

Anforderungsanalyse

für das
Labwork Management

PRAXISPROJEKT

ausgearbeitet von
Florian Herborn

vorgelegt an der
TECHNISCHE HOCHSCHULE KÖLN
CAMPUS GUMMERSBACH
FAKULTÄT FÜR INFORMATIK UND
INGENIEURWISSENSCHAFTEN

im Studiengang
ALLGEMEINE INFORMATIK

Prüfer: Prof. Dr. Christian Kohls
Technische Hochschule Köln

Gummersbach, im Juni 2017



Adressen: Florian Herborn
Parkweg 8
51491 Overath
florianHerborn2293@t-online.de

Prof. Dr. Christian Kohls
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.kohls@th-koeln.de

Kurzfassung

Das Ziel der Vorliegenden Arbeit war es, Anforderungen für das Labwork Management (LWM) des ADV-Labor^s zu ermitteln. Dazu wurden 13 ausgewählte, potentielle Stakeholder in Interviews über ihre Wünsche und Interessen, die dieses Tool betreffen befragt. Die dort ermittelten Erkenntnisse wurden anschließend ausgewertet um als Grundstein für die weitere Entwicklung des Tools zu dienen. Dazu wurden diese mittels User-Storys dokumentiert, und mit Story-Points und Kano-Faktoren priorisiert. Des Weiteren wurde eine Webseite entwickelt, die Zahlreiche Funktionalitäten zur Verarbeitung der Anforderungen bereitstellt. Die Ergebnisse dieser Arbeit zeigen, dass in dem aktuellen LWM noch viel Potential steckt, was mit der Umsetzung einiger dieser Anforderungen geweckt werden kann. Diese Arbeit ist deshalb sowohl für Studierende der Informatik, als auch für Lehrende in diesem Bereich und Nutzer des LWM interessant.

So hat man diese Korrektur zu lesen :

1. Ein komplettes Wort, das gelb markiert ist, sollte wegfallen.
2. Ein Wort dessen Anfangsbuchstabe markiert ist, hat falsche groß-klein-Schreibung.
3. Ein Wort mit einem oder mehreren markierten Buchstaben beinhaltet an dieser,(n) Stelle(n) Rechtschreibfehler.
4. Zu ergänzende Buchstaben habe ich einfach hinter das Wort geschrieben (siehe oben). Sonderfall : Rechtschreibfehler am Anfang des Wortes (siehe unten).
5. Leertasten, die markiert sind, bedeuten, dass ein Komma fehlt. Dieses befindet sich auch am linken Rand, um besser darauf aufmerksam zu machen.
6. Alle anderen Markierungen müsste ich kommentiert haben.

Englische Wörter, die auf „y“ enden, schreibt man im Plural mit „ie“.

Der Fehler mit dem Wort „Stories“ zieht sich durch die gesamte Arbeit.

Ich glaube zwar, dass ich alle erwischt und markiert habe, aber ich empfehle ctrl+f auf „Storys“ Gleiches bei TH Köln. Im Logo schreibt sich das exakt so. In der Arbeit hab ich mehrfach TH-Köln oder TH-Koeln gelesen. Auch hier empfehle ich ctrl+f auf „TH“

Inhaltsverzeichnis

Vorwort	6
1. Motivation	7
1.1. Ausgangssituation	7
1.2. Problemstellung	7
1.3. Lösungsansatz	7
2. Theorie	8
2.1. Requirements Engineering	8
2.1.1. Einfluss von Requirements Engineering auf das Projekt	8
2.2. Stakeholder	8
2.3. Anforderungen	9
2.3.1. Funktionale Anforderungen	10
2.3.2. Nicht-Funktionale Anforderungen	10
2.4. Kano-Modell	11
2.5. Penalty-Reward-Faktoren Ansatz	13
2.6. Anforderungen Ermitteln	13
2.6.1. Kreativitätstechniken	14
2.6.2. Beobachtungstechniken	18
2.6.3. Befragungsrechniken	18
2.7. Dokumentation	20
2.7.1. User-Storys	20
2.7.2. Use-Cases	21
2.7.3. Technical Storys	21
2.7.4. FunctionsMASTeR	21
2.8. Validation	21
2.9. Auswertung	22
2.9.1. Function-Point-Analyse	22
2.9.2. Story-Points	22
3. Umsetzung	24
3.1. Ermittlung	24
3.2. Zielgruppe	26
3.3. Dokumentation	26
3.3.1. Formulierung	26
3.4. Auswertung	28
3.4.1. Validation	28
3.4.2. Gruppierung	28
3.4.3. Priorisierung	31

3.4.4. Aufwandsschätzung	35
4. Visualisierung der Ergebnisse	36
4.1. Excel	36
4.2. Webseite	37
4.2.1. Gruppen	37
4.2.2. Labels	38
4.2.3. Informationen	39
4.2.4. Filter/Suche	41
4.2.5. Sortierung	42
4.2.6. Sichtbarkeit	43
5. Ergebnisse	44
Abbildungsverzeichnis	48
Glossar	49
Literaturverzeichnis	54
ANHANG	55
A. Fragebogen	56
Eidesstattliche Erklärung	58

Vorwort

Vor Ihnen liegt das Praxisprojekt „Anforderungsanalyse für das LWM des ADV-Labors der TH-Köln“. Die in dieser Dokumentation beschriebene Forschungsarbeit wurde in enger Zusammenarbeit mit dem Entwickler-Team des LWM durchgeführt, in dem auch ich seit einem Jahr als Backend-Entwickler tätig bin.

Für dieses **A**ngagement möchte ich meinen Kollegen Uwe Müsse und Alexander Dobrynin danken. Ohne ihr Zutun wäre ein großer Teil des hier erreichten Ergebnisses nicht in dieser Form zustande gekommen. Auch haben sie mir Hilfestellungen und Denkansätze in schwierigen Situationen gegeben.

Im Laufe dieses Projekts wurden zahlreiche Interviews mit Modulverantwortlichen und ihren Mitarbeitern geführt. Diese Interviews haben im Schnitt 1 - 1,5 Stunden gedauert, trotzdem haben mir diese Personen bereitwillig zur Verfügung gestanden, weshalb auch ihnen mein Dank gilt. Ohne diese Interviews wäre der Kern dieser Arbeit nicht realisierbar gewesen.

Ein weiterer Dank gebührt meiner Freundin, die **mit** in den Abschlussarbeiten dieser Arbeit den Rücken frei gehalten hat. Nur so war es mit Mögliche diese rechtzeitig fertig zu stellen.

Ich wünsche Ihnen viel Spaß beim Lesen.

Florian Herborn

1. Motivation

1.1. Ausgangssituation

Das aktuelle LWM wurde für die Praktika des ADV-Labors am Campus Gummersbach der TH-Koeln entwickelt. Ziel des Tools ist es, die Massenabfertigung eines Praktikums zu vereinfachen. Dies soll geschehen, indem der Staffelplan des Praktikums automatisch generiert wird und in Zuge dessen Konflikte mit anderen Praktika bestmöglich eliminiert. Es bietet zudem die Möglichkeit, Praktikumsteilnehmern die Anwesenheit an einem Termin, seine Leistungen sowie die erlangten Bonuspunkte zu vermerken.

1.2. Problemstellung

Da das LWM in erster Linie für das ADV-Labor entwickelt wurde, deren Praktika eine gleichbleibende Struktur aufwies, wurde das Tool genau für dieses Praktikumsschema modelliert. Da sich mittlerweile jedoch nicht nur die Struktur der Praktika des ADV-Labors verändert haben, sondern auch zunehmend mehr Außenstehende Module das LWM nutzen, ist das Tool zum derzeitigen Stand nicht mehr in der Lage, die Anforderungen der Nutzer optimal zu erfüllen.

1.3. Lösungsansatz

Um herauszufinden welche Funktionalitäten das LWM beinhalten muss, damit den Anforderungen aller Nutzer entsprochen wird, ist ein Verfahren Notwendig, welches die Wünsche und Meinungen der Nutzer in die Planung einbezieht. Dort bietet sich eine Anforderungsanalyse an. Deren Ergebnisse können anschließend zur Verbesserung der Nutzbarkeit des Tools beitragen. Mit einer anschließenden Aufwandsschätzung und Priorisierung können die wichtigsten Anforderungen herausgefiltert werden, um sie daraufhin umzusetzen.

Einheitlichkeit
Siehe Logo TH Köln

Wiederholung!
Aufsplitten in 2
Sätze und mit
„Aus diesem Grund“
verbinden

Damit es den
Anforderungen
entspricht

miteinbezieht

Sätze mit Komma
verbinden

2. Theorie

2.1. Requirements Engineering

Im Requirements Engineering (RE) werden die Anforderungen an ein Produkt, einen Prozess oder am Prozess beteiligte Personen definiert. Die daraus resultierenden Erkenntnisse dienen als Leitfaden für das zu entwickelnde System.

Mit dem RE wird dafür gesorgt, dass das Endprodukt des Projekts auch den geforderten Anforderungen entspricht. Ist das nicht der Fall, führt dies in der Regel dazu, dass ein Projekt nicht den gewünschten Erfolg bringt. Gerade deshalb ist dem RE ein angemessenes Maß an Aufmerksamkeit zu widmen.

2.1.1. Einfluss von Requirements Engineering auf das Projekt

Bei dem RE werden die Wünsche der Stakeholder schon in der Planungsphase in das Projekt mit^{einbezogen} und berücksichtigt. So können die Funktionalitäten genau auf die Nutzer des Systems abgestimmt und klar definiert werden. Diese Tatsache macht das RE zu einem wichtigen Abschnitt in der Entwicklung eines Software-Projekts. Auch ⁱⁿ existierenden Systemen kann das RE für eine Verbesserung der Usability und Stakeholder-Zufriedenheit sorgen, weshalb es in vielen Projekten in mehreren Iterationen ausgeführt wird. Die Anforderungen können nach der Umsetzung mit Techniken der **MCI!** (**MCI!**) getestet werden, um herauszufinden, ob sie auch tatsächlich der Intention der Nutzer entsprechen und für diese nutzbar umgesetzt wurden.

2.2. Stakeholder

Als Stakeholder werden, wie Dr. Jürgen Fleig (2016) definiert, alle Personen, Gruppen oder Institutionen bezeichnet, die von den Aktivitäten eines Unternehmens direkt oder indirekt betroffen sind oder die irgendein Interesse an diesen Aktivitäten haben.

Es gibt drei Arten von Stakeholdern (Schäfer 2008), Primäre-, Sekundäre- und Tertiäre Stakeholder. Die Unterscheidung liegt hier vor allem bei der Größe des Bezugs zum Projekt.

Primäre Stakeholder

Im Text würde ich es
nicht so machen.

Außerdem Einheitlichkeit

Die primären Stakeholder werden, wie auch Bornemann, Stefan (2017) berichtet, manchmal als "*die wichtigsten Beteiligten*" bezeichnet. Diese Gruppe besteht aus den Beteiligten, welche den engsten Bezug zu den Zielsystem haben. Aus diesem Grund haben deren Anforderungen auch den größten Einfluss auf dieses. Wenn diese nicht erfüllt werden, verliert das Ziel-System die wichtigste Nutzer-Gruppe.

Sekundäre Stakeholder

Im Gegensatz zu den primären sind sekundäre Stakeholder nur indirekt in einer Beziehung zu dem Projekt. Nehmen wir an, es würde ein Mehrfamilienhaus gebaut werden, so wären die zukünftigen Bewohner dieses Hauses und der Bauherr primäre Stakeholder und lokale Baufirmen, welche gegebenenfalls durch dieses Projekt einen Auftrag bekommen könnten, die sekundären.

Tertiäre Stakeholder

Tertiäre Stakeholder oder auch "*externe*" genannt stehen auf dem ersten Blick in keinem Bezug zu dem Projekt. Jedoch kann es sein, dass es Personen, Gruppen oder Organisationen gibt, welche dennoch ein berechtigtes Interesse an einem solchen Projekt haben. Dies könnten neben Behörden auch öffentliche Personen oder Organisationen sein. In dem zuvor genannten Beispiel des Baus eines Mehrfamilienhaus stellt das Bauamt einen tertiären Stakeholder dar. Gehen wir weiter davon aus, dass das Mehrfamilienhaus besonders umweltschonend gebaut werden soll, so könnten auch Umweltaktivisten als Stakeholder in Betracht gezogen werden, da diese zwar nicht direkt was mit dem Bau dieses Hauses oder dem Ergebnis in Verbindung stehen, sie dennoch ein berechtigtes Interesse an der Angelegenheit haben.

2.3. Anforderungen

$x = y + x?$

Voraussetzungen
würde hier passen

an

Anforderungen sind Wünsche und Anforderungen über die Funktionalität, Qualität und Randbedingungen, die Stakeholder von einer Anwendung fordern. Diese Anforderungen haben Einfluss auf die Usability und den Erfolg eines Projekts. Werden beispielsweise wichtige Anforderungen nicht erfüllt, so führt dies dazu, dass die Nutzer unzufrieden sind oder sogar auf alternative Anwendungen zurückgreifen. Anforderungen sind nicht nur Funktionen, die die Stakeholder sich wünschen, sondern auch Qualitätsmerkmale die vorausgesetzt werden. Auch Gesetze oder andere Randbedingungen können zu Anforderungen werden, sofern sie einen Einfluss auf das Projekt oder dessen Ergeb-

Abschnitten

nis haben. Grob wird unterschieden zwischen funktionalen Anforderungen und nicht-funktionalen Anforderungen welche in den nächsten **Zeilen** näher erklärt werden.

2.3.1. Funktionale Anforderungen

Unter funktionalen Anforderungen versteht man im Software-RE im Allgemeinen Funktionen, welche von dem Zielsystem ausgeführt oder bereitgestellt werden sollen. Jedoch werden auch Systeminteraktionen, die dem Nutzer zur Verfügung stehen darunter zusammengefasst.

Beispiele für funktionale Anforderungen sind:

"Das System soll Usern eine Anmeldung über Facebook ermöglichen."

oder

"Das System muss es dem Administrator ermöglichen, bei der Erstellung eines Kurses die Raumnummer in dem dieser Kurs stattfinden soll einzugeben."

oder ein weiteres Beispiel

"Der User ist für den Kommunikationsaufbau zum Support zuständig."

2.3.2. Nicht-Funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben, in welcher Art und Qualität das System die geforderte Leistung erbringen soll. Dazu ist es notwendig in zwei weiteren Subkategorien zu unterscheiden, die Qualitätseigenschaften und die Randbedingungen.

Qualitätseigenschaften

Mit einer Qualitätseigenschaft wird die "Qualität", in welcher die Anforderung erbracht werden soll, beschrieben. Diese Anforderungen können sich sowohl auf die Performanz als auch auf die Zuverlässigkeit, Wartbarkeit oder Portabilität beziehen.

Auch hier sind weitere Kategorien zu unterscheiden

Auch hier ist wieder in weiteren Kategorien zu unterscheiden. die technischen Anforderungen, welche das System direkt beschreiben und ergonomische Anforderungen welche eher die Ergebnisse und Darstellungen des Systems beschreiben.

Ein Beispiel für eine technische Anforderung ist:

"Das Zielsystem muss mit Java entwickelt werden."

Eine ergonomische Anforderung hingegen könnte wie folgt lauten:

"Das System muss die Statistik in einen angemessenen Format ausgeben"

Eine weitere Kategorie ist Singular Eine weitere Kategorie sind Anforderungen an die Dienstqualität sind eine weitere Kategorie :

"Das System muss die Berechnung der Ergebnisse innerhalb von 20 Sekunden fertigstellen."

Randbedingungen

Randbedingungen sind Anforderungen, die entweder den Entwicklungsprozess selbst, rechtlich-vertragliche Anforderungen oder gesetzliche Richtlinien, die Einfluss auf die Umsetzung oder das Ergebnis des Projekts haben können.

Beispiele hierfür sind:

"Das Entwickler-Team muss mit dem Auftraggeber monatliche Reviews zur Besprechung weiterer Schritte durchführen."

oder

"Der Auftraggeber leistet für jeden abgenommenen Meilenstein ein Drittel der vertraglich vereinbarten Summe für die Entwicklung des Systems."

Qualitätskriterien

Damit Anforderungen richtig interpretiert und verarbeitet werden können, gibt es Kriterien, die bestmöglich bei der Erfassung dieser einzuhalten sind:

Vollständigkeit	Korrektheit
Realisierbarkeit	rechtliche Klarheit
Notwendigkeit	Bewertbarkeit
Konsistenz	Eindeutigkeit
Testbarkeit	Verständlichkeit
Aktualität	

Durch die Einhaltung dieser Kriterien kann vermieden werden, dass Anforderungen nicht dem eigentlichen Zweck entsprechend implementiert werden.

2.4. Kano-Modell

Der erste Teil des Satzes ist eine rhetorische Frage, die mit Komma getrennt wurde

Wie können diese Anforderungen nun Priorisiert werden, schließlich gibt es ja wichtige und weniger wichtige. Das 1978 von Dr. Noriaki Kano vorgestellte Kano-Modell

teilt Anforderungen in die entscheidenden Produktfaktoren ein. Hierbei wird unterschieden zwischen Basisfaktoren, Leistungsfaktoren und Begeisterungsfaktoren (Rupp & Schüpferling 2017). Diese geben an, wie die **Anforderung** sich auf die Zufriedenheit der Stakeholder auswirken. Abbildung 2.1 verdeutlicht das Prinzip.

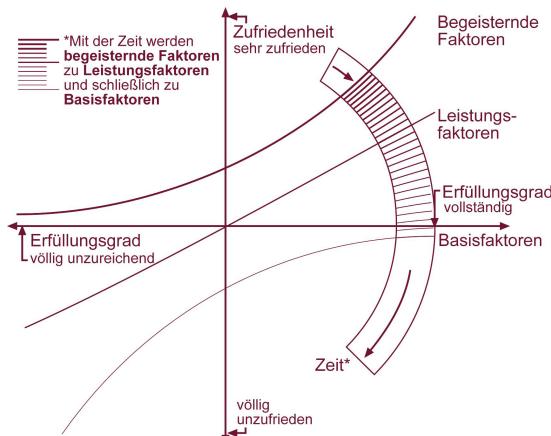


Abbildung 2.1.: Das Kano-Modell¹

Doch was sind Basis-, Leistungs- und Begeisterungsfaktoren? Um ein Verständnis aufzubauen werden diese im folgenden kurz erläutert. **Hier ist die rhetorische Frage richtig benutzt**

Basisfaktoren

Bin mir unsicher ob man eine Unzufriedenheit sagen kann

Basisfaktoren sind Features, welche von den Stakeholdern als selbstverständlich angesehen werden. Diese sollten in jedem Fall erfüllt werden, da sonst **eine** Unzufriedenheit der Kunden unabdingbar ist. Wenn ein solches Feature fehlt, kann dies sogar dazu führen, dass ganze Stakeholder-Gruppen wegfallen.

Leistungsfaktoren

Leistungsfaktoren hingegen beschreiben Features, die bewusst von den Stakeholdern verlangt werden, jedoch nicht zwingend notwendig sind. Fehlen zu viele dieser Faktoren hat auch das großen Einfluss auf die Zufriedenheit der Stakeholder. Wenn jedoch nur wenige dieser Faktoren fehlen, wird dies im Gegensatz dazu wie es bei den Leistungsfaktoren wäre meist geduldet und akzeptiert.

¹Eine Darstellung des Kano-Modell (Quelle: <http://www.sophist.de/fileadmin/SOPHIST/Blog/Kano-Modell.jpg> Sichtung: 13.06.2017)

Begeisterungsfaktoren

anders trennen

Unter Begeisterungsfaktoren versteht man Features, welche der Kunde noch nicht kennt, deren Wert dem Nutzer allerdings während der Nutzung deutlich wird. Solche Faktoren heben das Produkt vom Markt ab und tragen erheblich zum Erfolg des Produkts bei. Meist stellen diese Faktoren ein Alleinstellungsmerkmal des Produktes dar. Gute Begeisterungsfaktoren bleiben allerdings nicht solche, setzt sich ein Begeisterungsfaktor auf dem Markt erst einmal durch, wird dieser nach nicht all zu langer Zeit von Stakeholdern oft als Leistungsfaktor oder sogar als Basisfaktor angesehen.

Ein gutes Beispiel ist das Handy. Während früher mit einem Handy nur telefoniert werden konnte, hat sich irgendjemand ausgemalt wie es denn wäre, mit einem solchen Gerät auch Textnachrichten zu versenden. Dieses Feature war im ersten Moment ein Begeisterungsfaktor, denn die Nutzer kannten dieses Feature in dieser Form noch nicht, haben es aber im Laufe der Zeit lieben gelernt. Kurze Zeit später wurde diese Funktion allerdings bei dem Kauf eines Handys bewusst verlangt, wodurch es dann kein Begeisterungs- sondern ein Leistungsfaktor geworden ist. Heute würde es sogar als Basis-Faktor eingestuft werden.

2.5. Penalty-Reward-Faktoren Ansatz

Ein anderer Ansatz Anforderungen schon bei der Ermittlung zu bewerten ist der Penelty-Reward-Faktoren Ansatz. Hier geht es darum, die Dienstleistungsqualität einer Anforderung zu Messen. Dazu werden den Anforderungen Penelty- und Reward-Faktoren zugewieilt. Penelty-Faktoren sind solche, die das Gesamturteil über ein System verschlechtern, dagegen das Gesamturteil bei dem Wegfallen dieser nicht verbessert wird. Bei den Reward-Faktoren ist dies genau umgekehrt, hier wird durch das Vorhandensein die Zufriedenheit der Stakeholder verbessert, das Wegfallen hat allerdings keine negative Auswirkung auf das Zielprodukt. Wie das aussehen kann, ist auf der Abbildung 2.2 zu sehen.

Hier ist etwas mit dem Bild schieflgelaufen
Es ist mitten im Satz des nächsten Abschnitts

2.6. Anforderungen Ermitteln

Wie kommt man den jetzt an die Anforderungen der Stakeholder?

Vor der Ermittlung ist es wichtig festzulegen, welche Art von Anforderungen man in Erfahrung bringen will. Während man bei einem "fertigen" Projekt, möglicherweise

²Eine Darstellung der Penalty-Reward-Analysis (Quelle: <https://www.ifad.de/services/multivariate-verfahren/neuere-ansaetze/penalty-reward-analyse-pra/?lang=en> Sichtung: 13.06.2017)

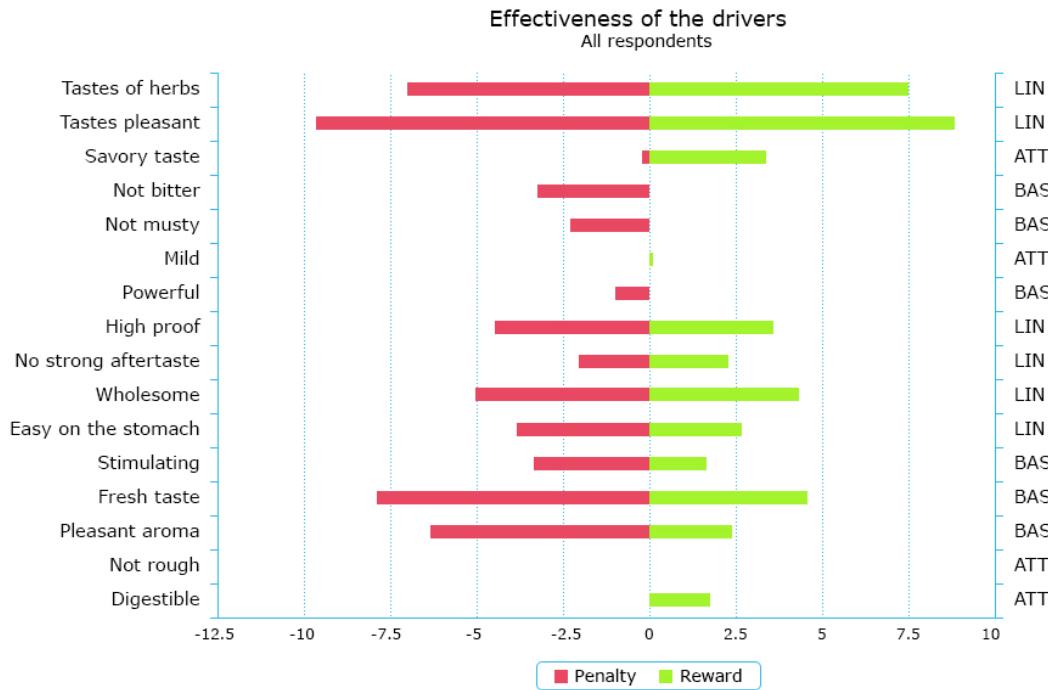


Abbildung 2.2.: Penalty-Reward-Analyse²

nur neue Features und Ideen in Erfahrung bringen möchte, ist es bei einem Projekt in der Entwicklungs- oder Planungsphase durchaus wichtiger vorerst die grundsätzlichen Wünsche und Anregungen der Stakeholder aufzunehmen. Dieser Schritt entscheidet, welche der vielen Methoden zur Ermittlung von Anforderungen eingesetzt wird. Welche Ermittlungs-Techniken es gibt wurde von Rupp & Schüpferling (2017) sehr gut und übersichtlich zusammengefasst, weshalb hier nur auf den Kern dieser Methoden eingegangen wird. Grundsätzlich unterscheiden Rupp & Schüpferling (2017) zwischen den Oberkategorien Kreativitäts-, Beobachtungs-, Befragungs- und Artefaktbasierte-Techniken. Was das ist, und wozu diese eingesetzt werden zeigt die folgende Übersicht.

2.6.1. Kreativitätstechniken

Eine Art zur Ermittlung von Anforderungen sind Kreativitätstechniken. Diese werden eingesetzt, um neue innovative Ideen zu entwickeln. Hier versucht man aus dem herkömmlichen Denken auszubrechen, um die Tür für neue Ideen zu öffnen. Dabei ist darauf zu achten, dass man sich vorher das richtige Umfeld schafft, um bessere Ergebnisse zu erzielen. Anschließend kann mit einer der folgenden Methoden der Kreativität freien Lauf gelassen werden.

Brainstorming

Die wohl bekannteste aller Kreativitätstechniken ist das Brainstorming. Dabei werden in einer Gruppe von fünf bis zehn Teilnehmern in circa 20 Minuten Ideen gesammelt und vorerst ohne weitere Kommentare notiert. Anschließend werden diese Ideen sorgfältig analysiert. Durch diese Methode ist es möglich mit der Inspiration Ideen anderer neue zu entwickeln. durch die Inspiration von Ideen anderer, eigene neue Ideen zu entwickeln

Diese Technik bietet den Vorteil, dass so viele Ideen in kurzer Zeit gefunden werden und die Teilnehmer ihre Ideen gegenseitig weiter entwickeln. Jedoch kann Brainstorming bei unterschiedlich dominanten Teilnehmern dazu führen, dass diese sich gegenseitig behindern. Auch ist diese Methode bei räumlich weit getrennten Stakeholdern problematisch.

Brainstorming paradox

Das Brainstorming paradox funktioniert fast genauso wie das normale Brainstorming, mit dem Unterschied, dass hier Ideen gesammelt werden, welche nicht erreicht werden sollen. Anschließend werden Maßnahmen entwickelt, welche diese Ideen vermeiden.

In manchen Situationen führt diese Methode schneller zu Ergebnissen. Auch werden so effektiv Risiken erkannt. Die Nachteile teilt sich diese Methode mit dem normalen Brainstorming.

Methode 6-3-5

Eine weitere Technik ist die Methode 6-3-5. Dabei handelt es sich um eine schriftliche Brainstorming-Methode. Hier bekommt jeder Teilnehmen einen Zettel, auf dem er innerhalb von circa fünf Minuten drei Ideen notiert. Anschließend wird dieser Zettel an den nächsten Teilnehmer weitergereicht, welcher, nachdem er die Ideen seines Kollegen durchgelesen hat, weitere drei Ideen hinzufügt. Dies wird solange wiederholt, bis jeder Teilnehmer einmal jeden Zettel hatte. Nach diesem Vorgang werden diese Ideen analysiert und zusammengefasst.

Da diese Technik schriftlich durchgeführt wird, lässt sie sich besonders bei einer komplizierteren Gruppendynamik einsetzen, da die in einer Diskussion aufkommenden Konflikte vermieden werden. Auch größere Distanzen zwischen den Stakeholdern können durch die Nutzung von Emails statt Zetteln bewältigt werden. Allerdings ist diese Methode gerade durch die schriftliche Form und die zeitliche Begrenzung nicht so Effektiv wie das normale Brainstorming.

Wechsel der Perspektive (6-Hut-Denken/Walt Disney-Methode)

In manchen Situationen bietet es sich an, einen Wechsel der Perspektive vorzunehmen. Um dies zu erreichen gibt es viele verschiedene Techniken. Eine ausführliche Variante ist das *6-Hut-Denken* von *Edward de Bono* mit sechs Perspektiven. Diese kann man sowohl alleine als auch in Gruppen einsetzen. Diese sechs Perspektiven lauten wie folgt:

1. Objektivität und Neutralität
2. Persönliches Empfinden und subjektive Meinung
3. Objektive, negative Argumente
4. Objektive, positive Eigenschaften
5. Neue Ideen
6. Prozess-Kontrolle

Die Walt Disney-Methode ist benannt nach Walt Disney, der angeblich (so Rupp & Schüpferling (2017)) für jede Sichtweise einen eigenen Raum hatte. Hierbei wird der Stakeholder aufgefordert, jede der folgenden Perspektiven in einem anderen Raum oder Zeitlich getrennt voneinander zu betrachten:

1. Träumer und Visionär
2. Realist
3. Kritiker

Das Problem wird bei dieser Methode aus all diesen Perspektiven betrachtet, wodurch man einen sehr guten Überblick über das Gesamtbild des Projekts bekommt.

Mit diesen Techniken können auch Stakeholder, welche in ihrer Denkweise sehr festgefahren sind veranlasst werden die Dinge einmal aus einer anderen Perspektive zu betrachten. Jedoch ist es für viele schwierig, sich in eine der genannten Rollen zu versetzen.

Analogietechniken (Bionik/Bisoziation)

Eine weitere Methode sind Analogietechniken. Bei einer Bionik werden als Denkmodell Analogie-Beispiele aus der Natur verwendet, um die dort gefundenen Lösungen auch auf das gegebene Problem anzuwenden. Um diesen Ansatz zu verstehen wird hier ein Beispiel von Rupp & Schüpferling (2017) zitiert.

"Denken Sie zum Beispiel an die Fusion zweier Firmen und vergleichen Sie es mit dem Vermischen von zwei Tierherden, Wie lange dauert es, bis

sich die Tiere der beiden Herden (die Mitarbeiter) vermischen? Die Leit-tiere werden in einen Konkurrenzkampf treten und eine neue Hierarchie erkämpfen. In einer Gefahrensituation, wenn zum Beispiel ein Raubtier die Herde angreift, werden sich die Tiere der beiden Tierherden als eine Herde verhalten, um ihre Chance zu verbessern, dem Raubtier zu entkommen. Diese Verhaltensmuster können in ähnlicher Form von den Mitarbeitern der beiden Firmen erwartet werden."

Bei der Bisoziation werden die Vorbilder im Gegensatz zu der Bionik nicht auf die Natur begrenzt.

Mit diesen Methoden ist es möglich, schwer vorstellbare Zusammenhänge oder komplexe Probleme durch den Kontextwechsel verständlich zu machen. Hierbei ist allerdings darauf zu achten, dass die verwendeten Vorbilder gut und passend ausgewählt werden. Dies nimmt unter Umständen viel Zeit in Anspruch.

Osborn-Checkliste

Die letzte der von Rupp & Schüpferling (2017) aufgeführten Kreativitätstechniken ist die Osborn-Checkliste. Diese besteht aus den Punkten:

- Kann man das Produkt auch anders verwenden?
- Gibt es etwas Ähnliches wie dieses Produkt, und was können wir davon nachahmen?
- Was lässt sich ändern? Kann man andere Funktionen einbauen?
- Wie kann man das Produkt erweitern, veredeln oder teurer machen?
- Wie kann man das Produkt vereinfachen oder auf Grundfunktionen reduzieren?
- Kann man das Produkt oder Teile davon ersetzen?
- Kann man das Produkt oder Teile davon umstellen, in der Reihenfolge verändern oder anders kombinieren?
- Kann man auch das Gegenteil mit dem Produkt machen?
- Kann man das Produkt oder die Idee mit etwas anderem kombinieren? Lässt es sich als Baustein für etwas anderes verwenden?
- Kann man es in seiner Materie verändern? Kann man es zusammendrücken, verflüssigen, durchlöchern oder anders transformieren?

Durch die Beantwortung dieser Fragen können gerade für ein bestehendes Produkt viele kreative Ideen gesammelt werden. Diese Methode für jede der Funktionalitäten eines großen Produkts anzuwenden kann allerdings zu aufwendig sein.

2.6.2. Beobachtungstechniken

Da aus verschiedenen Gründen nicht jeder Stakeholder in der Lage ist sein Know-how sprachlich auszudrücken oder die Zeit hat dies zu tun gibt es Beobachtungstechniken. Ihrer Mit diesen Techniken werden Stakeholder beobachtet um anhand seiner Arbeitsabläufe neue Anforderungen zu entwickeln. Diese Methoden haben den Vorteil, dass sehr detaillierte Anforderungen aufgenommen werden können, da diese in der Regel von einem Fachmann selbst notiert werden.

Feldbeobachtung

Eine Beobachtungstechnik ist die Feldbeobachtung. Hierbei werden die Tätigkeiten und Arbeitsschritte der Stakeholder mit ihrem zeitlichen Zusammenhang erfasst. Unklare Arbeitsschritte können durch das stellen von Fragen an den Stakeholder geklärt werden.

Gerade für die Beobachtung unterbewusster Tätigkeiten eignet sich diese Methode sehr gut. Zudem können so leichter Abweichungen in den Prozessen analysiert werden, da der Requirements-Engineer viele Personen und ihre Tätigkeiten beobachtet. Jedoch gibt es Arbeitsabläufe die schwer bis gar nicht beobachtbar sind, diese können mit dieser Methode nicht ermittelt werden.

Apprenticing

Bei dieser Methode beobachtet der Requirements-Engineer nicht den Stakeholder, sondern lernt die Tätigkeit unter der Anleitung von diesem selbst. So können die Anforderungen direkt aus der Perspektive des Stakeholders aufgenommen werden.

Im Gegensatz zur Feldbeobachtung fühlt sich der Stakeholder bei dieser Technik nicht beobachtet und unter Druck gesetzt. Auch bietet diese Methode psychologische Vorteile, weil der Requirements-Engineer selbst die Vor- und Nachteile eines Systems live aus der Sicht des Stakeholders erfährt.

2.6.3. Befragungsrechniken

Befragungsrechniken basieren darauf Stakeholder gezielt nach ihren Wünschen und Bedürfnissen zu befragen. Für die Ermittlung von nicht-funktionalen Anforderungen

ist diese Methode in den meisten Fällen nicht empfehlenswert, vielmehr wird diese dazu genutzt, Funktionale Anforderungen aufzunehmen. Durch die Befragung können je nach Umsetzung viele neue Sichtweisen auf ein Problem erworben werden.

Fragebogen

Eine dieser Techniken ist der klassische Fragebogen. Hierfür werden eine Reihe von Fragen von den Stakeholdern entweder online oder schriftlich beantwortet. Diese Antworten werden anschließend ausgewertet um daraus Anforderungen zu entwickeln.

Mit dieser Methode kann eine große Anzahl von Stakeholdern eingebunden werden. Allerdings ist anzumerken, dass gerade implizites Wissen der Stakeholder untergeht. Auch können bei der Beantwortung dieser Fragen Unklarheiten schlecht geklärt werden, was zu einer Fehlinterpretation führen kann.

Interview

Bei einem Interview werden dem Stakeholder die Fragen im Gegensatz zu einem Fragebogen in einem direkten Gespräch gestellt und protokolliert. Auch hier werden diese anschließend ausgewertet und daraus Anforderungen entwickelt.

Der Vorteil dieser Methode ist, dass in einem offenen Gespräch oft die Hemmung der Stakeholder gelindert werden. Des Weiteren kann der Verlauf des Gesprächs dynamisch angepasst werden. Mit dieser Technik kann besonders das Qualitätskriterium der Vollständigkeit leichter erfüllt werden als in manchen anderen Techniken. Jedoch sind solche Interviews sehr zeitaufwendig, was das Befragen vieler Stakeholder in den meisten Fällen erschwert.

Selbstaufschreibung

Eine weitere Form der Befragung ist die Selbstaufschreibung. Bei dieser Methode beschreibt der Stakeholder selbst Anforderungen, Änderungs- und Optimierungsvorschläge an das System. Die Qualität der Ergebnisse kann verbessert werden, indem die ausgewählten Stakeholder in die Grundlagen der Anforderungsanalyse eingewiesen werden. Durch die Selbstaufschreibung wird der Stakeholder nicht von dem Requirements-Engineer beeinflusst. Zudem muss er sein Wissen nicht erläutern sondern nutzt eine gleichbleibende Formulierung für seine Anforderungen.

On-Site-Customer

Anders als bei vielen anderen Methoden ist bei On-Site-Customer ein Interessenvertreter der Stakeholder während der Entwicklung ständig vor Ort. So können besonders in einzelnen Inkrementen die Anforderungen und deren Umsetzung optimiert werden.

Anforderungen werden so vor allem mündlich und sehr schnell ermittelt, dennoch ist es für viele schwierig einen geeigneten Mitarbeiter für solche Zwecke dauerhaft zu entbehren. Auch werden bei einer fehlenden Abstimmung zwischen dem On-Site-Customer und den anderen Stakeholdern nur die Interessen einer Person vertreten.

Rupp & Schüpferling (2017) beschreiben darüber hinaus noch Artefaktbasierte- und Unterstützende Techniken zur Ermittlung von Anforderungen. Diese beziehen sich größtenteils nicht auf die Stakeholder, sondern auf existierende Systeme oder andere Quellen. Diese Art der Anforderungsermittlung entspricht nicht dem eigentlichen Sinn dieser Arbeit, weshalb diese auch nicht weiter erläutert werden.

2.7. Dokumentation

Um Anforderungen nach der Ermittlung bestmöglich verarbeiten zu können, ist es notwendig diese in einem einheitlichen und fachlichen Stil zu dokumentieren. Auch hier findet man viele Techniken dies zu tun. Einen Überblick über die wichtigsten dieser Techniken geben die folgenden Abschnitte.

2.7.1. User-Storys

Eine Möglichkeit Anforderungen zu Dokumentieren sind User-Storys. Einzelne Systemfunktionalitäten werden in kleine Storys verpackt, welche aus der Sicht des Users formuliert sind. Diese dienen in wenigen Sätzen als Gedankenstütze zu der Anforderung. Eine User-Story könnte wie folgt lauten:

"Als Mitarbeiter möchte ich Urlaub vorab über ein elektronisches Formular beantragen können, damit ich eine offizielle Genehmigung erhalte und dann um dann zum Beispiel eine Ferienreise verbindlich buchen kann." zu können

User-Storys bilden zwar neben der eigentlichen Anforderung auch die Intention des Users ab, jedoch sind diese teilweise für die Entwickler schwieriger zu implementieren, da sie meist dafür zu grob formuliert sind.

2.7.2. Use-Cases

Ähnlich wie bei User-Storys bilden Use-Cases die Anforderung aus der Sicht des Users ab, diese sind aber in den meisten Fällen wesentlich grober formuliert. Auch werden hier oft sogar nur einzelne Aktionen beschrieben, einige Beispiele dazu könnten lauten:

"Geld Abheben" "Überweisung tätigen" oder "Kontostand überprüfen"

2.7.3. Technical Storys

Da Anforderungen nicht immer nur Funktionalitäten beschreiben, sondern manchmal auch technische Details, können diese gegebenenfalls nicht in User-Storys beschreiben werden. Um diese dennoch zu betrachten werden Technical Storys verwendet. Diese sind im Prinzip nichts anderes als User-Storys, welche aus technischer Sicht eine nicht-funktionale Anforderung beschreiben.

2.7.4. FunctionsMASTeR ist das so gewollt?

Eine weitere Alternative zur Dokumentation von Anforderungen beschreiben Die SOPHISTen (2013). Hierbei dient eine ganz einfach Schablone dazu, die wichtigsten Informationen einer Anforderung in einen Satz zu verpacken. Diese Schablone sieht wie folgt aus:

<System> <muss,sollte,wird> <Akteur>, fähig sein <Objekt> <Prozesswort>.

Diese Methode hat den Vorteil, dass so nicht nur funktionale sondern auch nicht-funktionale Anforderungen dokumentiert werden können. Auch ist diese Form der Beschreibung für die Entwickler gut Interpretierbar.

2.8. Validation

Die Validation ist ein wichtiger Schritt im Requirements Engineering. Nachdem die Anforderungen dokumentiert wurden muss geprüft werden, ob diese auch tatsächlich umsetzbar sind. Genauso kann es sein, dass Anforderungen über den Funktionsrahmen des Systems hinaus gehen und somit vorerst nicht mehr in Betracht gezogen werden. Dieser Schritt stellt also eine Überprüfung aller Anforderungen dar, wobei nicht valide Anforderungen aussortiert werden. Dies kann in Zusammenarbeit mit dem Entwickler-Team oder dem Product-Owner geschehen.

2.9. Auswertung

Die ermittelten Anforderungen müssen vor der eigentlichen Implementierungsarbeit ausgewertet werden. Das sorgt für eine detailliertere Priorisierung. So kann ein Plan erstellt werden, ob und in welcher Reihenfolge diese erfüllt werden. Die bekanntesten Methoden eine Auswertung durchzuführen werden im folgenden kurz erklärt.

2.9.1. Function-Point-Analyse

Mit der Funktion-Point-Analyse wird der Aufwand einzelner Anforderungen oder Anforderungsgruppen gemessen. Dieser Aufwand wird in Function-Points angegeben. Bei dieser Methode werden (so Dipl.-Ing. Klaus Lipinski (2013)) in drei Stufen die sogenannten unbewerteten oder auch unjustierten Funktion-Points (FP) bestimmt. Dazu werden in der ersten Stufe Parameter wie: Anzahl der Benutzer- Eingaben und Ausgaben, Anzahl der Benutzerabfragen, Anzahl der Datenbankspalten und die Anzahl der Schnittstellen zu externen Systemen in die Berechnung mit einbezogen. Die so errechneten FP werden anschließend in der zweiten und dritten Stufe durch Komplexitätsparameter und Konstanten des Entwicklungsprozesses gewichtet. Anschließend werden den Anforderungen Werte für bestimmte Charakteristika, wie die Wiederverwertbarkeit, zugewiesen.

Nach diesen drei Stufen spricht man von bewerteten oder auch justierten FP. Diese werden, um eine Aussage über den Implementierungsaufwand treffen zu können, anschließend mit Erfahrungswerten und der Produktivität von Software-Entwicklungsprozessen in Mitarbeiter-Monaten verglichen. verglichen und in Mitarbeiter-Monate umgerechnet

Diese Methode beruht zwar im Gegensatz zu anderen Methoden größtenteils nicht auf Schätzungen sondern auf Fakten und Erfahrungswerten, jedoch ist die Ermittlung des Aufwands mit dieser Technik sehr zeitaufwendig und Komplex. Auch zeigt laut der it-agile GmbH (2017) die Erfahrung, dass das vergleichende Schätzen, wie es bei Story-Points durchgeführt wird, zu deutlich schnelleren und besseren Ergebnissen führt.

2.9.2. Story-Points

Dopplung

Eine Aufwandsschätzung mit Story-Points ist wesentlich schneller als der langwierige Prozess der Function-Point-Analyse und stellt deshalb vor allem in agilen Projekten eine gern gesehene Alternative dar. Hierzu werden den zuvor erstellten User-Storys Story-Points zugewiesen, welche eine abstrakte Komplexität der einzelnen Storys angibt. Dabei wird eine Skala festgelegt, in welcher die Story-Poits vergeben werden.

Preuss (2013) empfiehlt als Skala eine abgeänderte Form der Fibonacci-Folge zu verwenden, welche aus den Elementen 1, 2, 3, 5, 8, 13, 20, 40 und 100 besteht. User-Storys mit vielen Story-Points bedeuten, dass diese Story in kleinere Storys aufgeteilt werden **muss** müssen um realistisch abgeschätzt werden zu können. Man kann allerdings auch eine einfache Skala von beispielsweise 1-10 nutzen.

Nach der Festlegung dieser Skala wird eine Story ausgesucht, die einen guten Mittelwert an Komplexität mit sich bringt. An dieser Story wird sich bei der Vergabe weiterer Story-Points orientiert.

Es gibt noch weiteren Methoden zur Aufwandsschätzung einer Anforderung wie beispielsweise das CoCoMo (mehr dazu auf betriebswirtschaft-lernen.net (2017)), die COSMIC-Methode der cosmic-sizing.org (2017), Lines of Code, Ad-hoc-Schätzmethoden oder umfangbasierte Schätzmethoden. Eine gute Übersicht über diese Methoden bietet das WinfWiki (2017).

An dieser Stelle kam mir als erstes in den Sinn :

Endlich bin ich in der Umsetzung und nicht mehr in den Erklärungen.

Das kann natürlich daran liegen, dass ich nichts mit dem Thema zu tun habe, aber für mich als Unbeteiligten, war das eine ganze Menge Erklärung bevor tatsächlich etwas über deine Arbeit kommt.

Habe aber leider dazu auch keinen Vorschlag wie man das interessanter gestalten könnte.

3. Umsetzung

- Das Anforderungsmanagement teilt sich in mehrere Abschnitte ein, jeder dieser Abschnitte stellt einen wichtigen Prozess für den Umgang mit Anforderungen dar. Der erste Schritt ist die Ermittlung, wobei die Anforderungen der Stakeholder ermittelt werden um sie daraufhin zu Dokumentieren. Ist die Dokumentation abgeschlossen, folgt die Auswertung der dokumentierten Anforderungen. Bei einem Agilen Projekt, wie es das LWM darstellt, wird dieser Vorgang in Iterationen bis zu einem beliebigen Detailgrad immer wieder verfeinert.

Kann den Zeilenumbruch nicht markieren
aber da fehlt ein Komma :D

3.1. Ermittlung

Aus der Problemstellung ist zu entnehmen, dass nicht das Ermitteln neuer ausgefallener Ideen und Feature^s das Ziel ist, sondern das in Erfahrung bringen der Wünsche und Voraussetzungen der Stakeholder. In anderen Worten wird das Ziel verfolgt Basis- und Leistungsfaktoren zu ermitteln. Aus diesem Grund wurde bewusst Abstand von den in Kapitel 2.6.1 beschriebenen Kreativitätstechniken genommen. Auch wurde sich gegen die in Kapitel 2.6.3 aufgeführten Fragebogen-Methode entschieden, da die befragten Stakeholder dazu nicht genug Kenntnisse über das Produkt besaßen.

Unter anderem aus diesem Grund etablierte sich die Interview-Technik. Bei dieser Technik geht es darum, ein Interview über vorbereitete und vorformulierte Fragen mit den gewählten Stakeholdern zu führen. So ist es möglich, während dem Gespräch auftretende Fragen direkt zu klären und somit auch implizite Anforderungen aufzudecken. Zudem öffnet diese Methode die Tür für neue Anforderungen oder Problemstellungen, welche bislang nicht in Betracht gezogen wurden. Ein weiterer Vorteil dieser Technik ist, dass die Hemmungen der Stakeholder durch ein offenes Gespräch gelindert werden.

Zu Beginn der Ermittlung wurde ein zweiseitiger Fragebogen (siehe Anhang A) als Leitfaden für die Interviews erstellt, welcher in kurzen Stichpunkten die wesentlichen Problemstellungen und Platz für die darauf bezogenen Notizen beinhaltet. Die dort aufgeführten Probleme beruhen auf Erfahrungen mit dem bestehenden Tool und wurden durch Brainstorming ergänzt. Mit diesem Formular konnte nicht nur der Umgang

Hast du bereits erklärt

Zeit mit den bekannten Problemen des Tools, sondern auch solche, die aus Ausnahmesituationen resultieren zur Ansprache gebracht werden.

Im Anschluss daran wurde mit Hilfe des Entwickler-Teams, sowohl Frontend als auch Backend, eine Eingrenzung der Stakeholder vorgenommen. Dieser ausgewählte Kreis wurde per E-Mail kontaktiert und um einen Gesprächstermin gebeten, sofern Interesse an der Nutzung eines Tools zu Verwaltung der Praktika besteht.

**wunderbar
formulierter
Abschnitt**

Bei dem Ablauf der Interviews ist grundsätzlich zwischen den aktuellen Nutzern und den potentiell zukünftigen Nutzern zu unterscheiden. Zwar wurde das Ziel verfolgt, losgelöst von dem aktuellen Tool über ein "Wunschsystem" zu sprechen, jedoch war dies einerseits bei den bestehenden Nutzern kaum möglich, da diese sich instinktiv immer wieder an dem derzeitigen Tool orientieren und andererseits konnte dieses bei den Stakeholdern, für die dieses System zu diesem Zeitpunkt noch unbekannt war, als perfekte Beschreibung der Thematik genutzt werden.

bekannt

**Komma am
Ende der Zeile**

Ein Interview begann sofern erforderlich mit der Beschreibung des bestehenden Tools, um eine Ausgangsgrundlage für das Gespräch zu schaffen. Nach der Darlegung der Problemstellung war jedem Gesprächspartner der Hintergrund dieser Anforderungs-ermittlung erkenntlich. Mit der Aufforderung zunächst vollkommen losgelöst von den vorgefertigten Fragen den Ablauf des Praktikums und dessen Bewertung zu erklären, konnten schon zu diesem Zeitpunkt eine Vielzahl dieser Fragen beantwortet werden. Diese Informationen wurden durch Fragen mit einem "Was wäre wenn"-Stil ergänzt, wobei zu beachten war, dass die Gesprächspartner nicht in eine vorgegebene Denkweise geleitet werden. So wurden auch die kleinsten Probleme, die bei der Nutzung eines solchen Tools auftreten können, aufgedeckt und dokumentiert.

**Man war bereits
an dieser Stelle
also wird man das wissen.
Falls du es drin lassen
willst, ersetze „werden“
durch „wurden“**

Die Anforderungen der Nutzer wurden während des Gesprächs bestmöglich in die Kano-Faktoren eingeteilt, welche in Kapitel 2.4 beschrieben werden. Dadurch konnte eine grobe Priorisierung nach Basisfaktoren, Leistungsfaktoren und Begeisterungsfaktoren schon vor der eigentlichen Auswertung vorgenommen werden.

ermöglicht

Durch die Interviews konnten über 150 Anforderungen ermittelt werden, wovon einige durch das gegenwärtige Tool schon abgedeckt werden und andere, welche für eine ganz neue Perspektive auf das System und dessen Möglichkeiten eröffnet.

**Bin mir aber nicht sicher weil ich
den Satz nicht 100% versteh**

3.2. Zielgruppe

Als Stakeholder ist grundsätzlich jeder zu betrachten, der aktiv an dem Tool oder dessen Nutzung beteiligt ist oder durch diese beeinflusst wird. Durch diese Definition befinden sich in der Gruppe der Stakeholder nicht nur das Entwickler-Team, sondern auch Studenten, Administratoren, Modulverantwortliche, Mitarbeiter, studentische Hilfskräfte (SHK) und wissenschaftliche Hilfskräfte (WHK).

heißt das nicht wissenschaftliche Mitarbeiter oder ist das egal?

Die Anforderungsermittlung über Studenten wurde nicht in Betracht gezogen, da dies nicht in den zeitlichen Rahmen dieser Arbeit gepasst hätte. Zudem sind Studenten zwar direkte Nutzer des Systems, jedoch Arbeiten sie im Gegensatz zu den anderen Stakeholdern verhältnismäßig wenig damit.

Die Anforderungen des Entwickler-Teams wurden im Laufe der Anforderungsauswertung und Validation aufgenommen.

Von den Verbleibenden Stakeholdern erwies es sich als Sinnvoll, die Modulverantwortlichen als Interessenvertreter anzusehen.

Einheitlichkeit

Durch das Modulhandbuch des Campus konnte eine Liste der existierenden Module erstellt werden. Die Email-Adressen der Modulverantwortlichen konnte von der Webseite der TH-Köln bezogen werden.

Mit der Vorgabe den Kreis der Nutzer auf den Bereich des Instituts für Informatik zu begrenzen beinhaltet diese Liste 61 Module.

Durch ein Gespräch mit dem Entwickler-Team konnte diese Liste anschließend auf circa 20 Module begrenzt werden, indem Module die kein Praktikum veranstalten aussortiert wurden.

Nach der Kontaktierung der jeweiligen Modulverantwortlichen bestätigten 14 von diesen ein Interesse an diesem Projekt und teilten einen Termin für ein Interview mit.

erklärten sich zu einem Interview bereit

3.3. Dokumentation

3.3.1. Formulierung

Um die auf den Interview-Bögen notierten Anforderungen mit allen nötigen Details zu Dokumentieren, wurde auf eine abgewandelte Form der in Kapitel 2.7.4 erläuterten

Schablone zurückgegriffen.

In einer etwas angepassten Form sieht der Satzbau wie folgt aus:

<Zielsystem> <Priorität> einem <Stakeholder> die Möglichkeit bieten, <Funktion>.

Erklärung:

<Zielsystem>

Stellt das referenzierte System dar. Hierbei steht "*Das bestehende System*" für das aktuelle LWM und "*Das zu entwickelnde System*" für ein neues System . Die Unterscheidungen dieser Systeme wird in Kapitel 3.4 genauer erläutert.

<Priorität>

Hier wird mit Hilfe des Kano-Modell von Prof. Noriaki Kano priorisiert. Für einen Basis-Faktor wird mit Anlehnung an Rupp & die SOPHISTen (2014) das Schlüsselwort "*muss*", für ein Leistungs-Faktor "*soll*" und für einen Begeisterungsfaktor "*wird*" eingesetzt. ist „wird“ wirklich ein gutes Wort für ein Begeisterungsfaktor?
Vielleicht habe ich auch Begeisterungsfaktoren falsch verstanden
aber das klingt als wäre es schon zu 100% festgelegt

<Stakeholder>

Hier wird die Liste von Stakeholdern, welche von dieser Anforderung Gebrauch machen würden eingesetzt.

<Funktion>

Die Funktion beschreibt den eigentlichen Sinn dieser Anforderung.

Ein Beispiel aus dieser Arbeit sieht wie folgt aus:

"Das bestehende System soll einem Modulverantwortlichen und einem Mitarbeiter die Möglichkeit geben, eine gelbe Karte für eine unbefriedigende Abgabe zu erteilen."

oder:

"Das zu entwickelnde System wird einem Modulverantwortlichen, einem Mitarbeiter, einer Studentischen Hilfskraft und einem Studenten die Möglichkeit geben, Termine in einer gegebenen Frist zu stornieren."

3.4. Auswertung

Für die Arbeit mit den Anforderungen, ist es notwendig, diese auszuwerten. Das LWM ist ein agiles Projekt weshalb es Sinnvoll war, auch die Anforderungsauswertung diesem Stil entsprechend durchzuführen. Hierzu wurden die Anforderungen nach der Validation in einer Art Epic zusammengefasst. Anschließend wurden diese Gruppen priorisiert. Hier setzt ein neuer Iterationsschritt ein, indem eine Gruppe mit der höchsten Priorität selektiert wiederum in Untergruppen eingeteilt wurde. Diese Untergruppen konnten anschließend wieder Priorisiert werden. Dieser Vorgang kann je nach Belieben wiederholt werden. Ist der gewünschte Detailgrad erreicht, fällt es leicht die Komplexität der einzelnen Anforderungen zu schätzen.

3.4.1. Validation

Der erste Schritt in der Auswertung war die Validation. Die Anforderungen wurden für ein bestehendes System aufgenommen, weshalb es vorkommen kann, dass Anforderungen schon erfüllt wurden. Um die spätere Verarbeitung zu vereinfachen wurde deshalb im Vorfeld eine Validation der Anforderungen vorgenommen. Hierbei wurden diese in drei Kategorien aufgeteilt. Es wurde unterschieden zwischen "*Erledigt*", "*Valid*" und "*Verworfen*". Die Anforderungen aus den Kategorien "*Erledigt*" und "*Verworfen*" werden zwar nicht gelöscht, da diese auch während der Implementierung von anderen Anforderungen von Relevanz sein können, jedoch werden diese bei der Priorisierung außen vor gelassen. Auch gibt es für diese Anforderungen im ersten Moment keine Aufwandsschätzung, da diese derzeit keinen Aufwand bedeuten.

3.4.2. Gruppierung

Eine unsortierte Liste von Anforderungen kann nicht so einfach ausgewertet werden, weshalb ein Verfahren gefunden werden muss, dieser Anforderungen in kleinere Gruppen zu sortieren. Mit diesen Gruppen fällt nicht nur die Übersicht leichter, sondern auch die spätere Priorisierung und Aufwandsschätzung.

Im Laufe der Anforderungsermittlung wurde deutlich, dass einige Anforderungen die eigentliche Grenze des aktuellen LWM überschreiten. So wurde beispielsweise ein Forum für das besprechen von Problemstellungen gefordert, was jedoch nicht direkt was mit der Praktikumsverwaltung selbst zu tun hat. Da diese Ideen jedoch nicht verworfen werden sollen, da sie einen Mehrwert für den Praktikumsverlauf darstellen, wurde mit dem Entwickler-Team die Entscheidung getroffen, die Anforderungen aufzuteilen. Die erste Gruppierung erfolgte somit durch die Unterscheidung, ob eine Anforderung einem

Was sind Epics?

Eine Info, dass die später erkärt werden, wäre nett

nicht betrachtet

neuen System, oder dem bestehenden System zuzuordnen ist.

Des Weiteren wurde sich für eine Gruppierung nach Aufgabengebiet entschieden. Dies ist ein sehr grobes Verfahren, was allerdings im Anbetracht der Menge von Anforderungen als angemessen empfunden wurde. Ganz nach dem Prinzip `divide and conquer` wurden erst wenige große Gruppen erstellt, die anschließend durch weitere Iterationen verfeinert werden.

Zeit

Auch wenn dem das Prinzip von Epics auch in diesem Anwendungsfall verfolgt wurde, viel die Böser 1-Klässler Fehler xD Entscheidung gegen die Beschreibung in Form einer User-Story, da dies auf diesem Abstraktionslevel nicht angebracht war. Ab einem gewissen Detailgrad ist dies jedoch durchaus sinnvoll.

Diese Gruppierung erfolgte durch die Aufteilung in Aufgabengebiete an einem Whiteboard.



Abbildung 3.1.: Gruppeneinteilung zweiter Ebene¹

So konnten die circa 150 Anforderungen in 24 Gruppen eingeteilt werden.

¹Ein Bild von der Gruppeneinteilung auf zweiter Ebene an einem Whiteboard

Platzhalter ersetzen

Nach der ersten Priorisierung, auf welche in Kapitel ?? näher eingegangen wird, wurde zunächst die Gruppe mit der höchsten Priorität ausgewählt, um sie in kleinere Einheiten zu zerlegen. Dies erfolgte in einem Meeting mit dem Entwickler-Team gemeinsam. Hierzu wurden die Anforderungen der Gruppe "Staffelplan" auf einem Whiteboard in eine Ablaufreihenfolge gebracht.

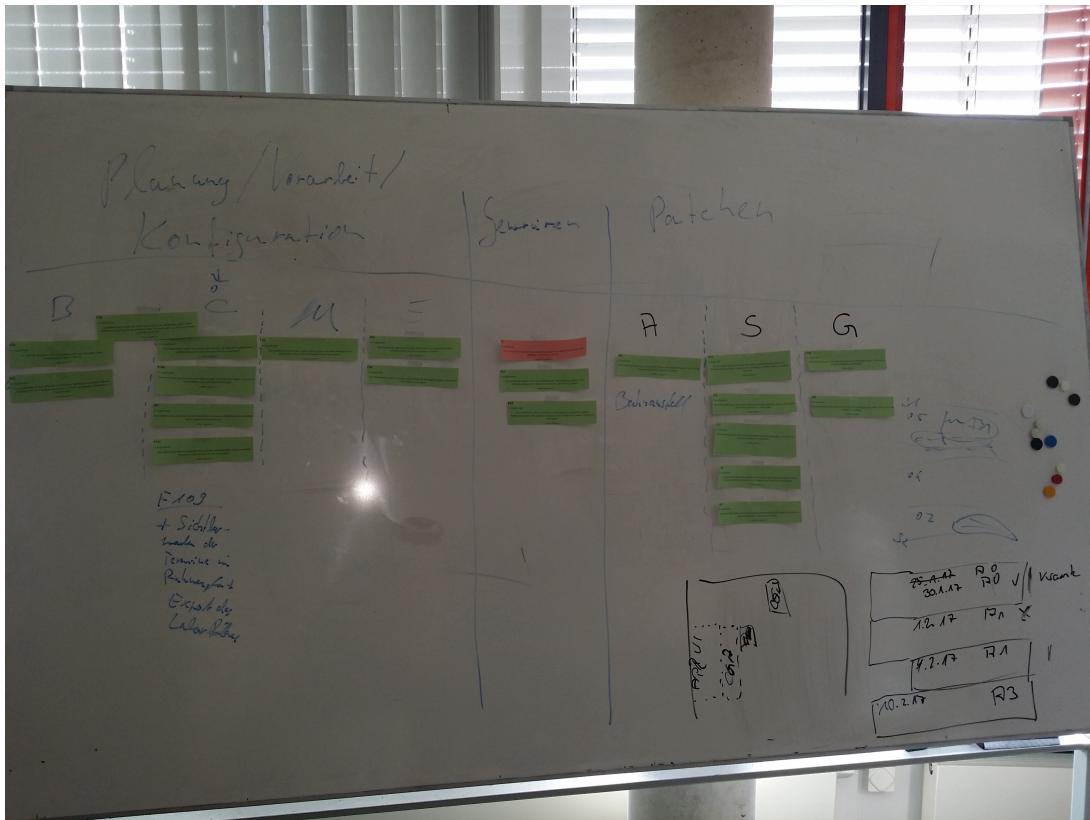


Abbildung 3.2.: Gruppeneinteilung dritter & vierter Ebene²

Es wurde zwischen drei Ablaufphasen unterschieden, diese wurden anschließend in weitere Teilgruppen aufgeteilt.

1. Planung/Vorarbeit/Konfiguration
 - a) B => Blockpraktika
 - b) C => Aktuelles Schema
 - c) M => Manuelle Eingabe
 - d) F => Flexible Eingaben

2. Generieren

²Ein Bild von der Gruppeneinteilung auf dritter & vierter Ebene an einem Whiteboard

3. Patchen

- a) A => Betrifft alle
- b) M => Betrifft Studenten
- c) G => Betrifft Gruppen

Nach dieser Gruppierung sind die Anforderungen so sortiert, dass man die einzelnen Gruppen als Arbeitsschritte abarbeiten könnte. Auch wurden im Laufe dieses Verfahrens neue Anforderungen aufgedeckt, die entweder aus der Sicht der Entwickler notwendig **sind** oder andere Anforderungen detaillieren.

Dieses Vorgehen wiederholt sich für alle Gruppen auf zweiter Ebene, sobald die Entwicklung des LWMs so fortgeschritten ist, dass die nächsten Anforderungen implementiert werden können.

3.4.3. Priorisierung

Die Priorisierung der Anforderungen ist notwendig, um erkennen zu können, wie wichtig eine Anforderung für die Zufriedenheit **aller** Stakeholder ist. Zudem wird diese Priorität nach der Aufwandsschätzung dazu genutzt, über die Umsetzungsreihenfolge der Anforderungen zu entscheiden.

Eine erste Priorisierung wurde nach der Gruppierung auf zweiter Ebene durchgeführt und wurde in Prozent angegeben. Dabei ist diese Zahl wie folgt zu interpretieren:

100% → **Alle** Stakeholder haben diese Anforderung als **Basisfaktor** bestimmt

0% → **Kein** Stakeholder hat diese Anforderung erhoben

Für die Berechnung dieses Wertes wurde den einzelnen Kano-Faktoren und der Stimme eines Stakeholders (Vote) ein Wert und ein Variablen-Name zugewiesen:

bap → Basis-Faktor → 9

lp → Leistungs-Faktor → 6

bep → Begeisterungs-Faktor → 3

vp → Vote → 1

Des **weiteren** werden folgende Parameter benötigt:

$v \rightarrow$ die Votes der Anforderung

$m \rightarrow$ die Maximalen Votes die für eine Anforderung abgegeben wurden

$p \rightarrow$ die Priorität in %

Die Formel sieht dann wie folgt aus:

$$p = \frac{v * vp + < bap | lp | bep >}{m * vp + bap}$$

Zeit ?

Nachdem die Aufwandsschätzung abgeschlossen ist, kann mit dem dort geschätzten Wert und der Priorität deutlich gemacht werden wie sich die Anforderung im Laufe der Entwicklung zu interpretieren ist.

Um die weitere Berechnung verstehen zu können, ist es wichtig zu wissen, dass die Komplexität einer Anforderung in Story-Points geschätzt wurde. Hierbei bedeuten 10 Story-Points, dass die Anforderung einen hohen Implementierungsaufwand mit sich bringt und 1 Story-Point, dass die Implementierung dieser am wenigsten komplex ist.

Das Prinzip dieser Interpretation wird mit der Abbildung 3.3 deutlich.

Auf Abbildung 3.3 ist zu erkennen,³ dass die Anforderung $F3$ zwar einen hohen Implementierungsaufwand bedeutet, diese allerdings eine Stakeholder-Priorität von 90% aufweist dagegen hat $F5$ nur eine Priorität von 20% jedoch ist diese wesentlich einfacher erfüllbar. Diese beiden Anforderungen sind haben also in erster Linie keine hohe Priorität bei der Entwicklung. $F4$ weist von den abgebildeten Beispielen die schlechteste Bewertung auf. Diese ist wenn überhaupt erst ganz am Ende zu betrachten, wenn alle anderen Anforderungen erfüllt wurden. $F1$ hingegen ist sofort umzusetzen, da diese nicht nur von den meisten Stakeholdern als wichtig angesehen wurde, sondern auch relativ einfach zu Implementieren ist.

Wieder ein Problem mit dem Bild

Durch die Abarbeitung in dieser Reihenfolge können die wichtigsten Wünsche der Stakeholder am schnellsten erfüllt werden, deshalb wurde für die Anforderungen bei diesem Projekt auch ein solcher Wert berechnet. Wie die Berechnung als Formel aussieht, wird nachfolgend gezeigt. Doch vorher müssen muss auch für diese Formel die Definitionen einiger Variablen geklärt werden:

$sp \rightarrow$ die Story-Points die der Anforderung bei der Aufwandsschätzung zugewiesen wurde

³Erklärung der Anforderungsauswertung anhand eines Beispiels

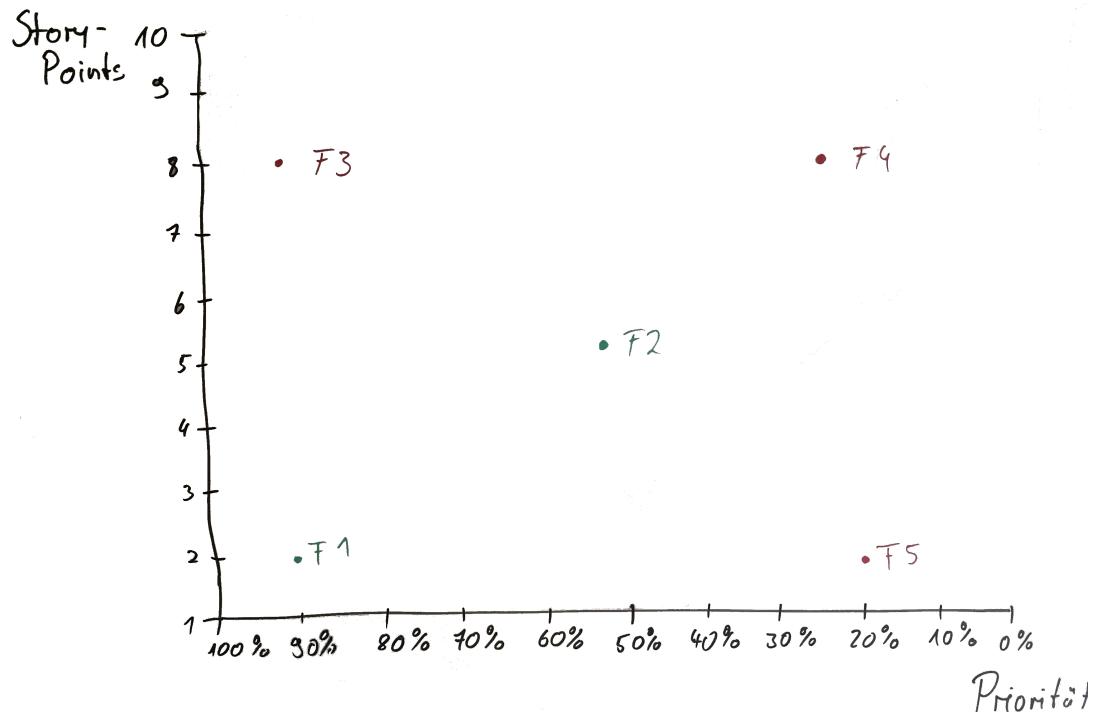


Abbildung 3.3.: Erklärung der Anforderungsauswertung³

$p \rightarrow$ die Priorität der Anforderung in %

$r \rightarrow$ das Ergebnis der Berechnungen. (Da für diese Variable kein passender Name gefunden werden konnte, wird diese ab jetzt auch weiterhin Ergebnis genannt)

Nun ist es es ohne Probleme möglich die Formel zu verstehen:

$$r = 100 - \sqrt{(sp - 1)^2 + (100 - p)^2} \quad \text{Gerngeschehen :P}$$

Der Kern dieser Formel ist der Satz des Pythagoras:

$$c^2 = a^2 + b^2 \rightarrow c = \sqrt{a^2 + b^2}$$

Hierbei ist c unser Ergebnis r , a sind die Story-Points -1 .

Doch warum Story-Points -1 ?

Wie in Kapitel 3.4.1 (Validation) beschrieben wurde, konnte im Vorfeld entschieden werden, ob eine Anforderung als "Erledigt", "Valide" oder "Verworfen" anzusehen ist.

Bei der Weiteren Auswertung sind erledigte oder verworfene Anforderungen wohl kaum noch von Interesse, deshalb werden diese in der Berechnung **aufßen vor gelassen**. Das hat zur Folge, dass eine Anforderung, welche in diese Berechnung mit einfließt einen **Mindestaufwand von 1 Story-Point haben muss**. Es gibt schließlich keine Anforderung die noch nicht implementiert wurde, es jedoch keinen Aufwand mit sich zieht, dies zu tun. Auch eine einzige Zeile Code hat schon einen Story-Point zu Folge.

Mit der Rechnung "Story-Points – 1" wurde also eine Normalisierung durchgeführt um mit einem Story-Point und 100% Priorität auch ein Ergebnis von 100% zu erzielen, denn das ist das beste, was eine Anforderung haben kann.

Schließlich ist noch das b zu betrachten, dieses wird durch $100 - p$ ersetzt, wobei p die Priorität der Anforderung ist, welche ohne Berücksichtigung der Komplexität errechnet wurde. Warum diese Priorität von 100 abgezogen wird, erkennt man in Abbildung 3.4.

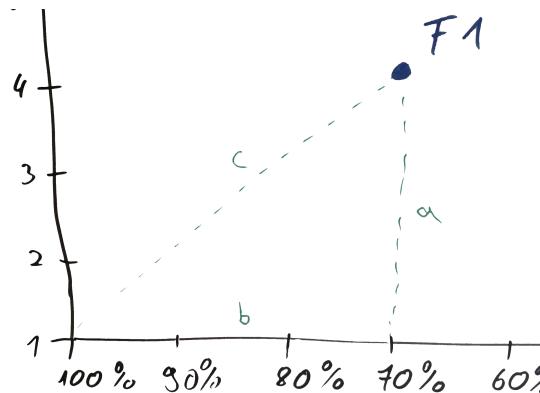


Abbildung 3.4.: Erklärung zur Rechnung $100 - p$ ⁴

Die y-Achse stellt die Skala der Priorität dar. Da Laut der durchgeföhrten Rechnung c , also die Länge des Ortsvektors^s die Bewertung der Anforderung darstellt, ist zu beachten, dass die y-Achse am Koordinatenursprung mit 100% beginnt. Somit stellt eine Priorität von 70% für b nicht den Wert 70 dar, sondern $100 - 70$ also 30.

Nachdem diese Berechnungen und somit die Bewertung der Anforderungen durchgeführt wurde, ist es dem Entwickler-Team möglich die Anforderungen mit der richtigen Priorität zu implementieren.

⁴Eine Bild zur Erklärung für die Rechnung $100 - p$ bei der Priorisierung

3.4.4. Aufwandsschätzung

Bei der Aufwandsschätzung der ermittelten Anforderungen wurde sich aufgrund der agilen Entwicklung des LWMs bewusst gegen die Anwendung der Funktion-Point-Analyse (FPA) oder anderen klassischen Schätzmethoden entschieden. Statt dessen wurde an dieser Stelle von Story-Points (SP) Gebrauch gemacht. Den Grund dafür liegt neben dem hohen Aufwand einer Function-Point-Analyse in Erfahrungsberichten, wie es der die ebenfalls dazu raten.

WTF

„Die Erfahrung (nicht nur der agilen Projektwelt) hat aber gezeigt, dass das vergleichende Schätzen in abstrakten Schätzmaßen zu deutlich schnelleren und besseren Ergebnissen führt.“ (it-agile GmbH 2017)

ein Intervall

Um eine Schätzung mit Story-Points durchführen zu können, werden die Anforderungen in User-Story (US) aufgeteilt. Diese Storys bekommen anschließend Story-Points zugewiesen. In diesem Fall wurde eine Range von 1 - 10 festgelegt, wobei 1 für eine niedrige und 10 für eine sehr hohe Komplexität steht. Hierbei ist zu Unterscheiden, dass im Gegensatz zu klassischen Schätzverfahren nicht der Aufwand sondern die Komplexität geschätzt wird. Einige Quellen bevorzugen es, als Story-Points ein Element aus einer angepassten Fibonacci-Folge zu nehmen. Diese besteht aus den Elementen 1, 2, 3, 5, 8, 13, 20, 40 und 100. So wird das Ziel verfolgt, zu verdeutlichen, dass die Schätzung einer Anforderung mit zunehmender Komplexität immer ungenauer wird.

Dopplung

Auch schon oben erklärt,
aber hier ist das sinnvoll.
Das würde ich durch ein
„Ganz nach dem Prinzip“
erweitern, um zu zeigen,
dass du weißt, dass du
das schon erklärt hast.

Damit nun die Story-Points ordnungsgemäß zugewiesen werden konnten, wurde eine Anforderung rausgesucht, welche einen guten Mittelwert an Komplexität bedeutet. Dieser Anforderung wurden die ersten Punkte zugewiesen. Relativ zu dieser Schätzung wurde anschließend auch den übrigen Anforderungen ein Wert gegeben. Dieser Prozess wurde in einem Meeting mit dem Entwickler-Team vor Ort im Zuge der detaillierteren Gruppierung durchgeführt. Dazu haben sich die Entwickler von Frontend und Backend zu der Komplexität der jeweiligen Implementierung abgesprochen und sich auf eine Anzahl an Story-Points geeinigt. Dieses Verfahren kann man genauso wie die Gruppierung auch in weiteren Iterationen weiter verfeinern. Dazu werden diese US wieder in kleinere aufgesplittet welche wiederum SP zugewiesen bekommen. Dieser Vorgang ist bei sehr groben US zu empfehlen.

Die somit Verteilten SP konnten anschließend zur weiteren Priorisierung genutzt werden.

4. Visualisierung der Ergebnisse

Die ermittelten Anforderungen sollen auch nach der Fertigstellung dieses Projekts noch zur Orientierung bei der Entwicklung des LWM dienen. Deshalb ist eine ordnungsgemäße und langfristige Festhaltung der Anforderungen unumgänglich. Für die weitere Auswertung und die Priorisierung ist es zudem notwendig, Berechnungen und Sortierungen mit den Anforderungen durchzuführen.

4.1. Excel

Numm.	Prinzip	Prioritätstyp zur Formalisierung	Beschreibung (Das zu entwickelnde System muss einen - die Möglichkeit bietet, ohne Praktiken festzulegen)	Administrator	Mitarbeiter	Ums.	Stadt	Umw. Tisch	System verkauf	Angeford. der	Abgeleit.	Gesamt Priorität mit allen Praktiken (Praktiken, Anzahl der)	Priorität eines User	Priorität eines	Vorliegt	Priorität eines	Ausforderung
														Praktiken	Anzahl der	User	Urg.
1	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	47%	40%	80%	1,00	3,00	2,00
2	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	51%	51%	100%	1,00	3,00	3,00
3	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	51%	51%	100%	1,00	3,00	4,00
4	nein		Das System muss die Möglichkeit bieten, die Anzahl für diese Tische zu ändern, seck was diese keine Praktiken hat.	✓	✗	✗	✗	✗	✗	2	0	45%	40%	100%	1,00	3,00	2,00
5	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	100%	1,00	3,00	3,00
6	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	100%	1,00	3,00	3,00
7	nein		Das System muss die Möglichkeit bieten, Aufgabe zu erstellen.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	100%	1,00	3,00	2,00
8	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	100%	1,00	3,00	3,00
9	nein		Das System muss die Möglichkeit bieten, Aufgabe aus Turms zu erstellen.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	100%	1,00	3,00	2,00
10	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	100%	1,00	3,00	3,00
11	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	45%	50%	100%	1,00	3,00	3,00
12	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	52%	51%	100%	1,00	3,00	4,00
13	nein		Das System muss die Möglichkeit bieten, ohne Praktiken festzulegen.	✓	✗	✗	✗	✗	✗	0	0	51%	50%	100%	1,00	3,00	3,00
14	zoll		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	67%	1,00	3,00	5,00
15	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	67%	1,00	3,00	5,00
16	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	67%	1,00	3,00	10,00
17	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	51%	50%	67%	1,00	3,00	10,00
18	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	50%	50%	67%	1,00	3,00	10,00
19	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,9%	10%	100%	1,00	3,00	9,00
20	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,9%	10%	100%	1,00	3,00	9,00
21	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,9%	10%	100%	1,00	3,00	10,00
22	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	100%	1,00	3,00	2,00
23	nein		Das System soll die Möglichkeit bieten, ohne Struktur über das Letzte Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	0	0	45%	40%	100%	1,00	3,00	3,00
24	nein		Das System soll die Möglichkeit bieten, Praktikengruppe Störz einzufügen.	✓	✗	✗	✗	✗	✗	0	0	4,0%	4,0%	100%	1,00	3,00	1,00
25	verd		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	2,5%	10%	100%	1,00	3,00	1,00
26	zoll		Das System soll die Möglichkeit bieten, Luftformal zu Generieren und zu Drucken.	✓	✗	✗	✗	✗	✗	0	0	50%	30%	67%	1,00	3,00	3,00
27	verd		Das System soll die Möglichkeit bieten, ohne Praktiken zu generieren.	✓	✗	✗	✗	✗	✗	1	0	21%	11%	100%	1,00	3,00	1,00
28	verd		Das System soll die Möglichkeit bieten, ohne Praktiken zu generieren (System erweitert Vorhanden).	✓	✗	✗	✗	✗	✗	2	0	1%	0%	100%	1,00	3,00	2,00
29	verd		Das System soll die Möglichkeit bieten, ohne Ressourcengruppe einzustellen (System erweitert Vorhanden).	✓	✗	✗	✗	✗	✗	2	0	1%	0%	100%	1,00	3,00	2,00
30	zoll		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,2%	6%	67%	1,00	3,00	8,00
31	zoll		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,2%	6%	67%	1,00	3,00	6,00
32	zoll		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,2%	6%	67%	1,00	3,00	6,00
33	zoll		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	4,0%	35%	67%	1,00	3,00	2,00
34	zoll		Das System soll die Möglichkeit bieten, ohne Struktur zu generieren.	✓	✗	✗	✗	✗	✗	0	0	6,6%	6%	67%	1,00	3,00	8,00

Abbildung 4.1.: Ausschnitt aus Excel¹

Der erste Ansatz war das Dokumentieren in Excel, dort wurden (wie in Abbildung 4.1 zu sehen) alle Anforderungen aufgenommen und erste Berechnungen der Prioritäten durchgeführt. Jedoch stellte sich im Laufe dieser Bearbeitung heraus, dass Excel nicht alle Funktionalitäten die benötigt werden zufriedenstellend abdeckt. Zudem erfolgt die Speicherung der Daten in einer lokalen Datei was die deren aktive Nutzung von mehreren Beteiligten erschwert. Also wurde die Entscheidung getroffen ein Programm zum

¹Ein Ausschnitt aus der Dokumentation der Anforderungen in Excel

entwickeln ?

Verwalten der Daten zu erstellen, damit dort alle benötigten Funktionen nach belieben implementiert werden können.

4.2. Webseite

hier kommt ein - hin, wenn sich das Verarbeitung auch auf Datenverarbeitung bezieht.
Datenhaltung und -verarbeitung
und es wird dann klein geschrieben

Eine einfache und effektive Lösung der Datenhaltung und Verarbeitung bot die Kombination einer Angular2-Webseite und einer MySQL-Datenbank. Durch vorherige Projekte bestand derzeit schon der Zugang zu einem Server und einer Datenbank, was diese Entscheidung stützte. Zudem ist die Erstellung einer Angular2-Webseite mit weniger Aufwand verbunden als ein Java-Programm, welches die andere Alternative gewesen wäre.

Ich persönlich würde immer Website anstatt Webseite schreiben

Durch diese Webseite bestand die Möglichkeit auch ohne eine persönliche Zusammenkunft mit dem Entwickler-Team einige Absprachen und Anpassungen durchzuführen. So konnten nicht nur dynamisch neue Anforderungen über ein perfekt dafür zurecht geschnittenes Interface hinzugefügt und mit den nötigen Informationen bestückt werden (Abbildung 4.2), sondern auch die Editierung dieser konnte in die davon abhängigen Berechnungen sofort mit einbezogen werden.

The screenshot shows a user interface for creating a new requirement. At the top, it says 'Neue Anforderung' and 'Gruppe: Auswertung'. Below this is a 'short description' input field. Under 'Beschreibung', there is a note: 'Das bestehende System muss einem Modulverantwortlichen die Möglichkeit geben,..'. To the right, there are 'Story-Points' set to 1. Below this are sections for 'Stakeholder' (with tabs for Entwickler-Team, Admin, Modulverantwortlicher, Mitarbeiter, Studentische Hilfskraft, and Student), 'System' (with tabs for Altes System and Neues System), 'Kano' (with tabs for Bedarf, Leistungsfaktor, and Begeisterungsfaktor), and 'Klassifizierung' (with tabs for Funktionale Anforderung and Nicht-Funktionale Anforderung). A red 'x' icon is visible next to the 'Funktionale Anforderung' tab.

Abbildung 4.2.: Erstellen einer Anforderung²

4.2.1. Gruppen

Ein weiterer Vorteil dieser Entscheidung war das einfach Gruppieren der Anforderungen. Um eine Anforderung einzufügen, wird diese direkt der jeweiligen Gruppe über den

²Erstellen einer Anforderung mit einem eigens dafür konzipierten Interface

"Plus-Button" hinzugefügt. Sofern eine bestehende Anforderung in eine neue Gruppe geschoben werden soll, kann dies mittels "Drag and Drop" realisiert werden. Auch bei diesem Vorgang werden alle Folgeberechnungen angepasst. Auch die Übersicht dieser Gruppen ist in dieser App wesentlich besser als sie in Excel gewesen wäre (siehe Abbildung 4.3).

Abbildung 4.3.: Ausschnitt der Webseite³

4.2.2. Labels

Das Programm bietet zudem ein Interface, mit dem man Labels für eine Gruppe erstellen kann (Abbildung 4.4)

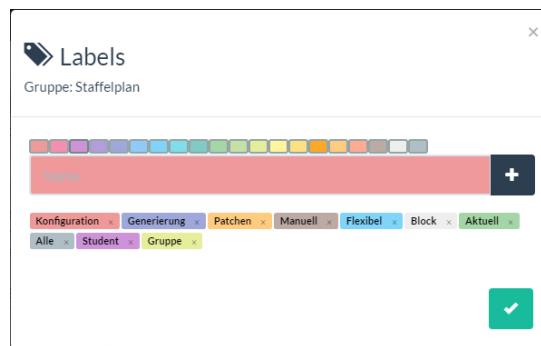


Abbildung 4.4.: Erstellen eines Labels⁴

³Ein Ausschnitt aus der Dokumentation der Anforderungen mit der Webseite

⁴Das Erstellen eines Labels in einer Gruppe über das User-Interface

Diese können anschließend den Anforderungen dieser Gruppe zugeordnet werden. (Abbildung 4.5).

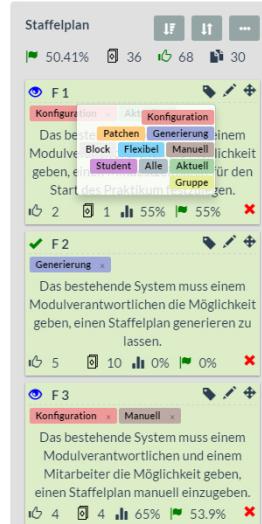


Abbildung 4.5.: Zuordnen eines Labels⁵

4.2.3. Informationen

Die Anforderungen selbst bieten alle nötigen Informationen auf einem Blick wie in Abbildung 4.6 zu erkennen ist.

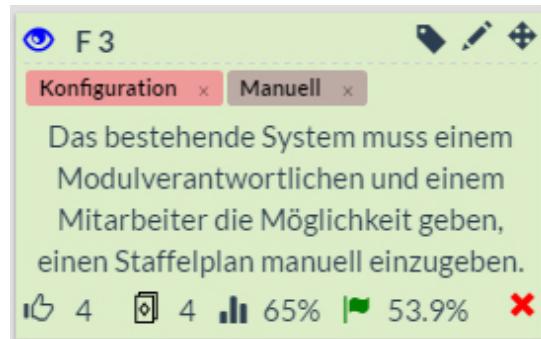


Abbildung 4.6.: Eine Anforderung im User-Interface⁶

Hier sind nicht nur die zugewiesenen Labels zu erkennen sondern diese Karte zeigt alle Informationen über diese Anforderung.

⁵Zuordnen eines Labels zu einer Anforderung mit dem User-Interface

⁶Die Detail-Ansicht einer Anforderung im User-Interface

-  Die Anzahl der Votes
-  Die Anzahl der Story-Points
-  Die Priorität in %
-  Das Ergebnis in %
-  Der Status der Anforderung, in diesem Fall "Valide"
-  Der Status der Anforderung, wenn sie "Erledigt" ist
-  Der Status der Anforderung, wenn sie "Verworfen" ist

Auch eine Gruppe hat alle Informationen direkt ersichtliche, die dort verwendeten Zeichen haben die gleiche Bedeutung wie bei einer Anforderung, mit dem Unterschied, dass diese Daten sich aus den Anforderungen der Gruppe zusammensetzen (Abbildung 4.7).



Abbildung 4.7.: Gruppeninformationen⁷

-  Der Durchschnitt aller Ergebnisse der beinhalteten Anforderungen
-  Die Summe aller Story-Points der beinhalteten Anforderungen
-  Die Summe aller Votes der beinhalteten Anforderungen
-  Die Anzahl beinhalteten Anforderungen

⁷Die Informationen einer Gruppe im User-Interface

4.2.4. Filter/Suche

Neben diesen Informationen stehen zahlreiche Filter-Methoden und Sortierungs-Methoden zur Verfügung, welche eine Visualisierung über Graphen oder Charts überflüssig machen, da durch diese Methoden alles Nötige ersichtlich ist. Welche Methoden das sind wird mit Abbildung 4.8 erklärt.

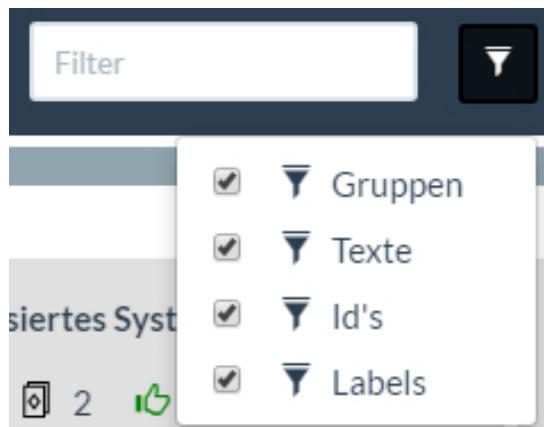


Abbildung 4.8.: Filter-Methoden⁸

Hierbei kann eingestellt werden, in welchen Elementen gesucht werden soll.

Gruppen:

Hier wird in allen Gruppennamen gesucht

Texte:

Hier wird in allen Texten, also in den Anforderungsbeschreibungen gesucht

Id's:

Bezieht die Id's der Anforderungen mit ein

Labels:

Bezieht die Labels mit ein, welche den Anforderungen zugeordnet wurden

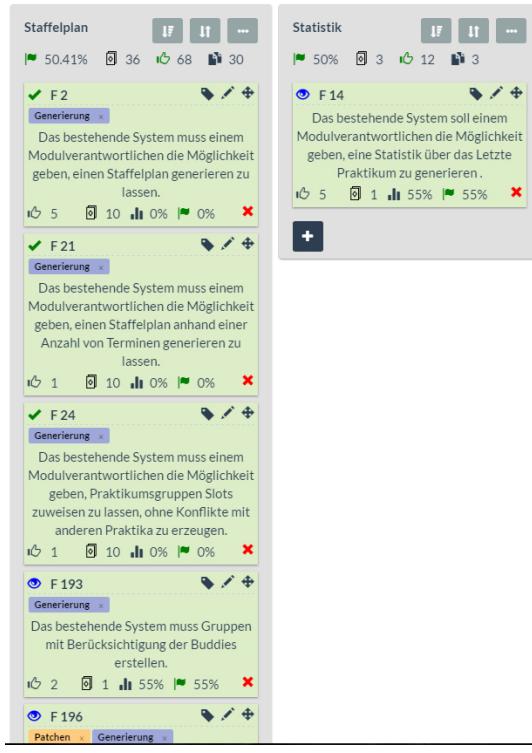
Zudem kann nach mehreren Texten gleichzeitig gesucht werden. Dazu wird für eine "und" Verknüpfung ein ", " verwendet und für eine "oder" Verknüpfung ein "; ". Hierbei ist zu beachten, dass eine "und" Verknüpfung immer höher priorisiert wird. SO könnte ein Filter-String wie folgt aussehen:

"Staffelplan, Generierung; Stat, 14"

Nach dem deaktivieren des Texte-Filters sieht das Ergebnis wie in Abbildung 4.9 aus.

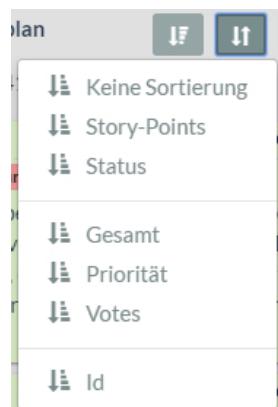
⁸Ein Ausschnitt von dem Header des User-Interface mit allen Filter-Methoden

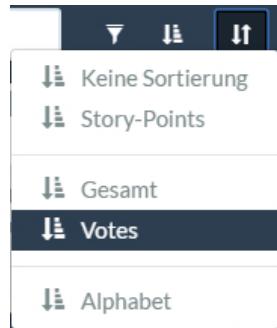
⁹Das Ergebnis der Filterung nach "Staffelplan, Generierung; Stat, 14"

Abbildung 4.9.: Filter-Ergebnis⁹

4.2.5. Sortierung

Durch die Kombination der Filter-Methoden mit den Sortierungs-Methoden können beinahe alle möglichen Anfragen erzeugt werden. Auch hier stehen einige Auswahlmöglichkeiten zur Verfügung. Die Abbildungen 4.10 welche die Sortierung in einer Gruppe anzeigt und 4.11 auf der die Möglichkeiten für die Sortierung aller Gruppen zu sehen ist sollten selbsterklärend sein.

Abbildung 4.10.: Sortierung innerhalb einer Gruppe¹⁰

Abbildung 4.11.: Sortierung gesamt¹¹

4.2.6. Sichtbarkeit

Da sich die Filterung nur auf die Anforderungen selbst bezieht kann es sein, dass nach einer Filterung die Gruppen, welche keine zutreffenden Anforderungen beinhalten zwar leer sind, jedoch trotzdem angezeigt werden. Dies hat den Grund, da es ja durchaus sein kann, dass nach der Filterung eine der gefilterten Anforderungen in eine dann leere Gruppe verschoben werden soll. Ist dies nicht erwünscht besteht die Möglichkeit auch dies zu deaktivieren. Auch die Sichtbarkeit der Anforderungen mit einem bestimmten Status kann eingestellt werden, wie auf Abbildung 4.12 zu erkennen ist.

Abbildung 4.12.: Sichtbarkeit¹²

All diese Funktionen haben wesentlich zur Bearbeitung und Auswertung der Anforderungen beigetragen und werden es auch noch in Zukunft tun. Diese Web-Anwendung steht unter der Adresse req.herborn-software.com zur Verfügung. Der Login wird nur auf Anfrage bei dem Verfasser dieses Dokuments vergeben, da Änderungen der Anforderungen direkte Auswirkung auf den Datenbestand haben, und diese Informationen auch nach diesem Projekt weiter verwendet werden.⁶⁶ [?]

¹⁰Ein Ausschnitt des User-Interface, auf dem die Möglichkeiten zur Sortierung innerhalb einer Gruppe zu sehen sind

¹¹Ein Ausschnitt des User-Interface, auf dem die Möglichkeiten zur Sortierung aller Gruppen zu sehen sind

¹²Ein Ausschnitt des User-Interface, auf dem die Möglichkeiten zur Verstellung der Sichtbarkeit zu sehen sind

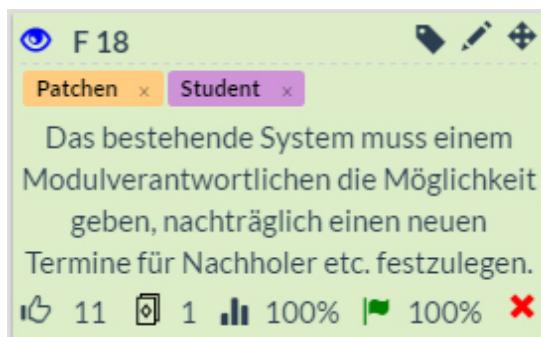
dass

Klingt etwas schwammig
Empfehle „vom“ damit
sich das auf „vergeben“
bezieht.

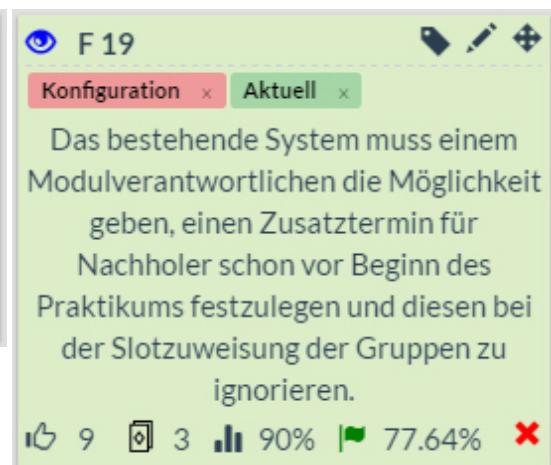
5. Ergebnisse

Die Auflistung von 150 Anforderungen wäre an dieser Stelle unangebracht, weshalb hier nur die Top-10 der wichtigsten Anforderungs-Gruppe "Staffelplan" aufgeführt werden. Diese Auflistung sollte genug über die Ergebnisse dieser Arbeit aussagen.

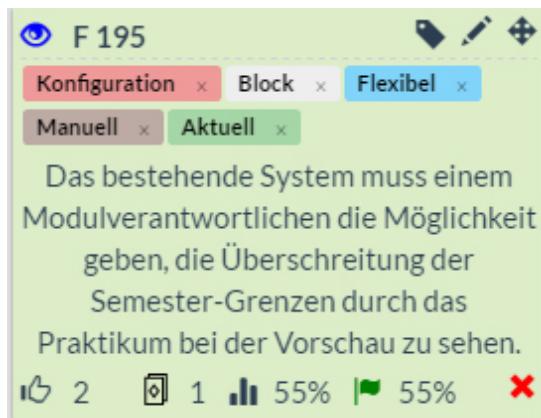
1. Platz



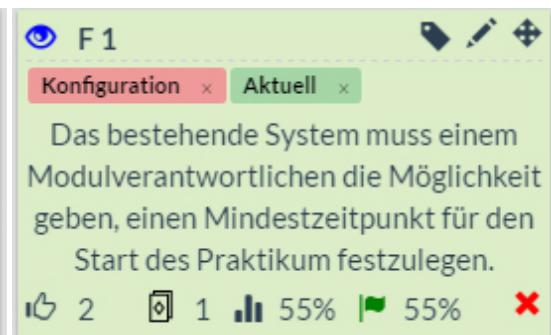
2. Platz



3. Platz



4. Platz



5. Platz

F 200

Konfiguration x Flexibel x Aktuell x
Block x Manuell x

Das bestehende System muss einem Modulverantwortlichen die Möglichkeit geben, die Konflikte bei der Staffelplangenerierung einsehen zu können. Hierbei kann eine Liste der betroffenen Personen / Gruppen angezeigt werden.

⌚ 2 ⏱ 1 55% 🚧 55% ✗

6. Platz

F 199

Konfiguration x Flexibel x Aktuell x
Block x Manuell x

Das bestehende System muss einem Modulverantwortlichen die Möglichkeit geben, den Staffelplan mit Hilfe der HOPS-Daten vorausfüllen zu lassen, sofern noch keine abweichenden Daten erstellt wurden. Dabei sollen Praktikums- und Vorlesungsdaten aus dem HOPS bezogen werden.

⌚ 2 ⏱ 1 55% 🚧 55% ✗

7. Platz

F 196

Patchen x Generierung x

Das bestehende System muss bei der Verschiebung eines Termins auch die nachfolgenden Termine anpassen.

⌚ 2 ⏱ 1 55% 🚧 55% ✗

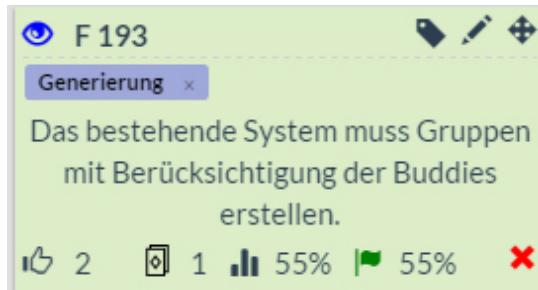
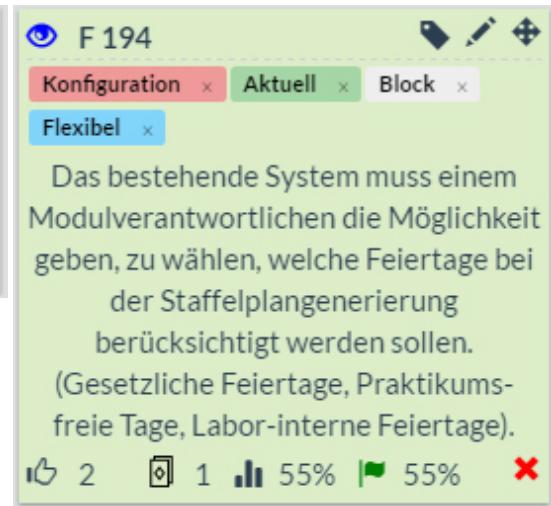
8. Platz

F 121

Patchen x Student x

Das bestehende System soll einem Modulverantwortlichen die Möglichkeit geben, Termine von Personen temporär zu verschieben und dabei zwischen einem Ersatztermin oder einem Straftermin zu unterscheiden.

⌚ 5 ⏱ 1 55% 🚧 55% ✗

9. Platz**10. Platz**

Fazit

Dieses Projekt hatte das Ziel, Anforderungen für das LWM zur Verbesserung der Nutzbarkeit und Erweiterung des Nutzerkreises zu ermitteln. Dazu wurden Interviews mit den potentiellen neuen Usern und den bisherigen Nutzern durchgeführt. Die dort entstandenen Ergebnisse wurden zu einer soliden Grundlage für die weitere Entwicklung des Tools aufgearbeitet.

Diese Arbeit bringt hervor, dass bei vielen Modulen, die dieses Tool noch nicht nutzen ein Interesse zur Nutzung vorhanden ist. Allerdings war dieses den Modulverantwortlichen bislang nicht bekannt, oder die benötigten Funktionen sind in der derzeitigen Anwendung noch nicht implementiert. Durch die Implementierung der hier ermittelten Anforderungen wäre es denkbar, auch diese Module als Nutzer zu gewinnen.

Des weiteren stellte sich heraus, dass das Programm auch bei den aktuellen Nutzern teilweise Wünsche offen lässt, weshalb die Berücksichtigung der hier erlangten Erkenntnisse auch die Zufriedenheit der jetzigen Nutzer steigern könnte.

Durch diese Arbeit wird deutlich, dass die Anforderungsanalyse ein wichtiger Schritt in der Entwicklung einer Software ist. Durch die sorgfältige Durchführung kann nicht nur die Usability einer Anwendung maßgeblich verbessert werden, sondern auch das Interesse bei einer viel größeren Nutzerzahl geweckt werden. Und gerade letzteres ist meistens der wichtigste Anreiz bei der Entwicklung einer Software.

Denn was ist denn eine Software ohne Nutzer?

Schönes Schlusswort

Abbildungsverzeichnis

2.1. Das Kano-Modell	12
2.2. Penalty-Reward-Analysis	14
3.1. Gruppeneinteilung zweiter Ebene	29
3.2. Gruppeneinteilung dritter & vierter Ebene	30
3.3. Erklärung der Anforderungsauswertung	33
3.4. Erklärung zur Rechnung 100 - p	34
4.1. Ausschnitt aus Excel	36
4.2. Erstellen einer Anforderung	37
4.3. Ausschnitt der Webseite	38
4.4. Erstellen eines Labels	38
4.5. Zuordnen eines Labels	39
4.6. Eine Anforderung im User-Interface	39
4.7. Gruppeninformationen	40
4.8. Filter-Methoden	41
4.9. Filter-Ergebnis	42
4.10. Sortierung innerhalb einer Gruppe	42
4.11. Sortierung gesamt	43
4.12. Sichtbarkeit	43

Glossar

- Anforderung Eine Anforderung (engl. Requirement) ist eine Aussage über eine zu erfüllende Eigenschaft (einem Ziel) oder zu erbringende Leistung eines Produktes, Systems oder Prozesses. Also etwas, das das Zielprodukt/-system können muss. (R.Heini 2017)
- Bonuspunkte Eine Prüfungsleistung in Punkten, die der Praktikumsteilnehmer während des Praktikums durch seine Leistungen sammeln kann.
- Epic Die Beschreibung einer Anforderung an eine Software auf einer hohen Abstraktionsebene.
- Funktion-Points Ein Maß für den Aufwand der Umsetzung einer Funktionalität. Diese finden Anwendung in der Function-Point-Analyse
- Funktionale Anforderungen Funktionale Anforderungen beschreiben gewünschte Funktionalitäten (was soll das System tun/können) eines Systems bzw. Produkts, dessen Daten oder Verhalten (R.Heini 2017)
- Modul Ein Modul aus dem Modulhandbuch der TH-Köln.
- Potentieller Nutzer . Ein möglicher zukünftiger Nutzer eines Systems.
- Produkt Ein weiteres Synonym für das zu entwickelnde System.
- Projekt Der Prozess zur Planung und Entwicklung eines Systems.
- Qualitätseigenschaften Qualitätseigenschaften beschreiben die Qualität in welcher die geforderte Funktionalität zu erbringen ist.
- Randbedingung Eine Randbedingung ist eine Bedingung die Einfluss auf das Projekt haben kann.
- Requirement-Engineer Die Person, welche das Requirement-Engineering durchführt.
- Requirement-Engineering Der Prozess der Ermittlung, Analyse, Auswertung und Bearbeitung von Anforderungen.
- Staffelplan Ein Terminplan für das Praktikum in dem alle Termine mit den dazugehörigen Gruppen / Praktikumsteilnehmer eingetragen sind.

- Stakeholder Ein Stakeholder ist eine Person, Organisation oder Gruppe von Personen, die am Projekt aktiv beteiligt ist oder durch den Projektverlauf oder das Projektergebnis beeinflusst wird (sowohl positiv wie auch negativ), oder die den Projektverlauf oder das Projektergebnis beeinflussen kann (auch dies positiv oder negativ). (R.Heini 2017)
- Story-Points Ein Maß zur Angabe der Komplexität einer User-Story.
- System System wird im Kontext dieser Arbeit als Synonym für das zu entwickelnde System eingesetzt.
- Usability Das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem Nutzungskontext benutzt werden kann, um ein Ziel effektiv, effizient und zufriedenstellend zu erreichen.
- Use-Cases Eine Art Funktionalitäten die ein System bieten soll zu beschreiben.
- User Der Nutzer eines Systems.
- User-Story Eine Art Anforderungen aus der Sicht des Nutzer zu dokumentieren.
- Ziel Ein Ziel bezeichnet einen in der Zukunft liegenden, gegenüber dem Heutigen im allgemeinen veränderten, angestrebten Zustand. Ein Ziel ist somit ein definierter und angestrebter Endpunkt eines Prozesses. (R.Heini 2017)

Abkürzungsverzeichnis

LWM	Labwork Management	3
RE	Requirements Engineering	3
US	User-Story	35
SP	Story-Points	35
FPA	Funktion-Point-Analyse	35
FP	Funktion-Points	22
SHK	studentische Hilfskräfte	26
WHK	wissenschaftliche Hilfskräfte	26

Literaturverzeichnis

betriebswirtschaft-lernen.net 2017

BETRIEBSWIRTSCHAFT-LERNEN.NET: *COCOMO (Constructive Cost Model)*. 2017. – <http://www.betriebswirtschaft-lernen.net/erklaerung/coco-mo-constructive-cost-model/>. Sichtung: 14.06.2017

Bornemann, Stefan 2017

BORNEMANN, STEFAN: *Stakeholder – Ihre Interessen und deren Analyse*. 2017. – <http://www.lead-conduct.de/2012/11/28/stakeholder/>. Sichtung: 22.05.2017

cosmic-sizing.org 2017

COSMIC-SIZING.ORG: *COSMIC Method*. 2017. – <http://cosmic-sizing.org/cosmic-fsm/>. Sichtung: 14.06.2017

Die SOPHISTen 2013

DIE SOPHISTEN: »Schablonen für alle Fälle«. 2013. – https://www.sophist.de/fileadmin/SOPHIST/Publikationen/Broschueren/SOPHIST_Broschuere_MASTeR.pdf. Sichtung: 24.05.2017

Dipl.-Ing. Klaus Lipinski 2013

DIPL.-ING. KLAUS LIPINSKI: *Function-Point-Methode*. 2013. – <http://www.itwissen.info/Function-Point-Methode-function-point-method.html>. Sichtung: 25.05.2017

Dr.-Ing. Ina Schaefer 2011

DR.-ING. INA SCHAEFER: *Anforderungsanalyse*. 2011. – <https://www.tu-braunschweig.de/Medien-DB/isf/sse/v2-re.pdf>. Sichtung: 24.05.2017

Dr. Jürgen Fleig 2016

DR. JÜRGEN FLEIG: *Was sind Stakeholder und was bedeutet das Stakeholder-Konzept?* 2016. – <https://www.business-wissen.de/hb/was-sind-stakeholder-und-was-bedeutet-das-stakeholder-konzept/>. Sichtung: 23.05.2017

Harward, Mark 2011

HARWARD, MARK: *Wasserfallmodell versus Scrum*. 2011. – <https://www.fernuni-hagen.de/imperia/md/content/ps/masterarbeit-harwardt.pdf>. Sichtung: 24.05.2017

it-agile GmbH 2017

IT-AGILE GMBH: *Agiles Schätzen*. 2017. – <https://www.it-agile.de/wissen/praktiken/schaetzen/>. Sichtung: 22.05.2017

Kuhn, Andrea 2017

KUHN, ANDREA: *Penalty-Reward-Faktoren Ansatz*. 2017. – <http://www.durch-lernen-zum-erfolg.de/lexikon/penalty-reward-faktoren-ansatz/>. Sichtung: 25.05.2017

Lauenroth, Kim 2011

LAUENROTH, KIM: *Eine kleine praktische Philosophie über das Requirements Engineering*. 2011. – <https://de.slideshare.net/adessoAG/vortrag-kim-lauenroth>. Sichtung: 06.06.2017

Preuss 2013

PREUSS, Sebastian: *Scrum-Projekte: Schätzungen auf Basis von Story Points*. 2013. – <https://blog.seibert-media.net/blog/2013/03/07/scrum-projekte-beschätzungen-auf-basis-von-story-points/>. Sichtung: 23.05.2017

R.Heini 2017

R.HEINI: *Glossar*. 2017. – <http://www.anforderungsmanagement.ch/glossar/>. Sichtung: 06.06.2017

Rupp & Pohl 2015

RUPP, Chris; POHL, Klaus: *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. 3. dpunkt.verlag, Heidelberg, 2015. – ISBN 3864916747, 9783864916748

Rupp & Schüpferling 2017

RUPP, Chris; SCHÜPFERLING, Dirk: *Anforderungsermittlung - Hellsehen für Fortgeschrittene*. 2017. – [http://zugang.sophist.de/DownloadDB.nsf/0/2e76df6e9fb899bfc125770b00577666/\\$FILE/Requirements%20Engineering%20Management,%205.%20Auflage%20-%20Kap%205%20Ermittlung.pdf#](http://zugang.sophist.de/DownloadDB.nsf/0/2e76df6e9fb899bfc125770b00577666/$FILE/Requirements%20Engineering%20Management,%205.%20Auflage%20-%20Kap%205%20Ermittlung.pdf#). Sichtung: 24.05.2017

Rupp & die SOPHISTen 2014

RUPP, Chris; SOPHISTEN die: *Requirements-Engineering und -Management*. 6. Carl Hanser Verlag GmbH Co KG, München, 2014. – ISBN 3446443134, 9783446443136

Schäfer 2008

SCHÄFER, Torsten F.: *Stakeholderorientiertes Integrationsmanagement bei Fusionen und Akquisitionen*. 1. Gabler Verlag, Wiesbaden, 2008. – ISBN 978-3-8350-0985-1

Taentzer 2014

TAENTZER: *Grundlagen der Anforderungsanalyse*. 2014. – <https://www.uni-mbawburg.de/fb12/arbeitsgruppen/swt/lehre/files/est1415/EST141028.pdf>. Sichtung: 25.05.2017

Weigand, Florian 2017

WEIGAND, FLORIAN: *Aufwandsschätzung in IT-Großprojekten „Function Point Methode.“* 2017. – <https://wwwmatthes.in.tum.de/file/stdmz5ua8sr7/Sebis-Public-Website/-/Proseminar/Weigand-Function-Point-Methode-Ausarbeitung.pdf>. Sichtung: 25.05.2017

Wiegand, Sven 2015

WIEGAND, SVEN: *Story-Points: Normierung der Skala und Planung.* 2015. – <http://www.itwissen.info/Function-Point-Methode-function-point-method.html>. Sichtung: 01.06.2017

WinfWiki 2017

WINFWIKI: *Methoden und Verfahren der Aufwandschätzung im Vergleich.* 2017. – http://winfwiki.wi-fom.de/index.php/Methoden_und_Verfahren_der_Aufwandsch%C3%A4tzung_im_Vergleich#Methoden_der_Aufwandsch.C3.A4tzung. Sichtung: 06.06.2017

Anhang

A. Fragebogen

Anforderungsermittlung Praktikumstool

Allgemeines

Modul: _____

Modulverantwortlicher: _____ **Labor:** _____

Aufgaben: _____ **Termine:** _____

Sind die Termine an die Vorlesung gekoppelt?

ja Nein

Praktikumsverlauf: _____

Praktikumsbewertung: _____

Zusatzleistungen: _____

Bonuspunkte: _____

Nachholtermine: _____

Praktikumsanerkennung: Ja nein

Spezielles

Krankheit von Studenten: _____

Kommunikation: _____

Bearbeitungszeit: _____

Bahnstreik: _____

Kooperation mit anderen

Modulen: ja nein

Wiederholung des Praktikums
obwohl schon einmal bestanden: ja nein

Bemerkungen

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 14. Juni 2017

Florian Herborn