How Good Design Survives Development

Florian Röder, Andreas Gartz, Ivo Jacobs

SprintEins GmbH

Zusammenfassung

Agile Softwareentwicklung ist mittlerweile in vielen Unternehmen zum Standardvorgehen für neue Projekte geworden (Loranger & Laubheimer, 2017). Idealerweise sind die Entwicklungsteams in der Lage, neue Funktionen schnell zu entwickeln und kontinuierlich in den Betrieb zu bringen, wenn sie Praktiken der Continuous Integration (CI) und Continuous Delivery (CD) einsetzen. Und obwohl in der agilen Entwicklung viele von interdisziplinären Teams sprechen, sind UX Designer[1] oft nicht in den Entwicklungsteams vertreten. UX Design wird nach wie vor in vielen Projekten zu Beginn betrieben, teilweise auch von externen Designagenturen und dann in "Wellen" der Entwicklung übergeben. Diese Übergaben führen einerseits zu Reibungsverlusten und andererseits dazu, dass iterative Anpassungen des Designs nicht effizient vorgenommen werden können. Das wirkt sich oft negativ auf die Qualität der User Experience aus. Anhand von Beispielen zeigen wir, welche Mittel und Wege – unserer Erfahrung nach – helfen, das UX Design im Produkt schneller und besser iterieren zu können und Übergaben so weit wie möglich zu vermeiden. Wir erörtern, ob es Sinn macht, in Anlehnung an DevOps von DesignOps zu sprechen.

1 Einleitung

Der Begriff DevOps findet seit 2009 Verwendung (DevOpsDays 2009 in Ghent) und verändert die Art und Weise wie Software ausgeliefert und aktualisiert wird. Um zu verstehen, warum das ggf. Einfluss auf die Arbeit von UX Designern hat, wollen wir erst die Begriffe DevOps und User Experience Design näher erläutern.

[1] Wir verwenden die Rolle UX Designer stellvertretend für alle UX relevanten Rollen wie UX Researcher, Interaction Designer, Visual Designer, Usability Tester, Usability Engineer.

Veröffentlicht durch die Gesellschaft für Informatik e. V. und die German UPA e. V. 2018 in
S. Hess, H. Fischer (Hrsg.):
Mensch und Computer 2018 – Usability Professionals, 02.–05. September 2018, Dresden.
 https://doi.org/10.18420/muc2018-up-0205

2 DevOps

Unter DevOps verstehen wir den Ansatz, die Softwareentwicklung, die Qualitätssicherung und die Auslieferung in den Betrieb möglichst effizient zu verzahnen, um Software in hoher Qualität schneller ausliefern zu können, und dadurch Anwendern oder Kunden schneller einen Wert liefern zu können. Dies erfordert im Allgemeinen eine Abstimmung der eingesetzten Tools, ein breiteres Verständnis der Teammitglieder und eine gut abgestimmte Zusammenarbeit im Team von Kollegen, die klassisch womöglich in getrennten Abteilungen gearbeitet hatten. („DevOps“, Wikipedia, 2018).

Eine positive Folge von DevOps ist die Möglichkeit, eine effizientere Feedbackschleife im Entwicklungsprozess zu nutzen. Durch entsprechende Analyse und Auswertung, wie die Software im Betrieb tatsächlich genutzt wird, gelangen relevante Erkenntnisse und Nutzerfeedback direkt zum Entwicklungsteam. Dies befähigt insbesondere agile Teams, direkt darauf zu reagieren.

2.1 User Experience Design

Unter User Experience Design (UX Design) verstehen wir den Ansatz, die Endnutzererfahrung von Produkten und Lösungen durch eine übergreifende, nutzerzentrierte Betrachtung (User Centered Design) zu optimieren. Etablierte, nutzerzentrierte Praktiken, um Lösungen mit guter User Experience zu erstellen – wie z.B. der Design Thinking Prozess – eignen sich, um mit relativ wenig Aufwand möglichst gute Lösungen zu finden und risikoarme Entscheidungen treffen zu können. Frühe, simulierte Tests mit Prototypen führen zu schnellen Erkenntnissen über die Zielgruppe, die Passung der Lösung und die Güte der Nutzererfahrung.

Die getestete Lösung wird anschließend in Form von Prototypen, Styleguides und ggf. Pattern Libraries für die Entwicklung spezifiziert und übergeben. Um eine gute User Experience sicherzustellen, findet allgemein eine Kontrolle bzgl. der Umsetzungsqualität des spezifizierten Designs in einer Korrekturschleife statt.

Unserer Erfahrung nach führen diese Übergaben und Korrekturschleifen zu massiven Reibungsverlusten zwischen UX Designern und Entwicklungsteams (die DevOps einsetzen) und mindern den Einfluss der Designer auf das tatsächlich ausgelieferte Produkt. Die Gründe hierfür sind z.B.

- Stetiges Feedback aus dem Betrieb bedingt kontinuierliche funktionale Korrekturen, mit denen die gewünschten UX Maßnahmen und Designkorrekturen konkurrieren
- Die Umsetzung von UX Maßnahmen kann schwerwiegende oder ungeplante Änderungen an der technischen Struktur bedingen und wird als fraglich bewertet
- Das Schaffen neuer Funktionen wird als wichtiger angesehen als die Verbesserungen bestehender Funktionen
- UX-relevante Erkenntnisse aus der Entwicklung und dem Betrieb stellen UX Maßnahmen in Frage, fließen aber nicht effektiv zurück zum UX Designer

Im Bericht State of UX Agile Development der Nielson Norman Group von 2017 wird vermerkt, dass UX Praktizierende ihren Einfluss auf ihr Projekt im Durchschnitt mit 4 von 7 bewerten (Loranger & Laubheimer, 2017). Diesen Einfluss gilt es zu erhöhen, damit die Nutzer auch davon profitieren können.

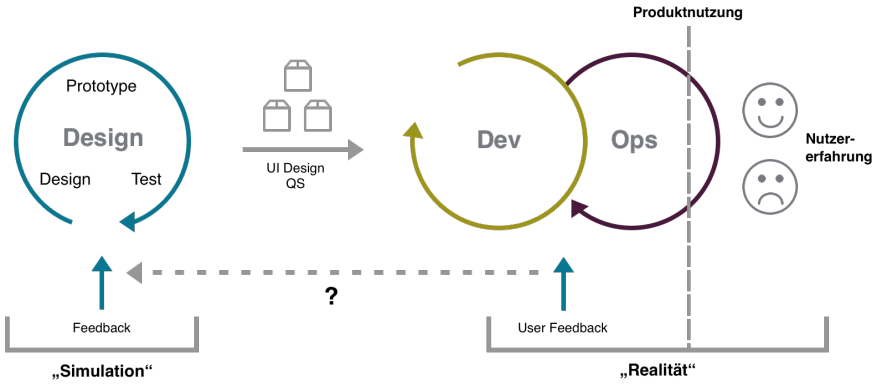

Abbildung 1: Das Design Team erhält nur Feedback aus Simulationen, während das Entwicklungsteam durch DevOps Praktiken reales Nutzerfeedback aus dem Betrieb erhält. Die Einbindung des Design Teams in diese Feedbackschleife ist schwierig und findet häufig nicht effektiv statt.

3 Fallbeispiel: UX im agilen Großprojekt

Durch einen unserer Kunden sind wir für das Thema UX Design in einem Großprojekt beauftragt. Das Projekt wurde mit einem agilen Entwicklungsteam aufgesetzt und wir hatten zu Beginn des Projekts, gemeinsam mit dem Kunden, eine initiale Lösung anhand eines Design-Thinking Prozesses erarbeitet. Als Übergabe für die Entwicklung hatten wir iterativ einen Prototyp in Zieltechnologie entwickelt, den wir vorab in mehreren Usability Tests prüfen konnten. Diese Tätigkeiten waren zu Beginn des Projektes noch vergleichsweise einfach mit dem Entwicklungsteam abzustimmen und nach wenigen Sprints konnten wir unsere Testergebnisse am lauffähigen Produkt verifizieren.

Im Laufe des Projektes sind weitere Teilprodukte und mehrere Teams hinzugekommen. Die Prozesse für ein Softwarerelease wurden im Sinne von DevOps deutlich verbessert und sind theoretisch nach jedem Sprint möglich (aktuell wird allerdings meist nach 4 Sprints, also 8 Wochen eine Veröffentlichung gemacht). Über Tools wie Adobe Analytics und eine Data-Warehouse Lösung können Nutzertätigkeiten und Bewegungen detailliert über mehrere Plattformen (z.B. Browser am Desktop-Computer, mobile App unter iOS oder Android) gemessen werden. Erkenntnisse aus diesen Daten können von den Teams direkt in Verbesserungen oder Änderungen überführt werden. Im Projektverlauf (das Projekt läuft seit ca. 3.5 Jahren) haben wir auch die eingesetzten UX Aktivitäten und Tools immer wieder verbessert. Dies wollen wir im Folgenden detailliert diskutieren.

3.1 Vom UX Design Team zum UX Chapter

Angefangen haben wir das Projekt nah am Fachbereich in einem separaten UX Team mit viel technologischem Verständnis. Die Ergebnisse wurden in Form von HTML Templates (Web) und Screendesigns mit detaillierten Spezifikationen (iOS und Android) den Entwicklungsteams übergeben. Folgende Details verringerten den wahrgenommenen Einfluss auf das tatsächliche Produkt und dessen UX durch die UX Designer:

- Feedback zu Prototypen erzeugte (zu viele) Iterationen an den Prototypen selber, aber entscheidend wäre gewesen, was davon letztendlich im Produkt landet.
- Ergebnisse aus Research Aktivitäten, Prototypen und User Tests waren bei Übergabe an die Entwicklung teilweise schon obsolet.
- Die Entwicklungsteams konnten nicht auf Designartefakte warten, was zur Folge hatte, dass mögliche negative Auswirkungen auf die UX kaum berücksichtigt wurden.
- Feedback aus dem Betrieb wurde nur gefiltert an das Design Team übergeben. Bei Zeitdruck sank auch die Bereitschaft Informationen überhaupt weiterzugeben.

Eine mögliche Lösung für die oben genannten Probleme ist die Integration von UX Designern direkt in den agilen Entwicklungsteams. Diese Schlussfolgerung ist nicht neu:

"[It] is problematic when it comes to integrating usability and good UX with Agile development teams. All experiences from our case studies indicate that UX people must be co-located with developers and other project team members. Indeed, UX should be considered a part of the project team, not an outside department." (Nielsen, 2010)

Allerdings ist es nicht einfach damit getan, den UX Designer in das Entwicklungsteam zu setzen. Vielmehr ist hier je nach Umfeld und Erfahrung aller Beteiligten wichtig, ein gemeinsames Verständnis für UX Tätigkeiten und auch das gegenseitige Verständnis für die Arbeit anderer Disziplinen zu entwickeln.

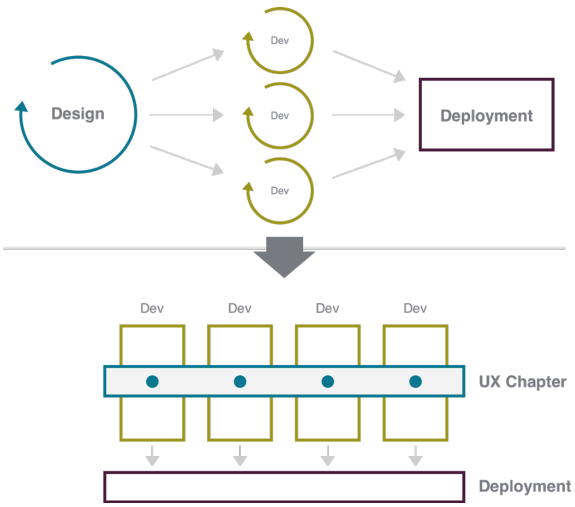

Abbildung 2: Die Organisationsstruktur vorher und nachher

Um dies erreichen zu können, wurde in dem Projekt die Organisation des Spotify Modells adaptiert. Dieses sieht die Zuordnung und Integration der UX Designer zu den Entwicklungsteams vor, ergänzt aber zusätzlich eine Vereinigung der UX Designer, ein sog. "UX Chapter", um übergreifende Fragen und Kriterien im Projekt[2] zu koordinieren (Knieberg, 2014). Die Kapazität der UX Designer verteilt sich entsprechend auf übergreifende Konzeptionierung im UX Chapter und die konkrete Umsetzung in den Entwicklungsteams. Diese Aufteilung kompensiert den verminderten Handlungsspielraum der UX Designer in agilen Entwicklungsteams – eine viel zitierte Problemstellung im Kontext von agiler Entwicklung (Laubheimer, 2017).

Wahrgenommene Vorteile dieser Organisation, die auch von den anderen Mitgliedern in den Entwicklungsteams benannt wurden, sind:

- Direkte Kooperation verbessert die Schätzung von Aufwänden und die Ergebnisqualität bzgl. der UX Maßnahmen in den Teams.
- Das generelle Verständnis für UX Kriterien steigt in allen Teams durch stetige Präsentation der UX Maßnahmen und UX Ergebnissen in Reviews.
- Die geteilte Verantwortung für notwendige UX Maßnahmen wird gestärkt und steigert die Motivation und Wahrscheinlichkeit, dass Verbesserungen den Weg ins Produkt finden.

Gleichzeitig erfordert diese Integration von UX Designern ein vertieftes Verständnis für die Aktivitäten der anderen Teammitglieder. Dies kann abhängig vom Kontext teilweise sehr technische Themen und Fragestellungen beinhalten. Unserer Erfahrung nach ist es nicht notwendig für die UX Designer alles bis ins kleinste Detail zu durchdringen, aber die grundlegende Bereitschaft, sich mit technischen Themen zu beschäftigen, ist fundamental.

3.2 UX Aktivitäten in kurzen Zyklen

Bei UX Aktivitäten wie User Research oder User Tests sind die kurzen Entwicklungszyklen eine Herausforderung. Um eine hohe Akzeptanz bzgl. der Aktivitäten zu erreichen, ist es hilfreich, diese möglichst effizient durchführen zu können.

3.2.1 UX Metrik

Wir haben Erfahrung mit dem Einsatz von verschiedenen Metriken zur Messung der UX gesammelt: AttrakDiff, UEQ und HEART (Happiness, Engagement, Adoption, Retention, Task Success) Framework (Rodden et. al, 2010).

Für uns hat sich das HEART Framework aus den folgenden Gründen als am effektivsten herausgestellt:

---

[2] Bei den übergreifenden Aufgaben geht es um Konzeptionsarbeit, die sich keinem einzelnen Team zuordnen lässt, wie z.B. die Nutzererfahrung über mehrere Teams oder Produkte hinweg, als auch die Schärfung und Visualisierung der mittelfristigen Produktvision.

- Das Framework setzt klare, transparente Ziele, die eine Ableitung erlauben, warum eine UX-relevante Metrik überhaupt erreicht werden sollte.
- Notwendige UX Verbesserungen lassen sich immer auf diese Ziele beziehen, die für das Team transparent sind. Dies erhöht die Wahrscheinlichkeit, dass die Verbesserungen auch in das Produkt integriert werden
- Das Framework hilft bei der Priorisierung verschiedener Metriken untereinander.

Auf Basis des HEART Frameworks haben wir über das UX Chapter ein UX Dashboard etabliert, auf dem alle relevanten Metriken aufgeführt werden und somit für eine bessere Transparenz für alle Teammitglieder sorgen.

3.2.2 Analytics

Ein direkter Zugang zu Analytics Tools[3] und Auswertungen aus einem Data-Warehouse befähigen das UX Chapter, die übergreifenden Metriken gemeinsam mit dem PO oder dem Team zu prüfen und zu aktualisieren. Eine vorhergehende Filterung und lange Dienstwege werden vermieden. Jedoch erfordert dies je nach Vorkenntnissen eine Schulung oder Einarbeitungszeit, da sichergestellt sein sollte, dass die Auswertung der Daten auf eine korrekte und konsolidierte Art und Weise geschieht. Hier hat sich das Einrichten von vordefinierten Dashboards oder Abfragen durch Experten als gute Praxis erwiesen. Dies befähigt die Kollegen aus dem UX Chapter nach einem Release, die Veränderung eigenständig zu beobachten oder für Verbesserungsideen auszuwerten[4].

3.2.3 Social Media Research

Als Ergänzung zu der quantitativen Analyse haben wir die Auswertung von Feedback aus Sozialen Medien ergänzt. Im einfachsten Fall werden hier über entsprechende Tools wie AppFollow Bewertungen aus dem Apple AppStore oder Google PlayStore nachgehalten. Darüber hinaus sammeln und gruppieren wir auch Anwender-Feedback aus Sozialen Medien wie Twitter, Instagram oder Facebook. Probleme werden insbesondere in den sozialen Medien ungefiltert geteilt und lassen sich effizient identifizieren[5]. Zur Auswertung setzen wir die Schritte ein, die in "Lean UX" beschrieben werden (Gotthelf & Seiden, 2016):

- Nach Mustern suchen: Wiederholtes, qualitatives Feedback wird mit quantitativen Metriken aus dem Betrieb verglichen und relevante Maßnahmen abgeleitet
- Ausreißer werden in einer Art Backlog gesammelt. Da diese Art von Research kontinuierlich stattfindet, können sich Ausreißer zu Mustern entwickeln
- Bei Zweifeln an dem Feedback werden möglichst andere Quellen gesucht und ergänzt

---

3 z.B. Adobe Analytics oder QlikSense

4 z.B. als Vorbereitung für einen Ideation Workshop

5 https://www.brandwatch.com/blog/social-media-for-ux/

3.2.4 Rapid User Testing

Der Aufwand für klassische User Tests setzt sich in etwa aus den folgenden Tätigkeiten zusammen: Anforderungen definieren, Probanden akquirieren, Prototyp vorbereiten, technisches Setup für die Durchführung aufsetzen, Interview Leitfaden erstellen, Durchführung mit zusätzlichen Beobachtern, Auswertung und Aufbereitung inkl. empfohlenen Maßnahmen. Zu Beginn testeten wir umfangreiche Nutzungsszenarien mit nur wenigen konkreten Fragestellungen, um Erkenntnisse über weite Teile der Nutzererfahrung explorativ zu sammeln.

Um für diesen Aufwand eine hohe Akzeptanz der Stakeholder zu erreichen, haben wir direkt von Beginn an darauf geachtet, alle Maßnahmen möglichst in einer vorgegebenen Zeitspanne durchführbar zu machen. Trotzdem lag der zeitliche Bedarf für die Ausführung anfangs bei 2–3 Wochen. Wenn man zusätzlich berücksichtigt, dass auch die Ergebnisse ausgewertet in Form umsetzbarer Artefakte[6] vorliegen müssen, ist schnell eine Dauer von 3 Wochen oder mehr erreicht.

Um die Geschwindigkeit und den Mehrwert von User Tests deutlich zu erhöhen, haben wir den inhaltlichen Umfang der User Tests stark reduziert, indem wir die zunehmend konkreten Fragestellung und Hypothesen gezielt in kleineren Nutzungsszenarien testeten. Darüber hinaus haben sich für uns „Remote User Tests“ bewährt, um den organisatorischen Aufwand deutlich zu reduzieren und Teammitglieder besser einzubeziehen. Dieser Ansatz ermöglicht neben einer Live-Übertragung des Tests auch die zeitliche Entzerrung der Termine – Testzeiten können flexibel geplant werden. Das erleichtert die Einbindung der Kollegen aus den Entwicklungsteams und des UX Chapters. Die breite Teilnahme an der Beobachtung erhöht anschließend die Akzeptanz der Ergebnisse im Projekt (Nielsen, J., 2010). Unser technisches Setup funktioniert in etwa wie folgt:

- Skype wird mit Screensharing und Videoübertragung aktiviert, um mit der Testperson zu kommunizieren und den Bildschirm der Testperson sehen zu können
- Während des Interviews wird der Link zum gesicherten Server mit dem Prototyp über den Chat geschickt
- Eine Instanz von Skype for Business[7] mit aktiviertem Screensharing wird genutzt, um den Skype-Chat mit der Testperson live an alle Beobachter zu übertragen
- Bildschirmaufzeichnung (z.B. per QuickTime) wird gestartet, um den Chat mit der Testperson in hoher Qualität lokal aufzunehmen und entscheidende Szenen im Nachhinein bei Bedarf auswerten zu können

Vor Beginn des Tests werden die Probanden gefragt, ob sie mit der Aufzeichnung und Übertragung einverstanden sind. Technische Schwierigkeiten haben sich bisher dabei nicht ergeben. Die Akquise der Probanden erfolgte über externe Agenturen, die auch "Remote" Probanden zur Verfügung stellen können.

Vorteile sind neben reduziertem Aufwand und schnelleren Ergebnissen, eine deutlich bessere Einbindung des Teams. Dieses Vorgehen eignet sich besonders für interaktive Prototypen, die

---

6 Im unserem konkreten Anwendungsfall sollten dies geschätzte User Stories sein

7 Wir verwenden gleichzeitig „Skype“ und „Skype for Business“ unter macOS, um zwei Live-Kanäle aufzubauen.

auf der Hardware der Probanden lauffähig sind. Als Alternative können Erkenntnisse aber auch aus dem Feedback zu einer Demonstration gewonnen werden.

3.3 Vom UX Design über ein Design-System zur Build Pipeline

Die wirkliche Nutzererfahrung entsteht erst bei der tatsächlichen Verwendung eines Produktes. Bei digitalen Produkten ist daher von entscheidender Bedeutung, was von einem angedachten UX Design auch tatsächlich in den Betrieb übernommen wurde. Aus vielen gängigen Tools, die UX Designer verwenden (Adobe Cloud, Sketch, etc.), lässt sich Design leicht in Form von statischen Styleguides spezifizieren. Diese haben sich jedoch in Zeiten von dynamischen Layouts, Gestensteuerung und animierten Übergängen nur als bedingt ausreichend herausgestellt. Diese Schwierigkeit verschärft sich, wenn Änderungen am Design in sehr kurzen Zeitabständen benötigt werden, was bei agiler Vorgehensweise eher die Regel als die Ausnahme ist.

Deshalb war die Zielsetzung für das UX Chapter, Übergaben von Design Artefakten zu vermeiden, um kontinuierlich Änderungen vornehmen zu können und so den Aufwand bei Teamkollegen möglichst gering zu halten.

3.3.1 Design Systeme

Einige haben bereits über die Verbreitung von Design-Systemen berichtet (Kholmatova, A., 2017). Vereinfacht ist ein Design-System die Idee, das Design der Benutzeroberflächen (User Interface Design, oder UI Design) in Form von Code zu spezifizieren und bereitzustellen. Dies hat den Vorteil, dass die immer dynamischer werdenden User Interfaces weitestgehend vollständig spezifiziert werden können. Ein Design System erleichtert die Integration des UI Designs in die Betriebsumgebung durch Frontend Entwickler und verringert Reibungsverluste gegenüber statischen Styleguides[8]. Eine weitere Idee bei einem Design System ist, dass auch das Design System wiederum als Produkt begriffen werden kann:

“A Design System isn’t a Project. It’s a Product, Serving Products” (Curtis, N., 2016)

Die Zielgruppe sind in diesem Fall die Softwareentwickler im Team. Es stellt sich daher die Frage, wie man den Entwicklern UI Design in Form von UI Komponenten optimal liefert, damit sie diese auch Verwendung finden. Wie tief ein Design System in die Betriebsumgebung eingebunden wird, lässt sich in drei Stufen einteilen:

Stufe	Beschreibung
1. Lebender Styleguide	Erleichtert die Übersetzung und Integration in die Zieltechnologie der Software, erzeugt dazu aber zusätzlichen Aufwand für die Entwickler
2. Design System in Zieltechnologie	Durch die Verwendung der gleichen Technologie verringert sich der Aufwand für die Integration bei den Entwicklern

[8] Da dynamisches Verhalten ausprobiert werden kann, hat der Entwickler eine bessere Vorgabe, was unserer Erfahrung dazu führt, dass weniger Abweichungen oder vergessene Details im Ergebnis zu finden sind.

Stufe	Beschreibung
3. Integrierte Build Pipeline	Direkte Zulieferung in Richtung Betrieb. Eine automatisierte Integration bedeutet praktisch keinen Aufwand für Entwickler

In dem Projekt haben wir alle Stufen zeitweise verwendet bzw. noch im Einsatz. Am effektivsten hat sich für uns Stufe 3 herausgestellt. Hier sind UI Frameworks, die eine komponentenbasierte Entwicklung erlauben[9] von großem Vorteil, weil sie eine technische Isolierung einzelner Bereiche fördern und somit auch punktuelle Anpassungen ermöglichen. Konkret bedeutet Stufe 3:

- UI Komponenten werden in Zieltechnologie vom UX Chapter erstellt.
- Texte in Sprachdateien und Grafiken können direkt durch den UX Designer im Repository (Versionsverwaltungssystem der Softwareentwickler) geändert werden.
- Eine vorhandene DevOps Struktur (z.B. eine Build-Pipeline[10] mit einer Entwicklungsumgebung) und automatisierte Layout Tests ermöglichen dem UX Designer, seine Änderungen ohne Risiko durchzuführen und das Ergebnis selber prüfen zu können.

Die Vorteile sind verringerter Aufwand (Entwickler müssen weniger integrieren und der UX Designer weniger kontrollieren) und eine vielseitigere Auslastung des UX Designers, was in einem interdisziplinären Team die Aufgabenverteilung erleichtert.

Dies erfordert allerdings die Bereitschaft der UX Designer, sich mit der Technologie zu beschäftigen und entsprechendes Wissen aufzubauen. Ebenso erfordert es von den Entwicklern die Bereitschaft, die UX Designer technologisch dahingehend zu unterstützen, direkte Änderungen am Code selbst vorzunehmen. Das kann auch bedeuten auf unnötig komplizierte technische Strukturen zu verzichten.[11]

Natürlich gibt es Konzepte und Änderungen, die nicht direkt durch den UX Designer integriert werden können. Hier können die Ansätze des lebenden Styleguides (Stufe 1) und des Design Systems (Stufe 2) genutzt werden, um die Lösungen effizient zu artikulieren.

---

9 Z.B. React oder Vuejs

10 Eine Praktik der Continuous Integration, bei der Änderungen automatisiert in einer Entwicklungsumgebung integriert und getestet werden können. Bei ausgereiften Strukturen können erfolgreich getestete Änderungen auch automatisiert in Betrieb genommen werden (Continuous Deployment).

11 Wir haben z.B. die Erfahrung gemacht, dass Sass durchaus eine höhere Hürde bedeutet als reines CSS, aber bei komponenten-basierter Entwicklung kaum noch Vorteile bietet. Diese Einschätzung ist allerdings von den Teammitgliedern abhängig.

4 DesignOps

In Anlehnung an DevOps ist bereits der Begriff DesignOps oder DesOps entstanden und wird im Internet diskutiert, siehe http://www.designops.org oder http://desops.io. Die drei Säulen von DesignOps könnten nach Samir Dash als Kultur, Prozess und Eco-System beschrieben werden (Dash, 2018). Das deckt sich mit den beschriebenen Kapiteln 2.1, 2.2. und 2.3. Unserer Erfahrung nach ist es ebenso für eine effektive Einbindung wichtig, alle drei Bereiche zu betrachten.

Das wichtigste Ziel ist, dass die Arbeit von UX Designern einen möglichst großen Einfluss auf das reale Ergebnis hat. Denn wer User Experience gestalten möchte, sollte sich auch dafür interessieren, wie die Güte der tatsächlich erreichten UX in der breiten Nutzung ist. Der Branchenreport UX/Usability von 2017 (Tretter et al., 2017) zeigte, dass immerhin 69% der Beschäftigten in dem Bereich eine stärkere Einbindung in die Entwicklungsprozesse und 60% eine stärkere Anerkennung der Relevanz der Themen wünschen.

DevOps in Verbindung mit DesignOps bietet UX Designern die Möglichkeit, näher an die tatsächliche Nutzung der Produkte zu rücken und durchaus größeren Einfluss zu nehmen. Es erfordert aber im Gegenzug auch die Bereitschaft, sich mehr Wissen in technischen Bereichen anzueignen.

4.1 Kritik

Der Begriff DesignOps ist wie auch DevOps ein Meta-Begriff und bietet viel Raum für Interpretationen. Es ist fraglich, ob der Begriff hilfreich ist, wenn es darum geht, UX Designer besser in agile Teams einzubinden. Er kann sich aber eignen, um im IT Umfeld die grundlegende Motivation und Idee zu kommunizieren, UX Designer näher an Betriebsumgebungen anzugliedern. Darüber hinaus ist von einem inflationären Gebrauch abzuraten, da die konkreten Ausprägungen dieser Entwicklung wenig gefestigt sind und noch individuell für jedes Team aufgesetzt und angepasst werden müssen. Es bleibt abzuwarten, ob der Begriff DesignOps an Bedeutung gewinnt und welche Ansätze, Methoden und Praktiken sich entlang des Begriffs formieren.

4.2 Operationalisierbarkeit

Die vorgeschlagenen Tools und Methoden haben in einzelnen Projekten für unsere Kunden eine spürbare Verbesserung gezeigt. Für eine Verallgemeinerung müssten diese jedoch noch genauer evaluiert werden, da wir nicht ausschließen können, dass auch noch viele andere Faktoren einen wichtigen Einfluss haben, die wir in dem Kontext bisher nicht betrachten konnten.

5 Zusammenfassung

Aufgrund unserer Erfahrung und Empfehlungen anderer Beiträge vertreten wir die These, dass UX Designer proaktiv auf technisch aufgestellte agile Teams zugehen müssen. Dazu sollten

sie die Bereitschaft besitzen, sich stärker mit Technologie zu beschäftigen und technologisches Wissen aufzubauen, um einfacher in agile Teams integriert werden zu können. Dies sollte sie in die Lage versetzen, ihren Einfluss auf die tatsächliche UX eines Produktes zu erhöhen. Der Begriff DesignOps fasst die dazu notwendigen Überlegungen zusammen und könnte durchaus hilfreich sein, um auch auf Verständnis bei Teams zu stoßen, die bereits DevOps praktizieren.

Danksagung

Wir danken allen Kollegen von SprintEins, die in Diskussionen und in den Projekten indirekt zu diesem Beitrag beigetragen haben. Wir danken ebenfalls Markus Nacar für die Erstellung der Schaubilder (1) und (2).

Literaturverzeichnis

Curtis, N. (26.02.2016). *A Design System isn't a Project. It's a Product, Serving Products* [Blog post]. Abgerufen von https://medium.com/eightshapes-llc/a-design-system-isn-t-a-project-it-s-a-product-serving-products-74dcfffef935

Dash, S. (2018). *The DesOps Enterprise (Volume 1) – The Overview & Culture: Re-invent Your Organization* [E-book]. Abgerufen von https://www.amazon.de/DesOps-Enterprise-Overview-Re-invent-Organization-ebook/dp/B07DHVNW7C/

Flohre, T. (15.02.2016). *BizDevOps* [Blog post]. Abgerufen von https://blog.codecentric.de/2016/02/bizdevops/

Gotthelf, J., Seiden, J. (2016). *Lean UX: Designing Great Products With Agile Teams.* Sebastopol, CA: O'Reilly Media Inc.

Kholmatova A. (2017). *Design Systems* [E-book]. Abgerufen von https://www.smashingmagazine.com/design-systems-book/

Knieberg, H. (27.03.2014). *Spotify Engineering Culture* [Blog post]. Abgerufen von https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

Laubheimer, P (24.09.2017). *Agile is not easy for UX: (How to) Deal with it* [Article]. Abgerufen von https://www.nngroup.com/articles/agile-not-easy-ux/

Loranger, H., Laubheimer, P. (05.02.2017). *The State of UX Agile Development* [Article]. Abgerufen von https://www.nngroup.com/articles/state-ux-agile-development/

Nielsen, J. (04.11.2009). *Agile User Experience Projects* [Article]. Abgerufen von https://www.nngroup.com/articles/agile-user-experience-projects/

Nielsen, J. (24.05.2010). *Involving Stakeholders in User Testing* [Article]. Abgerufen von https://www.nngroup.com/articles/stakeholders-and-user-testing/

Rodden, K., Hutchinson, H., Fu, X. (2010). *Measuring the User Experience on a Large Scale: User Centered Metrics for Web Applications* [Research publication]. Abgerufen von https://ai.google/research/pubs/pub36299

Tretter, S. et al. (2017). Branchenreport UX/Usability 2017 [Report]. Abgerufen von https://www.germanupa.de/sites/default/files/public/content/2017/2017-11-07/2017upbranchenreport.pdf

Wikipedia (04.07.2018). *DevOps* [Wikipedia article]. Abgerufen von https://de.wikipedia.org/wiki/DevOps#Einführung_von_DevOps

Autoren

Florian Röder

Nach dem Studium der Informatik arbeitete Florian Röder an der Schnittstelle zwischen Design und Entwicklung. Seit ca. 14 Jahren ist er in den Bereichen Softwareentwicklung, User Centered Design & Prototyping für namhafte Großkunden tätig und arbeitete im Bereich UX bei Firmen wie frogdesign. Seit 2014 ist er als Domain Lead "Vision" bei SprintEins für den Bereich Produktvision und UX in der agilen Entwicklung zuständig.

Andreas Gartz

Andreas Gartz ist Master of Arts im Bereich Mediendesign und beschäftigt sich seit 16 Jahren mit der Konzeption und Umsetzung von Webanwendungen und mobile Apps. Mit technischer Expertise und unternehmerischer Sichtweise engagiert sich Andreas Gartz für den Ausbau von effektiven Schnittstellen zwischen Design, Management und Entwicklung. Bei SprintEins ist Andreas Gartz als Projektleiter und Interaction Designer im Einsatz.

Ivo Jacobs

Ivo Jacobs ist als UX Designer und Projektleiter bei SprintEins tätig. Durch seine langjährige Erfahrung als Designer & Entwickler hat er viele Designprozesse und -tätigkeiten bei SprintEins mitgestaltet und etabliert. Aktuell ist er als Chapter Lead UX für die teamübergreifenden Maßnahmen in einem agilen Kundenprojekt verantwortlich.