and in local content-rich contexts.
- The different ways "backwards and forward" (BaF) navigation in a digital system is behaving should be considered.
- The BaF navigation should be implemented consistently throughout The Product.
- Allow the user to generate a table of content: or indexes of all included pages inside an application.

Kept from the initial set of guidelines but reformulated, see "**G2** Provide developers with flexible and informative functionality that supports the creation of the desired navigation structure within an application". It was reformulated to put more focus on how the user can navigate, than that it should be possible to create a good navigational structure.

5.3 Iteration Two: Sketching and Prototyping

The goal of iteration two was to continue exploring the ideas from the solution sketching session. As a first step, different low-fidelity prototypes were created using Balsamiq and Miro. These tools can be read more about in section 4.4. The mentors at The Company gave feedback regarding the prototypes. Afterwards, high-fidelity prototyping in Adobe XD began. These prototypes were the topic of a design critique session with a data scientist at The Company. To conclude the iteration, the prototypes were tested against the guidelines from iteration one, which can be found in section 5.2.6. A heuristic evaluation on the prototype was performed with the guidelines as heuristics. A description of a heuristic evaluation can be found in section 4.3.6.1. That evaluation led to reformulations of the guidelines, and the new version can be read more about in section 5.3.5.

5.3.1 Compiling Identified Issues

At this part of the process, multiple software issues and user-needs had been identified. We wanted to merge insights from literature research, interviews, and our studies of The Product, to find new patterns in the data. We tried to create a mind map to organise the data. The first attempt failed, it can be seen in 5.11. We could not identify distinct themes. The second version used a tree structure, it made organisation easier. These mind maps served as support when constructing the guidelines in this iteration, described in section 5.3.5.

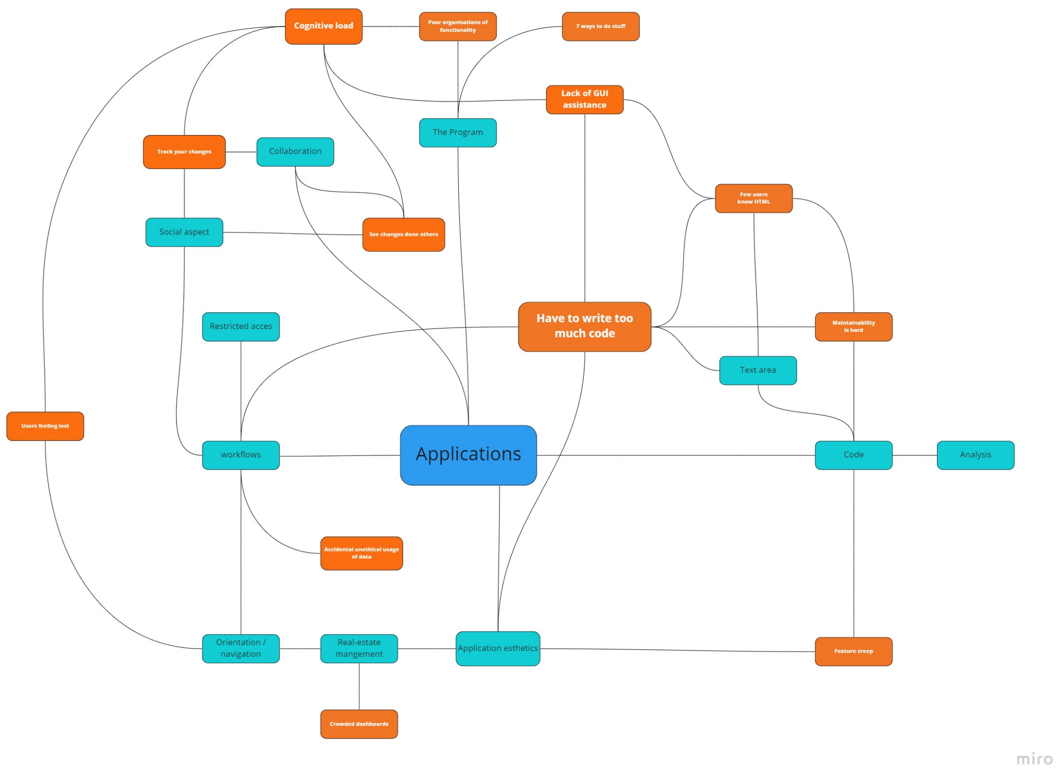


Figure 5.10: The first attempt at a mapping of the identified issues related to application building in The Product.

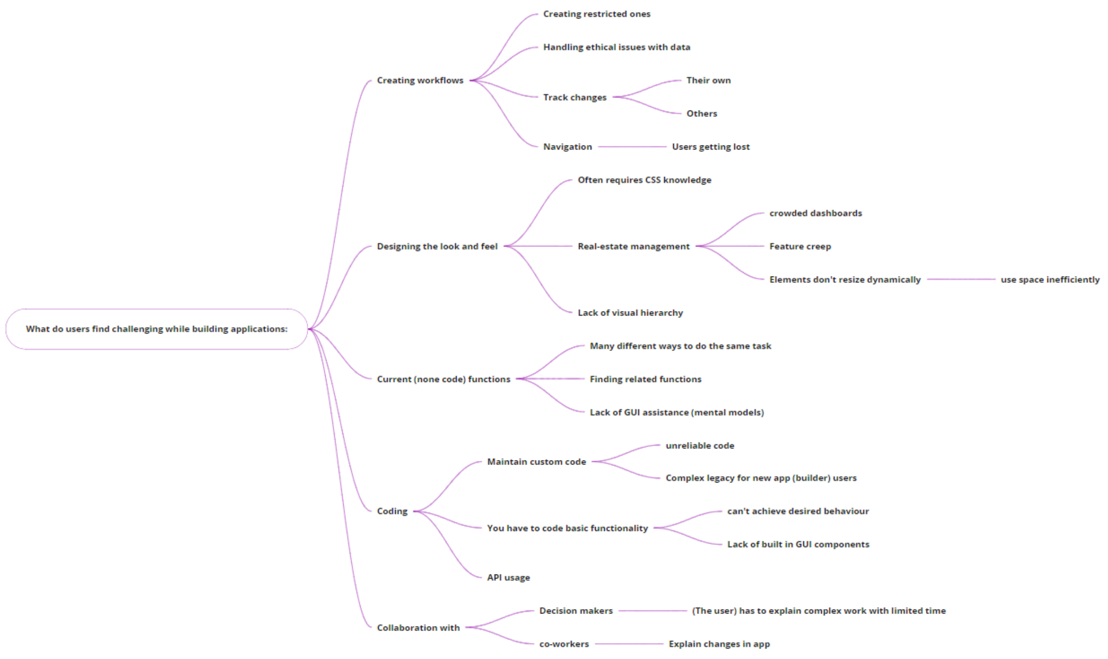

What do users find challenging while building applications:
Creating workflows
Creating restricted ones
Handling ethical issues with data
Track changes
Their own
Others
Navigation
Users getting lost
Designing the look and feel
Often requires CSS knowledge
Real-estate management
crowded dashboards
Feature creep
Elements don't resize dynamically
use space inefficiently
Lack of visual hierarchy
Current (none code) functions
Many different ways to do the same task
Finding related functions
Lack of GUI assistance (mental models)
Coding
Maintain custom code
unreliable code
Complex legacy for new app (builder) users
You have to code basic functionality
can't achieve desired behaviour
Lack of built in GUI components
API usage
Collaboration with
Decision makers
(The user) has to explain complex work with limited time
co-workers
Explain changes in app

Figure 5.11: A mapping of the identified issues related to application building in The Product.

5.3.2 Low-Fidelity sketch of Workflow and Components

For the low-fidelity prototyping in iteration two, the tools Balsamiq and Miro (described in section 4.4) were used. Balsamiq was chosen as it only offers black and white wireframes, which attempt to mimic a sketch drawn by hand. When interpreting such wireframes, it is simpler to focus on the structure and the concept behind the wireframes, not details like colours. Similar wireframes were created in Miro, but with colour to get a more vivid feeling. The concepts in focus were initially workflow creation and to create application components without code.

The component-based concept. The idea behind this concept is that application building becomes more approachable if the application builder does not have to program their application, but to create the application components with a graphical user interface (GUI). The wireframes can be seen in fig 5.12.

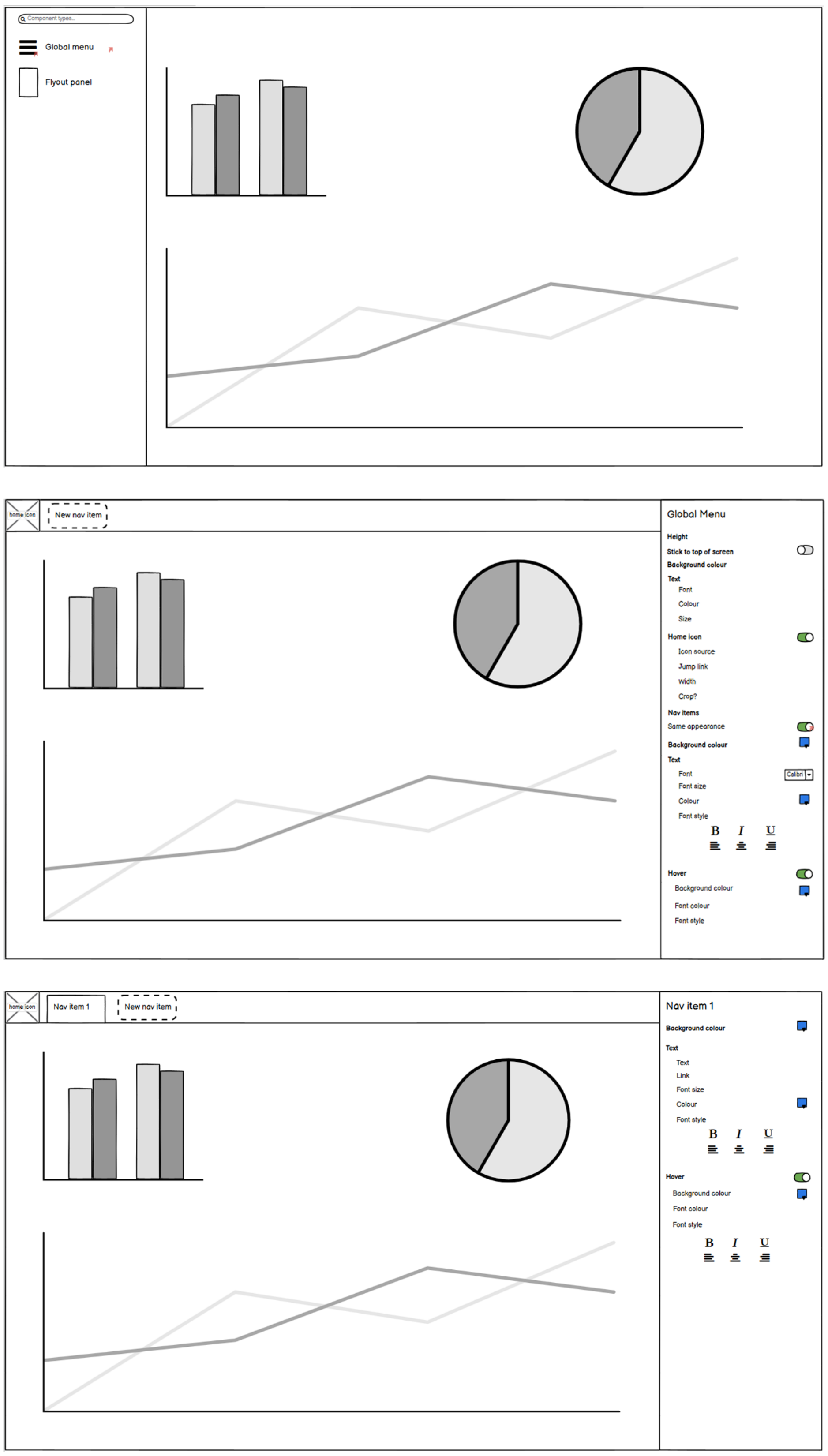


Figure 5.12: The component wireframes made in Balsamiq, starting at the top.

The top-most wireframe in fig 5.12 is the beginning of the component-based concept. The application builder is provided with a side panel to the left, which contains different components. The components can be everything from navigational components, like different types of menus, to simple text. They can be dragged and dropped onto the application where they act along with the visualisations.

In this example, the application builder chooses the global menu component and drags and drops it to the top of the dashboard (the middle wireframe in fig 5.12). As this is a global navigational component, the idea is this component will be placed at the same position throughout the application. When the global menu is placed and selected, its settings, which are placed in a side panel to the right of the screen, are visible. The application builder can only configure these settings.

In the nethermost wireframe in fig 5.12, the application builder can configure specific parts of the global menu by double-clicking them. Here, the application builder has double-clicked the “New nav item”-button that has a dotted outline in the middle wireframe in fig 5.12. As can be seen in the nethermost wireframe, the left side panel’s content changes, as it is the chosen tab’s settings, not the whole menu’s. This was performed as a way of accommodating the application builders’ customisation requirements.

This concept manages to encapsulate several guidelines but from different angles. The following are the guidelines considered during the construction:

- **G1** The user should not feel lost in an application. The global menu that the application builder can create in this prototype creates an application that is navigated easier. The idea is that the global menu should be able to indicate where the user currently is in the application. Consequently, acting like a ”You are here”-marker for the application user.

- **G2** The user should be able to achieve a desired look and feel. The global menu is highly customisable in both behaviour and appearance, see the right panel in the middle and lowest wireframe in fig 5.12.

- **G3** For the design of the application’s structure: keep the required amount of non-domain coding to a minimum. The global menu is created in a WYSIWYG style. Hence, limiting the required amount of code to a minimum.

- **G8** The dashboard screen estate is precious, it should primarily contain visualisations. The global menu is small, and it can be set to be invisible on some pages in the settings.

- **G11** The user should be able to navigate, with ease, within and between program contexts and in The Product. As mentioned, having a global menu in an application facilitates the navigation for the application user.

The workflow builder concept.

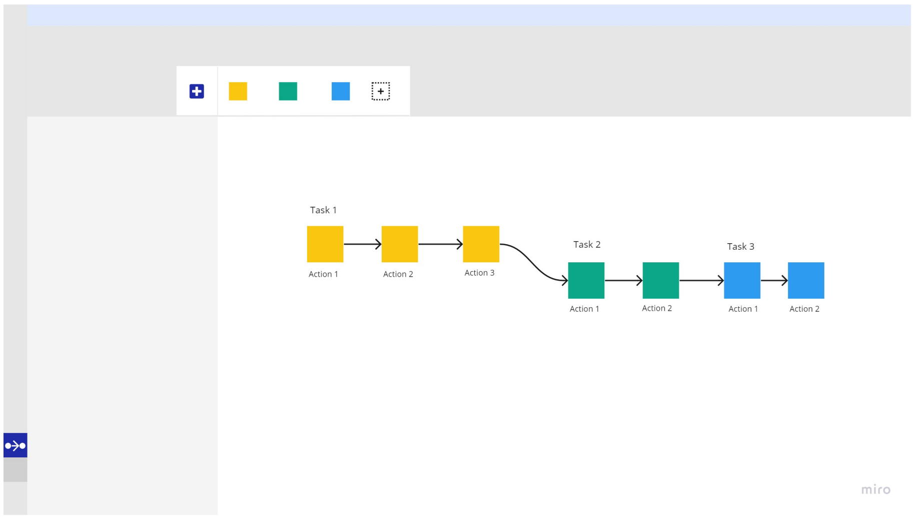
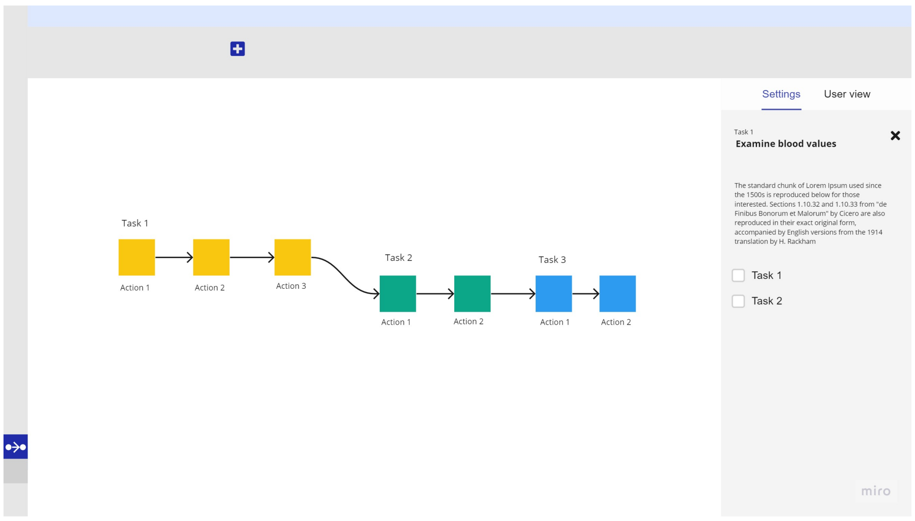


Figure 5.13: An early sketch of the workflow concept.

As previously mentioned, The Company had already identified a possible need for workflow functionality before the creation of this workflow concept (fig. 5.13). Their idea was still at a highly conceptual level. In essence, they informed us that the possibility to create sequenced processes in an analysis could be of use for many users and that it had to be flexible. It also seemed likely that a workflow concept could demonstrate and support many of the guidelines at the time. A more elaborate description about the workflow concept can be found in section 5.1.5.

The most influential source of inspiration for our concept was one of the data scientists we interviewed during iteration one, described in section 5.2.1. He briefly showed us the workflow-similar functionality in another product that The Company offers, as an example of he enjoys working. In this builder, users placed connectors between blocks on the canvas sequential processes. Each block represented a script or a function. A screenshot of the data science tool can be found in section 5.1.5.

A quick probe confirmed that Data Scientists generally use custom workflows frequently in their work. EUD research also found that end-user development is easier when the user finds the development environment familiar. This conclusion, in combination with the inherent flexibility, led us to explore a workflow builder concept. This concept clearly supported **G5**, but it also included additional guidelines. **G1** and navigational support would be considered in multiple ways, the canvas top-down perspective would be one example. The canvas environment would be low code, as **G3** recommends. To let others use the flows would follow what **G4** recommends. Finally, **G4** was also included, and fig 5.14 shows one of the first tests with function placement.



The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham

Figure 5.14: An early sketch of the workflow concept with an open panel on the right-side.

5.3.3 High-Fidelity Prototyping

The concept chosen to explore further with high-fidelity prototyping was the workflow concept. It was chosen as it encapsulates many parts of the application building process, and it can take place in The Product in several ways. For example, the component concept can be realised within the workflow concept as well, by making the workflow creation process component based. Moreover, the component-based concept described in section 5.3.2, is implemented in other applications like WebFlow [92]. Thus, already well-proven.

Adobe XD was chosen as a tool for high-fidelity wireframes, a description of that tool can be found in section 4.4. This tool was chosen, as we are already familiar with it since previous design projects, and it offers collaborative real-time editing.

After feedback from the mentors at The Company, the focus of the workflow concept shifted. It shifted from how the workflows are constructed to how they are manifested for the application user. It was easy to get caught in the details of the workflow creation process. And, as the purpose of this prototype is more about proof-of-concept rather than a complete idea, the prototype benefited from the focus shift.

For the application user perspective of workflows, two different scenarios were created. One scenario has a workflow directly connected to a visualisation, and the other has a workflow that spans over several pages in an application. In both scenarios, the application builder has already built an application in The Product, and created the workflows. They have also decided whether the workflows will be connected to one visualisation or the whole application.

Connected to a visualisation. In this scenario, the application builder has created a workflow that can be performed on a visualisation. They have decided which visualisation it is connected to, as well as what it does. The workflow can be found by pressing the “A” above the bar chart in fig 5.15. By pressing the “A”-button, a menu that contains several workflows appears, see fig 5.16. A workflow can either act on all of the data in the visualisation, or just a selection.

For this scenario, the chosen workflow is “Monitor value”. By pressing that alternative, the menus’ content changes to the actual workflow, see fig 5.17. All this content is created by an application builder, and cannot be changed by an application user. For this workflow, the application builder has made it possible for the application user to monitor the value in the visualisation and execute different actions based on a condition. In this iteration, it was not yet decided whether a workflow always should be called a workflow or an “action”. Therefore, in these figures, the title is “New action”.

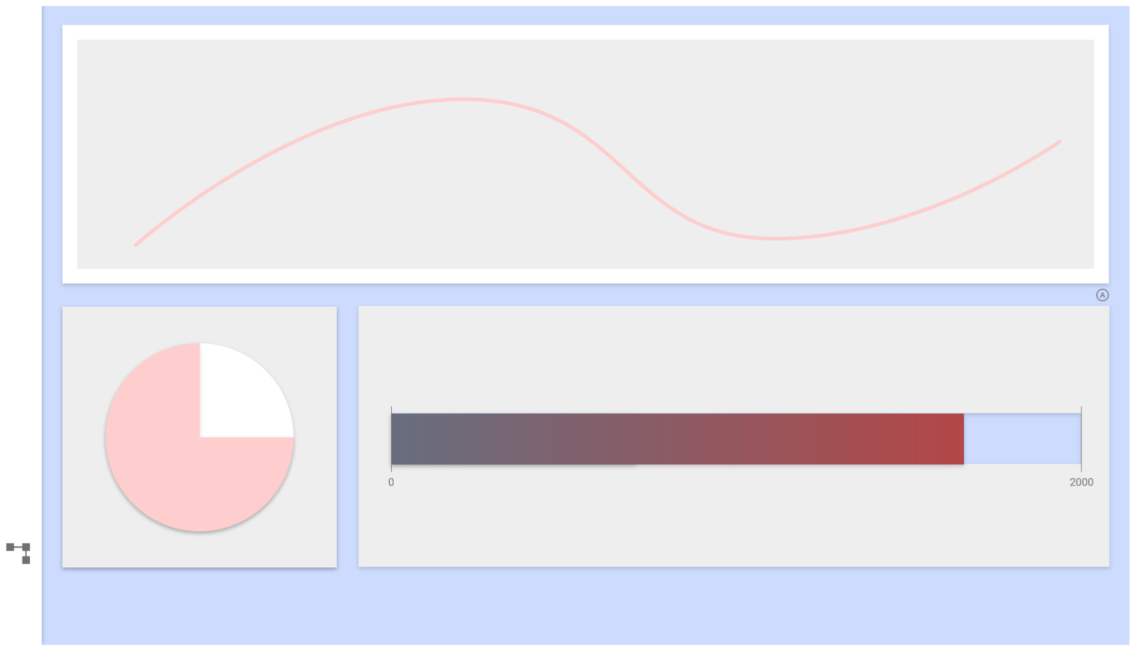


Figure 5.15: The beginning of the workflow connected to a visualisation. The workflow menu can be opened by clicking on the "A" at the top right of the bar chart.

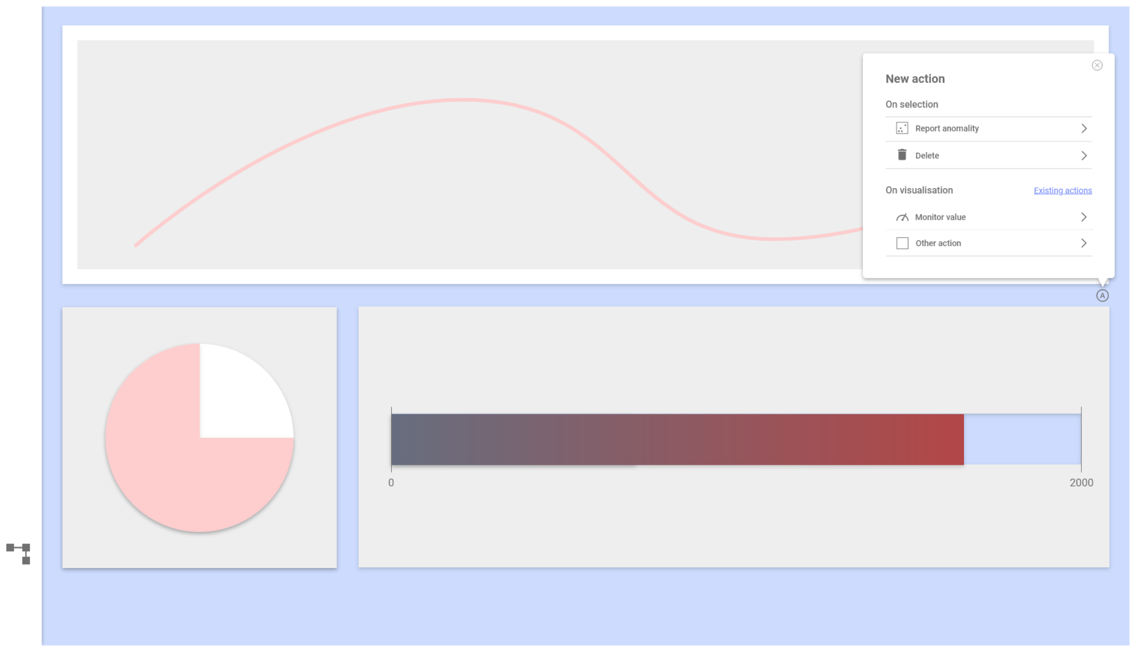


Figure 5.16: The menu containing workflows that appears after pressing the "A" in fig 5.15.

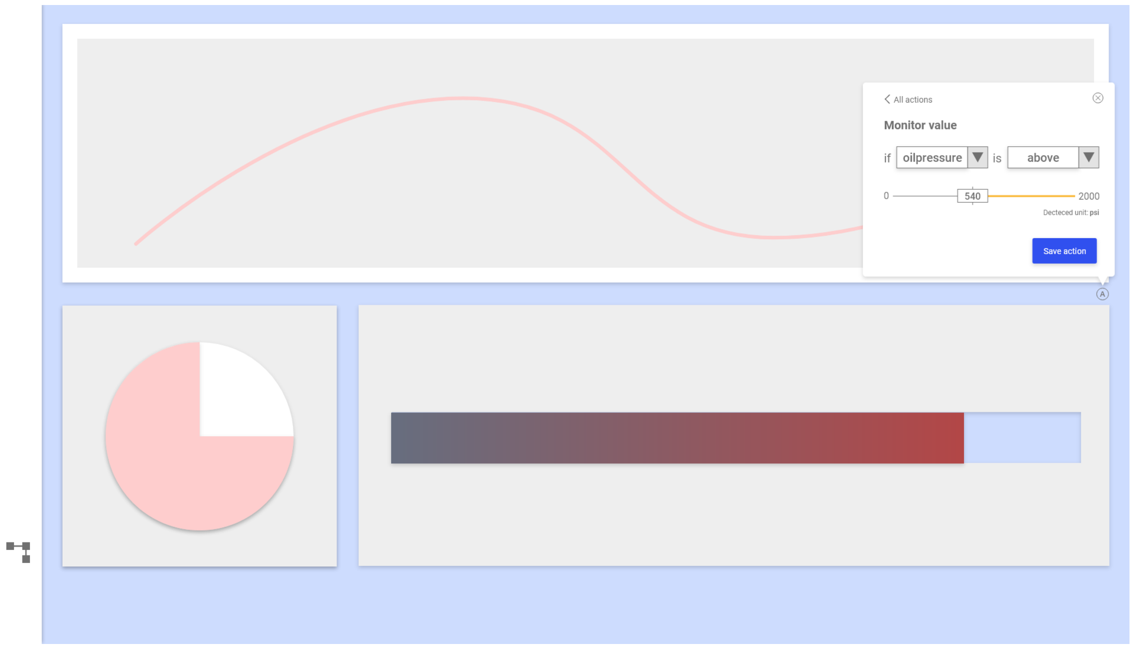


Figure 5.17: The content of the menu in fig 5.16 has changed to the actual workflow that can be performed on the visualisation.

When constructing these wireframes, the following guidelines were considered:

- **G7** Identify the different scopes that users want to apply an action to and offer the opportunity to apply actions to each scope, including a single instance. In this case, the application user can choose to act on the whole visualisation or just some data points. See fig 5.17.

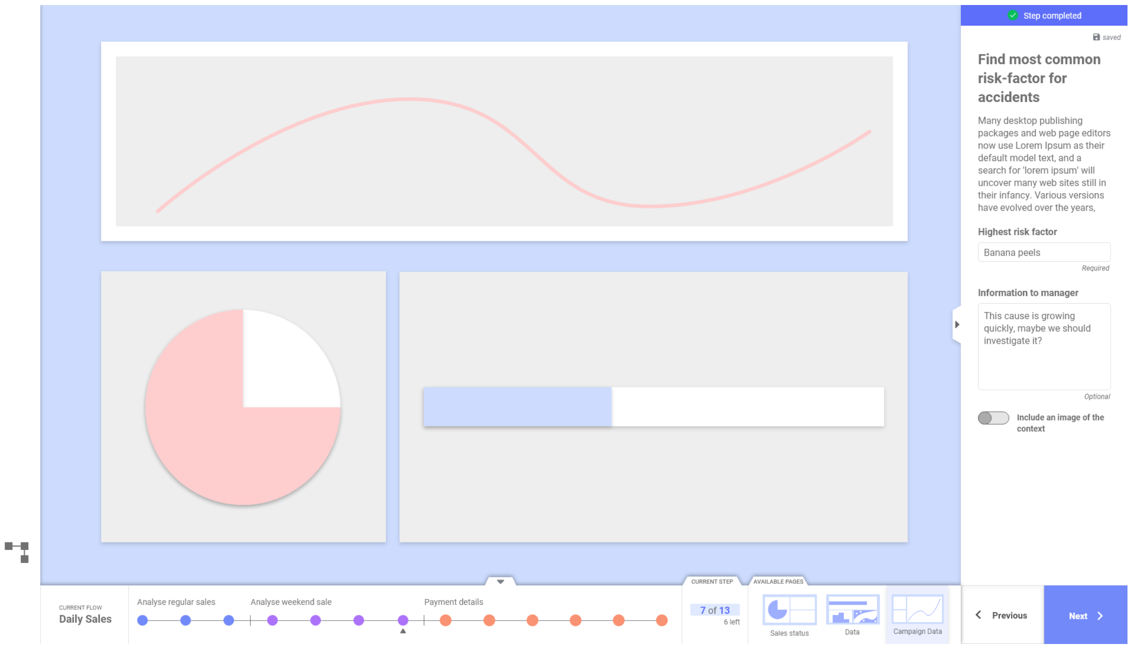
- **G8** The dashboard screen estate is precious, it should primarily contain visualisations. The workflows that can be applied to a visualisation are placed in a hidden menu to not occlude any space.

- **G11** The user should be able to navigate, with ease, within and between program contexts and in The Product.

Including several pages. In this scenario, the workflow spans over several pages in the application. That means that the workflow has a set of tasks that the application user should, or have to, do. These tasks are spread over one or more pages in the application, and in this case, there are several pages. The application user can always view the right-side and the bottom panel throughout the workflow. This can be seen in fig 5.18. To acquire as little screen real-estate as possible, the panels can be hidden by pressing the arrows.

The bottom panel shows where in the workflow the application user currently is. In this example they are currently on step seven, indicated by both a progress indicator and a text. The current step in the process can include several pages. Therefore, a list of the relevant pages for this step is listed to the left of the previous and next buttons. When the step is completed, the user is allowed to press next and proceed with step eight.

The right-side panel contains information about the current step, see fig 5.18. In this panel, the application user can attach or fill in the required information for the current step. This panel is created by the application builder.



Find most common risk-factor for accidents

Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years,

Highest risk factor: Banana peels (Required)

Information to manager: This cause is growing quickly, maybe we should investigate it? (Optional)

Include an image of the context

Figure 5.18: A workflow that spans over several pages.

In this scenario, the following guidelines were considered:

- **G1** The user should not feel lost in an application. As can be seen at the bottom of fig 5.18, there is a marker placed where the user is in the process. There is also a text that tells where the user is with numbers beside.
- **G8** The dashboard screen estate is precious, it should primarily contain visualisations. The panels that belong to the workflow are collapsible, indicated by the arrows on the edges of the panels.
- **G11** The user should be able to navigate, with ease, within and between program contexts and in The Product. The bottom panel shows pages that the user is allowed to go to in the current step. There is also a previous and next button, that takes the user to the previous and next step in the workflow respectively.

Finally, how to set whether a workflow is going to be a workflow that spans over several pages in an application, or if it is attached to a visualisation, was ideated. The lack of overview in an application building perspective in the current version of The Product was perceived as an issue. That, combined with the existing successful data canvas in The Product, led to an idea of providing an overview of all the existing workflows. This overview has a similar appearance as the data canvas to make it familiar to users and stakeholders. This wireframe feeds two purposes; it is the beginning of creating a workflow, and it provides an overview of all the workflows. The wireframe can be seen in fig 5.19.

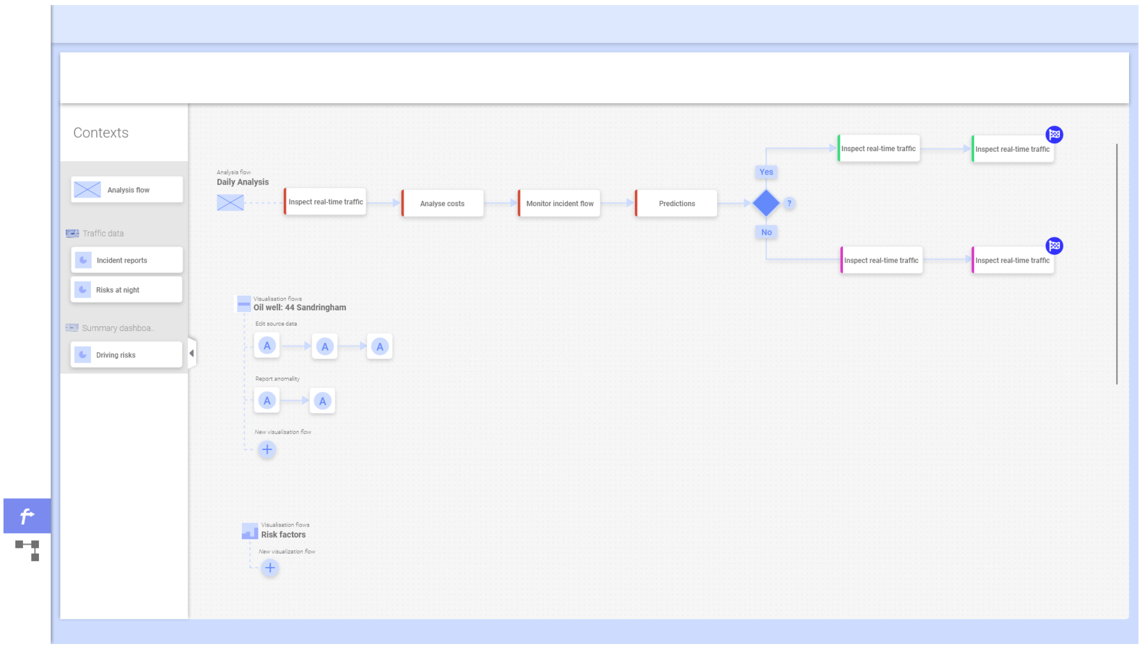


Figure 5.19: An overview of workflows.

As there is currently a clear distinction between two different kinds of workflows, and they are acting on the same overview, they had to be named. The workflow attached to a visualisation is called a *Visualisation Flow*, and the workflow that spans over several pages an *Analysis Flow*.

As mentioned, how to set whether a workflow will be an analysis flow or a visualisation flow can be set in this wireframe. At the left of the screen, there is a panel with different contexts (see fig 5.19). At the top of the panel, the application builder can choose to create an analysis flow. Below, all the visualisations that are a part of the application created earlier are shown. These are all the contexts where a workflow can act.

These contexts can be dragged and dropped onto the canvas to the right of the screen. When a context is dropped onto the canvas, the application builder can start creating the workflow by pressing the button with a plus sign beneath or beside the context.

The guidelines that were considered during the development of this wireframe was:

- **G1** The user should not feel lost in an application. By providing the application builder with an overview of all the workflows and pages in the application, the risk of feeling lost is lower. The “F” symbol at the low left of the screen, see fig 5.19 is also highlighted during the workflow overview mode, acting like a “You are here”-marker for the application builder.

- **G11** The user should be able to navigate, with ease, within and between program contexts and in The Product. By providing the application builder with a set of contexts that has a clear distinction, the application builder can easily switch between the visualisation and application context.

To summarise the prototyping phase, the workflow concept has been developed further and from different perspectives. The parts that have been explored are how it can look and behave for the application users, and how the application builders can set the type of workflow.

5.3.4 Design Critique with a Data Scientist

The main purpose of this interview was to receive feedback about the usefulness of workflows as a feature in The Product. Currently, the interviewed data scientist works as a consultant at The Company and builds applications. Prior to this position, he used The Product on a daily basis as a customer. His past and current experiences allowed him to provide rich feedback.

The interviewee stressed that the workflow concept had to be flexible. According to him, this characteristic is important since it is impossible to foresee all possible use-cases for a concept of this size. Worksflows must be customisable, rather than solving specific use-cases. We showed him how the workflow canvas with the drag and drop blocks works. He was positive about it and recognised it from other software he had used in his role as a data scientist. According to him, most data scientists have used a canvas with blocks and connectors in their work. Besides the inherent flexibility in this approach, he found it crucial to include flexibility in all aspects of the design. When asked if he preferred visualisation flow panels to be inline or floating, the essence of this answer was: Let users decide. In addition, he said that the entire experience seems visual. He appreciated this since he often visualises workflows when thinking about them. In summary, the prototype aligned with his mental model of workflows.

The interview influenced coming work with the prototype. We decided to let the application builder choose if the panel should be inline or insight. Until work with the prototype was finished, we reminded ourselves that users should decide.

5.3.5 Guidelines v. 2

At the end of iteration two, the guidelines from iteration one were tested and evaluated against the newly created prototypes. This was performed as a heuristic walkthrough, with the guidelines as heuristics. A description of heuristic walkthroughs can be found in section 4.3.6.1. As the main focus of iteration two was workflows, the guidelines were adjusted and reformulated to better fit workflows. They are still concerning applications as a whole, but they are more thoughtful in the context of workflows. In this iteration there is still ambiguity regarding the term “user”, meaning that “user” can refer to both the application builder and the application user. When the guidelines were created in Miro, an attempt of creating tags that were either directed towards The Company or the application builders was made. The guidelines from Miro can be seen in Appendix C. The reformulations in these guidelines were made as a way of clarifying the purpose of the guideline. The new guidelines were stated as follows:

G1 **Consider existing guidelines for User Experience.** Follow Nielsen’s 10 heuristics [89] and consider publishing The Products existing principles.

This guideline was added as an attempt to not write other guidelines regarding user experience that are not specific for applications that are built in an analytics platform.

G2 **Strive to empower domain experts as much as possible without requiring them to write custom code.** The Product should consider incremental learning.

This guideline is based on “**G3** For the design of the applications structure: keep the required amount of non-domain coding to a minimum.” from guidelines v.1 and “**G6** Customisation functionality should consider The Gentle Slope of Complexity” from guidelines v.0. It was reformulated to cover more cases of supporting the users that do not write custom code.

G3 **Some desired behaviours will always require coding to be accomplished.** Consider making it possible to integrate external systems inside The Product / connect to external systems. Consider to make pre-made components customisable through code.

This guideline was kept from guidelines v.1 but reformulated, see “**G9** Some users will always want to code: follow best practices for code development environments.” It was reformulated

G4 **‘Offer features on demand’ / Different sections with functionality should be flexible (resize, hide/collapse).** Hick’s law: the time it takes for a person to make a decision as a result of the possible choices he or she has: increasing the number of choices will increase the decision time logarithmically. Consider enabling the users to put their content both integrated in visualisations and dashboards, as in collapsible menus.

This guideline was kept from guidelines v.1 but reformulated, see “**G8** The dashboard screen estate is precious, it should primarily contain visualisations.” It was reformulated to focus more on how screen real-estate can be saved, rather than conveying that it should be saved "somehow".

G5 **The system should strive to always keep the user (passively) informed.** Analytical software is often large and feature rich. Useful information during a use case should be kept as close as possible to the current workspace. Offer modeless confirmations after an action (Saved etc.) Communicate a users position within process and its status.

This guideline was kept from guidelines v.1 but reformulated, see "**G11** The user should be able to navigate, with ease, within and between program contexts and in The Product." It was reformulated to cover more cases than just navigation. This guideline emphasises the importance of keeping the user informed in every way, rather than just in the navigational aspects.

G6 **Consider the behaviour of objects and the information presented about them (reusability).**

This guideline was kept from guidelines v.1, see "**G7** Identify the different scopes that users want to apply an action to and offer the opportunity to apply actions to each scope, including a single instance." It was reformulated to cover all the information that users get about objects, not just what actions can be applied and in which scopes.

G7 **Consider the relationship between tools and objects.**

This guideline was kept from guidelines v.1 but reformulated, see "G6 The Product's ontology must be designed with consideration to the user." It was reformulated to focus more on the relationship between tools and objects and not what the tools and objects are.

G8 **Consider keeping relevant actions as close as possible to an object.**

This guideline was kept from guidelines v.1 but reformulated, see "**G10** The distance between program functions that are often used together should be short." It was rewritten for clarity.

G9 **Consider to support the possibility to replicate and record activities done in relation to an analyses.**

This guideline was kept from guidelines v.1 but reformulated, see "**G5** The user should able to create custom workflows without the need to code." It was reformulated to not only focus on workflows. It was discussed during the user interviews in section 5.2.1 that it is sometimes hard to convey to others what has been done in an application and why. Therefore, being able to do that in some way is covered in this guideline, besides just workflows.

G10 **Program users seldom work alone, consider the needs users have as a group**

This guideline was kept from guidelines v.1 but reformulated, see "**G4** The Product should support collaboration". It was reformulated to focus on the users as a group, rather than to support collaboration in The Product in "some way".

G11 **Change by others should not come as a surprise for a users, communicate changes by default.** As an application grows, change should be visible from many levels, and especially from an overview perspective (summary of changes since last usage etc.)

This guideline was added in this iteration as a consequence of the guidelines regarding collaboration.

G12 **Many users are not professional designers and could benefit from context aware help (with design).** To avoid crowding applications with features, consider to provide pre-made templates and suggestions on how to structure large applications. Offer help in close proximity to where the user currently works. When offering help, avoid having users feel incompetent. Consider language such as "a suggestion to find out more" etc.

This guideline was kept from guidelines v.1 but reformulated, see "**G2** The user should be able to achieve a desired look and feel." It was reformulated as we got feedback from The Company that the users already can achieve their desired appearance. This puts more focus on the fact that many users are not professional interaction designers, and would benefit from support in those manners.

5.4 Iteration Three: Create a Coherent Concept

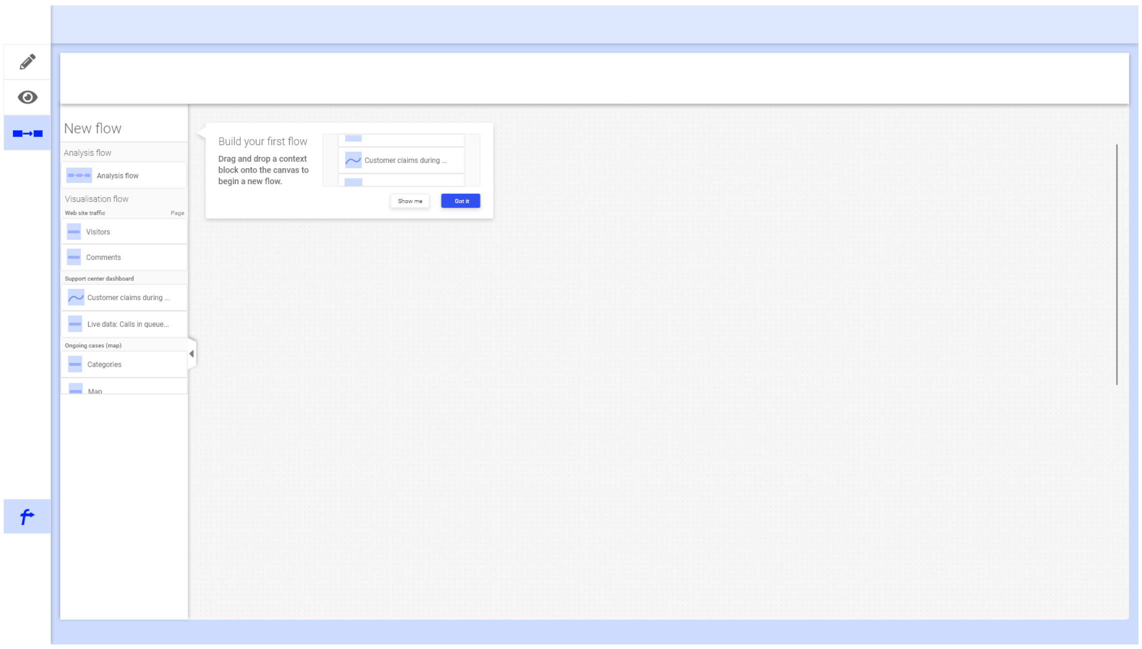
The purpose of the final iteration was to finalise the concept of workflows while also incorporating most of the guidelines into the final prototype. At this point, the workflow concept had been worked on from several different perspectives. The missing pieces were how all the perspectives act together in a final prototype. The last perspective that ties all the previous prototypes together is how a workflow can be constructed, which is described in section 5.4.1.

After the last perspective of the workflow concept had been constructed, the prototype was tested with users, described in section 5.4.2. After that, a design critique session with a product manager and an interaction designer was conducted, described in section 5.4.4. All the feedback and takeaways from the mentors at The Company, user tests, and design critique were taken into account when creating the last version of the guidelines, see section 5.4.4.

5.4.1 High-Fidelity Prototype of the Workflow Mode

In the last iteration of the prototypes, there was a need to create a coherent scenario for creating and performing a workflow. The missing pieces were how the process of creating a workflow is manifested. At this stage, how the workflows can be executed by an application user and how the application builder can manage and set what kind of workflow they want have been explored. This last prototype was meant to glue the pieces together. In this final scenario, the application builder has already built an application in The Product but not created any workflows yet.

The creation of a workflow starts at the overview of workflows, which can be seen in fig 5.20. When the application builder enters the workflow mode for the first time, they are prompted with a dialog that contains information about how to use the canvas. If the application builder chooses to press "Show me", an animation that drags a context block from the side panel onto the canvas is executed. If they press "Got it", the dialog will disappear.


Build your first flow

Drag and drop a context block onto the canvas to begin a new flow.

Figure 5.20: When the application builder enters the workflow mode for the first time.

To start building a workflow, a context from the left panel is dragged onto the canvas. When the context is dropped, a plus sign will appear beneath or beside the context, see fig 5.21. When the plus sign is pressed, the application builder gets redirected to the edit mode of a workflow, see fig 5.22. An important notice is that the left panel changes when changing mode in the workflow mode. For example, in this change of mode, the icon that is emphasised changed from the overview to the pen, see fig 5.21 and 5.22.

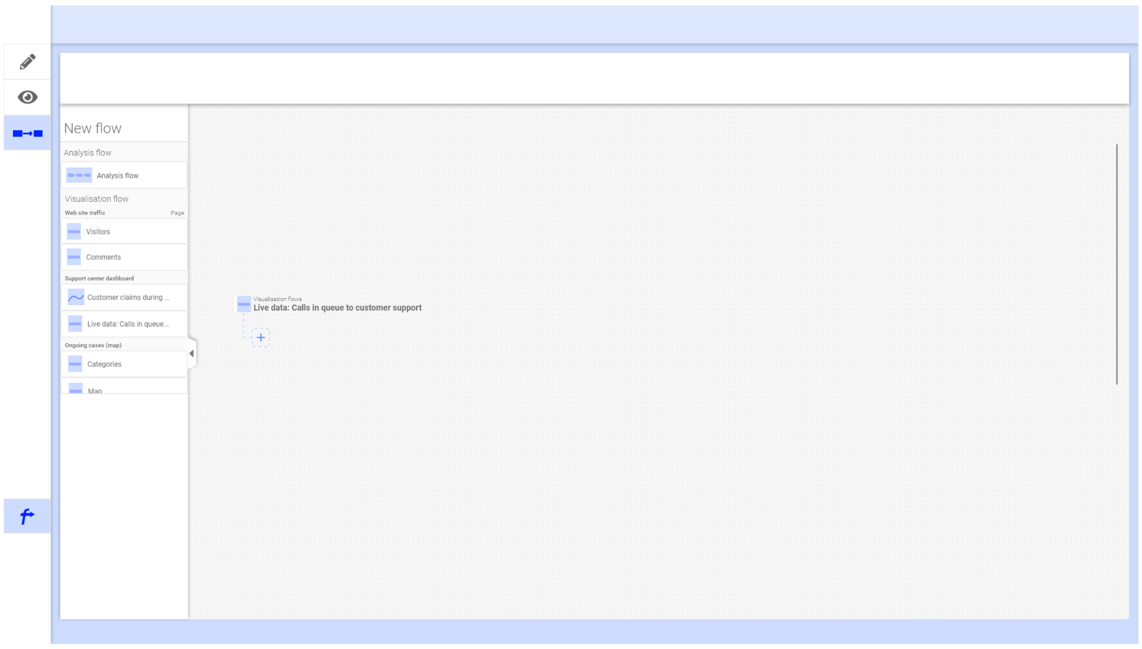


Figure 5.21: A visualisation is chosen as a context for the workflow.

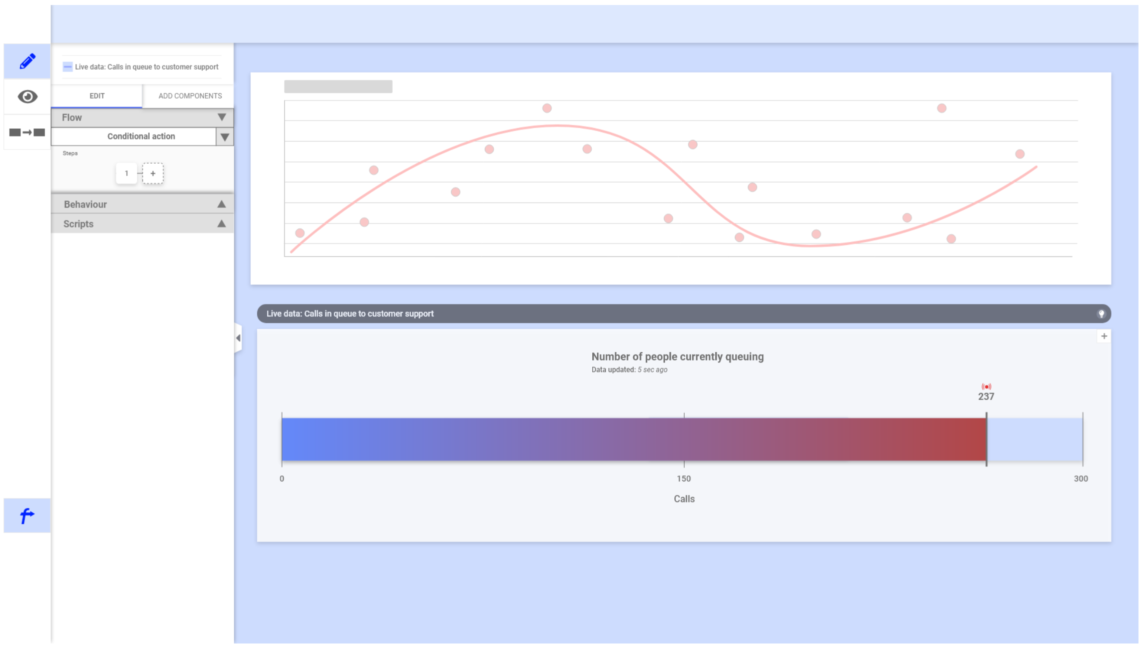

Live data: Calls in queue to customer support
EDIT
ADD COMPONENTS
Flow
Conditional action
Steps
Behaviour
Scripts
Live data: Calls in queue to customer support
Number of people currently queuing
Data updated: 5 sec ago
237
0
150
300
Calls

Figure 5.22: The workflow creation view.

The idea is that the application builder should be able to create a visualisation flow through a WYSIWYG interface. The left-side panel seen in fig 5.22 contains two tabs; one for information related to the currently chosen visualisation flow, and one for adding new components in the visualisation flow. The currently chosen visualisation flow can be changed via a drop-down menu.

When adding components to the visualisation flow, the desired component is first chosen from the panel and then dragged and dropped onto the blank canvas. This can be seen in fig 5.23, where a “Heading”-component has been chosen. The idea is that some default components should be provided by default, as text and buttons. If the application builder requires more specific or specialised components, they can create them by themselves. This has not been explored in the prototypes though, but one idea was to provide a WYSIWYG interface for that as well. In fig 5.24, the application builder has continued the visualisation flow and created some own components, which is reflected in the left-side panel. All the custom components can be shared via the library to avoid repetition and duplication. In fig 5.25, one can see that the components that have been placed in the visualisation flow are reflected in the hierarchical list in the left panel. If the application builder wants to dry-run the visualisation flow as an application user, they can press the eye beneath the pen in the left-most panel.

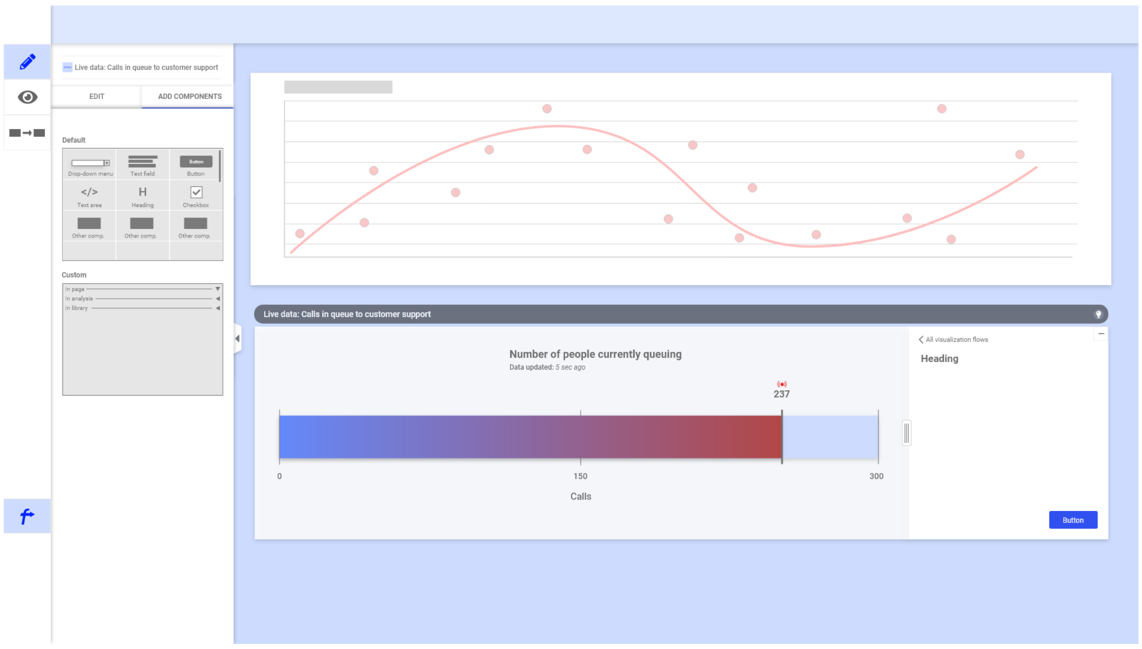


Figure 5.23: A “Heading”-component has been added to the visualisation flow.

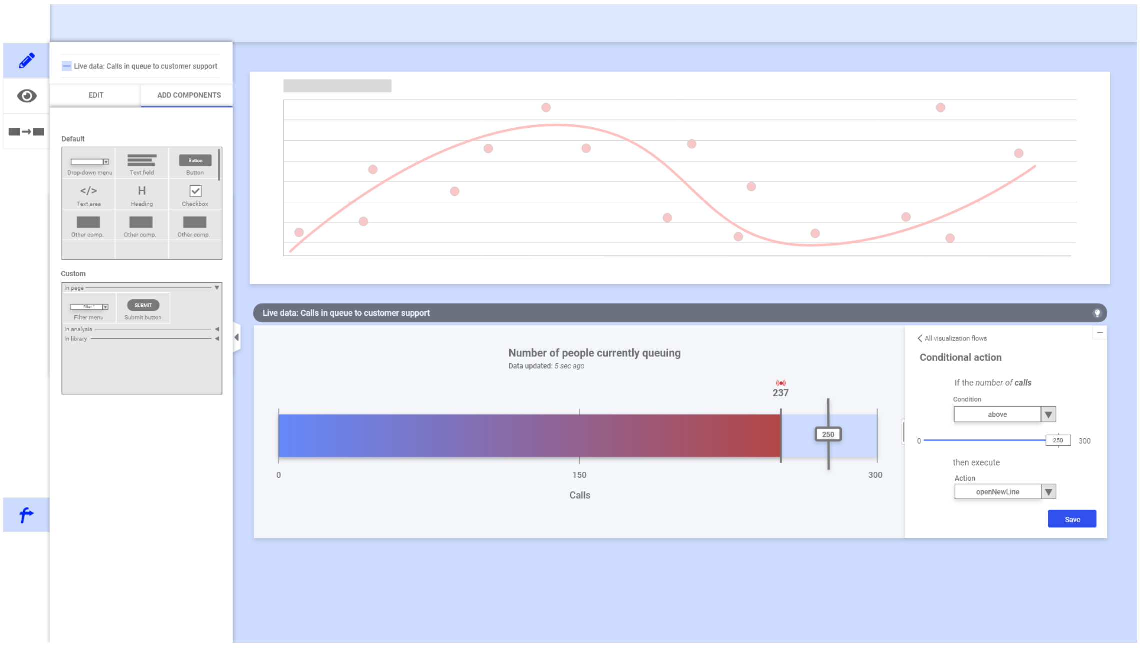


Figure 5.24: More components have been added to the visualisation flow.

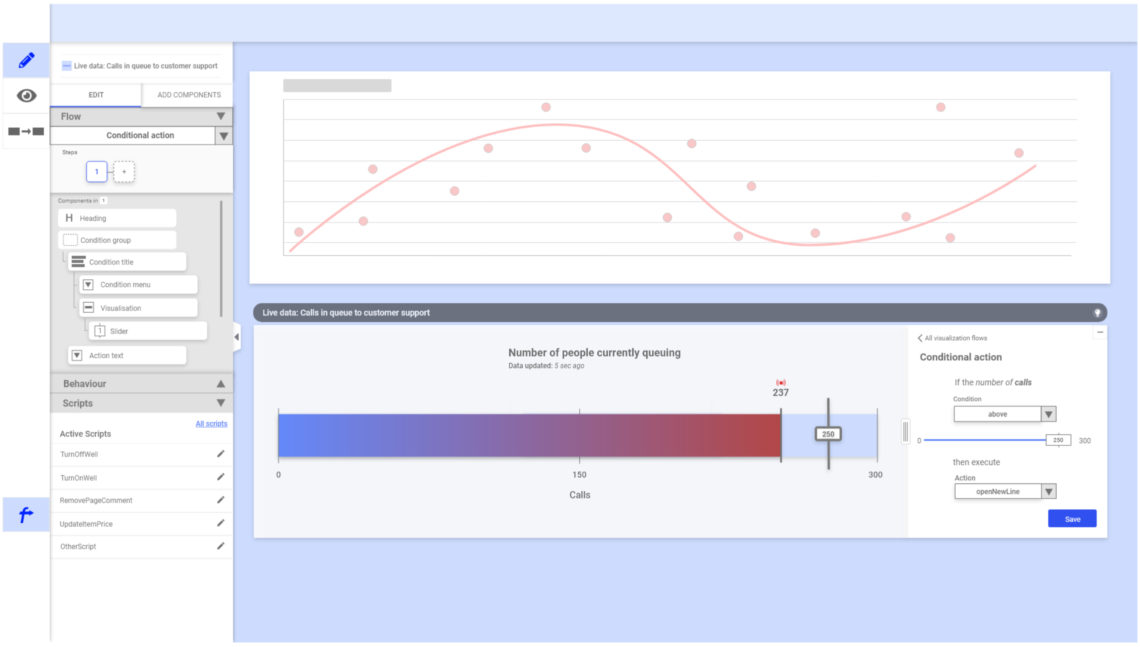
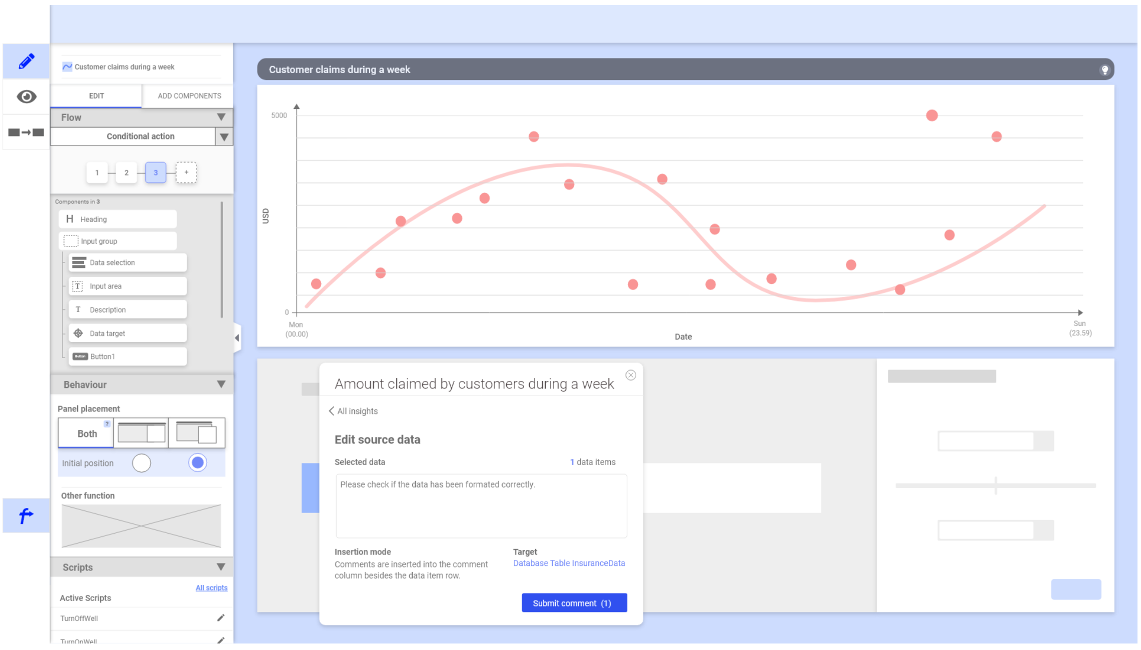


Figure 5.25: More components have been added to the visualisation flow and their structure are reflected in the left panel.

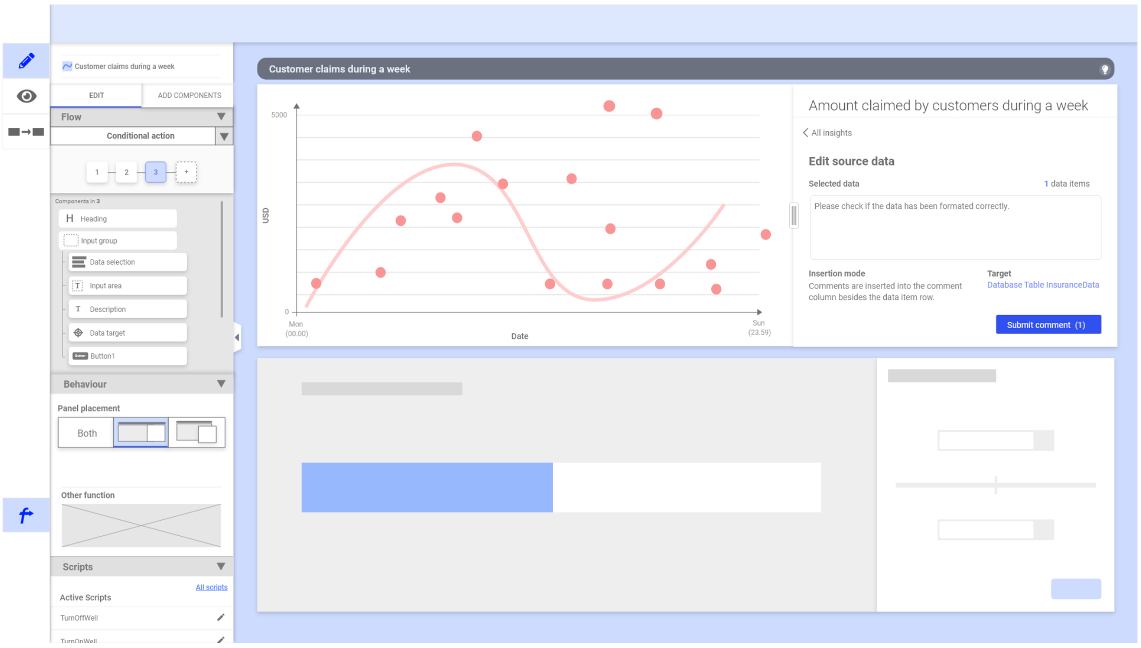
To manage screen real-estate has been an important consideration throughout the thesis. In the design critique session described in section 5.3.4, the data scientist stressed that sometimes visualisation flows are the purpose of an application. Therefore, it is required to be able to emphasise them more than placing them in a hidden menu. To accommodate this, the application builder can now decide whether the visualisation flow should be placed in the hidden menu, inside the visualisation or both. This can be seen in fig 5.26 and 5.27. In fig 5.26, the application builder has decided that the visualisation flow can be placed in both ways. When doing so, setting the initial position is required as well. If this is the case for the application user, they can drag and drop and "dock" the visualisation flow inside the visualisation.



Please check if the data has been formated correctly.

Comments are inserted into the comment column besides the data item row.

Figure 5.26: The visualisation flow is set to be, by the panel to the left, inside the menu at the top right of the visualisation (the light bulb icon).



Please check if the data has been formated correctly.

Comments are inserted into the comment column besides the data item row.

Figure 5.27: The visualisation flow is set, by the panel to the left, to be able to be set both in the menu at the top right of the visualisation, and inside the visualisation. The initial position is set to be inside the visualisation.

A thought during this iteration was: "What if a visualisation with its visualisation flows is a part of both an application *and* an analysis flow?". We thought that sometimes the things that are applicable to do with a visualisation changes depending on the context. Therefore, the visualisation flows that are associated with a visualisation can be configured differently when the visualisation is a part of an analysis flow and just by itself in an application.

In fig 5.28, the application builder is hovering over the visualisation in the analysis flow. Edit and preview buttons get visible, and the edit button is pressed. In fig 5.29, the application builder has been redirected to the edit view of the visualisation flow when it is a part of the analysis flow. The arrow at the bottom opens up the bottom panel of the analysis flow. To the right, there is a panel for the step where this visualisation flow exists within the analysis flow. This panel can be edited in the same way as the visualisation flow, by pressing the "Edit panel" in the top right corner.

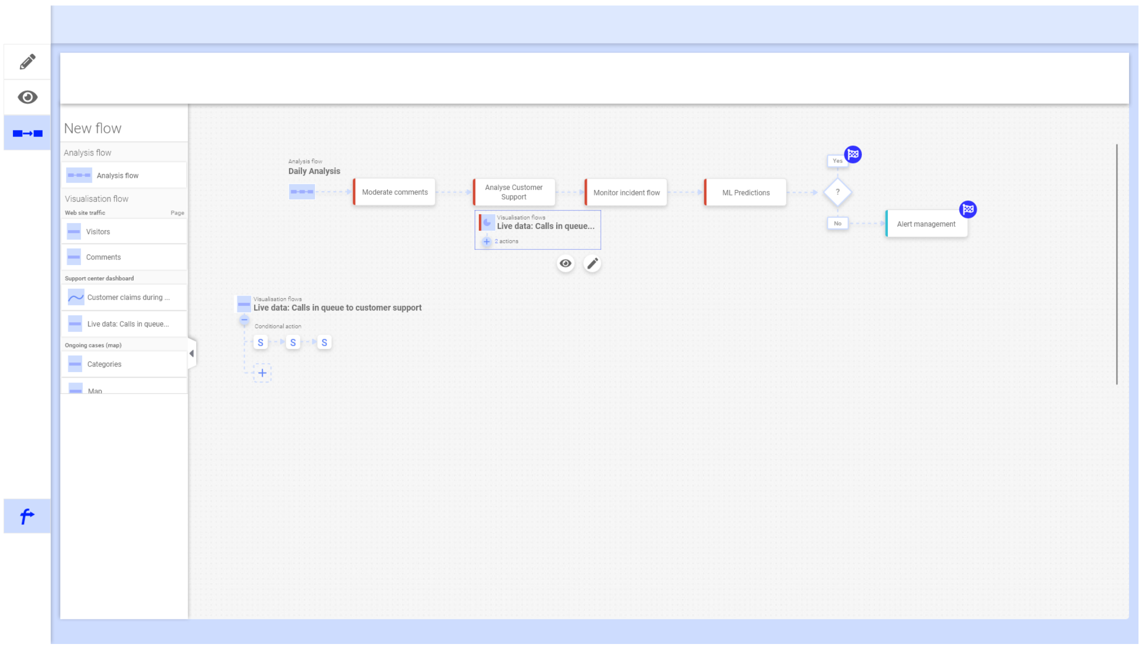


Figure 5.28: The application builder is hovering over the visualisation that is inside the analysis flow. This is the same overview as in fig 5.21, but with an analysis flow added.

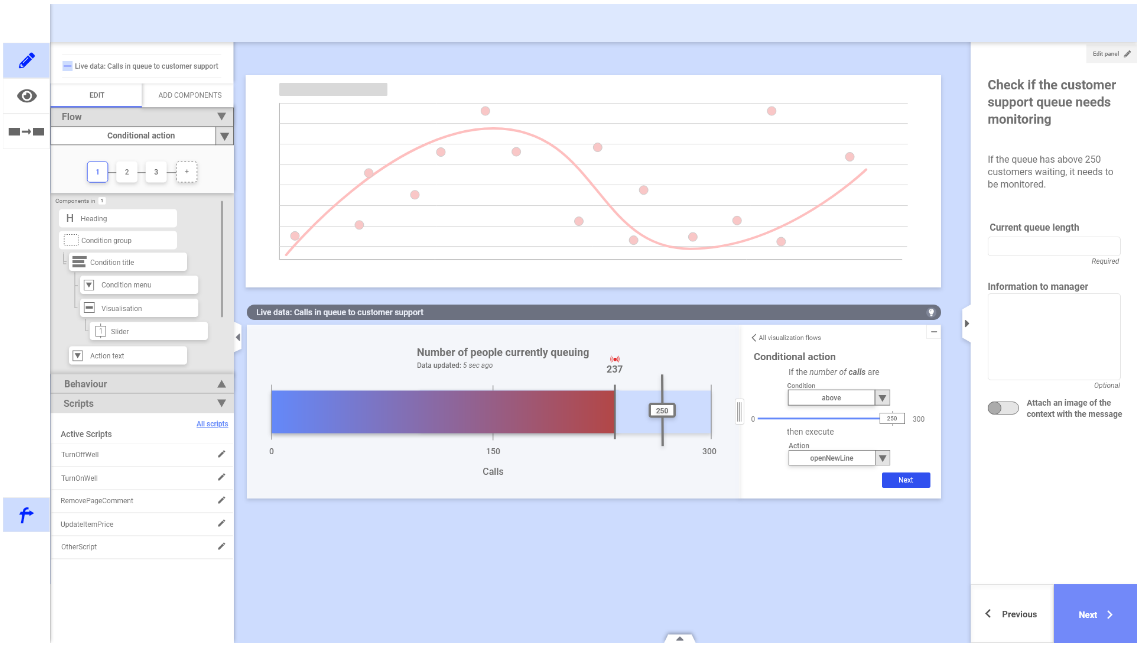
Several guidelines were considered during this prototyping phase. As this was the last iteration, the guidelines and the prototype were updated simultaneously. Hence, they inspired each other during their development process. This will be clear in the final set of guidelines, described thoroughly in section 6.3. However, the considered guidelines from version two were:

- "**G2** Strive to empower domain experts as much as possible without requiring them to write custom code". This guideline is the core of the whole workflow mode. An example of this can be seen in fig 5.23, where components can be dragged and dropped via a WYSIWYG interface.

- "**G3** Some desired behaviours will always require coding to be accomplished". This guideline is considered in fig 5.25, where the application builder is provided with an overview of all the scripts that are active in the current visualisation flow. These scripts can be attached to the components or the whole visualisation flow. Hence, making it possible to execute calls to external systems from within a visualisation flow.

- "**G4** 'Offer features on demand' / Different sections with functionality should be flexible (resize, hide/collapse)". All the workflows are possible to resize and hide. In fig 5.26, for example, the visualisation flow can be chosen to be either inside the visualisation or placed in the menu.

- "**G5** The system should strive to always keep the user (passively) informed". Where the application builder currently is in the workflow mode is communi-



Check if the customer support queue needs monitoring

If the queue has above 250 customers waiting, it needs to be monitored.

Figure 5.29: The edit mode of the visualisation flow when it is within an analysis flow.

cated by the left-most panel. In fig 5.28, for example, the application builder is in the overview mode. Information regarding how to use the workflow mode is communicated by the dialog in fig 5.20.

- "**G6** Consider the behaviour of objects and the information presented about them (reusability)". This guideline is considered mainly in fig 5.24. In the left panel, information regarding the current step in the visualisation flow is presented for the user. The components in that figure can be reused by saving them to the library.

- "**G9** Consider to support the possibility to replicate and record activities done in relation to an analyses". This is the workflow concept per se an example of.

- "**G12** Many users are not professional designers and could benefit from context aware help (with design)". By providing the application builder with premade default components, they get supported with the design. This can be seen in fig 5.23.

5.4.2 Usability Tests

As it was difficult to acquire experienced users of The Product to perform moderated user testing with, user tests on relatives and friends were conducted instead. The participants' backgrounds differed, but none of them had any prior experience with The Product at all, nor similar software. Two of the participants are students of the master's program Interaction Design and Technologies at the Chalmers University of Technology. The test was conducted on five persons.

The test followed a moderated usability test structure. As the participants had no experience of The Product, they were introduced to it before the tests started. Also, because of the lack of experience, the questions were mostly related to how the participants felt at certain stages and how they thought some interactions would be performed. The questions can be found in Appendix D.

5.4.3 Takeaways from the Usability Tests

The main takeaway from the usability tests was that the concept of workflows is difficult to grasp for a user with no experience of The Product. Therefore, during the tests, it was sometimes hard to differ between poor design choices and just uncertainty of the concept and its intentions over all.

Something that could be verified during the tests was design choices regarding guideline "**G5** The system should strive to always keep the user (passively) informed". As the users were new to The Product, they required guidance to get an idea of the intention of some views. For example, the dialog containing information on how to use the flow canvas and its corresponding side-panel seen in fig 5.20, were proven to be useful in all of the tests. This was also the case with the modeless feedback regarding saving, seen in fig 5.18. Design choices regarding the same guideline were also verified from a different perspective in the same wireframe. The users felt secure about where they were, which objects were related to which, and where they could go. An example of such a case can be seen in the progress indicator in fig 5.18. Moreover, design choices regarding "**G4** 'Offer features on demand' / Different sections with functionality should be flexible (resize, hide/collapse)" were also verified to be of value for the users. Examples of those design choices can be seen in fig 5.26, where the application builder can decide where a visualisation flow should be placed.

Some design choices that were to be considered for the future were some colours of the progress indicator in fig 5.18. Some participants thought they were too alike, which confused them. Additionally, the preview and edit buttons, see fig 5.22, took some time to find and should be considered placing in a different order.

5.4.4 Design Critique

The final prototype was presented for both a product manager and an interaction design working at The Company for a design critique session. The main purpose was to get feedback on the concept of workflows as it is constructed in this prototype, and whether it has a place in The Product for the future. As it is a completely new "mode" in The Product, feedback on how well it is understood by someone with experience in The Product was also welcomed.

Overall, both participants thought that the low-code style of the concept felt modern. They both expressed that it has a place in The Product, but there is a long way to get there. One idea that was presented was that, if the workflow mode would be implemented in The Product, it would initially only be supported by the API. That would test the concept on the more experienced users of The Product first. Then, it could maybe be realised in a GUI fashion.

Another thing that was discussed was the reusability aspects of workflows. As the workflows are currently created, the context of the workflow is decided first. This makes the workflows very specific and bound to a visualisation. Even though they would be sharable in theory, it may not be any value in sharing them.

In summary, both participants liked the concept. But, as it is just a concept, several areas still causes questions if implemented in a real product.

5.4.5 Guidelines v. 3

In this iteration, the guidelines address the ones who are supporting application builders. In the context of this thesis, this is The Company. They are all reformulated as an exhortation towards The Company, and they all strive to support the application builder in creating applications with their desired user experience. A final, more descriptive version of the guidelines, can be found in section 6.3.

G1 **Help application builders to follow fundamental design principles by offering good defaults, templates and automatic recommendations**

This guideline was kept from guidelines v. 2 but tweaked and reformulated. See "**G1** Consider existing guidelines for User Experience". It was reformulated to be more precise and to give more context on when and how.

G2 **Let the workspace contain unobtrusive contextual help with clear entry points to external company resources**

This guideline was kept from guidelines v.2 but reformulated, see "**G5** The system should strive to always keep the user (passively) informed". It was also separated into two guidelines, this one and "**G9** Applications should have a default navigation structure". It was reformulated as it was discussed in the interviews in section 5.2.2 that The Company has many external resources and often gets requests about features that are already existing.

G3 **Make application development less dependent on custom code by providing no-code development tools**

This guideline was kept from guidelines v. 2 but reformulated, see "**G2** Strive to empower domain experts as much as possible without requiring them to write custom code". That guideline was separated into two guidelines, this one and the guideline below. This guideline was reformulated to put focus on the application building process, not overall usage of The Product.

G4 **Together, the tools for application building should adhere to The**

Gentle Slope of Complexity

This guideline was kept from guidelines v.2 but reformulated, see “**G2** Strive to empower domain experts as much as possible without requiring them to write custom code”. That guideline was separated into two guidelines, this and the one above. This guideline was reformulated and taken back from the initial set of guidelines, as it focuses on how the tools of the application building process should be, not what they are.

G5 **Support application builders who customise through code**

This guideline was kept from guidelines v.2 but reformulated, see “**G3** Some desired behaviours will always require coding to be accomplished”. That guideline was separated into two guidelines, this one and “**G12** Support applications’ ability to communicate with external systems”. This guideline was reformulated to put more focus on the customisation capabilities in general of custom code.

G6 **Provide application builders with an adaptable workspace**

This guideline was kept from guidelines v.2 but reformulated, see “**G4** ‘Offer features on demand’ / Different sections with functionality should be flexible (resize, hide/collapse)”. It was rewritten for clarity purposes.

G7 **Allow when users create application components, the process should follow What You See Is What You Get**

This guideline has its origins in “**G2** Strive to empower domain experts as much as possible without requiring them to write custom code”, which got separated into two guidelines. This one and “**G3** Make application development less dependent on custom code by providing no-code development tools”. This guideline was added to emphasise that the low code tools should follow the WYSIWYG principle.

G8 **Have designated locations in the GUI where application builders can view and manage related groups of elements in an application**

This guideline was kept from guidelines v.2 but reformulated, see “**G6** Consider the behaviour of objects and the information presented about them (reusability)”. It was reformulated to clarify the purpose of the guideline.

G9 **Applications should have a default navigation structure**

This guideline was partly kept from guidelines v.2, see “**G5** The system should strive to always keep the user (passively) informed”. That guideline was separated into two, this one and “**G2** Let the workspace contain unobtrusive contextual help with clear entry points to external company resources”. This was made as both the navigational parts and the supportive parts of the original guideline felt equally important, and not directly related.

G10 **Offer visual and system support for collaborative application building**

This guideline was kept from guidelines v.2 but reformulated, see "**G10** Program users seldom work alone, consider the needs users have as a group". It was reformulated to clarify when collaboration should be supported.

G11 **Allow application builders to create, store and share custom workflows in multiple application layers**

This guideline was taken back from guidelines v.1 but reformulated, see "**G5** The user should able to create custom workflows without the need to code". It was reformulated to emphasise that the workflows should be able to act in different contexts. "Create, store and share" were added to focus on the whole workflow activity, not only the creation.

G12 **Support applications' ability to communicate with external systems**

This guideline has its origins in "**G3** Some desired behaviours will always require coding to be accomplished", which got separated into two guidelines. This one and "**G5** Support application builders who customise through code". This guideline was added to emphasise that the application should be able to communicate with external systems. This is currently done via custom scripts in The Product, which may not be the general case.

6 Results

An interactive prototype and a set of twelve guidelines are the result of this thesis and its iterative design process. Both products received minor adjustments after the final iteration to clarify their purpose further. Their intention is to be a source of valuable information in future development of analytical software.

6.1 Workflows

When users develop custom analytical applications, the ability to create workflows helps them implement an applications desired experience. Conceptually, workflows capture the frequent need to led application users through a sequential paths. Flows, short for workflows, become become an additional layer of interaction, on top of the default graphical user interface (GUI). This concept uses two kinds of workflows: *Analysis flows* and *Visualisation Flows*. How workflows exist in an application and relates to the application's ontology is illustrated in fig 6.1. In essence, Analysis workflows can span multiple application pages. A visualisation workflow can be connected to one or more visualisations and they can also be part of analysis workflows. Each analysis flow runs as a separate instance in an applications and the application builder decides which flows the included visualisations contain. The behaviour of both workflows is affected by changes to visualisations, their linked data, and vice versa.

As a general concept, workflows have many possible use cases. The prototype used an insurance company as a potential user. From now on, this section will refer to a figurative clinical study to explain how the concept works. Healthcare staff could use a workflow to record patient data and monitor their health. An application builder would add suitable conditions to the workflow. Additional sequence can be triggered if a staff member inputs an alarming value, possibly below the accepted minimum. An alert telling staff to not administer any additional medication could appear. The flow would then guide the nurse through a series of appropriate instructions. An entry in the patient journal would likely be required, or contacting a doctor. All this interaction can happen inside an application since it connected external systems. Generally, analysis workflows can span several pages, be linear or not, and have optional and required steps.

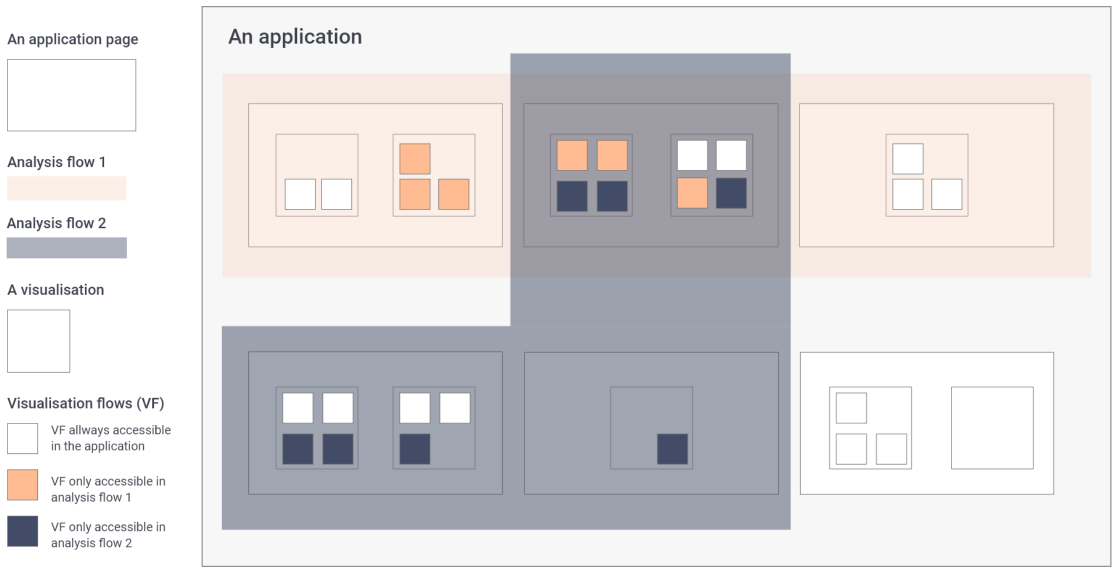
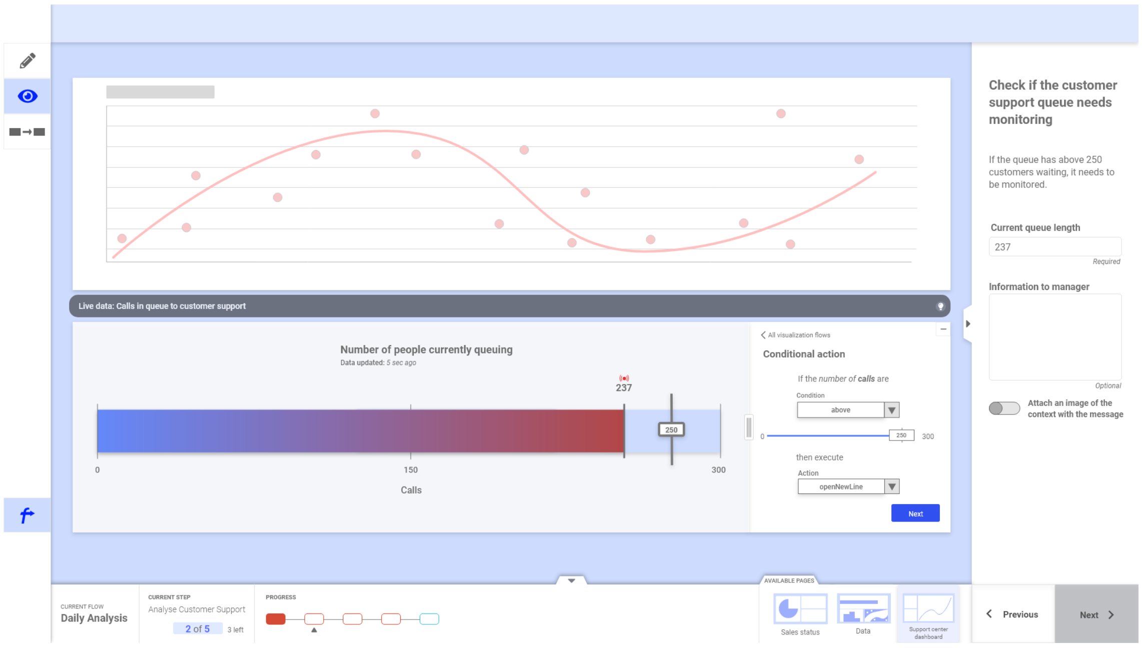

An application page
Analysis flow 1
Analysis flow 2
A visualisation
Visualisation flows (VF)
VF allways accessible in the application
VF only accessible in analysis flow 1
VF only accessible in analysis flow 2
An application

Figure 6.1: The ontology of an application and the relations between pages, visualisations, visualisation flows and analysis flows.

Later during the study, a nurse examined a visualisation showing patients' heart rate during the study. The nurse observes a decline by 170 beats, an unlikely reading. The workflow asks the nurse to mark the possible error in the chart. As required, the value gets a written comment, the nurse believes the value is an incorrect entry. The workflow adds the comment as metadata in the source database. This scenario is an example of how a *Visualisation Flow* can be used in analysis flow, where an action can be performed directly on the source data used by a visualisation.

Each analysis workflow can modify an application's default GUI and its content: active visualisation flows or any user-made elements, such as paragraphs or buttons. The inclusion of elements in the workflow sections also leaves more space on the pages for visualisations to use. An application can be layered with an arbitrary number of workflows, and application builders can place components and information inside the workflows as they wish.

As previously mentioned, analysis flows can span over several pages in an application and be both linear and non-linear. Figure 6.2 demonstrates an example of how a sequence in an analysis flow could look and behave for a user. In this example, the application is for an insurance company's call centre. The person using the workflow is supposed to check the customer support queue and report findings to the manager via the panel to the right. Reporting the queue length is required, and the person cannot proceed until it's done. In figure 6.3, the person has reported the current length and can proceed. The bottom and the right panel are collapsible, and follow the user throughout the flow. The bottom one shows available pages for the current step (non-linearity) and where the user is in the analysis flow. Further explanation of all the interactions with this screen can be found in 6.2.4.



Check if the customer support queue needs monitoring

If the queue has above 250 customers waiting, it needs to be monitored.

Current queue length: 237 (Required)

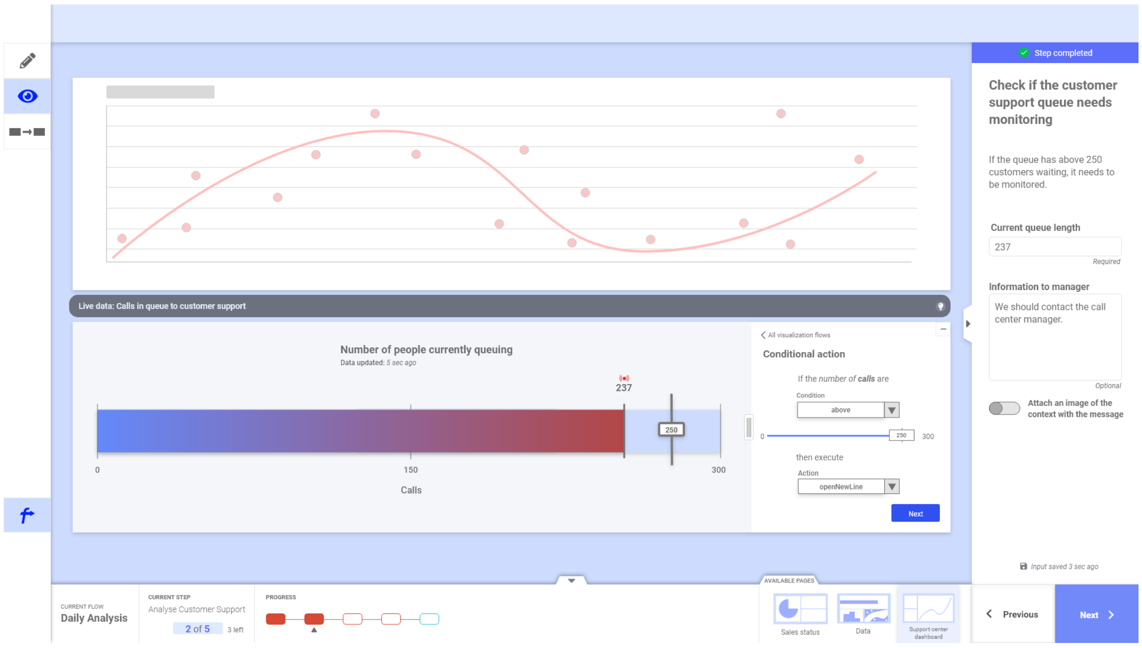
Information to manager (Optional)

Attach an image of the context with the message

If the *number of* ***calls*** are
Condition: above
0 — 250 — 300
then execute
Action: openNewLine
Next

Analyse Customer Support — 2 of 5, 3 left

Figure 6.2: An analysis flow for a call centre.



Step completed

Check if the customer support queue needs monitoring

If the queue has above 250 customers waiting, it needs to be monitored.

Current queue length: 237 (Required)

Information to manager: We should contact the call center manager. (Optional)

Attach an image of the context with the message

Input saved 3 sec ago

If the *number of* ***calls*** are
Condition: above
0 — 250 — 300
then execute
Action: openNewLine
Next

Analyse Customer Support — 2 of 5, 3 left

Figure 6.3: The steps required to continue the analysis flow are completed and the application user is allowed to proceed.

Figure 6.3 also shows a visualisation flow, attached to the visualisation “Live data: Calls in queue to customer support”. A user selects an condition, and tells the flow what script to execute when a condition is met. In this example, the condition is evaluated against the length of the queue. Above the title of the section, the user can return to the start page, it has all the flows that are available for use with this visualisation.

An alternative access point for visualisation flows is shown in figure 6.4. Here the flows are placed in an existing menu called “The insight menu,” opened via the light bulb at the top right of the visualisation. All data in the visualisation is selected by default. Users can select specific values, in fig 6.5, a user wants to write back metadata to the visualisation’s source data and selects the appropriate visualisation flow “Edit source data”. The user selects a single data point, possibly a deviation, and attaches a comment.

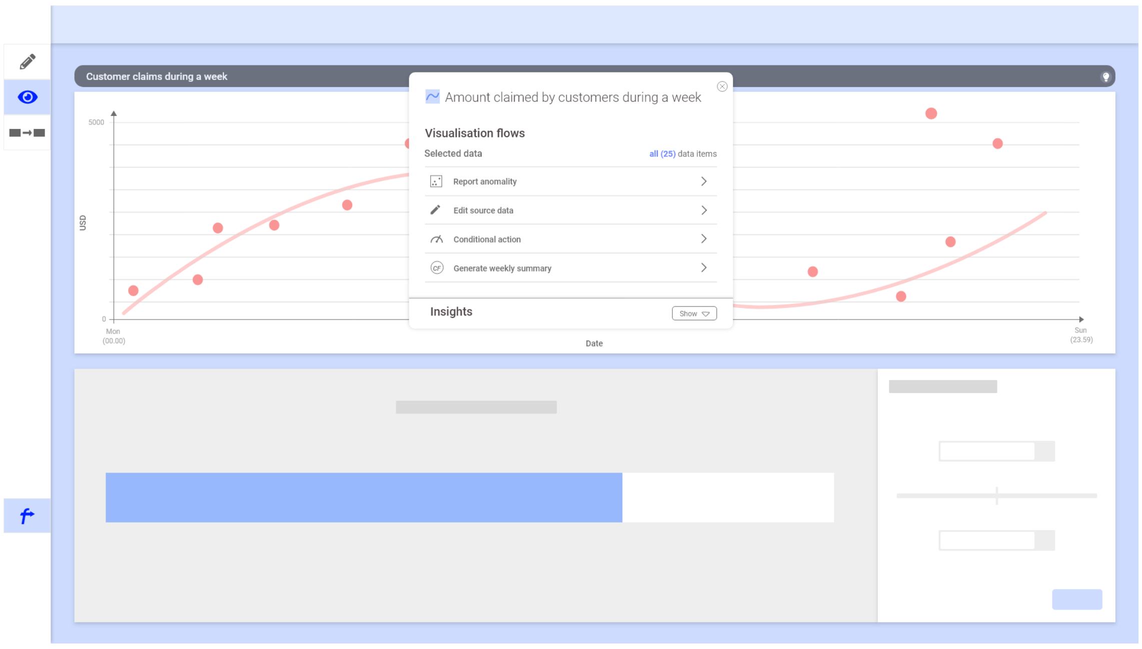
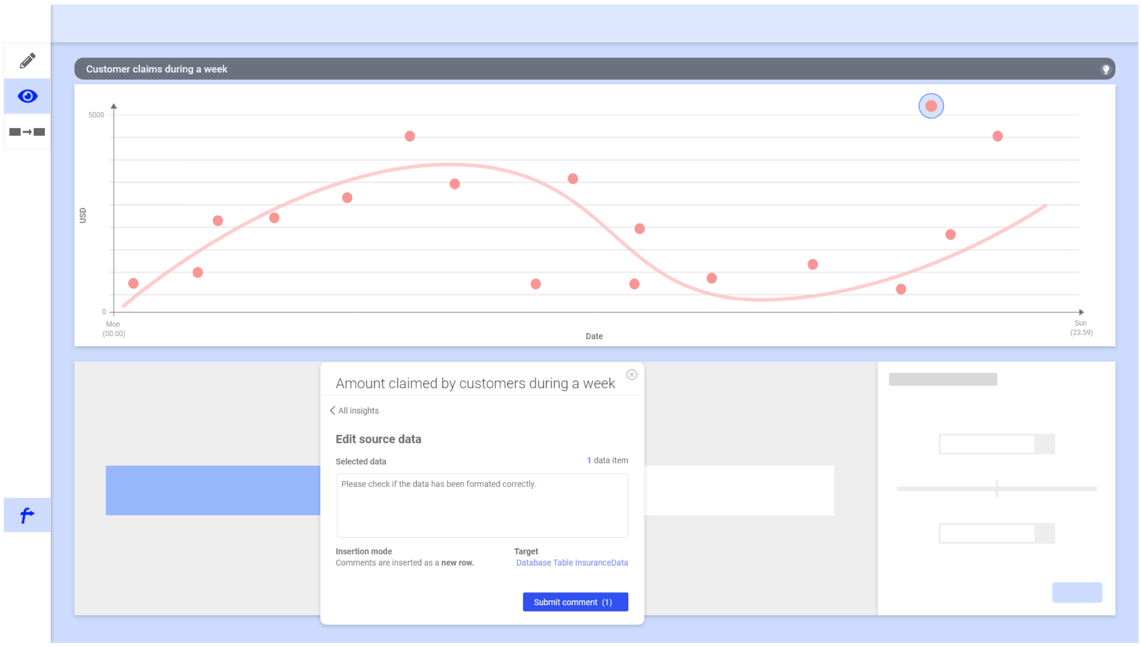


Figure 6.4: The menu containing all the visualisation flows is open.



Amount claimed by customers during a week

All insights

Edit source data

Selected data — 1 data item

Please check if the data has been formated correctly.

Insertion mode: Comments are inserted as a **new row.**

Target: Database Table InsuranceData

Submit comment (1)

Figure 6.5: The chosen visualisation flow is "Edit source data" and the user has selected one point to comment on.

6.2 Prototype

The final concepts demonstrate how application builders would construct workflows without having to write scripts. Workflows can either be part of visualisation or span across multiple application pages; these two levels are referred to as *Visualisation Flows* and *Analysis Flows*. A more detailed description of both can be found in 6.1.

As a whole, the prototype intends to show that software can support both end-user development and traditional software development - while remaining user-friendly for both groups. As reflected in the guidelines, the prototype also acts as an example of how the guidelines can be translated into GUI functionality. The secondary goal of the prototype is to be a source of inspiration in future developments in analytical software. Patterns and approaches, like block-based building, could likely be helpful in additional contexts.

Adobe XD was used to create the design, from low-fidelity to an interactive high-fidelity version. The design developed during three iterations and details about each iterations are described in chapter 5.

6.2.1 Entering Workflow Mode

Figure 6.6 shows one page of an application built inside The Product. At the bottom left, there is a button with a stylised "f" and users access the workflow canvas by pressing it. In figure 6.7, the application builder has entered the workflow mode. A panel with introductory information about the workflows on top of the blank canvas, if it is the first time a user accesses the environment. The left panel shows where the application builder is currently. At this moment, the overview mode is selected.

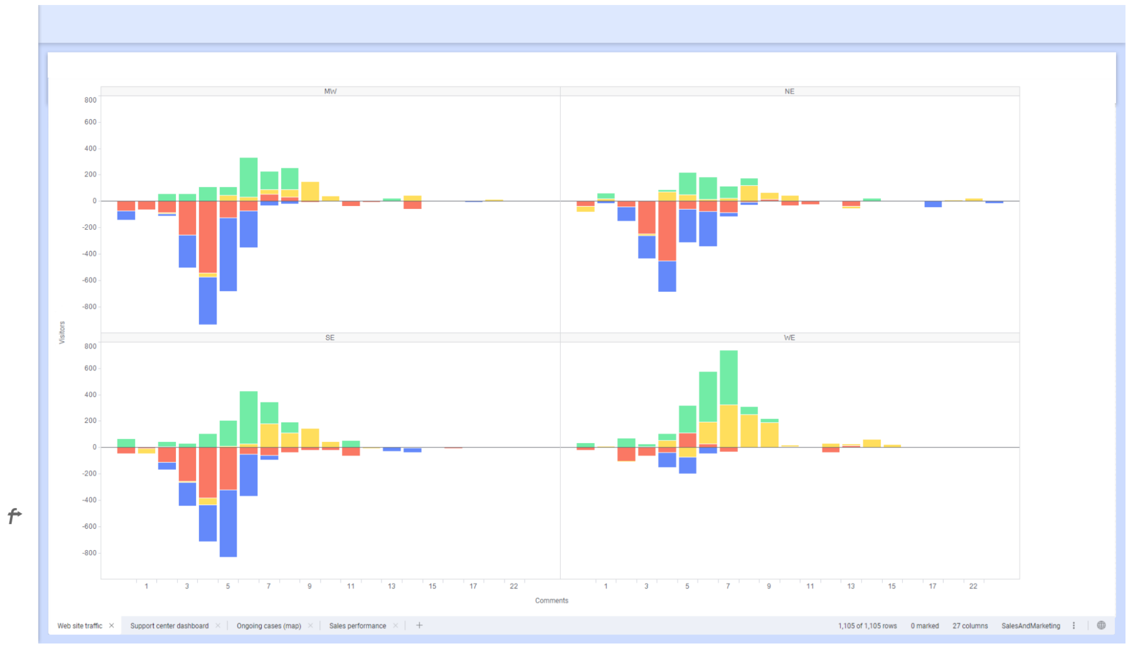


Figure 6.6: A page in an application. The button at the bottom-left is the button for the workflow mode.

The collapsible side panel with the heading “New flow” contains various context blocks from which a workflow can start. The top block is used to begin an analysis flow (application-level), others to work with a visualisation. Application builders can drag and drop these blocks onto the canvas, they can be placed anywhere on the scrollable canvas. It can also hold several workflows. In figure 6.8a, the visualisation named “Live data: Calls in queue to customer support” is selected, and in figure 6.8b, the entire analysis After the context block has been dropped onto the canvas, a plus sign appears next to it. If the application builder presses the plus sign, they will be redirected to the flow creation view.

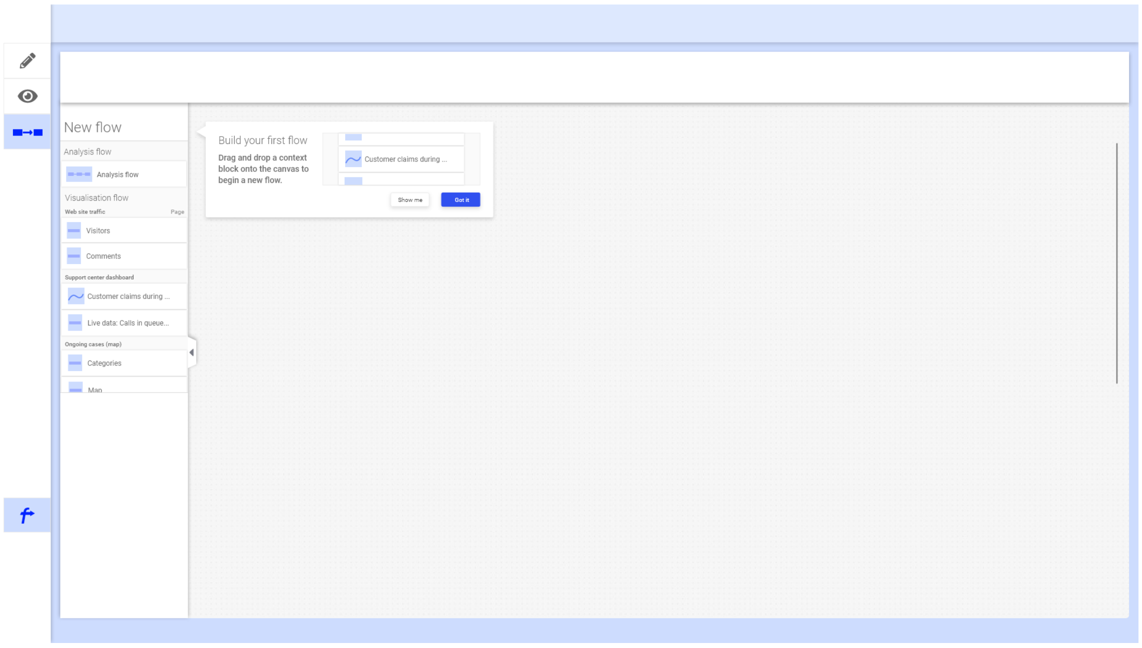
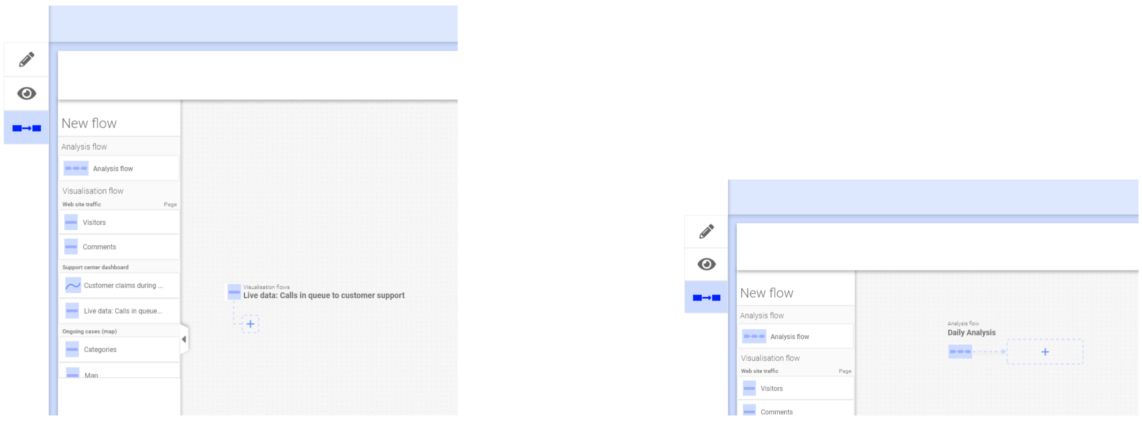


Figure 6.7: The workflow creation overview, entered for the first time.



(a) A visualisation is chosen as the context an dropped onto the canvas.

(b) The whole application is chosen as the context an dropped onto the canvas.

Figure 6.8: Different contexts are dragged and dropped onto the canvas.

6.2.2 Constructing a Visualisation Flow

When the application builder presses the plus sign on the chosen context, they get redirected to the flow construction view, and the button with a pen is in an active state, see fig 6.9. The controls for the selected workflow in the chosen visualisation are shown in the left panel. The top heading shows the currently selected visualisation, users simply click on another one to switch. The current workflow can be changed by pressing the drop-down menu with the text “Conditional action”, also the name of the current workflow.

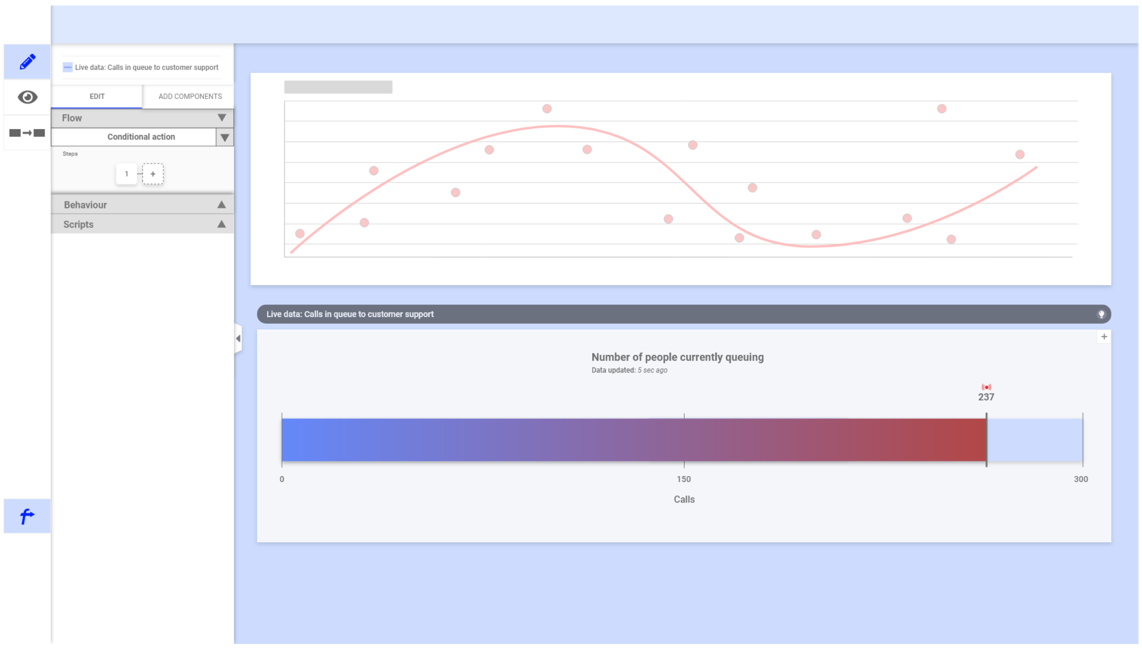


Figure 6.9: The first view when creating a visualisation flow.

The workflow editing panel has two tabs, one for behavioural and content adjustments in the current step, and one for adding components to its GUI. In figure 6.10, step one is selected. The panel is updated automatically and displays information about this step: included components and their relation.

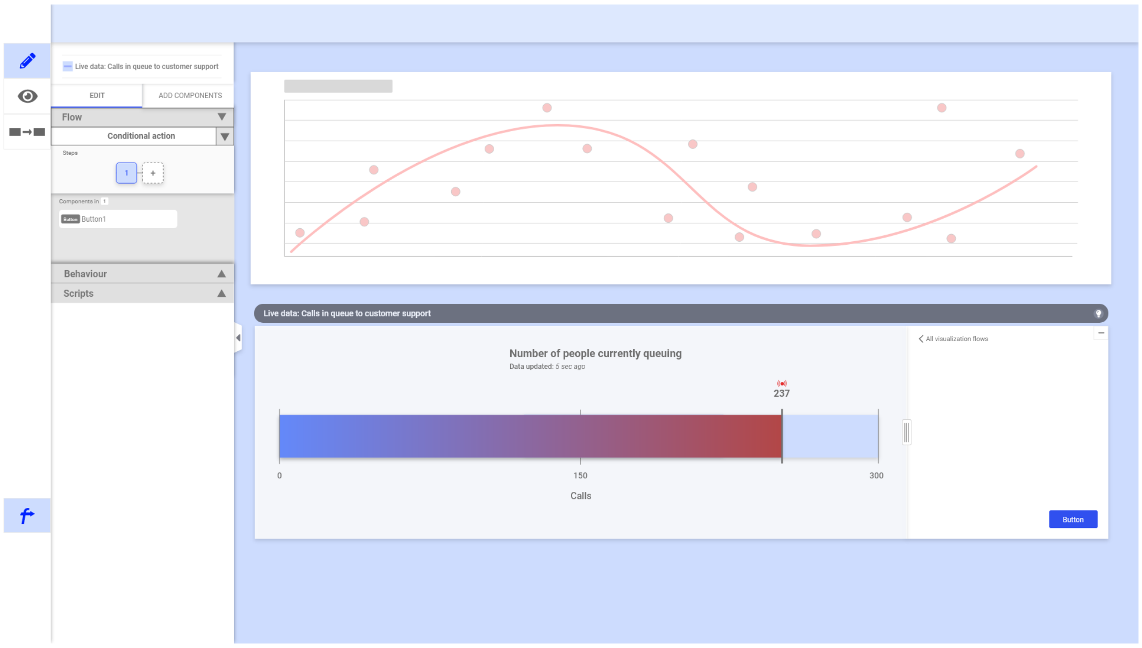


Figure 6.10: Step one of the workflow is pressed.

When pressing the “Add components” tab in the side panel, new components can be added to the canvas. In figure 6.10, the content of the side panel has changed to show all available components. There are default components and custom components. Application builders can create custom components and use them in other applications via The Product's cloud-based file system, commonly called the library. Besides custom components, other components used in this page and in the application are shown as well. Adding components follows the What You See Is What You Get (WYSIWYG) interface style: components are dragged from the side panel and then dropped onto the canvas. This approach is demonstrated in figure 6.11, where a "Heading"-component has been dropped onto the canvas.

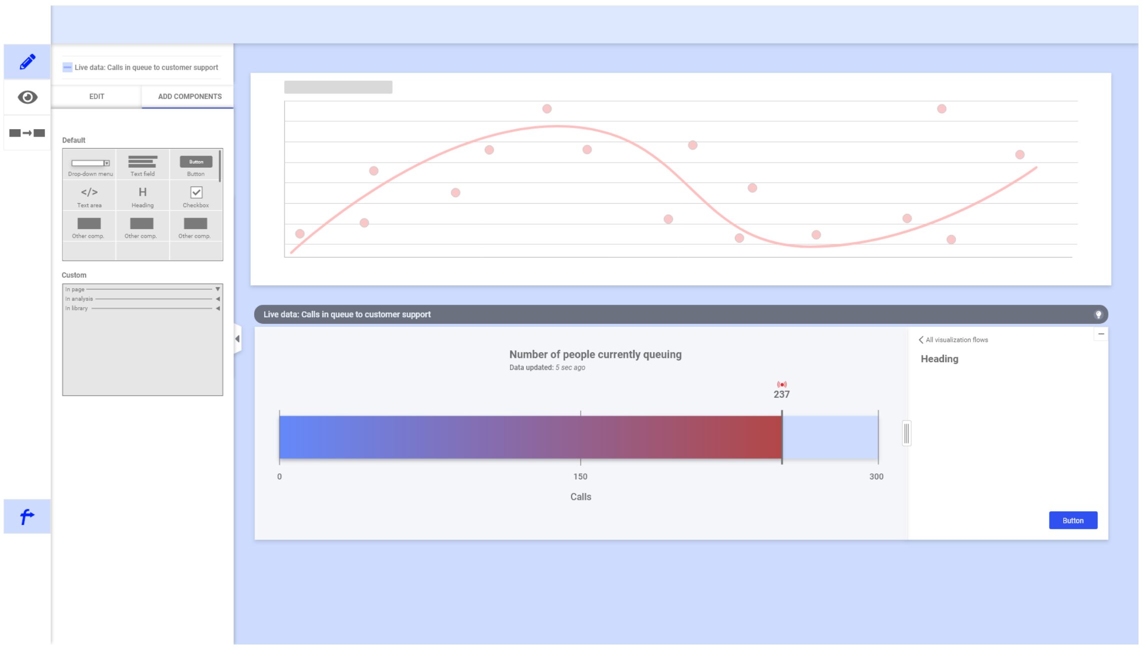
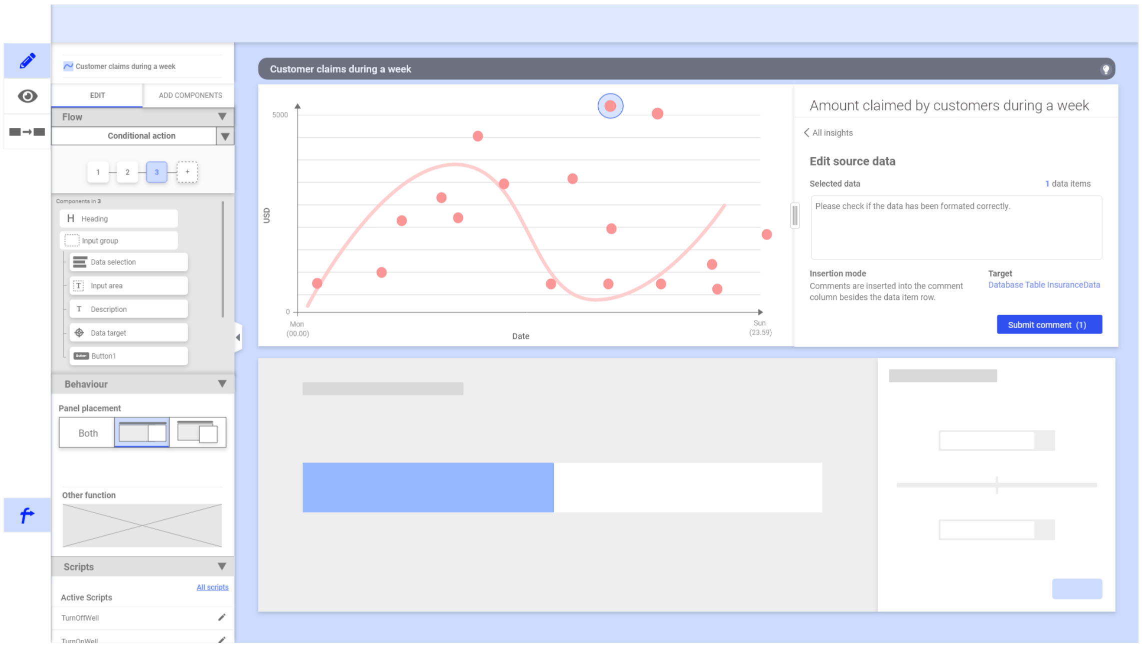


Figure 6.11: A "Heading"-component has been dropped onto the canvas.

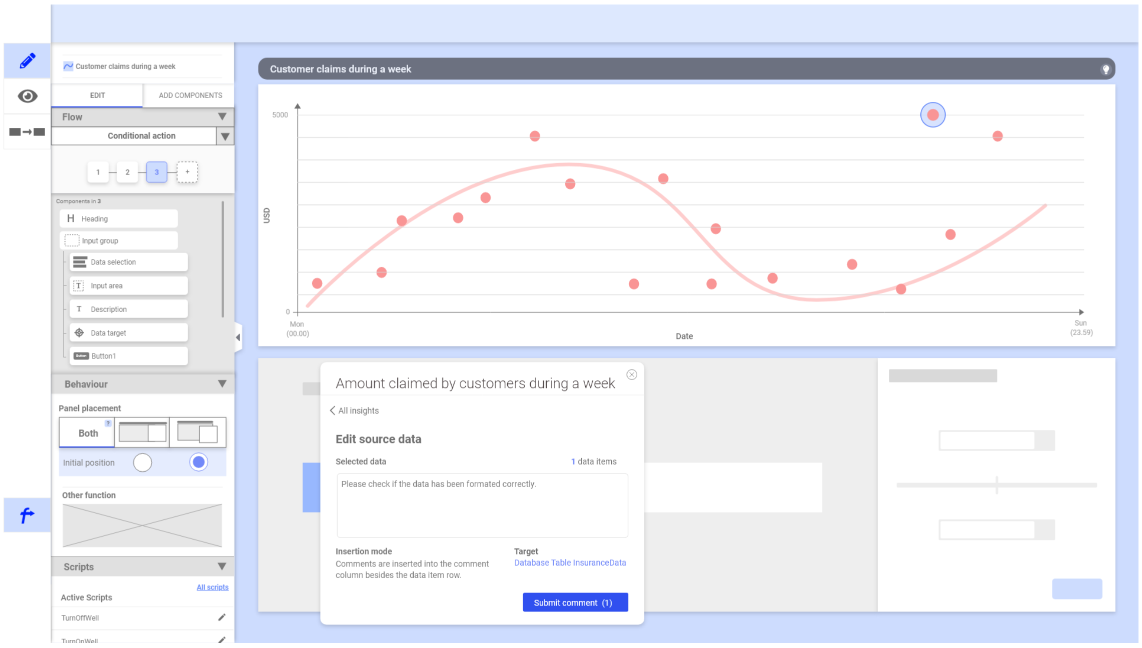
The application builder can decide how a visualisation flow positions itself in an application page. In figure 6.12, the visualisation "Customer claims during a week" is placed next to the visualisation, this is referred to as inline placement. The application builder can edit placement in the side panels "Behaviour" section. In figure 6.13, the application builder wants the visualisation to be available both inline, as in fig 6.12, and floating (right-most icon). When both modes are selected, application users decide how it appears. This approach lets application users switch freely between the two placement modes. When float mode is active, applications users access flows via the light bulb menu at the top-right of the visualisation. Consequently, users can hide the flow panel.



Please check if the data has been formated correctly.

Comments are inserted into the comment column besides the data item row.

Figure 6.12: The visualisation flow placed inside the visualisation.



Please check if the data has been formated correctly.

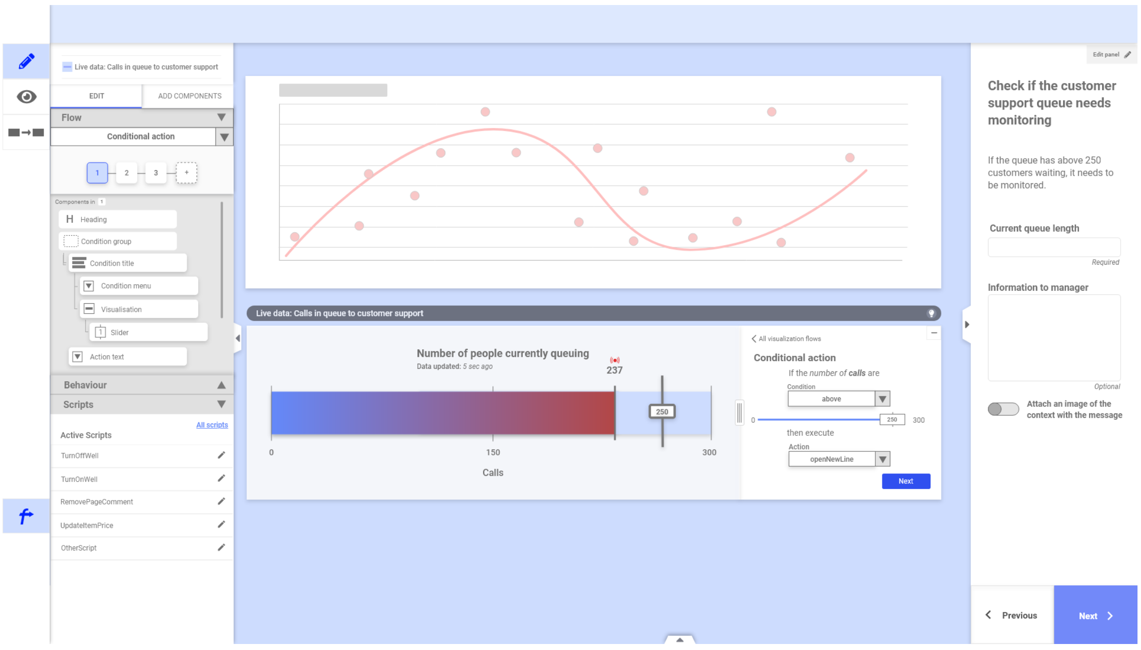
Comments are inserted into the comment column besides the data item row.

Figure 6.13: The visualisation flow can be positioned both besides the visualisation and in the light bulb menu, accessed via light bulb icon in the top-right corner of the visualisation.

Besides changing the placement of the visualisation flow and viewing the components used in a step, builder can also access custom scripts. They are placed under the "Scripts" title and shown at the bottom left of fig 6.13. The scripts can either be attached to components in the visualisation flow or be used to set the behaviour of the flow itself.

6.2.3 Editing an Analysis Flow

Visualisation flows belong to visualisations, and visualisations are placed in pages. An analysis consists of at least one page. Therefore, visualisation flows can exist within an analysis flow, section 6.1 provides a more comprehensive explanation. Figure 6.14 shows the editing of a visualisation flow that is part of an analysis flow. To get to this view, the application builder has first created an analysis flow in the canvas. One step of this analysis flow, which can be viewed from an overview perspective in fig 6.15, contains a page where the visualisation flow exists. However, details regarding the creation of each step of an analysis flow are excluded from this prototype. To edit the visualisation flow in step “Analyse customer support” of the analysis flow, the application builder presses the pen beneath the visualisation flow, shown in fig 6.15.



If the queue has above 250 customers waiting, it needs to be monitored.

Attach an image of the context with the message

Figure 6.14: The visualisation flow in edit mode when it is a part of an analysis flow.

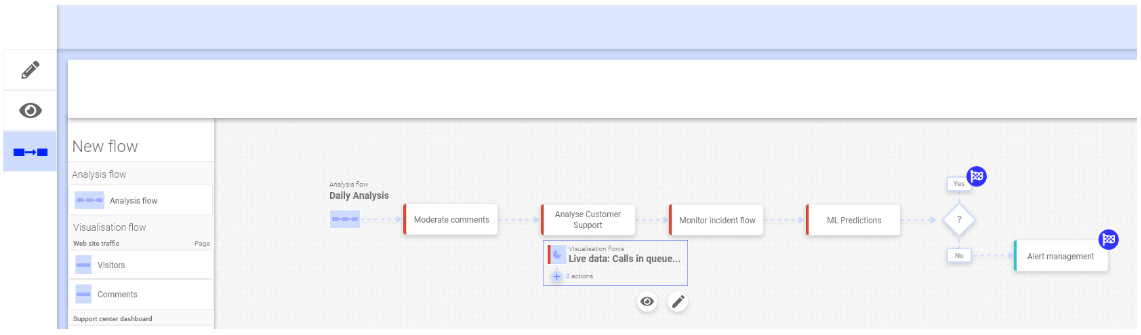
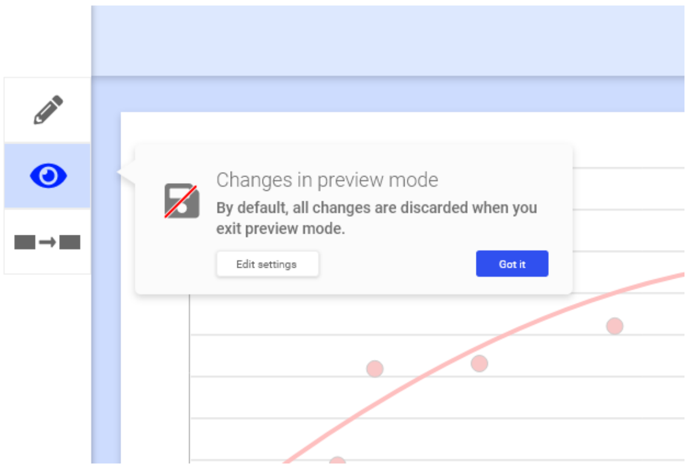


Figure 6.15: The overview of a complete analysis flow. The application builder hovers over the visualisation flow, which makes the eye and pen icon appear.

Multiple contexts can be changed during the editing of an analysis flow. The builder can edit visualisation flows that are part of the analysis flow, and the collapsible side panels with information about the current step. Switching to edit is done by pressing the "Edit panel" text at the top right. The panel would then be edited similarly to the visualisation flow, described in section 6.2.2. The right-side panel is always available for an application user when performing an analysis flow, and is collapsible in the same way as in fig 6.14. If the application builder would like to test their workflows, they can press the "eye"-icon in the left-most side panel. They can also skip the edit mode and test the workflow directly by pressing the "eye"-icon in fig 6.15.

6.2.4 Testing and Using a Workflow

There are two ways to activate a workflow. It can be done either by pressing the "eye"-icon in the left-most panel in fig 6.16 or by starting it directly in the application. When running the workflow from the "eye"-icon, it triggers a test version. No actions are saved, this is indicated by the window in fig 6.16. Application builder can test how their workflow will look and behave, from an application user perspective, without any consequences.



Changes in preview mode

By default, all changes are discarded when you exit preview mode.

Figure 6.16: The application builder has entered the preview mode of an analysis flow. This window reminds the application builder that no changes will be saved.

The application builder has, in preview mode, opened both panels belonging to the analysis flow in fig 6.17. The bottom panel shows the application user: the name of the analysis flow, the name of the current step, how many steps there are and how many left. Progress is also communicated visually by the progress indicator. In fig 6.17, the current step is not completed, indicated by "Next" buttons disabled state and the empty rectangle in the progress indicator. Available pages enables the analysis flow to not be completely linear. Section 6.1 describes their role in more detail.

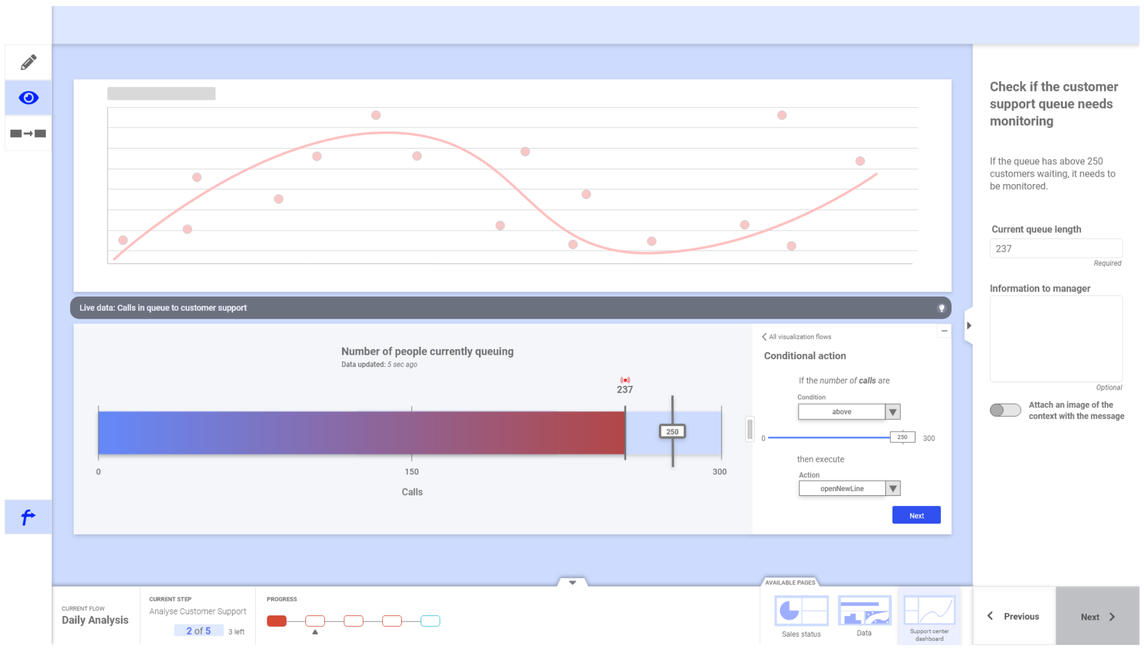


Figure 6.17: The analysis flow in preview mode with both panels open.

During some steps, application users have to complete the required tasks before proceeding. Figure 6.18 shows how the application uses modeless feedback when users have completed a step. The "Next" button switches from grey to green after completion.

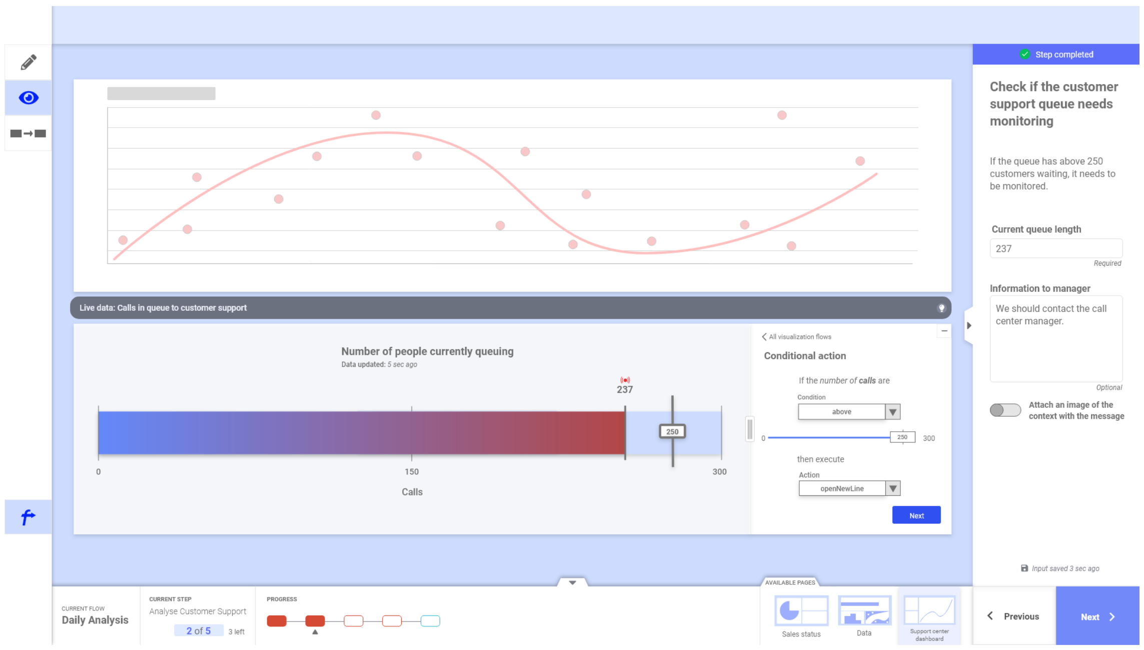


Figure 6.18: The steps required to continue the analysis flow are completed and the application user is allowed to proceed.

6.3 Guidelines

Those who develop analytical software with the possibility to create custom applications are the intended audience for these guidelines. The users who create applications are referred to as application builders, or builders for short. Usually, this person works as a data scientist or a similar position. When analytical software applies these guidelines, it becomes more likely that builders will have the user experience they desire. An additional intention is to increase the likelihood that builders can achieve their goals regarding user experience in their custom applications.

These guidelines have been influenced by captured insights from users and experts, multiple End-user development perspectives, theory from Information Architecture and more.

Guidelines

1	Help application builders to follow fundamental design principles
2	Let the workspace contain unobtrusive contextual help
3	Provide no-code development tools
4	Consider The Gentle Slope of Complexity
5	Support application builders who customise through code
6	Provide application builders with an adaptable workspace
7	Display a GUI component's final state during editing
8	Provide places that show all related application elements
9	Applications should have a default navigation structure
10	Offer visual and system support for collaborative application building
11	Allow application builders to create, store and share custom workflows in multiple application layers
12	Support applications' ability to communicate with external systems

Guideline 1

Help application builders to follow fundamental design principles

Offer good defaults, templates and automatic recommendations.

This guideline suggests two kinds of software support that could support application builders with design. Preventive support could stop users from making undesired design decisions. Reviewing support could help users improve existing application design. The support could be packaged into optional plugins/optional modifications. This approach allows builders to decide what type of design support they need. Design support as an integrated part of the product would probably be more tedious to develop, compared to separate modifications.

While some applications builders have design knowledge, many lack it. As a consequence, many struggle to follow essential principles, such as Nielsen's 10 heuristics [89] or the Gestalt Laws [93]. Builders may not have the tools to cope with growing applications, and as a result, dashboards can become crowded. The intended navigation within an application could also become hard to maintain. Moreover, applications sometimes lack a visual hierarchy and consequently become disorganised. How this category of support could be expressed in The Product has already been examined by a previous master thesis [94]. Custom components, such as complex menus or filters, might fail to comply with common design conventions. In summary, this guideline recommends two kinds of design support: preventive and reviewing.

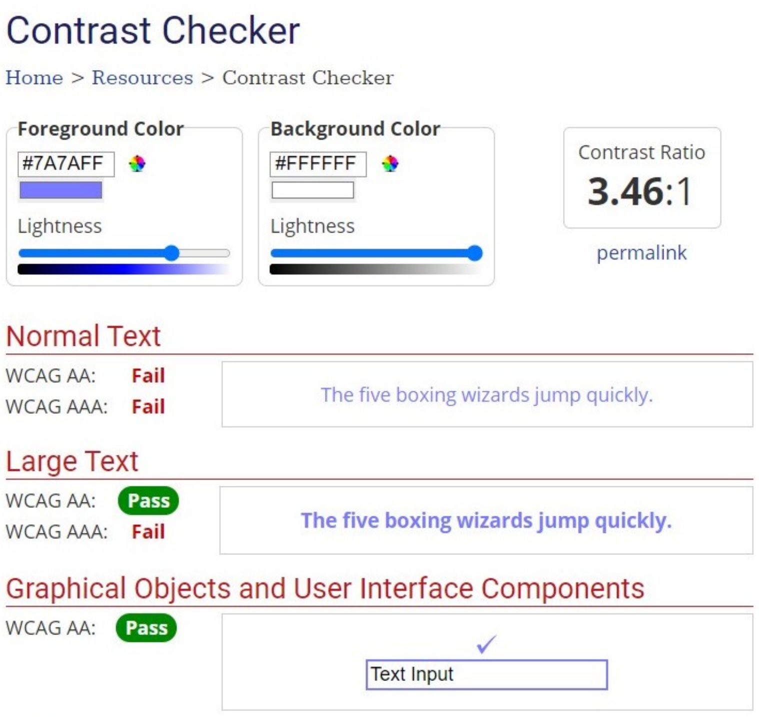
Preventive support could keep the application builder from including poor design decisions. One way to implement this support is by offering individual components with suitable defaults, and to offer templates for their behaviour as a group. By promoting an active community, it would also be possible to let customers create, share and use templates. Reviewing would be support that aims to correct poor design choices after they happened.

To offer software-based design feedback will be hard for some dimensions of design. Still, many tools for evaluation already exist. Many use simple mathematical equations to, for example, assess the accessibility of a GUI. Readability, colour contrast and colour-blindness are some these dimensions.

How to achieve it

Consider to publish a design system or a having single location for design resources. An example of this can be seen in Tableau's extension page, where they are providing design principles and guidelines for creating extensions that are similar to Tableau [14].

Offer optional design support plugins. An example of a function can be seen in figure 6.19.


Contrast Checker

Home > Resources > Contrast Checker

Foreground Color: #7A7AFF
Lightness

Background Color: #FFFFFF
Lightness

Contrast Ratio
3.46:1
permalink

Normal Text
WCAG AA: Fail
WCAG AAA: Fail
The five boxing wizards jump quickly.

Large Text
WCAG AA: Pass
WCAG AAA: Fail
The five boxing wizards jump quickly.

Graphical Objects and User Interface Components
WCAG AA: Pass
✓
Text Input

Figure 6.19: Webaim's contrast checker - one of many online tool available for evaluation of text and background colour combinations [95].

Guideline 2

Let the workspace contain unobtrusive contextual help

Provide rich help and clear entry points to external company resources in the workspace.

Since many application builders are not professional software developers, they need support with development. Regarding the help a program could offer, Brewer et al. writes that help systems, e.g. tool-tips and an FAQ, should be available at multiple levels. She also describes a correlation between the amount of information a system contains and the level of effort required by users to access it. Furthermore, sovereign desktop applications, described in section 3.8, should contain extensive modeless feedback. To conclude, accessing the information inline as, for instance, a tool-tip is easier than visiting an online community.

Many programs include direct access points to extensive help in brief help systems, to decrease the effort needed to obtain more information. By including a low effort to possibly read more, users do not have to begin a separate web search. Having to start a separate activity could break the user's flow within The Product. When viewing the users who build applications as developers, flow is shown to be a particularly contributing factor to their experience. This can be read more about in section 3.3.1.

How to achieve it

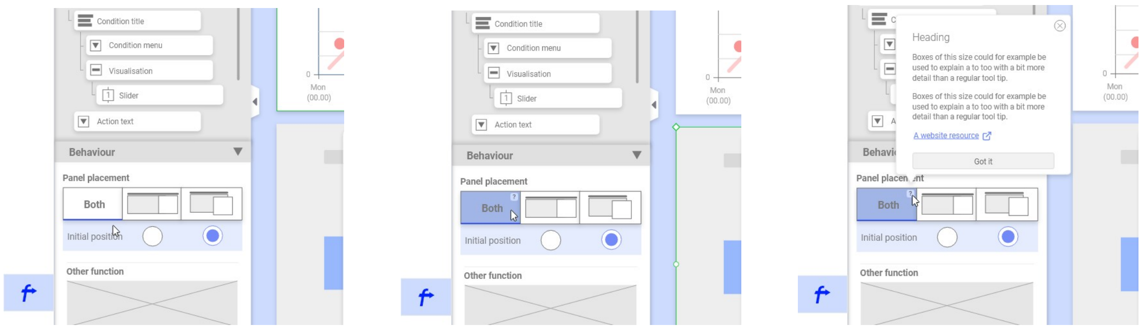
Provide contextually relevant information with entry points to external resources provided by The Company. By bringing existing information into the workspace, users are more likely to know where they can look for help.

Do not frame the support directly as help, rather as information. Many users avoid information strongly labelled as help since it makes them feel stupid [96].

Sometimes show first-time users large modeless help panels A large modeless panel can, with minimal interruption, suggest a variation of resources directly in the workspace.

Include a lot of visual elements to make information more appealing Let informative material reflect The Product's visual nature.

Rich in-line help


Heading

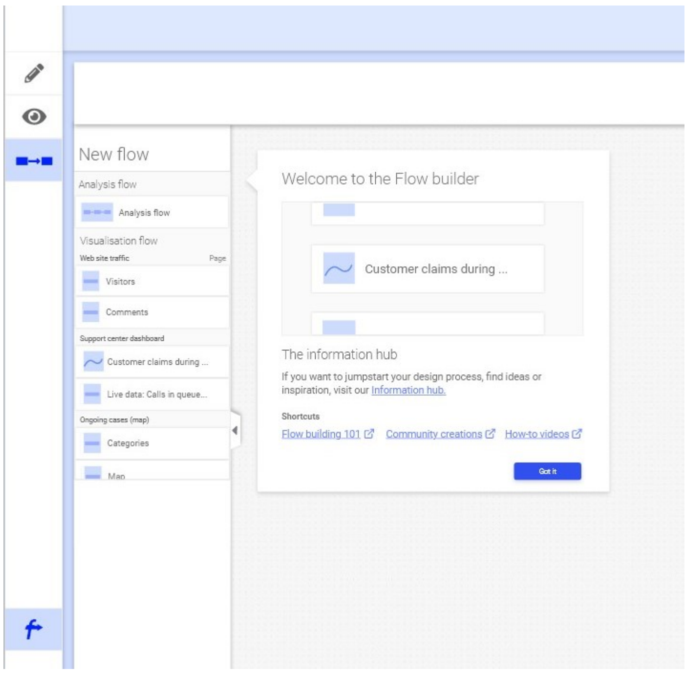
Boxes of this size could for example be used to explain a to too with a bit more detail than a regular tool tip.

Boxes of this size could for example be used to explain a to too with a bit more detail than a regular tool tip.

A website resource

Figure 6.20: An example from the prototype, demonstrating how the GUI could contain entry points to contextual help without becoming cluttered.

Large introduction panels These panels would appear automatically when the user encounters a new mechanism in the software. Having a large visual element could be beneficial. Its message would likely reach those who are quickly closes the panel. Users who read it more thoroughly could learn the location of suitable official resources about the mechanism.



Welcome to the Flow builder

The information hub

If you want to jumpstart your design process, find ideas or inspiration, visit our Information hub.

Shortcuts

Flow building 101 Community creations How-to videos

Figure 6.21: An example showing how users could receive information about workflows.

Guideline 3

Provide no-code development tools

To make application development less dependent on custom code.

Some users of analytical software are unable to use programming as a tool for application development due to insufficient technical knowledge. Others have the knowledge, but want to avoid application development through code. This opinion was discovered in the user interviews, see section 5.2.1. It can also be recognised in the general programming community, among novice programmers or people with no programming experience at all, according to end-user development (EUD) research [33].

Another subject in the previously mentioned interviews was that some company customers believe they should not let their data scientists program something unrelated to pure data analysis. In addition, custom code in some business areas requires approval from external agencies before use. This leads to inefficiency, since the approval process can be lengthy.

This guideline conflicts with G5 “Support application builders who customise through code”, but G5 could be viewed as an complement of this guideline.

How to achieve it

Allowing the application builders to create applications without having to program the components is a way of realising this guideline. Default components like buttons and menus could be provided, an example of this can be seen in fig 6.22 and in fig 6.23. Providing this could reduce the copy and paste of eventual code snippets between pages, and the applications would be easier to maintain in the long term.

Component-based building

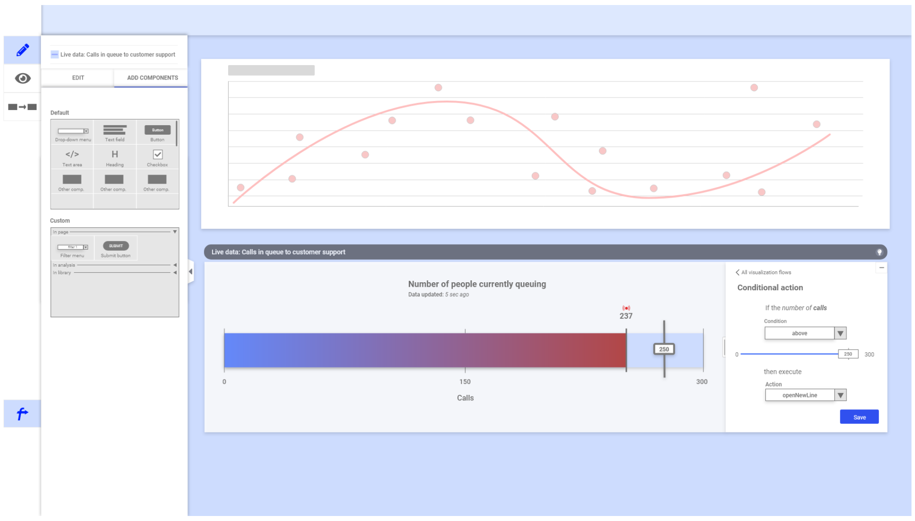

Live data: Calls in queue to customer support
EDIT
ADD COMPONENTS
Default
Custom
Number of people currently queuing
Calls
Conditional action
Save

Figure 6.22: Pre-made components that are provided by the software and can be instantly utilised by an application builder.

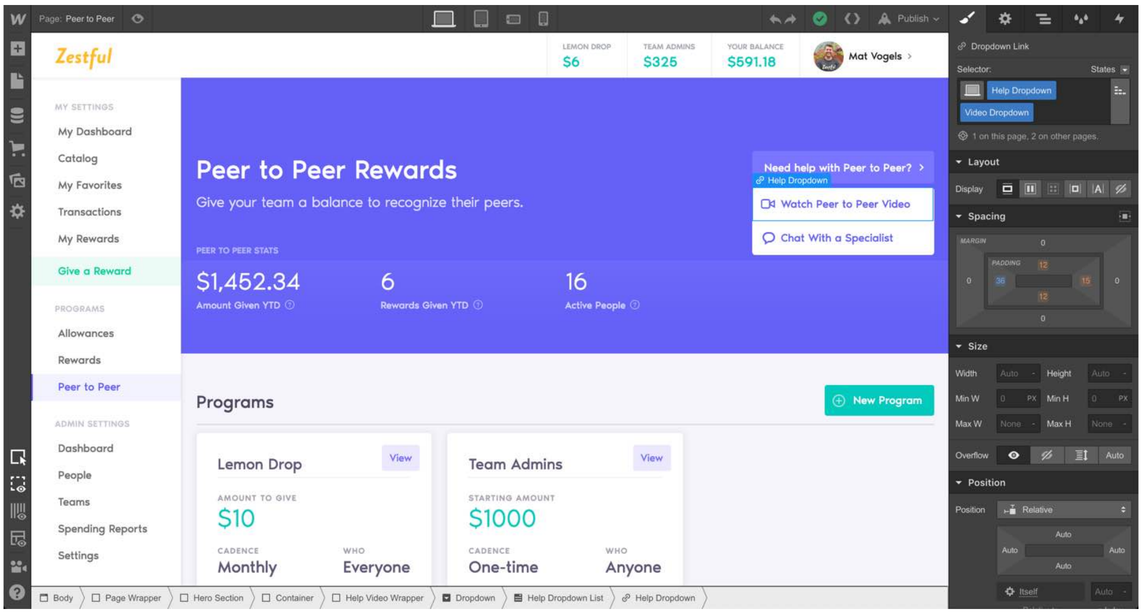


Figure 6.23: Build web applications by pre-made components, this example is from WebFlow [97].

Guideline 4

Consider The Gentle Slope of Complexity

All together, the tools for application building should adhere to The Gentle Slope of Complexity

The purpose of a gentle slope is to increase the users' ability to adapt the software they use, section 3.4.1 describes The Gentle Slope of Complexity in more detail. Software with a smooth slope will allow more users to transition from novice to experts adaptors. In the context of analytical applications, it means users first create applications by 'clicking' or drag and drop. In the end, they could grow to be comfortable with scripting. This transition is made possible by offering a sequential path towards the most complex adaption mechanisms/tools. Each point in the path represents an increasingly complex mechanism. Adaption through drag and drop could be the initial mechanism. Somewhere in the middle of the slope, users begin to write simple CSS code. Finally, application builders could become able to use scripting extensively.

When application builders have a goal that requires a mechanism they do not understand, such as scripting, they often delegate the task to an appropriate expert. Having to create the design through an external party will likely have a negative impact on the quality of the end result and result in increased costs [35].

How to achieve it

Define the software's slope. The slope represents the journey from novice to expert for a core function in a program.

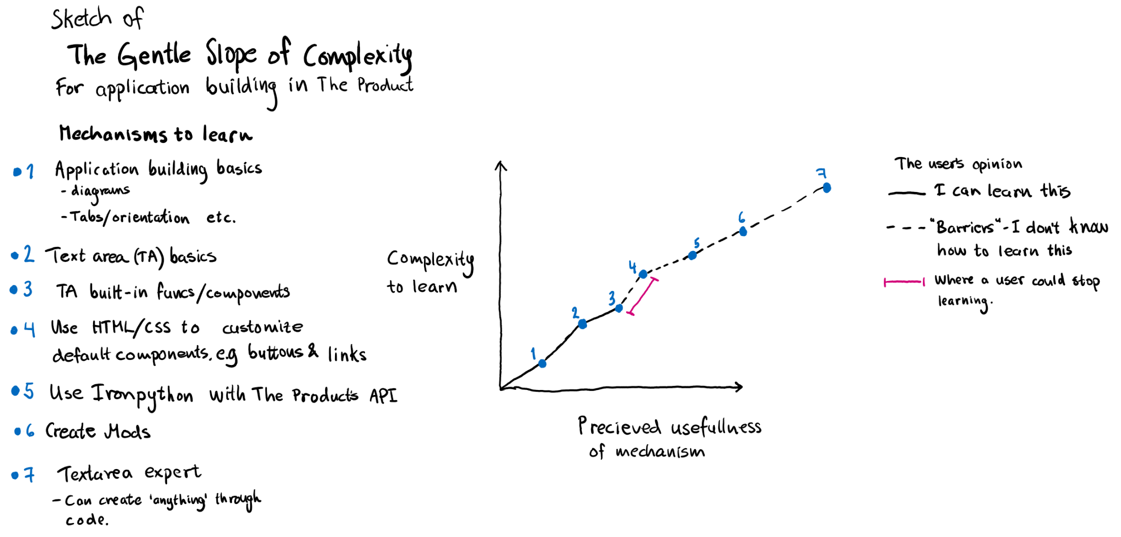


Figure 6.24: A draft of The Product's slope. It shows that most users stop learning when they need HTML / CSS to increase their adaptability.

Consider Hick's law

While working with an application, using the analytical software should require the smallest possible mental load. Hick's law [98] is a reminder of how easily options can become straining. Application builders can focus their energy on the application, when adjustments in The Product are straightforward. This consideration pairs well with guideline "G5 Support application builders who customise through code".

A curated set of options

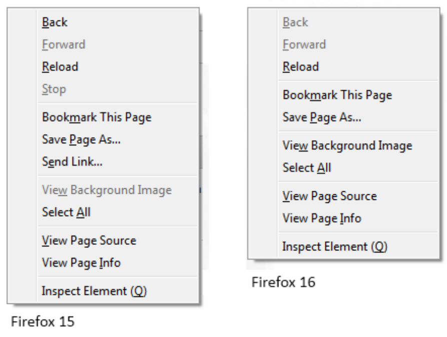


Figure 6.25: An example showing how Firefox applied Hick's Law in one of the browser's right-click menus [99].

Guideline 5

Support application builders who customise through code

Provide the possibility to edit and add custom code in applications.

Visual analytics is performed in several business sectors. Hence, the user group of analytics software is diverse. The diversity of such a user group produces various needs and requirements. A way of accommodating this is to provide highly customisable software, which allows the users to set the desired appearance and behaviour that their software needs. Customisable software is described in more detail in section 3.6. The need for highly customisable software was highlighted during the user interviews as well, see section 5.2.1.

This guideline is conflicting with "G3 Provide no-code development tools". The idea with this guideline is that every desired way of customisation is not predictable. To still accommodate the customisation requirements, providing the application builders with the ability to program the desired behaviours themselves is an option.

How to achieve it

There are several ways of constructing customisable software. One way is to allow the application builders to create their own scripts that can be executed by the software. This, in combination with an API of the software that can be utilised in the user-created scripts, makes it possible for the users to create almost any functionality they want. An example of this can be seen in fig 6.26. Letting application builders create scripts was emphasised as important during the interviews, especially by more advanced users (see section 5.2.1). One could also allow the application builder to edit pre-made components, as in “G7 Display a GUI component’s final state during editing”, as HTML. An example of this can be seen in fig 6.27.

Keeping scripts available

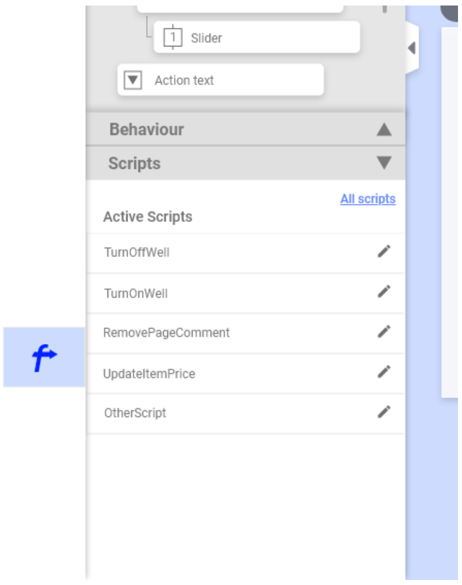
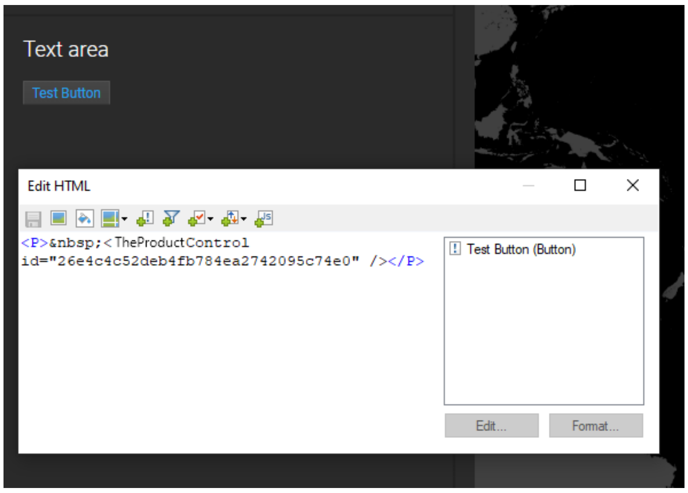


Figure 6.26: Scripts created by an application builder are gathered in a side panel.

Allow pre-made components to be edited via code



```
<P>&nbsp;<TheProductControl
id="26e4c4c52deb4fb784ea2742095c74e0" /></P>
```

Figure 6.27: A screenshot from The Product: HTML editing.

Guideline 6

Provide application builders with an adaptable workspace

Provide the possibility to hide elements unrelated to visualisations.

In a visual analytics custom application, the objects in focus should be visualisations. Therefore, it should be possible to place objects not directly related to visualisations in hidden menus, collapsible menus, side panels etc. Sometimes the purpose of an application may be the actions users take based on the visualisations though, causing the action itself to be as important as the visualisation. To decide this as an application builder was discovered to be of importance during user interviews, see section 5.2.1. As the purpose of an application may vary, letting the application builder decide what should be emphasised and where it should be placed is to consider. Also, Brewer et al. wrote that when allowing users to create artefacts, they should be provided with split-screen functionality. This is because users will resize their windows to work on several tasks simultaneously, even though it is not supported. Hence, it should be considered to support it natively.

How to achieve it

As above-mentioned, the application builders should be able to decide where to place objects and the size of those objects. The objects could be allowed to be placed in, for example, side panels and hidden menus, but also directly inside a visualisation. An example of when an application builder can decide this is shown in fig 6.28. Another example of when information that application users may want to hide to view the visualisations in full screen can be seen in fig 6.29.

Collapsible panels

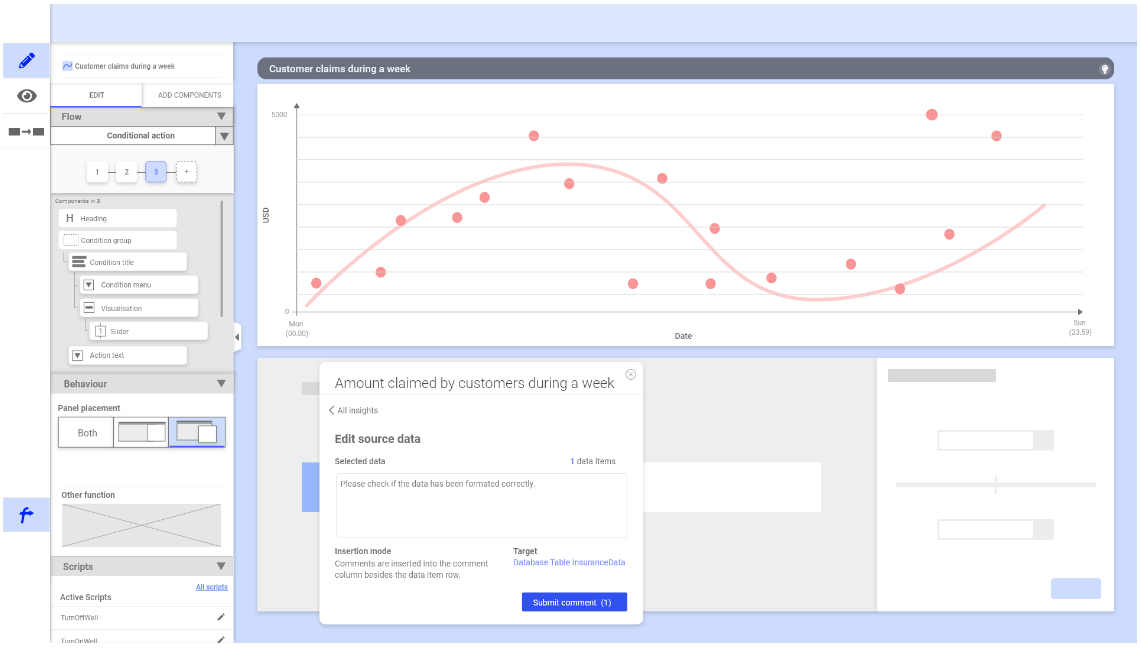


Figure 6.28: The application builder can decide whether the visualisation flow should be placed inside the visualisation or hidden in a menu.

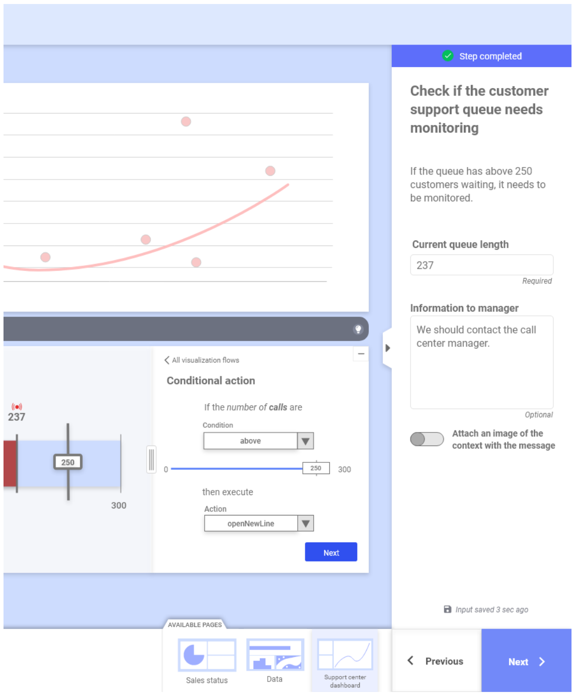


Figure 6.29: A collapsible side panel in an analysis flow.

Guideline 7

Display a GUI component's final state during editing

The process of customising application components should follow What You See Is What You Get.

What You See Is What You Get (WYSIWYG) is "a display generated by word-processing or desktop-publishing software that exactly reflects the document as it would appear in its finished state" [100]. In the context of application building, the intention is that the application builder can see how the application will look like while constructing it. Providing this interface style adds another layer of abstraction to the application building process. Hence, making application building more approachable to users with little or no programming knowledge [33]. A WYSIWYG interface also decreases the cognitive load of the application builder, as they do not have to imagine how the final result will look [91]. This interface style would be of particular value when objects on a dashboard resize, which was discovered during interviews in section 5.2.1 to be a common issue.

How to achieve it

When enabling WYSIWYG, it is important to consider "G6 Provide application builders with an adaptable workspace". Application builders should be allowed to decide the placement and size of objects. The WYSIWYG interface style can be seen in several applications, one example is in the web application builder Wix [7]. Other examples can be seen in fig 6.30 and from Miro [80] in fig 6.31.

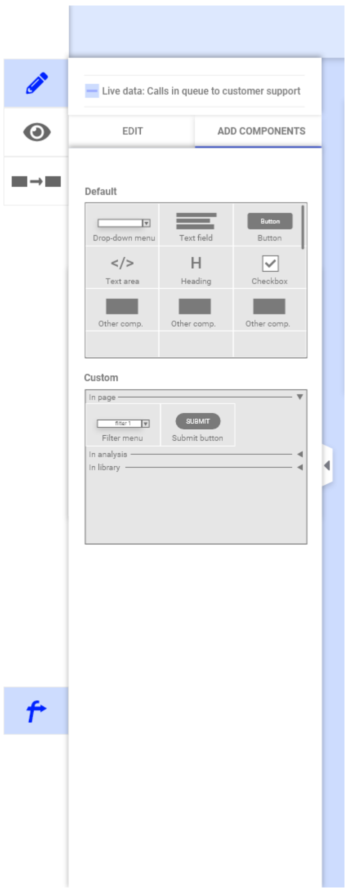


Figure 6.30: The application builder is provided with pre-made components in the side menu.

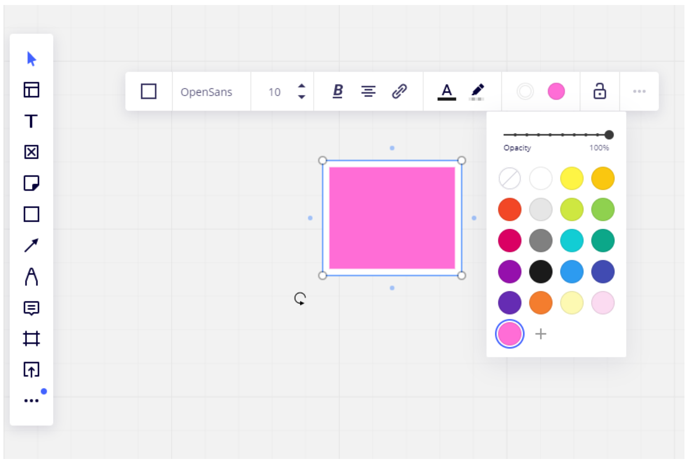


Figure 6.31: An example of a WYSIWYG interface in Miro [80]. A shape can be selected from the left menu and the shape's properties are configured in a menu that appears above the object when it is selected.

Guideline 8

Provide places that show all related application elements

Have designated locations in the GUI where application builders can view and manage related application elements.

When users build applications, they have to manage many different groups of objects, it could be buttons, scripts or visualisations. The amount in each group can also vary, and the functionality for managing them should be able to handle a fairly substantial increase in size. In other words, if most applications contain 20 buttons, there will still be some with more than a hundred. This issue with an unmanageable amount of components was recognised in applications studied in user interviews conducted by The Company, see section 5.1.2.

The program should also allow users to set the behaviour on different levels of objects. A level could be a single instance, all items on a page or within the application, it depends on the object's relationship with the program.

For some larger applications, even changing objects page by page could become tedious. For instance, many companies use design systems, described in section 3.5. If all components of a sort would have to change due to this system, the process would be facilitated by this type of overview functionality. As such, to avoid repetitive excise, the program should consider providing sections in the GUI where users can view all objects. Often it is likely wise to let such sections follow "G7 Display a GUI component's final state during editing". Being able to view the object's current appearance or at least part of it, makes it easier to achieve a consistent design.

Scripts can also be seen as objects, and users should be able to manage them similarly to GUI objects.

When the GUI offers a comprehensive set of tools for the customisation or configuration of objects, it should consider Hick's law [98]. Displaying all options at once could overwhelm the users. It might be better to provide a few by default, and offer clear paths to other ones. For tools not visible by default, the program distance should also be as short as possible.

How to achieve it

Figure 6.32 shows an example of application elements, components, are grouped and visualised in a hierarchy. This list also acts as an overview of the objects in the current canvas. Figure 6.33 shows hoe layers can be grouped and collapsed in Adobe Photoshop [101].

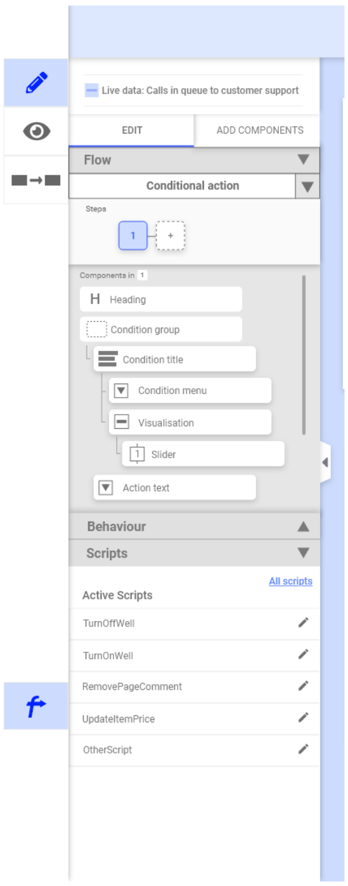


Figure 6.32: A panel for managing components in an application. The components can be grouped and their hierarchy is reflected in the list by indentations.

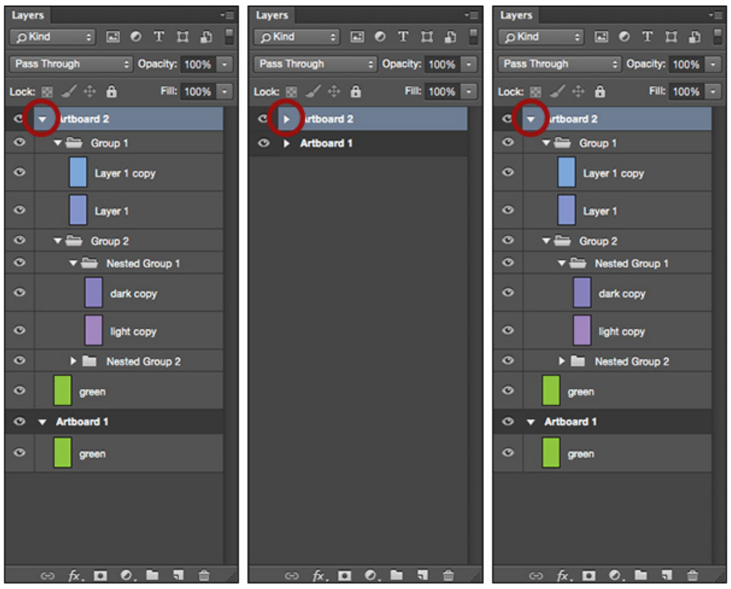


Figure 6.33: How different layers can be grouped and collapsed in Adobe Photoshop [102].

Guideline 9

Applications should have a default navigation structure

Provide tools and visual support that simply work with an application's navigation.

The feeling of being lost in an application is associated with an unpleasant user experience [103]. If the users do not know where they are, where they can go and what objects are related, the application quickly becomes complicated to use. In large applications, the navigational aspects become even more important to consider. This issue was discovered in the applications studied in the user interviews conducted by The Company, see section 5.1.2.

Interviews conducted during the thesis (see section 5.2.2) indicated that guided analyses are a common use case. Guided analyses are something in an application that is meant to guide the user step by step through a complex data analytics task. Depending on the size of the guide and the application, the user would be facilitated by both global, local and contextual navigation. This can be read more about in section 3.7.

How to achieve it

Navigational components could be utilised in not just guided analyses, but in any application that contains more than one page. Such navigational components could be a part of what is described in "G3 Provide no-code development tools". An important consideration with the components is to make them adaptable to the workspace, described more in "G6 Provide application builders with an adaptable workspace".

Examples of navigational components that would facilitate application builders are global menus, fly-out panels, and back and forward buttons. Figure 6.34 shows how progress indicators and back and forth buttons could facilitate the navigation.

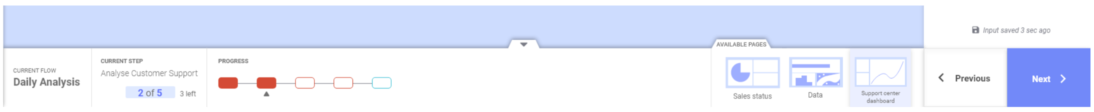


Figure 6.34: A bottom panel with a progress indicator and back and forth buttons. This shows the application user where they are and where they will go next.

Guideline 10

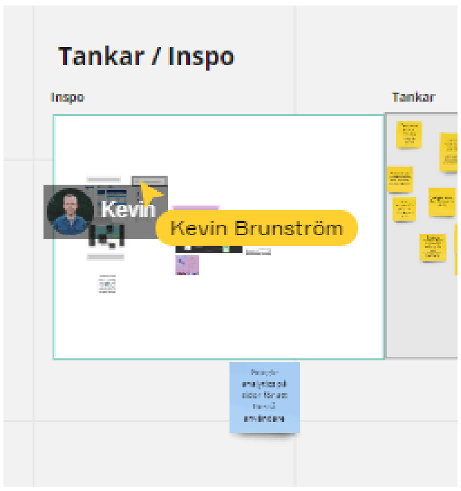
Offer visual and system support for collaborative application building

Consider enabling real-time, simultaneous collaboration.

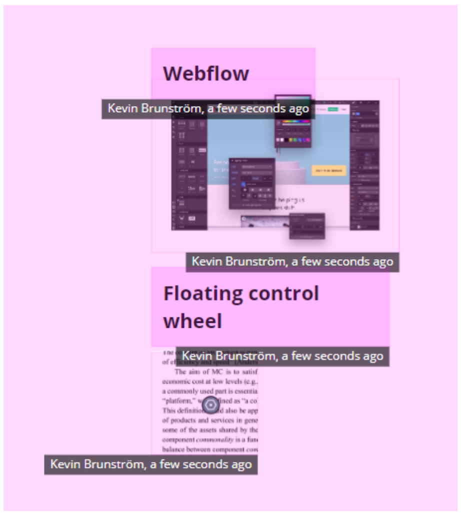
Data analysis is often about telling a story or conveying a message [5]. This naturally involves a person alone or a group building such a story, and a person or group as the recipient. In other words, data analysis is by nature a collaborative and social activity. Further, solving complex tasks is facilitated by working together, and building an application can be an example of such a complex task, according to end-user development [33]. It was discovered when studying interviews described in section 5.1.2 that building applications as a group was highly requested. Moreover, this was recognised in the design critique session described in section 5.4.4.

How to achieve it

To support collaboration when constructing applications can be realised in several ways. One way familiar for software developers is version control. To not cause the application building process seem like a code development project, this could be supported as in, for example, Adobe XD and Miro. How Miro enables collaboration can be seen in fig 6.35. Several application builders can be invited to an application building project and collaborate by dragging and dropping components onto canvases.



(a) A user can temporarily highlight the pointers of other users in a Miro canvas.


Webflow
Kevin Brunström, a few seconds ago
Kevin Brunström, a few seconds ago
Floating control wheel
Kevin Brunström, a few seconds ago
Kevin Brunström, a few seconds ago

(b) If a member has changed an object in a canvas, other users see it highlighted when they log in. This example is from Miro.

Figure 6.35: Different examples of collaboration features in Miro [80].

GUIDELINE 11

Allow application builders to create, store and share custom

workflows in multiple application layers

It should be possible to include them at several application layers.

As a function, the workflow prototype found in section 6.1, incorporates and supports many of the other guidelines. In summary, both the examined literature and interviews have detected this sort of functionality as beneficial. For instance, both the user interviews described in section 5.2.1 and the internal stakeholder interviews described in section 5.2.2 described guided workflows as a common use case. Furthermore, being able to create workflows brings value, since data analysis is increasingly becoming more and more of a collaborative activity [82]. The possible roles analytical applications could have is also growing. Application users can already analyse data, possibly in real time, and then affect external systems with a single click. Since the possibilities are nearly endless, the workflows have to be highly customisable, and be able to include numerous functionalities.

Most analysts are not professional software developers, and many also wish to minimise the amount of custom code in their application. This was discussed in both the earlier mentioned interviews. Enabling them to create workflows without having to write custom code thus supports them in their work.

How to achieve it

Workflows could be realised in analytical software in several ways. It could, for instance, be manifested as a simple to-do list incorporated in the application and ticked off by the application user. A more elaborate and detailed example of how workflows can be realised can be seen in section 6.2.

Guideline 12

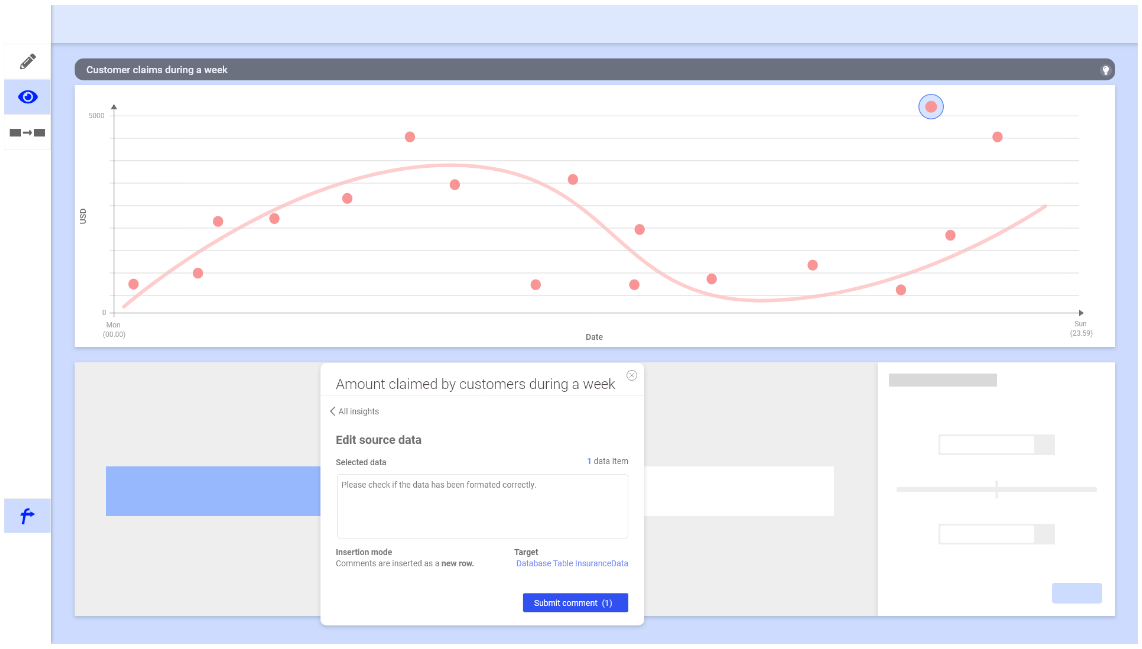
Support applications' ability to communicate with external systems

Consider to support interactions with common external system via GUI functionality.

As described in "G5 Support application builders who customise through code", visual analytics is performed in several business domains. As a consequence, customers may not use the same types of systems in their work. Besides the analytics software, they may all use one system for time reporting, another for email, one more for putting content on a website, and so on. Being able to affect external systems was discovered during interviews (see section 5.2.2) to be highly valued by users, as switching between different systems disrupts their daily workflows. For example, if an application is showing comments from a website, and a user of the application detects a comment with offensive language, they may want to block the person that commented directly in the application.

How to achieve it

If the application builder would like to integrate an external call that affects their other systems, a way of achieving that could be by API calls in a custom script, executed by the analytics software. In the example in fig 6.36, the application user wants to select a data point and write back metadata as a comment on that point to the source database. An example of when this is realised in an actual product is Tableau. In figure 6.37, a comment can be submitted on a data point, via an extension built with Tableau's extensions API [12].



Amount claimed by customers during a week

Edit source data

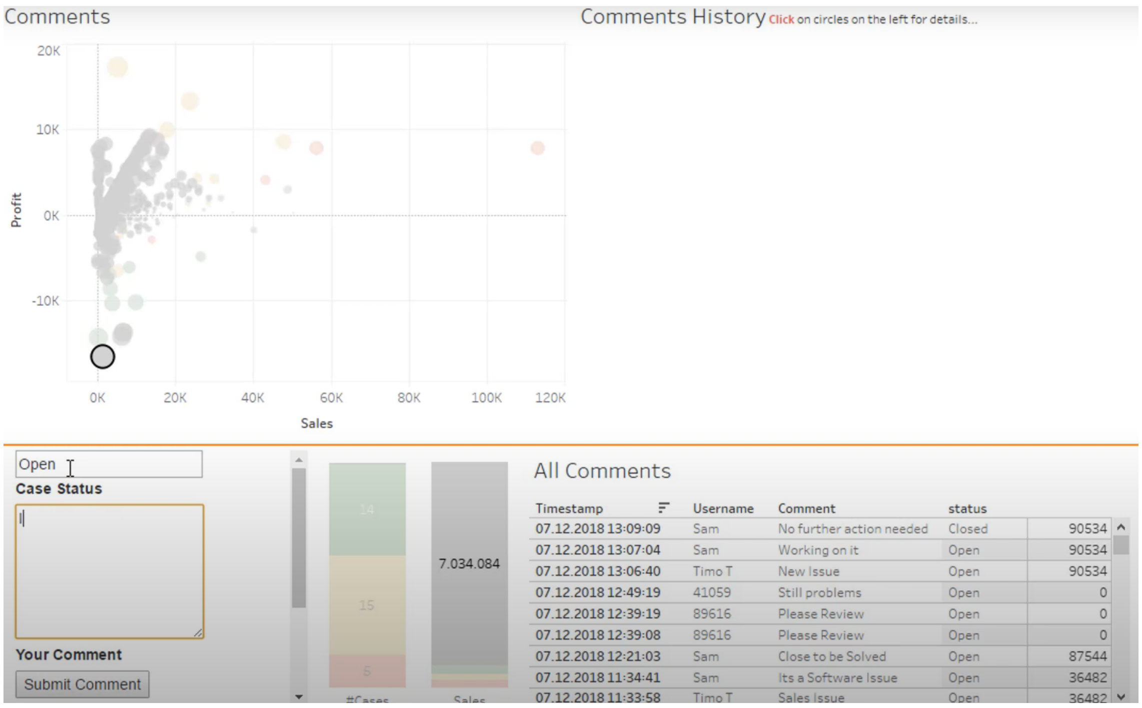
Selected data — 1 data item

Please check if the data has been formated correctly.

Insertion mode
Comments are inserted as a **new row.**

Target
Database Table InsuranceData

Figure 6.36: The application user selects a data point and writes metadata back as a comment on that point to the source database.



All Comments

Timestamp	Username	Comment	status	
07.12.2018 13:09:09	Sam	No further action needed	Closed	90534
07.12.2018 13:07:04	Sam	Working on it	Open	90534
07.12.2018 13:06:40	Timo T	New Issue	Open	90534
07.12.2018 12:49:19	41059	Still problems	Open	0
07.12.2018 12:39:19	89616	Please Review	Open	0
07.12.2018 12:39:08	89616	Please Review	Open	0
07.12.2018 12:21:03	Sam	Close to be Solved	Open	87544
07.12.2018 11:34:41	Sam	Its a Software Issue	Open	36482
07.12.2018 11:33:58	Timo T	Sales Issue	Open	36482

Figure 6.37: The user has selected a data point to comment on in a Tableau Extension [104].

7 Discussion

This chapter discusses the outcome of this thesis in relation to the initial expectations. Also addressed is what effect the reviewed literature, chosen methodology, and process had on the result.

Due to the research question's broad nature our work needed to find a specific focus, in the end it became end-user development (EUD) research. A broad answer would likely have resulted in a superficial knowledge contribution. Still, it is possible that a different focus would have provided a more satisfying answer to our research question. Besides this limitation, additional factors influence UX in general. Some factors not considered are, for example, organisational ones, hardware or events in a user's life. There is no reason to assume that these factors do not affect the experience during application building in analytical software.

Understanding whom users of analytical software are proved to be a complex topic. The line between user and developer of custom analytical applications is becoming increasingly blurred. While writing this thesis, The Company addressed this trend and replaced their personas with scenarios. Meanwhile, almost all EUD literature covered in this thesis distinguishes between developers and users. The situation is further complicated, as many application builders also use the applications they have built. To build and use is an iterative process, and few applications will reach a state of permanent completion.

7.1 Process

An early goal of the process was to decide from which perspective(s) the research question would be answered. During the first iteration, about three to four weeks were spent answering this question. As the whole execution period of this thesis was twelve weeks, one could argue this exploratory phase should have been shorter. If so, more time could have been spent testing and iterating the prototype and guidelines.

New perspectives that could answer the research question arose throughout the process, due to its highly exploratory nature. It was a time-consuming process to evaluate their potential value for the prototype and guidelines. Meanwhile, early discoveries, such as end-user development, also required time for refinement during each iteration. An example of a new perspective was navigation in applications, it got an increased focus during iteration two. We believed this perspective could be of value for the workflow concept. Consequently, work with the prototype often became a priority. This focus likely had a negative impact on the quality of the guidelines, even though findings from the prototype helped fuel their development. Allocating more time for the guidelines could have produced a better result. Mainly since guideline development requires empathy with application builder and analytical software developers.

Additional user tests, in particular evaluation tests, would have been beneficial. The conducted user tests were done with users who lack experience in analytical software and The Product. We did not do any summative evaluation of the prototype, partly since it was still at conceptual level. The intention with the prototype was to evaluate the guidelines and inspire future development of analytical software. A separate concept like workflows was not initially planned. Nevertheless, one could question the validity of the workflow concept and guidelines when considering the lack of summative evaluation. Therefore, the result should be regarded as possible considerations and suggestions.

7.2 Result

At first glance, the prototyp's fidelity appears high. Still, the concept of workflows is merely a concept. The fidelity of the prototypes could have been lower, to emphasize this in discussions with stakeholders and other designers. A lower fidelity could have removed some focus on details in the design. Also, more time could have been spent on developing the concept and primary parts of its interactions with application builders and users. For example, one issue for application builders discovered early in the process was the shareability of application components. Currently, there are no such components, but several application builders have requested them. Therefore, providing shareable and reproducible application components were considered early in the process. We imagined that the workflows could be shared via The Product's library, but realised workflows are currently bound to specific data and visualisation. The whole visualisation could of course be shared, but the actual visualisation flow with its canvas and steps is likely the parts application builders want to share. The same goes for the analysis flows. It is probably the side panel and progress steps that application builders wish to share, unbound to an application.

Another aspect of the prototype that lacks exploration is the programming aspect. The idea is that scripts can be attached to components and workflows, but how it can be achieved was not examined. Therefore, the prototype does not provide much value for the application builders who wish to code their applications. There are two reasons for this. One is that The Product targeted initially people who want to create visualisations of some data without code. The other reason is that those application builders who program applications are today the ones who can build applications at all. At least, that was our perception when researching the users who build applications. With that said, when discussing a possible place in The Product’s future, a product manager said it would probably first be provided via the application programming interface (API). Hence, probably enabling the workflow concept for programming application builders.

When work began with the guidelines, the primary audience was yet to be decided. This thesis had three main stakeholders; vendors of analytics software, application builders, and application users. Only in the last iteration was it decided whether the guidelines should target suppliers of analytical software or application builders. The final guidelines are directed towards the vendors of analytics software and suggest how they can support the application builder’s experience. We chose them since the research question focuses on improving the applications’ users’ experience. And application builders need support from the provider and its software to improve the experience. Another possibility would have been: one for the vendors and one for the application builders. However, to some degree, the presented guidelines do contain more direct advice for application builders. It should also be mentioned that all guidelines are based on the identified needs of application builders.

Producing generalisable guidelines was another goal with this thesis, and this trait has not been evaluated after the guidelines were completed. It would have been beneficial to learn how other vendors of analytical software value these guidelines. Nevertheless, generalisability was taken into account throughout the development process. Some existing solutions in The Product were included in the guidelines, even though they already exist. The solutions included can be described as larger mechanisms and concepts, their descriptions do not reveal any trade secrets. In our opinion, this approach strengthens the guidelines’ generalisability.

In connection with discussions about the primary audience, there were discussions about the scope of each guideline. One could find some guidelines universal and applicable for software in general, not only application building analytics software. However, as mentioned earlier, each guideline connects to one or more needs that builders have. The guideline regarding collaboration, for instance, might apply to nearly all sovereign desktop applications. Still, application builders viewed extensive software support for collaboration as essential. EUD literature also stressed its importance.

The guidelines could also have been presented in a different format. If the purpose had been to have a strong impact on the world of analysts, they would probably have been presented differently. They could have been more graphic, concise and powerful. That would have made them easier to remember. As of now, they adhere to academic rigour. While some might consider them lengthy, many others will hopefully find them inspiring and reliable.

7.3 Ethical Issues

An application builder could theoretically build any application they want in The Product. This includes applications to unethical monitoring of employees, maximising profits from online casinos, mapping minorities in cities, and the list is growing long. As long as data is collected on the subject, you could create an application in this respect. The question here is who you allow building analytical applications, not the enabling of building them per se.

Other issues with end-user development arise when the development is performed through code. The code could be malicious, either by accident or not. Who's fault is it when code in an application accidentally shuts down a whole system? This was something we did not focus on, as security with all its aspects is probably a whole thesis alone. To avoid the most obvious security problems in terms of code, our solutions rely mainly on no- or low-code.

7.4 Future Work

After evaluating the iterations, it becomes clear that each iteration examined different parts of the workflow, not with consideration to the whole concept. Each part of the concept would have benefited from further research in terms of both interaction and design options. Much effort was spent with *how* workflows could function, after they became the prototype's main theme. It was not feasible to define all details of their behaviour: there had to be some uncertainty. Sometimes it caused stress and led us to spend too much effort on minor details. The purpose of the design was to examine how workflows can be manifested in an application from the perspective of the user. To overlook how workflows are created and to focus only on the user experience of the application led to more ideas that are visible in the final prototype. Hence, how the workflows are controlled and executed from an application user perspective is the more thoughtful part of the concept. Letting application builders build applications and exclude those applications' users' experiences is probably a whole master's thesis alone.

Besides not iterating every part of the workflow concept enough, more user tests would have been beneficial. The Company attempted with no luck to get us in contact with company customers' employees for interviews. Therefore, the validations of the prototype have either been on employees at The Company or persons with no experience of analytical software and The Product at all. To evaluate the actual value of the workflow concept or the interactions of creating workflows, it would have been interesting to interview an experienced but not employed person.

Something out of scope yet almost explored is visual scripting. Visual scripting has been implemented in, for example, Scratch [105]. Visual scripting was touched upon during the exploration of visualisation flows. One visualisation flow allows the application user to set a conditional action depending on the visualised data. The actions are executed by "if, then" statements, but with text and drop-downs instead of pure code (see fig 6.3). The Company saw potential in these kinds of visualisation flows, and they would have been interesting to explore further as they capture the low-code approach of applications.

8

Conclusion

This thesis has, together with a company, explored the application building process inside an analytics platform. Also covered was the user experience within such applications. The following research question has been answered:

What should be considered when supporting application builders who aim to fulfil their end-users' desired experience in analytic software?

The thesis followed an iterative design process, which included research through design. Each iteration explored application building inside analytics software while simultaneously creating guidelines for providing such features in such software. Each iteration ended with a formative evaluation of the prototype and updated guidelines. Through this approach, the thesis aimed to produce practical design suggestions and research contributions. The produced guidelines are stated as follows:

1	Help application builders to follow fundamental design principles
2	Let the workspace contain unobtrusive contextual help
3	Provide no-code development tools
4	Consider The Gentle Slope of Complexity
5	Support application builders who customise through code
6	Provide application builders with an adaptable workspace
7	Display a GUI component's final state during editing
8	Provide places that show all related application elements
9	Applications should have a default navigation structure
10	Offer visual and system support for collaborative application building
11	Allow application builders to create, store and share custom workflows in multiple application layers
12	Support applications' ability to communicate with external systems

To conclude, the guidelines aim to support application builders in achieving their goals based on their prerequisites, rather than solving specific use case. This can be solved by providing the application builders with sufficient tools that support their building process.

Bibliography

[1] The Economist, "Fuel of the future, data is giving rise to a new economy," [Online]. Available: https://www.economist.com/briefing/2017/05/06/data-is-giving-rise-to-a-new-economy, 2017, accessed: 2021-12-10.

[2] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0268401214001066

[3] IDG, "Data and analytics survey," [Online] Available: https://cdn2.hubspot.net/hubfs/1624046/IDGE_Data_Analysis_2016_final.pdf?t=1492090430059, 2016, accessed: 2021-11-23.

[4] Keith D. Foote, "A brief history of analytics," [Online]. Available: https://www.dataversity.net/brief-history-analytics/, 2018, accessed: 2021-05-18.

[5] W. Cui, "Visual analytics: A comprehensive overview," *IEEE Access*, vol. 7, pp. 81 555–81 573, 2019.

[6] Unity, "Unity for all," [Online]. Available: https://unity.com/, 2021, accessed: 2021-12-09.

[7] Wix, "Skapa en professionell hemsida," [Online]. Available: https://sv.wix.com/, 2021, accessed: 2021-12-09.

[8] Infor, "Infor birst," [Online]. Available: https://www.infor.com/solutions/advanced-analytics/business-intelligence/birst, 2021, accessed: 2021-02-11.

[9] Microsoft, "Hitta klarhet när du behöver det," [Online]. Available: https://powerbi.microsoft.com/sv-se/, 2021, accessed: 2021-02-11.

[10] Tableau Software, "Answer questions at the speed of thought with tableau desktop," [Online]. Available: https://www.tableau.com/products/desktop, 2021, accessed: 2021-02-01.

[11] ——, "Tableau developer tools," [Online]. Available: https://www.tableau.com/developer/tools, 2021, accessed: 2021-02-01.

[12] ——, “Tableau dashboard extensions api,” [Online]. Available: https://tableau.github.io/extensions-api/, 2021, accessed: 2021-02-09.

[13] ——, “Design guidelines for dashboard extensions,” [Online]. Available: https://tableau.github.io/extensions-api/docs/ux_design.html, 2021, accessed: 2021-02-01.

[14] ——, “Tableau ui,” [Online]. Available: https://tableau.github.io/tableau-ui/, 2021, accessed: 2021-02-09.

[15] Facebook Inc., “Components and props,” [Online]. Available: https://reactjs.org/docs/components-and-props.html, 2021, accessed: 2021-02-09.

[16] QlikTech International AB, “Agent of transformation,” [Online]. Available: https://www.qlik.com/us/products/qlik-sense, 2021, accessed: 2021-02-01.

[17] ——, “Get started,” [Online]. Available: https://qlik.dev/, 2021, accessed: 2021-02-01.

[18] MicroStrategy Inc, “Join us at microstrategy world.now: our first virtual, interactive global conference,” [Online]. Available: https://www.microstrategy.com/en, 2021, accessed: 2021-02-01.

[19] ——, “Sdk,” [Online]. Available: https://community.microstrategy.com/s/topic/0TO44000000FliLGAS/sdk?language=en_US&lc_version=2021, 2021, accessed: 2021-02-01.

[20] ——, “Dossier authoring,” [Online]. Available: https://doc-archives.microstrategy.com/producthelp/10.11/Dossier_Authoring/WebHelp/Lang_1033/Content/About_MicroStrategy_Desktop.htm, 2021, accessed: 2021-02-11.

[21] R. Buchanan, “Wicked problems in design thinking,” *Design Issues*, vol. 8, p. 5–21, 1992.

[22] H. W. J. Rittel and M. M. Webber, “Dilemmas in a general theory of planning,” 1973. [Online]. Available: https://link.springer.com/article/10.1007/BF01405730#citeas

[23] C. Ware, *Information Visualization: Perception for Design.* Morgan Kaufmann Publishers Inc., 2004.

[24] F. Burstein, C. Holsapple, S. Negash, and P. Gray, “Chapter 45 business intelligence,” in *Handbook on Decision Support Systems 2 Variations.* Springer-Verlag, 2008, p. 175–195.

[25] D. Seminsar, 2010. [Online]. Available: http://www.allaboutux.org/files/UX-WhitePaper.pdf

[26] Allaboutux.org, “User experience definitions,” [Online]. Available: https://community.microstrategy.com/s/topic/0TO44000000FliLGAS/sdk?language=en_US&lc_version=2020, 2012, accessed: 2021-01-29. [Online]. Available: http://www.allaboutux.org/ux-definitions

[27] D. Saffer, *Designing for interaction : creating innovative applications and devices.*, ser. Voices that matter. New Riders Pub, 2010.

[28] F. Fagerholm and J. Münch, “Developer experience: Concept and definition,” in *2012 International Conference on Software and System Process (ICSSP)*, 2012, pp. 73–77.

[29] K. Kuusinen, “Are software developers just users of development tools? assessing developer experience of a graphical user interface designer,” [Online] Available: https://link.springer.com/chapter/10.1007%2F978-3-319-44902-9_14, Aug 2016, accessed: 2021-11-23.

[30] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience.* Harper & Row, 1990.

[31] N. J. Mihaly Csikszentmihalyi, Abuhamdeh S, “Flow,” in *Handbook of Competence and Motivation*, 2005, pp. 596–608.

[32] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, “End-user development: An emerging paradigm,” *SpringerLink*, vol. 9, Jan 2006.

[33] F. Paternò, “End user development: Survey of an emerging field for empowering people,” *ISRN Software Engineering*, Jun 2013.

[34] G. Fischer, K. Nakakoji, and Y. Ye, “Metadesign: Guidelines for supporting domain experts in software development,” *IEEE Software*, vol. 26, no. 5, pp. 37–44, 2009.

[35] M. Spahn, C. Dörner, and V. Wulf, “End user development: Approaches towards a flexible software design,” 2007. [Online]. Available: http://www.cs.cmu.edu/~./cdoerner/slides/EUD_Literatur_ECIS2008.pdf

[36] T. Ludwig, J. Dax, V. Pipek, and V. Wulf, “A practice-oriented paradigm for end-user development,” in *New Perspectives in End-User Development*, 2017.

[37] S. Vesselov and T. Davis, *Introducing Design Systems.* Apress, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4842-4514-9_2

[38] A. Kholmatova, *Design Systems: A Practical Guide to Creating Design Languages for Digital Products.* Smashing Media AG, 2017. [Online]. Available: https://books.google.se/books?id=UWhMswEACAAJ

[39] Jad Limcaco, “Design systems gallery,” [Online]. Available at: https://designsystemsrepo.com/design-systems, 2021, accessed: 2021-01-19.

[40] Adobe, “What’s new,” [Online]. Available: https://spectrum.adobe.com/page/whats-new/, 2021, accessed: 2021-02-08.

[41] ——, “Spectrum, adobe’s design system,” [Online]. Available: https://spectrum.adobe.com/, 2021, accessed: 2021-02-08.

[42] Microsoft, “Fluent design system,” [Online]. Available: https://www.microsoft.com/design/fluent/#/, 2021, accessed: 2021-02-08.

[43] T. Kuo, “Mass customization and personalization software development: A case study eco-design product service system,” *Journal of Intelligent Manufacturing*, vol. 24, 2013.

[44] M. Kratochvíl and C. Carson, *Growing modular: Mass customization of complex products, services and software.* Springer-Verlag Berlin Heidelberg, 01 2005.

[45] B. Hui, S. Liaskos, and J. Mylopoulos, “Requirements analysis for customizable software: a goals-skills-preferences framework,” in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, 2003, pp. 117–126.

[46] L. Rosenfeld, P. Morville, and J. Arango, “Defining information architecture,” in *Information architecture for the World Wide Web.* O’Reilly, 2015, pp. 23–39. [Online]. Available: https://www.vlebooks.com/vleweb/Product/Index/629255?page=0

[47] ——, “Defining information architecture,” in *Information architecture for the World Wide Web.* O’Reilly, 2015, p. 177. [Online]. Available: https://www.vlebooks.com/vleweb/Product/Index/629255?page=0

[48] A. Cooper, D. Cronin, C. Noessel, R. Reimann, and A. Cooper, “About face: The essentials of interaction design,” in *About face: the essentials of interaction design.* John Wiley and Sons, 2014, p. 205–237.

[49] C. Frayling and R. C. of Art (Great Britain), *Research in Art and Design*, ser. Royal College of Art research papers. Royal College of Art, 1993.

[50] J. Zimmerman, J. Forlizzi, and S. Evenson, “Research through design as a method for interaction design research in hci,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’07. Association for Computing Machinery, 2007, p. 493–502. [Online]. Available: https://doi.org/10.1145/1240624.1240704

[51] E. B.-N. Sanders and P. J. Stappers, “Co-creation and the new landscapes of design,” *CoDesign*, vol. 4, no. 1, pp. 5–18, 2008. [Online]. Available: https://doi.org/10.1080/15710880701875068

[52] L. Sanders, “On modeling an evolving map of design practice and design research,” *Interactions*, vol. 15, no. 6, p. 13–17, 2008.

[53] A. Pratt and J. Nunes, *Interactive Design: An Introduction to the Theory and Application of User-centered Design.* Rockport Publishers, 2012.

[54] D. Norman and J. Euchner, “Design for use,” *Research-Technology Management*, vol. 59, no. 1, pp. 15–20, 2016. [Online]. Available: https://doi.org/10.1080/08956308.2016.1117315

[55] J. Jones, *Design Methods*, ser. A VNR book. Wiley, 1992.

[56] Google LLC, “Transform the way your team works,” [Online]. Available: https://designsprintkit.withgoogle.com/, 2021, accessed: 2021-12-10.

[57] ——, “Phase 1: Understand,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase1-understand, 2021, accessed: 2021-02-08.

[58] B. Hanington and B. Martin, *Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions.* Rockport Publishers, 2012.

[59] Google LLC, “Share and engage with the design sprint community,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase2-define, 2021, accessed: 2021-02-04.

[60] Design Council, “What is the framework for innovation? design council’s evolved double diamond,” [Online]. Available: https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond, 2021, accessed: 2021-02-04.

[61] The Interaction Design Foundation, “What is design thinking?” [Online]. Available: https://www.interaction-design.org/literature/topics/design-thinking, 2021, accessed: 2021-02-04.

[62] Google LLC, “The golden path,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase2-define/golden-path, 2021, accessed: 2021-02-04.

[63] A. Cooper, R. Reimann, D. Cronin, C. Noessel, and A. Cooper, “Chapter 3 modeling users: Personas and goals,” in *About Face, the Essentials of Interaction Design*, 4th ed. Wiley, 2014, p. 61–99.

[64] C. Gray and M. Siegel, “Sketching design thinking: Representations of design in education and practice,” *Design and Technology Education: an International Journal*, vol. 19, no. 1, 2014.

[65] Google LLC, “Phase 3: Sketch,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase2-define, 2021, accessed: 2021-01-29.

[66] IDEO, “Methods,” [Online]. Available: https://www.designkit.org/methods, 2021, accessed: 2021-01-29.

[67] Google LLC, “Crazy 8’s,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase3-sketch/crazy-8s, 2021, accessed: 2021-05-13.

[68] ——, “Phase 4: Decide,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase4-decide, 2021, accessed: 2021-02-08.

[69] ——, “Present solution sketches,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase4-decide/present-solution-sketches, 2021, accessed: 2021-02-08.

[70] ——, “Dot vote,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase4-decide/dot-vote, 2021, accessed: 2021-02-08.

[71] P. S. P. Rex Hartson, “Chapter 11 prototyping,” in *The UX Book Process and Guidelines for Ensuring a Quality User Experience.* Elsevier, 2012, p. 391–427.

[72] J. Rudd, K. Stern, and S. Isensee, “Low vs. high-fidelity prototyping debate,” *Interactions*, vol. 3, no. 1, p. 76–85, 1996. [Online]. Available: https://doi.org/10.1145/223500.223514

[73] Google LLC, “Phase 6: Validate,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase6-validate, 2021, accessed: 2021-02-05.

[74] P. S. P. Rex Hartson, “Chapter 2 the wheel: A lifecycle template,” in *The UX Book Process and Guidelines for Ensuring a Quality User Experience.* Elsevier, 2012, p. 47–75.

[75] ——, “Chapter 12 ux evaluation introduction,” in *The UX Book Process and Guidelines for Ensuring a Quality User Experience.* Elsevier, 2012, pp. 427–467.

[76] ——, “Chapter 13: Rapid evaluation methods,” in *The UX Book Process and Guidelines for Ensuring a Quality User Experience.* Elsevier, 2012, pp. 467–503.

[77] Google LLC, “Cognitive walkthroughs,” [Online]. Available: https://designsprintkit.withgoogle.com/methodology/phase6-validate/cognitive-walkthroughs, 2021, accessed: 2021-02-05.

[78] S. Carr, “What is remote usability testing?” [Online]. Available: https://www.usertesting.com/blog/what-is-remote-usability-testing, 2021, accessed: 2021-02-05.

[79] ——, “Moderated vs. unmoderated usability testing: the pros and cons,” [Online]. Available: https://www.usertesting.com/blog/moderated-vs-unmoderated-usability-testing, 2021, accessed: 2021-02-05.

[80] Miro, “Where telecommuting teams get work done,” [Online]. Available: https://miro.com/, 2021, accessed: 2021-04-12.

[81] Balsamiq Studios, LLC, “Quick and easy wireframing tool,” [Online]. Available: https://balsamiq.com/wireframes/, 2021, accessed: 2021-04-12.

[82] A. Crisan, “Reflecting on a decade of data science and the future of visualization tools,” 2021. [Online]. Available: https://www.tableau.com/about/blog/2021/2/data-science-and-future-visualization-tools-reflection

[83] Google LLC, “Google scholar,” [Online]. Available: https://scholar.google.com/, 2021, accessed: 2021-03-18.

[84] IEEE, “Advancing technology for humanity,” [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp, 2021, accessed: 2021-03-18.

[85] ACM, Inc., “Acm digital library,” [Online]. Available: https://dl.acm.org/, 2021, accessed: 2021-03-18.

[86] Google LLC, “Google,” [Online]. Available: https://google.com/, 2021, accessed: 2021-03-18.

[87] A. Tiwari and A. K. Sekhar, “Workflow based framework for life science informatics,” *Computational Biology and Chemistry*, vol. 31, no. 5, pp. 305–319, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1476927107001107

[88] W. v. d. Aalst and E. Damiani, “Processes meet big data: Connecting data science with process science,” *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 810–819, 2015.

[89] J. Nielsen, “10 usability heuristics for user interface design,” [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/, 2020, accessed: 2021-05-03.

[90] L. Murphy, M. B. Kery, O. Alliyu, A. Macvean, and B. A. Myers, “Api designers in the field: Design practices and challenges for creating usable apis,” in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2018, pp. 249–258.

[91] D. Lightbown, *Designing the user experience of game development tools.* CRC Press, 2017.

[92] Webflow, Inc., “The modern way to build for the web,” [Online]. Available: https://webflow.com/, 2021, accessed: 2021-05-07.

[93] Interaction Design Foundation, “Gestalt principles,” [Online]. Available: https://www.interaction-design.org/literature/topics/gestalt-principles, 2021, accessed: 2021-05-10.

[94] A. Dhareshwar, “Make it easy to make analyses look good,” [Online]. Available: https://hdl.handle.net/20.500.12380/255507, 2018, accessed: 2021-05-12.

[95] Webaim, “Contrast checker,” [Online]. Available: https://webaim.org/resources/contrastchecker/, 2021, accessed: 2020-05-07.

[96] C. Brewer, J. Tidwell, and A. Valencia-Brooks, *Designing Interfaces, 3rd Edition.* O'Reilly Media, Incorporated, 2019. [Online]. Available: https://books.google.se/books?id=pMw8zQEACAAJ

[97] Webflow, Inc., "Case study: How zestful built its marketing site — and web app — with webflow," [Online]. Available: https://webflow.com/blog/how-zestful-built-its-marketing-site-and-web-app-with-webflow, 2021, accessed: 2021-05-07.

[98] Interaction Design Foundation, "Hick's law: Making the choice easier for users," [Online]. Available: https://www.interaction-design.org/literature/article/hick-s-law-making-the-choice-easier-for-users, 2020, accessed: 2021-05-13.

[99] J. Wein, "Flow: The psychology of optimal experience," [Online]. Available: https://msujaws.wordpress.com/2012/07/23/applying-hicks-law-to-the-firefox-context-menus/, 2012, accessed: 2021-05-10.

[100] Merriam-Webster Inc., "Wysiwyg," [Online]. Available: https://www.merriam-webster.com/dictionary/WYSIWYG, 2021, accessed: 2021-05-04.

[101] Adobe Inc., "Skapa. omskapa. med photoshop." [Online]. Available: https://www.adobe.com/se/products/photoshop.html, 2021, accessed: 2021-05-17.

[102] J. Kost, "Displaying layer group and artboard contents in the layers panel," [Online]. Available: https://jkost.com/blog/2015/11/displaying-layer-group-and-artboard-contents-in-the-layers-panel.html, 2015, accessed: 2021-05-17.

[103] L. Rosenfeld, P. Morville, and J. Arango, "Defining information architecture," in *Information architecture for the World Wide Web.* O'Reilly, 2015, pp. 175–211. [Online]. Available: https://www.vlebooks.com/vleweb/Product/Index/629255?page=0

[104] Tableaufans com, "Tableau commenting extension," [Online]. Available: https://www.youtube.com/watch?v=ekfXBI2qnZs, 2018, accessed: 2021-05-11.

[105] Scratch Foundation, "Create stories, games, and animations. share with others around the world." [Online]. Available: https://scratch.mit.edu/, 2021, accessed: 2021-05-07.

A
Interview Questions

Interview questions from the user interviews described in section 5.2.1.

- Could you describe how you consider navigation for users during the development process for an application? Great if you have examples!
  - Does the size of an application have an impact on how you consider navigation? E.g. 5 pages vs 60 pages.
  - When you follow a step-by-step process or explore data, how do you avoid having users feeling “lost” or not knowing what they should do next?
- We would like to hear your view on possible challenges when implementing navigation for users in applications.
  - In general, what do you view as the biggest challenges when implementing navigation in applications? Are some use cases particularly challenging?
  - And to be a bit more specific: Have you seen or done any kind of “work-around” to achieve the desired navigation functionality in an analysis?
- Can you think of any particular workflows / restricted flows that are valuable to customers, but challenging to implement? Or ones you would like to be able to build with ease?
  - When do you do that? How do you typically create it? Great if you have an example or could show us the steps needed!

One of the interviewees has worked with a rather new feature called *Visualization Mods*, described in 2.2.1. Therefore, a question regarding those was added:

- We also understood that you work with mods, how do you think it would be for users to use mods as a tool to create workflows?

Interview questions for the internal stakeholders, described in section 5.2.2.

Questions for the interview with the product manager whose focus is The Product as a whole:

- *(The interviewee has built a demo application in The Product where the users can, for example, view comments from their website and block IP addresses of spam comments.)* Can you describe the process of building the demo application?
  - Generally, what are you most uncertain of during the application development process? Is there something typical? If not, think of a certain case.
- We have got the impression that many who use applications want to share more than direct insights. For example, an analyst who wants to reproduce the complicated changes he has done in recent hours to a manager, colleague or similar. What kind of support does The Product give them today, and what kind of functionality do you think could have made it even easier?
- How do you experience the customers' attitude towards writing code in The Product?
  - Can you describe a common use case where customers wish that they did not have to code?
- What kind of support do new customers of The Product get today? Is there a difference between different customers? In that case, how?
- When it comes to building applications or big analyses in The Product, what development do you see that The Product has to have to continue attracting customers?
- The developers of applications in The Product is a diverse group, how do you relate to that?

Questions for the interview with the product manager whose focus is The Product's API:

- How do you experience the application building process in The Product? What is working and not?
- What kind of functionality do customers request regarding application development?
  - What is your experience of how customers experience application development in The Product?

- We have got the impression that there is a desire among some users to be able to write and manage a lot of their own code - to get the application they want. While others - so to speak - do not want to code a line (HTML / CSS) when building applications. How are you / the product management team trying to meet these different needs?

- Do you think users would have appreciated if some common use cases with the API could also be done using a graphical user interface in The Product?

- Is there a particular use case regarding Visualization Mods that would be interesting to highlight?

- We understand that you worked on building the community section to extend The Product. How do you see the role and place of the community, today some programs choose to integrate the help into The Product, closer to the work itself, so to speak. Is that something you think The Product's customers would have appreciated?

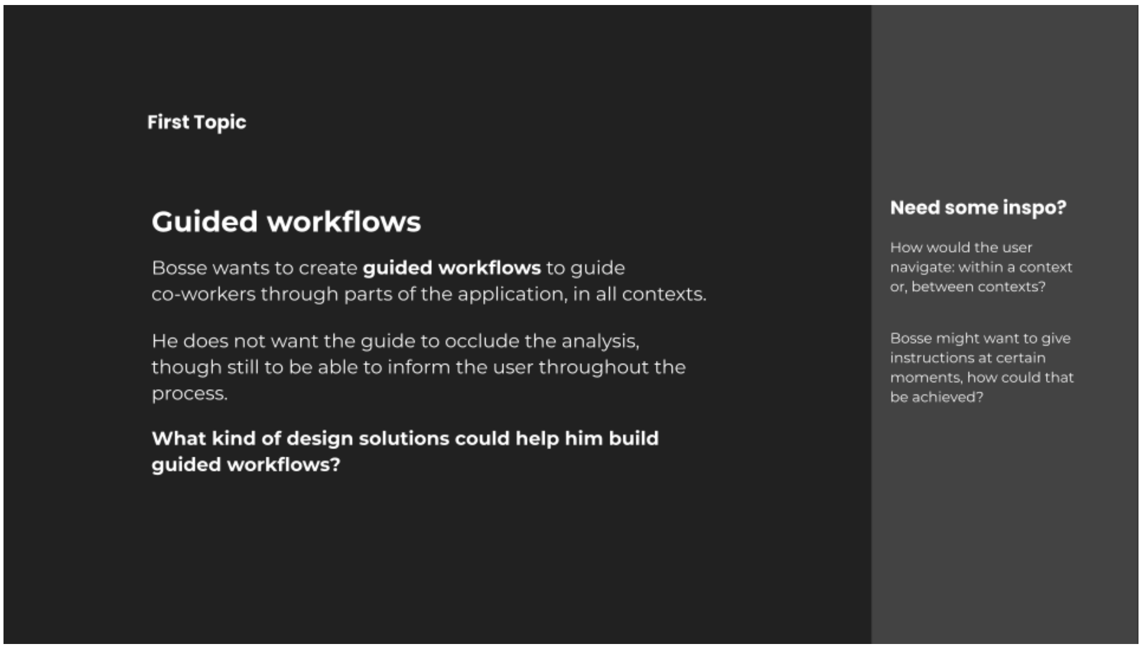
- When it comes to building applications or big analyses in The Product, what development do you see that The Product has to have to continue attracting customers?

- Could you tell us about what kind of workflows that customers need to be able to do?

B

Ideation Workshop

The following figures the main slides that were presented during the ideation workshop, described in section 5.2.4. Figure B.1 describes the firs topic, figure B.2 the second, and figure B.3 the third.



First Topic

Guided workflows

Bosse wants to create **guided workflows** to guide co-workers through parts of the application, in all contexts.

He does not want the guide to occlude the analysis, though still to be able to inform the user throughout the process.

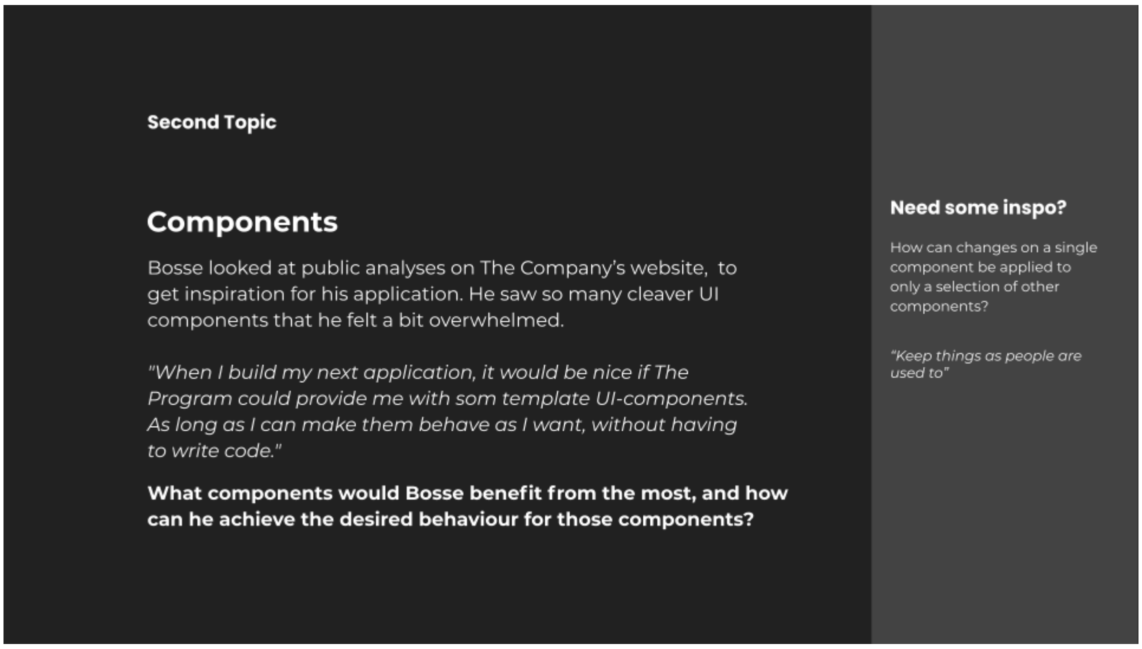
What kind of design solutions could help him build guided workflows?

Need some inspo?

How would the user navigate: within a context or, between contexts?

Bosse might want to give instructions at certain moments, how could that be achieved?

Figure B.1: The first topic of the workshop. It was about guided workflows in applications.



Second Topic

Components

Bosse looked at public analyses on The Company's website, to get inspiration for his application. He saw so many cleaver UI components that he felt a bit overwhelmed.

"When I build my next application, it would be nice if The Program could provide me with som template UI-components. As long as I can make them behave as I want, without having to write code."

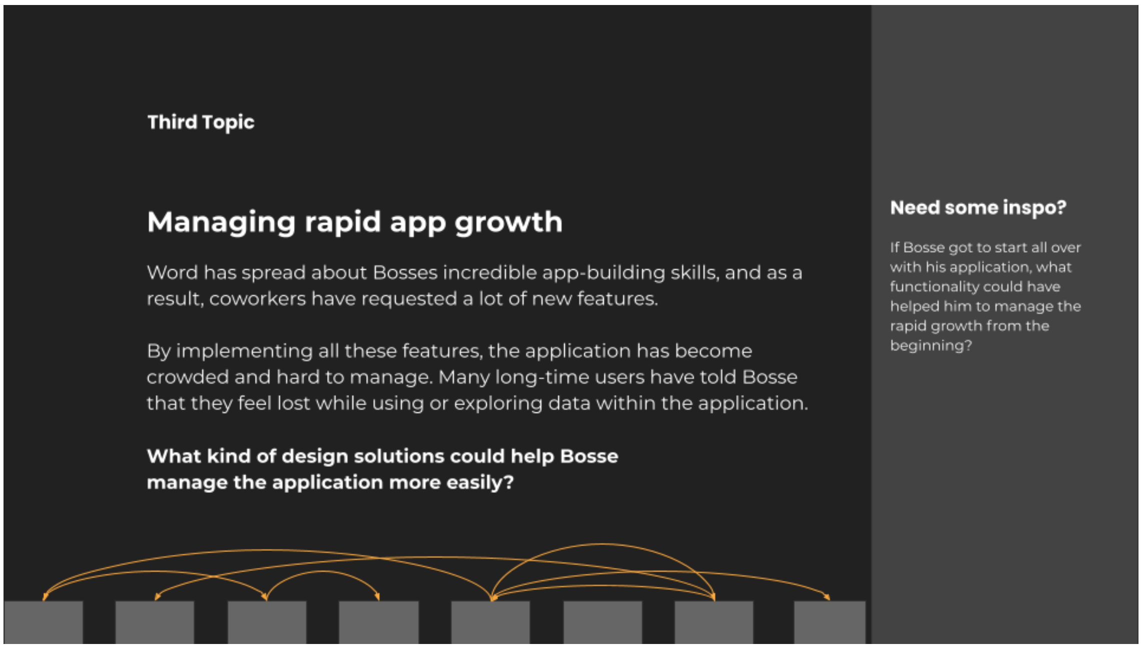
What components would Bosse benefit from the most, and how can he achieve the desired behaviour for those components?

Need some inspo?

How can changes on a single component be applied to only a selection of other components?

"Keep things as people are used to"

Figure B.2: The second topic of the workshop. It was about creating application components without code.



Third Topic

Managing rapid app growth

Word has spread about Bosses incredible app-building skills, and as a result, coworkers have requested a lot of new features.

By implementing all these features, the application has become crowded and hard to manage. Many long-time users have told Bosse that they feel lost while using or exploring data within the application.

What kind of design solutions could help Bosse manage the application more easily?

Need some inspo?

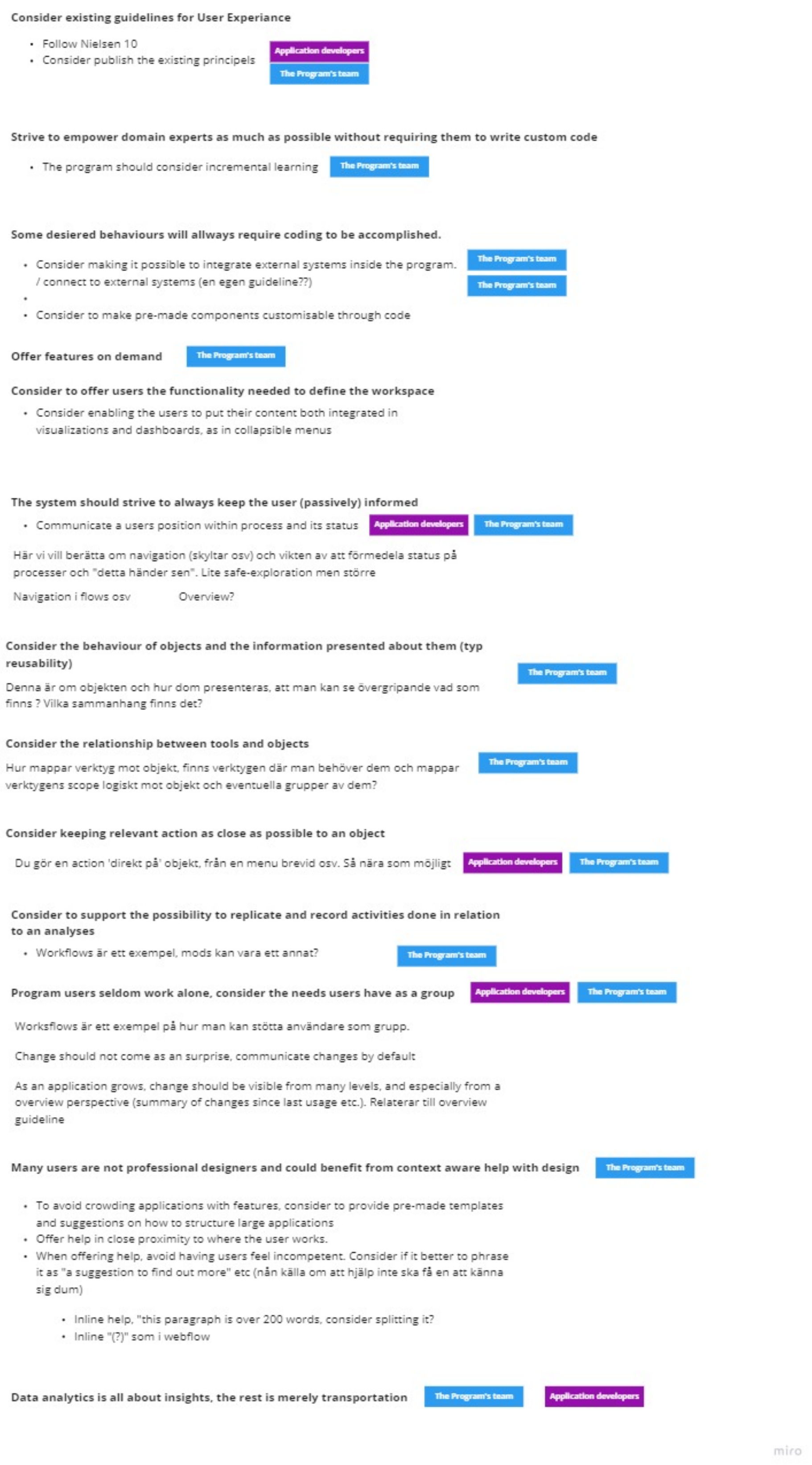
If Bosse got to start all over with his application, what functionality could have helped him to manage the rapid growth from the beginning?

Figure B.3: The third topic of the workshop. It was about managing rapid application growth, and applications that suffer from feature creep.

C

Guidelines v. 2

In this version of the guidelines, there is still ambiguity regarding the term "user", meaning that "user" can refer to both the application builder and the application user. When the guidelines were created in Miro, an attempt of creating tags either directed towards The Company or the application builders was made. The second version of the guidelines can be seen in fig C.1.



Consider existing guidelines for User Experiance

- Follow Nielsen 10
- Consider publish the existing principels

Application developers
The Program's team

Strive to empower domain experts as much as possible without requiring them to write custom code

- The program should consider incremental learning The Program's team

Some desiered behaviours will allways require coding to be accomplished.

- Consider making it possible to integrate external systems inside the program. / connect to external systems (en egen guideline??) The Program's team The Program's team
-
- Consider to make pre-made components customisable through code

Offer features on demand The Program's team

Consider to offer users the functionality needed to define the workspace

- Consider enabling the users to put their content both integrated in visualizations and dashboards, as in collapsible menus

The system should strive to always keep the user (passively) informed

- Communicate a users position within process and its status Application developers The Program's team

Här vi vill berätta om navigation (skyltar osv) och vikten av att förmedela status på processer och "detta händer sen". Lite safe-exploration men större

Navigation i flows osv Overview?

Consider the behaviour of objects and the information presented about them (typ reusability) The Program's team

Denna är om objekten och hur dom presenteras, att man kan se övergripande vad som finns ? Vilka sammanhang finns det?

Consider the relationship between tools and objects The Program's team

Hur mappar verktyg mot objekt, finns verktygen där man behöver dem och mappar verktygens scope logiskt mot objekt och eventuella grupper av dem?

Consider keeping relevant action as close as possible to an object

Du gör en action 'direkt på' objekt, från en menu brevid osv. Så nära som möjligt Application developers The Program's team

Consider to support the possibility to replicate and record activities done in relation to an analyses

- Workflows är ett exempel, mods kan vara ett annat? The Program's team

Program users seldom work alone, consider the needs users have as a group Application developers The Program's team

Worksflows är ett exempel på hur man kan stötta användare som grupp.

Change should not come as an surprise, communicate changes by default

As an application grows, change should be visible from many levels, and especially from a overview perspective (summary of changes since last usage etc.). Relaterar till overview guideline

Many users are not professional designers and could benefit from context aware help with design The Program's team

- To avoid crowding applications with features, consider to provide pre-made templates and suggestions on how to structure large applications
- Offer help in close proximity to where the user works.
- When offering help, avoid having users feel incompetent. Consider if it better to phrase it as "a suggestion to find out more" etc (nån källa om att hjälp inte ska få en att känna sig dum)
  - Inline help, "this paragraph is over 200 words, consider splitting it?
  - Inline "(?)" som i webflow

Data analytics is all about insights, the rest is merely transportation The Program's team Application developers

Figure C.1: The second version of the guidelines in progress.

D
User Test Questions

The following are the questions that were asked during the user tests described in 5.4.2. An important notice is that these questions were written in connection to the prototype. Hence, they are tightly coupled with the prototype's state at the time.

- *(On canvas without things on)* How would you use this panel?
- Did the window help you understand the panel?
- *(On stepping through analysis flow as a user)* Do you understand where you are in the process?
- *(On edit in analysis flow vis flow)* Where would you press to see this as a user?
- *(On the preview for analysis flow as a user)* Do you need to do anything to save what you have entered here?
- How would you change the value here? (1900)
- In what ways do you think you can change?
- *(After got it in edit mode)* What visualization can we change now?
- *(On flow that is not inline)* Here the flow is not inline, where would you look for it?
- What data are you acting on now?
- If you want to change what data is affected, what would you like to do then?
- How do you do for 1 point and 3?
- As a user, did you want to choose whether it "flows freely" or is inline?