Fachhochschule Frankfurt am Main
University of Applied Sciences

Fachhochschule Frankfurt am Main
University of Applied Sciences
Fachbereich 2: Informatik und Ingenieurwissenschaften

Bachelorarbeit

Thema: User Centered Design und agile Entwicklungsmethoden

Zur Erlangung des akademischen Grades: „Bachelor of Science“

Vorgelegt von: Mario Pohlmann
Matrikelnummer: 855019
Studiengang: Informatik (B.Sc.)

Referent: Prof. Dr. Erwin Hoffmann
Korreferent: Prof. Dr. Matthias Wagner

In Zusammenarbeit mit

cocomore

Cocomore AG
Gutleutstr. 30
D-60329 Frankfurt

Inhaltsverzeichnis

Kapitel

EINS

Einleitung

Diese Bachelorarbeit beschreibt eine mögliche Integration des User Centered Design (UCD)-Prozesses in agile Methode am Beispiel Scrum. Im Rahmen einer Überarbeitung der DIN EN ISO 9241 wurde der Prozess 2006 von „User Centered Design“ in „Human Centered Design“ umbenannt. In der Literatur wird jedoch noch häufig der Begriff „User Centered Design“ verwendet. Da beide Begriffe analog genutzt werden wurde auf die, zur Zeit noch, ungebräuchliche Verwendung des offiziellen Prozessnamens verzichtet.

Im ersten Teil dieser Arbeit wird der UCD-Prozess und einige der darin verwendeten Methoden vorgestellt. Der zweite Teil erläutert den Scrum-Prozess. Im abschließenden dritten Teil wird versucht, eine mögliche Verbindung der beiden Prozesse zu zeigen und an Hand eines Beispiels erläutert.

Für die Erstellung dieser Bachelorarbeit wurde der englische Scrum-Guide benutzt. Die jeweiligen Begriffe für Meetings, Rollen, etc. wurden daher nicht übersetzt.

Kapitel

ZWEI

Motivation

Die Cocomore AG wurde durch einen Kunden beauftrag eine E-Commerce-Seite neu zu entwickeln. Der Kunde vertritt über 3000 aus metallverarbeitenden Unternehmen.
Auf Grund von Größe und Umfang des Projektes (geschätzt etwa 900 Personentage) wurde zusammen mit dem Auftraggeber vereinbart, das Projekt mit Scrum als Projektmanagement-Framework umzusetzen.
Da das Produkt, neben dem technischen Teil (Verfügbarkeit & Performance), für die Benutzer leicht zu bedienen sein sollte, wurde bereits in der Angebotsphase die Integration des Bereiches UCD empfohlen. Der UCD-Bereich der Cocomore AG ist zuständig für die konzeptionelle Entwicklung von benutzerfreundlichen Bedienkonzepten. Der Kunde (Kunde) folgte dieser Empfehlung.

2.1. Gründe für den Einsatz von Scrum

Der Kunde hatte in der Vergangenheit mit mindestens drei verschiedenen Agenturen ein Redesign seines E-Commerce-Angebots durchgeführt. Die Ergebnisse waren nicht das was sich der Kunde erhofft hatte. Es sollte eine Performance-Steigerung sowie eine Erhöhung der Besucherzahlen erreicht werden. Beide Anforderungen konnten in den vorangegangenen Redesigns nicht erfüllt werden.
In den vorangegangenen Redesigns wurde auf klassische Projektmanagement-Frameworks zurückgegriffen. Der Kunde war von der Idee Scrum zu nutzen, und den Entwicklungsprozess aktiv zu begleiten und mitzugestalten zu können, leicht zu überzeugen. In den vorherigen Redesigns konnte der Kunde erst sehr spät Entwicklungsergebnisse überprüfen und aufgetretene Probleme zögerten eine Fertigstellung immer hinaus.
Ein weiterer Grund für den Einsatz von Scrum lag in dem großen Projektumfang (geschätzt etwa 900 Personentage). Der zuständige Projektmanager, sowie der Leiter des IT-Bereiches der Cocomore AG, schlugen dem Kunden auch aus diesem Grund Scrum als Projektmanagementtool vor. Durch die Verwendung von Scrum konnte der Kunde ständig den Fortschritt des Projektes überprüfen und bei Problemen eingreifen.
Der Einsatz linearer Entwicklungsmethoden hätte aus Sicht des Projektmanagers

und des IT-Leiters evtl. zu einer Wiederholung der Probleme aus den vorangegangenen Redesigns geführt.

2.2. Hintergrund der Verwendung von User Centered Design

Die durch den Kunden vorgegebenen Anforderungen an die Bedienoberfläche des Redesigns waren zu Beginn nur sehr grob formuliert:

1. Erhöhung der Besucherzahlen des E-Commerce-Angebots
2. Allgemeine Verbesserung der Suchfunktion
3. Optimierung der Darstellung von Suchergebnissen
4. Reduzieren der notwendigen Schritte um zu einem Ergebnis zu kommen

Auf Grund der Komplexität der Produktsuche und des verwendeten Filters wurde eine Ausrichtung auf technisch sehr versierte Nutzer festgestellt. Eine während der Angebotsphase durchgeführte Evaluation an Hand von Usability-Kriterien ergab, dass die Produkt- und Firmensuche für den Gebrauch in einem Arbeitsumfeld zu langwierig und komplex war.
Zudem bestand in der, zum Zeitpunkt der Angebotserstellung aktuellen, Produktverwaltung das Problem, dass ca. 1300 produktabhängige Optionen bei der Produkteingabe existierten.
Nachdem dem Kunden diese Problematik erklärt wurde, sollte zusätzlich zum Scrum-Prozess der UCD-Prozess eingesetzt werden.

2.3. Einbindung von User Centered Design in Scrum

Die Notwendigkeit, ein für die Cocomore AG praktikables Vorgehen zu entwickeln stellte sich, da durch den Scrum-Prozess nicht nach klassischem UCD-Modell gearbeitet werden konnte. Das klassische Vorgehen bei UCD besteht zwar wie auch bei Scrum aus iterativen Phasen, jedoch ist das Ziel von UCD die Erzeugung eines vollständigen Konzeptes für ein Produkt. Auch nach umfassende Recherche durch den Autor, und weitere an der Entwicklung beteiligte Personen, konnte kein Vorgehensmodell gefunden werden, das die Integration der beiden Prozesse beschrieb. Bücher und Online-Quellen beschrieben zwar die Integration einzelner, im UCD-Prozess angewandter Methoden, jedoch wurde auch vielfach von negativen Erfahrungen bei der Kombination der beiden Prozesse berichtet.

Das UCD-Team arbeitet nach den Richtlinien der DIN EN 9241 und verwendet somit den UCD-Prozess. Da Scrum keine echten Vorgaben zu der Integration eines weiteren Prozesses macht und die Cocomore AG in Zukunft Projekte vermehrt nach Scrum umsetzen will, entschied man sich ein generelles Vorgehen für die Integration des UCD-Prozesses in Scrum zu entwickeln.
Die Entwicklung wurde durch zwei im Unternehmen arbeitende Scrum Master sowie einen Senior Information Architect unterstützt.

Kapitel

DREI

User Centered Design

3.1. Was ist User Centered Design?

User Centered Design (UCD) ist ein Prozess um eine möglichst hohe positive User Experience (Nutzungserlebnis) zu erreichen. Die User Experience wird durch die Usability in starkem Masse beeinflusst.
Bei der Konzeption von Produkten nach UCD wird der Benutzer in den Mittelpunkt des Entwicklungsprozesses gestellt. Es wird dabei versucht, den Benutzer während der gesamten Entwicklung eines Projektes in den Entwicklungsprozess einzubinden und die Benutzeroberfläche und Interaktionen so zu optimieren, dass die fertige Anwendung den Nutzer in seiner Aufgabe optimal unterstützt.

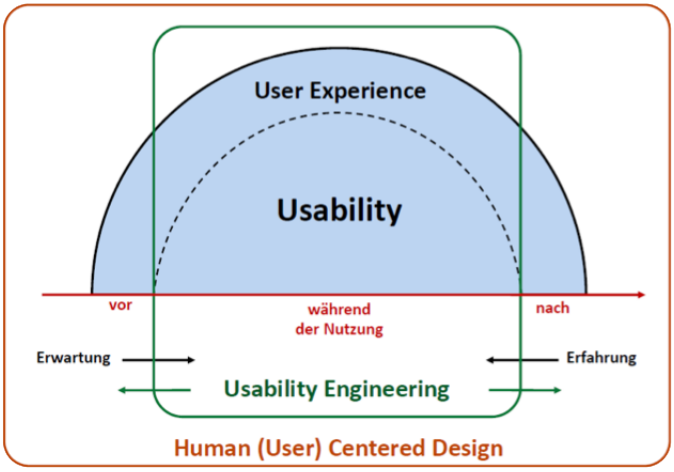

Abbildung 3.1.: Zusammenhang zwischen User Centered Design, Usability Engineering, User Experience und Usability (Quelle: artop GmbH)

Der äußere orange Rahmen symbolisiert die Anforderungen an einen benutzerorientierten Entwicklungsprozess (User Centered Design). Der grüne Rahmen steht für das Usabiltiy Engineering, das einen Ausschnitt des User Centred Design bildet. Usability Engineering konzentriert sich zwar auf das Überprüfen und Optimieren der Usability des Produktes („während der Nutzung“), ist jedoch nicht unabhängig

davon, was vor und nach der Nutzung passiert: bereits generierte Erwartungen und Vorerfahrungen sowie während der Nutzung neu gewonnene Erfahrungen spielen ebenfalls eine wichtige Rolle.
User Centered Design bietet den Rahmen in dem die Methoden des Usability Engineering angewandt werden.

3.2. Die vier Phasen des User Centered Design

Der UCD-Prozess gliedert sich in vier Phasen, welche auch in der DIN EN ISO 9241-210 beschrieben sind:

1. Analyse des Nutzungskontextes
2. Definition der Anforderungen
3. Konzeption und Entwurf
4. Evaluation

Der UCD-Design-Prozess ist iterativ. Wie auch bei agilen Entwicklungsmethoden sollte ein auf UCD basierendes Konzept auf Basis eines Prototypen (siehe 3.2.3 Konzeption & Entwurf) in der Evaluationsphase getestet und bewertet werden. Solange in der Evaluationsphase Probleme hinsichtlich der Benutzbarkeit festgestellt werden, sollte der Prozess erneut durchlaufen werden.

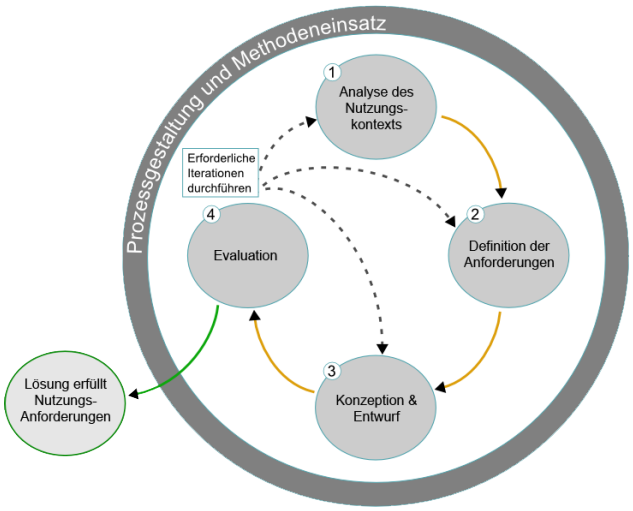

Abbildung 3.2.: Wechselseitige Abhängigkeit nach DIN EN ISO 9241-210 (Quelle: DIN EN ISO 9241-210)

Im weiteren Verlauf dieses Kapitels werden die einzelnen Phasen kurz erläutert und eine kleine Auswahl an gebräuchlichen Methoden vorgestellt. Einige der hier vorgestellten Methoden werden in späteren Kapiteln ausführlicher beschrieben, da sie für die Integration des UCD in Scrum genutzt werden können.

3.2.1. Analyse des Nutzungskontextes

Diese erste Phase des UCD befasst sich mit der Sammlung und Zusammenfassung von Informationen über die späteren Nutzer.

Zu diesem Zweck werden Nutzer aus der Zielgruppe in Interviews und Gesprächsgruppen, so genannten Fokusgruppen, zu ihren persönlichen Anforderungen an die zu entwickelnde Anwendung befragt. Die Ergebnisse dieser Gesprächsrunden werden dokumentiert und fließen, z.B. als Affinity-Diagramme oder Personas (3.2.2 Definition der Anforderungen), in die weitere Konzeption ein.

Contextual Inquiries - Bei einer Contextual Inquiry wird der Benutzer in seiner typischen Umgebung beobachtet. Während der Beobachtung führt der Benutzer definierte Aufgaben durch. Im Anschluss an die Beobachtung findet ein Interview statt. Die Fragestellung des Interviews zielt dabei auf ausgewählte Themen zu Einsatz und Nutzungsumfeld von benutzten Produkten ab.

Fokusgruppen - Fokusgruppen sind moderierte Gesprächsrunden mit Benutzern. Die Teilnehmer diskutieren über mögliche Probleme und Lösungen einer Anwendung.
Fokusgruppen dienen nicht alleine der Analyse des Nutzungskontextes, sondern können in späteren Phasen eines Projektes auch zur Evaluation eingesetzt werden.

3.2.2. Definition der Anforderungen

Auf Grund der in Abschnitt 3.2.1, Analyse des Nutzungskontextes, gesammelten Nutzerinformationen werden im weiteren Verlauf die eigentlichen Projektanforderungen definiert.

Zur Definition der Anforderungen existieren verschiedene Methoden. Auf Grund der Vielfalt und Komplexität der Methoden können hier nur drei exemplarisch vorgestellt werden:

Affinity-Diagramme - Aus den während der Analysephase erstellen Protokollen der Gesprächsgruppen und Contextual Inquiries werden die von den Nutzern benannten Anforderungen und Probleme an die zu entwickelnde Software auf Affinity Notes (Post-Its) geschrieben. Die Post-Its werden in verschiedenen Stufen gruppiert und benannt. Diese so genannte Affinity Wall bleibt während der gesamten Konzeptionsphase bestehen und kann dabei verändert oder erweitert werden.

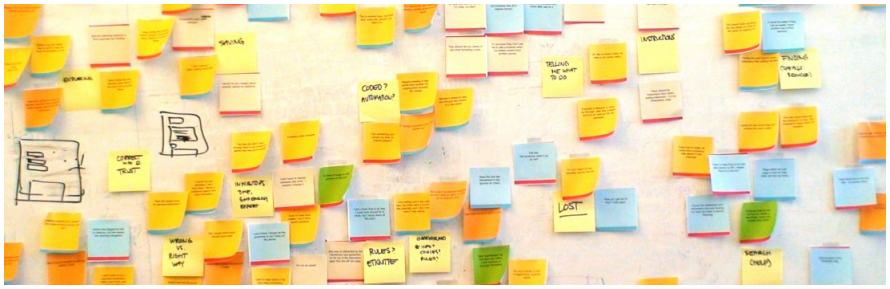

Abbildung 3.3.: Affinity-Diagramm (Quelle: ba-wiinf.spookee.de)

Mental Model - Die Erstellung eines Mental Model ist ähnlich der von Affinity-Diagrammen, jedoch werden Mental Models nicht mit Hilfe von Post-Its, sondern in einem Tabellenprogram (z.B. Excel) erstellt. Die einzelnen Anforderungen werden in die Tabelle eingetragen und anschließend thematisch gruppiert. Wie auch schon bei den Affinity-Diagrammen werden die jeweiligen Gruppen durch eine zusammenfassende Bezeichnung benannt. Die jeweiligen Gruppen werden anschließend erneut nach ihrer Bezeichnung gruppiert und ebenfalls wieder benannt. Im Gegensatz zu den Affinity-Diagrammen ist hier die Benennung nicht abstrakt sondern spiegelt die Motivation des Nutzers an die darunter liegenden Anforderungen wieder. Zur visuellen Aufbereitung werden die jeweiligen Gruppen in eine Säulendarstellung überführt. Zum Abschluss werden zu jeder Säule die zur Erfüllung der Anforderungen notwendigen Funktionen hinzugefügt. Der Aufbau des Mental-Model-Diagramms ist dabei in einen oberen (Anforderungen) und einen unteren (Funktionen) Bereich geteilt. Bereits vorhandene Funktionen werden farblich anders dargestellt als noch zu entwickelnde Funktionen.

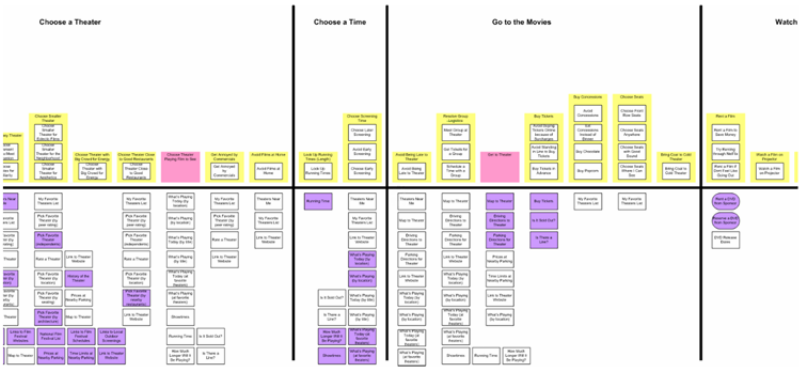

Abbildung 3.4.: Grafische Darstellung eines Mental Model (Quelle: Indi Young, Präsentation auf MSDN-Blogs)

Personas - Personas sind textliche Beschreibungen eines archetypischen Nutzers. Seine Eigenschaften werden in der Regel durch Einzelinterviews, Umfragen und/oder Benutzertests herausgefunden. Um eine ausreichende Basis an empirischen Informationen zu haben sollten mindestens 10 - 12 Nutzer befragt werden, bei größeren Projekten jedoch mehr. Auf Basis der gewonnen Informationen werden üblicherweise 2 - 5 Personas erstellt. Sie repräsentieren im späteren Verlauf der Entwicklung die Bedürfnisse und Wünsche der primären und sekundären Zielgruppen.

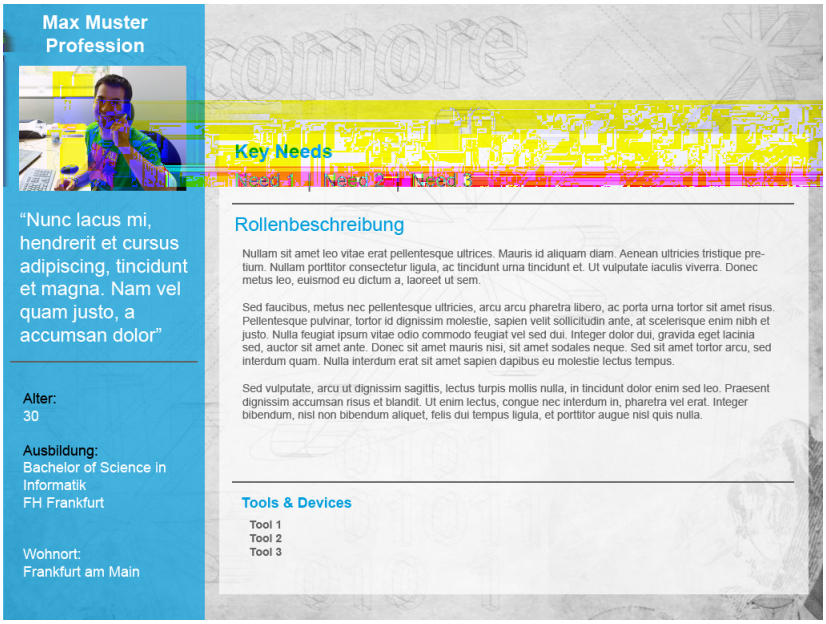

Abbildung 3.5.: Beispielhafter Aufbau einer Persona

3.2.3. Konzeption & Entwurf

Zu Beginn der Konzeption werden mögliche Use Cases und User Journeys erstellt die verdeutlichen, wie ein Benutzer das zu entwickelnde System verwendet. Hierbei werden bereits erste Interaktionen und Funktionen auf Grundlage der in Abschnitt 3.2.2, Definition der Anforderungen, festgelegten Anforderungen definiert.
Auf Basis der Use Cases werden anschließend erste Wireframes erstellt und nach Möglichkeit auch getestet. Wireframes stellen dabei die Struktur und Interaktionen der späteren Anwendung dar.
Wireframes können entweder auf Papier (handgezeichnet) oder mit Hilfe einer speziellen Software (z.B. Axure, Balsamiq Mockups) erstellt werden.
Neben möglichen Tests mit Benutzern dienen Wireframes auch als Diskussionsgrundlage innerhalb des Projektteams. Auf Basis der Wireframes und der Use Cases werden die späteren Abläufe, Nutzerinteraktionen und Navigation, in einem Fachkonzept definiert und ein Prototyp[1] erstellt.
Abhängig von der Projektplanung können bereits mit einer Rohversion des Konzeptes und des Prototypen erste Usability-Test (siehe Evaluation) durchgeführt werden.

Die Konzeption einer auf möglichst hohe Usability bedachten Anwendung basiert im wesentlichen auf der DIN EN ISO 9241, Ergonomie der Mensch-System-Interaktion. Für die Konzeption sind die Teile 11, Anforderungen an die Gebrauchstauglichkeit, und 110, Grundsätze der Dialoggestaltung, essentiell. Der Vollständigkeit halber werden beide Teile der DIN-Norm hier kurz vorgestellt.

[1] Ein Prototyp im UCD ist nicht zwangsläufig eine programmierte Oberfläche. Es kann sich dabei auch um einen Papier-Prototypen handeln.

3.2.3.1. DIN EN ISO 9241-11

Die DIN EN ISO 9241-11, Anforderungen an die Gebrauchstauglichkeit, definiert drei Leitkriterien in Abhängigkeit des jeweiligen Nutzungskontextes(2):

Effektivität - Die Genauigkeit und Vollständigkeit, mit der Benutzer ein bestimmtes Ziel erreichen
Fragestellung: Kann der Benutzer die vorliegende Aufgabe mit dem System möglichst vollständig und korrekt erfüllen?

Effizienz - Der im Verhältnis zur Genauigkeit und Vollständigkeit eingesetzte Aufwand, mit dem Benutzer ein bestimmtes Ziel erreichen
Fragestellung: Kann der Benutzer die Aufgabe mit möglichst wenig Aufwand lösen?

Zufriedenstellung - Freiheit von Beeinträchtigung und positive Einstellung gegenüber der Nutzung des Produkts
Fragestellung: Ist der Benutzer mit dem System zufrieden?

Der Nutzungskontext ist dabei abhängig von den folgenden Kriterien:

- dem Benutzer,
- den Zielen,
- den Arbeitsaufgaben,
- den Arbeitsmitteln,
- der physischen und sozialen Umgebung

3.2.3.2. DIN EN ISO 9241-110

Die DIN EN ISO 9241-110, Grundsätze der Dialoggestaltung, definiert zusätzlich sieben weitere Richtlinien(2):

Aufgabenangemessenheit - Ein interaktives System ist aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h., wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie

Selbstbeschreibungsfähigkeit - Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog er sich befindet, welche Handlungen unternommen werden können und wie diese ausgeführt werden können

Steuerbarkeit - Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog er sich befindet, welche Handlungen unternommen werden können und wie diese ausgeführt werden können

Erwartungskonformität - Ein Dialog ist erwartungskonform, wenn er den aus dem Nutzungskontext heraus vorhersehbaren Benutzerbelangen sowie allgemein anerkannten Konventionen entspricht

Fehlertoleranz - Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann

Individualisierbarkeit - Ein Dialog ist individualisierbar, wenn Benutzer die Mensch-System-Interaktion und die Darstellung von Informationen ändern können, um diese an ihre individuellen Fähigkeiten und Bedürfnisse anzupassen

Lernförderlichkeit - Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet

3.2.4. Evaluation

Nachdem das Fachkonzept und ein Prototyp vorliegen, sollten damit Funktionen und Design auf Benutzerfreundlichkeit getestet werden. Die Ergebnisse dieser Tests entscheiden darüber, ob vorherige Entwicklungsstufen erneut durchlaufen werden und das Konzept überarbeitet werden muss.

Zur Evaluation eines Konzeptes haben sich verschiedene Methoden etabliert, hier eine Auswahl von drei gängigen Methoden:

Card Sorting - Card Sorting ist eine Evaluierungsmethode zur Feststellung einer geeigneten Menü- und Navigationsstruktur. Potentielle Benutzer erhalten Karteikarten mit Themenbereichen und sollen diese sinnvoll gruppieren. Anschließend werden die einzelnen Gruppierungen ausgewertet und die Menü- und Navigationsstruktur ggf. angepasst. Card Sorting kann bei Redesigns von Anwendungen auch in der Anforderungsdefinition eingesetzt werden.

Heuristische Evalutation - Die heuristische Evaluation setzt nicht auf Tests mit potentiellen Anwendern, sondern auf die Überprüfung einer Anwendung durch fünf Experten. Als Experten werden bei der heuristischen Evaluation Personen angesehen, die das zu testende System kennen. Die Experten versuchen anhand von Heuristiken nach Nielsen & Molich (siehe Anhang) potentielle Probleme hinsichtlich der Usability zu finden, diese zu klassifizieren und zu priorisieren.

Usability-Tests - Usability-Tests sind eine der aufwendigsten Evaluationsmethoden. Hier werden Benutzer aufgefordert definierte Aufgaben an dem Prototypen/Produkt abzuarbeiten. Während der Aufgabenbearbeitung werden die Benutzer, ihre Handlungen und Äußerungen aufgezeichnet und später ausgewertet. Formale Usability-Tests werden üblicherweise in spezialisierten Unternehmen durchgeführt da die Einrichtung eines Testlabors sehr aufwendig und kostenintensiv ist.

Zusätzlich zu der Evaluation durch Benutzer und/oder Experten sollte das Konzept und der Prototyp auch von anderen Projektbeteiligten auf die spätere Machbarkeit hin überprüft werden.

3.3. Vorteile von User Centered Design

UCD bietet Auftraggebern und Entwicklern die Möglichkeit Produkte zu entwickeln, die auf die Bedürfnisse der Kunden abgestimmt sind. Neben der besseren Usability eines Produktes kann UCD allerdings auch helfen Kosten in der späteren Entwicklungsphase zu reduzieren. Auf Grund von detailierten Anforderungen reduziert sich die Wahrscheinlichkeit, dass zu einem späteren Zeitpunkt tiefgehende Änderungen am eigentlichen Designkonzept vorgenommen werden müssen.

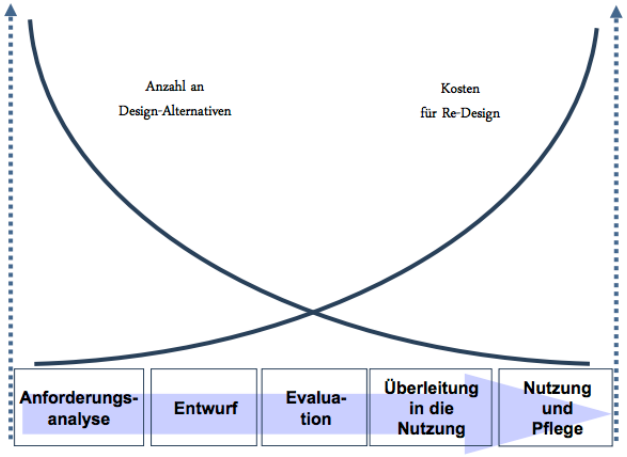

Abbildung 3.6.: Auswahl der richtigen Designalternative (Quelle: artop GmbH)

In frühen Phasen der Konzeption ist ein Wechsel zwischen verschiedenen Designalternativen wenig aufwendig. Je weiter die Entwicklung fortschreitet, desto weniger Designalternativen stehen zur Verfügung. Zusätzlich steigen mit jeder Phase die Kosten für Änderungen. In der Literatur wird ein Verhältnis von etwa 1:6 für Änderungen in der Konzeptionsphase zu Änderungen in der Implementierungsphase geschätzt.

3.3.1. Wir könnten auch einfach mal die Nutzer fragen!

In vielen Fällen lässt sich schlechte oder unzureichende Usability auf den Verzicht der Einbeziehung von Benutzern zurückführen.
Ein gerne zitiertes Beispiel ist die Anpassung eines Online-Shops durch Jared M. Spool aus dem Jahr 2009 ("The $300 Million Button").[2]

Spool wurde durch den Anbieter eines Online-Shops beauftragt, bei dem Redesign des Checkout-Prozesses unterstützend mitzuwirken.
Während Usability-Tests mit Benutzern ergab sich, dass es Probleme bei der Authentifizierung des Accounts während des Checkouts auftraten, da sich die Käufer nicht mehr an ihr Passwort erinnern konnten. Das Problem betraf nicht nur Neu-, sondern auch Bestandskunden die ihren Account schon seit längerer Zeit hatten.

[2] Der vollständige Artikel von Jared M. Spool zu dem Projekt ist auf http://www.uie.com zu finden.

Zur Sicherheit validierte Spool die Daten aus den Benutzer-Tests mit Daten aus dem Website-Tracking des Auftraggebers:

Anzahl Passwort-Resets:	$\sim$ 40% der Benutzer
Ausgeführte Passwort-Resets:	$<$ 25%
Abgeschlossene Checkouts nach Passwort-Reset:	$<$ 20%

Berechneter Verlust durch Checkout-Abbrüche:	$\sim$ **$300.000.000/Jahr**

Auf Grund dieser Zahlen wurde ein Checkout-Prozess ohne Registrierungs- oder Login-Zwang entwickelt. Innerhalb der ersten Woche nach dem Release dieser Lösung stieg der Umsatz des Online-Shops um konstante $6.000.000. In der gleichen Zeit sanken Passwort-Resets um konstante 80%.
Durch die Änderung des Checkout-Prozesses konnte der Online-Shop seinen Jahresumsatz um $312.000.000 steigern. Die finanziellen Aufwände für eine Usability-Optimierung waren dagegen vergleichsweise gering.

Was ist passiert? Spool fand durch Benutzer-Tests und Nutzeranalyse eine der größten Schwachstellen des Online-Shops. Unter Beachtung der drei Leitkriterien der DIN EN ISO 9241-11, Anforderungen an die Gebrauchstauglichkeit (siehe 3.2.1), definierte er einen alternativen Checkout-Prozess, der die Anforderungen der Benutzer beachtete.

Kapitel

VIER

Agile Entwicklung

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktionen mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

(Agiles Manifest, 2001)

4.1. Agile Entwicklungsmethoden

Agile Entwicklungsmethoden basieren auf der Idee, durch häufige Iterationen die Entwicklungszeit für Projekte zu verkürzen und ein flexibles System zur Problemlösung zu benutzen. Anders als bei linearen Entwicklungsmethoden wird ein zu entwickelndes System nicht vollständig geplant bevor die eigentliche Umsetzung erfolgt. Viel mehr ist das Ziel agiler Entwicklung, dem Kunden nach jeder Iteration ein funktionierendes Produkt zu liefern.
Der Kunde wird bereits sehr früh in den Entwicklungsprozess eingebunden und begleitet diesen bis zum fertigen Produkt.

Seit den Anfängen der agilen Entwicklung haben sich verschiedene Methoden etabliert, einige der bekanntesten sind

1. Extreme Programming;
2. Feature Driven Development;
3. Crystal (eigentlich eine Methodensammlung);
4. Usability Driven Development;
5. Scrum

Agile Entwicklungsmethoden bieten einige Vorteile, sowohl für die Entwickler, als auch für den Kunden:

- Agile Entwicklungsmethoden lassen sich auf nahezu alle Projekte anwenden
- Frühe Beteiligung des Kunden am Entwicklungsprozess
- Nach jeder Iteration erhält der Kunde ein funktionsfähiges Produkt
- Mehr Freiheit für den Entwickler bei der eigentlichen Umsetzung / Problemlösung
- Auf Grund der Flexibilität können Änderungen am Projekt schnell und meist auch kostengünstig realisiert werden

Die bei linearen Entwicklungsmethoden teilweise sehr aufwendigen Iterationen werden von Anfang an in den Prozess integriert und dienen der dynamischen Entwicklung eines Projektes.

4.2. Lineare Entwicklungsmethoden am Beispiel Wasserfallmodell

Im Gegensatz zu agilen Entwicklungsmethoden fordern lineare (klassische) Ansätze wie das Wasserfall- oder V-Model einen festen, schrittweisen Ablauf der Prozessphasen. Iterationen sind, zumindest in den ursprünglichen Ansätzen, nicht vorgesehen. Jede Phase bildet einen in sich abgeschlossenen Entwicklungsbereich und muss in der angegebenen Reihenfolge vollständig bearbeitet werden.
Klassische Methoden sind dokumentenbasiert. In jeder Phase entsteht ein Dokument/Programm, auf dessen Basis die jeweils nachfolgende Phase den Prozess fortführt.

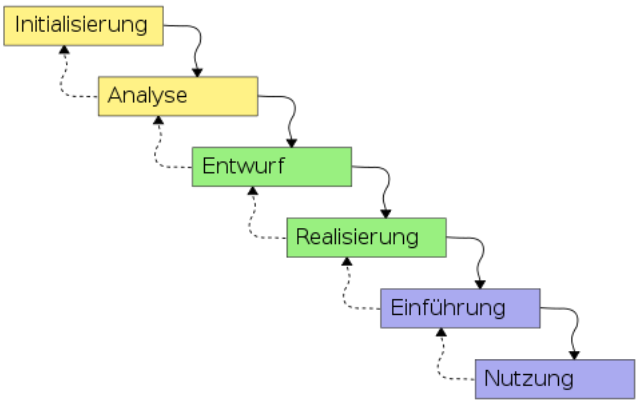

Abbildung 4.1.: Wasserfallmodell mit Rücksprungmöglichkeiten (Quelle: Wikipedia)

Das in der Abbildung dargestellte Wasserfallmodell bietet bereits Rücksprungmöglichkeiten zu der direkt voran gegangen Phase des Entwicklungsprozesses. Tauchen in einer späteren Phase Probleme auf, so muss der Prozess Phase für Phase rückwärts durchlaufen werden, bis zu der Phase an dem das Problem behoben werden kann.

Beispiel:
Ändern sich während der Realisierung die Anforderungen, so muss theoretisch das Projekt zurück in die Phase Entwurf um evtl. die Konzepte anzupassen. Behebt das nicht das Problem, muss eine erneute Anforderungsanalyse durchgeführt werden. Sind die, während der Realisierung, festgestellten Änderungen der Anforderungen hier behoben, muss das gesamte Grob-/Feinkonzept neu erstellt, oder zumindest überarbeitet werden, bevor es wieder in die Realisierungsphase übergehen kann.

Ein direkter Sprung von der Realisierung in die Analysephase würde in dem Beispiel weniger Zeit beanspruchen, jedoch lässt sich unter Umständen nicht klar sagen an welcher Stelle des Vorgehensmodells die Problemlösung stattfinden kann.

Vor- & Nachteile des Wasserfallmodells

Vorteile	Nachteile
• sehr einfach zu erlernender Prozess • Phasen sind sehr klar voneinander getrennt • einfache Plannung und Kontrolle	• Anforderungen sind nur schwer zu Projektbeginn vollständig definierbar • Reale Projekte folgen nur selten einem streng linearen Prozess • Iterationen sind nur indirekt möglich • Die Veröffentlichung des Produktes ist erst in einer sehr späten Phase (Einführung) möglich • Beteiligung des Kunden nur am Anfang und am Ende • Fehlererkennung (Tests) erst sehr spät im Projekt

4.3. Einführung in Scrum

Da sich Scrum in den letzten Jahren steigender Beliebtheit erfreut, und die Cocomore AG in Zukunft vermehrt Projekte nach dieser Methode bearbeiten will, wird im weiteren Verlauf dieser Arbeit nur noch auf Scrum eingegangen.

Scrum wurde Anfang der 90er Jahre durch Jeff Sutherland und Ken Schwaber, beide Unterzeichner des agilen Manifest, als Framework zur Entwicklung komplexer Projekte entwickelt und wurde ursprünglich als Entwicklungsmethode angesehen. In den letzten Jahren hat sich Scrum aber mehr und mehr als sehr mächtiges Projektmanagementtool etabliert.

Die Komplexität eines Projektes soll durch drei Prinzipien reduziert werden.

Transparenz
Innerhalb des Scrum Prozesses sind alle Handlungsabläufe, Fortschritte und auch Probleme dem gesamten Projektteam bekannt. Alle Beteiligten sollen in der Lage sein, den Projektablauf und die Projekteigenschaften zu jeder Zeit zu kennen und zu verstehen.

Zwei Beispiele zu Transparenz aus dem Scrum Guide:

- Die im Projekt verwendete Sprache muss von allen Beteiligten verstanden werden
- Es muss eine allgemeine Definition für „Done“ für alle Projektbeteiligten existieren

Inspektion
Die Prozesse, Artefakte und auch der Fortschritt eines Projektes müssen in regelmäßigen Abständen durch qualifizierte Personen überprüft werden. Ziel jeder Inspektion ist es, Abweichungen, die der erfolgreichen Realisierung eines Projektes im Wege stehen zu erkennen. Der Zeitraum zwischen zwei Inspektionen sollte so gewählt werden das er die Arbeit an einem Projekt nicht behindert.

Adaption
Werden während einer Inspektion Abweichungen, die außerhalb eines akzeptablen Rahmens liegen, festgestellt, und dadurch der erfolgreiche Projektabschluss gefährdet, so muss der dazugehörige Prozess/Artefakt schnellst möglich angepasst werden.

Innerhalb des Scrum-Prozesses existieren hierzu vier verschiedene Events die für die Einhaltung der drei Prinzipien sorgen sollen:

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review Meeting
- Sprint Retrospective

Das Ziel, das durch diese drei Prinzipien erreicht werden soll, ist die schnelle und kostengünstige Fertigstellung eines qualitativ hochwertigen Produkts.

Die Rolle des Kunden
Scrum bindet den Kunden von Beginn an in den Entwicklungsprozess mit ein. Der Kunde soll dem Entwicklungsteam helfen das Projekt besser zu verstehen. Im Gegenzug erhält der Kunde die Möglichkeit die Produktentwicklung zu steuern.

Die Rolle der Entwickler
Während der Produktentwicklung sind die beteiligten Entwickler in der Wahl der verwendeten Methoden und Tools frei. Ihr Ziel ist allein die erfolgreiche Umsetzung einer Produktidee in ein reales Produkt. Um ein hochwertiges Produkt zu entwickeln organisieren sie sich selbst

Scrum ist nachhaltig
Durch die von Scrum vorgesehene Kommunikation zwischen den Projektbeteiligten entsteht ein langfristiger Lerneffekt der die Kompetenzen der Beteiligten erweitert und vertieft.
Entwickler definieren den Zeitaufwand für eine Anforderung. Überarbeitung wird so verhindert.

4.3.1. Der Scrum-Prozess

Am Anfang des Scrum-Prozesses steht die Vision, die Idee eines Produktes. Hier werden die Anforderungsanalyse und -definition für das spätere Product Backlog durch den Kunden (Product Owner) durchgeführt. Die Anforderungen sammelt der Product Owner im Product Backlog in Form sogenannter User Stories welche meistens aus sogenannten Epics hergeleitet werden. Die Einträge im Product Backlog sind nach Geschäftswert, Kosten und Risiken priorisiert.
Das Product Backlog unterliegt regelmäßigen Veränderungen.

Der Projektverlauf wird durch einfache und wirkungsvolle Mittel geplant, gesteuert und kontrolliert. Die Umsetzung wird in Iterationen fester Länge (Timeboxing) unterteilt, sogenannte Sprints. Dies gilt gleichermaßen für Konzeptions- und Programmieraufgaben. Die typische Sprintlänge liegt zwischen zwei und vier Wochen und wird projektspezifisch gemeinsam von allen am Projekt beteiligten Gruppen festgelegt. Am Ende eines jeden Sprints entsteht ein sichtbares und testbares Ergebnis, meistens in Form von funktionierender und qualitätsgeprüfter Software, die zur Abnahme durch den Verantwortlichen bereitsteht.

Vor jeder Iteration wird in der Sprint-Planung auf Basis von Plandaten und bereits erhobenen Projektkennzahlen (Geschwindigkeit, Velocity) festgelegt, welche und wie viele User Stories umgesetzt werden sollen (Sprint Backlog). Der Fortschritt innerhalb des Sprints wird auf sogenannten Burn-Down-Diagrammen (erstellt durch den Scrum Master) verfolgt, in welchen das Verhältnis zwischen Restaufwänden und Restkapazität veranschaulicht wird. Die aktuellen Restaufwände werden vom Scrum Master im Rahmen von sogenannten Daily Scrum Meetings erhoben.
Die Vorstellung der Ergebnisse findet im Rahmen sogenannter Review-Meetings

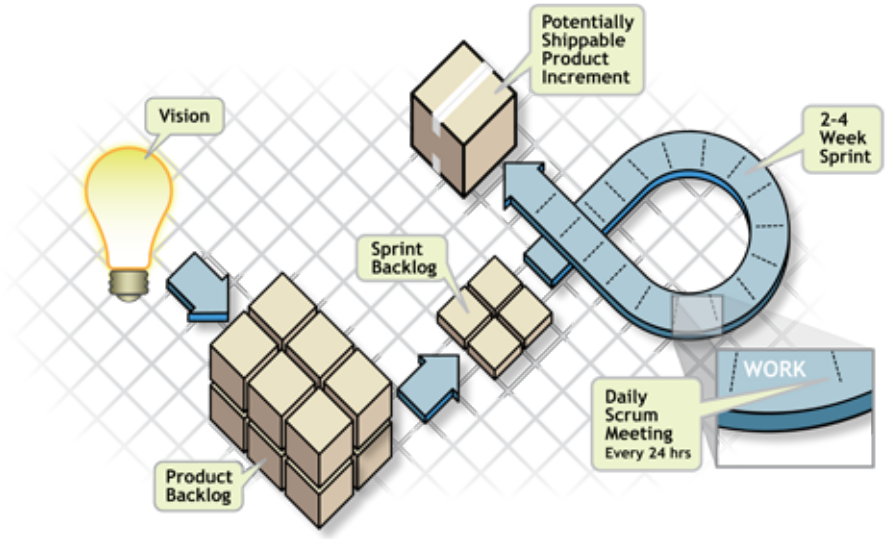

Abbildung 4.2.: Der Scrum-Prozess (Quelle: Scrum Alliance)

statt, in denen das gesamte Development Team dem Product Owner die Ergebnisse vorstellt. Die Abnahme der Ergebnisse findet zeitnah statt. Das Development Team arbeitet selbst organisiert unter Betreuung des Scrum Master, der für die Einhaltung des Prozesses verantwortlich ist. Im Mittelpunkt des Prozesses steht die Kommunikation des Development Teams mit dem Kunden bzw. seinen Repräsentanten. Damit die Kommunikation effizient abläuft, wird die Rolle des Kunden durch eine einzige Person vertreten, welche dem Umsetzungsteam jederzeit zur Verfügung steht, dem Product Owner. Der Product Owner ist detailliert im Bilde über alle sich in Entwicklung befindenden Anforderungen und muss befähigt sein Entscheidungen zu treffen.

Nach einem Sprint werden umgesetzte User Stories als „Done“ gekennzeichnet und neue hinzugefügt bzw. bestehende inhaltlich verändert.

Am Ende eines Sprints reflektiert das Development Team in sogenannten Retrospektiven offen und selbstkritisch über mögliche Verbesserungen, wie über erfolgreich, bewährte und beizubehaltende Praktiken. Mögliche Verbesserungen werden im Impediment Backlog festgehalten.

Da die Verfügbarkeit des Kunden am Ort der Umsetzung nicht immer gewährleistet werden kann, wird in einem solchen Fall der Product Owner beim Kunden durch einen Product Owner Spiegel am Ort der Umsetzung unterstützt. Dieser steht in permanenten Dialog mit dem Kunden und kann somit das Umsetzungsteam fachlich führen, wie in Scrum gefordert.

4.3.2. Vorteile von Scrum

Scrum, wie andere agile Methoden (XP, Lean), erfordert die adäquate Substitution von klassischen Methoden und Rollen. Basierend auf bisherigen Erfahrungen bestehen für den Auftraggeber, trotz evtl. vorliegender formaler Abweichung von üblicherweise angewandten Vorgehensmodellen, wie z.B. dem Wasserfallmodell, faktisch keine Nachteile, sondern im Gegenteil zusätzliche Vorteile.

Durch Scrum kann folgendes erreicht werden:

- Einhaltung der Qualität durch Festlegung der „Definition of Done“
- Einhaltung der Timings zugesicherter Feature-Umfänge durch Timeboxing (feste Release- Zyklen)
- Größtmögliche Transparenz über Status des Projektes (Projektfortschritt, Termine, Risiken, Qualität)
- Bestmöglicher Nutzen im Sinne des Geschäftswerts
- Budgetsicherheit, da nach Sprintende lauffähige Softwareversionen verfügbar sind und Transparenz über den Status der umgesetzten Features herrscht.

4.3.3. Rollen, Artefakte & Events

4.3.3.1. Rollen

Scrum Master
Der Scrum Master ist verantwortlich für die erfolgreiche Umsetzung des Scrum-Prozesses und ist in der Regel kein Mitglied des Projektteams. Er führt und begleitet den Scrum-Prozess und ist während eines Projektes der Ansprechpartner für alle Scrum-bezogenen Abläufe und Techniken. Der Scrum Master führt während eines Projektes die Burn-Down-Charts der Restaufwände eines Sprints und Projektes, organisiert und leitet die notwendigen Meetings und unterstützt die Projektbeteiligten bei der Veralatung und Pflege des Product Backlog.

Der Scrum Master nimmt für unterschiedliche Rollen unterschiedliche beratende Funktionen wahr: Der Product Owner wird durch den Scrum Master bei der Veraltung und Erstellung des Product Backlog unterstützt.
Dem Development Team hilft er bei der Selbstorganisation und dokumentiert für die Entwicklung relevante Störungen des Prozessablaufes im Impediment Backlog.

In Unternehmen begleiten und leiten Scrum Master die unternehmensweite Einführung des Scrum-Prozesses und helfen Mitarbeitern und Stakeholdern den Scrum-Prozess zu verstehen.

Product Owner
Der Product Owner ist verantwortlich für die Erstellung und Verwaltung des Product Backlog. Er entscheidet über die Priorität und die in einem Sprint umzusetzenden Anforderungen des Product Backlog. Änderungen am Product Backlog müssen von ihm und für ihn begründet werden.
Der Product Owner ist eine einzelne Person. Zwar kann ihm ein Team zur Seite gestellt werden, jedoch verbleibt die Verantwortung bei ihm alleine.

Development Team
Das Development Team besteht aus drei bis neun Personen, die an der Umsetzung des Product Backlog in ein funktionsfähiges Produkt für einen Sprint arbeiten. Die Zusammensetzung des Teams ist interdisziplinär, alle Bereiche die zur Erreichung des Sprint-Zieles notwendig sind, werden durch das Team abgedeckt. Development Teams organisieren sich selbst, wodurch eine Erhöhung der Effizienz und Effektivität erreicht werden soll. Niemand darf dem Team sagen, wie es Anforderungen des Product Backlogs umzusetzen hat.
Innerhalb des Development Teams gibt es keine Hierarchien.

4.3.3.2. Artefakte

Product Backlog
Das Product Backlog ist eine nach Prioritäten sortierte Liste aller Anforderungen an ein Produkt (Backlog Item) und wird vom Product Owner geführt und verwaltet. Das Product Backlog muss für alle Projektbeteiligten verständlich und verfügbar sein.
Das Product Backlog ist niemals final und besteht immer so lange wie das Produkt für das es erstellt wurde.
Das Product Backlog entwickelt sich zusammen mit dem Produkt. Änderungen an dem einen führen immer auch zu Änderungen an dem anderen.

Im Product Backlog sind alle Anforderungen, Funktionen, Features, Verbesserungen und Bugfixes während des gesamten Lebenszyklus eines Produktes dokumentiert. Einträge im Product Backlog enthalten neben der Anforderung/User Story (Backlog Item) und den Akzeptanzkriterien auch Beschreibungen, die Priorität und die geschätzte Dauer der Umsetzung.

Je höher ein Backlog Item priorisiert ist, desto genauer muss die dazugehörende Spezifikation sein. Durch dieses Vorgehen wächst das Product Backlog mit der Zeit an, da eine genaue Spezifizierung immer für die am höchsten priorisierten, und noch nicht umgesetzten, Elemente des Product Backlog vorgenommen werden muss.

Sprint Backlog
Das Sprint Backlog enthält alle Anforderungen die in einem Sprint umgesetzt werden sollen. Das Sprint Backlog definiert den Aufwand für das Development Team um die ausgewählten Anforderungen umzusetzen. Anforderungen können hier in einzelne Tasks heruntergebrochen und durch das Team separat priorisiert und die Dauer der Umsetzung geschätzt werden.

Product Increment
Das Product Increment ist das Ergebnis eines Sprints. Es erfüllt zwar noch nicht unbedingt alle Anforderungen des Product Backlog, ist aber trotzdem vollständig funktionsfähig und kann theoretisch veröffentlicht werden. Nach der erfolgreichen Abnahme durch den Product Owner wird das Product Increment ebenfalls als „Done“ angesehen.

Definition of Done
Die „Definition of Done“ ist die Festlegung, wann eine Anforderung des Product Backlog oder ein Product Increment als fertig anzusehen ist. Jeder Teilnehmer eines Projektes muss die „Definition of Done“ verstehen. Anforderungen sind genau dann fertig umgesetzt, wenn sie den Kriterien der „Definition of Done“ entsprechen.

Hier ein kurzes Beispiel für eine „Definition of Done“:

1. Eine Anforderung ist dann erfüllt, wenn alle Akzeptanzkriterien erfüllt sind, und
2. einen funktionalen Test bestanden hat, und
3. auf ihre Usability geprüft wurde, und
4. falls Anforderungen durch eine heuristische Evaluation geprüft wurden, dürfen maximal Probleme der Schwerestufe 2 (Leichte Probleme) enthalten sein.

Erst wenn alle diese Kriterien erfüllt sind, kann eine Anforderung als „Done“ angesehen werden.

Impediment Backlog
Das Impediment Backlog enthält alle Probleme und Lösungsansätze, die das Scrum-Team an der erfolgreichen und effizienten Erfüllung ihrer Aufgaben behindern. Der Scrum Master ist verantwortlich die Probleme schnellstmöglich zu beheben.

4.3.3.3. Events

Sprint
Ein Sprint ist ein zeitlich festgelegter Bereich (2 bis 4 Wochen) in dem ausgewählte Anforderungen umgesetzt werden. Jeder Sprint wird durch das Sprint Planning Meeting eingeleitet und durch das Sprint Review Meeting abgeschlossen. Während des Sprints finden tägliche Besprechungen (Daily Scrum) des Development Teams statt. Ziel jedes Sprints ist die Erstellung eines Product Increments das den „Definition of Done“ entspricht.
Jeder Sprint hat ein definiertes Ziel, das im Sprint Planning Meeting festgelegt wird.

Während eines Projektes können bestimmte Sprints als Milestones definiert sein, z.B. wenn der Funktionsumfang soweit vollständig ist, dass eine wirkliche Release-Version vorhanden ist oder ein Epic des Product Backlog vollständig in das Product Increment implementiert wurde.

Scrum sieht vor, dass Sprints durch den Product Owner abgebrochen werden können. Einer der Gründe für den Abbruch eines Sprints kann z.B. die durch den Product Owner nicht mehr gewünschte Umsetzung eines für den Sprint geplanten Epics sein.
Der Abbruch eines Sprints kann auch auf Grund von Empfehlungen durch Stakeholder, Development Team oder des Scrum Masters erfolgen. Die letzte Entscheidung über den Abbruch eines Sprints trifft jedoch immer der Product Owner.

Sprint Planning Meeting
Das Sprint Planning Meeting dient der Vorbereitung eines Sprints. Basis des Meetings ist das Product Backlog und, sofern schon vorhanden, das letzte Product Increment. Das Sprint Planning Meeting ist in zwei Abschnitte aufgeteilt.
Im ersten Abschnitt werden dem Development Team die durch den Product Owner ausgewählten Anforderungen vorgestellt. Die umzusetzenden Anforderungen werden im gesamten Projektteam besprochen und die für die erfolgreiche Umsetzung der Anforderungen notwendigen Aufgaben definiert. Wie viele Anforderungen in einem Sprint umgesetzt werden können, wird letztendlich durch das Development Team festgelegt, basierend auf bisherigen Erfahrungen bei vergleichbaren Anforderungen.
Im zweiten Abschnitt des Meetings wird durch das Development Team entschieden, wie es die Anforderungen in ein fertiges Product Increment umsetzen wird. Anforderungen können dazu in Tasks aufgeteilt und durch das Development Team hinsichtlich des Sprintziels priorisiert werden.
Ein Sprintziel kann z.B. die Umsetzung von mindestens 90% der geforderten Anforderungen sein.
Am Ende des Sprint Planning Meetings sollte das Development Team in der Lage sein, dem Product Owner und dem Scrum Master zu erklären, wie die Anforderungen innerhalb des Teams umgesetzt und das Sprintziel erreicht werden sollen.
Das Sprint Planning Meeting sollte, abhängig von der Sprintdauer, nicht länger als 8 Stunden sein.

Daily Scrum
Das Daily Scrum ist ein täglich stattfindendes Meeting mit einem maximalen Zeitaufwand von 15 Minuten. Zweck des Daily Scrum ist es dem gesamten Development Team einen Überblick über die Arbeit der letzten 24 Stunden und der nächsten 24 Stunden zu geben.

Themen die im Daily Scrum besprochen werden sollen sind:

- Was wurde seit dem letzten Daily Scrum getan,
- was ist bis zum nächsten Daily Scrum geplant,
- welche Hindernisse stehen im Weg?

Hindernisse in Bezug auf die effiziente Umsetzung von Anforderungen sollten in das Impediment Backlog aufgenommen werden.

An den Daily Scrums können alle Personen teilnehmen die an dem Projekt beteiligt sind oder sich dafür interessieren. Rederecht haben jedoch ausschließlich Mitglieder des Development Teams. Product Owner dürfen jedoch auf Fragen antworten.

Sprint Review
Das Sprint Review Meeting findet am Ende eines Sprints statt. Es dient zur Inspektion des Product Increment und, falls notwendig, der Anpassung des Product Backlog. Während des Meetings werden durch das Scrum-Team und die Stakeholder die Arbeitsergebnisse begutachtet. Auf Basis der Begutachtung und eventueller Änderungen am Product Backlog werden mögliche Schritte und Anforderungen für den folgenden Sprint erarbeitet, bzw. ausgewählt.

Zusätzlich werden Probleme, die bei der Bearbeitung von Anforderungen aufgetaucht sind, besprochen und gefundene Lösungen vorgestellt.

Der Sprint Review sollte, abhängig von der Sprintdauer, nicht länger als 4 Stunden (bei 4-wöchigen Sprints) dauern.

Sprint Retrospective
Im Anschluss an den Sprint Review findet die Sprint Retrospective statt.

> Der Sinn der Retrospektive ist es zu inspizieren, wie der letzte Sprint in Bezug auf die Menschen, Beziehungen, den Prozess und die Werkzeuge gelaufen ist.
>
> (Quelle: Scrum Guide, deutsche Version)

Hier sollten also alle, den Scrum-Prozess beeinflussenden, Erfahrungen diskutiert werden. Umsetzbare Verbesserungsmaßnahmen werden in das Impediment Backlog übernommen und in den folgenden Sprints umgesetzt.

Kapitel

FÜNF

User Centered Design und Scrum

Schwaber und Sutherland beschreiben in ihrem Scrum Guide(10) die Rahmenhandlung, die jeweiligen Schritte des Scrum-Prozesses. Interne Abläufe werden dagegen nicht oder nur sehr grob umrissen. Die beiden Autoren folgen in ihrem Guide nach näherer Betrachtung bereits weitestgehend den Prinzipien des UCD.
Als Abgrenzung zu den klassischen, linearen, Entwicklungsmethoden sei hier anzumerken, dass diese den UCD-Prozess in maximal 2 Phasen (Analyse & Entwurf) vorsehen, Scrum jedoch während des gesamten Entwicklungsprozesses.

Phase 1 & 2 – Analyse des Nutzungskontextes & Definition der Anforderungen
Im UCD-Prozess dienen die Analyse des Nutzungskontextes und die Definition der Anforderungen der Vorbereitung der späteren Konzeption. Innerhalb des Scrum-Prozesses können die beiden Phasen in die Erstellung des Product Backlog überführt werden. Zur Erinnerung, das Product Backlog enthält in seinem initialen Zustand nur die Anforderungen an ein Produkt und eine kurze Beschreibung der Anforderung. Eine detaillierte Ausarbeitung der einzelnen Anforderung ist in Scrum wie auch im UCD-Prozess in dieser Phase nicht vorgesehen.

Phase 3 – Konzeption & Entwurf
Die Überführung einer Anforderung in ein Konzept zur Umsetzung kann, und sollte, in Teilen ebenfalls in der Vorbereitung des Product Backlog vorgenommen werden. Vor dem Beginn der eigentlichen Entwicklung sollte mindestens ein Navigationskonzept und die Beschreibung zentraler Funktionen (z.B. Suchfunktion auf Webseiten) näher definiert werden. Die Konzeption einzelner Anforderungen kann in den folgenden Sprints als Task definiert werden. Die eigentliche Umsetzung erfolgt nach Abschluss des Tasks.
Scrum sieht vor, dass Anforderungen bereits im Product Backlog kurz beschrieben sind (z.B. durch User Stories) und während der Umsetzung in einem Sprint weiter ausgearbeitet werden.

Phase 4 – Evaluation
Scrum sieht bereits die ausführliche Evaluation der implementierten Funktionen vor. Die Integration von Evaluierungsmethoden des UCD stellt hier nur eine Erweiterung der Testphase dar.

Usability-Experten in Scrum
Durch Erfahrungen in verschiedenen Projekten hat sich gezeigt, dass die permanente Integration von Usability-Experten in eine feste Scrum-Rolle nicht für alle Projekte sinnvoll ist. Hier sollte im Kontext des Projektes entschieden werden wie eine Einbindung der Usability-Experten in den Scrum-Prozess sinnvoll ist.
Bei Projekten in denen bereits zu Beginn ein hohes Maß an Konzeption absehbar ist, sollten Mitarbeiter die für die Usability eines Produktes verantwortlich sind, je nach Phase in unterschiedlichen Teams vertreten sein.
Während der Erstellung des Product Backlog bietet sich die Mitarbeit im Team des Product Owner an. Abhängig vom Projekt können Usability-Experten während der Sprint-Phase auch in das Entwicklungsteam integriert werden.

5.1. Integration von User Centered Design in der Vorbereitungsphase

Untersuchungen durch die Universität Paderborn(7) ergaben, dass die Erstellung des Product Backlog durch Methoden des UCD unterstützt werden kann. Durch die Integration dieser Methoden zur Vorbereitung des Product Backlog lässt sich die im UCD-Prozess geforderte Festlegung auf Usability-Anforderungen innerhalb des Scrum-Prozesses umsetzen.
Zur Vorbereitung des Product Backlog können Nutzerinformationen z.B. durch Contextual Inquiries oder Interviews gesammelt und ausgewertet werden. Die durch die Auswertungen erhaltenen Anforderungen der Benutzer an die zu entwickelnde Software kann abschließend in das Product Backlog übernommen werden.

5.1.1. Anforderungsdefinition

In verschiedenen Projekten hat sich gezeigt, dass die im Vorfeld durchgeführte Nutzeranalyse auf Basis von Methoden des UCD die Qualität der Anforderungsdefinition verbessern kann. Die Definition der Anforderung für das Product Backlog kann sehr gut über ein Affinity-Diagramm oder ein Mental Model (vgl. Indi Young, Mental Models, 2008) erfolgen.
Da die Mental-Model-Methode in der Aufbereitung der Nutzerinformationen jedoch sehr aufwendig ist, wird im weiteren Verlauf jedoch nur auf Affinity-Diagramme eingegangen.

Für jede Person die an Contextual Inquiries, Interviews oder Fokusgruppen teilnimmt, entstehen üblicherweise 50-100 Affinity Notes. Um Anforderungen mit einer halbwegs statistischen Relevanz zu erhalten, sollten mindestens 12 Personen befragt werden. Daraus ergeben sich 600 bis 1200 Affinity Notes.
Eine Person kann am Tag etwa 50 bis 80 Affinity Notes für das Diagramm bearbeiten. Um also das Affinity-Diagramm in einer akzeptablen Zeit zu erstellen sollten alle Projektteilnehmer zumindest zeitweise an der Bildung der Affinity Wall teilnehmen. (vgl. Holtzblatt, Rapid Contextual Design, S. 161 ff.)

Allgemeines Vorgehen
Zu Beginn muss eine geeignet große Wandfläche bereitgestellt werden. Diese bildet die Affinity Wall und sollte während des gesamten Konzeptionsphase (in Scrum also bis zum Ende des Projektes) bestehen bleiben.
Die Entwicklung des Affinity-Diagramms wird vom UCD-Team geleitet und, im Fall von Diskussionen, moderiert. Diskussionen sind während der Erstellung der Affinity Wall erlaubt und gewünscht.

Die gesammelten Nutzerinformationen werden auf Affinity Notes (gelb) übertragen und an der Affinity Wall angebracht. Hier können auch Fragen, Probleme und Design-Ideen, die sich durch die Nutzerinformationen ergeben, durch weitere Affinity Notes ergänzt werden. Diese werden besonders gekennzeichnet (Fragen – grüne Markierung; Probleme – rote Markierung; DI – Design Idea).
Im Anschluss werden diese thematisch in Bereiche (Cluster) gruppiert. Während der Clusterung sollten die Teilnehmer kurz erläutern, warum die Zuordnung von Affinity Notes in den ausgewählten Cluster vorgenommen werden sollte.
Die jeweiligen Cluster werden durch blaue Labels benannt. Die Benennung der blauen Labels sollte die darunterliegenden Anforderungen gezielt zusammenfassen. Es ist hier möglich, die blauen Labels als User Story zu formulieren. Die Formulierung in Form von User Stories erleichtert den späteren Übertrag in das Product Backlog

Ein Beispiel für die Benennung von gelben und blauen Labels, z.B. für Anforderungen an ein E-Mail-Programm:

Blau – Ich will individualisierte Nachrichten verschicken
- **Gelb** – Benutzer will die Schriftart ändern können
- **Gelb** – Benutzer will Listen in E-Mails erstellen
- **Gelb** – Texte sollen farblich hervorgehoben werden können
- **Gelb** ...

Anschließend erfolgt eine weitere Clusterung der blauen Labels (auch hier wieder mit kurzer Erläuterung). Hierbei werden die Affinity Notes ebenfalls bewegt. Die Cluster der blauen Labels werden durch pinkfarbene Labels benannt.

Ein Beispiel für die Benennung von blauen und pinkfarbenen Labels, z.B. für Anforderungen an ein E-Mail-Programm:

Pink – Sortierung von Nachrichten
- **Blau** – Als Benutzer will ich meine Nachrichten gruppieren können
- **Blau** – Als Benutzer will ich Nachrichten farblich hervorheben können
- **Blau** ...

In der letzten Stufe werden die pinkfarbenen Labels wiederum geclustert und durch grüne Labels benannt.[1]

[1] Dieser Schritt kann ausgelassen werden, falls eine weitere Clusterung nicht sinnvoll ist, z.B. bei nur 10 pinkfarbenen Labels.

Sind alle Affinity Notes an der Wall angebracht und geclustert, sollte jedes Label 4-6 Unterelemente enthalten.

Die Affinity Wall ist nicht final. Jede Affinity Note und jedes Label kann zu jeder Zeit einem neuem neuen Cluster zugeordnet werden. Eine Änderung der Zuordnung zu einem Cluster muss jedoch mit den weiteren Projektteilnehmern die an der Bildung des Affinity-Diagramms beteiligt sind diskutiert werden.

Nachdem das Affinity-Diagramm erstellt wurde, sollte ein „Wall Walk“ mit allen Projektbeteiligten durchgeführt werden. Während des Wall Walk können die einzelnen Cluster nochmals überprüft und das Verständnis für die Nutzeranforderungen verbessert werden. Ebenfalls können während des Wall Walk weitere Design Ideen entwickelt und offene Fragen geklärt werden.[2]

Für die Erstellung des Product Backlog können die blauen Labels als Anforderungen übernommen werden. Die darunter liegenden Affinity Notes können als Tasks oder Akzeptanzkriterien der jeweiligen Anforderung ebenfalls in das Product Backlog einfließen.
Pinkfarbene Labels werden als Epics (Gruppen von Anforderungen) in das Product Backlog eingetragen.
Da die grünen Labels einen sehr hohen Abstraktionsgrad aufweisen, können diese unter Umständen als eine erste Navigationsstruktur in das Product Backlog einfließen.
Die Übernahme von Affinity Notes und Labels in das Product Backlog geschieht immer in Absprache mit dem Product Owner.

Um in späteren Iterationen des Product Backlog die Übersicht über die Affinity Wall zu behalten, empfiehlt es sich, die jeweiligen Affinity Notes mit der Nummer der Anforderung im Product Backlog zu versehen. Durch diese Nummerierung wird verhindert, dass Anforderungen mehrfach in das Product Backlog übernommen werden.

Sofern noch nicht durch die Übernahme der gelben Affinity Notes definiert, sollten für Usability-Anforderungen noch Akzeptanzkriterien hinzugefügt werden. Akzeptanzkriterien sind in Scrum die „Definition of Done“ und bestimmen wann ein Task oder eine Anforderung als erfüllt angesehen werden kann.

Nach der Erstellung des Product Backlog wird die Priorisierung der Anforderungen/Tasks durch alle Projektbeteiligten vorgenommen. Nachdem die erste Iteration des Product Backlog erarbeitet und durch den Product Owner und das Team priorisiert wurde, sollte als Vorbereitung auf den ersten Sprint die Spezifikation der am höchsten priorisierten Anforderungen durch das Team des Product Owners erfolgen. Das UCD-Team übernimmt darin die Spezifikation Usability-relevanter Anforderungen (Backlog Items).

[2] In der Praxis hat sich gezeigt, dass zwischen dem Ende der Erstellung eines Affinity-Diagramms und dem Wall Walk mindestens ein Tag vergehen sollte. Durch die Pause können die Beteiligten nochmals in Ruhe die Clusterung reflektieren.

Abbildung 5.1.: Geclustertes Affinity-Diagramm (Quelle: Flickr)

#	Epic	User Story / Task	Akzeptanz-Kriterien	Details Diskussion
32	Inhaltstyp Produkt	Als Tango-Benutzer (Verlagsadmin, FV-Admin, Firmen-Account) möchte ich neue Produkte anlegen können	- Es müssen folgende Felder pflegbar sein: - Produktname: Produktname; alphanumerisch Textfeld - Beschreibung: Beschreibung zum Produkt; Textarea inkl. Möglichkeit zum "Break" , Bullet-Points, Fett via WYSIWIG - Synonyme: Eintrag verschiedener Synonyme für die Produkte; ","-getrennte alphanumerische Eingabe - Internet: Eintrag eines Verweis auf die eigene Firmenhomepage; Linkfeld (Titel, URL); begrenzt auf einen Link - Produkt ID; alphanumerisches Textfeld - Sortierkriterium; alphanumerisches Textfeld - Punch-Out-URL; - Fachverbandszuordnung - Firmenzuordnung - Bilder (unbegrenzt) ; (weboptimiert; max 3mb upload, werden runtergerendert)	01.03.2012 In der Diskussion wurde festgestellt, dass eine User-Stroy für den Nomenklatureditor verfasst werden muss 16.03.2012 - Ordnerstruktur: Bitte die Pfade für die verschiedenen Mediatypen vorgeben (bei Firma und Produkten) - Produkt-ID: Muss automatisch und fortlaufend generiert werden / darf nicht editziert werden können - Die Fachverbandszuordnung erfolgt über die Zuordnung der Firma
33	Inhaltstyp Artikel	Als Tango-Benutzer möchte ich neue Artikel anlegen können	- Artikel-Seiten müssen über einen WYSIWIG-Editor editierbar sein - Es müssen Bildinhalte umgesetzt werden können - Es müssen Meta-Elemente gepflegt werden können (Title,& Body, Meta-Tags)	Beim Artikel handelt es sich um den Standard-Drupal inhaltstyp, welcher für Content-Seiten wie Impresum verwendet werden kann

Abbildung 5.2.: Beispiel Product Backlog nach Übertragung von Affinity Notes

5.1.2. Sprint Planning Meeting

Während des Sprint Planning Meetings kann entschieden werden wie das UCD-Team in den kommenden Sprint integriert wird.
Die Integration kann auf zwei Arten erfolgen:

1. Während des Sprints nimmt das UCD-Team eine reine Beratungsfunktion des Development Teams ein. Das UCD-Team ist in das Team des Product Owner integriert.

2. Das UCD-Team wird in das Development Team integriert und erarbeitet mit diesem auf Usability optimierte Lösungen für Oberflächendesign, Workflows und Funktionen.

Der weitere Verlauf des Sprint Planning Meetings ist abhängig von der für das UCD-Team gewählten Rolle.

Fall 1 – Integration des UCD-Teams in das Team des Product Owner
Falls das UCD-Team in das Team des Product Owner integriert wird, können während des anstehenden Sprints hoch priorisierte Backlog Items, die Einfluss auf die Usability haben, durch das UCD-Team spezifiziert und mit dem Product Owner abgestimmt werden.
Durch diese Maßnahme können Backlog Items vor der eigentlichen Umsetzung in einem Sprint spezifiziert und in späteren Sprints direkt umgesetzt werden.

Fall 2 – Integration des UCD-Teams in das Development Team
Für den Fall, dass das UCD-Team in das Development Team integriert wird, werden die für den Sprint ausgewählten Backlog Items, mit UCD-Bedarf, um den Task „Spezifiaktion/Wireframes“ erweitert. Auf Basis von früheren Anforderungsspezifikationen schätzt das UCD-Team die Aufwände für die Spezifikation und die Wireframes, die zur Erledigung des Tasks notwendig sind.

Die Integration des UCD-Teams in das Development Team kann zum Beispiel dann notwendig werden, wenn zwar schon eine Spezifikation zu einem Backlog Item vorliegt, diese jedoch während des laufenden Sprints überarbeitet werden muss.
Problematisch bei diesem Vorgehen ist hier die kurzfristige Einbeziehung des UCD-Teams, da UCD-Experten auf Grund der geringeren Aufwände, im Vergleich zur Implemntierung, bei der Entwicklung von Design-Lösungen in der Regel mehr als ein Projekt betreuen. Hier kann es zu Problemen auf Grund von fehlenden Kapazitäten kommen. Gegebenenfalls muss die Umsetzung einer Anforderung in einen späteren Sprint verschoben werden, dies kann unter Umständen zu Verzögerungen im der weiteren Projektplanung führen.

In beiden Fällen, der Überarbeitung einer Spezifikation und fehlender Kapazitäten, sollte durch den Scrum Master ein Eintrag in das Impediment Backlog erstellt werden. Ein möglicher Lösungsansatz für die Überarbeitung von Spezifikationen kann z.B. die Überprüfung der technischen Machbarkeit durch ein Mitglied des Development Teams vor der Aufnahme ins Product Backlog darstellen.

5.2. Sprint

Fall 1 – Integration des UCD-Teams in das Team des Product Owner
Ist das UCD-Team in das Team des Product Owner integriert, nimmt immer mindestens ein Mitglied des UCD-Teams an den täglichen Scrum Meetings in einer Zwitterrolle als Stakeholder der Benutzer und als Teil des Product Owner Teams teil. Parallel zum Sprint können bereits Backlog Items oder Epics in Absprache mit dem Product Owner für den folgenden Sprint spezifiziert und Wireframes, bzw. Prototypen erstellt werden.

Fall 2 – Integration des UCD-Teams in das Development Team
Wurde das UCD-Team in das Development Team integriert, kann die Spezifikation in direkter Zusammenarbeit mit den weiteren Mitgliedern des Development Teams erfolgen. Als Vorteil dieses Vorgehens ist zu sehen, dass sich der Bedarf an Usability-Beratung während der Sprints reduziert, da die Mitglieder des Development Teams mit der Zeit ein Gefühl für das Usability-Konzept der Anwendung entwickeln.

Unabhängig von der Art der Integration des UCD-Teams sollte zum Abschluss eines Sprints das Product Increment einem funktionalen und einem Test mit realen Benutzern nach Methoden des UCD unterzogen werden. Hier bietet sich ein zweistufiges Testverfahren an.
In der ersten Stufe wird durch die Entwickler ein funktionaler Test des Product Increment durchgeführt, um sicherzustellen das alle Funktionen ein korrektes Ergebnis liefern, z.B. bei der Berechnung von Finanzierungsangeboten. Im Anschluss daran wird eine Evaluation nach UCD-Kriterien durchgeführt.
Für alle Product Incremente sollte mindestens eine heuristiscche Evaluation durch Experten durchgeführt werden. Tests mit Benutzern sollten hingegen eher für die im Projekt definierten Milestones erfolgen.

Hintergrund für dieses Vorgehen sind die zu erwartenden Aufwände eines Benutzer-Tests mit echten Benutzern. In der Regel sollte ein solcher Test mit ca. 12-30 Probanden durchgeführt werden. Für jeden Probanden kann man mit einem Zeitaufwand von 45-90 Minuten für den Test rechnen. Darin enthalten sind die Testvorbereitung, der Test und die Nachbereitung durch Interviews. Weitere Probleme tauchen bei der Rekrutierung der Probanden und dem Testumfeld auf. Letzteres wird in der Regel nur durch spezialisierte Usability-Labors zur Verfügung gestellt da die Einrichtung eines solchen Labors in den Entwicklungsfirmen zu aufwendig und kostenintensiv ist.
Zum Vergleich, eine heuristische Evaluation durch fünf Experten ist, abhängig von der Anzahl der zu testenden Seiten, in der Regel nach einem Tag vollständig abgeschlossen.

5.2.1. Spezifiaktion & Wireframes

Scrum fordert eine kurze und verständliche Spezifikation einzelner Backlog Items oder Epics.
Die Spezifikation von Backlog Items/Epics wird im UCD-Prozess üblicherweise an Hand von Wireframes vorgenommen. Ein Wireframe enthält dabei in der Regel mehrere spezifizierte Anforderungen, und teilweise auch Epics, des Product Backlog.

Ein Beispiel aus einem Kundenprojekt (gekürzt)

Epic:	Suchfunktion
Anforderungen/Tasks:	Als Benutzer will ich die Suchergebnisse filtern können
Akzeptanzkriterien:	1. Der Benutzer soll nicht mehr als drei Klicks benötigen um zu einem Produkt zu kommen 2. Der Benutzer soll seine suche durch Auswahl/Eingabe weiterer Kriterien eingrenzen können 3. ...

Epic:	Suchfunktion
Anforderungen/Tasks:	Als Benutzer will ich passende Kategorien zu meinem Suchbegriff angezeigt bekommen
Akzeptanzkriterien:	1. Der Benutzer muss erkennen können ob es zu seinem Suchbegriff Unterkategorien gibt 2. ...

Seitenbeschreibung
Nach dem Klick auf „Untergruppe 3“ des Kunden werden dem Benutzer die Produkte der Firma dargestellt, die in der Kategorie „Untergruppe 3“ enthalten sind und in deren Titel oder Beschreibung das Suchwort „Produkt“ vorkommt. Die Darstellung der Firma inklusive der Kontaktoptionen gibt einen besseren Überblick und eine schnellere Möglichkeit der Kontaktaufnahme. Auf dieser Navigationstiefe können durch Eingabe von Parametern in der linken Marginalie die Ergebnisse weiter gefiltert werden. Hierbei werden nur die wichtigsten Parameter angezeigt, durch Klick auf „Weitere Produkteigenschaften“ lässt die Marginalie nach unten aufklappen und weitere Parametereingabemöglichkeiten erscheinen. Alle getätigten Eingaben werden zusammenfassend bei Klick auf „Ergebnisse filtern“ gesendet und umgesetzt. Die Ergebnisse auf der rechten Seite werden nach ihrer Treffergenauigkeit gefiltert. Ein Klick auf „Produkt“ führt zu Schritt IV, der Produktdarstellung.

Seitenelemente

#	Label	Beschreibung
1	Suchfilter	Benutzer können das Suchergebnis an Hand von Kategorien und Produkteigenschaften filtern.
2	Kategoriefilter	Benutzer sehen die zum Suchbegriff passenden Hauptkategorien sowie die Unterkategorien auf die der Suchbegriff ebenfalls zutrifft.
3	Produkteigenschaften-Filter	Der Benutzer kann die Anzahl der Ergebnisse seiner Suchanfrage durch Auswahl von Produkteigenschaften.

Tabelle 5.1.: Spezifikation des Wireframes

Wireframe

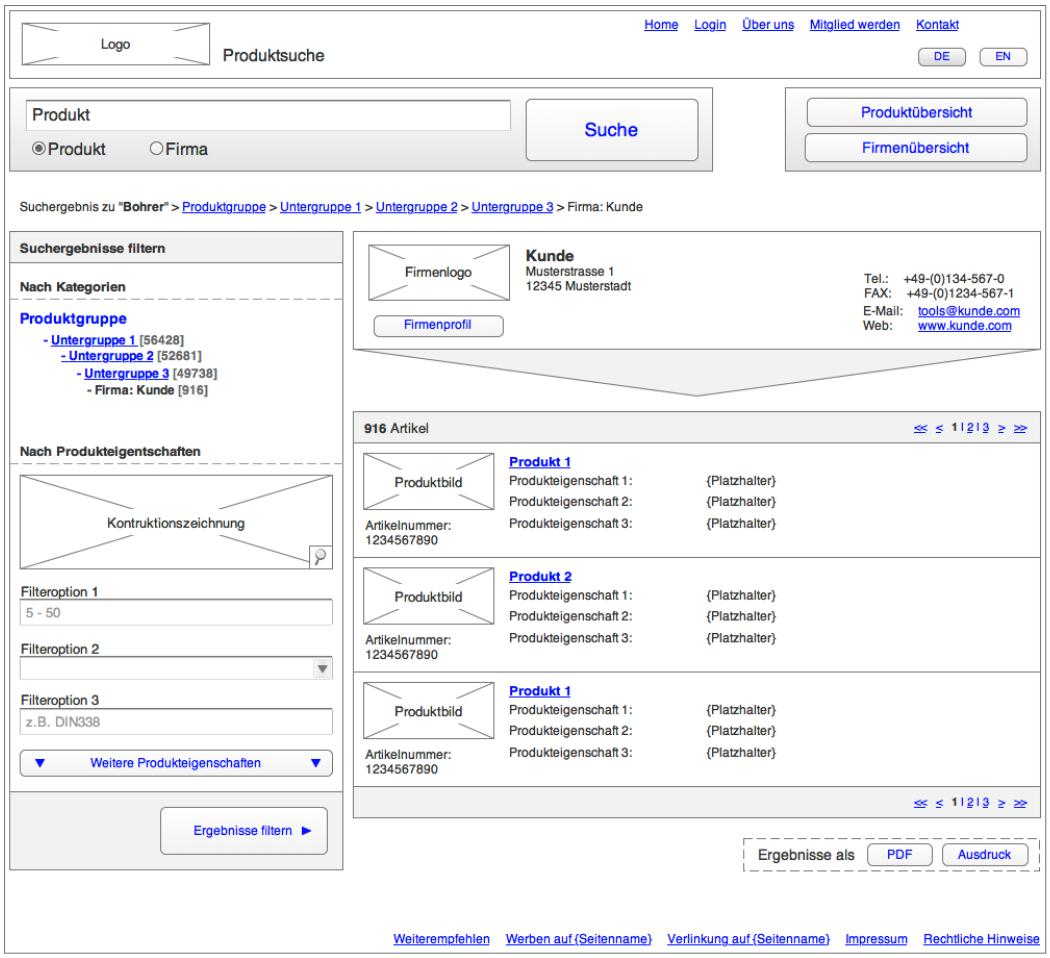

Abbildung 5.3.: Wireframe der Produktsuche

Die Spezifikation und der Wireframe beschreiben zwar die beiden angegebenen Anforderungen und erfüllen auch die Akzeptanzkriterien, aber zusätzlich ist hier auch

noch die Darstellung von Suchergebnissen und Firmeninformationen enthalten, die ebenfalls als Epics im Product Backlog existieren. Üblicherweise sollten in einem solchen Fall alle Anforderungen in einer Spezifikation enthalten sein.

Falls ein Backlog Item/Epic einen Workflow (z.B. Checkout-Prozess auf E-Commerce-Seiten) beschreibt, wird der gesamte Ablauf als Wireframes erstellt und als Clickdummy oder Prototyp gespeichert.

Fertige Spezifikationen und zugehörige Wireframes eines Backlog Item/Epic werden in einer separaten Datei gespeichert. Im Product Backlog wird der Speicherort der Datei als Anmerkung zu den betreffenden Backlog Items/Epics hinterlegt.

5.2.2. Heuristische Evaluation

Die am Ende jedes Sprints vorgesehene heuristische Evaluation nach Nielsen und Molich(1990) sollte durch den UCD-Bereich sowie Mitglieder des Development Teams und des Product Owner vorgenommen werden. Durch diese Zusammenstellung werden die beteiligten Entwickler und der Product Owner auf das Erkennen und das Finden möglicher Lösungen von Usability-Problemen geschult.

Die heuristische Evaluation basiert auf 10 von Nielsen und Molich definierten Regeln und dazugehörigen Handlungsanweisungen.

Hier als Beispiel die komplette erste Regel:

1. **Stelle einen einfachen und naürlichen Dialog her!**
   Ästhetik und minimalistisches Design: Dialoge sollten keine irrelevanten Informationen enthalten, da die Informationen um die Aufmerksamkeit des Benutzers konkurrieren.
   „Less is more"- keine irrelevanten oder wenig benötigten Informationen. Information in einer natürlichen und logischen Ordnung.
   - Grafische Gestaltung: Wichtigste Elemente herausstehend
   - Großbuchstaben können Aufmerksamkeit auf sich ziehen, sind aber 10% langsamer zu lesen
   - Interfaces sollen ohne farbige Darstellung funktionsfähig sein, farbcodierte Info durch weiteren Reiz ergänzen.

Da die Regeln und deren Bewertungskriterien sehr umfangreich sind, können hier nicht alle Heuristiken erläutert werden. Im Anhang findet sich jedoch eine Aufstellung der vollständigen Heuristiken.

Jede Seite einer Anwendung wird während der Evaluation betrachtet und die Elemente der Seite mit Hilfe der Heuristiken bewertet. Die Bewertung erfolgt über fünf Schweregrade:

Schweregrad 0: Kein Problem
Schweregrad 1: Unbedeutend – Problem kann behoben werden, muss aber nicht
Schweregrad 2: Leichte Usability-Probleme

Schweregrad 3: Schwere Usability-Probleme
Schweregrad 4: Katastrophal – Das Problem muss vor dem Einsatz des Produktes behoben werden

Durch eine heuristische Evaluation mit fünf Experten werden etwa 95% aller Usability-Probleme eines Produktes entdeckt.

Zu jedem Usability-Problem sollte eine mögliche Lösungsvariante angeboten werden. Die Ergebnisse der Evaluation werden im anschließenden Sprint Review Meeting besprochen und basierend darauf das Product Backlog, in Abstimmung mit dem Product Owner, angepasst.

5.2.3. Benutzer-Test

Zu jedem definierten Milestone sollte ein Benutzer-Test des aktuellen Product Increments durchgeführt werden. Das UCD-Team sollte, zusammen mit dem Product Owner, auf Basis der Anforderungen und Akzeptanzkriterien während jedes Sprints einen Testplan erstellen, bzw. erweitern. Auf Basis des Testplans sollten dann Benutzer-Tests in einem spezialisierten Labor durchgeführt werden. Die Tests werden von Mitgliedern des UCD-Teams begleitet. Product Owner und Mitglieder des Development Teams sollten ebenfalls anwesend sein.

Der Test sollte mindestens aus Beobachtung und anschließendem Interview des Probanden bestehen. Je nach Aufgabenstellung innerhalb des Milestone-Tests empfiehlt sich die Anwendung eines Eyetracking-Tests.

Während des Tests bekommt der Proband definierte Handlungsaufgaben (Testplan) die er innerhalb einer festgelegten Zeitspanne lösen soll. Während der Bearbeitung einer Aufgabe ist der Proband angehalten seine Handlungen und Erwartungen zu kommentieren. Der Test wird durch einen Interviewer begleitet und protokolliert.

Wird auch Eyetracking eingesetzt, werden während des Tests die Augenbewegungen des Probanden aufgezeichnet und zusätzlich zu den gemachten Beobachtungen ausgewertet. Durch einen Eyetracking-Test lassen sich Probleme in der Anordnung und der Sichtbarkeit von Elementen genauer analysieren als durch die einfache Beobachtung des Probanden.

Die während des Tests gemachten Beobachtungen werden analysiert und für die Planung des nächsten Sprints dokumentiert. Ggf. muss das Product Backlog angepasst und neu priorisiert werden.

5.3. Sprint Review Meeting & Abschluss des Sprints

Das Sprint Review Meeting kann wie bei Scrum vorgesehen abgehalten werden. Die im Sprint umgesetzten Backlog Items im Product Increment werden vorgestellt und diskutiert. Während des Sprints ermittelte Usability-Probleme und die Erkenntnisse aus den Testverfahren werden in das Product Backlog aufgenommen und in den folgenden Sprints berücksichtigt.

Kapitel

SECHS

Fallbeispiel

6.1. Projektbeschreibung

Die Cocomore AG sollte für einen Kunden ein Redesign einer E-Commerce-Seite durchführen. Entgegen des Namens ist der E-Market kein Shop. Er dient ausschließlich der Vermittlung von B2B-Kontakten zwischen Herstellern und Konsumenten von Industrie-Produkten.

Die E-Commerce-Seite setzt sich aus zwei Bereichen für unterschiedliche Zielgruppen zusammen:

- Frontend – Dient zur Produkt- und Firmensuche für Kunden der Hersteller von Industrie-Produkten
- Backend – Verwaltungstool der Produkt- und Firmendatenbank. Hersteller können hier Produkte für den E-Market anlegen und Pflegen.

Entwicklungszeit:	~ 900 Personentage,
Projektbeginn:	November 2011
Release-Termin:	September 2012

6.2. Produktanforderungen

Für die Angebotserstellung wurden durch den Kunden die folgenden Anforderungen festgelegt:

- Performance-Steigerung der Datenbank (MySQL)
- Redesign des Backends
  - Produkteingabe & Pflege
  - Firmenprofil
  - Benutzerrollen
  - Darstellung auf mobilen Endgeräten
  - Kategoriefilter
  - Semantische Suche
- Redesign des Frontends
  - Sub-E-Markets

  - Kategoriefilter
  - Semantische Suche
  - Suchmaschinenoptimierung
  - Darstellung auf mobilen Endgeräten
- Implementierung neuer Tools für Frontend und Backend
- OnSite-Tracking

6.3. Projektablauf

6.3.1. Rollen & Einteilung

Product Owner – Es wurden zwei Product Owner definiert. Ein Product Owner war der zuständige Projektleiter beim Kunden, da dieser jedoch auch andere Aufgaben wahrnehmen musste, wurde dem zuständigen Projektmanager der Cocomore AG ebenfalls diese Rolle zugewiesen. Durch diese Einteilung konnte sichergestellt werden, dass die Interessen des Kunden während des Prozesses vertreten wurden. Der Product Owner bei Cocomore wurde vor Beginn des Projektes in einem durch die Scrum Alliance angebotenen Test zertifiziert (Certified Product Owner).

Product Owner Team – Auf Grund der durch den Kunden definierten Anforderungen an das Produkt wurde der Product Owner durch ein eigenes Team für die Konzeption und Priorisierung des Product Backlog unterstützt. das UCD-Team wurde hier integriert.

Development Team – Das Team bestand aus sechs Entwicklern aus den Bereichen Datenbanken und Frontend-Entwicklung.

Scrum Master – Als Scrum Master wurde die stellvertrende Leiterin des IT-Bereichs eingesetzt. Sie ist zertifizierter Scrum Master und hat vorher schon Projekte mit Scrum erfolgreich bearbeitet.

6.3.2. Vorbereitung des Product Backlog

Die Erstellung des Product Backlogs wurde vorerst in zwei Gruppen durchgeführt. Eine Gruppe sollte die Anforderungen für die Überarbeitung der Datenbank, semantische Suche, sowie für alle weiteren technischen Belange des Projektes erstellen. Eine zweite Gruppe sollte in der Zeit die Anforderungen aller für Benutzer sichtbaren Bereiche erarbeiten.
Die semantische Suche wurde in dieser Gruppe bearbeitet da hier die technischen Voraussetzungen geklärt werden mussten.
In einer zweiten Stufe sollten die Anforderungen in ein Product Backlog überführt werden und auf Basis der beiden Anforderungsteile ergänzt werden.
Das UCD-Team nahm an der Erarbeitung der für Benutzer sichtbaren Bereiche teil.

Auf Grund des Projektumfangs und der Projekteigenschaften wurde die Anforderungsdefinition der sichbaren Bereiche vorab in die Frontend und Backend aufgeteilt und zwei Affinity-Diagramme erstellt.
Die für das Frontend erarbeiteten Anforderungen beeinflussten in großem Umfang auch die Anforderungen für das Backend, so dass dieses Vorgehen für alle Beteiligten sinnvoll erschien.

Da der Bereich des Backends auf Grund von Benutzerverwaltung, Produkteingabe und weiteren Features sehr umfangreich wurde (etwa 1500 Affinity Notes), wird im weiteren Verlauf dieses Kapitels nur auf den Frontend-Bereich eingegangen.

6.3.2.1. Definition der Anforderungen

Durch den Kunden wurde bereits eine umfassende Nutzeranalyse durchgeführt. Das UCD-Team der Cocomore AG nahm in diesem Fall nur eine erneute Auswertung der Nutzeranalyse vor und übertrug diese in ein Affinity-Diagramm.
An der Entwicklung des Affinity-Diagramms nahmen, neben dem UCD-Team, der verantwortliche Projektmanager sowie drei Entwickler teil. Da dies für den Projektmanager und die Entwickler das erste mal war das sie an der Erstellung eines Affinity-Diagramms teilnahmen wurde ihnen die Methode gründlich erklärt.

Das Affinity-Diagramm aus rund 300 Affinity Notes erstellt und in 58 blaue Labels sowie 13 pinkfarbene Labels eingeteilt. Auf eine weitere Clusterung in grüne Labels wurde verzichtet.
Während der Erstellung der Affinity Notes und der Benennung der Labels wurde durch die Beteiligten darauf geachtet das diese bereits im Hinblick auf das Product Backlog formuliert wurden.

Während der Entwicklung des Affinity-Diagramms traten Probleme hinsichtlich der Benennung der einzelnen Labels auf. Dies lies sich jedoch auf die mangelnde Erfahrung der Beteiligten mit Affinity-Diagrammen zurückführen.

Im Anschluss wurde das Affinity-Diagramm für das Backend erstellt. Hier ergaben sich etwa 1500 Affinity Notes, 300 blaue, 50 pinkfarbene und 13 grüne Labels.

6.3.2.2. Übertragung ins Product Backlog

Da die Notes und Labels bereits als User Stories formuliert waren, gestaltete sich die Übertragung in das Product Backlog relativ unkompliziert: Aus den pinkfarbenen Labels wurden wie vorgesehen die Epics gebildet. Die darunterliegenden blauen Labels konnten als Backlog Items (Anforderungen) übernommen werden.
Einige der Affinity Notes konnten ebenfalls als Akzeptanzkriterien übernommen werden.

Nach der Übernahme der, durch die andere Gruppe, definierten Anforderungen für den technischen Teil, wuchs das Product Backlog auf etwa 650 Backlog Items in 60 Epics an. Abschließend wurden die Backlog Items vom Product Owner in Abstimmung mit dem Developer Team priorisiert.

Die Priorisierung sah vor, im ersten Sprint die technischen Rahmenbedingungen für die weiteren Entwicklungsschritte zu schaffen. Danach sollten sukzessive die Bereiche Backend und daran anschließend das Frontend entwickelt werden.

Um den Teams Zeit zu geben die Backlog Items für den Sprint vorzubereiten (Definition von Akzeptanzkriterien, Spezifikation der am höchsten priorisierten Backlog Items, etc.) wurde für das Sprint Planning Meeting und der Beginn des ersten Sprints ein Termin 14 Tage später angesetzt.
Das Development Team nahm in dieser Zeit ebenfalls erste Zeitschätzungen für die hoch priorisierten Backlog Items vor um dem Product Owner die Auswahl für den anstehenden Sprint zu erleichtern.

6.3.2.3. Vorbereitung auf den ersten Sprint

Das UCD-Team spezifizierte in der Zeit bis zum ersten Sprint auf Basis von Gestaltungsrichtlinien die Startseite und das Navigationskonzept des Backends.

Die Startseite des Backends wurde auf Grund von 10-15 Hauptbereichen[1] als Dashboard konzipiert. Über das Dashboard sollten alle Bereiche des Backends erreichbar sein. Die jeweiligen Bereiche sollten eine linke Navigationsleiste erhalten, sowie ein Dropdown-Menü um schnell zu anderen Bereichen wechseln zu können.

6.3.3. Sprint Planning Meeting

Während des ersten Sprint Planning Meetings wurden durch den Product Owner die für den ersten Sprint ausgewählten Backlog Items vorgestellt.
Die Vorstellung der Backlog Items sowie die Besprechung der Items nahmen etwa 4 Stunden in Anspruch.

Während des zweiten Teils des Sprint Planning Meetings wurden erste Umsetzungsvorschläge der für den Sprint vorgesehenen Backlog Items durch das Development Team erläutert.
Anschließend fand bei einigen der Backlog Items eine weitere Aufteilung in zu erledigende Tasks statt.
Als Sprintziel wurde die vollständige Installation und Einrichtung des verwendeten Content Management System (CMS) definiert.

Auf Grund der Vorbereitungszeit bis zum ersten Sprint Planning Meeting konnte das UCD-Team im Bereich des Product Owner verbleiben. Dies änderte sich auch nicht in späteren Sprints, da das UCD-Team die Spezifikation der für den nächsten Sprint ausgewählten Backlog Items immer direkt mit dem Product Owner absprechen und für das Sprint Planning Meeting vorbereiten konnte.

[1] lt. Jacob Nielsen sollten nicht mehr als 5-9 Menü-Elemente verwendet werden.

6.3.4. Sprint

Während der Sprints nahm das UCD-Team nur beratende Funktionen des Development Teams wahr. Spezifikationen die während ein Sprints auf Grund von technischen Gegebenheiten geändert werden mussten wurden in das Product Backlog übernommen. Änderungen ergaben sich häufig durch die Verwendung von JavaScript-Funktionen oder Gestaltungsproblemen mit CSS2, da hier einige Funktionen oder Layoutüberlegungen nicht in angemessener Zeit umsetzbar waren.

An den Daily Scrums nahm immer mindestens ein Mitglied des UCD-Teams teil um direkt auf Fragen, die die Usability beeinflussen, reagieren zu können und ggf. im Anschluss an das Meeting eine Lösung mit den Entwicklern zu erarbeiten.
Hier hat sich gezeigt, dass der Beratungsbedarf des Development Teams, wie zu erwarten, am Anfang recht ausgeprägt war sobald es um Probleme ging, die auch die Usability des Produktes oder einer Funktion beeinflussten. Nach und nach wurden jedoch durch das Development Team immer mehr Lösungen für derartige Probleme entwickelt, welche auch aus Sicht des UCD akzeptabel bis sehr gut waren.

In späteren Sprints nahm das UCD-Team vor dem Abschluss eines Sprints heuristische Evaluationen des Product Increment vor. Der Einsatz von heuristischen Evaluationen am Ende eines Sprints hat sich als eine sinnvolle Praktik herausgestellt. Durch die Schrittweise Spezifikation einzelner Funktionen (Epics) und Seiten ergaben sich z.B. Inkonsistenzen bei der Benamung von Buttons oder Interaktionsabläufen. Diese konnten mittels der heuristischen Evaluation erkannt und teilweise noch vor Beendigung des Sprints gelöst werden.

Neben der beratenden Tätigkeit konnte das UCD-Team an der weiteren Spezifikation der Backlog Items arbeiten und teilweise auch andere Projekte bearbeiten. Kapazitätsprobleme traten nicht auf.

6.3.5. Sprint Review & Sprint Retrospective

Durch die, während des Sprints vorgenommenen, Tests konnte im Sprint Reviews in der Regel ein finales Product Increment vorgestellt werden.

Im weiteren Verlauf des Projektes hat sich gezeigt, dass auf Grund der ausführlichen Nutzeranalyse und Anforderungsdefinition in der Anfangsphase praktisch keine Änderungen an Anforderungen hinsichtlich der sichtbaren Bereiche ergeben haben.

Während der im Anschluss stattfindenden Sprint Retrospective wurde das System zur Integration mit UCD vom Product Owner, sowie auch dem Development Team als positiv beschrieben. Verzögerungen durch fehlende oder unzureichende Spezifikationen seien nicht aufgetreten.

Kapitel

SIEBEN

Fazit

Nach der eingehenden Betrachtung des Scrum- und UCD-Prozesses lässt sich sagen, dass die beiden Prozesse sich gut ergänzen können, und die Prinzipien und Methoden des UCD dem Projektmanagement mit Scrum helfen kann. Während der Umsetzung tauchten jedoch auch Probleme, durch die unterschiedlichen Ansätze der beiden Prozesse, auf.
Da die Cocomore AG eine Full-Service-Agentur ist, und mehrere Projekte parallel bearbeitet, kann nicht sichergestellt werden, dass die notwendigen Kapazitäten im Bereich UCD kurzfristig zur Verfügung stehen. Durch eine intensive Kommunikation zwischen Development Team, Product Owner und Scrum Master lassen sich solche Engpässe jedoch frühzeitig identifizieren und können berücksichtigt werden.

Eine weitere Hürde kann im Bereich der Erstellung des Product Backlog liegen. Die Vorbereitung und Definition der Anforderungen durch Methoden des UCD nimmt mehr Zeit in Anspruch als die klassische Anforderungsdefinition durch den Product Owner.
Die möglichst genaue und frühe Analyse und Definition von Anforderungen bietet für Auftraggeber und Auftragnehmer jedoch eine gute Möglichkeit, ein aus funktionaler und nicht-funktionaler Sicht, vollständiges Produkt zu entwickeln.
Die Definition funktionaler und nicht-funktionaler Anforderungen durch eine professionelle Anforderungserhebung kann den Zeit- und Kostenfaktor von Projekten reduzieren.
Ein Product Backlog mit nahezu vollständig definierten funktionalen und nichtfunktionalen Anforderungen muss im Projektverlauf meist nur minimal angepasst werden. Durch diesen Umstand reduziert sich die Wahrscheinlichkeit für ein komplettes Redesigns eines Produktes, bzw. einzelner Funktionen, auf Grund von Anforderungsänderungen oder Ergänzungen.
Nach einer Studie von Jacob Nielsen aus dem Jahr 1993 sind die vier häufigsten Gründe für die Nichteinhaltung von Zeit- und Kostenplanung auf eine unzureichende Einbindung von UCD-Methoden zurückzuführen.
Eine weitere Studie, über Erfolgsfaktoren von Softwareprojekten der Otto-von-Guericke-Universität Marburg aus dem Jahr 2008, fand heraus, das Projekte die mit agilen Methoden umgesetzt werden zu etwa 58% erfolgreich abgeschlossen werden. Projekte ohne agile Methoden haben laut der Studie eine Erfolgsquote

von 18%.
Führt man die Ergebnisse der beiden Studien zusammen, so lässt sich sagen, das durch den Einsatz von UCD und Scrum, bzw. agilen Methoden, die Projektkosten weiter gesenkt werden können und das Projekt mit hoher Wahrscheinlichkeit erfolgreich abgeschlossen werden kann.

Des weiteren lassen sich auf diese Art Produkte entwickeln, die dem Bedarf der Benutzer gerecht werden und keine, aus Benutzersicht, unnötigen Funktionen enthalten. Funktionen, die der Benutzer nicht erwartet oder braucht, können ausgelassen werden.
Nach einer Studie von Mauro New Media aus dem Jahr 2002 werden üblicherweise nur etwa 5% der in einer Software enthaltenen Funktionen 95% der Zeit genutzt. 70% der Funktionen werden nur selten oder gar nicht benutzt.

Durch die Verwendung von UCD-Methoden im Scrum-Prozess ergeben sich auch weitere Vorteile. Die von Scrum vorgesehenen Tests auf korrekte Funktion des Product Increment können durch Usability-Tests erweitert werden. Hierdurch wird ein Produkt nicht nur auf Funktionalität, sondern auch auf die Benutzbarkeit hin getestet.
Zum Beispiel bei Produkten die in Unternehmen eingesetzt werden, lässt sich durch diese Tests sicherstellen, dass der Benutzer seine Arbeit effizient(er) und effektiv(er) erledigen kann. Durch eine Steigerung dieser beiden Faktoren lassen sich zwar die Kosten für die Produktentwicklung nicht senken, jedoch kann die Arbeitsbelastung der Mitarbeiter gesenkt werden (Stichwort Nachhaltigkeit). Zusätzlich kann die Produktivität eines Mitarbeiters gesteigert werden.
Werden bei der Layout-Erstellung für eine Anwendung zusätzlich noch, durch die in der DIN EN ISO 9241-110 definierten, Anforderungen an die Gebrauchstauglichkeit (siehe 3.2.3.2, 9, ff.), Richtlinien eingehalten, reduziert sich ebenfalls der Dokumentations- und Schulungsbedarf und somit die Kosten eines Produktes.
Durch das ausführliche Testen, auf funktionale und nicht-funktionale Anforderungen, lässt sich ebenfalls einer der größten Kostenfaktoren, Support und Fehlerbehebung, nach einer Produktveröffentlichung reduzieren. Durch Tests der funktionalen und nicht-funktionalen Anforderungen lassen sich technische und Usability-Probleme bereits früh in der Entwicklung feststellen. Die Aufwände für Support und Fehlerbehebung können dadurch gesenkt werden.

Neben den direkten monetären Vorteilen, welche die Verbindung von Scrum und UCD bietet, existieren auch „weichere“ Verbesserungen. Im Forrester Report 2000b, Retail & Media Data Overview, wurden Kunden von E-Commerce-Platformen auch zum Thema Usability befragt und 42% der Befragten gab an, dass eine gute Usability für sie ein wichtiges Kriterium für die Nutzung einer Webseite sei. Da unter Usability nicht nur die einfache Benutzbarkeit sondern auch die korrekte Funktion eines Produktes verstanden werden kann, können durch die Verbindung von Scrum und UCD auch Umsatzzahlen und Conversion-Raten verbessert werden.

Insgesamt bedeutet zwar die Integration des UCD-Prozesses Anpassungen im Ablauf eines Scrum-Projektes (Rollenwechsel des UCD-Teams), allerdings können die-

se Anpassungen den Scrum-Prozess in hohem Maße ergänzen und verbessern.
Die Verbindung von Scrum und UCD bietet für alle Produktbeteiliten Vorteile. Auftraggeber sparen z.B. durch geringeren Dokumentations & Schulungsaufwand und sinkenden Bedarf an Problembehebung nach einem Release. Entwickler können ihre Zeit für die Entwicklung neuer Funktionen einsetzen statt alte zu ändern und Benutzer erhalten ein funktionsfähiges und benutzbares Produkt das ihren Bedarf berücksichtigt.

Abbildungsverzeichnis

Abkürzungsverzeichnis

Literaturverzeichnis

[1] Brown, Dan M.: *Konzeption und Dokumentation erfolgreicher Webprojekte*. mitp / bhv, 2009. – ISBN 9783826655074

[2] Europäisches Komitee für Normung (Hrsg.): *EN ISO 9241 Ergonomie der Mensch-System-Interaktion*. Europäisches Komitee für Normung, 2006 9

[3] Garrett, Jesse J.: *The elements of user experience: user-centered design for the web*. New Riders, 2010. – ISBN 9780321683687

[4] Holtzblatt, Karen ; Wendell, Jessamyn B. ; Wood, Shelley: *Rapid contextual design: a how-to guide to key techniques for user-centered design*. Elsevier, 2005. – ISBN 0123540518

[5] Khazaeli, Cyrus D.: *Systemisches Design: Intelligente Oberflächen für Information und Interaktion*. Rowohlt Taschenbuch Verlag, 2005. – ISBN 3499600781

[6] Krug, Stephen: *Don't make me think!: a common sense approach to web usability*. New Riders, 2006. – ISBN 9780321344755

[7] Nebe, Karsten ; Duchting, Markus ; Zimmermann, Dirk: Integration von User Centred Design Aktivitäten in Agile Softwareentwicklung. In: *Usabiltiy Professionals* (2007) 25

[8] Richter, Michael ; Flückiger, Markus: *Usability Engineering kompakt*. Spektrum Akademischer Verlag Heidelberg, 2010. – ISBN 0123540518

[9] Sutherland, Jeff: *Scrum Handbook*. Scrum Training Institute, 2010

[10] Sutherland, Jeff ; Schwaber, Ken: *The Scrum Guide*. scrum.org, 2011 24

[11] Wikipedia: Agile Softwareentwicklung. In: *http://de.wikipedia.org/wiki/Agile_Softwareentwicklung*

Anhang

A

Heuristiken nach Nielsen & Molich

1. Stelle einen einfachen und natürlichen Dialog her!

Ästhetik und minimalistisches Design: Dialoge sollten keine irrelevanten Informationen enthalten, da die Informationen um die Aufmerksamkeit des Benutzers konkurrieren.

„Less is more"- keine irrelevanten oder wenig benötigten Informationen. Information in einer natürlichen und logischen Ordnung.

- Grafische Gestaltung: Wichtigste Elemente herausstehend
- Großbuchstaben können Aufmerksamkeit auf sich ziehen, sind aber 10% langsamer zu lesen
- Interfaces sollen ohne farbige Darstellung funktionsfähig sein, farbcodierte Info durch weiteren Reiz ergänzen.

2. Sprich die Sprache der Benutzer!

Übereinstimmung zwischen dem System und der realen Welt: Das System sollte die Sprache des Benutzers sprechen, und zwar nicht mit systemorientierter Terminologie, sondern mit Worten, Phrasen und Konzepten, die den Benutzern vertraut sind. Dabei soll die natürliche und logische Reihenfolge eingehalten werden.

Der Dialog sollte in klaren Worten und Konzepten ausgedrückt werden, die sich eher am Benutzer als am System orientieren.

- Wenig Fremdsprachen, unübliche Wortbedeutungen vermeiden
- Für spezialisierte User eigene spezialisierte Terminologie verwenden
- Für spezialisierte User eigene spezialisierte Terminologie verwenden
- Interaktionen aus der Benutzerperspektive zeigen: „Sie haben .. gekauft" statt „Wir haben Ihnen ... verkauft"
- Keine Namensgebungsrestriktionen, wenn dies nicht möglich, konstruktive Fehlermeldungen
- Metaphern erleichtern Repräsentation aber können auch fehlleiten

3. Minimiere die „Gedächtnislast“ der Benutzer!

Wiedererkennen, statt sich erinnern: Objekte, Optionen und Aktionen sollten sichtbar sein. Die Benutzer sollten sich nicht an Informationen aus einem früheren Teil des Dialogs mit dem System erinnern müssen. Instruktionen sollen sichtbar oder leicht auffindbar sein.

Das Kurzzeitgedächtnis der Benutzer ist begrenzt. Sie sollten sich deshalb nicht an Information erinnern müssen, die in einem ganz anderen Bereich des Dialogs von Bedeutung war. Erkennen funktioniert besser als aktives Erinnern.

- Sichtbarkeit der Objekte für den Nutzer
- Instruktionen für den Umgang mit dem System sollten sichtbar oder leicht erreichbar sein. Bei verlangtem Input sollte das System das Format beschreiben und ein Beispiel geben (z.B. Geburtsdatum).
- Das System sollte auf wenigen, weit verbreiteten Regeln basieren. (Bsp: Kommando „Paste“)

4. Sei konsitent!

Konsistenz und Standards: Benutzer sollten sich nicht fragen müssen, ob verschiedene Begriffe oder Aktionen dasselbe bedeuten. Deshalb sind Konventionen einzuhalten.

- Interne Konsistenz: Dasselbe Kommando hat immer den gleichen Effekt. Dieselbe Info soll an derselben Stelle auf allen Bildschirmen und Dialogboxen auf die gleiche Weise formatiert sein.
- Externe Konsistenz: Konventionen die auch in anderen Anwendungen als Standards gelten werden eingehalten.
- Regel bezieht sich nicht nur auf das Screendesign, sondern auch auf Aufgaben und Erwartungen.

5. Liefere Feedback!

Sichtbarkeit des Systemstatus: Das System soll die Benutzer ständig darüber informieren, was geschieht, und zwar durch eine angemessene Rückmeldung in einem vernünftigen zeitlichen Rahmen.

Das System sollte den User kontinuierlich darüber informieren was es tut und wie es den Input des Users verarbeitet.

- Seitenüberschriften, Info darüber, wo man sich gerade befindet
- Feedback nicht erst bei Fehlern, sondern auch positives oder partielles Feedback.
- Antwortzeit (200 ms). Bei langer Antwortzeit ist Feedback besonders wichtig
- Wenn ultimatives Feedback nicht möglich ist, sollte ein Fortschrittsbalken verwendet werden.
- Systemfehler: auch bei Systemabsturz sollte informatives Feedback gegeben werden.

6. Stelle klar markierte Ausgänge zur Verfügung!
Benutzerkontrolle und -freiheit: Benutzer wählen Systemfunktionen oft fälschlicherweise aus und benötigen einen „Notausgang“, um den unerwünschten Zustand wieder zu verlassen.

Benutzer werden immer Fehler machen, unabhängig davon, wie gut das Interface ist. Exits unterstützen Lernen, da gefahrlos Alternativen und Operationen ausprobiert werden können.

- Exits sollten visualisiert sein
- alle Dialogboxen sollten Cancel-Buttons haben
- Escape sollte nutzbar sein, um in den letzten Zustand zurückzugelangen
- Undo sollte weitverbreitet möglich gemacht sein
- Unterbrechung von Aktionen muss möglich sein
- Hohe Responsivität Systems: Das System sollte auf neuere Aktionen mit höherer Priorität reagieren.

7. Stelle Abkürzungen zur Verfügung!
Flexibilität und Effizienz der Benutzung: Häufig auftretende Aktionen sollten vom Benutzer angepasst werden können, um Fortgeschrittenen eine schnellere Bedienung zu erlauben.

Obwohl Systeme mit der Kenntnis einiger weniger Regeln bedienbar sein sollen, muss es für erfahrene Nutzer die Möglichkeit geben, häufig ausgeführte Operationen über Shortcuts schnell auszuführen:

- Funktions-/ Kommandotasten
- Buttons für Zugriff auf wichtige Funktionen
- Makros
- User sollten ihre Interaktionsgeschichte wieder verwenden können
- Persistenz

8. Liefere gute Fehlermeldungen!
Hilfe beim Erkennen, Diagnostizieren und Beheben von Fehlern: Fehlermeldungen sollten in natürlicher Sprache ausgedrückt werden (nicht nur in Fehlercodes), präzise das Problem beschreiben und konstruktiv eine Lösung vorschlagen.

Hilf den Benutzern, Fehler zu erkennen und zu verhindern. Fehlermeldungen bieten die Möglichkeit, beim Verständnis des Systems zu helfen. Fehlermeldungen sollten nicht mysteriös sein und für den Benutzer unverständliche „innere“ Computerzustände anzeigen.

- Präzise statt vage und generell!
- Statt „Kann die Datei nicht öffnen“: „Kann „XX“ nicht öffnen, weil sich die Datei nicht auf dem Datenträger befindet.“
- Sollten dem Benutzer konstruktiv helfen, das Problem zu lösen.
- Fehlermeldungen sollten höflich sein und den Benutzer nicht einschüchtern oder beschuldigen.

9. Verhüte Fehler!

Fehlerverhütung: Noch besser als gute Fehlermeldungen ist ein Design, das Fehler verhütet.

- Viele Situationen sind als fehlerprovozierend bekannt. Systeme sollten so gestaltet sein, dass die Benutzer erst gar nicht in diese Situationen kommen.
- Beispiel: Eingabe von Worten - Unterstützung durch Anzeige von Dateinamen mit Auswahlmöglichkeit.
- Autovervollständigen

10. Liefere Hilfe und Dokumentationen!

Hilfe und Dokumentation: Jede Information der Hilfe oder Dokumentation sollte leicht zu finden sein, auf die Aufgabe abgestimmt sein und die konkreten Schritte zur Lösung auflisten. Außerdem sollte sie nicht zu lang sein.

Benutzer brauchen schnelle, aufgabenorientierte Hilfe, die mit Beispielen arbeitet und leicht zu finden ist.

1. Suchen: Index, Inhaltsverzeichnis oder Überblick-Diagramm müssen vorhanden sein. Terminologie des Systems und viele Synonyme (auch von anderen Herstellern) anzeigen. Verlinkte Themen.
2. Verstehen: Aufgabenorientierte Beschreibung, viele Beispiele.
3. Anwenden: Hilfesystem sichtbar in separatem Fenster. Möglichkeit des Kopierens der Beispiele um das dargestellte direkt auszuführen. Check, ob korrekt ausgeführt.