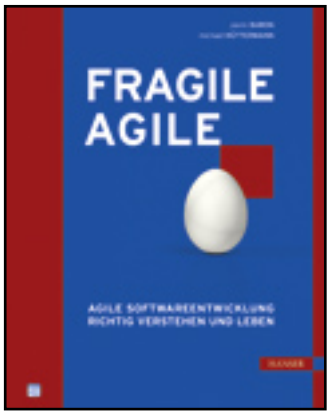
HANSER



Leseprobe

Pavlo Baron, Michael Hüttermann

Fragile Agile

Agile Softwareentwicklung richtig verstehen und leben

ISBN: 978-3-446-42258-2

Weitere Informationen oder Bestellungen unter

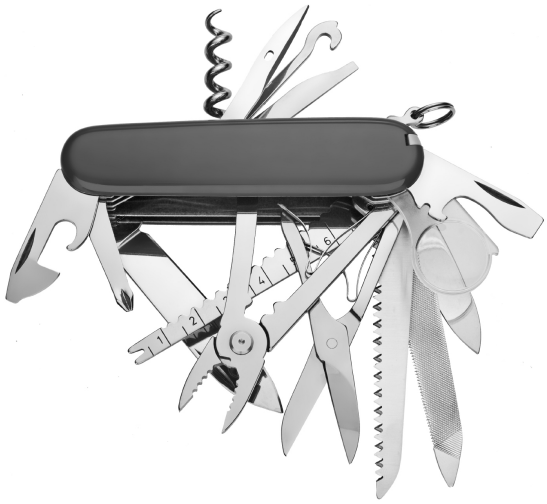
http://www.hanser.de/978-3-446-42258-2

sowie im Buchhandel.

„Nur wer sein Ziel kennt, findet den Weg."

Laotse, chinesischer Philosoph,
Begründer des Taoismus

4 Wechselnde Anforderungen



Quelle: ©iStockphoto.com/SklepSpozywczy

"Welcome changing requirements, even late in development.
Agile processes harness change for the customer's competitive advantage."

„Sei sogar spät während der Entwicklung offen für sich ändernde Anforderungen. Agile Prozesse machen sich Anforderungsänderungen zunutze, um den Wettbewerbsvorteil des Kunden zu erhöhen."

Anekdote

Eine hochschwangere Frau sagt zu ihrem Mann: „Du, ich will da was, aber ich bin mir noch nicht ganz sicher, was das ist."

Der Mann: „Was ist es Schatz? Du weißt, ich erfülle dir jeden Wunsch."

Frau: „Ich glaube, ich hätte gerne ein bisschen Hundekot."

Mann: „Was???"

Frau: „Ja genau! Hundekot!"

Mann: „Geht es dir nicht gut?"

Frau: „Ich will, ich will, ich will!" Bricht in Tränen aus.

Mann: „Na gut, sollst du haben." Geht und kehrt mit einer stinkenden Papiertüte nach Hause zurück.

Frau: „Hast du es?"

Mann: „Jaaa!"

Frau: „Zeig mal". Sieht in die Tüte und verzieht das Gesicht. „Pfui, das stinkt ja!"

Mann: „Was hast du denn erwartet? Ist doch Hundekot."

Frau: „Schmier es aufs Brot."

Mann „Was???"

Frau: „Tu es!" Bricht in Tränen aus.

Mann: „Ist ja gut!" Schmiert die stinkende Masse auf das Brot und hält es weit von seinem Gesicht weg in Richtung seiner Frau: „Da!"

Frau: „Probier's mal."

Mann: „Was??? Niemals! Du bist übergeschnappt, weit jenseits dessen, was sich eine schwangere Frau erlauben kann."

Die Frau wird extrem hysterisch und fängt an zu seufzen und sich vor Schmerzen an den Bauch zu fassen. Der Mann ist extrem erschreckt und sagt schließlich: „Gut, ich tu es!" und tut es tatsächlich, wobei er das Erbrechen kaum zurückhalten kann.

Frau skeptisch und nun völlig schmerzfrei: „Und? Wie schmeckt's?"

Mann um Atem ringend: „Wie Hundekot eben!"

Frau nach einer kurzen Denkpause: „Nah, will ich doch nicht."

In diesem Prinzip wird der Umgang mit Kundenanforderungen thematisiert. Anforderungen sind die Grundlage aller Aktivitäten. Sie motivieren die konkreten Arbeiten. Dies ist in allen Vorgehensmodellen so. Anforderungen müssen definiert und verabschiedet werden. Sie müssen stabil sein, aber nicht stabil bis

in die Ewigkeit. Die Welt dreht sich weiter, neue Erkenntnisse werden gewonnen, neue Marktsituationen tun sich auf, auch Anforderungen entwickeln sich weiter.

Zu wohldefinierten Zeiten soll es möglich sein, die verabschiedeten Anforderungen zu ändern, zu konkretisieren oder zugunsten anderer hintanzustellen. Durch Flexibilität beim Umgang mit den Anforderungen bleibt das Projekt manövrierbar, und der größtmögliche Nutzen für den Kunden bleibt auf dem Radar. Denn das ist, was zählt: den konkreten Nutzen für den Kunden zu optimieren, zu jeder Zeit.

Optimiere zu jeder Zeit den Nutzen für den Kunden!

Vermeiden Sie es tunlichst, die an Sie herangetragenen Anforderungen als in Stein gemeißelt zu verstehen – sie sind es nicht und veralten schneller, als die Tinte trocknet, mit der sie geschrieben sind. Geben Sie es auf, daran zu glauben, dass ein Softwareprojekt hinsichtlich der Anforderungen berechenbar und vorhersehbar ist, denn das führt nur dazu, dass der Aufwand für Änderungen (engl. „Changes“) den Initialaufwand um ein Mehrfaches übersteigt.

Versuchen Sie stattdessen, die Unsicherheit hinsichtlich der Anforderungen zu berücksichtigen. Vermeiden Sie unnötig frühe, harte, endgültige Entscheidungen – sei es im Design, im Code oder im Vertrag. Denn die Kunden sind genauso Menschen, und es gibt keinen Grund, sie dafür zu bestrafen, dass sie sich die künftige Lösung nicht auf dem Papier vorstellen können, sondern zunächst etwas benötigen, was sie anfassen und womit sie spielen können, um erst dann sagen zu können, wie sie es wirklich gerne hätten. Zudem erlauben Sie dem Markt, während der Entwicklung der Lösung Einfluss darauf zu nehmen, denn nahezu jede Branche ändert sich heutzutage fortlaufend. Der Markt und die Branche nehmen keine Rücksicht auf Sie und Ihre Kunden.

4.1 Die Bedeutung

Eine fundierte Anforderungsspezifikation ist vermutlich eins der drei wichtigsten Dinge in einem IT-Projekt – das zweite ist die dauerhaft sichergestellte Kaffeeversorgung, das dritte bleibt zur eigenen Entlastung an dieser Stelle frei. Die Anforderungsspezifikation ist das grundlegende Fundament eines IT-Projektes. Steht sie auf wackeligen Füßen, so wird alles andere niemals die notwendige Konstanz und Konsistenz erreichen. Nur wenn Sie die Wünsche des Kunden in Form von konkreten, priorisierten, wohlstrukturierten Anforderungen vorliegen haben, verfügen Sie über eine solide Grundlage für die Umsetzung. Dabei ist es zunächst mal unerheblich, wie Sie die Kundenwünsche extrahieren (zum Bei-

spiel per Interview oder Kano-Modell[1]) und wie Sie die Anforderungen beschreiben.

Anforderungen sollten über bestimmte Eigenschaften verfügen.

Unabhängig vom Vorgehensmodell sollten Anforderungen ohne Redundanzen, wohlstrukturiert, eindeutig, vollständig, korrekt, widerspruchsfrei, objektiv überprüfbar, realisierbar und verständlich definiert werden.

Anforderungen sind dabei funktional oder nicht-funktional. Funktionale Anforderungen beschreiben die vom System bereitzustellende Funktionalität, nichtfunktionale Anforderungen dagegen Faktoren wie Benutzerfreundlichkeit oder Skalierbarkeit, also wirklich kriegsentscheidende Dinge.

Anforderungen können grafisch (z. B. durch Anwendungsfallmodellierung), tabellarisch und/oder natürlich-sprachlich beschrieben werden. Als Ergebnis einer umfassenden Sammlung von Anforderungen entstehen gewöhnlich ein Fachkonzept und/oder ein Lastenheft. Für die ganz pragmatisch Denkenden reicht aber auch ein fundierter Eintrag in einem Ticketsystem oder letztendlich auch ein buntes Kärtchen an der Wand.

Aus einem Lastenheft kann die Entwicklung ein Pflichtenheft ableiten, aus einem Fachkonzept ein DV-Konzept, aus einem fachlichen Ticket können technische Tasks herunter gebrochen werden, aus bunten Kärtchen können andersfarbige Kärtchen für technische Tasks an die Wand gehängt werden.

Anforderungsmanagement ist für den Erfolg immens wichtig.

Eine Anforderung ist auch ein Mittel, um das System, insbesondere während der Umsetzung, selbst besser zu verstehen. Das System wird in kleine Teile zerlegt und beschrieben. Erst die so zugeordneten Verantwortlichkeiten, Akteure, Aktivitäten, Eingangs- und Ausgangsbedingungen beschreiben das System. Die Anforderung ist somit eine gemeinschaftliche Diskussionsgrundlage aller im Projekt vertretener Rollen, wie Kunde, Architekten, Tester und Entwickler. Die Stakeholder können aufbauend auf den Anforderungen ihre Arbeit tun (Beistellleistungen etc.) und ggf. darauf neben den Anforderungsdokumenten weitere Artefakte erstellen (wie z.B. ein Architekt Architekturdokumente erstellt).

Anforderungen werden nicht nur beschrieben, damit man sie beschreibt und damit einfach nur einen Schrank füllt. Das wäre dann die landläufig bekannte „Schrankware“. Ja, stimmt, das gibt es auch. Nein, Anforderungen sollen tatsächlich umgesetzt werden. Und was ist dann? Dann muss die Erfüllung der Anforderung validiert, also zu einer Zeit Ist und Soll gegenübergestellt werden. Die Erfüllung funktionaler Anforderungen lässt sich vergleichsweise einfach überprüfen, wenn denn jeweils geeignete Abnahmekriterien pro Anforderung defi-

[1] http://de.wikipedia.org/wiki/Kano-Modell

niert sind. Abnahmekriterien gehören zu den Anforderungen dazu und müssen gleichzeitig zu den Anforderungen beschrieben und verabschiedet werden.

Rekursionsstufe 1: Der Umgang mit Anforderungen hängt von Anforderungen ab

Wie konkret mit Anforderungen umgegangen werden soll, hängt von Anforderungen ab, die an das Anforderungsmanagement gestellt werden. Diese müssen im Grunde zunächst erhoben und bewertet werden, ehe eine konkrete Umsetzung erfolgt. Wie wollen Sie sonst feststellen, welcher Weg für Sie der richtige ist, den Sie für Ihre speziellen Projektbedürfnisse einschlagen wollen? Aus diesen Anforderungen kann sich beispielsweise ergeben, dass ein Ticketsystem als Werkzeug zum Anforderungsmanagement ausreicht.

Abnahmekriterien sind häufig in natürlicher Sprache verfasst und beinhalten die Ausgangssituation des Abnahmetests, das Ereignis, das zum zu validierenden Stand führende Ereignis und ein erwartetes, objektiv messbares Ergebnis. Diese Eigenschaft der „Interpretationslosigkeit“ darf nicht unterschätzt werden. Die Validierung von Anforderungen muss nämlich tatsächlich jederzeit objektiv überprüfbar sein, unabhängig von einem Wohlwollen des Kunden, subjektiver Tagesform oder unterschiedlicher individueller Wahrnehmungen. Wie sonst kann der Entwickler messbar überprüfen, ob er die Vorgabe erfüllt hat?

Läuft die Abnahme unstrukturiert und nach Tagesform, so ist niemandem geholfen.

Abnahmekriterien sollten genauso wie die Anforderungen vom Domänenexperten geschrieben werden und ermöglichen nicht nur die Validierung der Umsetzung von Anforderungen, sondern erhöhen auch die Qualität von Anforderungen signifikant. Die Projektrollen und Verantwortlichkeiten sind nicht überall gleich verteilt. So kann es auch ein Testmanager sein, der die Tests aufbauend auf den Anforderungen erstellt. Wenn man sich einen Testmanager leisten kann. Zur Not tut es auch ein Entwickler, der den jeweiligen Hut aufhat, also die Rolle ausübt.

An dieser Stelle wird ein wachsamer, agil veranlagter Leser bereits aufschreien oder alternativ gegebenenfalls sogar anfangen zu gähnen. Warum denn gleich so bürokratisch, fragen Sie sich? Die Antwort ist: Wie wollen Sie es denn anders machen? Mit dem Wissen um die Anforderungen steht und fällt der Erfolg der Lösung. Und dieses Wissen kann bei komplexeren Systemen niemals ausschließlich in den Köpfen sein. Es muss in irgendeiner Form geschrieben werden, sortiert, kategorisiert. Und priorisiert, damit man weiß, in welchen Schritten was abzuarbeiten ist.

Soviel zu den wichtigen Grundlagen. Es gibt einschlägige Literatur[2], die sich nur mit dem Thema befasst und die der interessierte Leser auf der Suche nach weiteren Details und einer umfassenden Diskussionen konsultieren kann. Nun gehen wir dazu über, wie mit den Anforderungen im agilen Umfeld umgegangen wird, denn im Gegensatz zum „klassischen" bzw. – und das kann man klar und deutlich sagen – utopischen Versuch, alles über das System bereits im Vorfeld herausfinden, sieht es hier häufig etwas anders aus. Denn während man noch die Anforderungen an das System sammelt und sortiert, ändert sich die Marktsituation zwanzig Mal. Dem begegnet die agile Softwareentwicklung gerne mit einer anderen Strategie.

Anforderungen und agiles Vorgehen

Agile Projekte pointieren den iterativen, inkrementellen Charakter der Softwareerstellung (für eine detaillierte Diskussion zu Inkrementen und Iterationen siehe Kapitel 5). So wird häufig darauf verzichtet, vor Entwicklungsstart eine vollständige Anforderungsspezifikation in großer Granularität zu definieren. Stattdessen wird mit einem Satz Anforderungen gestartet, der von der Granularität her das Zielsystem ausreichend beschreibt. Die im nächsten Schritt umzusetzenden Anforderungen werden entsprechend detailliert beschrieben. Von allen umzusetzenden Anforderungen werden die zur anstehenden Entwicklung ausgesucht, die den höchsten Wert für den Kunden liefern (siehe dazu auch Kapitel 3).

Bevor eine neue Iteration startet, und das kann je nach Vorgehen auch als Release, Sprint oder Meilenstein bezeichnet werden (Leute, die es ganz genau nehmen, liefern sich intensive Debatten, um diese Begriffe zu unterscheiden … aber häufig bleibt es bei einer eher akademischen Kategorisierung), werden die Anforderungen aus dem großen Pool ausgesucht, die in der nächsten Iteration umgesetzt werden sollen. Diese Anforderungen müssen nun in eine hinreichende Spezifikationstiefe überführt werden. Dies kann einerseits vor den Entwicklungsarbeiten geschehen, andererseits auch während der Entwicklung noch feinjustiert werden. Feinjustieren heißt aber nicht, die Anforderungen währenddessen immer wieder komplett umzuwerfen oder neue Anforderungen umzusetzen. Vielmehr ist der Satz der in einer Iteration umzusetzenden Anforderungen fix. Als Satz kann die Gesamtheit der Komplexität oder die Summe des Aufwands angesehen werden.

Es existiert ein Spannungsfeld zwischen Iterationslänge und Anforderungsstabilität.

Im Spannungsfeld zwischen Iterationslänge und Anforderungsstabilität gilt es, das richtige Maß zu finden. Denn wie es das Prinzip ausdrückt, ist der Kunde König, und die Anforderungen (bzw. die Auswahl, welche

[2] „Requirements Engineering und Management", C. Rupp, Hanser, 5. Auflage, 2009

nun umgesetzt werden sollen) sind auszurichten an der Marktsituation bzw. den aktuellen Kundenbedürfnissen, also jeweils den aktuellen Prioritäten anzupassen. Einerseits ist Anforderungsstabilität vonnöten: Dies hilft auch bei dynamischen Umgebungen, die Arbeiten stabil zu einem planbaren Ergebnis zu bringen. Weitere Vorteile sind, dass dadurch überhaupt erst Aktivitäten verfolgt und der Durchsatz gemessen werden können. Eine selbstverstärkende Wirkung: Nur was gemessen werden kann, kann auch verbessert werden (engl. „you can't improve what you can't measure“).

Am Ende einer Iteration ist die Anforderung mit der korrelierenden Umsetzung final in der hinreichenden Spezifikation vorhanden. Mit der Versionsbereitstellung wird also nicht nur die Software selbst, sondern auch alle zugehörigen Artefakte eingefroren und bereitgestellt, um eine fachlich und technisch konsistente Version bereitstellen zu können. Dieser Ansatz wird häufig vor dem Hintergrund eines „Application Lifecycle Managements“[3] diskutiert: Es ist im Kontext einer Rückverfolgbarkeit von Interesse, reproduzierbar ermitteln zu können, welche Anforderungen genau in welchem Artefakt umgesetzt wurden.

Doch was heißt nun Anforderungsspezifikation? Auf einer grobgranularen Ebene und aus Sicht des fachlichen Release-Managements sind das beispielsweise aussagekräftige Release-Notes. Feingranularer und eher vor dem Hintergrund des technischen Release-Managements hat in letzter Zeit der Begriff der aufgabenbasierten Entwicklung (engl. „task based development“) Verbreitung gefunden. Dabei geht es darum, die Ausrichtung der täglichen Arbeiten werkzeugunterstützt zu sichern und die Umsetzung und Bereitstellung der Anforderung auf Artefaktebene zu verfolgen.

Aus dem wahren Leben

Apropos Ticketsystem: Anforderungen ausschließlich im Ticketsystem zu halten, kann (muss aber nicht) eine suboptimale Idee sein, insbesondere bei größeren Neuprojekten. Eine Firma tat das – mit dem Ergebnis, dass ihr Ticketsystem nur zwei Hierarchieebenen zur Verfügung stellte, wohl aber mehr nötig gewesen wären. Insgesamt brach man die gesamte Menge der in das neue Medium zu übertragenen Anforderungen auf ca. 900 Einzeltickets herunter, die irgendwie im Bezug zu einander standen. Die Suche war aufwendig, die Relationen selten vorhanden, weil nicht gepflegt. Wiederholungen und Fragmentierungen von Anforderungen über mehrere Tickets hinweg beherrschten das Bild, und die technische Expertise über das Ticketsystem war sehr begrenzt. Man hat sich für diesen konkreten Fall

[3] „Agile ALM“, M. Hüttermann, Manning, 2010

auf das falsche bzw. falsch konfigurierte Tool verlassen, E-Mails ungefiltert und nicht sonderlich aufbereitet ins Ticketsystem kopiert, und hinterher wusste niemand mehr, was sich an welcher Stelle befand. Als es Monate später Bedarf zum Nachschlagen gab, lautete die Antwort des Entwicklungsleiters: „Schau doch im Ticketsystem nach, da haben wir doch alles schön sauber stehen!"

Es soll an dieser Stelle abschließend erneut betont werden, dass das gewählte Vorgehen (und gewählte Werkzeuge) von Rahmenbedingen und Anforderungen abhängen. Häufig sind Ticketing-Systeme ein sehr gutes Mittel, ein Anforderungsmanagement zu unterstützen.

Nun noch ein Wort zur Darreichungsform. Wie Sie die Anforderungen beschreiben und wo Ihre Anforderungsspezifikation platziert wird, das sind Fragestellungen tertiärer Natur. Hier sind viele Darreichungsformen denkbar, von Anforderungsfällen (engl. „Use Cases") in Fachkonzepten, ausschließlich Freitext in Lastenheften oder Einträgen in Ticketsystemen. Allen Medien sollte gemeinsam sein, dass sie entweder vom Kunden geschrieben werden oder in der Sprache des Kunden verfasst sind. Hier helfen beispielsweise Ansätze wie Behavior-Driven Development[4]. Nur zu häufig sprechen Techniker eine ganz andere Sprache und haben eine ganz andere Sicht auf die Anwendung und die Fachlichkeit. Dies gilt es zu verhindern oder zumindest zu minimieren.

Wie macht es die Prominenz?

In Scrum existiert ein Produkt-Backlog, in dem alle umzusetzenden Features liegen. Vor jeder Iteration, dem Sprint, wird aus dem Produkt-Backlog das Sprint-Backlog gefüllt. Die Kategorie der Einträge ist dabei nur von untergeordneter Bedeutung, zum Beispiel Change Requests, Bugs, Neues Feature.

Eine Interpretation im agilen Vorgehen ist, das gesamte Programm testbar zu designen und zu programmieren. Hier werden Testfälle, auch Akzeptanztests genannt, ausführbar geschrieben (beispielsweise mit dem Rahmenwerk Fit[5]/Fitnesse[6]). Ein weiterer wichtiger Aspekt ist es, diese Tests in der Domänensprache des Kunden zu verfassen.

[4] http://en.wikipedia.org/wiki/Behavior_Driven_Development
[5] http://fit.c2.com
[6] http://fitnesse.org/

Anforderungen müssen fortgeschrieben werden.

Im optimalen Fall werden nicht nur während der kontinuierlichen Weiterentwicklung der Software, sondern insbesondere auch während fortlaufender Refaktorisierungen[7] die Anforderungen und somit auch die Abnahmekriterien fortgeschrieben. Hier ist also die Grenze fließend zwischen eingefrorenen, „abgenickten" Anforderungen und Change-Requests. Viele machen damit gute Erfahrungen, alles, was den Zustand der Software ändert (Bugfixes, Erweiterungen), einheitlich als Change aufzufassen. Dies kann insbesondere vor dem Hintergrund der Disziplinen Change-Management und Software Configuration Management von Interesse sein.

Fassen wir doch nun die elementaren Aspekte für ein erfolgreiches, agiles Vorgehen im Zusammenhang mit dem Anforderungsmanagement zusammen. Wichtige Aspekte sind:

- Gehen Sie diszipliniert vor, und kommunizieren Sie transparent.
- Überführen Sie Ihre klare Vision in konkrete Anforderungen, die Sie iterativ/inkrementell umsetzen.
- Wählen Sie Strategien, um die nächsten Anforderungen auszusuchen (zum Beispiel das Kano-Modell).
- Prüfen Sie den Einsatz einer vernünftigen Planung, z.B. einer Meilensteinplanung (und deren Kommunikation!).
- Denken Sie daran, dass die Ergebnisse der Iterationen objektiv messbar sein sollten.
- Berücksichtigen Sie, dass Anforderungen Eigenschaften wie „widerspruchsfrei" oder „vollständig" besitzen.
- Der Domainexperte sollte die Anforderungen nicht nur vorgeben, sondern auch selbst schreiben.
- Die als nächstes umzusetzenden Anforderungen sollten Sie feingranularer beschreiben, als Anforderungen, die Sie zu einem späteren Zeitpunkt umsetzen möchten.

Eine agile Herangehensweise ist auch in eher reichhaltigen Vorgehensmodellen keine Seltenheit mehr. So ist beispielsweise die rollierende Planung[8] (engl. „rolling wave planning") nicht unüblich und macht eben genau das. Dabei werden die Inhalte des nächsten Releases sehr konkret bis auf Arbeitspaketgranularität heruntergebrochen, spätere Releases jedoch inhaltlich nur grob abgesteckt.

[7] http://www.refactoring.com

[8] http://de.wikipedia.org/wiki/Rollierende_Planung

4.2 Die Fehlinterpretationen

Wenn der im vorherigen Abschnitt angesprochene Spagat zwischen Iterationslänge und Anforderungsstabilität nicht gelingt und die Balance zwischen Stabilität und Flexibilität verloren geht, droht höchste Gefahr. Im Endeffekt münden diese Extreme in Entwicklungen ohne zugrunde liegende Anforderungen (genauer: ohne Anforderungsspezifikation) oder in puristischer Ausrichtung an Anforderungen, ohne Änderungen zuzulassen.

Requirements-Less Development

Es gibt viele Grautöne beim Vorgehen ohne Anforderungen. Fangen wir bei Dunkelschwarz an: Es existieren keine Anforderungen. Eine mögliche Konsequenz einer anforderungslosen Entwicklung ist in Bild 4.1 dargestellt. Dann die Frage: Was machen Sie da eigentlich, und für wen? Eine Nuance heller, aber immer noch tiefschwarz: Es existieren zwar Anforderungen, diese werden aber nicht (halbwegs) formal festgehalten.

Bild 4.1: Spring! Anforderungslose Entwicklung ähnelt dem Suizid auf diese Weise. Auf jede Weise eigentlich. Aber auf diese ganz besonders. (Quelle: ©iStockphoto.com/jgroup)

Unser Beileid, Sie haben verloren. Wie möchten Sie ohne beschriebene Anforderungen Ihre Entwicklungen ausrichten? Gegen was möchten Sie die Entwicklung überprüfen, also ob die Umsetzung mit dem Ziel kongruent ist? Doch wohl bitte nicht frei Schnauze.

Ohne halbwegs systematische Anforderungserhebung werden Sie im Chaos landen.

Sie benötigen nicht selten schon alleine aus rechtlichen Gründen eine fundierte Grundlage, also eine Beschreibung des Soll-Zustands. Das Problem dabei ist nur, dass ein Softwaresystem ein Großteil seines Lebenszyklus in der Wartung verbringt. Diese Wartung wird selten in Form von Projekten organisiert. Wenn man nicht aufpasst, wird die Wartung schnell zum sogenannten Tagesgeschäft. Und für dieses Tagesgeschäft macht man sich oft nicht die Mühe, Anforderungen in einer halbwegs formalen Art und Weise zu begegnen.

Aus dem wahren Leben

Es war einmal ein hochmotivierter Abteilungsleiter. Der hatte ganz viele Entwickler unter sich. Alle sprachen direkt mit den Kunden der internen Fachabteilung. Als ein strukturiertes Vorgehen eingeführt werden sollte, blockte der Abteilungsleiter ab. „Nein, nein, das können wir hier so nicht machen. Wir haben hier so ein immens wichtiges Tagesgeschäft. Wir müssen sofort handeln. Für solchen Firlefanz haben wir keine Zeit." Doch trotz des direkten Kontakts zu den Kunden performte die Entwicklungsabteilung recht schlecht. Warum? Kommunikation und Vorgehen verliefen von der Hand in den Mund, ohne jede Planung und Koordination. Es gibt nichts daran auszusetzen, dass Entwickler direkt mit den Fachleuten reden, nur befreit sie das nicht davon, die Anforderungen zu erfassen und vor allem zu priorisieren.

Das Tagesgeschäft basiert oft nicht auf Anforderungen und ist deshalb häufig ein bodenloses Aufwandsgrab.

Die aus dem Tagesgeschäft resultierenden Aufwände werden in etlichen Entwicklungsabteilungen schlichtweg nicht kalkuliert, sondern es werden Leute zur Aufrechterhaltung dieses Tagesgeschäfts (zumindest teilweise) abgestellt, die sich mit diesem beachtlichen und kaum gefilterten Aufwandsgrab beschäftigen. Fernab vom Tagesgeschäft kann aber natürlich auch in der Projektarbeit anforderungslos entwickelt werden. Wenn die Arbeiten, insbesondere in einem kommerziellen Projekt (das sich per Definition über Inhalte, Kosten und Zeit definiert), nicht an konkreten Anforderungen ausgerichtet sind, konterkariert das das essenzielle Ziel eines Projekts, und im allerschlimmsten Fall kann die Arbeit zu einer Spielerei verkommen, getreu dem Motto „Was mach ich denn heute mal?". Ein befreundeter Projektleiter verglich eine konkrete Situation einmal mit einer „Bastelbude".

Aus dem wahren Leben

Das mit den Kärtchen ist so eine Sache. Einmal sollte ein Team von einem Raum in den anderen umziehen. Leider hatte man vergessen, die Kärtchen vom Whiteboard abzunehmen. Der Hausmeister und dessen Gehilfen hatten den Auftrag, das Board abzuschrauben und in den anderen Raum zu schleppen. Zum Wochenende war die Sache erledigt. Bloß haben sie im Eifer des Gefechts einige bis fast alle Kärtchen auf dem Boden verteilt, und die Putzfrau hat sie am Morgen brav eingesammelt und in den Müllsack getan. Und das Team wusste nicht mehr, wo es stand, denn keiner hatte zwischendurch Fotos geschossen. Pech gehabt.

Anforderungen ohne Meilensteine umzusetzen, ist eine weitere suboptimale Ausprägung. Dies führt zu Death March-Projekten, siehe Kasten unten. Mal angenommen, Sie haben einen gewissen Satz an Anforderungen. Wunderbar! Wenn Sie ohne Meilensteine arbeiten (wir grenzen hier bewusst leichtgewichtig ab zwischen traditionellen Projektmanagementbegriffen und artverwandtem, agilem Vorgehen), wie wollen Sie dann den aktuellen Stand der Umsetzung messen? Wie wollen Sie rechtzeitig auf Probleme aufmerksam werden? Vielleicht kennen Sie, wenn Sie Glück haben, genau einen Meilenstein, den finalen Fertigstellungstermin. Ist das nicht ein bisschen wenig? Im agilen Vorgehen ist jedes aus einem Release purzelnde Inkrement ein lauffähiger Meilenstein, der dem Kunden zur Begutachtung (nicht nur auf dem Papier, sondern lauffähig) vorgelegt wird. Daraus zu gewinnende Verständnissynergien, Erfahrungswerte und Lerneffekte sind durch nichts zu ersetzen, schon gar nicht durch auf bunten Folien skizzierte Zielbilder.

Death Sprint und Death March

Dem Death Sprint liegt ein überhitzter Projektplan zu Grunde. Nach außen sieht das Projekt zunächst sehr erfolgreich aus: Immer wieder werden neue Versionen mit neuen Eigenschaften abgeschlossen. Allerdings leidet die Qualität des Produktes sowohl nach außen sichtbar wie auch technisch, was allerdings nur der Entwickler erkennt. Die Qualität nimmt mit jeder „erfolgreichen" neuen Iteration ab.

Das Gegenteil von Death Sprint ist der Death March. Das Projekt zieht sich ewig hin. Der Misserfolg ist objektiv sichtbar. Schlimmstenfalls ist das Projekt kein Projekt mehr, sondern nur eine zeitlich nicht abgeschlossene Aneinanderreihung von Aktivitäten. Es fehlen konkrete Zusagen für Termine und Lieferung. Kann auch bewusst in Kauf genommen werden, um von Defiziten in der Organisation und einem unklaren Zielbild abzulenken.

Eng verwoben damit ist die Konstellation, dass Sie keine Abnahme- bzw. Akzeptanzkriterien haben. Das ist bemerkenswerterweise recht häufig der Fall und häufig eine Konsequenz daraus, dass keine Anforderungen definiert bzw. diese nicht ausreichend präzisiert vorliegen. Ein guter Indikator für suboptimales Anforderungsmanagement ist, wenn kein Abnahmekriterium festgelegt werden *kann*. „Abnahme" erfolgt dann in Abhängigkeit von der subjektiven Sicht, der Tagesstimmung und vom Wohlwollen der Beteiligten. Das hat mit koordinierter, objektiv nachvollziehbarer Umsetzung und Abnahme nichts zu tun.

Puristisch anforderungsgetriebene Entwicklung

Übertreibt Ihr Projekt die Ausrichtung an Anforderungen bis ins Unermessliche, so haben Sie ebenfalls einen ungemütlichen Projektverlauf. Die Anforderungen werden erhoben und definiert, das ist gut. Sie werden allerdings zu früh auf Halde definiert. Haben Sie ein lange laufendes Projekt und spezifizieren Sie ausführlich sämtliche Anforderungen und schließen Korrekturen während der Laufzeit aus, so produzieren Sie „auf Vorrat", da nicht alle Anforderungen direkt und gleichzeitig umgesetzt werden, und Sie verschenken den richtigen Augenblick:

Anforderungen, genauso wie Entscheidungen grundsätzlich, sollten erst zum spätestens möglichen Augenblick gefällt werden. So profitieren Sie von Erkennt-

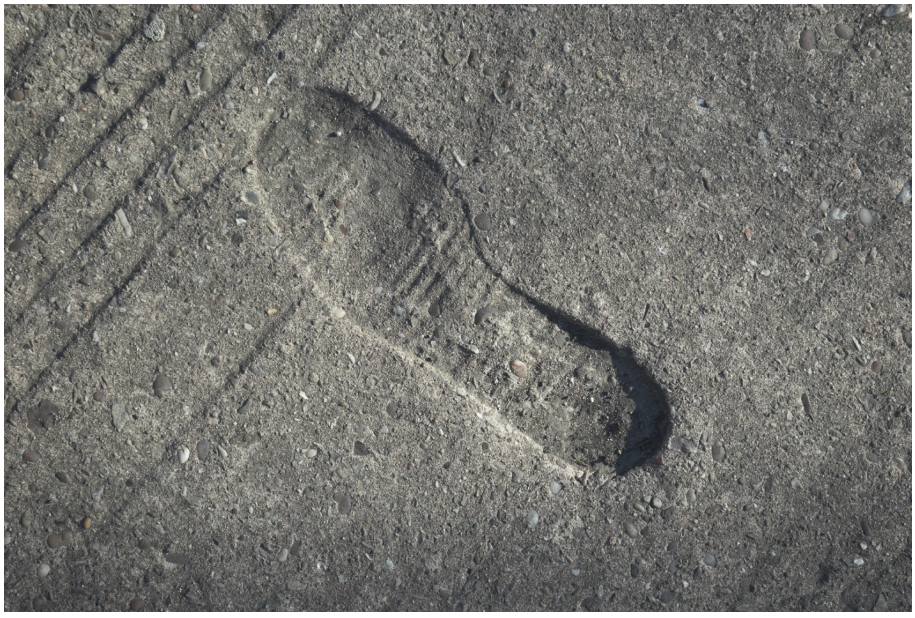


Bild 4.2: Lieben Sie nicht auch solche Fußabdrücke im frischen Zementboden, insbesondere wenn er bereits über Nacht getrocknet ist? Dann wissen Sie sicher auch die Anforderungen zu schätzen, die sich nachträglich genauso „leicht" ändern lassen.
(Quelle: ©iStockphoto.com/Marbury)

nissen auf dem langen Weg bis zu diesem Zeitpunkt. Mit anderen Worten: Wenn Sie die Entscheidung über sämtliche Anforderungen früh, zu früh fällen, verzichten Sie auf potenzielle, neue Erkenntnisse[9]. „Auf Halde produzieren" wird in Kapitel 3 ebenfalls thematisiert.

Wenn Sie Ihre Anforderungen, wie auf dem Bild 4.2 dargestellt, quasi in Zement gießen, verschenken Sie die Marktflexibilität, die so wichtig ist für die Konkurrenzfähigkeit eines jeden Produktes. Und nachdem die Softwareprojekte in der Regel langläufig sind, ist es unerlässlich, noch während der Entwicklung auf Marktveränderungen zu reagieren. Durch puristisch anforderungsgetriebene Entwicklung geht also nicht selten Marktflexibilität und Manövrierbarkeit verloren.

[9] Vgl. „Lean Software Development", Poppendieck und Poppendieck, Addison-Wesley, 2003