



TECHNISCHE HOCHSCHULE NÜRNBERG  
GEORG SIMON OHM

Fakultät Informatik

# Analyse und Überarbeitung des Graphical User Interfaces von EB GUIDE Studio 6 zur Steigerung der Usability

Bachelorarbeit im Studiengang Medieninformatik

vorgelegt von

Sandra Schumann

Matrikelnummer 302 0357

Erstgutachter: Prof. Dr. Korbinian Riedhammer

Zweitgutachter: Prof. Dr. Matthias Teßmann

© 2020

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.



## Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Schumann

Vorname: Sandra

Matrikel-Nr.: 3020357

Fakultät: Informatik

Studiengang: Medieninformatik

Semester: Wintersemester

2019/2020

### Titel der Abschlussarbeit:

Analyse und Überarbeitung des Graphical User Interfaces von EB GUIDE Studio 6 zur Steigerung der Usability

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

---

Ort, Datum, Unterschrift Studierende/Studierender

## Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☒ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von            Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigelegt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

---

Ort, Datum, Unterschrift Studierende/Studierender



## Kurzdarstellung

Die vorliegende wissenschaftliche Arbeit behandelt die Analyse und anschließende, teilweise Überarbeitung des User Interfaces von EB GUIDE Studio 6, mit der Zielsetzung dessen Usability zu verbessern. Um dies zu erreichen, wurde sich an den einzelnen Iterationsschritten des Human-Centered Design Process orientiert. Für die Identifizierung der Schwächen im Interface wurden Modellierer innerhalb der Zielgruppe bei ihrer täglichen Arbeit beobachtet und befragt. Für drei dieser Schwächen wurden, nach allgemein gültigen Gestaltungsprinzipien, Verbesserungen erarbeitet, welche teilweise mithilfe eines Prototyping Tools und teilweise, im bestehenden Projekt, mit C# und WPF umgesetzt wurden. Die zu bearbeitende Testaufgabe wurde so ausgelegt, dass die Nutzer, während des Tests, mit allen eingearbeiteten Verbesserungen interagieren. Um vergleichbare Werte zu erhalten, wurde die Aufgabe von je fünf Nutzern mit dem bestehenden und dem überarbeiteten Interface durchgeführt. Bei der Messung der Usability, wurde bei der Auswertung der Tests auf die Effizienz und Fehlerrate der Nutzer geachtet.

Nach einer Iteration des Design Process wird deutlich, dass die Anpassungen die Usability teilweise erhöht haben, die Verbesserungen jedoch noch Schwächen enthalten, die Nachbesserung verlangen. Hierfür können, aufbauend auf der, durch diese Arbeit bereit gestellten Grundlage, weitere Iterationen des Design Process durchgeführt werden, bis die Verbesserungen die Benutzeranforderungen hinreichend erfüllen.



# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Elektrobit Automotive GmbH	1
1.2. Abteilung User Experience	1
1.3. Motivation	2
1.4. Zielsetzung	2
1.5. Aufbau der Arbeit	3
<b>2. Theorie</b>	<b>5</b>
2.1. Grafische Benutzeroberfläche	5
2.2. Ergonomie	5
2.3. Usability	6
2.4. User Experience	8
2.5. Kognitive Last	9
2.6. Human - Centered Design Process	10
2.7. Gestaltprinzipien der Usability	11
2.8. EB Guide Studio	17
<b>3. Analysen</b>	<b>21</b>
3.1. Ausgangssituation	21
3.2. Vorgehensweise	22
3.3. Ergebnisse	23
3.4. Verbesserungen	29
3.4.1. Auswahlkriterien	29
3.4.2. Auswahl nach Auswahlkriterien	31
3.4.3. Design der Verbesserungen	33
<b>4. Umsetzung</b>	<b>37</b>
4.1. Prototyp Multiselektion und Template Properties	38
4.1.1. Axure RP	38
4.1.2. Interaktionsmöglichkeiten	42
4.1.3. Vorgehensweise	43
4.2. Implementierung Filter	49
4.2.1. Zielsetzung	49
4.2.2. Projektaufbau	49

4.2.3. Vorgehensweise . . . . .	51
<b>5. Usability Test . . . . .</b>	<b>55</b>
5.1. Lookback . . . . .	55
5.2. Remote Usability Test . . . . .	56
5.3. Arbeitsaufgaben . . . . .	58
5.4. Ergebnisse . . . . .	61
5.4.1. Ergebnisse überarbeitetes Interface . . . . .	62
5.4.2. Ergebnisse altes Interface . . . . .	65
5.4.3. Vergleich und Interpretation . . . . .	66
5.5. Ausblick . . . . .	68
<b>6. Fazit . . . . .</b>	<b>71</b>
<b>A. Subtasks . . . . .</b>	<b>75</b>
<b>B. Arbeitsaufgabe Implementierter Filter . . . . .</b>	<b>79</b>
<b>C. Arbeitsaufgabe Prototyp . . . . .</b>	<b>81</b>
<b>D. Arbeitsaufgabe bestehende Software . . . . .</b>	<b>83</b>
<b>Abbildungsverzeichnis . . . . .</b>	<b>85</b>
<b>Tabellenverzeichnis . . . . .</b>	<b>87</b>
<b>Auflistungen . . . . .</b>	<b>89</b>
<b>Literaturverzeichnis . . . . .</b>	<b>91</b>



# Kapitel 1.

## Einleitung

In diesem einführenden Kapitel wird zunächst kurz das Partnerunternehmen der Abschlussarbeit mit der zugehörigen Abteilung User Experience vorgestellt. Weiterhin wird die Motivation dieser Arbeit und das verfolgte Ziel erläutert, bevor es noch einen Überblick über die folgenden Kapitel gibt.

### 1.1. Elektrobit Automotive GmbH

Partner der Bachelorarbeit ist die Firma Elektrobit Automotive GmbH - im Folgenden nur noch als EB bezeichnet. EB ist ein vielfach ausgezeichnetes, internationales Unternehmen, welches sich auf die Entwicklung von Produkten und Dienstleistungen im Bereich der Automobilindustrie spezialisiert hat. Mit über 30 Jahren Branchenerfahrung bietet EB seinen Kunden unter anderem innovative Lösungen für das vernetzte Fahrzeug, Human Machine Interface Technologien (HMI), Navigations- und Fahrassistenzsysteme und Steuergeräte. Die Automotive Software von EB befindet sich in über 1 Billionen Geräten, die in mehr als 90 Millionen Fahrzeugen weltweit Verwendung finden. Mit über 2300 beschäftigten Mitarbeitern, verteilt auf 3 Kontinente und 9 Länder, und einer durchschnittlichen jährlichen Wachstumsrate von über 10 %, ist EB ein weltweit etabliertes Unternehmen mit Hauptsitz in Erlangen[\[Eleka\]](#).

### 1.2. Abteilung User Experience

Jedes Gerät, das für den alltäglichen Gebrauch gedacht ist, sollte eine erfolgreiche Interaktion gewährleisten. Dafür ist eine Schnittstelle zwischen Mensch und Maschine (HMI), die einen intuitiven, einfachen und schnellen Umgang mit diesem Gerät ermöglicht, unabdingbar. Um die Erfahrungsqualität im Allgemeinen möglichst hoch zu halten, wünschen Nutzer sich auf ihre Bedürfnisse angepasste User-Interfaces in allen Lebensbereichen, womit der Bereich UX auch in der Automobilbranche einen hohen Bedeutungsgrad genießt. Die

Abteilung User Experience von EB befasst sich vor allem mit der Entwicklung multimodaler HMIs für Kombiinstrumente, Head Units und Head-Up Displays. Diese werden von EB von der Konzeptphase bis hin zur Serienentwicklung mit Hilfe der Software EB-GUIDE entwickelt.

### 1.3. Motivation

Die, in dieser Arbeit untersuchte Software, EB GUIDE, wird sowohl intern bei Elektrobit genutzt, als auch extern als Produkt vertrieben. Bei firmeninterner Nutzung eines Produktes, hängt die Effizienz der damit durchgeführten Arbeit, direkt von dessen Usability ab. Zeit, die Nutzer damit verbringen unklare Funktionen der Software zu verstehen, stellt bezahlte Arbeitszeit dar, in der jedoch kein tatsächlicher Arbeitsfortschritt verzeichnet werden kann. Darüber hinaus führt das fehlerhafte Ausführen, oder nicht Auffinden von Softwarefunktionen zu steigender Unzufriedenheit des Nutzers, was ebenfalls die Produktivität negativ beeinflusst.

Für den Vertrieb eines Softwareproduktes ist schlechte Usability ebenfalls fatal. Sobald es mehr, als eine Software für ein Anwendungsgebiet gibt, werden Firmen die Software nutzen, welche die beste Usability aufweist. Diese Entscheidung begründet sich auf extern auf den gleichen Argumenten, die firmenintern für das Nutzen einer Software mit hoher Usability sprechen.

Eine schlechte Usability von EB Guide hat demnach sowohl interne, als auch vertriebliche, negative Folgen, weshalb Elektrobit es anstrebt die Usability der Software fortlaufend zu verbessern.

### 1.4. Zielsetzung

Ziel dieser Bachelorarbeit ist es, Schwachstellen im User Interface von EB GUIDE zu identifizieren und durch deren Verbesserung die Usability von EB GUIDE zu erhöhen. Dafür ist zuerst eine Analyse der Arbeitsabläufe innerhalb der Modellierungsarbeit nötig, um entsprechende Probleme im User Interface zu erkennen. Anschließend gilt es, Konzepte zu entwickeln welche, durch Anpassung und Überarbeitung der entsprechenden Komponenten in der Benutzerschnittstelle, diese Probleme beheben oder minimieren. Diese Konzepte werden anschließend grafisch und interaktiv visualisiert, um abschließende Usability-Tests durchführen zu können. Dabei wird eine identische Aufgabenstellung von Probanden mit dem alten und dem überarbeiteten Interface durchgeführt, um durch den Vergleich der Ergebnisse festzustellen, ob die Anpassungen die Usability von EB Guide erhöht haben.

## 1.5. Aufbau der Arbeit

Auf den folgenden Seiten werden zunächst die theoretischen Grundlagen erläutert, die benötigt werden um eine Usabilitystudie durchführen und verstehen zu können. Dazu zählt auch das Prinzip des Human-Centered Design Process, nach dessen Iterationsschritten die Kapitel dieser Arbeit grob gegliedert sind. Anschließend an die theoretischen Grundlagen werden Analysen an der bestehenden Software und am Arbeiterverhalten des Nutzer durchgeführt, um damit die existierenden Schwächen des Interfaces festzulegen und Verbesserungen erarbeiten zu können. Diese werden im darauffolgenden Kapitel, in Form eines Prototyps umgesetzt oder direkt implementiert um die abschließenden Usability Tests durchführen zu können. Abschließend werden die Ergebnisse dieser Tests ausgewertet und es wird ein kurzer Ausblick gegeben, welche Punkte in folgenden Iterationen des Design Process weiter verfolgt werden können.



## Kapitel 2.

### Theorie

Um die folgenden Analysen und Umsetzungen in dieser Arbeit verstehen zu können, ist es notwendig, einen Überblick über elementare Begriffe und Methoden der Usability zu haben. Aus diesem Grund werden diese im Folgenden erläutert, zusammen mit einigen Gestaltungsprinzipien, die es bei dem Entwurf eines Designs zu berücksichtigen gibt. Abschließend gibt es einen Überblick über den grundlegenden Aufbau und die Funktionen der Software EB GUIDE, für die im Rahmen dieser Arbeit die Usability untersucht und verbessert werden soll.

#### 2.1. Grafische Benutzeroberfläche

Der Begriff „Benutzerschnittstelle“ bezeichnet alle Komponenten eines interaktiven Systems, die dem Benutzer Interaktionsmöglichkeiten mit selbigem System bieten, um ein verfolgtes Ziel zu erreichen. Die grafische Benutzeroberfläche (GUI) bezeichnet hierbei den sichtbaren Anteil des Systems und damit nur einen Teil der gesamten Benutzerschnittstelle, zu der auch nicht sichtbare Teile wie z.B. die Funktionslogik gehören[\[Saro 16\]](#). Heutzutage sind die meisten Benutzeroberflächen auch grafische Benutzeroberflächen, mit denen in den häufigsten Fällen die Interaktion mit dem Nutzer über direkte Manipulation stattfindet[\[Niel 95\]](#).

#### 2.2. Ergonomie

Unter Ergonomie versteht man im Allgemeinen die „Lehre von der menschlichen Arbeit und die Erkenntnis ihrer Gesetzmäßigkeiten“ [\[Bull 94\]](#). Hierbei ist es wichtig zu verstehen, dass dabei der Fokus nicht ausschließlich auf einer technischen Komponente liegt, sondern das Zusammenspiel von Mensch, der zugeteilten Aufgabe und den verfügbaren Werkzeugen betrachtet wird[\[Saro 16\]](#). In Bezug auf Software bedeutet Ergonomie also konkret, diese gut handhabbar und benutzerorientiert zu gestalten.

## 2.3. Usability

Mit immer höherer Komplexität von Systemen und Anwendungen kam der Begriff und das Verlangen nach „Benutzerfreundlichkeit“ auf. Dieser Begriff suggeriert, dass lediglich die einfache Benutzung eines Systems ausschlaggebend ist, vernachlässigt hierbei jedoch die Notwendigkeit, den Nutzer beim Erreichen seiner Ziele passend zu unterstützen. Dies ist auch der Grund dafür, dass bald, statt auf „Benutzerfreundlichkeit“ auf „Gebrauchstauglichkeit“ (engl. Usability) geachtet wurde. Im Gegensatz zur Ergonomie handelt es sich bei Usability nicht um eine eigenständige, wissenschaftliche Disziplin, sondern um eine qualitative Anforderung an ein System[Saro 16]. Konkret spricht man bei einer Software-Anwendung von einer hohen Usability, wenn sie von der für sie bestimmten Zielgruppe effizient verwendet werden kann, also das verfolgte Ziel zufriedenstellend erreicht wird[Rich 16]. Hierfür ist es entscheidend, sich bewusst zu machen, dass ein technisches System oder Software immer Teil eines großen Handlungsablaufes ist und dazu dient, Schritte des Selbigen zu erledigen. Deshalb muss das System den Anforderungen dieses Ablaufes entsprechen und darf während der Entwicklung nicht getrennt davon betrachtet werden[Saro 16].

Usability wird üblicherweise gemessen, indem man Nutzern, die der Zielgruppe entsprechen einige vordefinierte Aufgaben mit dem zu messenden System abschließen lässt. In einigen Fällen kann es allerdings auch unnötig sein eine Aufgabe vorzudefinieren, da die Messung auch im alltäglichen Arbeitsablauf mit dem System stattfinden kann. In beiden Fällen ist es jedoch wichtig die Ergebnisse immer im Kontext des Nutzers und der absolvierten Aufgabe zu betrachten[Niel 95]. Wie bereits erwähnt, misst die Usability, wie zufriedenstellend ein Ziel erreicht werden kann. Deshalb ist es nicht möglich, die Usability für ein komplettes System zu messen, da diese immer kontextabhängig ist. Daraus ergibt sich auch die Notwendigkeit bei einer Usabilitystudie die Testpersonen identische, oder zumindest ähnliche Aufgaben durchführen zu lassen. Ein Textprogramm kann beispielsweise gut geeignet sein, um einen Brief zu tippen, jedoch völlig ungeeignet um größere Datenmengen zu verwalten. Es würde also für eine Aufgabenstellung eine hohe Usability und für eine andere eine niedrige, aufweisen. Würde man diese Ergebnisse kontextunabhängig vergleichen, könnte keine valide Aussage über die Usability des Systems getroffen werden.

Nach Nielsen kann die Usability an fünf Attributen gut gemessen werden, welche im Folgenden kurz erläutert werden[Niel 95].

**Erlernbarkeit** Das Attribut der Erlernbarkeit ist in vielerlei Hinsicht das fundamentalste von allen hier aufgeführten. Dies ist der Fall, da die erste Interaktion, die ein Nutzer mit einem System hat, meist das Erlernen dessen Funktionen beinhaltet. Gemessen wird dieses Attribut mithilfe von Nutzern, indem man die Zeit misst, die diese benötigen, um ein vorher festgelegtes Level an Kompetenz zu erreichen. Um einzustufen, ob dieses Level erreicht

wurde, kann beispielsweise überprüft werden, ob der Nutzer eine bestimmte Aufgabe ohne Probleme oder in einem gesetzten Zeitlimit bewältigen kann. Die Testpersonen sollten der letztendlichen Zielgruppe entsprechen und das System vorher noch nicht genutzt haben.

**Effizienz** Sobald die Lernkurve des Nutzers nicht mehr rapide ansteigt, wie es bei der anfänglichen Nutzung eines Systems meist der Fall ist, fokussiert er sich darauf effizient und produktiv mit dem System arbeiten zu können. Da effizientes Arbeiten nur ab einem gewissen Wissensgrad möglich ist, sollte bei der Messung dieses Attributes auf Testpersonen zurückgegriffen werden, die bereits Erfahrung mit dem System gemacht haben, gegebenenfalls sogar Experten sind. Die Einstufung, ab wann ein Nutzer als Experte gilt, kann entweder der Nutzer selbst treffen, oder man setzt voraus, dass Personen nach einem festgelegtem Benutzungszeitraum als Experten gelten. Eine typische Art Effizienz zu messen ist es, mit einer der genannten Möglichkeiten einen Nutzer als Experte einzustufen, Testpersonen zu finden, die diesen Kriterien entsprechen und die Zeit zu messen, die diese Personen benötigen um Testaufgaben abzuschließen.

**Wiedererkennungswert** Neben Experten und neuen Nutzern gibt es die Gruppe der gelegentlichen Nutzer eines Systems. Für diese Gruppe ist das Attribut der Wiedererkennung besonders wichtig, da sie im Gegensatz zu neuen Nutzern nicht lernen müssen, wie das Programm funktioniert, sondern sich lediglich an bereits Erlerntes erinnern müssen. Das Attribut der Erlernbarkeit wirkt also ebenfalls unterstützend, um einem System einen hohen Wiedererkennungswert zu geben. Aber prinzipiell gilt es zu beachten, dass das Neuerlernen eines Systems eben nicht mit dem Wiedereinstieg in selbiges gleichzusetzen ist. Diese beiden Attribute können sich also durchaus gegenseitig unterstützen, sind jedoch nicht austauschbar. Messbar ist der Wiedererkennungswert prinzipiell durch zwei Methoden. Eine davon ist die Durchführung eines Standardtests mit Nutzern, die einige Zeit nicht mit dem System interagiert haben. Hierbei gilt es die Zeit zu messen die sie benötigen um spezifizierte Aufgaben abzuschließen. Alternativ kann man einen Test durchführen, bei dem die Erinnerungen der Testpersonen geprüft werden. Hierbei werden die Nutzer nach dem Test gebeten, die Effekte verschiedener Interaktionen, das Aussehen von Icons und die Bezeichnung bestimmter Befehle zu benennen. Die Anzahl der korrekten Antworten, die man hierbei erhält, legt den Wiedererkennungswert des Systems fest. Obwohl diese zweite Testvariante leichter durchzuführen ist, besitzt sie die Schwäche, vor allem bei komplexeren Systemen nicht sonderlich aussagekräftig zu sein. Es wurde festgestellt, dass Nutzer sich hier nicht an genaue Bezeichnungen oder Aussehen der Befehle erinnern konnten, jedoch kein Problem hatten, diese in den Arbeitsabläufen zu finden oder zu nutzen. Deshalb ist der erste Testansatz empfehlenswerter, auch wenn dieser mehr Aufwand mit sich bringt.

**Fehler** Nutzer sollten während ihrer Interaktion mit dem System so wenig Fehler wie möglich machen. Unter einem Fehler versteht man in diesem Zusammenhang jede ausgeführte Aktion, durch die nicht das gewünschte Ergebnis erzielt wird. Die Fehlerrate eines Systems wird dementsprechend daran gemessen, wie viele Fehler einem Nutzer unterlaufen, während er eine Aufgabenstellung bearbeitet, und kann deshalb begleitend zu anderen Attributen untersucht werden.

**Zufriedenheit** Das letzte Attribut der Usability ist die Zufriedenheit, welche sich in diesem Fall darauf bezieht, wie angenehm es für den Nutzer ist, mit einem System zu interagieren. Zum einen kann dieses Attribut medizinisch, zum Beispiel mit Hilfe von EEGs, der Beobachtung der Pupillenerweiterung oder der Herzrate, untersucht werden. Da die Testpersonen jedoch meist ohnehin aufgeregt sind, sind solche zusätzlichen medizinischen Maßnahmen einer gewünschten entspannten Atmosphäre nicht zuträglich, abgesehen davon, dass der Aufwand für solche Untersuchungen sehr hoch ist. Eine andere Methode, die Zufriedenheit zu messen, ist es, die Nutzer einfach direkt zu fragen. Da man eine objektive Einschätzung der Zufriedenheit der Nutzer erhalten will, ist es hier natürlich nötig, mehrere Nutzer zu befragen, um aus den einzelnen subjektiven Meinungen eine objektive Meinung zu erhalten. Diese Befragung findet meist in Form eines Fragebogens statt; da diese recht komplexe Art der Evaluierung in dieser Arbeit jedoch keine Verwendung findet, werden die Vorgehensweisen an dieser Stelle nicht weiter erläutert.

Da es sich bei der in dieser Arbeit untersuchten Software EB GUIDE Studio - Näheres hierzu in Abschnitt 2.8 - um eine Anwendung handelt, mit der die meisten Nutzer über einen längeren Zeitraum auf hohem Niveau arbeiten, wird in dieser Usabilitystudie hauptsächlich das Attribut der Effizienz untersucht. Ergänzend dazu wird weiterführend dokumentiert, wie viele Fehler bei dem Nutzer durch die neuen Ergänzungen im Interface entstehen. Genaueres zu den Untersuchungen und Messungen folgt in Kapitel 5 dieser Arbeit.

## 2.4. User Experience

Entgegen einer häufigen Annahme bezeichnen Usability und User Experience (UX) nicht das Gleiche. Usability umfasst lediglich einen Teil der gesamten User Experience eines Systems[[Knig 19](#)]. UX bezieht sich nicht nur auf die reine Nutzungszeit eines Systems, sondern berücksichtigt auch den Zeitraum davor und danach, bezeichnet als Antizipierte Nutzung und Verarbeitung der Nutzungssituation. Usability ist hierbei, wie in Abb. [2.1](#) zu sehen, als wichtiger Faktor der User Experience in der aktiven Nutzungsphase zu betrachten, jedoch nicht mit dem Begriff gleichzusetzen [[Saro 16](#)]. Durch die zusätzliche Betrachtung der Effekte auf den Nutzer vor und nach der Nutzung, wie beispielsweise Erwartungen an das



Produkt und Akzeptanz dessen, entstehen hier auch Verbindungen zur Gestaltung der Benutzerschnittstelle und dem Produktdesign[Rich 16]. Zusammenfassend lässt sich festhalten, dass Usability zwar die funktionsbezogene Betrachtungsweise abdeckt, die User Experience als Ganzes jedoch auch emotionale Faktoren bezüglich des Designs und der Ästhetik berücksichtigt, um das Nutzungsvergnügen möglichst hoch zu halten. Zusätzlich ist eine gute User Experience notwendig, wenn ein Produkt auf dem Markt bestehen will. Sobald es mehr als ein Produkt zur Lösung der gleichen Aufgabenstellung gibt, wird das mit der besseren User Experience Verwendung finden[Knig 19].

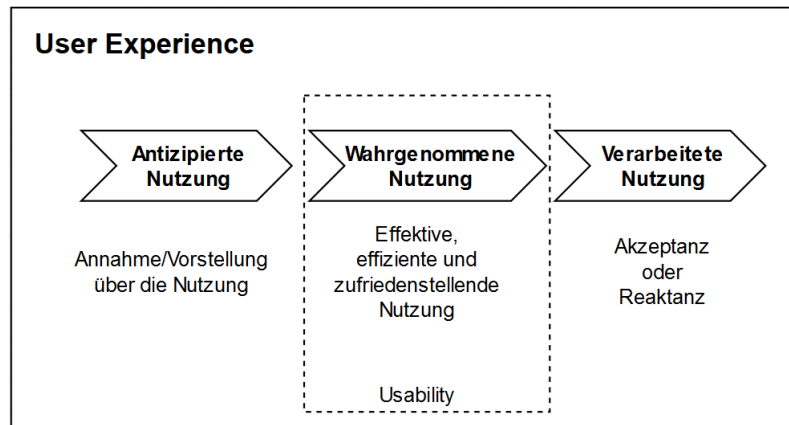


Abbildung 2.1.: Zusammenhang Usability und User Experience (nach [Saro 16])

## 2.5. Kognitive Last

Die Kognitive Last – auch als mentale Auslastung bekannt – bezeichnet im Allgemeinen die Differenz zwischen den Ressourcen (kognitiv, psychomotorisch oder wahrnehmend), die benötigt werden, um eine Aufgabe zu erfüllen, und den tatsächlich zur Verfügung stehenden Ressourcen. Die Größe der kognitiven Last beeinflusst also direkt die Leistung, die wir bei der Ausführung einer bestimmten Aufgabe erbringen können. Bei einer optimalen Auslastung erbringen Menschen maximale Leistung, ist sie hingegen zu klein oder zu groß, geht die Leistung zurück, was in einem Fall auf Unterforderung und damit einhergehende Langweile, im anderen Fall auf Überlastung und Ablenkung zurückzuführen ist[Miuc 19]. Kann der Nutzer sich nur auf eine fordernde Tätigkeit konzentrieren erreicht er das Maximum seiner Leistungsfähigkeit. Ist er jedoch gezwungen, sich auf mehrere Aktivitäten oder Umgebungsreize - beispielsweise Warnungen - gleichzeitig zu konzentrieren (Multitasking bzw. Mehrfachaufgabenperformanz), ist seine Performance in Bezug auf beide Tätigkeiten eingeschränkt. Beim Führen eines Fahrzeuges ist die Leistung des Fahrers beispielsweise maximal, wenn seine gesamte Aufmerksamkeit dem Fahren gilt. Muss er sich jedoch zeitgleich noch auf Warnungen des Autos konzentrieren, wird seine Leistung bei der Wahrnehmung

beider Einflüsse verringert. Auch ist beim Fahren die visuelle Modalität durch den ständigen Blick auf die Straße bereits sehr ausgelastet, weshalb zur Darstellung von Warnung auch aus Gründen der Auslastung auf Warnhinweise in Form von Videos oder Animationen verzichtet werden sollte, und eher auf die anderen Modalitäten des Menschen, wie Hören und Fühlen, ausgelagert werden. Dieses Phänomen lässt sich von der Situation der Benutzeroberflächen im Auto auf alle HMI's übertragen. Es ist also wichtig abzuwägen, wie man Warnungen in Benutzeroberflächen darstellt, um keine mentale Überlastung des Benutzers hervorzurufen.

## 2.6. Human - Centered Design Process

Human-Centered Design ist ein Ansatz zur Entwicklung interaktiver Systeme, welcher das Ziel verfolgt, möglichst nützliche Systeme zu entwickeln und diese gut benutzbar zu machen. Das wird vor allem durch das Fokussieren auf den Benutzer und dessen Bedürfnisse und durch die Anwendung von Ergonomie, Wissen und Techniken der Usability während des Entwicklungsprozesses erreicht[Elekd]. Der in Abb. 2.2 zu sehende User Experience Engineering Process beschreibt den Human Centered Design Process für Elektrobot und basiert auf der ISO Norm 9241 - 210. Das Model sieht Iterationen benutzerorientierter Aktivitäten vor, durch die fortlaufend der Erkenntnisstand der Entwickler erhöht wird, was bei dem ursprünglichen Ansatz eines Wasserfallmodells, bei dem stufenweise von Planung nach Durchführung vorgegangen wird, nicht der Fall war.

Die ISO 9241-210 Norm beschreibt das benutzerorientierte Vorgehen in Entwicklungsprojekten, und gibt Anregungen wie Usability-Engineering-Aktivitäten in Selbigen angewendet werden können. Allgemein definiert die Norm das Vorgehen so, dass die Gestaltung des Systems auf vorangehenden Analysen der Nutzer, deren Arbeitsaufgaben und Arbeitsabläufen aufbaut. Um aussagekräftige Analysen zu erhalten, werden die Nutzer während des Entwicklungsprozessen immer wieder aktiv mit einbezogen, was vor allem durch die fest eingeplanten Iterationen geschieht. Optimalerweise vereint das Gestaltungsteam fachübergreifende Kenntnisse und kann die in Abb. 2.2 benannten Rollen optimal besetzen[DIN ].

Der Human-Centered Design Process kann und sollte bei der Entwicklung jedes interaktiven Systems verwendet werden, welches eine Benutzeroberfläche aufweist, und wird deshalb auch in der Entwicklung dieser Bachelorarbeit angewendet werden. Aufgrund der Tatsache, dass bei dieser Arbeit einerseits eine zeitliche Einschränkung vorliegt, und andererseits die Bearbeitung von einer Person und nicht von einem Team durchgeführt wird, werden die Iterationen und die einzelnen Schritte in einer abgeschwächten Form bearbeitet werden. Genauere Erläuterungen dazu finden sich in den entsprechenden Abschnitten dieser Arbeit.

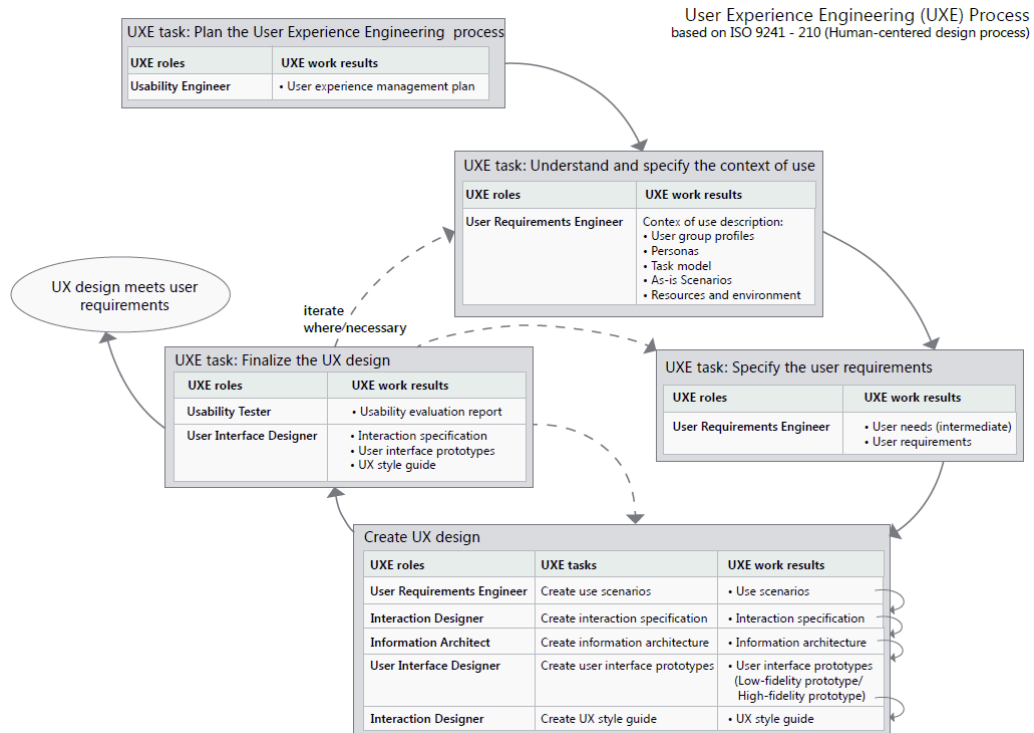


Abbildung 2.2.: Human - Centered Design - Process bei Elektrobit

## 2.7. Gestaltprinzipien der Usability

Durch die Interaktion mit einem Produkt oder einer Anwendung entstehen bei Nutzern immer Erfahrungen. Ob diese Erfahrungen positiv oder negativ behaftet sind, wird maßgeblich dadurch beeinflusst, ob das Design der Anwendung den Nutzer unterstützt oder eher behindert. Bei der Interaktion mit einem Produkt muss der Nutzer immer herausfinden, wie er mit ihm umgehen soll, welche Funktionen es bietet oder was überhaupt der Zweck des Produktes ist. Diese Eigenschaft wird als Discoverability oder Erkennbarkeit bezeichnet und ist für eine gute User Experience möglichst hoch zu halten. Discoverability resultiert aus denen im Folgenden erläuterten Gestaltprinzipien der Usability, weshalb diese bei jeden Entwurf eines Systems berücksichtigt und einbezogen werden sollten[[Norm 16](#)].

### Konsistenz

Konsistenz im Design wirkt sich insofern positiv auf die Usability eines Systems aus, dass die Benutzbarkeit steigt, wenn gleiche Funktionen in einem System auch auf identische Art und Weise dargestellt werden. Dadurch wird es dem Nutzer ermöglicht, bereits Gelerntes in einem neuen Kontext anzuwenden, ohne effektiv darüber nachdenken zu müssen. Dadurch kann Neues schneller erlernt werden und der Nutzer kann sich auf relevante Aspekte einer

Aufgabe konzentrieren. Konsistenz in Systemen lässt sich in vier Kategorien unterteilen, die im Folgenden jeweils kurz erläutert werden.

**Ästhetische Konsistenz** Ästhetische Konsistenz bezieht sich auf Konsistenz in Stil und Aussehen. Dadurch entsteht ein hoher Wiedererkennungswert und es wird Zugehörigkeit signalisiert. Das einfachste Beispiel hierfür ist ein Firmenlogo, welches immer konsistent in Farbgebung und Schrift auftaucht.

**Funktionale Konsistenz** Funktionale Konsistenz bezieht sich auf die Konsistenz von Bedeutung und Aktion, beispielsweise zeigt eine Ampel immer Orange an, bevor sie rot wird. Der Nutzer kann also sicher mit einer immer gleichen darauffolgenden Aktion auf den aktuellen Zustand rechnen. Usability und Lernfähigkeit werden durch funktionale Konsistenz erhöht, indem diese bereits verinnerlichten Abläufe einfach auf neue Situationen übertragen werden können. So wird beispielsweise ein einheitlicher Playbutton für alle Geräte, von Kassettenrekordern bis zu Streamingdiensten, verwendet. Die Nutzung bereits bekannter Symbole ermöglicht dem Nutzer also eine intuitive Interaktion und macht es möglich die Aufmerksamkeit auf unbekannte Aspekte zu richten, die tatsächlich noch neu erlernt werden müssen.

**Interne Konsistenz** Interne Konsistenz bezieht sich auf Konsistenz mit anderen Elementen im System, beispielsweise sind Wegweiser in einem Park konsistent zueinander. Dies erzeugt ein Vertrauensgefühl bei den Nutzern, vermittelt ein durchdachtes Designkonzept und erweckt nicht den Eindruck, dass alle Komponenten zufällig zusammengestellt wurden.

**Externe Konsistenz** Externe Konsistenz bezieht sich auf Konsistenz mit anderen Elementen in der Umgebung. Die Vorteile der internen Konsistenz werden hierbei systemübergreifend erweitert. Das Erreichen externer Konsistenz gestaltet sich jedoch auch schwieriger, da unabhängige Systeme selten exakt gleiche Designstandards haben.

Nicht alle dieser Konsistenzstandards können oder sollten in allen Fällen angewendet werden. Es lässt sich jedoch festhalten, dass Elemente in einer logischen Gruppe immer ästhetisch und funktional konsistent sein sollten. Insgesamt sollte ästhetische und funktionale Konsistenz, wenn möglich, immer berücksichtigt werden. Durch ästhetische Konsistenz kann bei der Einführung einmaliger Identitäten Wiedererkennung gewährleistet werden und funktionale Konsistenz kann aus den bereits genannten Gründen die Usability eines Systems erhöhen. Abschließend sollten Systeme immer interne Konsistenz aufweisen. Existieren bereits Designstandards, gilt es diese zu analysieren.[\[Lidw 10\]](#)

## Sichtbarkeit

Das Prinzip der Sichtbarkeit besagt, dass Systeme benutzerfreundlicher sind, wenn der aktuelle Status des Systems mögliche Aktionen und deren Auswirkungen für den Nutzer deutlich erkennbar sind. Es beruht auf der Erkenntnis, dass Menschen schneller und besser Lösungswege finden, wenn sie aus einer Reihe von Optionen auswählen können, anstatt sich selbstständig an alle Möglichkeiten aus dem Stegreif erinnern zu müssen[Lidw 10]. Das macht es zu dem wichtigsten Prinzip für hohe Usability in komplexen Systemen, jedoch auch zu dem, welches am meisten verletzt wird[Norm 16]. Häufig wird versucht, alle möglichen Optionen, die ein System bietet, sichtbar zu machen, was vor allem in komplexeren Systemen dazu führt, dass die relevanten Optionen durch Informationsüberladung auf Seiten des Nutzers schwerer zu erreichen sind. Anstatt die Usability zu erhöhen, wird also genau das Gegenteil erzielt. Lösungen für diese Problemstellung sind beispielsweise eine hierarchische Anordnung der Elemente oder Kontextsensitivität des Systems. Bei der Hierarchischen Anordnung werden Funktionen und Informationen in logische Kategorien unterteilt und in übergeordneten Menüs versteckt, welche bei Bedarf ausgeklappt werden können. Ein Beispiel hierfür ist ein Dropdownmenü, wie man es aus vielen Programmen wie beispielsweise Microsoft Word (Abb. 2.2) kennt. In einem kontextsensitiven System werden Aktionsmöglichkeiten und Informationen, je nach aktuellem Status des Systems, versteckt oder angezeigt. Beispielsweise bekommt man in einer Modellierungssoftware mit unterschiedlichen Elementen immer nur die Eigenschaften angezeigt, die für das aktuell ausgewählte Element relevant sind, und nicht alle im System existierenden[Lidw 10].

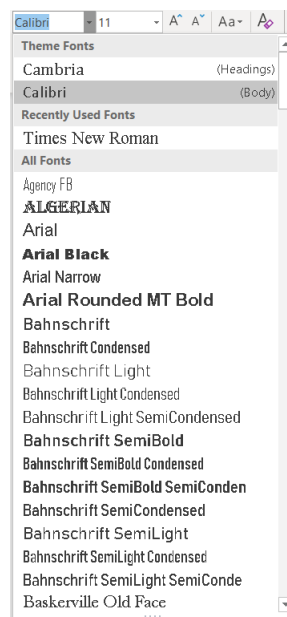


Abbildung 2.3.: Dropdownmenü in Word

## Affordanz

Affordanz bezeichnet die Möglichkeiten, mit einem Objekt zu interagieren. Beispielsweise kann eine Checkbox an- und wieder abgewählt oder ein Schieberegler nach oben und unten geschoben werden. Beispiele hierfür lassen sich direkt in EB GUIDE finden und sind in Abb. 2.4 zu sehen. Die sichtbare Affordanz bezeichnet die Eigenschaft, dass einem Objekt die Interaktionsmöglichkeiten bereits angesehen werden können, ohne mit ihm interagiert zu haben. Sichtbare Affordanz ist vor allem in der User Interface Gestaltung wichtig, weil praktisch betrachtet alle Pixel auf einem Bildschirm anklickbar sind, jedoch in den meisten Fällen keine Aktion durch das Klicken ausgelöst wird. Deshalb ist es wichtig, dem Nutzer durch das Aussehen der Elemente zu vermitteln, ob diese, wenn sie geklickt werden, eine Aktion auslösen, um dem Nutzer wahlloses Klicken durch das Interface zu ersparen, bis ein interaktives Objekt gefunden wird. Solche Probleme können visuell gelöst werden, indem man Objekte im User Interface wie Objekte in der echten Welt aussehen lässt, also beispielsweise einen Button dreidimensional gestaltet. Alternativ kann man alle anklickbaren Objekte etwas anders gestalten als den Rest der Objekte im Interface, um dem Nutzer zu vermitteln, dass hier eine Interaktion stattfinden kann [Knig 19].

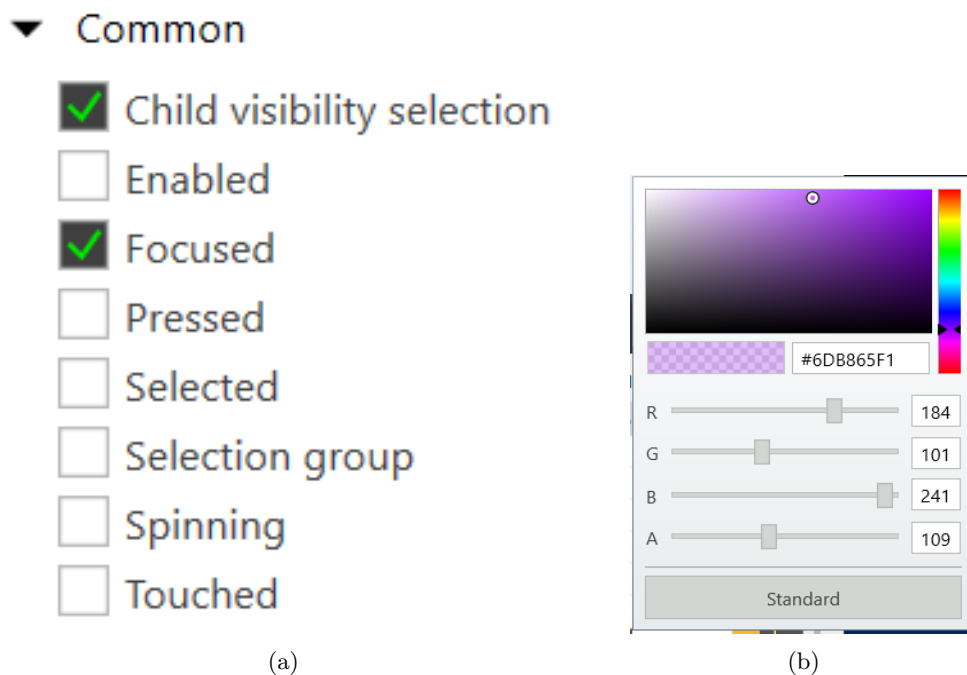


Abbildung 2.4.: Checkbox und Schieberegler in EB GUIDE

## Rückmeldung

Eine der grundlegendsten Richtlinien um die Usability eines Systems zu erhöhen, ist es, dem Nutzer immer eine Rückmeldung auf seine Aktionen zu geben. Das bedeutet, dem Nutzer

immer den aktuellen Systemstatus anzuzeigen und wie seine Aktion vom System interpretiert wurde. Eine Rückmeldung ist vor allem auch dann wichtig, wenn die gewünschte Aktion nicht erfolgreich vom System ausgeführt wurde. Durch fehlende Rückmeldungen kommt Misstrauen beim Nutzer auf, weil ihm nicht vermittelt wird, ob auf seine Aktionen auch eine Reaktion des Systems erfolgt[Knig 19]. Dieses Problem tritt beispielsweise auf, wenn ein System lange braucht, um eine Eingabe zu verarbeiten und es versäumt, dies dem Nutzer mitzuteilen. So ein Verhalten könnte zur fälschlichen Annahme führen, das System wäre defekt, oder dazu, dass der Nutzer beginnt, neue Interaktionsmöglichkeiten anzuklicken. Nielsen hat hierfür konkrete Zeitspannen definiert, ab denen der Nutzer eine Rückmeldung erhalten muss oder in denen noch darauf verzichtet werden kann.[Jako 93] Reagiert das System innerhalb von 0.1 Sekunden, nimmt der Nutzer dies als sofortiges Feedback wahr und keine gesonderte Meldung ist nötig. Bewegt sich die Reaktionszeit zwischen 0.1 und 1.0 Sekunden, ist für gewöhnlich auch keine besondere Rückmeldung vom System nötig, auch wenn der Nutzer hier bereits nicht mehr das Gefühl hat, direkt mit den Daten zu interagieren, da eine kurze Verzögerung stattfindet. Sobald die Wartezeit auf Seiten des Nutzers jedoch eine Sekunde überschreitet, sollte sich der Cursor in eine Sanduhr oder Ähnliches verwandeln und damit die Rückmeldung geben, dass das System beschäftigt ist. Sollte die systemseitige Verarbeitung der Eingabe länger als 10 Sekunden dauern, ist es ratsam die Ladeanzeige durch eine konkrete Fortschrittsleiste zu ersetzen.

## Mapping

Mapping bezeichnet die Beziehung zwischen Bedienelementen und den Effekten, welche durch deren Aktivierung ausgelöst wird. Wenn der Effekt, den eine Nutzerinteraktion nach sich zieht, den Erwartungen des Nutzers entspricht, handelt es sich um gutes Mapping. Dies ist beispielsweise bei einem elektronischen Fensterheber der Fall. Wird hier der Hebel nach oben bewegt, hebt sich das Fenster, bewegt man den Hebel nach unten, senkt es sich. Gutes Mapping wird zum Großteil durch die Ähnlichkeit, das Verhalten oder der Bedeutung innerhalb eines Layouts erreicht. Entspricht beispielsweise das Layout von Herdplattenreglern der Anordnung der Platten, wird gutes Mapping durch das Layout erzeugt und auch der bereits beschriebene Fensterheber arbeitet mit diesem Verhalten. Die Tatsache, dass ein Notfallknopf rot eingefärbt wird, ist darauf zurückzuführen, dass die meisten Menschen die Farbe Rot mit Gefahr oder dem Stopplicht einer Ampel assoziieren. In jedem dieser Fälle macht die Ähnlichkeit es möglich, den Effekt der Handlung vorherzusehen und vereinfacht dadurch die Bedienung für den Nutzer. Aktionsmöglichkeiten müssen so platziert werden, dass ihre Position und ihr Verhalten dem Layout und dem Verhalten der Anwendung angepasst sind. Außerdem sollte es vermieden werden, durch eine identische Aktion verschiedene Reaktionen auszulösen[Lidw 10].

## Einschränkungen

Einschränkungen in einem System limitieren die Interaktionsmöglichkeiten für den Nutzer. Wird beispielsweise ein Button ausgegraut, der im aktuellen Kontext ohnehin keine Aktion ausführen würde, wird der Nutzer rein optisch daran gehindert, eine nicht zielführende Aktion auszuführen. Durchdachte Einschränkungen dieser Art machen ein Design einfacher nutzbar und reduziert deutlich die Wahrscheinlichkeit von Fehlschlägen während der Systeminteraktion [Lidw 10].

**Physische Einschränkungen** Physische Einschränkungen limitieren den Bereich, in dem Aktionen ausgeführt werden können, indem sie Eingaben des Nutzers umwandeln oder umleiten. Eine Art der Einschränkungen ist das Konvertieren der Eingabe in lineare oder kurvenförmige Bewegung, wie es beispielsweise bei der Interaktion mit einer Scrollbar der Fall ist. Mithilfe von Achsen können wirkende Kräfte in Rotationsbewegungen umgewandelt werden, was eine Kontrolloberfläche mit unendlicher Größe auf einem kleinen Feld erzeugt. Dies ist am Beispiel einer Mauskugel einer früheren Computermaus gut zu erkennen, da deren dreidimensionale Drehbewegungen auch auf zwei Dimensionen übertragen werden. Die letzte Form der Einschränkung passiert über Barrieren, welche die Eingabe verlangsamen, komplett ausbremsen oder umleiten. Die Einfassung eines Computerbildschirms beschränkt beispielsweise physisch die Interaktionsfläche für den Nutzer. Allgemein sind Physische Einschränkungen nützlich um die Anzahl fehlerhafter Eingaben zu vermeiden, oder manche Eingaben erst gar nicht zu ermöglichen[Norm 16].

**Psychologische Einschränkungen** Psychologische Einschränkungen limitieren die Anzahl möglicher Aktionen durch Nutzung des Wissens über das Verhalten und die Denkweise der Nutzer. Dies passiert durch den Einsatz von Symbolen, das Nutzen bekannter Konventionen oder durch das bereits erwähnte Mapping. Symbole sind sinnvoll um Dinge zu benennen, zu erklären oder auch um Warnungen visuell, auditiv oder fühlbar darzustellen. Mit Konventionen macht man sich bekannte Interaktionsmöglichkeiten zunutze, weshalb sie sich gut eignen Systeme sowohl konsistent, als auch leicht benutzbar zu machen. Mappings sind, aus bereits erwähnten Gründen nützlich, um dem Benutzer zu vermitteln, welche Aktionen aufgrund der Sichtbarkeit, Position oder dem Aussehen der Elemente möglich sind[Norm 16].

Zusammenfassend sollten Einschränkungen im Allgemeinen verwendet werden, um die Benutzbarkeit eines Systems zu vereinfachen und die Anzahl von Fehlern zu minimieren. Physische Einschränkungen dienen hierbei eher der Vermeidung ungewollter Eingaben oder gefährlicher Aktionen; Psychologische Einschränkungen sollen das Design eines Systems für den Nutzer klarer und intuitiver gestalten.



## Vorteile für den Nutzer

Durch das Befolgen der soeben erläuterten Gestaltprinzipien bei dem Entwurf eines User Interfaces, ist es möglich, sich die Art und Weise, wie Nutzer visuelle Reize verarbeiten, zunutze zu machen, um die Benutzerfreundlichkeit des Systems zu erhöhen. Das passiert vor allem dadurch, dass man die kognitive Last des Nutzers verringert, während er sich mit dem Interface auseinandersetzt, indem man bereits bekannten Gestaltprinzipien nutzt. Das bedeutet, dass der Nutzer seine Energie nicht darauf verschwenden muss, darüber nachzudenken, wie mit den Bestandteilen des Interfaces interagiert werden kann, oder was deren Funktionen sein könnten, sondern die Aktionen intuitiv stattfinden können[[Knig 19](#)].

## 2.8. EB Guide Studio

Wie bereits in Abschnitt 1.2 erwähnt, dient EB GUIDE der Entwicklung multimodaler HMIs. Um nicht nur das Design, sondern auch das Verhalten von User Interfaces bestimmen zu können, und eine Auslieferung auf das Zielsystem zu ermöglichen besteht die Produktlinie EB GUIDE aus den verschiedenen, in Abb. 2.5 aufgeführten, Komponenten. Hierbei wird zwischen jenen für das Graphical User Interface (GUI) und für die Sprachsteuerung unterschieden. Da die Sprachkomponenten jedoch für diese Arbeit nicht weiter relevant sind, werden diese in den folgenden Kapiteln auch nicht genauer erläutert. Innerhalb des GUI Bereiches bildet EB GUIDE Studio das tatsächliche Modellierungstool, mit dem das Verhalten und Aussehen der Benutzeroberfläche definiert wird. Für das entwickelte Modell stellt das EB GUIDE Target Framework auf dem Zielsystem die Laufzeitumgebung bereit.[[Elekb](#)]

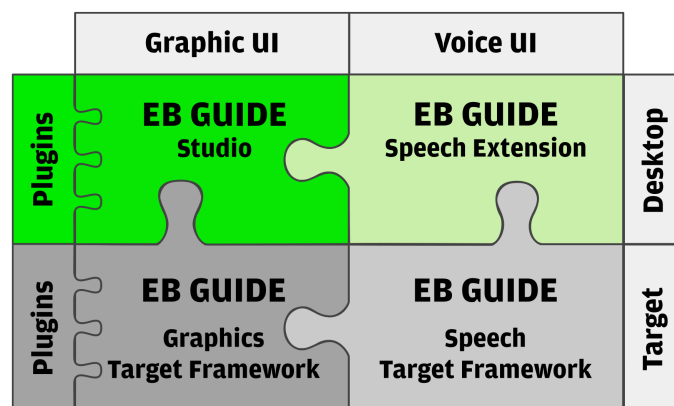


Abbildung 2.5.: Aufbau EB GUIDE

EB GUIDE Studio ist das Interface von EB GUIDE mit dem nach dem What-You-See-Is-What-You-Get (WYSIWYG) Prinzip User Interfaces modelliert werden. Durch das WYSIWYG Prinzip ist es während des Modellierens einer View bereits möglich das Endergebnis des Designs zu sehen. Eine View bezeichnet in diesem Kontext einen projektspezifische Ansicht innerhalb des Interfaces, welche aus Widgets besteht, welche die Interaktionsschnittstelle zwischen Nutzer und System darstellen. Das Verhalten des Interfaces hingegen wird mithilfe einer Zustandsmaschine, der sogenannten State Machine modelliert, die auf dem UML-Prinzip aufbaut. Eine State Machine ist deterministischer, endlicher Zustandsautomat, welcher das dynamische Verhalten des Systems beschreibt. Innerhalb von EB GUIDE besteht eine State Machine aus einer beliebigen Anzahl von Zuständen, wobei jeder Zustand einer View zugeordnet ist. Die Trennung der Logik und des Designs wird in EB GUIDE Studio grafisch durch zwei unterschiedliche Arbeitsoberflächen gestaltet, in welchen den Modellierern jeweils die entsprechenden Tools und Elemente zur Verfügung stehen.

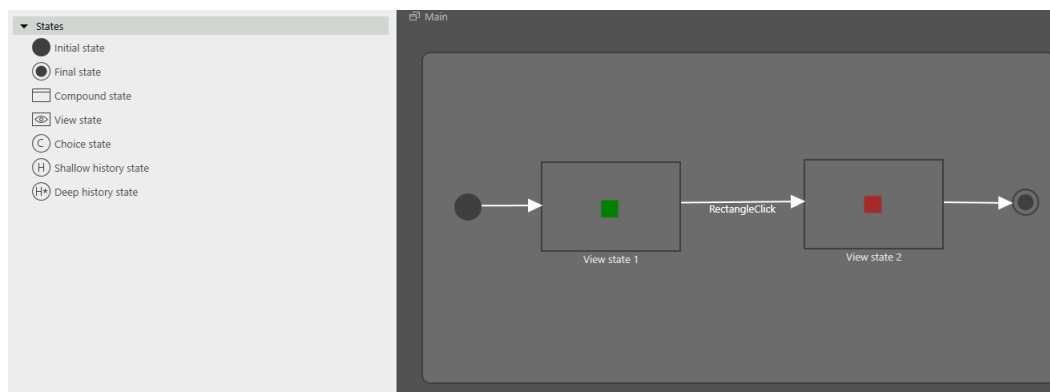


Abbildung 2.6.: EB Guide Studio Statemachine

In Abb. 2.6 ist die State Machine für ein simples Beispiel zu sehen. Links neben der Arbeitsfläche befindet sich eine Toolbox, mit deren Inhalt per Drag and Drop auf der Arbeitsfläche die benötigte Logik definiert wird. Wie bei UML-Diagrammen gibt es einen Initial State der den Startpunkt angibt und einen Final State, der die State Machine beendet. Die ebenfalls zu sehenden View States stehen für einen Screen im Endprodukt, dementsprechend wird in den View States auch das Aussehen der Interfaces definiert. Die Verbindungen zwischen den States werden als Transitionen bezeichnet und durch Events ausgelöst. Events stellen hierbei beliebige Ereignisse dar die durch Elemente in der View ausgelöst werden können. Der Auslöser für das Event wird mithilfe einer eigens für EB GUIDE entwickelten Skriptsprache als Trigger für dieses gesetzt. Die häufigsten Auslöser sind Benutzeraktionen mit Widgets, die in Abb. 2.7 zu sehen sind. Widgets sind Elemente, mit denen das Aussehen des Interfaces im sogenannten View Editor bestimmt wird und die sich in Basis- und 3D-Widgets einteilen lassen. Alle Basiswidgets verfügen über Basiseigenschaften wie Höhe, Breite und Farbe, sowie über spezifische Eigenschaften wie zum Beispiel „Touch-Released“ bei einem

Button.[[Elek](#)] Beispielsweise wird das Event „RectangleClick“ durch das Klicken auf das grüne Rechteck in View State 1 ausgelöst. Dies kann in Abb. 2.6 an der Transition zwischen View State 1 und View State 2 nachvollzogen werden. Ist die Benutzeraktion abgeschlossen, findet der Übergang von einem View State in den anderen statt und der Nutzer sieht nun statt dem grünen ein rotes Rechteck. Damit diese Aktion erfolgreich ausgeführt wird, muss das grüne Rechteck die Eigenschaft „Touch-Released“ zugewiesen bekommen. Über die EB GUIDE Skriptsprache wird ihm dann mitgeteilt, das Event RectangleClick zu feuern sobald das grüne Rechteck berührt wurde. Da sich dieses Event in der State Machine an der Transition zwischen den beiden States befindet, wird nun durch einen Klick auf das grüne Rechteck ein Bildschirmwechsel zwischen den beiden View States ausgelöst.

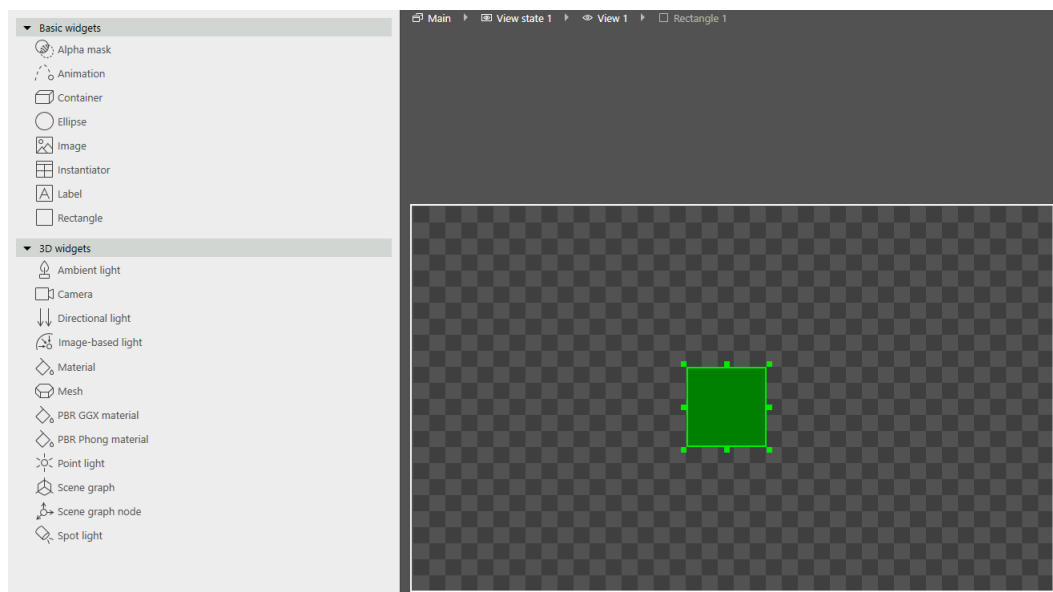


Abbildung 2.7.: EB Guide VIEW



## Kapitel 3.

### Analysen

Wie in Abschnitt 2.3 bereits erwähnt, liegt eine hohe Usability genau dann vor, wenn ein System von der für es bestimmten Zielgruppe effizient verwendet werden kann.[\[Rich 16\]](#) Um dies zu gewährleisten ist es vorab nötig diese Gruppe zu kennen, zu analysieren und eventuelle Schwierigkeiten in der Benutzung des Systems aufzudecken. Die Erkennung dieser Schwierigkeiten muss aus Gründen, die ebenfalls in Abschnitt 2.3 erläutert wurden, immer in Relation zu der aktuell ausgeführten Aufgabe geschehen. Aus diesem Grund werden im Folgenden zuerst die Ausgangssituation beschrieben, in der die Analysen durchgeführt wurden. Anschließend folgt ein Überblick über die Analysemethoden bevor abschließend die Ergebnisse dargestellt werden.

#### 3.1. Ausgangssituation

Im ersten Schritt des Human-Centered Design Process, zu sehen in Abb. 2.2 des vorherigen Kapitels, gilt es den Benutzungskontext zu verstehen. Im Laufe dieses Kapitels wird die Rolle des User Requirements Engineer erfüllt, welcher für gewöhnlich für die Dokumentation, Validierung und Verwaltung der Anforderungen an ein Produkt zuständig ist. Aus all den möglichen Aufgaben die in dessen Rolle in diesem Iterationsschritt erfüllt werden sollen, wird in dieser Arbeit auf die Nutzergruppe, deren tägliche Aufgaben und die Arbeitsumgebung genauer eingegangen.

**Arbeitsumgebung** Das Arbeitsumfeld bildet das in Abschnitt 2.8 bereits erwähnte EB GUIDE 6. Es ist hierbei situationsabhängig ob die Modellierer mit dem Speech-Anteil von EB GUIDE arbeiten oder nicht. Für diese Arbeit werden die Interaktionen mit dem Speech-Anteil ignoriert und sich nur auf die Usability von EB GUIDE Studio konzentriert. Auch auf beide Target Frameworks wird im Folgenden nicht mehr weiter eingegangen, da die Nutzerinteraktion mit EB GUIDE über die Schnittstelle EB GUIDE Studio stattfindet.

**Nutzergruppe** Die Zielgruppe für die Analysen im Rahmen dieser Arbeit deckt sich mit der Nutzergruppe von EB GUIDE. Im Rahmen dieser Arbeit wurden nur Personen beobachtet und analysiert, die bei Elektrobit beschäftigt sind. Aus den getätigten Beobachtungen geht hervor, dass die Nutzer teilweise sehr routiniert und auch täglich mit der Software arbeiten, andere benötigen diese hingegen nur sporadisch in ihrer Arbeit. Es werden also Experten und gelegentliche Nutzer in dieser Arbeit untersucht, Nutzer die noch nie mit der Software gearbeitet haben werden nicht beachtet. Wie in Abschnitt 2.3 dargestellt ergibt sich hieraus die Möglichkeit das System auf Effizienz, Wiedererkennungswert, Fehler und Zufriedenheit zu testen, welche jedoch im Folgenden noch eingegrenzt werden.

**Arbeitsaufgaben** Der Großteil der Modellierer arbeitet in laufenden Projekten von Elektrobit und modelliert dort mithilfe von EB GUIDE 6 Human Machine Interfaces für die Automobilbranche. Ein anderer Teil der Zielgruppe arbeiten an Kundendemonstrationen mit deren Hilfe dargestellt wird, was mit der aktuellen Version von EB GUIDE 6 modelliert werden kann. Beide Gruppen setzen bei ihrer Arbeit Spezifikationen um, die nach den Wünschen des Kunden direkt von diesem oder von Designfirmen erstellt werden. Diese Spezifikationen bestehen meist aus einem schriftlichen Teil, der die Logik beschreibt nach der das Interface arbeiten muss, sowie aus einem grafischen Teil der die Anordnung von Icons und Texten darstellt.

### 3.2. Vorgehensweise

Nachdem nun der Benutzungskontext analysiert wurde, gilt es im zweiten Schritt des Prozesses die Benutzeranforderungen zu spezifizieren. Diese Benutzeranforderungen sind testbar, eindeutig, konsistent und beinhalten die Definition identifizierter Bedürfnisse der Nutzer. Zu unterscheiden sind qualitative und quantitative Benutzeranforderungen, wobei beide eine Basis für das Design des interaktiven Systems bieten und durch Evaluierung des Systems verifiziert werden können. Qualitative Anforderungen beziehen sich auf die Art und Weise wie das System genutzt wird, um das Ziel zu erreichen, quantitative Anforderungen hingegen setzen messbare Ziele für die Usability und User Experience.[\[Eleke\]](#)

Um repräsentative Benutzeranforderungen zu erhalten, ist es notwendig, dass die Bedürfnisse der Nutzer, aus denen die Anforderungen gebildet werden, tatsächlich auch den Bedürfnissen der Zielgruppe entsprechen. Diese Bedürfnisse erhält man beispielsweise durch Interviews oder Beobachtungen innerhalb der Zielgruppe. Diese Vorgehensweisen werden im Rahmen dieser Arbeit kombiniert angewandt, wodurch die Rollen Interviewer und Beobachter auf eine Person reduziert werden. Die Nutzer werden bei ihrer täglichen Arbeit beobachtet und währenddessen aufgefordert ihre aktuellen Arbeitsschritte zu erklären, wodurch

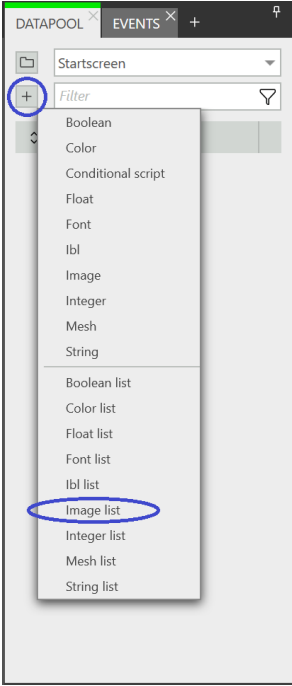
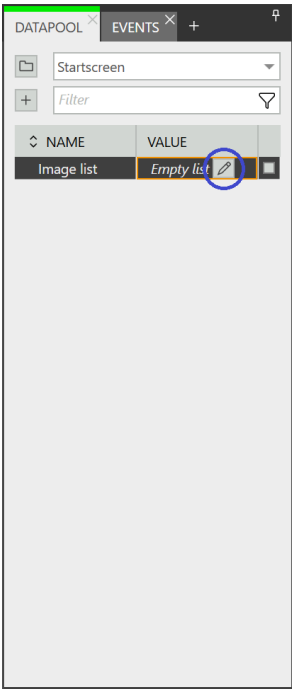
sie bei Bedarf auch automatisch Kritik am Interface äußern können und dem Beobachter/-Interviewer begleitendes befragen ermöglicht wird.

### 3.3. Ergebnisse

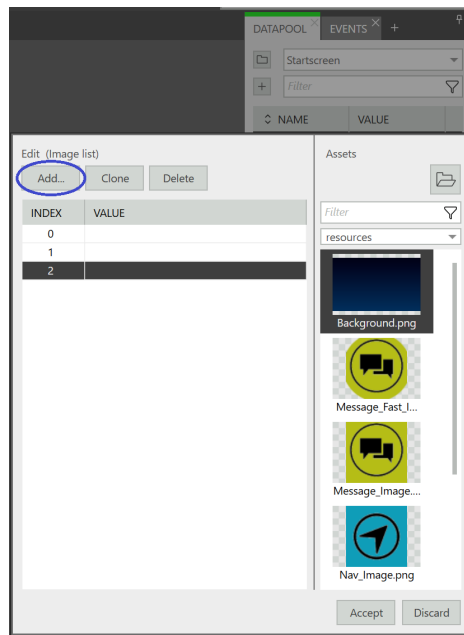
Die soeben beschriebenen Vorgehensweise wird bei der Beobachtung/Befragung von fünf Modellierern angewandt. Da hier immer der gesamte Ablauf einer typischen Arbeitsaufgabe beobachtet wird, variiert die Dauer der Sitzungen. Im Durchschnitt werden die Modellierer jedoch 2 bis 3 Stunden bei ihrer täglichen Arbeit beobachtet und währenddessen zu - für den Beobachter unklaren - Abläufen befragt. Bei der Durchführung der Beobachtungen fallen einige Dinge auf, die die Modellierer bei ihrer Arbeit behindern oder erheblich verlangsamen und welche alle in den folgenden Abschnitten aufgeführt werden. Teilweise fällt den Nutzern das selbst auf und sie weisen den Beobachter darauf hin, teilweise haben sie sich bereits so an die Arbeitsschritte gewöhnt, dass die Behinderung nur einem Außenstehenden auffällt, den Nutzern selbst jedoch nicht mehr.

Im Folgenden werden zuerst Beobachtungen allgemein erläutert, bevor die Bedürfnisse der Nutzer formuliert werden, damit es abschließend möglich ist die qualitativen Benutzeranforderungen zu definieren. Die Quantitativen Anforderungen werden in Kapitel 5, im Rahmen des tatsächlichen Usability Tests aufgeführt.

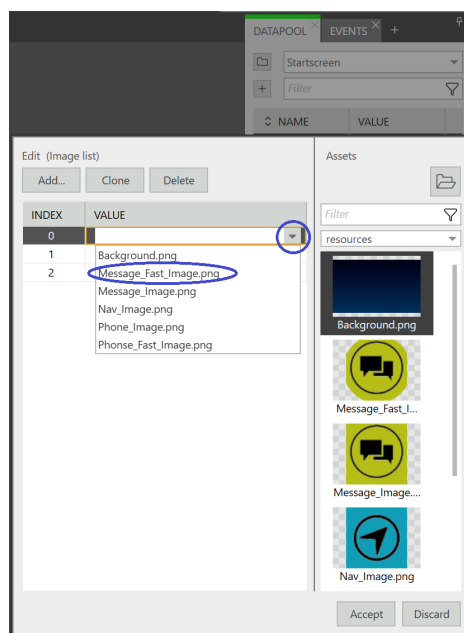
**Image List** Bei einer Image List handelt es sich um ein Datapool Item. Datapool Items sind Modelemente, die benutzt werden können, um Daten von der Applikation an das Interface zu senden, oder umgekehrt. Außerdem können damit Daten gespeichert werden, die entweder nur von Seiten des Interfaces genutzt werden, oder nur von der Applikationsseite[[Elek](#)]. In Tabelle 3.1 kann das befüllen eines solchen Datapool items am Beispiel einer Image List nachvollzogen werden.

Arbeitsschritt	Beschreibung
 <p>The screenshot shows the 'DATAPOOL' tab selected. Below the 'Startscreen' dropdown, there is a '+' button in a blue circle. A dropdown menu is open, listing various data types: Boolean, Color, Conditional script, Float, Font, Ibl, Image, Integer, Mesh, String, Boolean list, Color list, Float list, Font list, Ibl list, Image list (highlighted with a blue circle), Integer list, Mesh list, and String list.</p>	<p>Anfangs besteht für den Nutzer die Möglichkeit aus einer Vielzahl von Datapool Items zu wählen. Hierfür werden bereits zwei Klicks benötigt, dieser Vorgang ist jedoch einmalig und kann auch nicht reduziert werden</p>
 <p>The screenshot shows the 'DATAPOOL' tab with a table view. The table has two columns: 'NAME' and 'VALUE'. The first row is 'Image list' with the value 'Empty list'. A blue circle highlights the 'Empty list' text, which has a small pencil icon next to it.</p>	<p>Hier initialisiert der Nutzer das Befüllen der Liste durch einen Klick auf den „Stift“ Button, wodurch sich das Pop Up in der Folgenden Grafik öffnet.</p>





Die hier zu sehenden Platzhalter für Index und Value müssen einzeln durch einen Klick auf den „Add..“ Button hinzugefügt, ...



... und danach individuell mit dem gewünschten Images befüllt werden.

Tabelle 3.1.: Usability - Schwäche Image List

Für die minimale Auswahl in diesem Beispiel ist der Aufwand noch überschaubar, es gilt jedoch zu bedenken, dass in tatsächlichen Projekten Image Lists mit mindestens 100 Images erstellt werden. Das bedeutet für den Nutzer mehrere Stunden Arbeit, die keinen wirklichen Mehrwert liefern und den Joy of Use deutlich mindern. Joy of Use bezeichnet im Allgemeinen eine Erweiterung der Usability dar und beschreibt die positiven Erfahrungen eines Nutzers.

Diese werden vor allem durch das effiziente Erreichen der gesteckten Ziele erreicht, was bei einer sich immer wieder wiederholenden Tätigkeit nicht gegeben ist.

**Resultierende Benutzeranforderung** Nutzer, die eine Datapool Liste anlegen, müssen die Möglichkeit haben mehrere Images gleichzeitig zu dieser Liste hinzuzufügen.

**Navigation** Soll ein neues Element in der View hinzugefügt werden, wird der Widget Tree in der Navigation, in Abb. 3.1 zu sehen, nicht automatisch ausgeklappt. Durch Beobachtung kann festgestellt werden, dass die Nutzer, nachdem sie ein neues Element eingefügt haben, dieses sofort umbenennen. Für diesen Vorgang muss das Element im Widget Tree erst gesucht werden, wobei sich je nach angelegten Ebenen im Projekt die Anzahl der hier benötigten Klicks exponentiell steigert. Sollte der Nutzer in Abb. 3.1 beispielsweise gerade das NavImage eingefügt haben, müssen erst vier Ebenen ausgeklappt werden, bevor das Umbenennen möglich ist. In größeren Projekten ist es üblich mehrere Templates ineinander zu schachteln, was bis zu 10 Ebenen im Widget Tree führen kann. Ein automatische Ausklappen des selbigen würde den Nutzern also vor allem in komplizierteren Projekten unnötigen Aufwand ersparen.

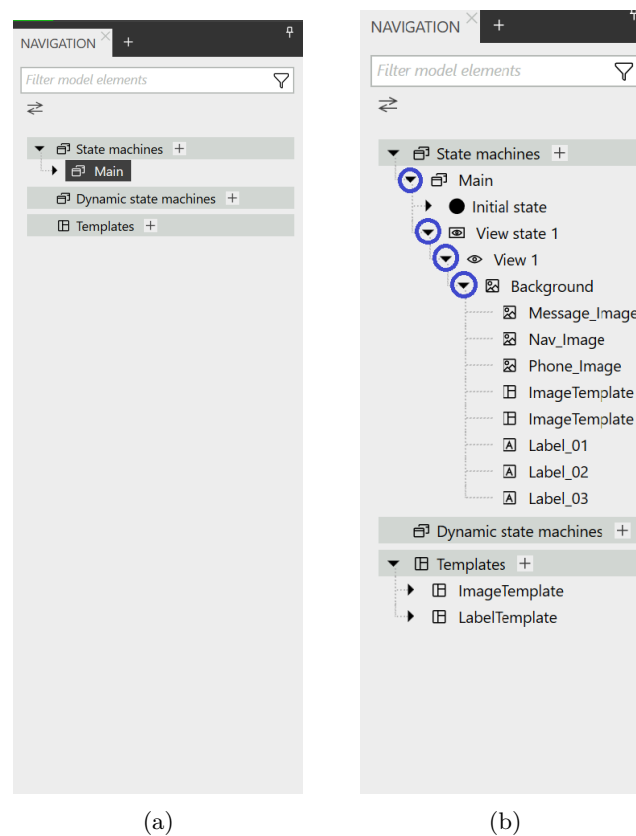


Abbildung 3.1.: Usability - Schwäche Navigation

**Resultierende Benutzeranforderung** Nutzer, die ein neues Element zu ihrer View hinzufügen, müssen die Position dieses Items sofort im Widget Tree nachvollziehen können.

**Template Properties** Ein Widget Template ermöglicht die Definition eines individuellen Widgets, welches beliebig oft in einem EB GUIDE Model benutzt werden kann. Es besteht die Möglichkeit Templates zu erstellen, die auf bereits existierenden Widgets aufbauen, oder von einem anderen Template abgeleitet sind. Nach der Erstellung kann das Template nach den eigenen Wünschen und Bedürfnissen angepasst werden, beispielsweise durch das Hinzufügen von Properties.

Ein Widget Template besitzt außerdem ein Template Interface, welches jene Properties des Templates beinhaltet, die für jede Instanz des Templates sichtbar und veränderbar sein sollen. Jede Instanz eines Templates erbt also die Properties des Template Interfaces. Diese werden Template Properties genannt.[\[Elekc\]](#)

In Abb. 3.2 ist zu sehen, wie die Funktion „publish to template interface“ ausgeführt wird. Hierfür ist zuerst ein Rechtsklick auf das Viereck hinter dem gewünschten Property nötig, bevor noch auf Add to template interface geklickt werden muss. Während der Beobachtung wird deutlich, dass vor allem der Rechtsklick den Arbeitsablauf sehr beeinträchtigt, da dieser in EB GUIDE selten verwendet wird. Zum Großteil wird mit dem normalen Linksklick gearbeitet, weshalb das auch für diesen Fall wünschenswert wäre. In Teil b) von Abb. 3.2 sieht man, dass das verlinkte Property nun einen blauen, statt einem weißen Kreis aufweist. Alle anderen Optionen, wie „Add link to widget property“ wirken sich nach ihrer Anwendung farblich auf das eben angeklickte Quadrat und nicht auf den Kreis aus. Es wäre also denkbar die Funktion „publish to template interface“ einfach durch einen Linksklick auf den Kreis zu aktivieren.

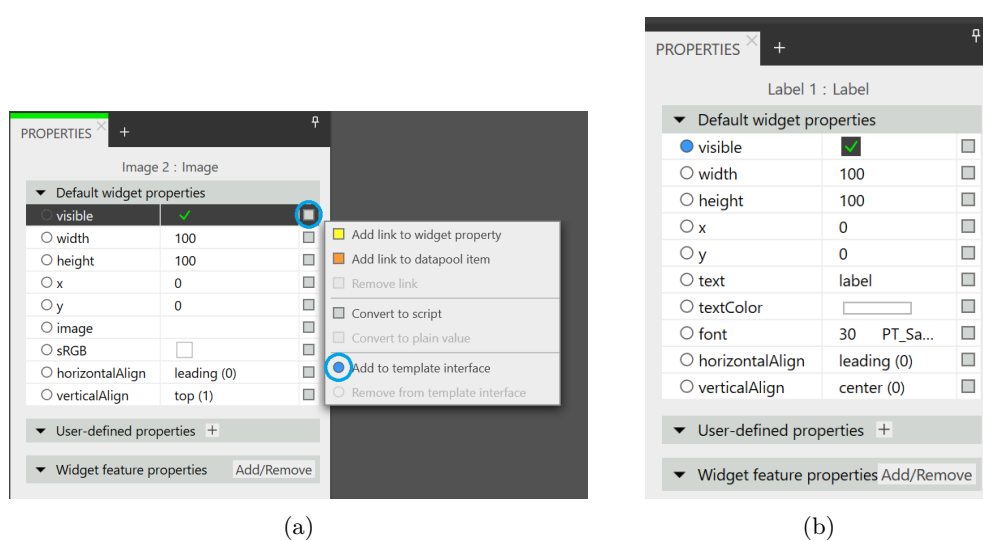


Abbildung 3.2.: Usability - Schwäche Template Properties

**Resultierende Benutzeranforderung** Nutzer, die für ein Property die Funktion „publish to template interface“ ausführen möchten, müssen dies auf eine intuitive und direkte Art und Weise tun können.

**Widget Feature Properties** Neben den Default Widget Properties wie Breite und Höhe, die jede Widgetinstanz besitzt, existieren noch die so genannten Widget Feature Properties. Diese Features fügen weitere anpassbare Funktionen für das Aussehen und Verhalten der Widgets hinzu. Wie in Abb. 3.3 zu sehen, sind die Features in Kategorien unterteilt, die grafisch in Form eines Dropdownmenüs dargestellt sind.

Bei der Beobachtung der Nutzer fällt auf, dass viele genau wissen welches Feature sie hinzufügen wollen, jedoch häufig die zugehörige Kategorie nicht präsent haben. Dies führt dazu, dass jedes Dropdownmenü aufgeklappt wird, bis das gewünschte Feature gefunden wird. Dieser Problematik könnte mit einer Filtermöglichkeit entgegengewirkt werden

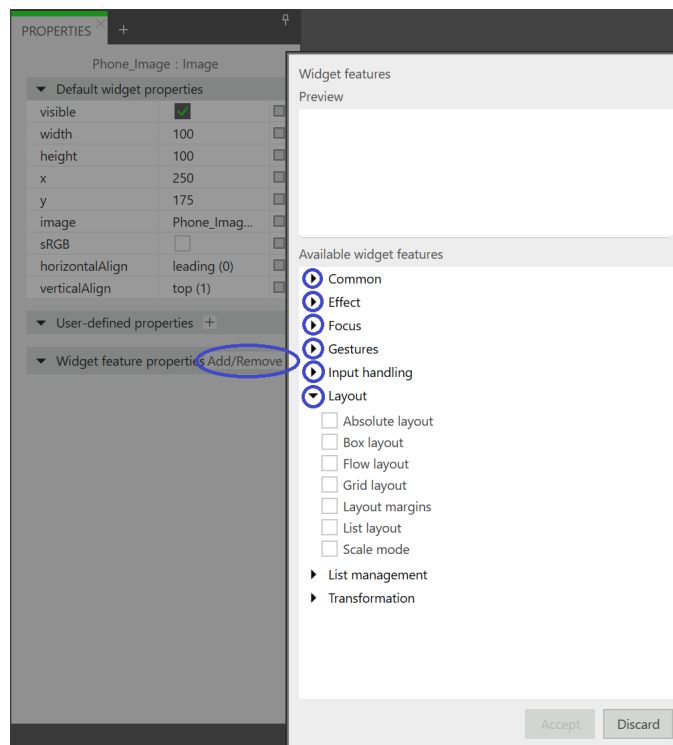


Abbildung 3.3.: Usability - Schwäche Widget Feature Properties

**Resultierende Benutzeranforderung** Nutzer, die den Namen eines gewünschten Features wissen, müssen die Möglichkeit haben dieses Feature aus allen Vorhandenen herauszufiltern.

**Mehrfachselektion** Falls bei mehreren Objekten ein Property auf den gleichen Wert gesetzt werden muss, ist es naheliegend für den Nutzer dies durch die zeitgleiche Selektion der betroffenen Elemente zu lösen. Aktuell bietet EB GUIDE jedoch noch keine Unterstützung für Multiselektion. Sind zwei Elemente ausgewählt lassen sich diese lediglich gemeinsam mit der Maus bewegen, die Properties sind wie in Abb. 3.4 erkennbar, für den Nutzer nicht sichtbar.

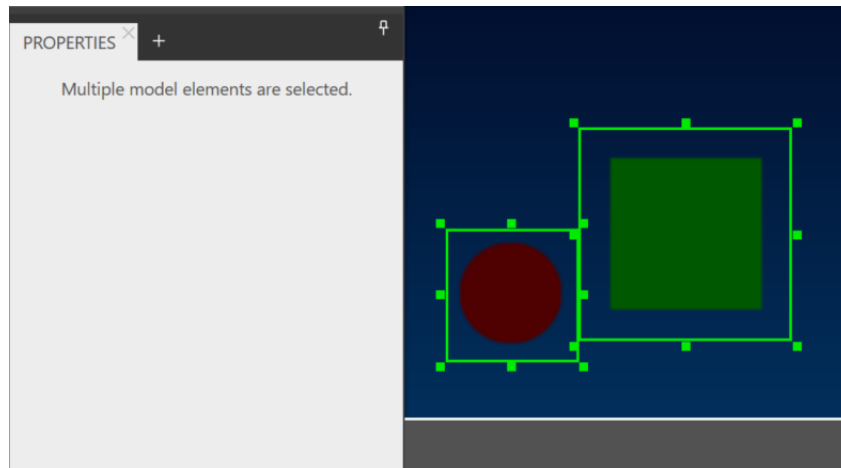


Abbildung 3.4.: Usability - Schwäche Mehrfachselektion

**Resultierende Benutzeranforderung** Nutzer, die mehrere Objekte gleichzeitig selektiert haben, müssen die Möglichkeit haben deren Properties nach ihren Wünschen zu ändern.

## 3.4. Verbesserungen

Aufgrund der zeitlichen Einschränkungen dieser Arbeit ist es nicht möglich alle aufgeführten Schwächen weiter zu analysieren. Im Folgenden wird erläutert auf welchen Grundlagen die Auswahlkriterien festgelegt werden. Daraufaufgehend wird jede Verbesserung anhand dieser Kriterien eingestuft, bevor abschließend das angestrebte Design für die Verbesserungen erläutert wird. In diesem Abschnitt wird bereits der dritte Iterationspunkt des Designprozesses angeschnitten, indem hier der Styleguide für den, im Folgenden erstellten Prototypen, festgelegt wird. Dies geschieht auf Grundlage der bereits erläuterten Gestaltprinzipien der Usability.

### 3.4.1. Auswahlkriterien

Da innerhalb dieser Arbeit nur eine Iteration des Human-Centered Design Process durchgeführt werden kann, ist es wichtig sich auf Schwächen zu fokussieren die gewissen Kriterien

gerecht werden. Zum einen sollte bei einem Teil dieser Schwächen die Möglichkeit bestehen innerhalb einer Iteration bereits verwertbare Ergebnisse zu erhalten. Um jedoch auch einen längerfristigen Mehrwert für Elektrobot zu schaffen, sollte ein anderer Teil auch Verbesserungen beinhalten die höchstwahrscheinlich weiterer Iterationen zum Ausreifen benötigen. In Anbetracht dieser Angabe und des zeitlichen Rahmens ist es also wünschenswert eine Mischung aus ein bis zwei kleineren und einer aufwändigeren Anpassung zu finden, die sich einem zusammengehörigen Test vereinbaren lassen. Die aufwändigere soll hierbei Stoff für weitere Analysen, nach Abschluss dieser Arbeit liefern, die kleineren sollen nach einer Iteration schon zu konkreteren Ergebnissen führen.

Um die Anpassungen in einem Test überprüfen zu können, ist es notwendig sich auf zwei Prinzipien der Usability zu beschränken die gemessen werden, da der Umfang des abschließenden Testes sonst zu umfangreich werden würde. Wie in Abschnitt 3.3.1 erläutert, ergeben sich aufgrund der Zielgruppe aktuell die Möglichkeiten einer Überprüfung auf Effizienz, Wiedererkennungswert, Fehler und Zufriedenheit. Wie bereits erwähnt findet die Evaluierung der Zufriedenheit meist mit Fragebögen statt, die jedoch bei der Untersuchung der anderen Prinzipien nicht anwendbar sind, weshalb die objektive Zufriedenheit in dieser Arbeit nicht untersucht wird. Die Untersuchung des Wiedererkennungswertes wäre ebenfalls möglich. Da es sich bei der ausgewählten Gruppe jedoch um eine Mischung aus Experten und Gelegenheitsnutzern handelt wären hier getrennte Testaufgaben für die Gelegenheitsnutzer nötig, da der Wiedererkennungswert nur bei diesem Teil der Zielgruppe untersucht werden kann. Außerdem ist bei den im vorherigen analysierten Schwächen keine aufgetreten, die direkt mit diesem Prinzip in Verbindung gebracht werden könnte.

Bei den durchgeführten Beobachtungen wurden jedoch Usabilityschwächen erkannt, welche sich höchstwahrscheinlich auf die Effizienz der Nutzer auswirken. Diese Beeinträchtigung findet bei den gefunden Schwächen der Image List, der Navigation und der Widget Feature Properties dadurch statt, dass die Nutzer hier viele Menüs ausklappen oder für jedes Element - im Fall der Image List - eine aufwändige Folge von Klicks durchlaufen müssen. Bei der Funktion „publish to template interface“ lässt sich ebenfalls vermuten, dass der nötige Rechtsklick die Nutzer in ihrer Effizienz beeinflusst. Ebenso ist zu erwarten das das gleichzeitige Anpassen von Attributen durch eine funktionale Multiselektion den Modellervorgang beschleunigt. Ob diese Vermutungen sich wie erwartete auf das Arbeitsverhalten der Modellierer auswirken wird mit den abschließenden Usability Tests herausgefunden. Da für eine Messung dieses Attributes ohnehin eine Zeitmessung nötig ist, kann man währenddessen auch gut die vom System auftretenden Fehler und deren Auswirkungen auf den weiteren Arbeitsverlauf festhalten. Die ausgewählten Verbesserungen müssen also hauptsächlich auf Effizienz und gegebenenfalls noch auf auftretende Fehler untersuchbar sein.

Zusätzlich zu den bereits genannten Kriterien, wird darauf geachtet vor allem solche Verbesserungen zuerst zu berücksichtigen, die dem Nutzer, nach objektiven Meinungen, aktuell den größten Mehrwert liefern.

### 3.4.2. Auswahl nach Auswahlkriterien

**Image List** Die Anpassung der Befüllung der Image List ist als eine umfangreiche, aufwändige Anpassung einzuordnen. Allerdings ist hier bereits absehbar, dass eine Anpassung die Effizienz deutlich steigern würde, da ein gruppiertes Einfügen auf jeden Fall eine Zeiterparnis mit sich bringt. Darüber hinaus haben sich, für die Lösung dieses Problems einige Teams bereits Plug-Ins implementiert, die genau dieses Problem beheben. Diese sind zwar nicht Bestandteil des offiziellen Tools, werden jedoch von den internen Teams bereits routiniert benutzt, weshalb diese Anpassung auf kurze Sicht keinen Mehrwert liefern würde, sondern dies erst für neu startende Projekte der Fall wäre. Die Anpassung der Image List würde also zum einen nicht das Kriterium erfüllen für weitere Analysen geeignet zu sein, da der Mehrwert der Anpassung bereits sehr deutlich ist. Zum anderen existiert bereits eine temporäre Lösung für dieses Problem.

**Navigation** Bei der Anpassung der Navigation würde es sich um eine kleinere Anpassung handeln. Allerdings gibt es für diese Problematik ebenfalls bereits eine Lösung die in Abb. 3.5 zu sehen ist. Es handelt sich hierbei um ein Panel in EB GUIDE Studio, welches sich Standardmäßig unterhalb des Navigationspanels befindet. Greift man noch einmal das Beispiel aus Abb. 3.1 auf, sieht die Lösung in EB GUIDE Studio für dieses Problem aktuell so aus, dass für jedes ausgewählte Objekt dessen Umgebung in der Outline angezeigt wird. Dies funktioniert auch gut für neu eingefügte Elemente, da diese defaultmäßig nach dem Einfügen ausgewählt werden.

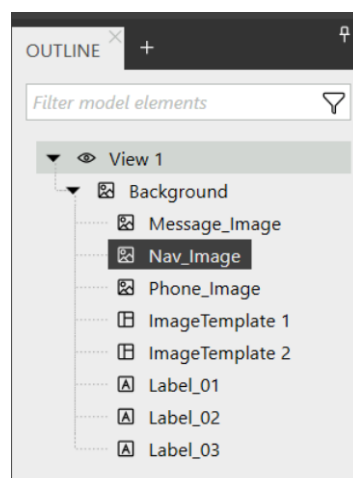


Abbildung 3.5.: Ergänzung zur Navigation in Form einer Outline

Während der Beobachtung der Nutzer fällt jedoch auf, dass keiner der Nutzer diesen Outliner zu benutzen scheint, sondern lieber direkt im Widget Tree arbeitet. An diesem Punkt wären also weitere Analysen nötig, ob manche Nutzer eventuell nichts von der Existenz dieser Funktion wissen oder ob ihnen die Nutzung zu unhandlich ist. Aufgrund dieser ausstehenden Analysen ist diese Schwäche jedoch auch nicht für eine Überarbeitung in dieser Bachelorarbeit geeignet.

**Template Properties** Die Verlagerung des Hotspots für die Funktion „publish to template interface“, kann ebenfalls als eine kleine Änderung betrachtet werden. Für diese Anpassung lässt sich mit einer Gegenüberstellung des alten und neuen Interfaces gut die benötigte Zeit der Nutzer messen, außerdem kann man hier gut herausfinden, ob die Verlagerung für die Nutzer intuitiv ist oder ob doch die alte Variante bevorzugt genutzt wird. Es werden also bereits nach der ersten Iteration Ergebnisse auftreten, die eine konkrete Beurteilung des Vorgehens zulassen, weshalb sich diese Anpassung für den Test eignet.

**Widget Feature Properties** Bei dem Filter für die Widget Feature Properties handelt es sich um eine kleine bis mittelgroße Anpassung. Der Mehrwert ist hier allerdings, vor allem für Experten, die die Namen der Features kennen, bereits absehbar. Trotzdem eignet sich diese Anpassung ebenfalls gut für die Untersuchung der Effizienz. Dass es eine Steigerung der Benutzerfreundlichkeit gibt, ist stark zu erwarten, ob es den Nutzern jedoch tatsächlich auch zu einem schnelleren Arbeitsablauf verhilft ist unklar. Zudem ist es hier interessant, ob eine komplette, neue Ergänzung im Interface von den Nutzern überhaupt selbstständig erkannt wird. Diese Anpassung kann also als zweite kleinere Anpassung neben der Anpassung der Template Properties untersucht werden.

**Mehrfachselektion** Bei der Anpassung für die Mehrfachselektion, besteht die Besonderheit, das EB GUIDE hierfür im Gegensatz zu den anderen aufgeführten Schwächen, noch keinerlei Implementierung bietet. Es ist aktuell lediglich möglich mehrere Elemente gleichzeitig auszuwählen und in der View zu bewegen, es existiert jedoch kein Panel welches die aktuellen Properties anzeigt. Dadurch ist es sehr wahrscheinlich, dass mehrere Iterationen nötig sind um die Funktionen zufriedenstellend für den Nutzer darzustellen. Ebenfalls ist eine Untersuchung auf Effizienz sehr gut möglich, da ein direkter Vergleich des entstehenden Zeitaufwandes mit und ohne Multiselektion getätigt werden kann. Wegen dieser Merkmale eignet sich die Multiselektion also sehr gut als große Anpassung in dieser Arbeit.



### 3.4.3. Design der Verbesserungen

**Template Properties** Wie in Abb. 3.6 zu sehen ist für diese Änderung keine tatsächliche Abwandlung des Designs notwendig, das sich lediglich der Hotspot verlagert. Trotzdem werden hier einige Gestaltprinzipien der Usability berücksichtigt. Die nötige Rückmeldung an den Nutzer geschieht durch die Änderung der Farbe des Kreises. Dieses Verhalten ist bereits vorhanden, allerdings wird durch die bisherige Lösung eine minimale Verletzung des Mappings verursacht, da durch eine Interaktion mit dem Quadrat eine Rückmeldung mithilfe des Kreises erzeugt wurde. Das Mapping wird also durch das hier vorgestellte Design verbessert.

Im Gegensatz dazu wird jedoch durch die Anpassung die interne Konsistenz verletzt, da die Interaktion mit den Template Properties auch noch an anderen Stellen möglich ist, wo die Anpassung zum aktuellen Zeitpunkt noch nicht integriert werden kann. Diese Verletzung kann jedoch nach Validierung durch den Usability Test im Nachhinein angepasst werden, oder der Hotspot auf dem Kreis könnte als zusätzliche Ergänzung für Expertennutzer existieren.

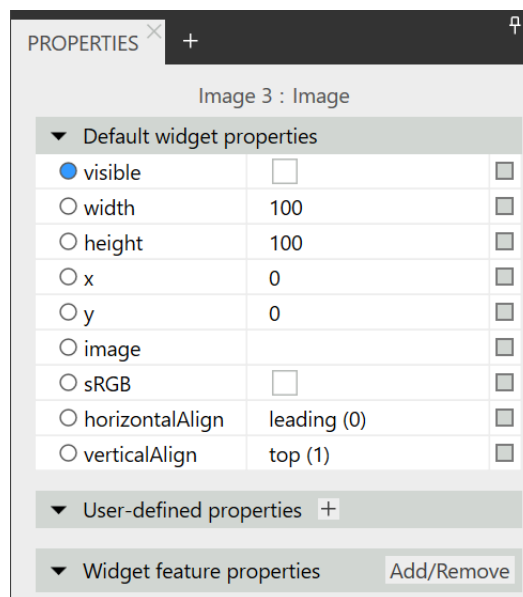


Abbildung 3.6.: Verbesserung Template Properties

**Widget Feature Properties** Für das Hinzufügen einer Filterfunktion ist das Einfügen einer Filterleiste in das Interface, wie in Abb. 3.7 zu sehen, nötig. Teil a) der Abbildung zeigt hier zur Erinnerung noch einmal die bestehende Implementierung, in Teil b) ist der ergänzte Filter zu sehen der in c) live nach dem eingegebenen Begriff filtert.

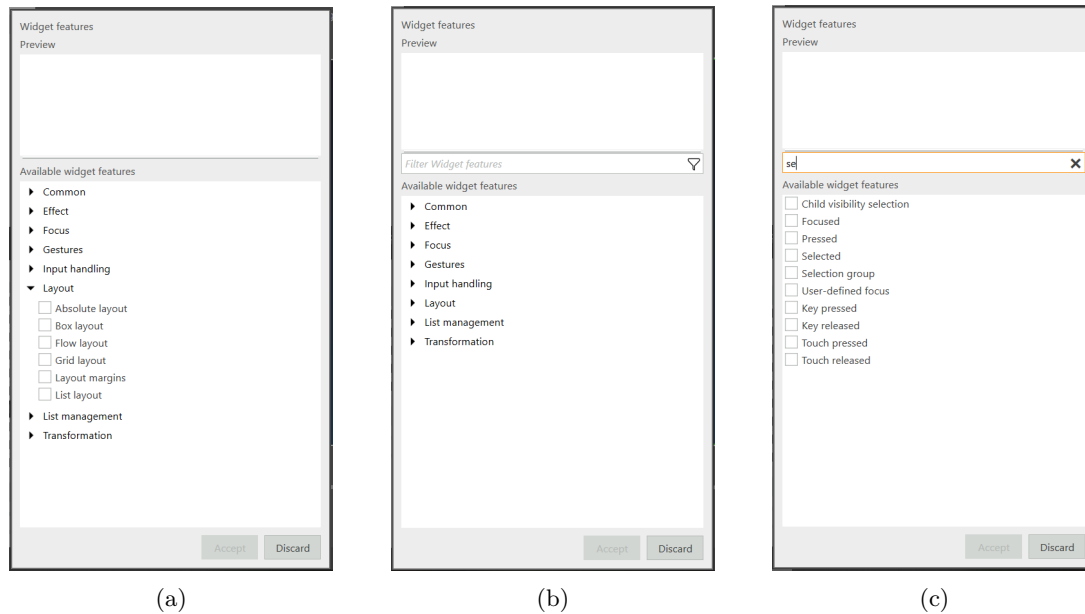


Abbildung 3.7.: Verbesserung Feature Property Properties

Als Gestaltprinzip der Usability wird hier, zum einen die ästhetische, funktionale und interne Konsistenz des Designs, zum anderen Such- und Filterfunktionen, die in der Benutzeroberfläche von EB GUIDE Verwendung finden gewahrt. Dadurch ergibt sich automatisch eine externe Konsistenz, da das bereits bestehende Design dieser Leisten von Elektrobitt an das Design bereits bekannter Suchleisten angelehnt wurde.

Aktuell wird die in Abschnitt 2.7 erwähnte Problematik der Sichtbarkeit in komplexen Systemen durch die ausklappbaren Menüs gelöst. Dies führt jedoch in diesem konkreten Fall häufig dazu, dass all diese Menüs, während der Suche nach dem richtigen Feature, aufgeklappt werden müssen. Der geplante Filter soll die Nutzer insofern unterstützen, dass sie die Sichtbarkeit der Features nach ihren eigenen Bedingungen anpassen können, wenn sie wissen nach welchem Feature sie suchen.

Das Gestaltprinzip der Affordanz wird durch den ausgegrauten Text in der Filterleiste berücksichtigt, da der Nutzer durch den Text „Filter Widget features“ die direkte Aufforderung erhält was hier getan werden kann. Zum anderen bekommt er durch den grauen Text suggeriert, dass hier etwas anklickbar ist und eine Eingabe getätigt werden kann.

Die Rückmeldung, die für den Nutzer sehr wichtig ist, erfolgt durch die Tatsache, dass der Filter live auf die Änderungen des Nutzers reagiert, es also eine sehr schnelle Anpassung der Liste der Features gibt.

**Mehrfachselektion** Um die Anpassung der Properties bei Mehrfachselektion zu ermöglichen ist es nötig ein Property Panel anzuzeigen, wie in Abb. 3.8 zu sehen. Wie bei der normalen Auswahl eines Elementes werden dort Properties wie width/height und die Koordinaten angezeigt, zusätzlich ist es mit den neuen Alignment Actions noch möglich die ausgewählten Elemente aneinander auszurichten. Diese Funktion gibt es bisher in EB GUIDE Studio noch nicht, erschien jedoch im Kontext der Mehrfachselektion eine geeignete Ergänzung zu sein.

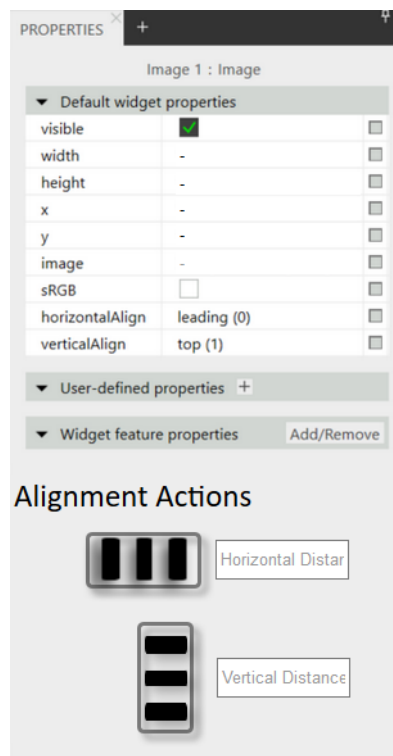


Abbildung 3.8.: Verbesserung Mehrfachselektion

Auch hier wird die ästhetische, funktionale und interne Konsistenz bei den Textfeldern der Properties gewahrt, da sie exakt so aussehen, wie die bereits Bekannten bei der Auswahl eines Elements. Externe Konsistenz wird durch die Striche gewahrt, die als Platzhalter bei den Values der Properties fungieren, wenn die ausgewählten Elemente zum Beispiel bei der x-Koordinate unterschiedliche Werte aufweisen. In EB GUIDE ist es allgemein so, dass nur die Properties sichtbar sind, die für die aktuell ausgewählten Elemente relevant und veränderbar sind. Diese Einschränkung der Sichtbarkeit ist dementsprechend auch bei der Multiselektion zu beachten. Wenn also beispielsweise ein Bild und ein Text gleichzeitig

ausgewählt sind, ist es nicht nötig eine Text Property bereitzustellen, da diese nicht bei beiden Elementen anpassbar wäre.

Das Prinzip der Affordanz wird bei den Buttons für die Alignment Actions berücksichtigt, indem diese einen Schatten bekommen um sie klickbar erscheinen zu lassen. Ebenfalls wird es wieder bei den Textfeldern neben den Buttons berücksichtigt, wie es auch bei der Filterfunktion bereits der Fall ist.

Eine Rückmeldung an den Nutzer, dass seine Eingaben erfolgreich waren, erfolgt durch die sofortige Veränderung der Elemente im View, entweder durch Änderungen der Position, oder der Breite und Höhe.

Zusätzlich dazu soll es noch möglich sein Bilder in Templates über Multiselektion einzufügen. Das Design hierfür ist in Abb. 3.9 zu sehen. Die Design Prinzipien werden hier auf die gleiche Art und Weise berücksichtigt wie es bei den Buttons für die Alignment Actions bereits erläutert wurde. In diesem Fall ist jedoch kein Textfeld nötig, da hier kein Abstand oder ähnliches angegeben muss. Bei einem Klick auf den Button wird ein ausgewähltes Bild in ein gleichzeitig ausgewähltes Template eingefügt.

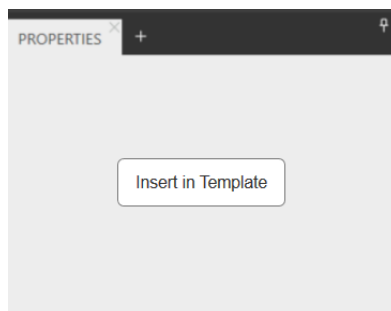


Abbildung 3.9.: Verbesserung Mehrfachselektion Einfügen in Template

## Kapitel 4.

### Umsetzung

Nach Abschluss der Spezifikation der Benutzeranforderungen wird nun in den dritten Schritt des Human-Centered Design Process übergegangen. Hier gilt es, dass UX Design zu erstellen, wobei der Styleguide bereits am Ende des letzten Kapitels fertiggestellt wurde. In den folgenden Abschnitten werden noch die benötigten Interaktionsmöglichkeiten definiert und ein, auf diesen Erkenntnissen aufbauender Prototyp erstellt.

Der Großteil der Anpassungen wird in Form eines Prototypen umgesetzt, um weitere Evaluierungen leichter durchführen zu können und um eventuelle Anpassungen innerhalb kommender Iterationen leichter umsetzen zu können. Konkret werden die große Anpassung der Multiselektion und die kleine Änderung für die Template Properties innerhalb des Prototypen umgesetzt und untersucht, der Filter für die Widget Feature Properties hingegen wird direkt im EB GUIDE Projekt implementiert. Dies hat den Hintergrund, dass es bei der Multiselektion absehbar ist, dass hier noch einige Anpassungen nötig sein werden, bis das Design und die Funktionalität an die Bedürfnisse der Nutzer angepasst sind. Das liegt vor allem daran, dass hier noch keinerlei Grundlagen in EB GUIDE vorhanden sind, auf denen aufgebaut werden kann. Bei den Template Properties hat die Umsetzung innerhalb des Prototypen den Hintergrund, dass hier nicht sicher gesagt werden kann, ob die Verlagerung der Funktion „publish to template interface“ von den Nutzern tatsächlich angenommen oder überhaupt an der angedachten Stelle intuitiv erwartet wird. Es ist hier auch sehr wahrscheinlich, dass an der Darstellung, Positionierung und der Kombination mit der bisherigen Position noch einige Anpassungen stattfinden müssen. Daher ist es sinnvoll, sich in diesem Fall strikt an den Human-Centered Design Process zu halten und innerhalb dessen Iterationen, solange das UX Design zu verbessern bis es sicher den Benutzeranforderungen entspricht. Ansonsten würde hoher Entwicklungsaufwand in Anpassungen fließen die noch häufig verändert oder komplett verworfen werden müssen. Die Anpassung des Prototypen kann in diesen Fällen schneller, minimalistischer und mit kleinerem Aufwand erfolgen.

Im Gegensatz dazu macht es Sinn die Filterfunktion direkt in EB GUIDE zu implementieren. Aus den bereits durchgeführten Analysen geht hervor, dass dieser Filter einen großen Mehrwert für die Nutzer bieten würde. Im Gegensatz zu den beiden anderen Änderungen gibt es hier keinen großen Spielraum mehr für Anpassungen, da sich das Design an den bereits

bestehenden Filter und Suchfunktionen innerhalb von EB GUIDE orientiert. Die einzige eventuelle Anpassung die hier noch auftreten könnte, wäre eine Änderung der Position. Ob dies jedoch innerhalb eines Prototypen oder im Code stattfindet, macht - betrachtet man den zeitlichen Aufwand - keinen Unterschied. Daher ist es hier effizienter, die Änderungen direkt zu implementieren und das Verhalten nicht zuerst innerhalb eines Prototypen zu simulieren.

## 4.1. Prototyp Multiselektion und Template Properties

In den folgenden Abschnitten werden alle relevanten Aspekte im Bezug, auf die im Prototyp umgesetzten Änderungen erläutert. Zuerst folgt eine kurze Begründung für die Wahl des Prototyping Tools. Da für diese Arbeit eine Einarbeitung in dieses Tool erfolgt ist, wird dieses darauf folgend in seinen Grundfunktionen erläutert. Dies ist auch notwendig um die abschließend dargestellte Vorgehensweise zur Erstellung des Prototyps nachvollziehen zu können. Danach werden die nötigen Interaktionsmöglichkeiten definiert, die dem Nutzer für einen erfolgreichen Umgang mit dem Prototypen simuliert werden müssen, bevor letztlich die Vorgehensweise beschrieben wird um diese Anforderungen umzusetzen.

### 4.1.1. Axure RP

Axure RP ist eine Software die es ermöglicht Wireframes, Prototypen, Dokumentationen und Spezifikationen für Web-, Mobil- oder Desktopanwendungen zu erstellen. Dies wird durch Drag and Drop Aktionen, Skalierung und Formatierung von Widgets ermöglicht, mit deren Hilfe man das gewünschte Endprodukt erstellen kann. Für diese Arbeit ist vor allem die Möglichkeit einen Prototypen zu erstellen wichtig. Ausschlaggebend für die Wahl dieser Software ist, dass AXURE RP 9 die Möglichkeit bietet dynamische Inhalte zu erstellen und logische Bedingungen in den Prototypen einzubinden. Da es sich bei EB GUIDE Studio um ein Modellierungstool handelt, ist ein rein statischer Prototyp nicht ausreichend. Ebenfalls reicht es nicht aus feste Hotspots zu haben bei deren Klick etwas passiert, sondern es ist notwendig, die in EB GUIDE herrschenden Einschränkungen für den Nutzer simulieren zu können.

Ein weiterer Grund sich für diese Software zu entscheiden ist es, dass die UX Designer bei Elektrobit ebenfalls mit AXURE RP arbeiten. So ist zum einen gewährleistet, dass bei den Usabilitytest keine Schwierigkeiten mit der verwendeten Software auftreten würde, zum anderen ist dadurch gleichzeitig ein Ansprechpartner bei auftretenden Problemen vorhanden.

In Abb. 4.1 ist das Standardprojekt von AXURE RP zu sehen, welches die Grundlage eines jeden neuen Projektes bildet.

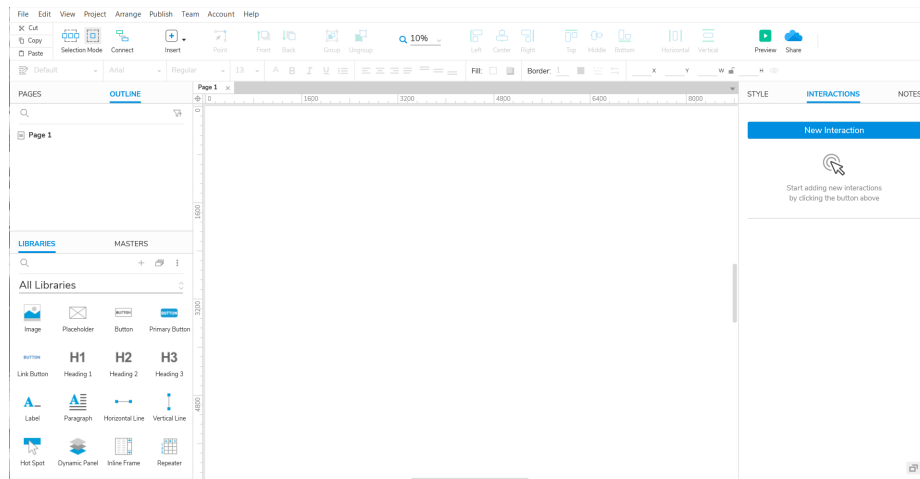


Abbildung 4.1.: Axure RP 9

In den mittig platzierten Workspace können Elemente aus den Libraries links oder eigene, externe Elemente eingefügt werden. Zieht man nun beispielsweise eine Droplist und drei Kreiselemente aus den Libraries links in den Workspace, passt sich die Outline wie in Abb. 4.2 zu sehen entsprechend an. Mithilfe dieser Outline kann man innerhalb des Projektes navigieren, was vor allem praktisch ist wenn sich mehrere Elemente überlagern. Elemente lassen sich auch zu Dynamic Panels gruppieren, was vor allem dann nötig ist, wenn man ein Drag und Drop Verhalten simulieren möchte, da dies die einzigen Widgets sind die dieses Verhalten unterstützen.

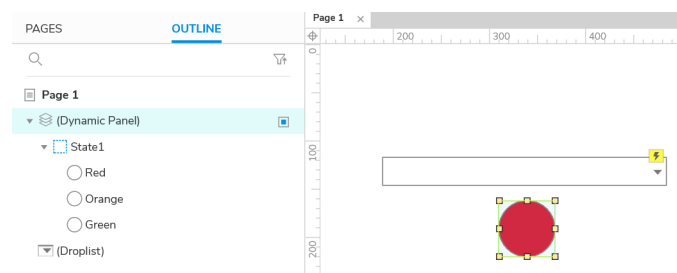


Abbildung 4.2.: Axure RP 9 Outline

Um den drei eingefügten Kreisen unterschiedliche Farben zuzuweisen ist es nötig deren Properties zu bearbeiten. Dies ist in AXURE RP über das Style Panel möglich, welches in Abb. 4.3 zu sehen ist. Hier ist es beispielsweise ebenfalls möglich Skalierungen vorzunehmen oder Umrandungen und Schatten hinzuzufügen.

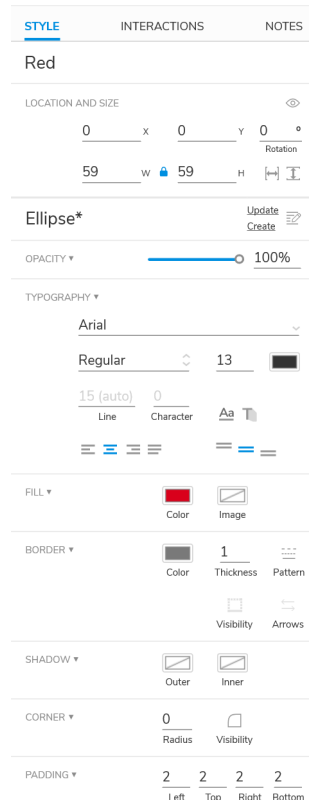


Abbildung 4.3.: Axure RP 9 Style Properties

Möchte man nun erreichen, dass die Farbe des sichtbaren Kreises sich an der Auswahl innerhalb der Droplist orientiert, muss Logik zu dem Prototypen hinzugefügt werden. Zuerst ist es, wie in Teil a von Abb. 4.4 zu sehen, nötig Optionen zu der Droplist hinzuzufügen. Innerhalb des Interaction Panel besteht dann die Möglichkeit, über IF Conditions abzufragen, welche Option aktuell ausgewählt ist. Passend dazu wird der Kreis mit der gewünschten Farbe angezeigt, und die Visibility der anderen Kreise entsprechend auf null gesetzt.

Hat man die Modellierung abgeschlossen, oder möchte Anpassungen überprüfen, bietet AXURE RP die Möglichkeit einer lokalen Preview. Diese verhält sich exakt so, wie sich der abgeschlossene Prototyp verhalten wird und eignet sich deshalb gut für selbständige Validierung der bisherigen Umsetzung. Testet man das soeben modellierte Beispiel in der Preview sieht man in Abb. 4.4, dass sich die Farbe des Kreises, wie gewünscht, immer an die Auswahl der Droplist anpasst.

Ist die Modellierung beendet, bietet AXURE RP die Möglichkeit, das Projekt in die AXURE CLOUD zu laden. Hier können andere Mitarbeiter über einen generierten Link auf das funktionale Endprodukt zugreifen, ohne das eigentliche Projekt sehen oder bearbeiten zu können. Dadurch ist es nun einerseits möglich andere UX Designer Funktionalitäten des Prototypen testen zu lassen, wobei sie auch an den entsprechenden Stellen Kommentare und TODO's hinterlassen können. Außerdem besteht durch die Online Verfügbarkeit die



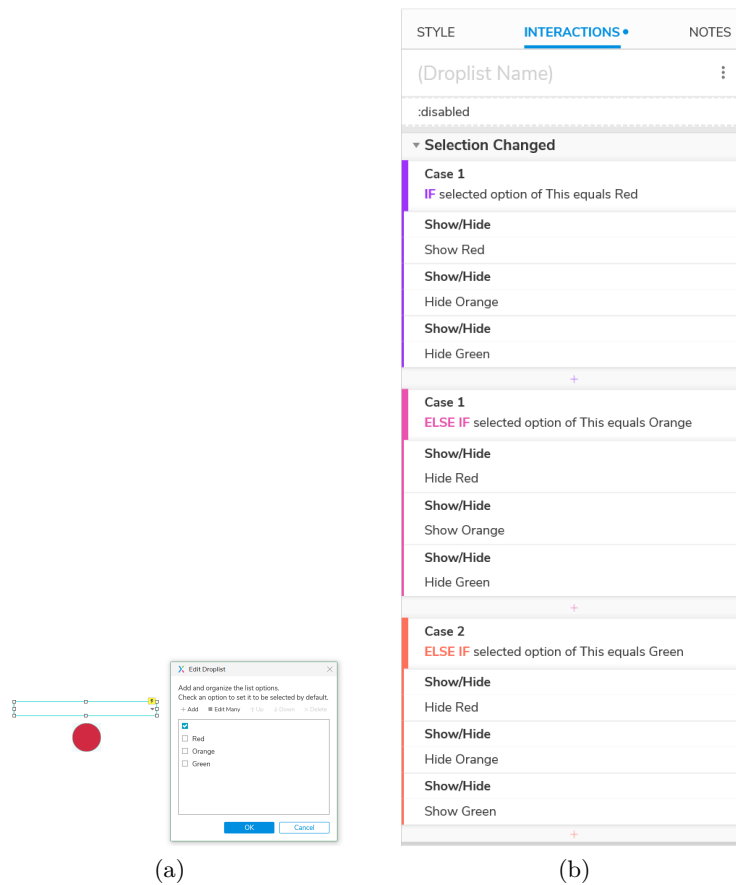


Abbildung 4.4.: Axure RP Droplist und Conditions



Abbildung 4.5.: Axure RP Preview

Möglichkeit den Prototypen für die Testpersonen im Rahmen des Usability Tests zugänglich zu machen.

#### 4.1.2. Interaktionsmöglichkeiten

Im dritten Schritt des Human-Centered Design Process ist es die Aufgabe des Interaction Designers die Interaktionsmöglichkeiten, die der Prototyp bieten muss, zu spezifizieren. Dies ist deshalb notwendig, da es gerade bei Prototypen die ein komplexes System simulieren sollen, nicht möglich oder nötig ist alle Funktionen des Systems zu simulieren. Daher wird vor der Erstellung des Prototyps festgelegt welche Interaktionen von den Nutzern durchführbar sein müssen um die geplanten Verbesserungen validieren zu können.

Für die Prüfung der Usability der Ergänzungen ist es jedoch nicht ausreichend nur die neuen oder ergänzten Funktionen zu simulieren. Um diese überhaupt anwenden zu können muss der Nutzer einen gewissen Fortschritt innerhalb des Modellierungsprozesses erreicht haben. Um an diesen Punkt zu gelangen ist es unbedingt notwendig, dass dem Nutzer gewisse Grundfunktionen von EB GUIDE zur Verfügung stehen. Das bedeutet für den konkreten Fall dieser Bachelorarbeit, dass es für den Nutzer möglich sein muss wie gewohnt Elemente per Drag and Drop in den View zu ziehen, und diese dort auch frei bewegen zu können. Ebenfalls müssen die Properties aller Widgets frei anpassbar sein und die Elemente müssen wie gewohnt auf die Eingaben des Nutzers reagieren. Da sich die Funktion „publish to template interface“ innerhalb der Templates befindet, muss auch die Möglichkeit bestehen Templates anzulegen.

Darauf aufbauend müssen noch die neuen oder veränderten Funktionen in den Prototypen integriert werden. Dazu zählt konkret, dass die Funktion „publish to template interface“ verlagert wird und die vorhandene Möglichkeit dafür vorerst nicht simuliert werden muss. Zusätzlich muss es nun möglich sein, mehrere Elemente gleichzeitig auszuwählen und hierfür ein Property Panel einzublenden. Die Anpassung der Properties müssen sich in diesem Fall auch auf alle angewählten Elemente auswirken und diese müssen gemeinsam im View bewegt werden können. Zusätzlich müssen noch die neuen Alignment Actions, sowie die Funktion „Insert in Template“ angezeigt werden und ausführbar sein.

Da sehr viele unterschiedliche Widget- und Templatetypen in EB GUIDE existieren, ist es an dieser Stelle des Prozesses auch sinnvoll sich bereits Gedanken über den abschließenden Usability Test zu machen und die Funktionen des Prototypen entsprechend einzuschränken. Die Modellierer bei Elektrobit setzen hauptsächlich HMIs für den Fahrzeuginnenraum mit EB GUIDE um. Daher erscheint es sinnvoll für den Usability Test einen minimalistischen Startscreen eines solchen Interfaces modellieren zu lassen. Da diese, wenn man die Interaktionslogik außen vor lässt, nur aus Images und Labels bestehen ist es ausreichend diese beiden Widgets funktionsfähig zu machen. Zusätzlich dazu ist es noch notwendig die Menge

an verfügbaren Templates einzuschränken. Für den soeben erläuterten Use Case ist es hier ebenfalls ausreichend nur Image Templates erstellen zu können. Genauere Erläuterungen zu den Aufgaben innerhalb des Usability Test folgen entsprechend in Kapitel 5.

### 4.1.3. Vorgehensweise

Im folgenden werden die grundlegenden Vorgehensweisen erläutert die angewandt werden, um den Prototypen zu erstellen. Es ist im Rahmen dieser Arbeit nicht möglich alle exakten Anpassungen zu erläutern die umgesetzt werden. Es werden jedoch alle maßgeblichen Konzepte und Funktionen aufgeführt die in Zusammenhang mit den eingebauten Neuerungen stehen. Alle nicht ausführlich erklärten Änderungen sind auf ähnliche Art und Weise umgesetzt, wie solche die in den nächsten Abschnitten erläutert werden.

**Grundlage** Vor allem bei Expertennutzern ist es wichtig, dass der Prototyp sich nicht nur verhält wie die gewohnte Software, sondern sich auch optisch an ihr orientiert. Aus diesem Grund ist der Ansatz zur Modellierung des Prototypen, einen Screenshot der aktuellen Version von EB GUIDE Studio zu machen und diesen als statischen Hintergrund, wie in Abb. 4.6 zu sehen, für das weitere Vorgehen zu verwenden. Darauf aufbauend können nach und nach Interaktionsmöglichkeiten innerhalb des Prototypen platziert werden.

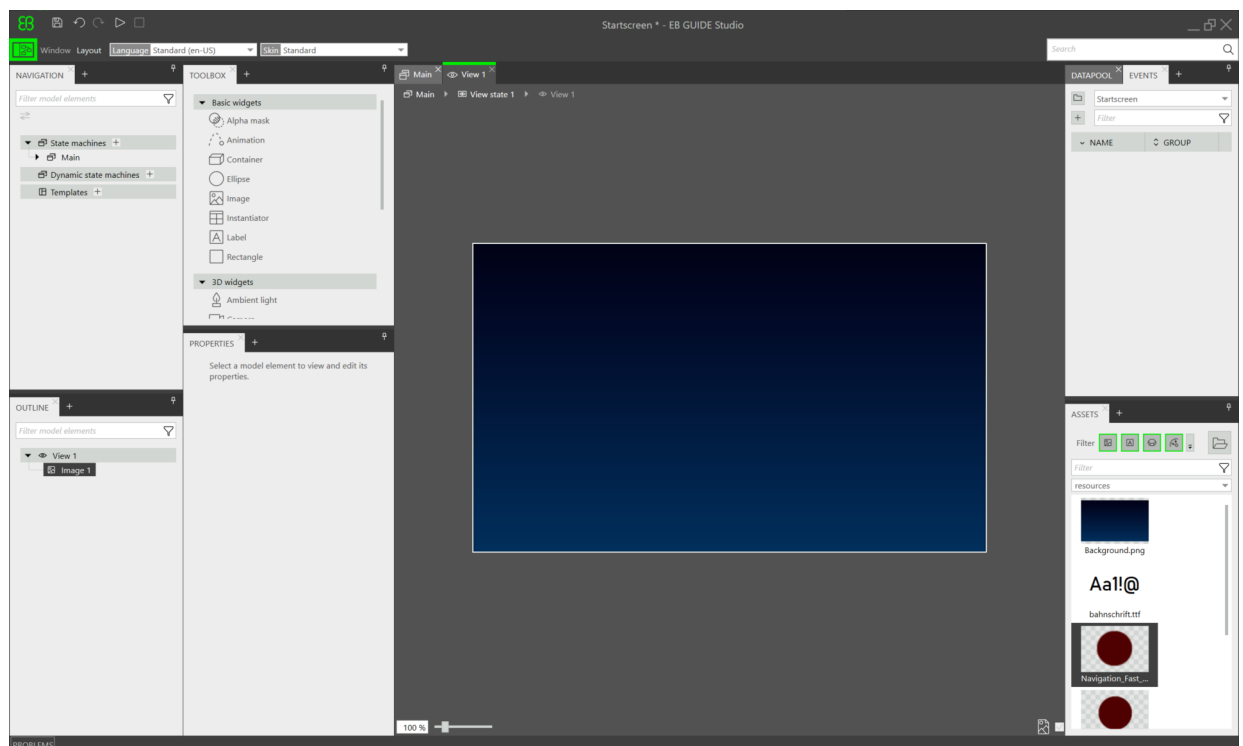


Abbildung 4.6.: Grundlage des Prototyps

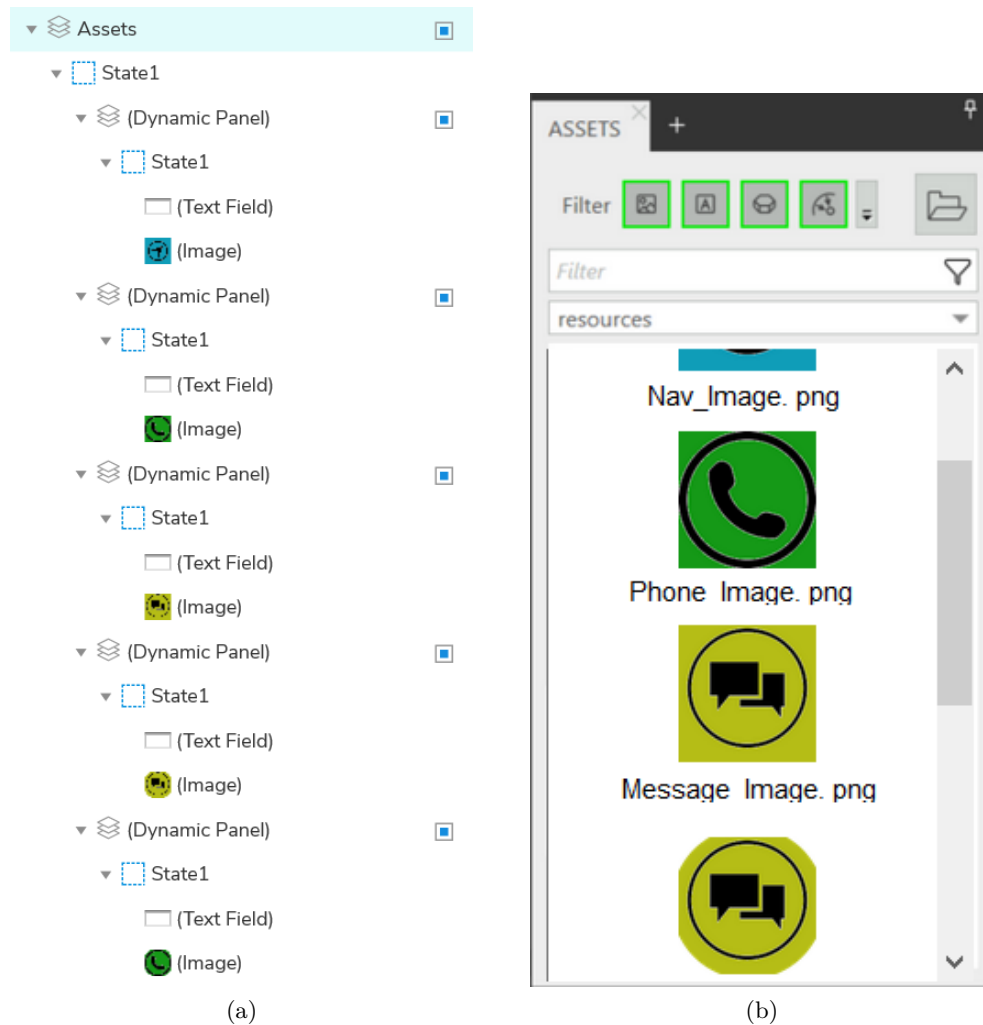


Abbildung 4.7.: Scrollbar für Assets

**Assets und Toolbox** Die Assets und die Toolbox sind wichtige Interaktionsmöglichkeiten für den Nutzer, da von hier alle zur Verfügung stehenden Widgets oder Ressourcen in den View gezogen werden. Für beide Elemente wird eine Scrollbar benötigt, da die Nutzer dies an dieser Stelle gewohnt sind, und sonst keine Möglichkeit bestünde alle Widgets und Elemente anzuzeigen. In AXURE ist es hierfür nötig ein Dynamic Panel zu erstellen und es mit allen benötigten Elementen zu befüllen, die in der Scrollbar vorhanden sein sollen. Für ein Dynamic Panel ist es möglich eine feste Größe anzulegen, sollten dessen Elemente in der vorgegebenen Dimension nicht alle Platz finden, hat man nun die Möglichkeit Vertikales oder Horizontales Scrolling zu aktivieren. In Abb. 4.7 sieht man beispielhaft die Umsetzung für das Assetspanel. Teil a) zeigt die Implementierung mithilfe von ineinander geschachtelter Dynamic Panels, in Teil b) ist das entsprechende Ergebnis im Prototypen zu sehen.

Der Grund dafür, dass jedes einzelne Bild in den Assets ebenfalls ein Dynamic Panel ist, ist der Tatsache geschuldet, dass dies die einzigen Widgets sind die Drag und Drop unterstützen.

Sie bieten die einzigartigen Interactions „Drag Started“, „Dragged“ und „Drag Dropped“ über welche gesteuert werden kann, wie sich die Items während des Vorgangs verhalten. Hier erweist es sich als praktisch „Drag Dropped“ mit einer If Bedingung zu versehen, damit das Element nur über dem dafür vorgesehenen View abgelegt werden kann und nicht zum Beispiel im Bereich der Datapoolitems.

**Widget Tree** Sobald etwas in den View hinzugefügt wird, muss dies für den Nutzer ebenfalls im Widget Tree sichtbar werden. Dies ist mithilfe eines Screenshots gelöst - zu sehen in Abb. 4.8 - wobei sich noch weitere Elemente im Widget Tree befinden, die vorläufig überdeckt werden. Fügt der Nutzer nun das entsprechende Element zum View hinzu, wird die Abdeckung entfernt und das Objekt kann nun über den Widget Tree ausgewählt werden. Die Interaktionsmöglichkeit wird von AXURE mit den gelben Blitzen gekennzeichnet und ist ebenfalls mit logischen Bedingungen umgesetzt.

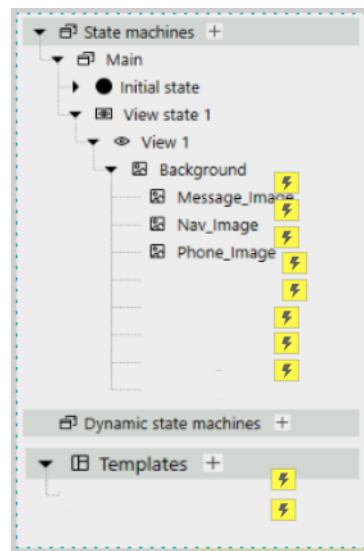


Abbildung 4.8.: Widget Tree

**Properties** Um die Interaktion mit allen eingefügten Elementen möglich zu machen, erhält jedes Element sein eigenes Property Panel, welches in Abb. 4.9 zu sehen ist. Damit dem Nutzer immer klar ist, welches Element aktuell ausgewählt ist, wird auch hier das Verhalten von EB GUIDE nachgebaut, indem jedes ausgewählte Element einen grünen Rahmen erhält. Die Auswahl hierfür muss über das Element selbst oder den Widget Tree möglich sein.

Ist das Objekt aktuell ausgewählt erscheint ein entsprechendes Property Panel, welches ebenfalls einen Screenshot beinhaltet der mit Textfeldern überlagert wird. Diese Felder sind mit globalen Variablen verknüpft, welche wiederum die Properties der Bilder im Workspace von AXURE anpassen. Dadurch wird es ermöglicht, dass Eingaben in das Textfeld das

Objekt verändern und umgekehrt, Bewegungen des Objektes die Werte in den Textfeldern aktualisieren. Damit wird für den Nutzer das exakt gleiche Verhalten nachmodelliert, welches in EB GUIDE existiert. Ein identisches Verhalten ist für die Texte modelliert, nur haben diese statt eines Image Properties ein Text Property.

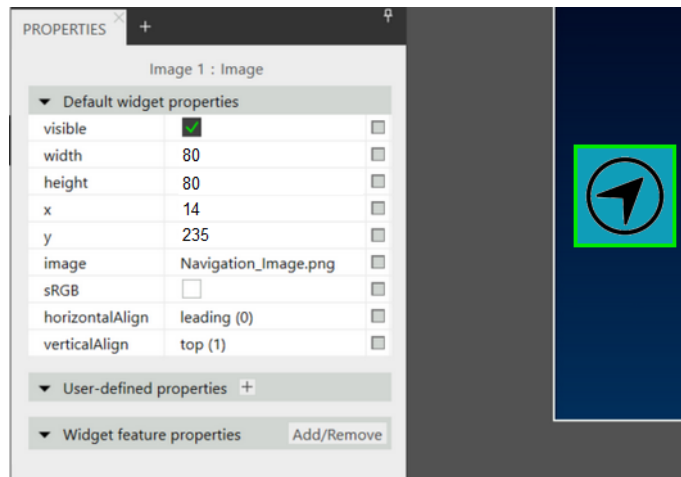


Abbildung 4.9.: Properties Panel

**Multiselektion** Ist nur ein Objekt ausgewählt wird der Wert der globalen Variable im entsprechenden Textfeld angezeigt. Dieser Wert soll bei der Multiselektion jedoch nur angezeigt werden, wenn er für alle ausgewählten Objekte identisch ist. Da ansonsten ein Strich sichtbar sein soll, ist es hier notwendig die Properties der ausgewählten Objekte zu vergleichen. Hierfür muss zuerst abgefragt werden welche Objekte ausgewählt sind, um dann deren Properties zu vergleichen. Da AXURE bei den logischen Bedingungen nur „Match Any“ oder „Match All“ unterstützt ist keine Kombination von AND und OR Bedingungen möglich. Deshalb wird an diesem Punkt nur mit AND Bedingungen in den Abfragen gearbeitet, da jedoch für alle möglichen Kombinationen der drei zur Verfügung stehende Bildern jede Property einzeln abgefragt und angepasst werden muss, ergibt das eine Anzahl von 36 Abfragen.

CASE NAME									
Phone/Nav/Mess_x									
Match All									
is selected of	Phone_Image	equals	value	true					
is selected of	Nav_Image	equals	value	true					
is selected of	Message_Image	equals	value	true					
value of variable	STRG_Down	equals	value	true					f <sub>x</sub>
value of variable	x_Phone	equals	value	[[x_Navigation]]					f <sub>x</sub>
value of variable	x_Phone	equals	value	[[x_Message]]					f <sub>x</sub>

Abbildung 4.10.: Beispielbedingung Multiselektion

Auf identische Art und Weise wurde diese Funktionalität für die verfügbaren Labels umgesetzt. Aufgrund der Performance des Prototyps und der Fehleranfälligkeit wurde auf diese Anpassung bei der gleichzeitigen Auswahl von Texten und Bildern verzichtet. Es ist hier möglich die Properties anzupassen, es wird jedoch nicht verglichen, ob deren Variablen den gleichen Wert aufweisen, stattdessen wird immer ein Strich angezeigt. In Abb. 4.10 ist die logische Bedingung für den Fall zu sehen, dass alle drei Bilder gleichzeitig ausgewählt sind und deren x-Koordinate jeweils identisch ist.

Zeitgleich mit den Properties werden bei der Mehrfachselektion die Alignment Actions angezeigt. Hier wird im Prototypen die Umsetzung in Bezug auf den Testcase eingeschränkt. Da ein Startscreen modelliert werden soll, reicht die Möglichkeit die Bilder Horizontal aneinander auszurichten, die Vertikale Ausrichtung ist von Texten an Bildern möglich. In Abb. 4.11 ist die Umsetzung im Prototyp zu sehen, wobei Teil a) bei zwei ausgewählten Bildern und Teil b) bei der gleichzeitigen Auswahl von Bild und Label sichtbar ist. Die Alignment Actions liegen bei der Auswahl von zwei Objekten des gleichen Typs unten, und bei der Auswahl von zwei unterschiedlichen oben. Bei letzterem Fall kann eher davon ausgegangen werden, dass keine Properties gemeinsam angepasst werden müssen, sondern die zwei unterschiedlichen Objekte eher aneinander ausgerichtet werden sollen. Die tatsächliche Ausrichtung wird durch Klick auf den Button ausgelöst, indem der eingegebene Abstand auf den Variablenwert des weiter links oder weiter oben liegenden Objektes aufaddiert wird und der globalen Variable des zweiten Items zugewiesen wird. Dadurch verschiebt sich das zweite Objekt und der Abstand der zwei Objekte entspricht dem eingegebenem Wert.

**Templates** Das Anlegen von Templates wird ebenfalls mit Screenshots simuliert die mit Hotspots versehen werden. Zusätzlich besteht hier noch die Notwendigkeit einen Imagetab neben dem View anzulegen, da das Erstellen der Templates in einer anderen View stattfindet. In diesem Tab existiert ebenfalls ein Property Panel, die Funktion „publish to template interface“ geschieht wie geplant über einen Klick auf den Kreis, der sich nach der Auswahl blau einfärbt. Wird hier ein Wert angepasst wirkt sich das über eine globale Variable auch auf das Template Interface aus. Eine Änderung im Template Interface darf jedoch nie ein Property des Templates anpassen. Deshalb wurden hier zwei getrennte Sets von globalen Variablen angelegt, welche nur in eine Richtung synchronisieren.

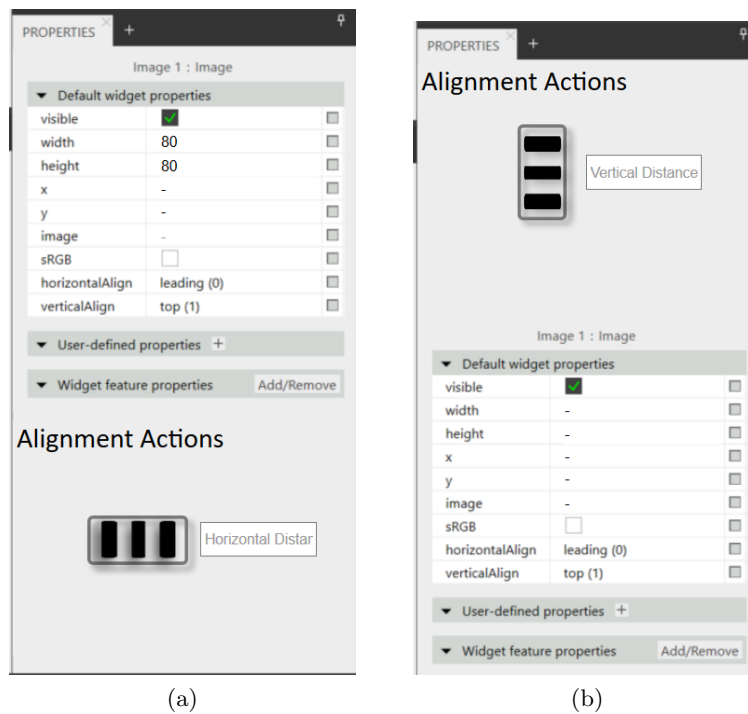


Abbildung 4.11.: Alignment Actions

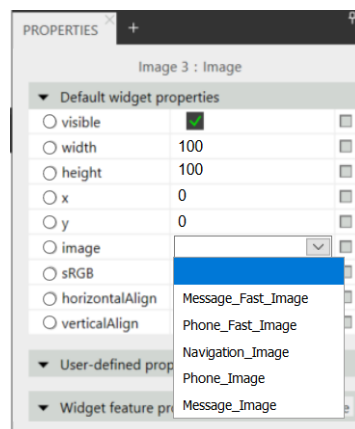


Abbildung 4.12.: Droplist für Images

Die Zuweisung der Bilder im Template findet über ein Droplist statt, welche in Abb. 4.12 zu sehen ist. Die Vorgehensweise, wie der sichtbare Inhalt eines Dynamic Panels mithilfe einer Droplist angepasst werden kann, wurde bereits in Abschnitt 4.1.1 erklärt. Statt der bunten Kreise befindet sich in diesem Fall jeweils eine Ausführung aller Bilder in den Dynamic Panels welche je nach Auswahl aus der Droplist sichtbar oder unsichtbar geschaltet werden können. Die alternative Vorgehensweise „Insert in Template“ funktioniert nach dem selben Prinzip. Nur wird hier nicht aus der Droplist abgefragt, sondern welches Bild aktuell im Template ausgewählt ist. Dieses wird dann, nach einem Klick auf den Button entsprechend angezeigt.



## 4.2. Implementierung Filter

Nach Fertigstellung des Prototyps ist es nun nötig den Filter zu implementieren. Da hier eine direkte Umsetzung im Sourcecode von EB GUIDE geplant ist, gibt es im Folgenden, nach der Formulierung der Zielsetzung, einen groben Überblick über den Aufbau des Projektes. Wie bei der Erstellung des Prototypen im vorherigen Kapitel wird abschließend noch aufgezeigt mit welchem Vorgehen die formulierten Ziele umgesetzt werden.

### 4.2.1. Zielsetzung

Das Ziel dieses Arbeitsabschnittes ist die Umsetzung des in Abschnitt 3.1.3 definierten optischen und funktionalen Designs. Da es sich um eine Implementierung handelt ist hier keine Einarbeitung in ein Tool wie AXURE RP notwendig, vielmehr muss der Aufbau des bereits bestehenden Projektes verstanden werden. Konkret muss für die Umsetzung des Designs eine Filterfunktion implementiert werden, die die bereits vorhandenen Widget Feature Properties nach den Wünschen des Nutzers filtert. Hierfür wird für den Nutzer eine Eingabemöglichkeit benötigt, mit der er interagieren kann, wie er es von den Filterfunktionen anderer Anwendungen gewohnt ist. Zusätzlich sollen bei der Nutzereingabe die aktuell bestehenden Kategorien verschwinden und in Echtzeit nur die passenden Features ohne zugehörige Kategorie angezeigt werden.

### 4.2.2. Projektaufbau

**Programmiersprachen und Entwicklungsumgebung** EB GUIDE Studio wird als WPF Anwendung umgesetzt und ist somit in C# geschrieben. WPF ist Teil des .NET-Frameworks 3.0, auf dessen Basis dem Entwickler eine große objektorientierte Klassenbibliothek zur Verfügung steht. Innerhalb dieser Klassenbibliothek ist auch C# zu finden, welche eine typsichere, objektorientierte Allzweck-Programmiersprache ist. Für die Deklaration der Elemente des User Interfaces benutzt WPF die Beschreibungssprache XAML, welche auf XML basiert.

Da das Projekt firmenintern als eine Visual Studio Solution vorliegt, werden auch sämtliche Implementierungen im Rahmen dieser Arbeit innerhalb dieser Entwicklungsumgebung umgesetzt.

**Design Pattern** In der Projektentwicklung wird das Model-View-ViewModel (MVVM) Design Pattern angewandt. Bei früheren Entwicklungen von User Interfaces, haben Entwickler häufig eine grafische View erstellt und im Nachhinein die Logik hinzugefügt. Dies geschieht meist in Form von Eventhandlern, Initialisierungen oder Datenmodellen die im

so genannten Code Behind liegen, über welchen die Logik der Anwendung gesteuert wird. Diese Entwicklungsmethode führt zu einer starken gegenseitigen Abhängigkeit von Interface und der dahinter liegenden Logik. Dadurch entstehen Problematiken wie die Tatsache, dass nicht mehrere Entwickler gleichzeitig an der gleichen View arbeiten können, oder gegenseitig Code unbrauchbar gemacht wird. Logik und Aussehen eines Interface an einer Stelle zu bündeln führt also zu schlechter Wartbarkeit, Erweiterbarkeit und ist kaum testbar[[Unde](#)]. Die aufgeführten Probleme entstehen durch die starke Abhängigkeit der folgenden Komponenten.

- View (User Interface)
- Model (Im User Interface angezeigte Daten)
- Zusammenfügender Code (Eventhandling, Abhängigkeiten, Logik)

Innerhalb des MVVM Pattern wird dieser zusammenfügende Code als das View Model bezeichnet. Dessen grundlegende Aufgabe ist es, eine Trennung der View und des Models zu arrangieren und dadurch die Erstellung der Struktur und die Wartung der Anwendung zu vereinfachen. Sobald sich ein Wert innerhalb des View Models ändert, wird die Anpassung, mithilfe von Data Binding und Benachrichtigungen, automatisch im View aktualisiert. Wenn der Nutzer hingegen mit dem View interagiert, zum Beispiel einen Button drückt, wird eine, sich im View Model befindende Command ausgelöst, die die gewollte Aktion ausführt. Während dieses Prozesses modifiziert das View Model die Daten des Models, die View selbst führt nie eine direkte Änderung auf dem Model aus. Tatsächlich weiß die View nicht von der Existenz der Model Klasse, während das Model nicht weiß das die View existiert.

Das Pattern umfasst also die drei Hauptbestandteile:

- View (User Interface)
- Model (Datenzugriffe, Modelklassen)
- ViewModel (Vermittler zwischen View und Model)

Wie in Abb. 4.13 zu sehen, fungiert das ViewModel als Schnittstelle zwischen dem Model und der View. Es stellt ein Data Binding zwischen den beiden Instanzen bereit und verarbeitet über Commands alle Eingaben des Nutzers. Die View bindet ihre Kontrollwerte an Properties des ViewModels, welche im Gegenzug die Daten der Modelobjekte bereitstellen.



Abbildung 4.13.: Model-View-ViewModel (MVVM) Design Pattern

Im aktuellen Projekt werden alle benötigten Models, meist in Form von Interfaces, in die ViewModel Klasse `WidgetFeaturesOverlayViewModel` importiert. Aufgrund des Datenschutzes ist es nicht möglich, die genauen Models und die vollständige Implementierung der Klassen in dieser Arbeit offenzulegen. Die in den folgenden Kapiteln erläuterten Ergänzungen, werden in die soeben erwähnte ViewModel Klassen und der dazugehörigen View `WidgetFeaturesOverlayViewModel.xaml` eingearbeitet.

**CollectionViewSource** Die zu filternden Elemente liegen innerhalb des Projektes bereits in Form einer `CollectionViewSource` vor. Diese besitzt die Eigenschaften `View` und `Source`, welche jeweils unterschiedliche Varianten einer Collection beinhalten können. Die View kann man hier als eine, über der Source liegenden Ebene verstehen, mit deren Hilfe die Darstellung der Collection manipuliert werden kann. Das geschieht beispielsweise durch sortieren, filtern und gruppieren von Elementen, ohne jedoch die darunter liegende Source an sich zu verändern. Falls die Source das Interface `INotifyCollectionChanged` implementiert, werden alle Änderungen, die durch das `CollectionChanged` event entstehen, an die View weitergeleitet.[\[bota\]](#)

Im konkreten Fall des vorliegenden Projektes besteht die Source aus einer `ObservableCollection`, welche die Feature Models des aktuell ausgewählten Widgets beinhaltet. Eine `ObservableCollection` ist eine Collection dynamischer Daten, welche Benachrichtigungen für das Hinzufügen, Entfernen, oder Neuladen der Liste bereit stellt. [\[botc\]](#) Aktuell wird die `CollectionViewSource` gruppiert und sortiert, wodurch die Einteilung in Kategorien und die alphabetische Sortierung in der View erzeugt wird.

#### 4.2.3. Vorgehensweise

Zusätzlich zur Gruppierung und Sortierung stellt die `CollectionViewSource` ein Filter Event bereit. Diese Filter können mithilfe der View auf die Collection angewendet werden. Das bedeutet konkret, dass ein Item zwar in der Collection Source existiert, mithilfe des Filterevents jedoch nur ein ausgewählter Teil dieser Collection in der View angezeigt wird.[\[botb\]](#) Das Event kann durch Setzen eines EventHandlers genutzt werden, welcher die gewünschte Filterlogik bereitstellen kann. Dieser EventHandler ist in Auflistung [4.1](#), in den Zeilen 22 bis 23 zu sehen, die dazugehörige Filterlogik in Auflistung [4.2](#).

```

1  public ICollectionView AvailableWidgetFeatures
2  {
3      get
4      {
5          if (_availableWidgetFeatures == null ||
6              _filtertext != null)
7          {
8              var viewSource = new CollectionViewSource();
9              if (string.IsNullOrEmpty(FilterText))
10             {
11                 viewSource.GroupDescriptions
12                     .Add(new PropertyGroupDescription("Category"));
13             }
14
15             viewSource.SortDescriptions
16                 .Add(new SortDescription
17                     ("Category", ListSortDirection.Ascending));
18             viewSource.SortDescriptions
19                 .Add(new SortDescription
20                     ("Name", ListSortDirection.Ascending));
21             viewSource.Source = WidgetFeatures;
22
23             // FilterSource is local Delegate
24             viewSource
25                 .Filter += new FilterEventHandler(FilterSource);
26             viewSource.View.Refresh();
27             _availableWidgetFeatures = viewSource.View;
28         }
29         return _availableWidgetFeatures;
30     }
31 }

```

Auflistung 4.1: CollectionViewSource()

Innerhalb der Filterfunktion muss zuerst abgefragt werden, ob der Filtertext leer ist oder Text beinhaltet. Falls dies der Fall ist, ist es nicht nötig die Features zu filtern, weshalb alle verfügbaren mithilfe von `Accepted` zurückgegeben werden. Sollte der Filtertext jedoch initialisiert worden sein, wird jedes Item innerhalb des Events als `FeatureViewModel` gespeichert. `FeatureViewModels` enthalten alle relevanten Informationen über die Features. Dazu zählt auch deren Namen, weshalb an dieser Stelle überprüft werden kann, ob die Eingabe

des Nutzers einem Featurenamen entspricht oder in diesem enthalten ist. Sollte dies der Fall sein, wird das Feature der gefilterten View hinzugefügt, anderenfalls wird das Feature durch den Filter entfernt.

```
1 private void FilterSource(object sender, FilterEventArgs e)
2 {
3     if (string.IsNullOrEmpty(FilterText))
4     {
5         e.Accepted = true;
6         return;
7     }
8
9     var widgetFeature = e.Item as FeatureViewModel;
10
11     if
12     (widgetFeature.Name.ToUpper().Contains(_filtertext.ToUpper()))
13     {
14         e.Accepted = true;
15     }
16     else
17     {
18         e.Accepted = false;
19     }
20 }
```

Auflistung 4.2: Widget Properties Filter

Die Grundstruktur des in Auflistung 4.1 zu sehenden ICollectionView war im Projekt bereits vorhanden. Zusätzlich zu dem eben erwähnten Filter ist noch die If Abfrage in Zeile 9 ergänzt worden. Diese dient dazu die ausklappbaren Kategorien auszublenden, sobald ein Filterbegriff eingegeben wird. Die zweite Bedingung der If Abfrage in den Zeilen 5 und 6 ist ebenfalls neu hinzugefügt worden, um die Filterung in Echtzeit zu ermöglichen. Bis jetzt war der erste Teil der Abfrage ausreichend, da die Liste nur einmal bei der Anzeige des entsprechenden Panels aktualisiert und angelegt werden musste, nach der Implementierung der Filterfunktion ist dies jedoch bei jeder Änderung des Filterbegriffes notwendig. Aus diesem Grund wird nun auch abgefragt ob der Filtertext aktuell Text enthält, damit bei jedem ChangeEvent die Liste neu gefiltert wird.

Ausgelöst wird dieses ChangeEvent über den in Auflistung 4.3 zu sehenden Setter des Filtertextes. Sobald eine Änderung im Textfeld registriert wird, schickt die View ein Update an das ViewModel, wodurch der Filtertext angepasst wird. Zusätzlich dazu wird dem Availa-

bleWidgetFeatures in Auflistung 4.1 über das OnPropertyChanged Event mitgeteilt, dass die View aktualisiert werden muss.

```

1 public string FilterText
2 {
3     get => _filtertext;
4     set
5     {
6         _filtertext = value;
7         OnPropertyChanged( " AvailableWidgetFeatures " );
8     }
9 }

```

Auflistung 4.3: Filtertext Properties

Das Update vonseiten der View wird aufgrund des, in Auflistung 4.4 in Zeile 5 bis 6 zu sehenden, UpdateSourceTriggers ausgelöst. Hier wird über die Angabe des Paths festgelegt, dass bei jeder Änderung der Text Property die Properties des FilterTextes aktualisiert werden. Zusätzlich wird hier ein Delay von 100 ms eingeführt, um die Filterung in Echtzeit zu ermöglichen. Dieses Zeitintervall führt dazu, dass der Nutzer auch bei einer schnellen Eingabe nicht durch eine verzögerte Aktualisierung in seinem Arbeitsablauf unterbrochen oder verlangsamt wird.

```

1 <DockPanel Grid.Row=" 2 ">
2     <guide:SearchBox
3         Name=" AvailableFeaturesFilter "
4         HintText=" Filter Widget features "
5         Text="{ Binding Path=FilterText ,
6             UpdateSourceTrigger=PropertyChanged , Delay=100} " />
7 </DockPanel>

```

Auflistung 4.4: Filterpanel in xaml

## Kapitel 5.

### Usability Test

Aufbauend auf allen bisher getätigten Analysen und Anpassungen des Interface, gilt es abschließend noch den Prototypen und die Implementierung mithilfe eines Usability Tests zu evaluieren. Hierbei wird im Schritt „Finalize the UX design“ des Human-Centered Design Process die Rolle des Usability Testers abgedeckt, welcher für die Durchführung der Auswertung des Usability Tests zuständig ist. Auf weitere Anpassungen des Interface wird verzichtet.

In diesem abschließenden Kapitel werden die Grundlagen des verwendeten Tools für den Usability Test erläutert. Darauf folgend wird die Art des durchgeführten Tests definiert und dargelegt auf welchen Grundlagen diese Wahl getätigt wurde. Aufbauend auf den theoretischen Erläuterungen wird die letztendliche Testaufgabe konkret definiert und die nötigen Anweisungen, sowie die Materialien zur Auswertung der Tests erstellt. Abschließend werden die Ergebnisse der durchgeführten Auswertungen der Tests dargestellt und interpretiert, bevor es einen Ausblick auf das mögliche weitere Vorgehen gibt.

#### 5.1. Lookback

Bei Lookback handelt es sich um eine Cloudbasierte Softwarelösung, die es ermöglicht User Experience geräteübergreifend zu dokumentieren. Bei der Durchführung der Tests besteht die Möglichkeit, als Tester aktiv teilzunehmen oder die Probanden unmoderiert mit dem Prototyp oder der Software interagieren zu lassen. Möchte der Usability Tester teilnehmen - also einen „In Person“ Test durchführen - kann aus den beiden Möglichkeiten gewählt werden, die Tests mit dem Nutzer innerhalb einer Live-Session oder persönlich vor Ort an einem Rechner abzuhalten.

Für die durchzuführenden Tests erstellt der Usability Tester innerhalb von Lookback ein Projekt. Zugunsten des Datenschutzes besteht hier die Möglichkeit die Projekte auf privat zu setzen und nur ausgewählten Personen des Teams Zugriff zu den Aufnahmen der Tests zu

gewähren. Innerhalb der Projekteinstellungen können Instruktionen für die Nutzer bereitgestellt werden, die während des Tests erscheinen und durch die Aufgaben führen. Das ist vor allem dann hilfreich, wenn die Testperson den Test autonom durchführen soll.

Sollte der Test nicht gemeinsam in einem Raum stattfinden, kann mithilfe eines Links der Test mit den Probanden geteilt werden. Der Tester bekommt eine Benachrichtigung, sobald der Proband bereit ist bzw. mit der autonomen Durchführung begonnen hat. Während der Session hat der Tester die Möglichkeit mit Zeitstempel versehene Notizen zu machen oder mit eventuell teilnehmenden Teamkollegen zu chatten. Diese Möglichkeit der Livedokumentation erleichtert die darauffolgende Auswertung des Test erheblich, da man sofort Auffälligkeiten festhalten kann und diese in der Aufzeichnung leichter auffindbar sind. Nach Abschluss des Tests speichert Looback die Aufnahme in der Cloud, wodurch diese für die nachträgliche Auswertung des Tests abrufbar bleibt. [All 20]

Die Möglichkeit der Livedokumentation bildet das ausschlaggebende Argument für die Wahl des Tools, ebenfalls besteht die Möglichkeit in der hochgeladenen Aufnahme Kommentare hinzuzufügen, was die Auswertung sehr erleichtert. Zusätzlich ist es wichtig den Test online durchführen zu können, da viele Modellierer nicht am Hauptstandort von EB beschäftigt sind, wodurch persönliches Testen nicht immer möglich ist.

## 5.2. Remote Usability Test

Die im vorherigen Kapitel bereits kurz benannten „In Person“ Tests finden in der Regel in einem Usability Labor statt. Dies bezeichnet einen abgeschlossenen Raum ohne Störungsquelle, der mit der benötigten Hard- und Software ausgestattet ist und in dem sich nur der Proband und der durchführende Tester aufhalten.

Bei einem Remote Usability Test wird die Arbeitsaufgabe nicht gemeinsam im Labor, sondern räumlich getrennt durchgeführt. Hier lässt sich noch zwischen asynchronen und synchronen Tests unterscheiden. Bei letzteren besteht, trotz der räumlichen Trennung, eine direkte Verbindung mithilfe von Webcam und Sprachübertragung, zwischen Tester und Proband. Gleichzeitig dazu wird der Bildschirminhalt der Testperson übertragen, um es dem Tester zu ermöglichen dessen Interaktionen zu verfolgen und durch die Aufgaben führen zu können. Durch diese direkte Verbindung ist es für den Tester ebenfalls möglich, während, oder unmittelbar nach dem Test, Fragen zu stellen. Bei synchronen Tests besteht der Vorteil darin, mit geringem Aufwand örtlich weit verteilte Nutzer in den Test einbinden zu können. Zusätzlich dazu findet der Test in der gewohnten Umgebung des Nutzers statt, es wird also keine unnatürliche Situation in einem Labor geschaffen, was die Nervosität der Nutzer eventuell steigern könnte. Auch können die Probanden hier mit ihrer gewohnten Hard- und Software arbeiten, was vor allem für repräsentative Werte, die die Effizienz betreffen, von



Vorteil ist. Durch den Umstand der gewohnten Arbeitsumgebung entsteht jedoch in vielen Fällen auch ein technischer Zusatzaufwand aufseiten des Nutzers. So kann beispielsweise ein Headset und eine Webcam benötigt, oder zusätzliche Software auf dem Arbeitsgerät installieren werden müssen. Dies ist in einem Usability Labor nicht notwendig, da hier eine einmalige Einrichtung der benötigten Hard- und Software stattfindet, die exakt auf den geplanten Test ausgelegt ist.

Im Rahmen eines asynchronen Tests besteht zusätzlich zu der räumlichen, auch noch eine zeitliche Trennung von Proband und Tester. Identisch zum synchronen Test werden hier Gesicht und Bildschirminhalt der Testperson aufgezeichnet, der Nutzer kann jedoch aufgrund der Unabhängigkeit den Test zu einer, ihm passenden Zeit durchführen. Da er dadurch jedoch auch auf sich allein gestellt ist, empfiehlt es sich zusätzliche Kommentare und Fragebögen in den Test einzubauen, um die direkte Befragung und Hilfestellung des synchronen Tests zu ersetzen. Hier besteht ebenfalls der Vorteil, dass mit geringem Aufwand, großräumig verteilte Nutzergruppen eingebunden werden können. Zusätzlich können durch die vollautomatische Erfassung der Ergebnisse auch sehr große Nutzergruppen effizient evaluiert werden. Allerdings entstehen durch die zeitliche Trennung die Nachteile, dass - abgesehen von der Aufnahme - keine Beobachtungsdaten existieren. Es kann also beispielsweise nicht darum gebeten werden, etwas genauer zu erläutern oder einen anderen Lösungsweg zu testen. Die wohl größte Problematik bildet jedoch die Tatsache, dass nicht auf unerwartete Handlungen des Nutzers reagiert werden kann, was vor allem, bei nur teilweise funktionalen Prototypen, fatal sein kann. Versuchen die Probanden hier einen Lösungsweg einzuschlagen, der in der vorliegenden Softwareversion nicht implementiert ist und auch nicht bedacht wurde, erhält die Testperson eventuell keine Rückmeldung des Systems. Dies kann ohne weitere Unterstützung durch den Tester zum Abbruch des Test führen, was zu Unzufriedenheit aufseiten des Testers und Nutzers führen kann.[[Saro 16](#)]

Da Lookback jeden der eben erläuterten Tests unterstützt, wirkt sich das Tool nicht einschränkend aus und die Wahl kann aufgrund relevanter Gründe getroffen werden. Aufgrund der Tatsache, dass innerhalb der Testaufgabe mit einem Prototyp interagiert werden muss, ist ein synchroner Test dem asynchronen vorzuziehen. Da dieser nicht alle, in GUIDE zur Verfügung stehenden, Interaktionsmöglichkeiten simuliert und die Funktion „publish to template interface“ verlagert wurde, ist es wahrscheinlich, dass die Probanden an gewissen Punkten Unterstützung benötigen. Es ist ebenfalls zu erwarten, dass in Bezug auf die neuen Funktionen Fragen bei den Testpersonen auftauchen werden, die nicht in der Arbeitsaufgabe beantwortet werden können. Andererseits wird auch der Tester einige Aktionen genauer hinterfragen oder herausfinden wollen, warum gewisse Aktionen nicht durchgeführt wurden.

Grundsätzlich ist ein Test im Usability Labor, aufgrund der Ungestörtheit, immer dem Remote Test vorzuziehen. Zum einen erstreckt sich die Nutzergruppe für diese Arbeit über mehrere Firmenstandorte von Elektrobit, weshalb es nicht möglich ist den Test mit allen

Probanden in einem Labor durchzuführen. Zum anderen soll mithilfe der Tests überprüft werden, ob eine Effizienzsteigerung erzielt werden kann. Die Ausstattung in den Laboren entspricht nie hundertprozentig der gewohnten Umgebung der Nutzer, weshalb hier auch verminderte Leistungen, aufgrund der fremden Hardware zu erwarten sind. Es wäre möglich, die Tests mit einem Teil der Probanden im Labor und mit dem anderen Remote durchzuführen. Um die Ergebnisse jedoch so gut vergleichbar wie möglich zu halten, werden alle Untersuchungen unter den gleichen Bedingungen, in einem Remote Test, durchgeführt.

### 5.3. Arbeitsaufgaben

Um in der abschließenden Arbeitsaufgabe alle Änderungen testen zu können, ist es nötig sich an den ursprünglich analysierten Benutzeranforderungen zu orientieren, auf deren Grundlage die Änderungen entworfen und der Prototyp gebaut wurde. Daher werden zu den, in Kapitel 3.0.3. bereits formulierten, qualitativen Anforderungen nun die, im Rahmen eines Tests messbaren, quantitative Nutzeranforderungen, für die ausgewählten Anpassungen ergänzt.

#### **Quantitative Benutzeranforderung für Template Properties**

**Messung der Effizienz:** Nutzer, die die Funktion „publish to template interface“ über einen Linksklick auf den Kreis ausführen, sollen messbar schneller sein, als Nutzer, die dies über einen Rechtsklick auf das Quadrat tun.

**Messung der Fehlerrate:** Nutzer, die die Funktion „publish to template interface“ über einen Linksklick auf den Kreis ausführen, sollen eine niedrigere Fehlerrate aufweisen, als Nutzer, die dies über einen Rechtsklick auf das Quadrat tun.

#### **Quantitative Benutzeranforderung für Widget Feature Properties**

**Messung der Effizienz:** Nutzer, die die Filterfunktion nutzen, sollen schneller ein gewünschtes Widget Feature Property finden, als Nutzer die diese händisch suchen müssen.

**Messung der Fehlerrate:** Bei Nutzern, die die Filterfunktion nutzen, soll die Fehlerrate sich, bei der Suche nach dem gewünschten Widget Feature Property, auf 0 reduzieren.

#### **Quantitative Benutzeranforderung für Mehrfachselektion**

**Messung der Effizienz:** Nutzer, die mehrere Objekte gleichzeitig selektiert haben, sollen deren korrekte Positionierung und Skalierung schneller abgeschlossen haben, als Nutzer die keine Mehrfachselektion benutzen.

**Messung der Fehlerrate:** Nutzer, die Objekte mithilfe der Mehrfachselektion skalieren oder positionieren, sollen diesen Vorgang mit einer geringeren Fehlerrate abschließen, als Nutzer die keine Mehrfachselektion nutzen.

Zusätzlich wird hier darauf geachtet, inwiefern die Ergänzungen genutzt werden, vor allem die neu eingeführten „Alignment Actions“ und die Funktion „Insert in Template“ und welche Verbesserungen hier noch einzubringen sind.

Aus den aufgeführten Benutzeranforderungen lassen sich wiederum Subtask ableiten, anhand deren überprüft werden kann welche Interaktionen, vonseiten des Nutzers, in welcher Zeitspanne abgeschlossen werden, an welcher Stelle Fehler auftreten und welche Möglichkeiten nicht genutzt werden. Hierfür ist es notwendig, jede Interaktion in kleine Subtasks zu untergliedern, um die exakte Stelle feststellen zu können, an der die Interaktion mit dem Interface zu Fehlern führt. Bei der Filterung der Feature Widget Properties kann dies eventuell schon daran scheitern, dass nicht in das Texteingabefeld geklickt wird, oder erst durch die Eingabe eines falschen Filterbegriffes. Die formulierten Subtasks für den in dieser Arbeit durchgeführten Test, lassen sich in Anhang A nachvollziehen.

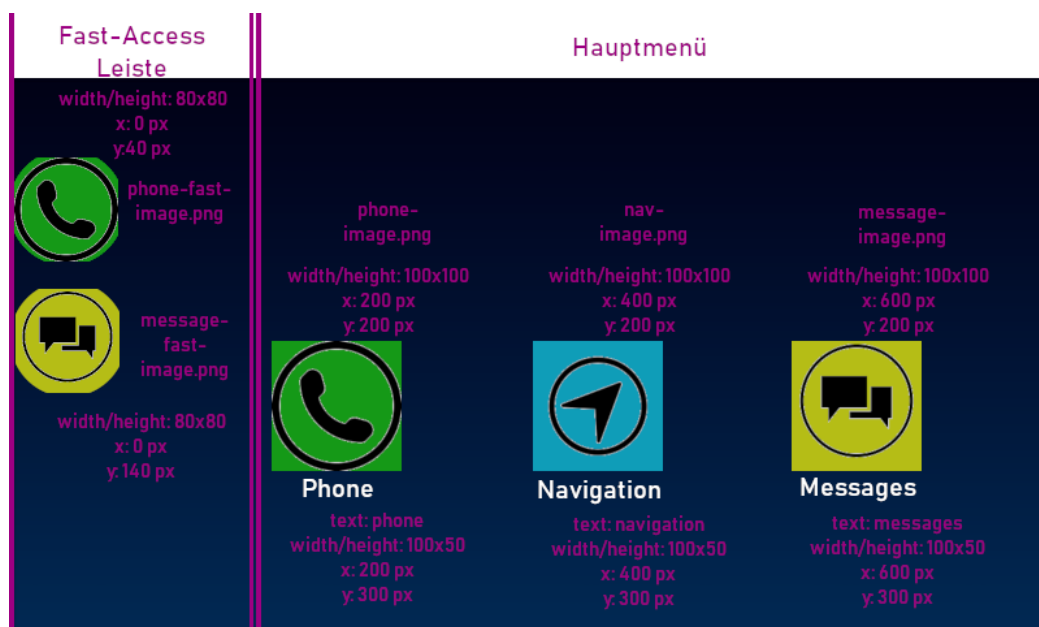


Abbildung 5.1.: Styleguide

Auf Grundlage dieser Subtasks, scheint es weiterhin sinnvoll, den Testpersonen die Modellierung eines Startscreens eines Human Machine Interface der Automobilbranche als Testaufgabe modellieren zu lassen. Hierfür ist es notwendig einen grafischen Styleguide zu erstellen an dem sich die Nutzer, wie aus ihrer täglichen Arbeit gewohnt, orientieren können. Darüber hinaus ist es auch, für die Evaluierung der Tests, notwendig die Nutzer die exakt gleichen Aufgaben durchführen zu lassen. Der, in Abb. 5.1 zu sehende, Styleguide zeigt einen Startscreen mit drei Bildern und Bildunterschriften im Hauptmenü und zwei Bildern in der Fast-Access-Leiste. Für jedes dieser Elemente sind, die für den Modellierer relevanten Eigenschaften sichtbar, sowie die Namen der Bilder, wie sie auch in den Assets von EB GUIDE verwendet werden. Diese Anmerkungen sind, innerhalb des Styleguides, alle

in pink gehalten, um sie deutlich vom tatsächlichen Design abzugrenzen. Wobei die Rahmen für das Hauptmenü und die Fast-Access Leiste ebenfalls der Orientierung dienen und zum Verständnis der, im Folgenden beschriebenen Arbeitsaufgabe notwendig sind.

Zusätzliche zu dieser grafischen Darstellung der Arbeitsaufgabe, wird noch eine textuelle Angabe erstellt, die für die Durchführung des Tests nötig ist. Es werden insgesamt drei verschiedene Angaben erstellt, die in den Anhängen **B**, **C** und **D** zu finden sind. Da ein Teil der Anpassungen implementiert und der andere Teil mithilfe eines Prototyps umgesetzt wurde ist es nötig den Test in zwei Teile aufzuspalten, weshalb auch zwei getrennte Angaben existieren. In Testaufgabe **B** gilt es den implementierten Filter zu testen, weshalb die Nutzer hier aufgefordert werden, die drei Bilder für das Hauptmenü zu platzieren und mit Widget Feature Properties zu versehen. Für diesen ersten Teil wird den Nutzern ein vorbereitetes Projekt mit den benötigten Assets zur Verfügung gestellt, um zu gewährleisten, dass Probanden mit einer identischen Ausgangssituation starten. Es wird hier explizit nicht auf die implementierten Änderungen hingewiesen, da überprüft werden soll, ob diese von den Nutzern intuitiv benutzt wird. Um jedoch zu gewährleisten, dass jeder Nutzer das Filterfeld theoretisch nutzen kann, wird darum geben die Bilder mit Widget Feature Properties zu versehen.

Für Aufgabe **C** ist es notwendig den Übergang in den Prototyp so nahtlos wie möglich zu gestalten und gleichzeitig wieder eine identische Ausgangssituation für alle Nutzer zu schaffen. Das initiale Aussehen des Prototyps beinhaltet deshalb die drei eingefügten Bilder im Hauptmenü, entspricht also der Situation, mit der das vorherige Projekt in GUIDE von den Testpersonen verlassen wurde. Da die Multiselektion eine Neuerung ist, die eher unwahrscheinlich selbstständig von den Nutzern entdeckt wird, wird hier zu Anfang darauf hingewiesen, dass der vorliegende Prototyp Multiselektion unterstützt. Mit der Fast-Access Leiste, welche mithilfe von Templates modelliert werden soll, wird die Verlagerung der Funktion „publish to template interface“ überprüft. Ebenfalls wird in diesem Zuge darauf hingewiesen, dass nun auch Bilder mithilfe von Multiselektion in Templates eingefügt werden können, jedoch ohne genaue Erklärung, auf welche Art und Weise das funktioniert. Abschließend sollen noch die Bilder und Labels positioniert und skaliert werden. Es wird hier nicht mehr explizit auf die Multiselektion verwiesen, da herausgefunden werden soll, inwiefern die Probanden dies, nach der Information zu Beginn der Angabe, intuitiv für die restliche Aufgabe nutzen.

Die letzte Aufgabe **D** dient zur Erlangung der Vergleichswerte der Effizienz und wird deshalb in der unmodifizierten Version von GUIDE umgesetzt. Da hier kein Übergang in den Prototyp notwendig ist, ist eine zusammenhängende Arbeitsaufgabe ausreichend. Die Probanden sollen hier die identische Aufgabe durchführen und werden, angepasst an die beiden vorherigen Angaben, in der Nutzung von Templates eingeschränkt oder dazu aufgefordert. Den Nutzern hier ein komplett freies Arbeiten zu gestatten ist nicht möglich, da möglichst

identische Arbeitsschritte getätigt werden müssen um einen validen Vergleich anstellen zu können.

## 5.4. Ergebnisse

Die Anzahl der - in einem mit  $n$  Nutzern durchgeführten Test - gefundenen Usabilityprobleme lässt sich laut Nielsen und Landauer nach folgender Formel berechnen.

$$N(1 - (1 - L)^n)$$

Wobei  $N$  der kompletten Anzahl der Usabilityprobleme im System entspricht und  $L$  der Anzahl der entdeckten Probleme nach dem Test mit einem Nutzer.  $L$  entspricht hier im Durchschnitt 31%, was zu der folgenden Grafik führt. [Niel 93]

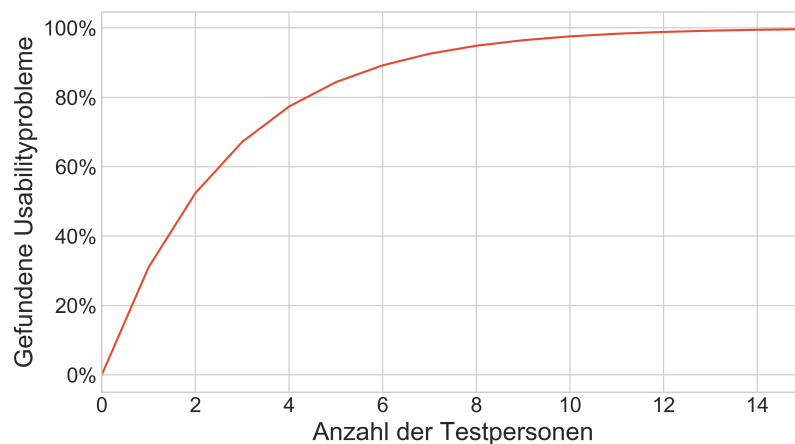


Abbildung 5.2.: Aufdeckungsrate von Usabilityproblemen

Bei Betrachtung von Abb. 5.2 wird deutlich, dass - solange mit keinem Nutzer getestet wurde - es auch keinen Aufschluss über die Usability Probleme eines Systems gibt. Nach dem Test mit einem Nutzer, werden bereits fast ein Drittel der Schwächen aufgedeckt. Wird der Test mit einem zweiten Nutzer durchgeführt, decken sich bereits einige der Beobachtungen, es werden aber durchaus noch neue Schwächen entdeckt und unterschiedliche Messergebnisse treten auf. Je mehr Nutzer untersucht werden, desto häufiger decken sich die Beobachtungen und Messungen mit den Ergebnissen der bereits untersuchten Nutzer.[Jako 00] Entgegen einer häufigen Annahme ist es also nicht unbedingt sinnvoll einen Usability Test mit einer möglichst großen Zahl an Testpersonen durchzuführen, da der Tester, ab einem gewissen Punkt, fast ausschließlich identische Probleme sehen und Messwerte erhalten wird. Neue Informationen erhält man meist bis zur Untersuchung des fünften Nutzers, weshalb dies auch die optimale Anzahl an Probanden darstellt.

Wird nun aber, wie im Falle dieser Arbeit, eines der messbaren Usability Attribute verglichen, ist es nötig die Anzahl der Nutzer zu verdoppeln. Es werden zwei verschiedene Versionen eines Systems verglichen, die jeweils von fünf Nutzer getestet werden. Dadurch werden bei jedem System etwa 80% bis 90% der Schwächen aufgedeckt. Bei der Messung der Attribute Effizienz und Fehlerrate kann ein repräsentativer Mittelwert gebildet werden.

Alternativ gäbe es die Möglichkeit den Test mit nur fünf Nutzern durchzuführen, was zu dem Vorteil führen würde, die Ergebnisse einer Person mit dem neuen und alten Interface direkt vergleichen zu können. Allerdings würde hier die Problematik entstehen, dass der Nutzer die gleiche Arbeitsaufgabe doppelt durchführen würde, was zu einer Verfälschung der Ergebnisse führt, da die Angaben und Koordinaten der Objekte bereits bekannt sind. Die Durchführung der identischen Arbeitsaufgabe ist jedoch dringend notwendig, um vergleichbare Werte zu erhalten. Aus diesem Grund bevorzugt Nielsen die Messung mit der doppelten Anzahl an Probanden, statt Nutzern zweimal die gleiche Aufgabe durchführen zu lassen.[\[Jako 00\]](#)

Die im Folgenden dargestellten Ergebnisse wurden durch das Testen einer Gruppe aus zehn Personen erzeugt, die aus sechs Expertennutzern und vier Gelegenheitsnutzern besteht. Zwar wäre eine 1:1 Zusammensetzung der Gruppe der Testpersonen wünschenswerter, firmenintern standen jedoch zum Zeitpunkt der Messung nicht mehr Personen zur Verfügung, weshalb es einen Expertennutzer mehr in der Testgruppe gibt. Die Einstufung eines Expertennutzers wird dadurch definiert, dass die Person aktuell aktiv in einem Projekt mit EB GUIDE 6 arbeitet und dies auch schon seit mindestens 2 Monaten ununterbrochen tut. Gelegenheitsnutzer sind all diejenigen, die zwischen dem Nutzen der Software immer mindestens 2 Wochen pausieren und eine einführende Schulung erfolgreich abgeschlossen haben.

Aus den gemessenen Werten wird jeweils ein Mittelwert gebildet, zusätzlich wird bei den Neuerungen untersucht welcher der möglichen Wege zur Problemlösung genutzt wird und welcher nicht. Über die Fehlerrate kann ergänzend aufgedeckt werden, ob noch Verbesserungen am Design vorgenommen werden müssen.

#### 5.4.1. Ergebnisse überarbeitetes Interface

Der Prototyp und der implementierte Filter werden demnach mit fünf Probanden getestet, welche sich aus drei Experten und zwei Gelegenheitsnutzern zusammensetzen. Nach der Durchführung der Aufgabe werden den Nutzern noch zusätzliche Fragen zu ihrem Verhalten gestellt, welche in der folgenden Auswertung ebenfalls aufgeführt werden.

**Widget Feature Properties** Der Filter wird im Test von drei Probanden genutzt, die anderen beiden, jeweils ein Experte und ein Gelegenheitsnutzer, geben anschließend an die Funktion nicht gesehen zu haben. Ein Nutzer wendet den Filter erst an, nachdem er zwei Menüs auf der Suche nach dem richtigen Feature ausgeklappt hat. Er und die anderen

Beiden geben anschließend an, den Filter nicht als Neuerung wahrgenommen zu haben, ihn jedoch intuitiv genutzt zu haben.

Die Auswahl eines Feature Properties dauert mithilfe des Filters durchschnittlich 2 Sekunden. Zwei der Nutzer erzeugen eine Fehlerrate von 2 Fehlern bei der Auswahl. Aus der Suche nach dem Scale Mode filtern die Probanden mit dem Wort „Scal“. Dadurch taucht, wie in Abb. 5.3 zu sehen, in der Ergebnissen zusätzlich zu dem Scale Mode das Feature Scaling auf, welches dann fälschlicherweise ausgewählt wird.

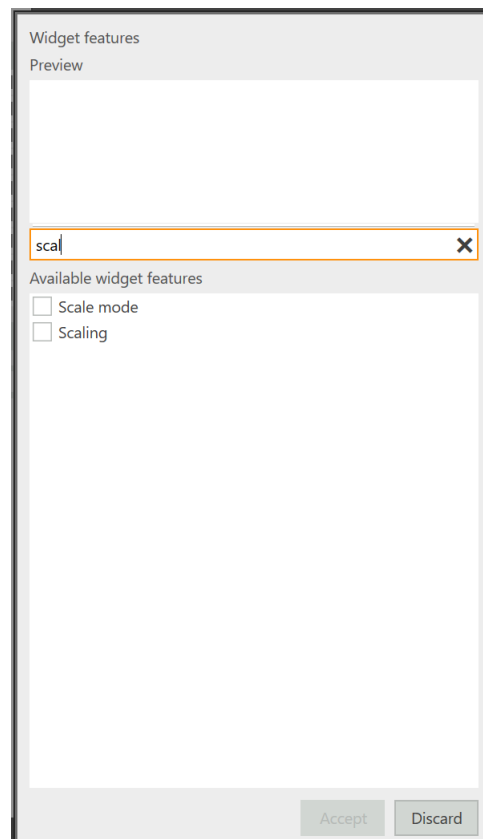


Abbildung 5.3.: Filterung nach Scale Mode

**Publish to template interface** Die Verlegung des Hotspots für die Funktion „publish to template interface“ wird ebenfalls von drei, der fünf Probanden selbstständig gefunden, die alle benötigten Properties dadurch in durchschnittlich 4,6 sek publishen können. Die neue Position wird von einem Experten und einem Gelegenheitsnutzer nicht entdeckt, jedoch - nach Aufklärung vonseiten des Testers - als nützlicher Shortcut eingestuft. Bei den nötigen Klicks für die das Publishen der Objekteigenschaften wird eine Fehlerrate von 0 erzielt.

**Multiselektion** Für die Skalierung der drei Bilder des Hauptmenüs nutzen alle Probanden die Multiselektion und schließen den Vorgang in durchschnittlich 7,2 Sekunden ab, wobei ihnen keine Fehler unterlaufen.

Bei den Labels haben nur vier Testpersonen die Multiselektion in durchschnittlich 8 Sekunden mit einer Fehlerrate von 0 genutzt, der fünfte Nutzer hat, nach eigenen Aussagen, aus Gewohnheit jeden Text einzeln positioniert, obwohl vorher für die Bilder die Mehrfachselektion verwendet wurde.

Bei der Positionierung muss unterschieden werden, ob die Arbeitsaufgabe mithilfe der Alignment Actions oder mit einfacher Multiselektion gelöst wurde.

Ein Proband macht bei den Bildern Gebrauch von den Alignment Actions und schließt die Positionierung aller drei Bilder in 5 Sekunden ab. Allerdings lässt sich hier eine Fehlerquelle beobachten. Der Nutzer legt zuerst den Abstand zwischen den Objekten fest, vergisst jedoch das Ankerbild, an dem sich die anderen Bilder ausrichten, an die richtige Position zu setzen. Nachdem dieses korrekt positioniert ist muss die Ausrichtung erneut durchgeführt werden. Drei der anderen Nutzer haben die Alignment Actions bei den Bildern nicht gesehen, ein Vierter hat sie zwar wahrgenommen, hatte zu diesem Zeitpunkt die Bilder jedoch einzeln schon korrekt positioniert, dass er von der Nutzung abgesehen hat.

Bei den Labels nutzt dieser Proband die Alignment Actions, ebenso die Testperson, die diese Funktion auch bei den Bildern angewandt hat. Dadurch kann die Positionierung hier in durchschnittlich 6 Sekunden abgeschlossen werden, es tritt jedoch ein identischer Fehler mit dem Ankerobjekt auf, was hier ebenfalls eine Fehlerrate von 1 ergibt. Die Probanden die diese Funktion nicht nutzen, geben hierfür die gleichen Gründe an wie bei den Bildern.

Zusätzlich gibt es bei den Texten die Möglichkeit diese mithilfe der Alignment Actions an den Bildern auszurichten. Auf die Idee ein Bild und den dazugehörigen Text gleichzeitig auszuwählen kommt jedoch keine der Testpersonen. Dies ist darauf zurückzuführen, dass sich erst noch an die neuen Funktionen gewöhnt werden muss und die gleichzeitige Auswahl verschiedener Objekte zum aktuellen Zeitpunkt noch zu verwirrend ist.

Die Positionierung über normale Multiselektion wird bei den Bildern von vier Nutzern durchgeführt, wobei die identischen y-Koordinaten zusammen angepasst werden und danach die x-Koordinate für jedes Bild einzeln. Mit diesem Vorgehen ist die Positionierung in durchschnittlich 3,5 Sekunden abgeschlossen und es werden zusätzlich keine Fehler gemessen.

Bei den Texten ist das Vorgehen für die Positionierung identisch, es treten ebenfalls keine Fehler auf und die durchschnittliche Zeitspanne beträgt 4,6 Sekunden.

Die Funktion „Insert in Template“ wird von keinem der Nutzer entdeckt. Bei der abschließenden Befragung geben auch alle Probanden an, dass sie diese Funktion als nicht sinnvoll erachten.



### 5.4.2. Ergebnisse altes Interface

Um die Vergleichswerte zu erhalten wurde die Testaufgabe mit dem bestehenden Interface ebenfalls mit fünf Personen durchgeführt. Die Zusammensetzung aus drei Experten und zwei Gelegenheitsnutzern ist hier identisch zu der Testgruppe des Prototypen.

**Widget Feature Properties** Da hier keine Filterfunktion zur Verfügung steht treten, wie zu erwarten, mehr Fehler bei den Nutzern auf. Von den fünf Probanden begehen zwei Nutzer 2 Fehler und ein Nutzer 3 Fehler. Die restlichen beiden Testpersonen finden das gewünschte Feature auf Anhieb, müssen hier jedoch einige Sekunden überlegen, weshalb bis zu Wahl des gewünschten Properties durchschnittlich 8,8 Sekunden vergehen.

**Publish to template interface** Die Funktion „publish to template interface“ wird in der aktuellen Version von den Probanden in durchschnittlich 16 Sekunden abgeschlossen. Eine Person erzeugt hier eine Fehlerrate von 1, indem sie nach dem Rechtsklick auf das Rechteck hinter dem zu verlinkenden Property zuerst die falsche Funktion auswählt. Bei allen anderen Probanden lässt sich ebenfalls beobachten, dass sie kurz überlegen, welche der in Abb. 5.4 zu sehenden Alternativen ausgewählt werden muss.

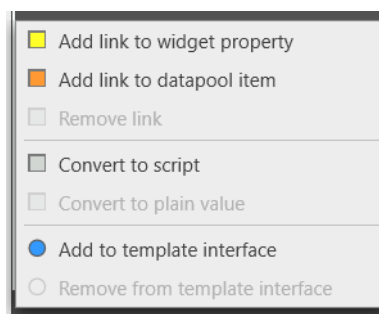


Abbildung 5.4.: Auswahlmöglichkeiten nach Rechtsklick auf Templateproperties

**Positionieren/Skalieren ohne Multiselektion** Das Skalieren und Positionieren der Texte und Bilder wird in der aktuellen Version von GUIDE von den Nutzern meist so gelöst, dass ein Bild mit allen nötigen Features ausgestattet und dieses dann kopiert wird. Anschließend ist es noch nötig die unterschiedlichen Eigenschaften der Kopien entsprechend der Spezifikation anzupassen.

Da bei der Skalierung die Werte nach dem Kopieren nicht mehr verändert werden müssen, kommen die Probanden hier auf eine durchschnittliche Arbeitszeit von 4,8 Sekunden, in der der Kopiervorgang bereits einberechnet ist. Fehler treten hier nur auf, wenn die Probanden die Werte nicht händisch eingeben, sondern mit dem etwas komplizierteren Ansatz der Data-poolitems arbeiten. Hierbei werden die Eigenschaften, wie Breite und Höhe mit einem Item

verlinkt, dessen Wert sich an einer Stelle zentral anpassen lässt und alle verlinkten Objekte entsprechend verändert. Fehler treten hierbei durch Verlinkung des falschen Datapoolitems oder der Eingabe des falschen Wertes auf. Die Anzahl bemisst sich in diesem Testfall auf 2 Fehler bei einem Probanden, der diesen etwas komplizierteren Modellierungsansatz wählt.

Bei der Skalierung der Labels arbeitet keiner der Nutzer mit Datapoolitems, weshalb hier eine Fehlerrate von 0 erzeugt wird und der Vorgang für alle Texte in durchschnittlich 9 Sekunden abgeschlossen ist.

Für die Positionierung der Bilder und Labels müssen die Eigenschaften der kopierten Elemente angepasst werden. Dies geschieht für erstere in 8,5 Sekunden, bei den Labels dauert es durchschnittlich 10,8 Sekunden. Hierbei tritt bei den Bildern ebenfalls ein Fehler bei der Arbeit mit den Datapoolitems auf, bei den Texten beläuft sich die Fehlerrate bei allen Nutzern auf 0.

### 5.4.3. Vergleich und Interpretation

Auf Grundlage der gemessenen Daten kann nun ein direkter Vergleich zwischen dem bestehenden und dem überarbeiteten Interface von EB GUIDE getätigt werden. Weiterhin können Beobachtungen über die Nutzung der Ergänzungen interpretiert werden und anhand der Messwerte evaluiert werden, ob die definierten quantitativen Benutzeranforderungen erfüllt wurden.

**Widget Feature Properties** Mithilfe des Filters gelingt es den Probanden 8,8 Sekunden schneller das gewünschte Feature Property zu finden, wodurch die Anforderung an die Effizienz erfüllt ist. Die angestrebte Fehlerrate von 0 kann jedoch aktuell noch nicht erzielt werden. Die Fehler können sich vermutlich darauf zurückführen lassen, dass die Nutzer an die übergeordneten Kategorien gewöhnt sind und sich deshalb nicht sicher sind, ob sie Scaling oder Scale Mode auswählen müssen. Hier bestünde die Möglichkeit in einer weiteren Iteration zu testen ob sich die Fehleranzahl wieder verringert, wenn die Kategorien während des Filterns eingeblendet bleiben.

**Publish to template interface** Die neue Position der Publish Funktion wurde zwar nicht von allen Nutzern selbstständig gefunden, wurde jedoch im Nachhinein von allen als sinnvoll eingestuft. Die erreichte Fehlerrate von 0 stellt hier eine Verbesserung dar, da bei der jetzigen Implementierung teilweise falsche Optionen ausgewählt werden und die Nutzer lange überlegen müssen. Zusätzlich schließen die Probanden ihr Vorgehen 11,4 Sekunden schneller ab, was sich auf den fehlenden Rechtsklick zurückführen lässt. Damit sind beide Benutzeranforderungen für diese Funktion erfüllt. Trotzdem kann die neue Position die vorherige nicht in Gänze ersetzen, da es für Erstnutzer nicht möglich ist, die Funktion des

Klicks zu erkennen. Daher bietet sich eine Kombination der bisherigen Implementierung und der Neuerung an. Nutzer, die sich der Funktion bewusst sind, können den schnelleren Weg über Klick auf den Kreis wählen, Modellierer die sich noch mit der Funktionsweise von GUIDE vertraut machen, können die bisherige Variante mit dem erklärenden Text wählen.

**Multiselektion** Die Funktion „Insert in Template“ wurde von keinem der Probanden benutzt und auch bei einer nachträglichen Befragung als nicht hilfreich eingestuft. Deshalb erscheint es hier sinnvoll diese Funktion nicht weiterzuverfolgen.

Die Alignment Actions beschleunigen die Positionierung der Bilder und der Texte, bringen jedoch aktuell noch eine höhere Fehlerrate mit sich, als die bis jetzt übliche Art der Positionierung. Die begangenen Fehler lassen sich jedoch auch darauf zurückführen, dass diese Funktion in GUIDE eine komplett neue und daher ungewohnte Ergänzung bietet. Neue Funktionen müssen erst getestet und verstanden werden. Die falsche Ausführungsreihenfolge ist ein Fehler, der vermutlich nur zu Anfang und nicht mehr bei regelmäßigem Gebrauch auftritt. Diese Möglichkeiten zu erweitern und fortlaufend zu untersuchen würde sich also lohnen. Auch das Alignment von Text an Bildern sollte, zumindest für eine weitere Iteration, weiter verfolgt werden, um herauszufinden, ob die Nutzer, da sie nun von dieser Funktionalität wissen, diese auch benutzen. Die Anforderung der höheren Effizienz wird durch die Alignment Actions demnach bereits erfüllt, in Bezug auf die Fehlerrate sind jedoch noch Nachbesserungen erforderlich.

Die Positionierung der Elemente ohne Alignment Actions, sondern nur mit Multiselektion, hat innerhalb des durchgeführten Tests mit 3,5 Sekunden zeitlich am besten abgeschnitten, auch wurden hier keine Fehler durch die Nutzer begangen. Gleichzeitige Anpassung von Properties scheint sowohl zeitlich sinnvoll, als auch für den Nutzer intuitiv zu verlaufen und erfüllt zudem die formulierten Benutzeranforderungen.

Bei der Skalierung der Bilder bietet die Multiselektion, auf den ersten Blick, keinen zeitlichen Vorteil. Es gilt jedoch zu bedenken, dass im Testfall nur drei Bilder vorhanden sind, der Kopiervorgang dieser Bilder also kein großes zeitliches Gewicht hat. Beinhaltet eine Spezifikation jedoch beispielsweise zehn, statt nur drei Bilder, geht durch den Kopiervorgang bereits mehr Zeit verloren. Den Ansatz, die Objekte zu kopieren und dann entsprechend anzupassen, haben die Nutzer nur bei den Bildern, jedoch nicht bei den Texten verfolgt. Deshalb lässt sich bei den Messwerten innerhalb des Prototyps eine zeitliche Verbesserung gegenüber der bisherigen Version verzeichnen. Da sich die Multiselektion für die Positionierung bereits als Mehrwert erwiesen hat, ist es sinnvoll die gleichzeitige Anpassung aller Properties bei Mehrfachselektion zu ermöglichen. Es würde den Nutzer irritieren, wenn nur gewisse Eigenschaften gleichzeitig anpassbar wären, weshalb es sinnvoll ist, auch weiterhin width und height der Objekte zusammen anpassen zu können. In weiteren Iteration wäre es hier jedoch sinnvoll zu untersuchen, ob bei größeren Projekten tatsächlich ein zeitlicher

Mehrwert eintritt, da dies hier lediglich vermutet wird. Die Benutzeranforderungen sind, im Bezug auf die Skalierung der Objekte, dementsprechend noch nicht komplett erfüllt. Die Fehlerrate beläuft sich zwar weiterhin auf 0, allerdings ist ein Mehrwert im Bezug auf die Effizienz noch nicht sichergestellt.

## 5.5. Ausblick

Die vorliegenden Ergebnisse werden auf Grundlage einer Iteration des Human Centered Design Process erlangt. Je nachdem, ob nach dieser Iteration die Benutzeranforderungen erfüllt sind, ist es notwendig das Design anzupassen und nochmals zu evaluieren oder es als abgeschlossen zu betrachten und zur Implementierung überzugehen.

Wie im vorherigen Abschnitt bereits festgehalten erfüllt die Filterfunktion noch nicht alle Anforderungen und bedarf in Bezug auf die Fehlerrate noch einer Nachbesserung. Diese gilt es in einem weiteren Test erneut zu evaluieren.

Die neue Position der Funktion „publish to template interface“ erfüllt alle definierten Bedingungen und kann deshalb, zusätzlich zur bestehenden Funktion, implementiert werden.

Die Funktionen der Multiselektion gilt es an diesem Punkt differenziert zu betrachten. Da diese Ergänzung komplett neu in EB GUIDE sind, gibt es hier noch viele Möglichkeiten und Funktionen, die hinzugefügt und evaluiert werden können. Das gleichzeitige Anpassen von Eigenschaften, wie der Position und Skalierung von Objekten, hat sich im Rahmen dieser Iteration bereits als nützlich erwiesen und kann ebenfalls implementiert werden, da herausgefunden wurde, dass die Modellierer diese Funktion auch nutzen. Dadurch wird eine Grundlage für weitere Interaktionsmöglichkeiten dieser Art gelegt. So ist es beispielsweise üblich die Eigenschaften, die nun gemeinsam anpassbar sind, mit Datapoolitems zu verlinken. Bei den Befragungen nach dem Tests äußern hier einige der Nutzer den Wunsch diese Verlinkung ebenfalls, mithilfe der Multiselektion durchführen zu können, was für eine weitere Iteration im Prototyp ergänzt und evaluiert werden kann. Zusätzlich wird angegeben, dass über dem Propertie Panel angezeigt werden soll, welche Objekte aktuell zusammen ausgewählt sind, da dies bis jetzt nur im View erkennbar ist.

Da bei den durchgeführten Tests deutlich wird, dass die Nutzer sich für die Alignment Actions interessieren, kann nun in folgenden Iterationen noch eine Verbesserung und Erweiterung der Funktionalitäten angestrebt werden. Für diesen Test war der Prototyp so ausgelegt, dass bereits bekannt war, welches der drei Bilder des Hauptmenüs links, rechts und in der Mitte liegt. Dadurch hat sich die Anordnung, durch einen Klick auf den Button, automatisch an die Spezifikation angepasst. Für tatsächliche Arbeitsabläufe ist dies nicht der Fall, weshalb es für den Nutzer möglich sein muss, anzugeben in welcher Anordnung die

Bilder platziert werden sollen. Dies wäre zum Beispiel dadurch umsetzbar, dass der Nutzer ein Objekt als Ausrichtungspunkt deklarieren kann.

Die Multiselektion bietet also noch viele Möglichkeiten, die in weiteren Durchführungen des Human Centered Design Process den Prototyp zu ergänzen und die neuen Funktionen zu evaluieren, die Funktion „Insert in Template“ gilt es jedoch zum aktuellen Zeitpunkt zu verwerfen.



## Kapitel 6.

### Fazit

Im Rahmen dieser Bachelorarbeit, wurde eine Iteration des Human-Centered Design Process durchlaufen. Hierbei wurden Usabilityschwächen im Interface von EB GUIDE 6 entdeckt, woraufhin für drei dieser Defizite Lösungsansätze erarbeitet wurden. Durch diese eingearbeiteten Verbesserungen, konnte eine Steigerung der Usability verzeichnet werden, auch wenn nicht alle definierten Benutzeranforderungen zufriedenstellend erfüllt werden konnten. Für den zeitlichen Rahmen, in dem nur eine Iteration des Design Process vorgesehen war, entspricht dieses Ergebnis jedoch den Erwartungen, und kann mithilfe weiterer Iterationen noch verbessert werden.

Die, für die Bewertung der abschließenden Messungen benötigten Benutzeranforderungen, wurden hierbei auf der Grundlage von durchgeführten Untersuchungen der Arbeitsabläufe von Nutzern formuliert. Während dieser Untersuchungen, wurden Personen aus der zutreffenden Zielgruppe, bei ihrer täglichen Arbeit beobachtet. Dadurch konnten repräsentative Schwachstellen im User Interface von EB GUIDE aufgedeckt werden, für die qualitative Nutzeranforderungen definiert werden konnten. Aus allen gefunden Schwächen wurden drei ausgewählt, wobei darauf geachtet wurde eine Anpassungen mit großem, und zwei Anpassungen mit kleinem Umfang zu wählen, anhand deren Usabilitymerkmale, Effizienz und Fehlerrate überprüft werden können.

Zu diesem Zeitpunkt der Arbeit war noch nicht absehbar, dass ein Teil der ausgewählten Anpassungen mithilfe eines Prototyps und ein Teil in EB GUIDE implementiert umgesetzt werden würde. Diese Tatsache führte bei dem abschließenden Test zur Notwendigkeit diesen in zwei Teile aufzuspalten, was für die Nutzer, durch die Unterbrechung des Arbeitsablaufes. Daher wäre es bereits an dieser Stelle hilfreich gewesen, sich über die Umsetzung der Anpassungen Gedanken zu machen, und nur solche zu wählen, die einen zusammenhängenden Testablauf ermöglichen.

Bei der Überarbeitung der ausgewählten Schwächen - der Funktion „publish to template interface“, den Filter für die „Widget Feature Properties“ und die „Mehrfachselektion“ - wurde sich beim Entwurf des Designs, an bekannten Gestaltprinzipien orientiert. Für die Umsetzung dieser Entwürfe mussten zusätzlich Interaktionsmöglichkeiten definiert werden, die der

Prototyp bzw. Software bereitstellen muss, um eine erfolgreiche Interaktion zu gewährleisten. Im Rahmen der durchgeführten Tests wurde deutlich, dass zwar nicht alle gewünschten Interaktionen durchgeführt werden konnten, es jedoch für jeden Nutzer möglich war das Ziel der Arbeitsaufgabe zu erreichen. Allgemein ist es - bei einer so umfangreichen Software wie EB GUIDE - nicht möglich oder nötig alle Funktionen derselbigen innerhalb eines Prototyps zu simulieren. Da alle Nutzer ihr verfolgtes Ziel erreichen konnten, lässt sich jedoch festhalten, dass die Interaktionsmöglichkeiten die Bedürfnisse der Nutzer ausreichend abgedeckt haben, ohne zu großen Aufwand für den Modellierer des Prototyps zu erzeugen.

Die Umsetzung - in Form eines Prototyps - wurde mit AXURE RP gelöst, was eine Einarbeitung in dieses Tool erfordert hat. Da die Dokumentation hierzu jedoch sehr umfangreich ist und auch Hilfestellung von Mitarbeitern von Elektrobat möglich war, war die Software, rückwirkend betrachtet, gut geeignet. Auch war es möglich alle definierten Interaktionsmöglichkeiten - aufgrund der Möglichkeit Logik und globale Variablen in den Prototyp einzubauen - wie gewünscht umzusetzen. Die Implementierung des Filters wurde mithilfe von WPF und C# gelöst, wobei bereits im Studium erlangte Kenntnisse zu diesen Programmiersprachen genutzt werden konnten. Deshalb war es lediglich nötig, sich in das Konzept des MVVM Pattern einzuarbeiten, sowie die Struktur des bestehenden Projektes zu verstehen.

Die abschließenden Tests wurden in Form von Remote Usability Tests umgesetzt. Aufgrund der örtlichen Verteilung der Probanden war dies eine offensichtliche Wahl, die auch zu gut vergleichbaren Ergebnissen bei der Auswertung führte. Die Wahl der Arbeitsaufgabe kann insofern positiv bewertet werden, dass die Nutzer intuitiv wussten was sie tun sollten und in welchem Zusammenhang die textuelle Angabe und der Styleguide zu betrachten sind.

Als Ergebnisse der Tests kann abschließend festgehalten werden, dass der Filter der „Widget Feature Properties“ zwar die Effizienz steigert, jedoch aktuell noch zu Fehlern führt. Die neue Position der Funktion „Publish to template interface“, verringert die Fehlerrate der Nutzer und steigert gleichzeitig die Effizienz, weshalb diese Anpassung als abgeschlossen betrachtet und implementiert werden kann. Im Rahmen der Multiselektion kann die Funktion „Insert in Template“ verworfen werden, da die Probanden diese Funktion einstimmig als nicht intuitiv eingestuft haben. Die „Alignment Actions“ steigern aktuell die Effizienz, erhöhen jedoch - vermutlich aufgrund der Fremdheit der neuen Funktionalität - aktuell noch die Fehlerrate. Die gemeinsame Anpassung von Eigenschaften ausgewählter Elemente, bringt bei der Positionierung einen Mehrwert der Effizienz und senkt die Fehlerrate, bei der Skalierung von Objekten ist aktuell jedoch keine Verbesserung erkennbar.

Aufgrund der Tatsache, dass der Prototyp einsatzfähig vorliegt, können damit noch weitere Iterationen des Design Process durchlaufen werden und weiter auf die, noch bestehenden Usabilityschwächen eingegangen werden. Es wurden mit dieser Arbeit also zum einen Ergebnisse erlangt, die in EB Guide als Verbesserung implementiert werden können. Zum anderen wurde eine Grundlage geschaffen um weitere Iterationen durchlaufen zu können



und die große Anpassung der Multiselektion noch um weitere Inhalte ergänzen und testen zu können.





## Anhang A.

### Subtasks

#### Aufgaben und Subtasks

##### Teil 1

#### 1. Einfügen der 3 Hauptbilder und hinzufügen Widgetproperties

- 1.1 Bilder aus Assets in View ziehen
- 1.2 Add Widget Feature Properties Button klicken
- 1.3 Klicken in das Filterfeld
- 1.4 Eingabe Filterbegriff
- 1.5 Anwählen der gewünschten Properties

##### Teil 2

#### 2. Template erstellen und Properties verlinken

- 2.1 Image Template anlegen
- 2.2 Klicken auf Kreis (publish to template Interface)
- 2.3 Properties setzen

#### 3. Template verwenden

- 3.1 Template in View ziehen
- 3.2 Published Properties setzen
- 3.3 Bild zuweisen/einfügen
  - 3.3.1 gewünschtes Bild in den View ziehen
  - 3.3.2 Template auswählen
  - 3.3.3 STRG drücken
  - 3.3.4 Bild auswählen
  - 3.3.5 Button "Insert in Template" drücken

#### 4. Positionieren/Skalieren Bilder

- 1. Positionieren
  - 1.1 Erstes Bild korrekt positionieren (x,y)
  - 1.2 STRG drücken
  - 1.3 alle gewünschten Bilder auswählen
  - 1.4 Horizontal Distance eingeben
  - 1.5 Button/Enter klicken

##### oder

- 1.1 Mehrere Auswählen mit STRG
- 1.2 Werte für alle anpassen
- 2. Skalieren
  - 2.1 STRG drücken
  - 2.2 gewünschte Bilder auswählen
  - 2.3 in Properties Feld klicken (width/height)
  - 2.4 gewünschten Wert eingeben

## 5. Positionieren/Skalieren Texte

1. Labels erstellen ( Drag & Drop aus Toolbox)

2. Textproperty anpassen

3. Positionieren

3.1 Ausrichten aneinander

3.1.1 Ersten Text korrekt positionieren (x,y)

3.1.2 STRG drücken

3.1.3 alle gewünschten Texte auswählen

3.1.4 Horizontal Distance eingeben

3.1.5 Button/Enter klicken

**oder**

3.2 Ausrichten an Bildern

3.2.1 STRG drücke

3.2.2 Bild auswählen

3.2.3 Vertical Distance eingeben

3.2.4 Button/Enter klicken

**oder**

1.1 Mehrere Auswählen mit STRG

1.2 y Werte für alle anpassen

4. Skalieren

4.1 STRG drücken

4.2 gewünschte Texte auswählen

4.3 in Properties Feld klicken

4.4 gewünschten Wert eingeben





## Anhang B.

### Arbeitsaufgabe Implementierter Filter

Für ein neues Projekt hast du gerade die erste Spezifikation für einen Startscreen bekommen, die dir in Form eines Styleguide vorliegt.

Um ein Gefühl für das Projekt zu bekommen, modellierst du den Startscreen nach den Vorgaben des Kunden in EB Guide 6.

#### **Teil 1 (in EB Guide 6)**

Füge zuerst die drei großen Bilder für das Hauptmenü zu deinem View hinzu, und versehe sie mit den Widget Properties, die dir für das weitere Vorgehen sinnvoll erscheinen.

Bearbeite in diesem Schritt bitte keine anderen Eigenschaften der Bilder, und verwende keine Templates.





## Anhang C.

### Arbeitsaufgabe Prototyp

#### Teil 2 (Prototyp)

Der vorliegende Prototyp bietet einige Aktionsmöglichkeiten, wenn mehrere Objekte gleichzeitig selektiert sind. Diese kannst du für das Lösen dieser Aufgabe benutzen.

Halte für die Multiselektion STRG gedrückt.

Füge die beiden Bilder für die Fast-Access Leiste mithilfe von Image Templates hinzu. Die Funktion „publish to template interface“ befindet sich jetzt an einer anderen Position und ist für die Properties width, height, x, y und image möglich.

Für das Zuweisen der Bilder innerhalb der Templates bietet die Multiselektion jetzt auch eine neue Möglichkeit.

Das Anlegen von WidgetProperties ist hier nicht nötig.

Die drei Bilder für das Hauptmenü liegen bereits in deinem View.

Setze jetzt noch den Rest der Spezifikation um.

Löse diesen Teil der Aufgabe bitte ohne Templates.

Anmerkungen:

Das Hintergrundbild ist bereits korrekt positioniert, und die nötigen Icons liegen im Ressourcenordner von EB Guide.

Die Bilder müssen nicht anklickbar sein, Events feuern oder verarbeiten



## Anhang D.

### Arbeitsaufgabe bestehende Software

Für ein neues Projekt hast du gerade die erste Spezifikation für einen Startscreen bekommen, die dir in Form eines Styleguide vorliegt.

Um ein Gefühl für das Projekt zu bekommen, modellierst du den Startscreen nach den Vorgaben des Kunden in EB Guide 6.

Zuerst fügst du die drei Bilder für das Hauptmenü ein, arbeite hier bitte ohne Templates und versehe sie mit den Widget Properties die dir für das weitere Vorgehen sinnvoll erscheinen.

Füge die beiden Bilder für die Fast-Access Leiste mithilfe von Templates hinzu.

Die noch fehlenden Texte setzt du bitte ohne die Nutzung von Templates um.

Anmerkungen:

Das Hintergrundbild ist bereits korrekt positioniert und die nötigen Icons liegen im Ressourcenordner von EB Guide.

Die Bilder müssen nicht anklickbar sein, Events feuern oder verarbeiten.

# Abbildungsverzeichnis

2.1.	Zusammenhang Usability und User Experience (nach [Saro 16]) . . . . .	9
2.2.	Human - Centered Design - Process bei Elektrobit . . . . .	11
2.3.	Dropdownmenü in Word . . . . .	13
2.4.	Checkbox und Schieberegler in EB GUIDE . . . . .	14
2.5.	Aufbau EB GUIDE . . . . .	17
2.6.	EB Guide Studio Statemachine . . . . .	18
2.7.	EB Guide VIEW . . . . .	19
3.1.	Usability - Schwäche Navigation . . . . .	26
3.2.	Usability - Schwäche Template Properties . . . . .	27
3.3.	Usability - Schwäche Widget Feature Properties . . . . .	28
3.4.	Usability - Schwäche Mehrfachselektion . . . . .	29
3.5.	Ergänzung zur Navigation in Form einer Outline . . . . .	31
3.6.	Verbesserung Template Properties . . . . .	33
3.7.	Verbesserung Feature Property Properties . . . . .	34
3.8.	Verbesserung Mehrfachselektion . . . . .	35
3.9.	Verbesserung Mehrfachselektion Einfügen in Template . . . . .	36
4.1.	Axure RP 9 . . . . .	39
4.2.	Axure RP 9 Outline . . . . .	39
4.3.	Axure RP 9 Style Properties . . . . .	40
4.4.	Axure RP Droplist und Conditions . . . . .	41
4.5.	Axure RP Preview . . . . .	41
4.6.	Grundlage des Prototyps . . . . .	43
4.7.	Scrollbar für Assets . . . . .	44
4.8.	Widget Tree . . . . .	45
4.9.	Properties Panel . . . . .	46
4.10.	Beispielbedingung Multiselektion . . . . .	46
4.11.	Alignment Actions . . . . .	48
4.12.	Droplist für Images . . . . .	48
4.13.	Model-View-ViewModel (MVVM) Design Pattern . . . . .	51
5.1.	Styleguide . . . . .	59
5.2.	Aufdeckungsrate von Usabilityproblemen . . . . .	61

5.3. Filterung nach Scale Mode . . . . . 63

5.4. Auswahlmöglichkeiten nach Rechtsklick auf Templateproperties . . . . . 65

# Tabellenverzeichnis

3.1. Usability - Schwäche Image List . . . . .	25
--	----





## Auflistungen

4.1. CollectionViewSource() . . . . .	52
4.2. Widget Properties Filter . . . . .	53
4.3. Filtertext Properties . . . . .	54
4.4. Filterpanel in xaml . . . . .	54



## Literaturverzeichnis

- [All 20] “All Lookback’s features”. <https://lookback.io/features/>, 10.01.2020. [Online; accessed 13-Januar-2020].
- [bota] dotnet bot. “CollectionViewSource Klasse (System.Windows.Data)”. <https://docs.microsoft.com/de-de/dotnet/api/system.windows.data.collectionviewsource?view=netframework-4.8>. [Online; accessed 02-Januar-2020].
- [botb] dotnet bot. “CollectionViewSource.Filter Ereignis (System.Windows.Data)”. <https://docs.microsoft.com/de-de/dotnet/api/system.windows.data.collectionviewsource.filter?view=netframework-4.8>. [Online; accessed 02-Januar-2020].
- [botc] dotnet bot. “ObservableCollection<T> Class (System.Collections.ObjectModel) | Microsoft Docs”. <https://docs.microsoft.com/en-us/dotnet/api/system.collections.objectmodel.observablecollection-1?view=netframework-4.8>. [Online; accessed 10-Januar-2020].
- [Bull 94] H.-J. Bullinger. *Ergonomie: Produkt- und Arbeitsplatzgestaltung. Technologie-management - Wettbewerbsfähige Technologieentwicklung und Arbeitsgestaltung*, Vieweg+Teubner Verlag, Wiesbaden and s.l., 1994.
- [DIN ] “DIN EN ISO 9241-210 Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme (ISO 9241-210:2010)”.
- [Eleka] Elektrobit Automotive GmbH. “About Elektrobit (EB) - Elektrobit”. <https://www.elektrobit.com/about/>. [Online; accessed 19-Oktober-2019].
- [Elekb] Elektrobit Automotive GmbH. “EB GUIDE Home”. <https://infohub.automotive.elektrobit.com/display/EBGUIDE/EB+GUIDE+Home>. [Online; accessed 19-Oktober-2019].
- [Elekc] Elektrobit Automotive GmbH. “EB GUIDE Studio - User guide”.

- [Elekd] Elektrobit Automotive GmbH. “User Experience Engineering”. <https://infohub.automotive.elektrobit.com/display/QMS2019x2/User+Experience+Engineering>. [Online; accessed 03-Oktober-2019].
- [Eleke] Elektrobit Automotive GmbH. “User requirements”. <https://infohub.automotive.elektrobit.com/pages/viewpage.action?spaceKey=QMS2019x2&title=User+requirements>. [Online; accessed 27-Dezember-2019].
- [Jako 00] Jakob Nielsen. “Why You Only Need to Test with 5 Users”. <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>, 2000. [Online; accessed 13-Januar-2020].
- [Jako 93] Jakob Nielsen. “Response Time Limits: Article by Jakob Nielsen”. <https://www.nngroup.com/articles/response-times-3-important-limits/>, 1993. [Online; accessed 12-Dezember-2019].
- [Knig 19] W. Knight. *UX for Developers*. Apress, Berkeley, CA, 2019.
- [Lidw 10] W. Lidwell, K. Holden, and J. Butler. *Universal principles of design: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design / William Lidwell, Kritina Holden, Jill Butler*. Rockport, Beverly, Mass., [rev. and updated] ed. Ed., 2010.
- [Miuc 19] R. Miucic. *Connected vehicles: Intelligent transportation systems / Radovan Miucic, editor. Wireless networks, 2366-1186*, Springer, Cham, Switzerland, 2019.
- [Niel 93] J. Nielsen and T. K. Landauer. “A mathematical model of the finding of usability problems”. In: S. E. Ashlund, Ed., *Human factors in computing systems*, pp. 206–213, IOS Press, Netherlands, 1993.
- [Niel 95] J. Nielsen. *Usability engineering*. AP Professional, Boston and London, [new ed.] Ed., 1995?
- [Norm 16] D. A. Norman. *The design of everyday things*. Verlag Franz Vahlen, München, Überarbeitete und erweiterte auflage Ed., 2016.
- [Rich 16] M. Richter. *Usability und UX kompakt [electronic resource]: Produkte für Menschen. IT kompakt*, Springer Vieweg, Berlin, 4. auflage Ed., 2016.
- [Saro 16] F. Sarodnick and H. Brau. *Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung*. Hogrefe, Bern, 3., unveränderte auflage Ed., 2016.

- [Unde] “Understanding the basics of MVVM design pattern – Microsoft Gulf Technical Community”. <https://blogs.msdn.microsoft.com/msgulftcommunity/2013/03/13/understanding-the-basics-of-mvvm-design-pattern/>. [Online; accessed 02-Januar-2020].