

FH Aachen

Fachbereich Elektrotechnik und Informationstechnik

Studiengang Media and Communications for Digital Business

Bachelorarbeit

Component Sprints: Entwicklung eines neuartigen Ansatzes zur Optimierung des Übergangs vom Prototypen zum MVP

vorgelegt von

Leo Bernard

Matrikel-Nr. **3069756**

Referent:

Marco Motullo

Datum:

17. Juni 2014

Besonderer Dank gilt Daniel Wirtz, René Nauheimer, Marcus Weiner und Moritz Gunz

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum

Leo Bernard

Inhalt

Abbildungsverzeichnis	I
Tabellenverzeichnis	II
0.1 Einleitung	1
0.2 Motivation / Zielsetzung	1
0.3 Begründung	1
0.4 Vorgehen	2
0.5 Theoretische Grundlagen / Einordnung	2
0.5.1 Prozess der Produktentwicklung	2
0.5.2 Auswahl relevanter Konzepte und Frameworks	3
0.5.3 Design Sprints	3
0.5.4 Prototyping	10
0.5.5 Minimum Viable Products	11
0.5.6 Design Systems	12
0.5.7 Atomic Design	12
0.5.8 Rolle des Component Sprints	16
0.6 Ausarbeitung des Konzepts für Component Sprints	16
0.7 Empirischer Test des entwickelten Konzepts	16
0.7.1 Ablauf	16
0.7.2 Ergebnisse	16
0.7.3 Erkenntnisse / Learnings	16
0.8 Qualitative Studie zur Verfeinerung des Konzepts	16
0.8.1 Studiendesign	16
0.8.2 Ergebnisse	17

0.9 Weiteres Vorgehen 17

Literatur 18

Abbildungsverzeichnis

0.1	Klassischer Entwicklungszyklus nach Jake Knapp [Knapp (2018), S. 65]	3
0.2	Design Sprint Zyklus [Google Ventures (2019)]	4
0.3	Struktur des gesamten Design Sprints [Knapp u. a. (2016), S. 17]	4
0.4	Ergebnis der Heat map Übung	7
0.5	8
0.6	Kategorisierung der Arten von Prototyping [Verner u. a. (o.D.), S. 7] . . .	10
0.7	Struktur des Atomic Designs [Frost (2016), K. 2]	13
0.8	Header-Organismus [Frost (2016), K. 2]	13

Tabellenverzeichnis

0.1 Einleitung

0.2 Motivation / Zielsetzung

Ziel dieser Bachelorarbeit ist es, ein Konzept für einen Sprint zu entwickeln, der den Übergang vom Prototypen aus einem Design Sprint zum fertigen Minimum Viable Product optimal überbrückt.

0.3 Begründung

Design Sprints wurden 2012 von Jake Knapp entwickelt (Knapp u. a. 2016, S. 5) und werden dazu eingesetzt, innerhalb von einer Woche eine Produktidee zu einem Prototypen auszuarbeiten und an echten Nutzern zu testen. Design Sprints können für jede Art von Produkten – physisch und digital – eingesetzt werden. In dieser Arbeit konzentriere ich mich jedoch auf den Einsatz von Design Sprints im Softwarebereich, also z.B. für Apps und Websites.

Nach einem erfolgreichen Design Sprint entwickeln Firmen meist auf Basis des dort entwickelten Prototypen ein Minimum Viable Product, oder auch MVP, das sie auf den Markt bringen können.

Der in dieser Bachelorarbeit ausgearbeitete Sprint, den ich Component Sprint nenne, soll Entwicklern die Entwicklung des MVP vereinfachen, indem sie bereits alle nötigen UI-Komponenten in Form einer Pattern Library zur Verfügung gestellt bekommen. So können sie sich bei der Entwicklung des MVP mehr auf die Implementierung der Logik konzentrieren.

Ich möchte den Component Sprint nach einer festen Struktur aufbauen, sodass er – wie ein Design Sprint – in einer Woche zuverlässig wertvolle Ergebnisse liefert.

0.4 Vorgehen

Um das Konzept für Component Sprints zu entwickeln, werde ich zuerst bestehende Grundlagen erörtern und relevante Konzepte und Frameworks vorstellen.

Basierend auf diesen Frameworks werde ich die erste Version des Component Sprints entwickeln. Darauf folgend werde ich einen empirischen Test des entwickelten Konzepts in Form eines Design Sprints mit anschließendem Component Sprint durchführen und die Ergebnisse sowie Erkenntnisse zusammenfassen.

Um das Konzept weiter zu verfeinern werde ich eine qualitative Studie in Form von drei Interviews mit Personen aus relevanten Bereichen durchführen, in denen ich das Konzept sowie bestehende Ergebnisse vorstelle und weitere Meinungen einhole.

Letztlich werde ich Schritte zur Weiterentwicklung des Konzepts anreißen und darlegen, wie sich Component Sprints in Zukunft ausbauen lassen können.

0.5 Theoretische Grundlagen / Einordnung

0.5.1 Prozess der Produktentwicklung

In den letzten 20 Jahren hat sich der Prozess, in dem Software entwickelt und veröffentlicht/verbreitet wird, stark verändert. Vor 20 Jahren wurde Software hauptsächlich als CD verkauft. Ein normaler Release-Zyklus lag bei mehreren Monaten. Bugs in der Software konnten erst in der nächsten Version behoben werden, Feedback von Benutzern konnte erst Monate später umgesetzt werden.

Durch die Verbreitung des Internets wurde es einfacher, neue Versionen regelmäßiger zu veröffentlichen. Es ist heutzutage normal, dass Software täglich neue Updates

bekommt. Analytics Tools wie Google Analytics und Mixpanel ermöglichen es Firmen, genaue Insights zum Benutzerverhalten in ihrer Software zu erlangen. So können potentielle Problemstellen in Programmen genauer identifiziert und behoben werden.

In den folgenden Kapiteln werde ich mich auf die heutige Entwicklung von Apps und Websites konzentrieren.

0.5.2 Auswahl relevanter Konzepte und Frameworks

0.5.3 Design Sprints

Ein normaler Zyklus in der heutigen Entwicklung von Apps und Websites kann in vier Schritte unterteilt werden:

- **Ideate:** Generierung und Ausarbeitung von Ideen
- **Build:** Umsetzung der Ideen in Software
- **Launch:** Veröffentlichung der Software auf dem Markt
- **Data:** Sammeln von Insights und User Feedback

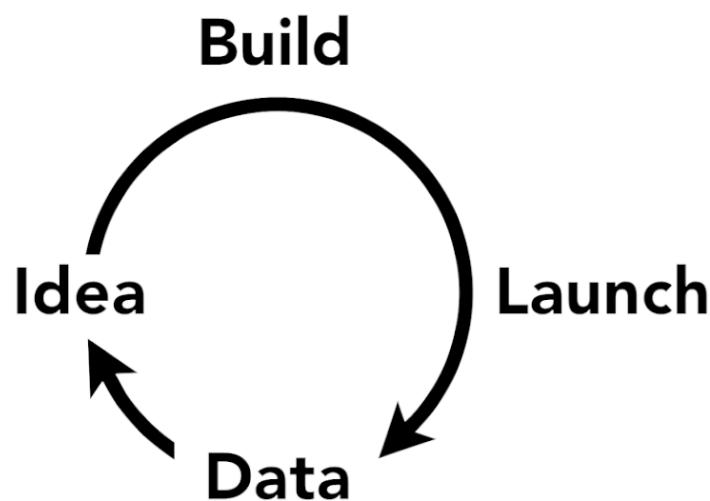


Abb. 0.1: Klassischer Entwicklungszyklus nach Jake Knapp [Knapp (2018), S. 65]

Bei der Entwicklung von Apps ist es normalerweise notwendig, diesen Zyklus mehrere Male zu durchlaufen, um eine optimale Version zu erhalten. Dabei ist vor allem der Build-Schritt zeitaufwändig. Wenn der Launch nicht erfolgreich ist, wurden diese Ressourcen verschwendet.

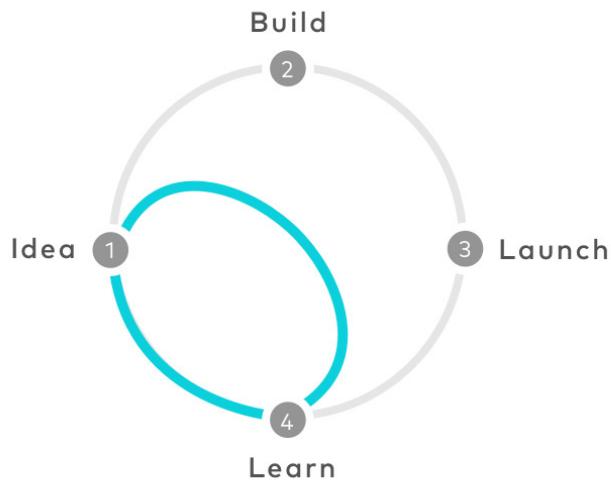


Abb. 0.2: Design Sprint Zyklus [Google Ventures (2019)]

Der Design Sprint ist ein Ansatz, der diesen Zyklus optimiert. Hierbei werden die Build- und Launch-Schritte durch Prototyping und User Testing ersetzt, sodass User Feedback schneller gesammelt werden kann.

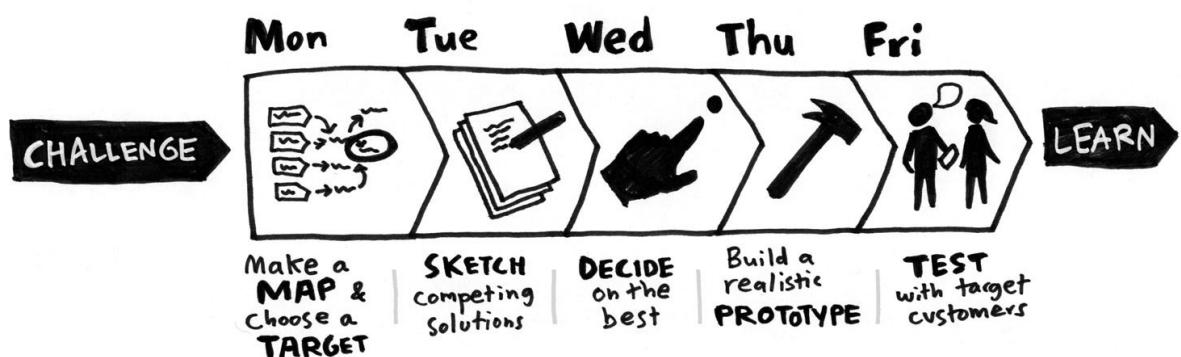


Abb. 0.3: Struktur des gesamten Design Sprints [Knapp u. a. (2016), S. 17]

Ein Design Sprint dauert in der Regel fünf Tage – Montag bis Freitag – und wird vor Ort mit einem interdisziplinären Team von 4-7 Personen und einem Moderator durchgeführt. Die Tage sind zeitlich fest strukturiert und jegliche Form von Ablenkung,

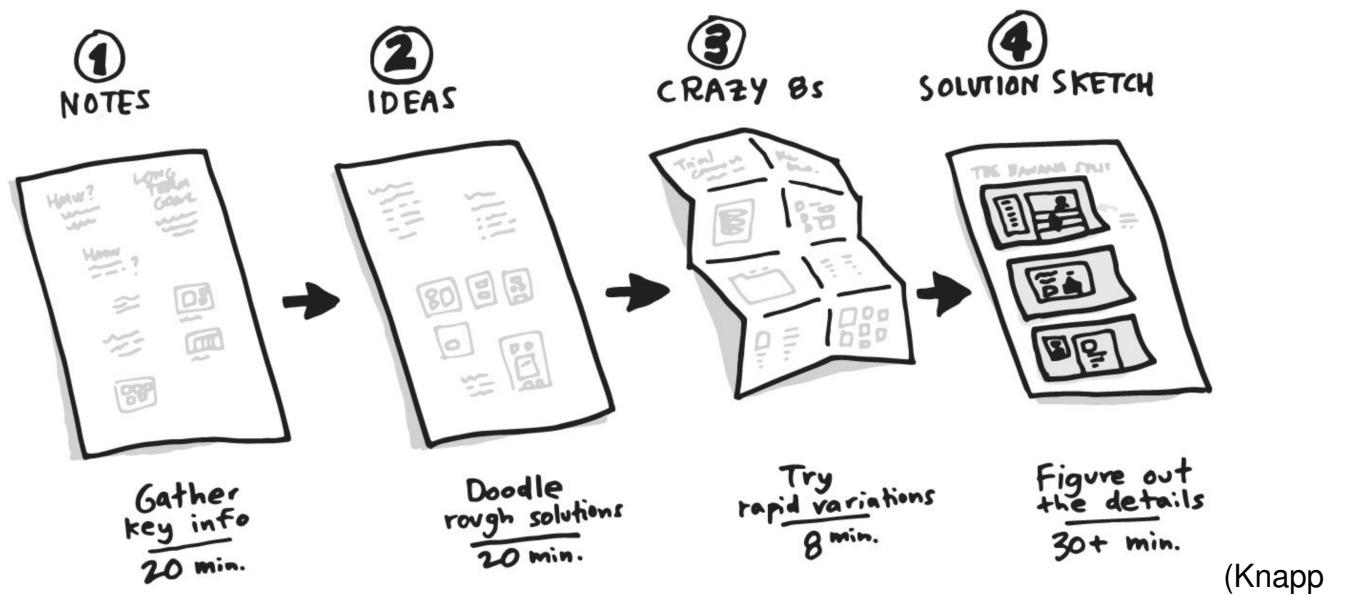
wie z.B. Handys oder Laptops, ist nicht erlaubt. Ziel dieser Maßnahmen ist es, dass sich das Team fokussiert mit den Aufgaben auseinandersetzen kann (Knapp u. a. 2016, S. 41). Gesammelte Informationen werden auf Post-its festgehalten, sodass sie für jedes Teammitglied jederzeit sichtbar sind (Knapp u. a. 2016, p. 20).

Generell wird in Design Sprints auf lange Diskussionen verzichtet. Diese sind nicht effizient und geben Teilnehmern mit einer starken Meinung zu viel Gewichtung und Einfluss auf die Entscheidungen (Kahneman 2013, K. 7). Stattdessen werden Entscheidungen per Abstimmung, meist mit Hilfe von Klebepunkten, getroffen. Der CEO, Gründer, Produktmanager oder Head of Design hat als Decider in Abstimmungen eine größere Entscheidungskraft (Knapp u. a. 2016, S. 34).

Am Montag wird die Problemstellung definiert. Vormittags definiert das Team nach einer kurzen Einführung das Langzeitziel des Projektes und Fragen, die für den Sprint interessant sein könnten. Darauf folgend wird eine Abbildung der erwarteten Customer Journey angelegt (Knapp u. a. 2016, S. 65).

Nachmittags werden sogenannte Experteninterviews durchgeführt. Ziel dieser Interviews ist das Sammeln und Festhalten von Wissen, das häufig auf einzelne Teammitglieder verteilt ist (Knapp u. a. 2016, S. 68). Während der Interviews legen alle Mitglieder in Stille Notizen auf Post-its in Form von "How Might We"- (oder auf Deutsch "Wie können wir")-Fragen an. Eine Frage kann beispielsweise lauten: "Wie können wir das User-Onboarding vereinfachen?" oder "Wie können wir unser Produkt freundlich wirken lassen?". Diese Fragen werden daraufhin an einer Wand sortiert, per Abstimmung priorisiert und die wichtigsten in die Customer Journey eingeordnet (Knapp u. a. 2016, S. 80-88).

Am Dienstag sucht das Team online nach Konzepten, Komponenten und anderen Beispielen, die im Kontext der am Montag definierten Fragen und Ziele relevant sein könnten und präsentiert diese kurz. Für jede Komponente wird ein Post-it erstellt (Knapp u. a. 2016, S. 98-100).



u. a. 2016, S. 109)

Basierend auf diesen Ideen werden nun von allen Teammitgliedern Skizzen und Notizen angelegt. Diese werden über weitere Übungen zu einem Solution Sketch pro Person ausgearbeitet. Diese Sketches bilden in maximal 3 Schritten detailliert die Interaktion der Nutzer mit dem Produkt ab und bleiben den anderen Teammitgliedern gegenüber bis zum nächsten Tag versteckt (Knapp u. a. 2016, S. 114-118).

Am Mittwoch entscheidet sich das Team für den besten Solution Sketch. Diese Entscheidung wird in 5 Schritten durchgeführt: Art museum, Heat map, Speed critique, Straw poll und Supervote (Knapp u. a. 2016, S. 131).

Im ersten Schritt werden die Solution Sketches, die am Vortag erstellt wurden, aufgedeckt und nebeneinander aufgehängt. Die Sketches sind anonym, um eventuellen persönlichen Einflüssen entgegenzuwirken (Knapp u. a. 2016, S. 132).

Darauf folgend markiert jedes Teammitglied auf jedem Solution Sketch mit Klebepunkten interessante Konzepte und Ideen. Fragen und Anmerkungen können per Post-it hinzugefügt werden. Dieser Schritt wird in Stille durchgeführt und gibt dem Team einen guten Überblick über interessante Bereiche in den Sketches (Knapp u. a. 2016, S. 132-135).

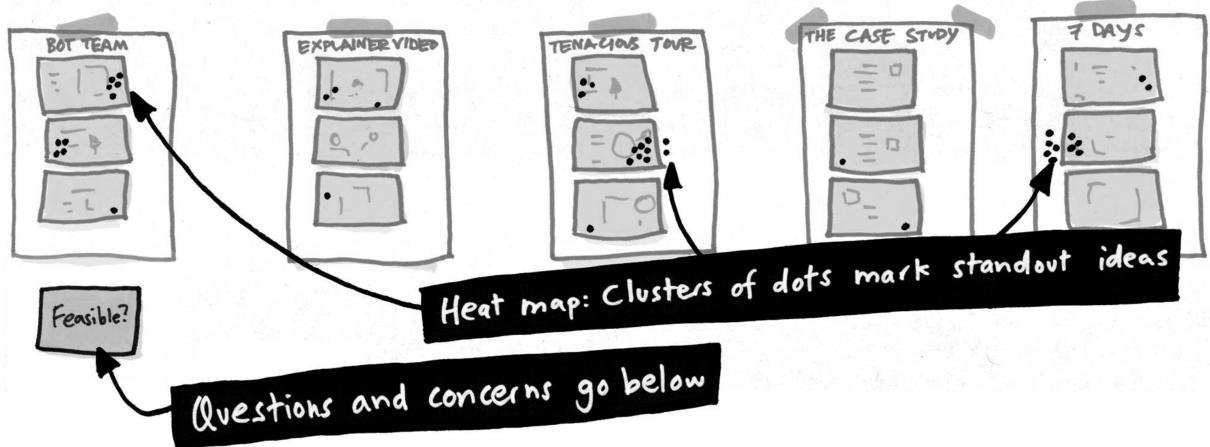


Abb. 0.4: Ergebnis der Heat map Übung

Im dritten Schritt wird jeder Solution Sketch drei Minuten lang besprochen. Die Besprechung folgt einer festen Struktur, damit sie konstruktiv bleibt und sich das Team nicht in Details verliert. Die Ergebnisse der Besprechungen werden auf Post-its über den Solution Sketches aufgehängt (Knapp u. a. 2016, S. 135-137).

Im vierten Schritt stimmt jedes Teammitglied für den Solution Sketch, den er am besten findet, ab und erklärt seine Entscheidung in einem ein-minütigen Vortrag. (Knapp u. a. 2016, S. 138-140).

Im letzten Schritt entscheidet sich der Decider für seine Favoriten, basierend auf den Abstimmungsergebnissen der anderen Teammitglieder, den Sprintfragen und dem festgelegten Langzeitziel. Diese Entscheidung ist absolut und die Grundlage für das folgende Prototyping (Knapp u. a. 2016, S. 140-142). Wenn mehr als ein Favorit ausgewählt wurde, wird gemeinsam entschieden, ob alle favorisierten Solution Sketches in einen Prototyp übernommen werden oder ob mehrere Prototypen erstellt werden sollen (Knapp u. a. 2016, S. 145).

Am Mittwoch Nachmittag wird basierend auf den gewählten Solution Sketches ein oder mehrere Storyboards auf einem Whiteboard erstellt. Dies geschieht im Team, wobei der Decider auch hier das letzte Wort hat. Das Storyboard dient als Grundlage für den Prototypen und sollte so detailliert sein, dass der User Flow klar ist (Knapp u. a. 2016, S. 148-156). Knapp betont in diesem Schritt das Mantra “When in doubt, take risks.”: Der Design Sprint ist eine gute Möglichkeit, riskantere Entscheidungen zu testen.

Der Donnerstag ist dem Prototyping gewidmet. Hier wird auf Basis des erstellten Storyboards ein Klick-Dummy, bzw. Throwaway Prototype erstellt. Dieser Prototyp soll sich für Testnutzer am Freitag echt anfühlen, muss aber nur die Funktionalität abbilden, die am Freitag wirklich getestet wird. Umso mehr sich der Prototyp wie ein echtes Produkt anfühlt, desto mehr werden die Testnutzer authentisch reagieren (Knapp u. a. 2016, S. 168-170).

Das Prototyping geschieht in Teamarbeit. So erstellen beispielsweise zwei Personen einzelne Komponenten oder Screens und eine Person kombiniert diese zu einem einheitlichen Prototypen. Zusätzlich dazu ist eine Person für das Schreiben von Texten und eine oder mehr Personen für das Sammeln von Assets wie Bilder und Icons verantwortlich (Knapp u. a. 2016, S. 171).

Die Person, die am nächsten Tag die Interviews mit Testnutzern durchführen wird, konzipiert währenddessen ein Skript mit dem die Nutzer durch den Prototypen geführt werden (Knapp u. a. 2016, S. 171).

Am Freitag wird der Prototyp an fünf Nutzern getestet. Nach einer kurzen Einleitung und generellen Fragen zum Nutzer wird dieser durch den Prototypen geführt. Der Interviewer versichert den Nutzern dabei, dass nicht sie, sondern das Produkt, getestet werden. Sie können daher nichts falsch machen und es ist in Ordnung, wenn sie eine Aufgabe nicht verstehen oder ausführen können.

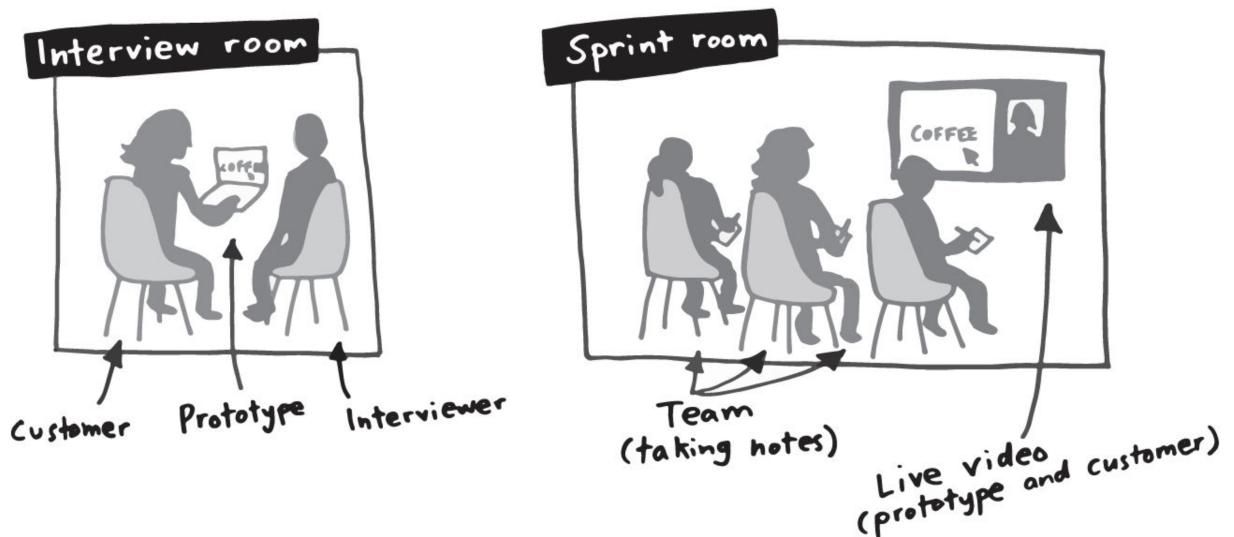


Abb. 0.5

(Knapp u. a. 2016, S. 174)

Interviewer und Testnutzer sitzen beim Interview zu zweit in einem Raum. Das Gespräch wird per Video an den Rest des Sprint-Teams übertragen, der in einem anderen Raum sitzt. Diese Personen machen sich während des Interviews in stiller Einzelarbeit auf Post-its Notizen zu dem Feedback des Testnutzers (Knapp u. a. 2016).

Nach den Interviews werden die Notizen sortiert und gruppiert, sodass sich daraus Schlüsse über das weitere Vorgehen schließen lassen können. Im Team wird nun mit Blick auf die am Montag definierten Sprintfragen und Langzeitziele über das weitere Vorgehen diskutiert. Der Decider hat auch hier das letzte Wort.

Design Sprints werden von vielen Unternehmen nicht 100% so durchgeführt wie es in dem Buch *Sprint* (Knapp u. a. 2016) vorgeschlagen wird. Die Design Sprint Agentur AJ&Smart¹ bietet beispielsweise einen Design Sprint in vier Tagen unter dem Namen “Design Sprint 2.0” an. Hier werden die ersten beiden Sprinttage in einen Tag zusammengeführt und die Übungen so angepasst, dass sie in der kürzeren Zeit durchgeführt werden können. So werden beispielsweise einzelne Experteninterviews in ein Gruppeninterview zusammengeführt, das auf 30 Minuten beschränkt ist (Smart u. a. 2018).

Bei Crisp Studio, einer Design Sprint Agentur aus Aachen², orientieren wir uns am vier-tägigen Sprint von AJ&Smart. Dazu tauschen wir die Reihenfolge einiger Übungen, da sie so aus unserer Sicht mehr Sinn ergeben. Wir legen beispielsweise die Sprintfragen an den Anfang des ersten Sprinttages, sodass alle Teammitglieder bei den Sprintfragen und dem Langzeitziel bereits auf dem gleichen Stand sind. Zusätzlich enthalten unsere Sprints neue Übungen, wie z.B. Note-n-Map (siehe Cruchon 2019), die den Prozess weiter vereinfachen. In unseren Design Sprints sind nicht alle Teammitglieder an allen vier Tagen anwesend. Die Teammitglieder des Kunden sind hier nur an den ersten beiden Tagen anwesend. Dies vereinfacht die Organisation des Sprints erheblich, da unsere Kunden sich nicht eine ganze Woche freihalten müssen. Das Prototyping und

¹<https://ajsmart.com>

²<https://crisp.studio>

Testing wird von der Agentur übernommen (Nauheimer 2019).

0.5.4 Prototyping

Prototypen sind in ihrer Grundform im Duden als “[...] zur Erprobung und Weiterentwicklung bestimmte erste Ausführung [...]“ definiert. Das Wort stammt vom Griechischen “prótótypos”, übersetzt “ursprünglich”, ab (Dudenredaktion (o. J.) 2019). Prototypen dazu eingesetzt, Ideen mit minimalem Aufwand zu testen. Prototyping bezeichnet die Erstellung eines Prototypen.

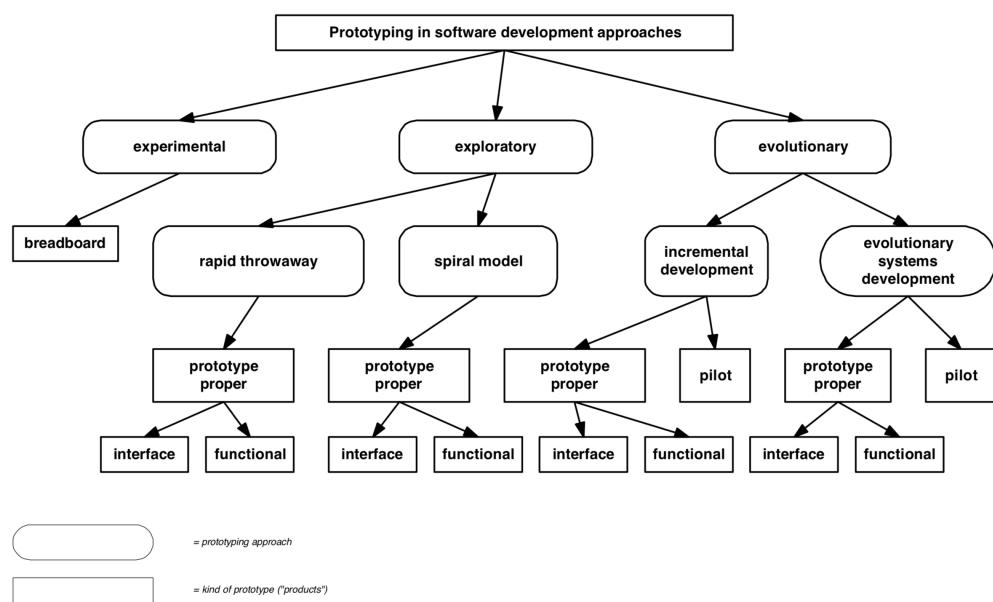
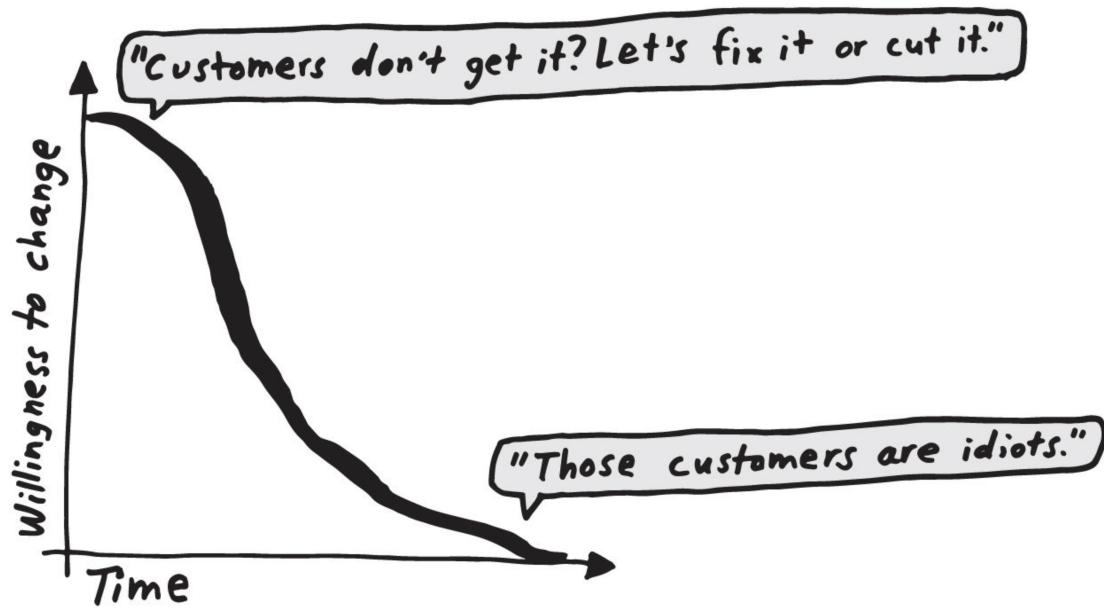


Abb. 0.6: Kategorisierung der Arten von Prototyping [Verner u. a. (o.D.), S. 7]

Im Kontext der Softwareentwicklung lässt sich Prototyping grob in drei Kategorien einteilen: Experimentelles, erforschendes und evolutionäres Prototyping (Verner u. a. o.D., S. 7). Diese Einteilung ist nicht absolut und die Begriffe variieren zwischen Quellen. So kann beispielsweise dem evolutionären Prototyping auch das Rapid Application Development (RAD) zugeordnet werden, das als Begriff erstmals 1988 von Barry Boehm (Boehm 1988) im Kontext seines Spiral Models vorgestellt wurde. Dieser Begriff wurde jedoch auch durch das 1991 von James Martin vorgestellte Modell (Martin 1991) geprägt, welches sich im Ansatz von Boehms Modell unterscheidet.

Der Prototyping-Schritt in Design Sprints ist ein Rapid Throwaway Prototyping oder kurz

Throwaway Prototyping, das sich dem erforschenden Prototyping unterordnen lässt. Bei diesem Ansatz werden Prototypen mit Fokus auf Geschwindigkeit entwickelt. Die Qualität des Programmcodes und die Sorgfältigkeit der Entwicklung sind dabei nicht wichtig, da der Prototyp nach der Evaluation nicht weiterentwickelt, sondern verworfen wird (Crinnion 1992, S. 18).



u. a. 2016)

(Knapp

Durch die Geschwindigkeit der Entwicklung fällt es dem Team meist leichter, Ideen zu verwerfen, da es so noch nicht zu viel Zeit in die Entwicklung dieser investiert hat (Knapp u. a. 2016).

0.5.5 Minimum Viable Products

Ein Minimum Viable Product, kurz MVP, auf Deutsch "minimal überlebensfähiges Produkt" (*Minimum Viable Product* 2019), ist laut Eric Ries ein Produkt, das dem Entwicklerteam mit dem geringsten Aufwand die maximale Menge an validiertem Wissen über seine Kunden bietet (Ries 2009).

MVPs werden eingesetzt um zu erforschen, wie der Zielmarkt auf ein potentielles Produkt oder ein neues Feature bei bestehenden Produkten reagieren würde. So wird

das Risiko minimiert, Zeit in ein die Entwicklung eines Produktes oder Features zu investieren, das letztendlich nicht genutzt wird und sich nicht verkaufen lässt.

Die Definition von “Minimum” im Kontext des MVP ist laut Ries nicht festgelegt. So kann der Entwicklungsaufwand je nach Produkt zwischen wenigen Tagen und mehreren Monaten liegen (Ries 2009).

Ähnlich wie im Fall des Begriffs “Prototyping” gibt es auch für das MVP verschiedene Definitionen. Der Begriff wurde erstmals 2001 von Frank Robinson als “[...] unique product that maximizes return on risk for both the vendor and the customer”³ (Robinson 2001) definiert. Steve Blank definierte MVPs 2010 als “[...] Customer Development tactic to reduce engineering waste and to get product in the hands of Earlyvangelists soonest”⁴ (Blank 2010).

Ries’ Definition hat aktuell jedoch die größte Reichweite. Die meisten Definitionen bauen entweder auf seiner Definition oder auf der Definition von Steve Blank auf (Lenarduzzi u. a. 2016, S. 119).

0.5.6 Design Systems

0.5.7 Atomic Design

Atomic Design ist ein System zur hierarchischen Organisation und Strukturierung von UI Komponenten, das erstmals 2013 von Brad Frost auf seinem Blog vorgestellt und 2016 in dem Buch “Atomic Design” veröffentlicht wurde (Frost 2016, Foreword).

Das in seinem Buch beschriebene System bezieht sich hauptsächlich auf die Entwick-

³Übersetzt: “einzigartiges Produkt, das den Return on Risk sowohl für den Lieferanten als auch für den Kunden maximiert”

⁴Übersetzt: “Kundenentwicklungs-Taktik, um technische Verschwendungen zu reduzieren und das Produkt schnellstmöglich in die Hände von Earlyvangelists zu bekommen”

ABSTRACT

CONCRETE

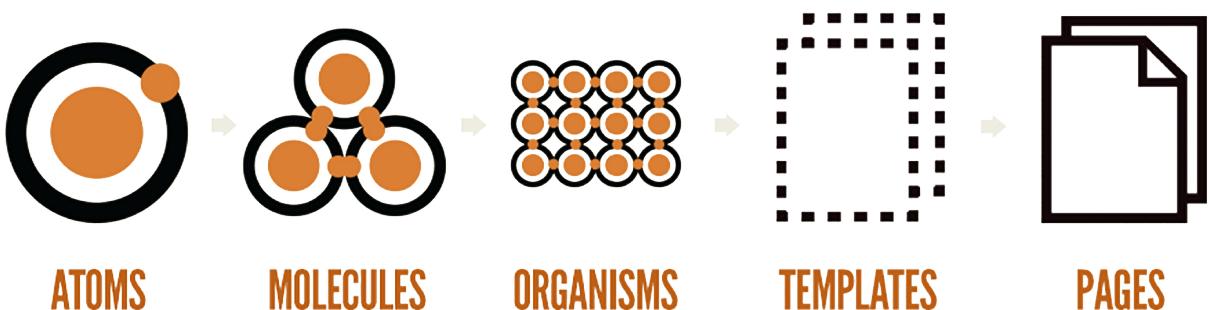


Abb. 0.7: Struktur des Atomic Designs [Frost (2016), K. 2]

lung von Webseiten, lässt sich jedoch auch für jede andere Form von Benutzerinterfaces einsetzen. Atomic Design unterteilt die eingesetzten UI Elemente in Atome, Moleküle, Organismen, Templates und Seiten. Komponenten aus diesen Kategorien bauen jeweils auf Komponenten aus der vorherigen Kategorie auf (Frost 2016, K. 2).

Atome sind im Atomic Design die kleinste Einheit der Komponenten und können nicht weiter in kleinere Teile zerlegt werden. Gültige Atome sind beispielsweise Buttons, Eingabefelder, Labels, Überschriften und alle Elemente, die sich durch einen HTML Tag darstellen lassen (Frost 2016, K. 2).

Moleküle bauen auf diesen Atomen auf und Kombinieren diese zu Einheiten, die einen bestimmten Zweck erfüllen. So könnte ein Molekül "Suchfeld" beispielsweise aus den Atomen "Button" und "Eingabefeld" bestehen. Dieses Molekül kann nun überall verwendet werden, wo ein Suchfeld benötigt wird.

Organismen sind komplexe Komponenten, die aus mehreren Atomen, Molekülen und anderen Organismen bestehen. Ein gültiges Molekül ist beispielsweise ein Header. Dieser könnte aus einem Logo-Atom, einem Menü-Molekül und einem Suchfeld-Molekül bestehen.



Abb. 0.8: Header-Organismus [Frost (2016), K. 2]

Templates bestehen aus mehreren Organismen und definieren das Layout in dem diese auf der Seite angeordnet werden sollen. Diese Templates sind unabhängig von Inhalten und arbeiten mit Platzhaltern für diese um die Inhaltsstruktur zu definieren (Frost 2016, K. 2).

Seiten sind mit Inhalten gefüllte Templates. Sie repräsentieren den fertigen Zustand der Website und lassen sich gut dazu verwenden um sicherzustellen, dass die Inhalte in den definierten Templates gut aussehen. So können eventuelle Probleme wie zu lange Überschriften oder unzureichende Bildformate behoben werden, indem entweder das Template oder die Inhalte angepasst werden (Frost 2016, K. 2).

Die von Frost vorgestellte Hierarchie ist nicht linear. So sollten beispielsweise nicht erst alle Atome gestaltet werden, bevor die Gestaltung der Moleküle beginnt, da dies zu mit hoher Wahrscheinlichkeit nicht zu einem einheitlichen Design führen wird (Frost 2016, K. 2).

Durch den Einsatz von Atomic Design ist es einfacher, ein Benutzerinterface sowohl als Ganzes zu betrachten, als auch sich auf einzelne Komponenten zu konzentrieren. Die hierarchische Struktur trägt dazu bei, dass Komponenten nicht versehentlich in verschiedenen Kontexten mehrmals gestaltet werden und ermöglicht dadurch spätere Anpassungen an einzelnen Komponenten, die sich konsequent auf das gesamte Design auswirken (Frost 2016, K. 2).

Benutzerinterfaces, die auf dieses System setzen, sehen meist einheitlicher aus und sind dadurch für Benutzer einfacher zu nutzen. Sie lassen sich dazu in Zukunft schneller um neue Komponenten erweitern ohne dass das Entwicklungs- und Designteam dabei die Übersicht verliert (Frost 2016, K. 4).

Frost schlägt in seinem Buch vor, die entwickelten Komponenten in einer Komponentenbibliothek zu sammeln, die direkt mit dem fertigen Produkt verbunden ist, sodass die dort enthaltenen Komponenten immer den aktuellen Zustand der Website abbilden. So werden in Zukunft Änderungen an den Komponenten direkt auf das Produkt angewendet und es kommt nicht zu unerwarteten Unterschieden zwischen Design und Code (Frost

2016, K. 3).

0.5.8 Rolle des Component Sprints

0.6 Ausarbeitung des Konzepts für Component Sprints

0.7 Empirischer Test des entwickelten Konzepts

0.7.1 Ablauf

0.7.2 Ergebnisse

0.7.3 Erkenntnisse / Learnings

0.8 Qualitative Studie zur Verfeinerung des Konzepts

0.8.1 Studiendesign

Begründung

Auswahl der Interviewpartner

Entwicklung des Interviewleitfadens

Durchführung der Interviews

- Jeremy (Head of Design Experience) hat viel Design Sprint Erfahrung bei Trivago

- Ulf (UX Engineer)
- Angeli (Product Owner)

0.8.2 Ergebnisse

0.9 Weiteres Vorgehen

Literatur

- Blank, Steve (4. März 2010). *Perfection By Subtraction – The Minimum Feature Set*. URL: <https://steveblank.com/2010/03/04/perfection-by-subtraction-the-minimum-feature-set/> (besucht am 13.08.2019).
- Boehm, Barry W (1988). „A Spiral Model of Software Development and Enhancement“. In:
- Crinnion, John (1992). *Evolutionary Systems Development: A Practical Guide to the Use of Prototyping Within a Structured Systems Methodology*. Perseus Publishing. ISBN: 0-306-44139-X.
- Cruchon, Stéphane (2. Mai 2019). *The Design Sprint Note-n-Map*. URL: <https://sprintstories.com/the-design-sprint-note-n-map-a9bf0ca88f51> (besucht am 24.07.2019).
- Dudenredaktion (o. J.) (2019). *Prototyp*. In: *Duden online*. URL: <https://www.duden.de/node/115811/revision/115847> (besucht am 30.07.2019).
- Frost, Brad (2016). *Atomic Design*. OCLC: 971562254. ISBN: 978-0-9982966-0-9.
- Google Ventures (2019). *The Design Sprint — GV*. URL: <http://www.gv.com/sprint> (besucht am 16.07.2019).
- Kahneman, Daniel (2013). *Thinking, Fast and Slow*. Penguin.
- Knapp, Jake (30. Okt. 2018). „How to Stop Wasting Time—Jake Knapp at Amplify“. Self Improvement. URL: <https://www.slideshare.net/amplitudemobile/how-to-stop-wasting-time-jake-knapp-at-amplify> (besucht am 19.08.2019).
- Knapp, Jake, John Zeratsky und Braden Kowitz (2016). *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. First Simon & Schuster hardcover edition. OCLC: ocn913304205. New York: Simon & Schuster. 274 S. ISBN: 978-1-5011-2174-6.

- Lenarduzzi, V. und D. Taibi (Aug. 2016). „MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product“. In: *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), S. 112–119. DOI: 10.1109/SEAA.2016.56.
- Martin, James (1991). *Rapid Application Development*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc. ISBN: 0-02-376775-8.
- Minimum Viable Product* (6. Aug. 2019). In: *Wikipedia*. Page Version ID: 191100310. URL: https://de.wikipedia.org/w/index.php?title=Minimum_Viable_Product&oldid=191100310 (besucht am 15.08.2019).
- Nauheimer, René (23. Mai 2019). *Mister Spex – Rethinking The Way We Buy Eyewear Today*. URL: <https://blog.crisp.studio/mister-spex-%E2%80%93-rethinking-the-way-we-buy-eyewear-today/> (besucht am 24.07.2019).
- Ries, Eric (3. Aug. 2009). *Minimum Viable Product: A Guide*. URL: <http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html> (besucht am 13.08.2019).
- Robinson, Frank (2001). *Minimum Viable Product / SyncDev*. URL: <http://www.syncdev.com/minimum-viable-product/> (besucht am 13.08.2019).
- Smart, Michael und Jonathan Courtney (14. Feb. 2018). *Design Sprint 2.0 / What Is Google Design Sprint Process and Framework*. URL: <https://ajsmart.com/design-sprint-2-0/> (besucht am 24.07.2019).
- Verner, June und Mahil Carr (o.D.). „Prototyping and Software Development Approaches“. In: () .