Erfahrungen bei der UX-getriebenen Entwicklung Privatsphäre-schonender Anwaltssoftware

Roland H. Steinegger
roland.steinegger@leguli.ch
leguli, Engineering Steinegger GmbH
Rheinfelden, Schweiz

Anna Zinßer
mail@ungleich-anders.de
ungleich anders | anna zinßer – freie Kreative, ux designerin
Karlsruhe, Deutschland

ABSTRACT

Sicherheits-Vorbehalte sind ein starker Grund für Anwälte und Kanzleien keine oder nur sehr eingeschränkt Soft-ware-as-a-Service (SaaS) zu nutzen. Die Erkenntnisse aus der Sicherheits-Forschung finden in bestehender Software wenig Einsatz, wodurch die Vorbehalte derzeit selten adressiert werden. An einem Praxisbeispiel wird gezeigt, wie Sicherheit und Datenschutz in einen UX-getriebenen agilen Entwicklungsprozess integriert werden. Die entstehende Software unterstützt die zentralen Prozesse einer Kanzlei bzw. eines Steuerberatungsbüros. Beispielsweise mit einer Ende-zu-Ende-Verschlüsselung, auch der Dokumente, soll Vertrauen geschaffen werden. Wir gehen auf das Konfliktfeld von Benutzbarkeit und Sicherheit im Rahmen der Cloud-Software ein, geben unsere Erfahrungen mit verschiedenen Werkzeugen wider und zeigen Methoden, die sich etabliert haben oder auf Grund schlechten Feedbacks verworfen wurden.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**; • **Software and its engineering → Software development methods**; • **Information systems** → *Web applications.*

KEYWORDS

Agile Unified Process, Benutzbarkeit, Sicherheit, Muster, Erfahrungsbericht

1 EINLEITUNG

Die Digitalisierung bei kleineren Kanzleien und Steuerberaterbüros ist bisher eher stockend voran gegangen. Ein Kernproblem ist dabei die Angst vor Cloud-Anwendungen und dem damit einhergehenden Kontrollverlust über die Daten, im Vergleich zu Software, die auf Servern in den Kanzleibüros betrieben wird. Gleichzeitig fordern die Kunden in zunehmendem Maße die Nutzung des digitalen ein, wollen ihre Steuerdokumente oder andere Unterlagen digital einreichen und sind nur noch bedingt gewillt dies analog zu tun. Sicherheit und Benutzbarkeit sind in diesem Umfeld treibende Faktoren für den Erfolg von Software.

Mensch und Computer 2021, Workshopband, Workshop on 7. Usable Security und Privacy Workshop

https://doi.org/10.18420/muc2021-mci-ws21-395

Dem Druck der MandantInnen wird derzeit auf zwei Arten nachgegeben. Einerseits wird den MandantInnen die Entscheidungen über das Sicherheitsniveau aufgebürdet, indem der Weg zur Kommunikation von diesen gewählt werden soll. So werden sicherheitskritische, personenbezogene Daten unverschlüsselt übertragen, beispielsweise per Mail, Cloud-Datenablagen, wie Dropbox, Google Drive sowie iCloud oder sogar per Whatsapp. Andererseits argumentieren die Kanzleien, dass es keine sichere SaaS-Software gibt und deshalb auch Software, die keine spezielle Maßnahmen zum Schutz von Daten umsetzt, eingesetzt werden darf. Hierzu sind nach der Argumentation nur ein Vertrag zur Datenverarbeitung mit dem SaaS-Anbieter mit bestimmten rechtlichen Zusicherungen und organisatorische Maßnahmen notwendig [12].

Die Mehrzahl der bestehenden Softwarelösungen für Kanzleien benötigt eine lokal installierte Anwendung, die mit einem Server, der üblicherweise mit einem Server in den Räumen der Kanzlei betrieben wird, kommuniziert. Externer Zugriff wird in diesem Szenario durch ein virtuelles Netzwerk (engl. virtual private network) ermöglicht. Diese Variante ist üblich, da die schützenswerten Kanzleidaten so nicht Dritten zugänglich sind. Daneben gibt es wenige SaaS-Angebote, die jedoch mit wenigen Ausnahmen, keine Ende-zu-Ende-Verschlüsselung anbieten. Die Daten sind somit zumindest für den Softwareanbieter einsehbar, was von Kanzleien teilweise bereits als Sicherheitsproblem angesehen wird. Dies ergab eine von uns durchgeführte Marktanalyse, bei der wir Lösungen aus Vergleichsportalen [4, 10, 25] gesammelt und deren öffentliche Informationen zusammengetragen. Zudem haben wir Einzelinterviews mit AnwältInnen in einer für ein Startup ausreichend repräsentativen Form durchgeführt.

Das Nachrüsten der bestehenden Software, die auf den Betrieb auf Servern der Kanzlei ausgelegt ist, auf ein akzeptables Sicherheitsniveau für eine öffentlich zugängliche SaaS-Angebot, scheint zu teuer oder zu komplex zu sein und es ist vielleicht aus unternehmenspolitischer Sicht nicht gewollt - das Problem wird jedenfalls nicht angegangen. Bei einer großen etablierten Software wird beispielsweise das Öffnen der Desktopanwendung als Remote-Desktop-Übertragung in den Browser als SaaS-Lösung angeboten; der Desktop ist somit im Browser zu sehen und kann bedient werden. Andere Anbieter mit Sicherheit wird nicht als Teil der Benutzererfahrung (engl. user experience, UX) gesehen und vernachlässigt, wie es bereits Steward et al. als allgemeines Problem identifiziert haben [21].

Diesem Problem stellen wir einen Softwareentwicklungsprozess entgegen, bei dem der Komplexität von Sicherheit und Benutzbarkeit durch ein agiles Vorgehen angegangen wird. Dabei setzen wir

auf bekannten Prozessmodellen auf und beschreiben das Vorgehen aus unserer Erfahrung bei der Entwicklung einer Software für Anwaltskanzleien als Startup von Beginn an. Die Einstellung der Prozessbeteiligten spielt hierbei eine große Rollen, weswegen wir ebenfalls auf das Spannungsfeld von Sicherheit und Benutzbarkeit für EndbenutzerInnen eingehen und den Nutzen für die UX hervorheben, wenn Sicherheit transparent gemacht wird.

2 BESCHREIBUNG DES SZENARIOS: SICHERE CLOUD-ANWALTSSOFTWARE

Damit die etablierten Methoden und Muster im Abschnitt 3 besser eingeordnet werden können, beschreiben wir vorab das Szenario. Im Fallbeispiel wird eine Software für AnwältInnen und TreuhänderInnen, die selbstständig oder in kleinen Teams arbeiten, entwickelt. Die Software soll deren Kernprozesse und insbesondere die Kommunikationen mit MandantInnen unterstützen. Dies umfasst die Verwaltung von Mandaten und Kontaktpersonen, die Planung von Vorgängen mit Aufgaben im Rahmen eines Mandats und den Austausch von Dokumenten und Nachrichten. Die Zusammenhänge der Konzepte zeigt Abbildung 1.

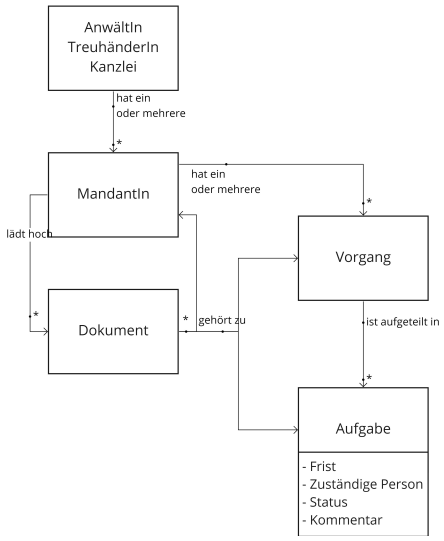

Figure 1: Domänenmodell der Konzepte der Anwendung

Die Anwendung ist primär per Browser zu bedienen und wird einer Vielzahl von Benutzern als Software-as-a-Service (SaaS) zur Verfügung gestellt. Es werden zwei Schwerpunkte bei der Entwicklung gelegt: einerseits sollen den Sicherheitsbedenken der Zielgruppe im Bezug auf eine SaaS-Lösung durch die Fokussierung auf Datensicherheit und Privatsphäre begegnet werden. Andererseits soll die Benutzbarkeit der Anwendung ein treibender Faktor bei der Etablierung der Software am Markt sein.

Die Anwendung wird nach dem im vorigen Kapitel beschriebenen Prozess iterativ entwickelt. Das bereits produktiv eingesetzte minimale Produkt umfasst eine leichtgewichtige Mandats-, Vorgangs- und Aufgabenverwaltung sowie das sichere Bereitstellen von Dokumenten von MandantInnen. Leichtgewichtig ist die Lösung in dem Sinne, dass nur das nötigste für die zwei Szenarien "Hochladen von Dokumenten durch MandantInnen" und "Gemeinsame Arbeitskoordination mit Aufgaben" umgesetzt sind. Es ist beispielsweise nicht einmal möglich erstellte Objekte (Mandate, Vorgänge, Aufgaben oder Dokumente) zu entfernen. Mit den ersten TestbenutzerInnen des Systems wird diese Basisfunktionalität im beschriebenen Prozess weiterentwickelt.

In Abbildung 2 ist links das Hauptmenü der Anwendung und rechts der sogenannte Schreibtisch zu sehen. Das Zusätzlich zum Schreibtisch verweist das Menü auf die Vorgänge, die Mandate und die Dokumentenablage. Im unteren Bereich ist zudem ein Feedback-Formular referenziert. Auf dem dargestellten Schreibtisch sollen die wichtigsten Informationen für den Start in den Tag zu finden sein. Hierzu gehören die (Aufgaben-) Fristen und Termine, sowie die zuletzt bearbeiteten Vorgänge und neue Dokumente, die beispielsweise von MandantInnen bereitgestellt wurden.

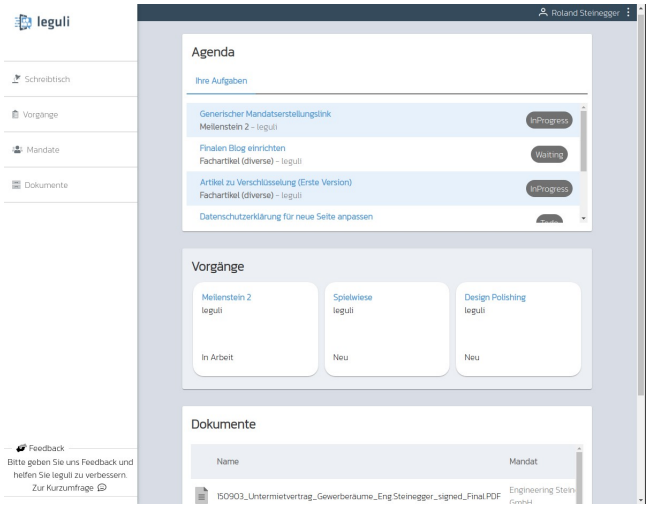

Figure 2: Anwendung mit Hauptmenü links und "Schreibitsch"-Ansicht rechts

3 INTEGRATION VON SICHERHEIT UND BENUTZBARKEIT IN DEN AGILEN ENTWICKLUNGSPROZESS

In diesem Kapitel geben wir einen Überblick über das Entwicklungsvorgehen im Projekt. Der grobe Entwicklungsprozess von der Feature-Idee über die Analyse einer sicheren und benutzerfreundlichen Lösung bis hin zu deren Umsetzung ist beschrieben.

3.1 Entwicklungsprozess im Spannungsfeld Sicherheit und Benutzbarkeit

Zwei gleichwertige Designprinzipien steuern unseren Entwicklungsprozess:

- Alle Daten von Nutzern sind ausschließlich für (berechtigte) Nutzer lesbar, auf dem Cloud Server werden nur verschlüsselte Daten abgelegt. Datensicherheit auf aktuellen technologischen Stand ist ein fester Bestandteil der Anwendung.
- Die Oberfläche ist einfach bedienbar, unterstützt eine zuverlässige und effiziente Aufgabenbewältigung und entspricht

den Erwartungen der Nutzer. Sicherheit und Bedienbarkeit stellen keinen Widerspruch dar, sondern gehen Hand in Hand.

Im Entwicklungsprozess, muss sichergestellt werden, dass diese nicht-funktionale Anforderungen einfließen und kontinuierlich überprüft werden. Werkzeuge dazu finden sich bereits in unterschiedlichen Ansätzen wie der Nutzer-zentrierten Gestaltung [14, 15], und der Agilen Software Entwicklung.

Die Herausforderung in der Praxis besteht darin diese Puzzleteile zusammenzusetzen, ohne dass sie sich gegenseitig aushebeln. So kann beispielsweise der Fokus auf Sicherheit und die strenge Reglementierung “Alle Daten sind verschlüsselt” dazu führen, dass auf einfach zu realisierende technische Konzepte zurückgegriffen wird, die jedoch nicht der Erwartungskonformität der Nutzer entsprechen. Oder andersherum, um den Nutzern die gewohnten Interaktionsmuster zur Verfügung zu stellen, werden Kompromisse zu Lasten der Sicherheit der Anwendung eingegangen.

Diese Spannungsfelder werden daher innerhalb des Entwicklungsprozess gemeinsam mit EndbenutzerInnen aufgelöst. Dieses nutzer-partizipative Vorgehen führt darüber hinaus dazu, dass das Thema “Sicherheit” nicht länger als technische Restriktion und "Innovationsbremse" wahrgenommen wird, sondern durchaus als zusätzlicher Innovationsfaktor gedacht werden kann. In dem Innovationsmodell von IDEO (2012) “three lenses of innovation” wird die Schnittmenge aus Machbarkeit (engl. Feasibility), Attraktivität (engl. Desirability) und Rentabilität (engl. Viability) als sogenannter “Sweet Spot" der Innovation betrachtet. Ergänzt um den Faktor der Sicherheit ergeben sich hier neue Spannungsfelder, aber auch Raum für neue Lösungsideen (siehe Abbildung 3).

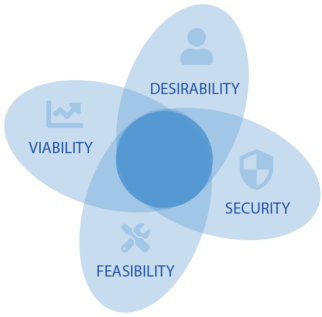

Figure 3: IDEOs "Three Lenses of Innovation" ergänzt um den Faktor Sicherheit. (https://designthinking.ideo.com/).

3.2 Der Entwicklungsprozess

Um das Spannungsfeld zwischen Benutzbarkeit und Sicherheit in der Interaktion mit EndbenutzerInnen aufzulösen wird auf einen strukturierten Entwicklungsprozess mit einem etablierten Werkzeugkasten und schnellen Feedback-Zyklen samt starken Fokus auf die Wünsche der Endbenutzer gesetzt. Wir kombinieren hierfür den Agile Unified Process (AUP) nach Edeki et al. [5] als strukturgebenden Anteil mit Lean-Software-Development-Prinzipien (LSDP) [17], die auf die Benutzbarkeit abzielen. Das Softwareentwicklungsteam ist dabei funktions- und themenübergreifend (engl. cross-functional) aufgestellt und es ist immer mindestens eine Person mit UX- und eine Person mit Sicherheits-Expertise im Team.

3.2.1 Prozessübersicht. In AUP werden in größeren Iterationen neue Versionen entwickelt, die üblicherweise in einer Abnahmeumgebung (engl. staging environment) in Betrieb genommen werden und in regelmäßigen Abständen in die Produktivumgebung übernommen werden. Der Softwareentwicklungsprozess wird dabei in in vier Phasen eingeteilt:

- `Auftaktphase` in der der Rahmen hinsichtlich neuer oder zu ändernder Funktionalität für die nächste Version gesteckt wird,
- `Ausarbeitungsphase` in der die zuvor ausgewählten Anforderungen verfeinert werden in Richtung der schlussendlichen Umsetzung,
- `Konstruktionsphase` in der die Hauptarbeit bei der Umsetzung der Anforderungen liegt und abschließend
- `Überführungsphase` in der die fertige Version ausgerollt und somit in der Abnahme- und/oder Produktivumgebung verfügbar gemacht wird.

Um das zuvor diskutierte Spannungsfeld in diese Phase einfließen zu lassen, gehen wir insbesondere auf die verwendeten Artefakte und Werkzeuge in der Auftakt- und Ausarbeitungsphase, wie in Abbildung 4 dargestellt, ein. In diesen Phasen haben die beiden nicht-funktionalen Anforderungen, Sicherheit und Benutzbarkeit, aus unserer Erfahrung heraus den größten Einfluss.

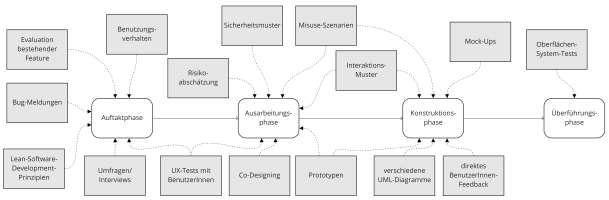

Figure 4: Übersicht der Phasen und wichtigsten Artefakte je Versionsiteration.

AUP zeigt den groben Rahmen für die Softwareentwicklung auf und behandelt verschiedenen Arbeitsprozesse (engl. workflows), die während der Phasen durchgeführt werden (auf diese gehen wir im Folgenden nicht näher ein). Allerdings lässt der AUP offen, mit welchen Werkzeugen die Durchführung sichergestellt wird. Wir setzen deshalb auf eine digitale Tafel in der Feature auf “Zetteln” zur Erfassung und Nachverfolgung festgehalten werden (siehe Abbildung 5), vergleichbar mit einem Kanban-Tafel (engl. kanban board) [3]. Auf dem “Zettel” eines Features wird das Feature beschrieben, werden Erkenntnisse, die bei der Ausgestaltung des Features gewonnen wurden, protokolliert, es wird auf Artefakte, die im Bezug mit dem Feature stehen (Mock-Ups, Prototypen, Architekturentwürfe etc.), verwiesen und die verantwortliche Person ist zu erkennen. Das Werkzeug der Kanban-Tafel wird vor allem auch zur Steuerung des Flusses genutzt, das heißt in den einzelnen Status-Spalten der Tafel dürfen sich nur eine bestimmte Zahl von Zetteln befinden.

Eine Tafel bildet dabei alles notwendige für die Fertigstellung einer Version im Sinne des AUP ab, wobei fertige Feature auch bereits nach Fertigstellung ausgeliefert werden können. Dies soll

möglichst schnelle Feedback-Zyklen ermöglichen, wodurch das Feature noch in derselben Version nachgebessert werden kann. Die Versionen sollen lediglich einen groben Rahmen geben und nicht einschränken.

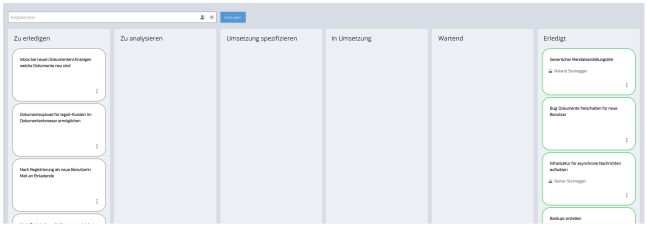

Figure 5: Tafel mit Spalten für die verschiedenen Status der Features und Bugs vergleichbar mit Kanban-Tafel.

3.2.2 Eingesetzte Werkzeuge und Erfahrungen. Die Einflüsse auf die *Auftaktphase* sind breit gefächert. Das Entwicklungsteam entscheidet gemeinsam mit den Produktverantwortlichen, welche Feature in der nächsten Version umgesetzt werden. Das Ergebnis dieser Phase sind grobe Feature-Beschreibungen und Ideen, die in einer Feature-Sammlung gesammelt und ergänzt werden. Sicherheit spielt in dieser Phase eine untergeordnete Rolle.

- `Evaluation bestehender Feature`: Gemeinsam mit BenutzerInnen werden die Feature der aktuellen Version evaluiert. Hierzu verwenden wir beispielsweise Fragebögen in der Anwendung. Für einen kleinen Teil unserer BenutzerInnen ist dies ein stark genutzter Weg Feedback zu geben, wobei sich dies derzeit primär auf TestbenutzerInnen (von uns Early Adopter genannt) bezieht, die einen kostenlosen Zugang zur Anwendung erhalten.
- `Benutzungsverhalten`: Die Häufigkeit der Benutzung von Features wird als Metrik herangezogen. In unserem Szenario hat dies zumindest auf Probleme bei der Sichtbarkeit eines Features für BenutzerInnen hingewiesen. Durch eine prominentere Platzierung konnten wir BenutzerInnen zum Feature führen.
- `Bug-Meldungen`: Die BenutzerInnen melden uns Bugs, beispielsweise per Support-Anfrage. Wir versuchen hier im Team abzuschätzen, wie groß das verursachte Problem ist und priorisieren Bugs auch direkt in die aktuelle Iteration mit ein.
- `Lean-Software-Development-Prinzipien`: Zu den hier relevanten Prinzipien zählen “Entscheide spätestmöglich” und “Liefere so schnell wie möglich”. Diese Prinzipien führen bei unserem Projekt oft dazu, dass wir ein Feature zuerst leichtgewichtig umsetzen und Feedback der BenutzerInnen einholen. Falls dies nicht möglich ist, gehen wir direkt in eine Diskussion mit unseren BenutzerInnen mit Mock-Ups oder anderen Artefakten und holen Einschätzungen zu den Feature ein.
- `Umfragen/Interviews`: Unklarheiten oder Ideen etc. werden in Umfragen oder Interviewleitfäden verpackt und mit den (potentiellen) BenutzerInnen durchgeführt. Insbesondere zu Beginn der Startup-Unternehmung haben wir hierauf gesetzt. Öffentliche Umfragen verbreitet über Social Media hatten jedoch kaum Rücklauf. Wie in Abschnitt 2 beschrieben, nutzen wir zusätzlich ein Feedback-Formular mit wenigen Fragen (Stimmung, Durchgeführte Aktivität, Probleme, Positives) zur aktuellen Arbeitssitzung; hierüber erhielten wir vereinzelt nützliches Feedback. Interviews mit einzelnen BenutzerInnen waren hingegen sehr ertragreich und liefern dem Entwicklungsteam bis heute eine wichtige Orientierung. Bei in der Fachlichkeit unerfahrenen Teams ist es aus unserer Sicht unabdingbar hier zu investieren. Durch mehrere längere Interviews des Entwicklungsteams mit BenutzerInnen wurde in der frühen Phase ein gewisses Fachwissen aufgebaut.
- `UX-Tests mit BenutzerInnen`: Mit den BenutzerInnen werden Szenarien durchgespielt, beispielsweise mit klickbaren Oberflächen ohne dahinter liegende Funktionalität oder rein mit Mock-Ups, durch die sie geführt werden. Dies ist für uns insbesondere dann eine wichtige Wahl, wenn wir bei einem Feature Uneinigkeit haben, wie es den Benutzenden den größtmöglichen Nutzen liefert. Die Diskussion und Entscheidung kann so einfach ausgelagert werden. Mit dem Testbenutzerinnen haben wir dies alle 1-2 Monate durchgeführt.

In der *Ausarbeitungsphase* haben *Sicherheit* und *Benutzbarkeit* in unserem Prozess den Haupteinfluss auf die resultierende Softwarelösung. Dementsprechend finden hier auch Methoden Anwendung, bei denen die Themen miteinander in Berührung kommen:

- `Risiko-Abschätzung`: Bei potentiellen Schwachstellen wird das Risiko festgestellt, d.h. es wird beispielsweise für eine Schwachstelle deren negative Folgen und die Eintrittswahrscheinlichkeit abgeschätzt. Die Risiko-Abschätzung hatte in unserem Projekt bereits Einfluss auf Feature-Umsetzungen. Vorteil hierbei ist auch, dass eine solche Abschätzung im Rahmen mancher Zertifizierungen, bei uns der ISO-27001-Zertifizierung, anfallen und vom Team so als Einfluss genutzt werden kann.
- `Misuse-Szenarien`: Es werden Szenarien beschrieben, die eine ungewollte Benutzung der Anwendung skizzieren. Dies können sowohl gezielte Angriffe als auch unbeabsichtigte Handlungen sein. [19] Wir nutzen diese Methode in gemeinsamen Brainstormings zu neuen Features. In jeder solchen Sitzung identifizieren wir für Features, die nicht nur marginale Änderungen darstellen, mindestens eins, in der Regel drei bis vier, missbräuchliche Szenarien. Bei der Methode bringen sich in unserem Entwicklungsteam sowohl die Sicherheits- und UX-ExpertInnen als auch die Personen im Team mit anderen Schwerpunkten ein.
  Beispielhaft diskutierten wir das Feature "Passwort zurücksetzen". Vom Team wurde folgende Szenarien aufgeführt: "absichtliche Aussperren von Benutzern durch Zurücksetzen deren Passworts", woraufhin die Idee eines zweiten Faktors bei der Änderung aufkam. Darauf folgend die Szenarien "Zurücksetzen des Passworts zu einem von einem Angreifer gewünschten (bspw. wegen Zugangs zum E-Mail-Account)", "das zurücksetzen des Passworts auf ein Älteres bei einer Support-Anfrage, bei der missbräuchliche Nutzung vorgeworfen wird und der Angreifer das alte Passwort kennt", "das Vertippen beim Eingeben des neuen Passworts" und

weitere. Die Lösung wurde hierdurch umfangreicher, aber auch sicherer. Dem Support wurden Anweisungen zum Umgang mit den genannten Anfragen mitgegeben, der Benutzer wird per Mail auf Änderungen am Account hingewiesen. Durch den zusätzlichen Fokus der UX-Expertin wurde in der Mail auch direkt auf den Weg zur Sperrung des Kontos hingewiesen etc.

- `Sicherheitsmuster`, `Interaktions-Muster`, `Usable Security Patterns`: Softwaremuster sind etablierte und verbreitete Lösungen mit Stärken und Schwächen für wiederkehrende Probleme und basierend auf den Arbeiten von Alexander [1]. *Sicherheitsmuster* beziehen sich speziell auf Probleme mit Sicherheitsbezug [26]. Es existieren über 550 Sicherheitsmuster [24], allerdings ist die Auffindbarkeit ein fortwährendes Forschungsproblem [16, 24]. Sofern Sicherheitsmuster für passende Probleme eingesetzt werden, erhöhen diese das Sicherheitsniveau der Anwendung [7]. Wegen der Schwierigkeiten beim Auffinden ist ein breites Wissen über Sicherheitsmuster notwendig, weswegen diese Muster nur in Teilen in unserem Projekt zum Einsatz kommen.
  Für die Benutzerinteraktion mit Bezug zu Sicherheit ist die Menge der Sicherheitsmuster in der Forschung überschaubar gering und die Benutzbarkeit wird in wenigen Fällen betrachtet [24]. Ein Beispiel für ein kombiniertes Muster ist "Voller Zugriff mit Fehlern" [18], welches besagt, dass der Benutzerin alle Funktionalität einer Anwendung in der Oberfläche angezeigt werden soll und, sofern sie keinen Zugriff hat, soll ihr eine Fehlermeldung angezeigt werden. Hinsichtlich der Benutzbarkeit widerspricht dies beispielsweise dem Prinzip, dass die Benutzererwartungen möglichst erfüllt werden sollen (engl. least surprise) [8].
  `Interaktions-Muster` sind etablierte Lösungen bei der Interaktion von BenutzerInnen mit Benutzungsoberflächen. Die Verwendung von Mustern und wiederkehrenden Lösungen bei der Interaktion mit Benutzern ist viel weiter verbreitet als im Sicherheitsbereich, was die Menge an öffentlichen Sammlungen von Praktizierenden zeigt [2, 11, 13, 23]. In den genannten Sammlungen wird aber nicht auf eine formale Beschreibung geachtet und der Begriff des Musters wird nicht immer im Sinne einer "guten" Lösung verstanden. Diese Sammlungen dienen in unserem Projekt häufig als Diskussionsgrundlage im Team und werden anschließend beispielsweise mit Misuse-Szenarien überprüft.
  Darüber hinaus gibt es eine Reihe von speziellen Mustern für benutzbare und sichere Lösungen [9]. Die Anwendbarkeit dieser Muster in unserem Projekt überprüfen wir derzeit noch.
- `UX-Tests mit BenutzerInnen`: Beschreibung, siehe vorheriger Abschnitt. In der Ausarbeitungsphase sind diese Tests noch einmal wichtiger aus unserer Erfahrung. Dies ist insbesondere deshalb der Fall, weil wir bei den Lösungen in dieser Phase zwischen Benutzbarkeit und Sicherheit abwägen müssen. Für uns war nicht klar, ob BenutzerInnen verstehen, dass das Versenden eines Dokumentenupload-Links mit dem eigenen Mail-Programm sicherer ist, da die Daten nicht dem Anbieter preisgegebenen werden, als wenn die Mail vom Anbieter versendet wird und somit die persönlichen Daten an diesen weitergegeben werden. Wir boten das Feature mit dem Versand der Mail im eigenen Browser an und fragten im Anschluss. Keiner/m BenutzerIn war dies klar, weswegen wir Hinweistexte ergänzten.
- `Co-Designing`: Beim Co-Designing werden Stakeholder mit in die Benutzungs-Oberflächengestaltung mit einbezogen und können ihre Ideen mit einbringen. Für uns sind Co-Designing-Sitzungen vor allem ein Mittel um ein gemeinsames Verständnis im Entwicklungsteam zu schaffen. Die Feature werden in Oberflächen übersetzt. Es kommen Design-Werkzeuge zum Einsatz, mit denen rasch verschiedene Varianten zusammengestellt werden können. Für die gemeinsame Arbeit im Team ist das bei uns ein unverzichtbares Werkzeug. Auf diese Weise können auch Misuse-Szenarien erkannt werden, da die Probleme durch die Mock-Ups fassbar werden. Im Team gab es jede Woche eine 2-stündige Co-Designing-Sitzung.
- `Prototypen`: Eine bestimmte Eigenschaft eines Feature wird umgesetzt, um diese Eigenschaft selbst zu erfahren oder mit BenutzerInnen zu erkunden. Dies kann beispielsweise ein UX-Prototyp sein, der primär auf die Erfahrung der Benutzenden abzielt, aber keinerlei Funktionalität umsetzt. Es kann aber beispielsweise auch auch ein funktionaler Prototyp sein, der die Oberfläche nur so weit wie notwendig umsetzt, um eine Funktionalität zu testen. Beides setzen wir ein und diskutieren verschiedene Varianten mit unseren BenutzerInnen oder nehmen sie in Co-Designing-Sitzungen mit und schärfen das Feature dort weiter.

Die Methoden in der Konstruktions- und Überführungsphase dienen primär der Umsetzung dessen, was in der Ausarbeitungsphase festgelegt wurde. Die wichtigen Entscheidungen sind in der Regel bereits in dieser vorherigen Phase gefällt worden. In wenigen Fällen treten bei der Feature-Umsetzung noch Probleme auf, die die beschriebene Lösung in Frage stellen. In diesen Fällen können Feature auch wieder in die Ausarbeitungsphase zurückversetzt werden oder werden womöglich aus der Version entfernt.

Ansonsten werden die Feature-Umsetzungen weiter spezifiziert. Mock-Ups und UML-Diagramme sind entweder in der vorherigen Phase bereits entstanden und werden genutzt oder sie werden erstellt, weiter verfeinert etc. Diese Artefakte dienen der schlussendlichen Implementierung als Vorlage. Bei Unklarheiten wird von den BenuterzInnen direktes Feedback eingeholt.

Wichtigster Punkt für die Thematik Sicherheit und Benutzbarkeit sind allerdings die automatiserten Akzeptanztests, die die Feature in der Abnahmeumgebung aus Sicht der Benutzenden testet und sicherstellt, dass bestehende Funktionalität nicht beeinflusst wird. In unserer Anwendung ist das beispielsweise das Testen, ob bestehende, verschlüsselte Daten nach der Versionsänderung weiterhin zugegriffen werden können. Hinzu kommen Negativtests, dass BenutzerInnen nicht plötzlich Zugriff auf anderen Daten haben etc. Die Tests sind bei uns ein zentraler Bestandteil, um den BenutzerInnen ein Gefühl von Sicherheit durch eine stabil laufende Software zu geben.

3.2.3 Erfahrungsbericht und Diskussion der Skalierbarkeit. Für uns hat sich das beschriebene Vorgehen in einem kleinen Startup-Kontext

bewährt. Es konnte innerhalb eines halben Jahres ein Produkt, dass bestimmte BenutzerInnenbedürfnisse (siehe Abschnitt 2) erfüllt, entwickelt werden. Dies bestätigen die durchgeführten Interviews und Umfragen. Es konnten eine Reihe von Sicherheitsthemen angegangen und Sicherheitsmuster umgesetzt werden, wobei dies auf die Kenntnisse der Sicherheitsexperten zurückzuführen ist und der Bezug zur Benutzbarkeit fehlt; hierzu zählen beispielsweise die rollenbasierte Zugriffskontrolle (engl. rollenbasierte Zugriffskontrolle) [18], Entmilitarizierte Zone (engl. demilitarized zone) [18] oder die Risikobasierte Authentifizierung (engl. risk-based authenticator) [20]. Das Verständnis und die Aufmerksamkeit für Sicherheit hat sich durch das frühe einbeziehen von Experten im Team gesteigert. Auch beim Thema Benutzbarkeit konnte durch die gemeinsame Arbeit mit Expertin im Team ein gemeinsames Verständnis aufgebaut werden und es wurden verschiedene etablierte Methoden aus den im vorigen Abschnitt 3.2.2 eingeführten "Muster"-Sammlungen eingesetzt. Das Feedback der BenutzerInnen aus Umfragen und Interviews (siehe Abschnitt 3.2.2) ist sowohl bei der gefühlten Sicherheit und zum Thema Benutzbarkeit positiv.

Inwieweit ein ähnlicher Prozess in anderen Kontexten umgesetzt werden kann ist schwer abzuschätzen. AUP macht relativ wenige Vorgaben beispielsweise im Vergleich zu anderen agilen Methoden wie Scrum [22], das heißt es kommt auch auf die Erfahrung der Beteiligten an. In unserem Fallbeispiel waren keine unerfahrenen Personen im Entwicklungsteam beteiligt und es konnte sich aufeinander verlassen werden. Wenn auch eine Team Majurity im Sinne von Elrod et al. [6], die sich auf längere Zusammenarbeit miteinander bezieht, nicht gegeben war. Trotzdem ist die Majurity der Einzelnen ein wichtiger Punkt in einem Prozess, der Eigenverantwortung einfordert. Die Skalierung des Prozesses auf größere Projekte bedarf weiterer Überlegungen hinsichtlich der Absprachen zwischen den Teams, was beispielsweise die Funktionalität und die Softwarearchitektur angeht.

4 ZUSAMMENFASSUNG

In diesem Artikel haben wir unsere Erfahrungen bei der Umsetzung des Agile Unified Process in Kombination mit Lean Software Development und insbesondere den eingesetzten Methoden zur Erreichung von Benutzbarkeits- und Sicherheitszielen widergespiegelt. Demnach ist der Einsatz in unserem Startup-Szenario und einem erfahrenen funktions- und themenübergreifenden Team positiv. Wir haben zudem gezeigt, wie wir mit verschiedene Methoden aus dem Sicherheits- und UX-Bereich Muster und Best Practices aus den verschiedenen Bereichen in einem komplexen Umfeld angewendet haben.

Die Übertragung auf einen größeren Kontext und mehr Teams, sowie mit unerfahrenen Beteiligten muss weiterhin evaluiert werden. Die Fortführung des Projekts unter der Zunahme der Komplexität der Anwendung und einem größeren Team, das auch mit unerfahrenen Entwicklern bestückt wird, wird Herausforderungen mit sich bringen, die in aufbauenden Arbeiten betrachtet wird.

Sicherheit als weiteren Innovationsfaktor, der zudem von BenutzerInnen zunehmend eingefordert wird, wurde ebenfalls betrachtet. Die UX einer Anwendung misst sich nach unserer Erfahrung auch daran, wie die gefühlte Sicherheit ist; dies ergaben die durchgeführten Umfragen. In diesem Bereich scheint es lohnend zu sein, eine leicht zugängliche Sammlung von Mustern kombiniert aus Sicherheits- und UX-Mustern aufzubauen, die diesem Bedürfnis gerecht wird. Der für die Benutzenden transparente Umgebung mit Sicherheitsmechanismen in der Anwendung hat jedenfalls ein positives Feedback eingebracht.

ACKNOWLEDGMENTS

Wir danken allen die direkt oder indirekt zu diesem Artikel beigetragen haben, beispielsweise durch ihr (BenutzerInnen-) Feedback bei der Entwicklung der Software.

REFERENCES

[1] Christopher Alexander. 1977. *A pattern language: towns, buildings, construction.* Oxford university press.
[2] Floriz Andrew McCarthy. 2021. *Little Big Details.* Retrieved 09. Juli 2021 from https://twitter.com/littlebigdetail
[3] N Arora. 2001. Kanban Guide: Demand Scheduling for Lean Manufacturing. *Add ValueConsulting Inc* (2001).
[4] Danilo Creutzburg. 2021. *Kanzleisoftware und -programme im Test und Vergleich - Top 10-Anbieter.* Retrieved 09. Juli 2021 from https://systemhaus.com/kanzleisoftware/
[5] Charles Edeki. 2013. Agile unified process. *International Journal of Computer Science* 1, 3 (2013), 13–17.
[6] P. David Elrod and Donald D. Tippett. 1999. An Empirical Study of the Relationship Between Team Performance and Team Maturity. *Engineering Management Journal* 11, 1 (1999), 7–14. https://doi.org/10.1080/10429247.1999.11415013 arXiv:https://doi.org/10.1080/10429247.1999.11415013
[7] Eduardo B. Fernandez, Nobukazu Yoshioka, and Hironori Washizaki. 2018. Evaluating the Degree of Security of a System Built Using Security Patterns. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (Hamburg, Germany) *(ARES 2018)*. Association for Computing Machinery, New York, NY, USA, Article 43, 8 pages. https://doi.org/10.1145/3230833.3232821
[8] Simson Garfinkel. 2005. *Design principles and patterns for computer systems that are simultaneously secure and usable.* Ph.D. Dissertation. Massachusetts Institute of Technology.
[9] Peter L Gorski, Emanuel von Zezschwitz, Luigi Lo Iacono, and Matthew Smith. 2019. On providing systematized access to consolidated principles, guidelines and patterns for usable security research and development†. *Journal of Cybersecurity* 5, 1 (12 2019). https://doi.org/10.1093/cybsec/tyz014 arXiv:https://academic.oup.com/cybersecurity/article-pdf/5/1/tyz014/31572414/tyz014.pdf tyz014.
[10] Sven Graf. 2021. *Anwaltssoftware - Die 13 besten Tools 2021 im Vergleich.* Retrieved 09. Juli 2021 from https://trusted.de/anwaltssoftware
[11] Gabriel Lewis Jane Portman. 2021. *UI patterns.* Retrieved 09. Juli 2021 from http://uipatterns.io/currency-input
[12] Christian Laux. 2021. *Cloud - Legal Aspects. Legal Innovation Konferenz, Vortragsreihe.* Retrieved 09. Juli 2021 from https://www.youtube.com/watch?v=OdJizFReYrQ
[13] Brian Lovin. 2021. *App dissection - Exploring the best interaction patterns, visual styles, and design decisions of well-known apps.* Retrieved 09. Juli 2021 from https://systemhaus.com/kanzleisoftware/
[14] Don Norman. 2013. *The design of everyday things: Revised and expanded edition.* Basic books.
[15] Donald A Norman and Stephen W Draper. 1986. User centered system design: New perspectives on human-computer interaction. (1986).
[16] Poonam Ponde, Shailaja Shirwaikar, and Christian Kreiner. 2016. An Analytical Study of Security Patterns. In *Proceedings of the 21st European Conference on Pattern Languages of Programs* (Kaufbeuren, Germany) *(EuroPlop '16)*. Association for Computing Machinery, New York, NY, USA, Article 33, 26 pages. https://doi.org/10.1145/3011784.3011821
[17] Mary Poppendieck and Tom Poppendieck. 2003. *Lean software development: an agile toolkit.* Addison-Wesley.
[18] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. 2013. *Security Patterns: Integrating Security and Systems Engineering.* Wiley. https://books.google.de/books?id=2ZwSAAAAQBAJ
[19] Guttorm Sindre and Andreas L Opdahl. 2001. Capturing security requirements through misuse cases. *NIK 2001, Norsk Informatikkonferanse 2001, http://www.nik. no/2001* (2001).
[20] Roland H Steinegger, Daniel Deckers, Pascal Giessler, and Sebastian Abeck. 2016. Risk-based authenticator for web applications. In *Proceedings of the 21st European Conference on Pattern Languages of Programs*. 1–11.
[21] Curtis Steward Jr, Luay A Wahsheh, Aftab Ahmad, Jonathan M Graham, Cheryl V Hinds, Aurelia T Williams, and Sandra J DeLoatch. 2012. Software security: The

dangerous afterthought. In *2012 Ninth International Conference on Information Technology-New Generations*. IEEE, 815–818.

[22] Jeff Sutherland and JJ Sutherland. 2014. *Scrum: the art of doing twice the work in half the time*. Currency.

[23] Anders Toxboe. 2021. *Design patterns*. Retrieved 09. Juli 2021 from http://ui-patterns.com/patterns

[24] Hironori Washizaki, Tian Xia, Natsumi Kamata, Yoshiaki Fukazawa, Hideyuki Kanuka, Takehisa Kato, Masayuki Yoshino, Takao Okubo, Shinpei Ogata, Haruhiko Kaiya, Atsuo Hazeyama, Takafumi Tanaka, Nobukazu Yoshioka, and G. Priyalakshmi. 2021. Systematic Literature Review of Security Pattern Research. *Information* 12, 1 (2021). https://doi.org/10.3390/info12010036

[25] Markus Weins. 2021. *Kanzleisoftware im Vergleich*. Retrieved 09. Juli 2021 from https://anwaltskanzleisoftware.de/vergleich-2/

[26] Joseph Yoder and Jeffrey Barcalow. 1997. Architectural patterns for enabling application security. In *Proceedings of the 4th Conference on Patterns Language of Programming (PLoP'97)*, Vol. 2. Citeseer, 30.