Ausgabe 01/2020
www.agilereview.de

agile review

Einblicke • Erfahrungen • Erkenntnisse

Agilität aufgleisen!

Team Liftoff

Technische Exzellenz

Strategie für Klarheit

Idealized Design

editorial

Liebe Leserinnen und Leser,

immer wieder werden agile Teams gestartet. Und jeder denkt: „Das ist doch einfach!“ Für jemanden mit viel Erfahrung in agilen Vorgehensweisen ist es das auch. Aber nicht immer ist so eine Person zur Stelle, wenn es darum geht, das nächste agile Team aufzusetzen oder die nächste Organisation auf agil umzustellen. Deswegen wollen wir mit dieser Ausgabe zeigen, wie man agile Teams und agile Organisationen aufgleisen kann, sodass sie den richtigen Weg gehen. Haben Sie Ihre eigenen Erfahrungen und Erlebnisse gemacht, von denen wir berichten dürfen? Ich würde mich freuen, von Ihnen zu hören!

Den Start machen dieses Mal meine Kollegen Marco Bonk und Markus Gärtner. Sie haben sich das Buch „Lift-Off“ zur Hand genommen und in der Praxis mit zwei verschiedenen Teams erprobt. Lesen Sie ab Seite <?> ihre Erfahrungen und Gedanken dazu. Bei einem Projektstart ist viel zu bedenken und dabei geraten manche technischen Fähigkeiten in den Hintergrund. Welche negativen Auswirkungen das haben kann und wie wichtig es ist, bereits ganz am Anfang technische Exzellenz aufzubauen beschreiben ab Seite <?> Steven Collins und Stefan Roock.

Ab Seite <?> führen Ilja Preuß und Myriam Troester ein Interview, das zum Machen aufruft. Ihre Interviewpartner beschreiben anschaulich, dass viel Nutzen entstehen kann, wenn sich Menschen in einer Organisation nicht aufhalten lassen. Sven Günther bespricht in unserer Rubrik „Behind the Board“ ab Seite <?> ein Kanban-Board aus der Praxis. Hier können Sie sehen, wie Hamburg Wasser mittels Portfolio-Kanban die Projekte organisiert.

Eine Strategie zu entwickeln ist ein anspruchsvolles Vorhaben. Markus Andrezak beschreibt ab Seite <?> ein alternatives Vorgehen, indem er Marker, Optionen und Arbeit benutzt. Mithilfe von Idealized Design zeigt mein Kollege Ralf Lethmate wie mächtig das Setzen von Rahmen darin ist, um Innovationen zu fördern, ohne selbstorganisierten Teams konkrete Lösungen vorzugeben. Ab Seite <?> lernen Sie, was es damit auf sich hat.

Die Buchtipps ab Seite <?> gehen in dieser Ausgabe auf die Autoren James Womack und Daniel Jones, Donald Reinertsen, Henrik Kniberg und Mathew May ein. Und in unserer Rubrik „Auf einen Kaffee mit...“ hat sich Nadine Wolf dieses Mal mit Laura Austermann verabredet. Lesen Sie das Interview ab Seite <?>.

Ich wünschen Ihnen ganz viel Lesespaß, Ansatzpunkte für das Aufgleisen Ihres Teams und freue mich über Ihr Feedback!

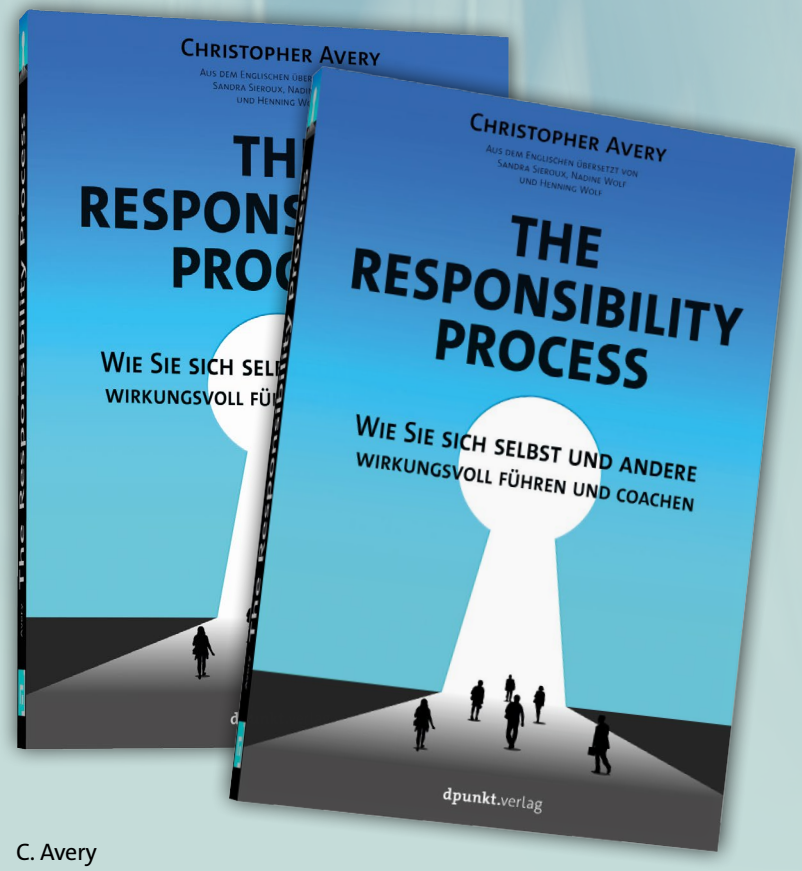
Wolf-Gideon Bleek, Chefredakteur der agile review

Treten Sie gern mit dem Autor in Verbindung: wgb@it-agile.de

INHALT

Markus Andrezak erklärt die unterschiedlichen Ebenen von Markern, Optionen und Arbeit und wie man diese miteinander koppelt ohne ihre wertvollen, unterschiedlichen Eigenschaften zu beeinträchtigen.

Wissen für agile Leader



C. Avery

The Responsibility Process

Wie Sie sich selbst und andere wirkungsvoll führen und coachen

Aus dem Englischen von S. Sieroux, N. Wolf und H. Wolf

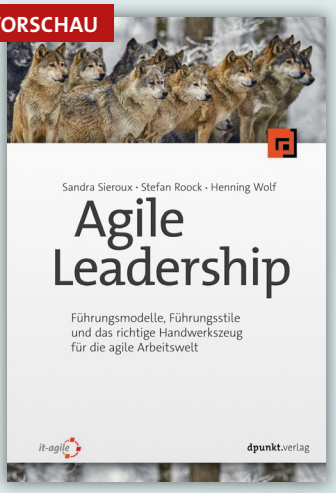
2019, 294 Seiten, Broschur
€ 24,90 (D) · ISBN 978-3-86490-577-3

Der Autor zeigt mit »The Responsibility Process™« den Weg und die Zwischenschritte hin zu echter Verantwortungsübernahme. Er gibt Ihnen konkrete Werkzeuge, Praktiken und Leadership-Weisheiten an die Hand, mit denen Sie lernen, diesen Prozess bewusst einzusetzen. So können Sie sich selbst und anderen kraft- und wirkungsvolles Handeln ermöglichen.

»This book changed how I talk about responsibility in my training courses and with my teen daughters. An ownership mindset is critical to business agility. Avery brilliantly dissects how ownership (or the lack of it) works in the mind and what to do about it to lead yourself and others to freedom, happiness, and results that matter.«

Zitat zur engl. Originalausgabe von Mike Cohn, Autor von »Succeeding with Agile«

VORSCHAU



S. Sieroux · S. Roock · H. Wolf

Agile Leadership

Führungsmodelle, Führungsstile und das richtige Handwerkszeug für die agile Arbeitswelt

2. Quartal 2020, ca. 230 Seiten
komplett in Farbe, Broschur
ca. € 29,90 (D)
ISBN 978-3-86490-696-1

VORSCHAU

L. D. Marquet

Reiß das Ruder rum!

Eine wahre Geschichte über Führung und darüber, wie Mitarbeiter zu Mitgestaltern werden

Aus dem Englischen von R. Lethmate, U. Reupke und W. Wiedenroth

2. Quartal 2020, ca. 272 Seiten
Broschur
ca. € 24,90 (D)
ISBN 978-3-86490-737-1

dpunkt.verlag GmbH · Wieblinger Weg 17 · D-69123 Heidelberg
fon: 0 62 21 / 14 83 40 · fax: 0 62 21 / 14 83 99
e-mail: bestellung@dpunkt.de

Buch + E-Book: www.dpunkt.de

dpunkt.verlag

Technische Exzellenz von Beginn an

Agile Arbeitsweisen werden im IT-Mainstream und außerhalb der Softwareentwicklung erfolgreich eingesetzt. Der Erfolg wird vor allem der Fokussierung auf den Geschäftswert mithilfe von kurzen Iterationen in Form von Sprints, Priorisierung nach Geschäftswert mit Backlogs[1] oder auch Transparenz über den Projektfortschritt mit Burndown-Charts und Daily Standups zugeschrieben. Versuchen nicht-agile Organisationen das Vorgehen und den Erfolg zu kopieren, werden die weniger sichtbaren technischen Fähigkeiten übersehen, nicht mit übernommen oder nicht aufgebaut.
Eine Fokussierung auf den Geschäftswert in der Softwareentwicklung ist nur mit bestimmten Entwicklungstechniken möglich. Wir beschreiben die entstehenden Probleme, wenn das Team die agilen Entwicklungstechniken nicht verwendet. Diese sind schwerwiegend, werden aber erst spät sichtbar. Dann ist eine Kurskorrektur nur mit Aufwand möglich. Wir zeigen früher sichtbare Indizien für Schwächen bei den agilen Entwicklungspraktiken. Anschließend skizzieren wir, welche Entwicklungstechniken möglichst früh genutzt werden sollten, um den Problemen entgegenzuwirken, und wie diese Entwicklungstechniken praktikabel eingeführt werden können. Von Steven Collins und Stefan Roock

Technische Herausforderungen agiler Entwicklung

Das iterativ-inkrementelle Vorgehen agiler Entwicklung arbeitet ohne lange Planungsphasen. Was genau benötigt wird, lernt das Team erst auf dem Weg. Dieser Ansatz kann nicht dauerhaft funktionieren, wenn klassische Entwicklungstechniken verwendet werden, denn diese schaffen nicht die notwendigen Voraussetzungen für flexible Priorisierung und kontinuierliche Lieferfähigkeit.

flexible Priorisierung

Eine besondere technische Herausforderung entsteht durch die flexible Priorisierung der Features (z. B. durch den Product Owner). Die Entwicklungsreihenfolge orientiert sich beim agilen Vorgehen nicht mehr an vermeintlichen technischen Gegebenheiten, sondern am Geschäftsnutzen. Nur so entsteht für das Unternehmen ein echter Vorteil. Wenn Feature nach Feature entwickelt wird, ohne den existierenden Code kontinuierlich anzupassen, entsteht allerdings schnell unwartbarer Code. Tatsächlich kann man mit iterativer Entwicklung noch schneller unwartbaren Legacy-Code (siehe Kasten) produzieren als mit dem Wasserfall-Vorgehen. Es entsteht eine steile Aufwandskurve (siehe Abb. 1).

1 | Der Einfachheit halber verwenden wir die Scrum-Terminologie. Die Aussagen des Artikels gelten aber für jede Form iterativ-inkrementeller Entwicklung. ▶

Wenn die Entwicklungskosten so schnell steigen, kommt die ganze Entwicklung schnell zum Erliegen. Stattdessen brauchen wir eine flache Aufwandskurve (siehe Abb. 2). Wenn diese hergestellt werden kann spielt es bzgl. der Entwicklungskosten tatsächlich keine nennenswerte Rolle, wann ein neues Feature entdeckt und priorisiert wird. **Eine flache Aufwandskurve ist also eine notwendige Bedingung für agile Entwicklung.**

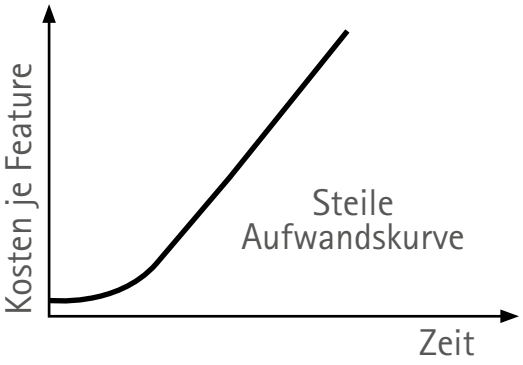


Abb. 1: Kosten je Feature bei einer steilen Aufwandskurve

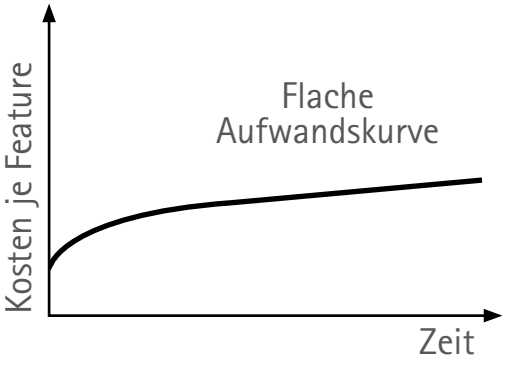


Abb. 2: Kosten je Feature bei einer flachen Aufwandskurve

Legacy Code

Code-Altlasten werden als Legacy Code bezeichnet. Legacy Code ist aus heutiger Sicht unzureichend strukturierter Code, so dass dieser nur mit großen Aufwänden und hohem Risiko zu ändern ist. Legacy Code hat in den meisten Fällen deutliche Schwächen in der Modularisierung, besitzt unnötige Code-Duplizierungen und ist unzureichend mit automatisierten Tests abgesichert.

Lieferfähigkeit

Bei der klassischen Entwicklung wird das System „von unten nach oben" entwickelt (siehe Abb. 3). Als Metapher wird häufig der Hausbau verwendet: zuerst das Fundament, dann der Keller, dann das Erdgeschoss etc.
Diese Denkweise ist allerdings inkompatibel mit den agi

Abb. 3: Horizontale Entwicklungsrichtung bei klassischer 3-Schichten-Architektur

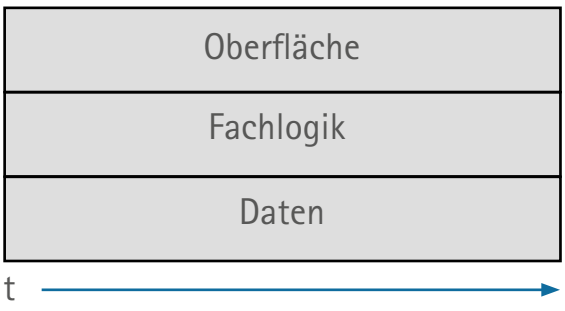


Abb. 4: Vertikale Entwicklungsrichtung bei agiler Architektur

len Ansätzen. Schließlich wollen wir möglichst früh Kundenfeedback erhalten und möglichst früh ausliefern. Im o. g. Vorgehen ist das allerdings erst spät möglich, nämlich dann, wenn die Oberfläche zumindest teilweise existiert.
Bei der agilen Vorgehensweise müssen wir die verschiedenen Schichten parallel entwickeln (siehe Ab. 4).
Mit jeder neuen Iteration muss ein Teil des existierenden Codes geändert werden – schließlich soll das neue Inkrement nicht zusammenhangslos neben dem alten stehen, sondern integriert sein (siehe Abb. 5).

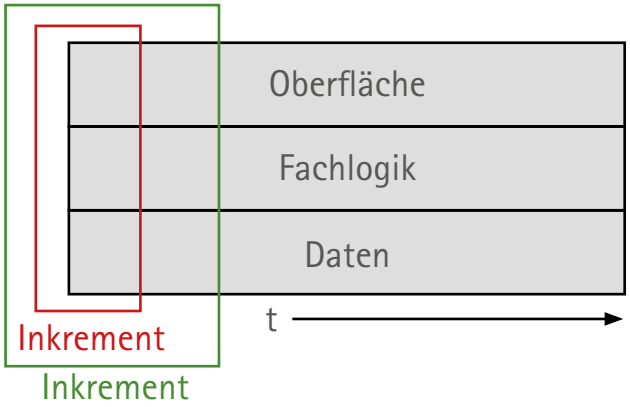


Abb. 5: Vertikales Produktinkrement

Probleme durch entwicklungstechnische Schwächen

Sorgt das Team nicht kontinuierlich für gut strukturierten und aufgeräumten Code, spricht man von technischen Schulden (siehe Kasten), an deren Zins und Zinseszins schon viele Projekte gescheitert sind. Es entsteht eine steile Aufwandskurve mit fatalen Folgen:

- Die Weiterentwicklung wird immer langsamer.
- Dadurch kann nicht mehr ausreichend schnell auf die Wünsche der Kunden und Stakeholder reagiert werden.

Technische Schulden

Wenn Entwickler (z. B. aus Zeitdruck) der Code-Qualität nicht genügend Aufmerksamkeit und Zeit widmen, entstehen sogenannte technische Schulden. Sie nehmen quasi einen (Zeit-)Kredit auf, auf den sie Zinsen zahlen. Durch Einsparungen bei der technischen Qualität des Codes, ist das Team kurzfristig schneller. Die Qualitätsdefizite machen das Team dann aber langsamer, weil weitere Änderungen am Code immer teurer werden. Im schlimmsten Fall verschlingen die technischen Schulden den Großteil der Entwicklungskapazität und führen faktisch zum Bankrott des Teams.

- Dadurch steigt der Druck auf der Entwicklungsteam weiter an. Mehr Druck führt zu immer mehr Bugs und technischen Schulden im System.
- Die Weiterentwicklung wird immer schlechter vorhersagbar und planbar.

Leider werden diese Effekte erst verzögert sichtbar. In dem Moment, in dem sie deutlich werden, ist das Kind bereits in den Brunnen gefallen und kann nur mit ganz erheblichem Aufwand wieder herausgeholt werden. Daher beschreiben wir im folgenden Abschnitt, an welchen Indizien man früher erkennen kann, dass es Probleme mit den agilen Entwicklungstechniken gibt.

Fallbeispiel: Story Point-Rennen

In einem von uns betreuten Projekt arbeiteten zwei Scrum-Teams am selben Produkt. Die ersten Sprints sahen vielversprechend aus; beide Teams wurden immer schneller (siehe Abb. 6).
Auf dieser Basis wurde dem Lenkungsausschuss avisiert, dass die Entwicklung vermutlich sogar vor der Deadline abgeschlossen werden kann. Kurz nach dieser freudigen Nachricht brach' für den Product Owner allerdings eine ▶

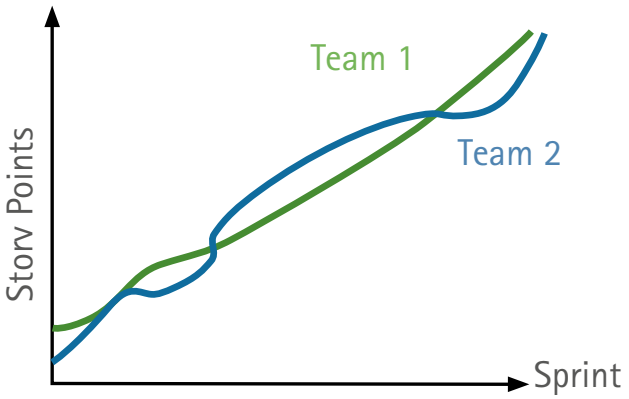


Abb. 6: Burn-Up-Chart zweier Teams zu Projektbeginn

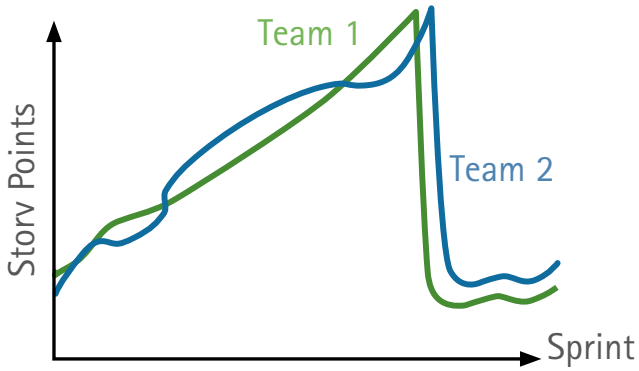


Abb. 7: Auswirkung von mangelnder Code-Qualität auf die Entwicklungsgeschwindigkeit

Welt zusammen. Plötzlich kam die Entwicklung beider Teams fast zum Erliegen und die Restaufwände stiegen immer weiter an.

Zu diesem Zeitpunkt wurden wir hinzugezogen, um die Situation zu analysieren. Es stellte sich heraus, dass ein implizites Story-Point-Wettrennen stattgefunden hatte. Im Projekt wurden die Entwicklungsgeschwindigkeiten der beiden Teams veröffentlicht und verglichen. Das spornte die Teams an, schneller zu sein als das jeweils andere Team auf Kosten der Code-Qualität. Die stetig sinkende Code-Qualität ließ sich durch besondere Kraftanstrengungen eine Weile verdecken; irgendwann konnten die Teams die Probleme jedoch nicht mehr ignorieren und mussten mehrere Sprints in die Renovierung des Codes investieren. Neben dem Verlust dieser Sprints bestand eine weitere Konsequenz in der schlechten Vorhersagbarkeit für die Restentwicklung. Es konnte extrem schlecht abgeschätzt werden, bis wann die Entwicklung des ersten Releases abgeschlossen werden konnte.

Indizien für entwicklungstechnische Schwächen

Product Owner sind den o. g. Problemen allerdings nicht hilflos ausgeliefert. Es gibt Indizien, anhand derer man frühzeitig erkennen kann, dass das Team die agilen Entwicklungspraktiken nicht ausreichend verwendet:

- technische User Stories
- keine flexible Priorisierung
- große User Stories
- schwankende Entwicklungsgeschwindigkeit

technische User Stories

Das klassische Entwicklungsvorgehen geht wie der Hausbau von unten nach oben vor: Datenbank, Fachlogik, Benutzungsoberfläche. Insistiert das Team darauf, bei agilem Vorgehen zuerst eine „umfassende Basis“ oder „Infrastruktur“ zu entwickeln, deutet das darauf hin, dass es noch nicht in der Lage ist, das System Feature für Feature zu bauen und dabei die Code-Qualität zu erhalten.

keine flexible Priorisierung

In unseren Product-Owner-Kursen berichten die Teilnehmer immer wieder, dass sie nicht flexibel priorisieren können – weder nach Geschäftswert noch nach Stakeholder-Interessen. Stattdessen schreibt das Team faktisch die Reihenfolge der Entwicklung vor. Die Ursache ist nicht

Unwillen des Teams, sondern technische Unfähigkeit. Es weiß schlicht nicht, wie es wartbaren Code schreiben kann, wenn Features in beliebiger Reihenfolge entwickelt werden sollen.

große User Stories

User Stories lassen sich konzeptionell immer wieder zerlegen, bis sie fast beliebig klein werden. Einige Teams neigen allerdings dazu, große User Stories zu definieren, mitunter mit dem Argument, sie könnten die Story gleich richtig fertig machen, wie sie „schon mal dabei wären". Hier scheuen sich die Entwickler davor, einmal entwickelten Code nochmal ändern zu müssen. Hinter diesem Argument steckt häufig die Angst, diese Änderungen nicht kostengünstig durchführen zu können.

schwankende Entwicklungsgeschwindigkeit

Wenn das Team es nicht schafft, den Programmcode auf gleichbleibend hoher Qualität zu halten, schwankt die Entwicklungsgeschwindigkeit von Sprint zu Sprint: das Team muss immer wieder anhalten, um Code aufzuräumen und neue Features umständlich in den existierenden Code zu integrieren: Stop-and-go.

Die Rolle des Scrum Masters

Scrum Master können einen technischen Hintergrund haben, müssen es aber nicht. Letztlich liegen die Entwicklung der Software und die damit verbundenen technischen Entscheidungen im Entwicklungsteam. Das bedeutet aber nicht, dass der Scrum Master mit den technischen Aspekten der Entwicklung nichts zu tun hätte.
Der Scrum Master muss dafür sorgen, dass dem Team die technischen Herausforderungen und das dafür geeignete Handwerkszeug bekannt sind. Nur dann kann das Team eine informierte Entscheidung darüber treffen, wie es den technischen Herausforderungen begegnen möchte.

Frühe Entwicklungstechniken

Es gibt eine große Fülle agiler Entwicklungstechniken, die sowohl Architektur- und Entwurfsfragen als auch das Vorgehen beim Entwickeln thematisieren. Glücklicherweise muss man nicht alle Techniken von Anfang an beherrschen. Die folgende Auswahl ist nach unserer Erfahrung ein guter Start, um nachhaltig agiles Arbeiten zu ermöglichen. Fortgeschrittene Techniken (siehe unten) helfen später dabei, noch effektiver und reaktionsfähiger zu arbeiten. Wir schlagen diese Starttechniken vor und erläutern sie anschließend:

- Modularisierung mit Domain Driven Design
- Dimensional Planning
- Kontinuierliche Integration
- Automatisierte Tests
- Refactoring

Modularisierung mit Domain Driven Design

Wenn Sie sich einen Laptop eines großes Herstellers aus Kalifornien zusammenstellen, werden Sie vermutlich lange in die Planung ihrer Konfiguration investieren, um beispielsweise eine Entscheidung zu treffen, wie viel Arbeitsspeicher ihr Gerät haben soll. Da der Arbeitsspeicher fest im Gerät verbaut wird, müssen Sie heute wissen, was die Anforderungen an ihr Gerät in den kommenden Jahren sein werden. Obwohl Ihnen heute 16 GB Hauptspeicher reichen würden, sind Sie vielleicht gezwungen, die fast ein Drittel teurere Variante mit 64 GB zu wählen. So versuchen Sie, sich für eine Zukunft zu wappnen, in der Anwendungen noch höhere Hardwareanforderungen als heute stellen. Wäre es nicht toll, wenn Sie jetzt die günstigere Variante wählen und erst dann zusätzliche Arbeitsspeicher-Module hinzukaufen könnten, wenn die ▶

Anforderungen für Anwendungen tatsächlich steigen? Überlegen Sie mal, was Sie heute mit dem gesparten Geld alles Sinnvolles anfangen könnten!

Das gleiche Prinzip gibt es in der Softwareentwicklung auf verschiedenen Ebenen: Wenn Sie ihr zu entwickelndes System fest an ein anderes System aus Ihrem Unternehmen koppeln, so werden neue Anforderungen nur noch durch koordinierte Änderungen umsetzbar. Das resultiert in verlängerten Durchlaufzeiten neuer Anforderungen, denn die notwendigen Änderungen müssen zuerst in dem System umgesetzt werden, von dem Sie sich abhängig gemacht haben. Wie Kunden bei der Zusammenstellung ihrer Laptop-Konfiguration, so können auch Unternehmen in dieser Situation nicht mehr flexibel handeln. Deshalb betreiben viele Unternehmen einen immensen Planungs- und Verwaltungsaufwand, nur um Anforderungen in weiter Zukunft zu koordinieren, die bis zu ihrer Realisierung eventuell wertlos geworden sind.

Im schlimmsten Fall haben Sie sich fest an ein zentrales System ihres Unternehmens gebunden, das den Flaschenhals für die Entwicklungsgeschwindigkeit in Ihrem Unternehmen darstellt. Ein häufiges Beispiel ist die zentrale Kunden- oder Benutzerverwaltung, an die sich andere Systeme koppeln. Ist diese Kundenverwaltung nicht erreichbar, fallen alle abhängigen Systeme aus. Schmerzhaft ist es, wenn Tests nicht ausgeführt werden können oder keine Aussagekraft haben, weil ein zentrales System in der Testumgebung nicht dauerhaft verfügbar ist. Innerhalb der eigenen Systemgrenzen kann man bei der technischen Umsetzung von fachlichen Anforderungen so feste Kopplungen herstellen, dass man die einzelnen Bestandteile nicht mehr getrennt voneinander ändern kann. Eine Änderung an einem Bereich der Geschäftslogik führt dann zu Anpassungen an allen Bereichen des Systems.

Neben den fachlichen Aspekten kann man technische Aspekte wie die Wahl einer konkreten Datenbank fest verlöten. Stellt sich diese Datenbank dann später als ungeeignet heraus, kann man diese nicht mehr sauber aus dem Rest des Systems entfernen.

Sowohl auf technischer, als auch auf fachlicher Ebene sowie auch auf System-Ebene hilft Domain-driven Design bei der Modularisierung. Durch Modularisierung sind die Bestandteile des Systems nur noch über definierte Schnittstellen miteinander gekoppelt. Änderungen in einem Modul führen nicht mehr zwangsläufig zu Änderungen in einem anderen Modul. Hat ein Team beispielsweise die Verwendung von Umsystemen ausreichend modularisiert, so kann das Team im Sprint Review den Fortschritt an hochpriorisierten Features zeigen, selbst wenn notwendige Änderungen an Umsystemen noch nicht erfolgt sind.

Story Splitting mit Dimensional Planning

Die aus Fachsicht wertschöpfenden Features sind groß und müssen klein geschnitten werden (Story Splitting). Unerfahrene Teams neigen zu Zerlegungstechniken, die Umbauarbeiten an existierendem Code vermeintlich minimieren. Leider führt diese Strategie dazu, dass die wichtigsten Dinge unnötig spät erledigt werden.

Besser sind Zerlegungsstrategien, die die wirklich wichtigen Dinge zuerst erledigen. Die Zerlegungstechnik Dimensional Planning arbeitet mit Walking Skeletons - man beginnt mit einem ersten (durchaus hässlichen) Durchstich (das Skelett) und hängt anschließend das Fleisch an. So wird für die Fachseite früh erkennbar, wie das System in Gänze funktioniert und relevante technische Risiken werden früh adressiert. Dimensional Planning geht mit regelmäßiger Änderung existierenden Codes einher und das Team hat zu lernen, wie es das Walking Skeleton bereits mit hoher Code-Qualität entwickelt, so dass die späteren Änderungen leichtfallen.

Kontinuierliche Integration

Durch die Entwicklung entlang von Features anstelle von Schichten, arbeiten Entwickler immer wieder an denselben Modulen. Damit die Entwickler sich dabei nicht ständig in die Quere kommen und schnell merken, wenn das doch der Fall gewesen sein sollte, hilft kontinuierliche Integration (Continuous Integration): Die Entwickler integrieren ihre Code-Änderungen möglichst häufig in die gemeinsame Code-Basis, deren Konsistenz dann über automatisierte Tests sichergestellt wird.

Automatisierte Tests

Wenn existierender Code geändert wird, besteht immer die Gefahr, dass sich Fehler in vormals korrekten Code einschleichen. Automatisierte Tests helfen dabei, diese Fehler so früh zu identifizieren, dass sie mit geringem Aufwand beseitigt werden können.

Automatisierte Tests können auf unterschiedlichen Ebenen arbeiten (Akzeptanztests, Oberflächentests, Integrationstests, Unittests etc.).

Dabei besteht eine große Herausforderung darin, die richtige Menge an Tests zu schreiben. Viele Systeme haben zu wenig Tests. Es gibt aber viele Systeme, die zu viele Tests haben. Diese werden problematisch, wenn existierender Code geändert wird. Dann müssen Tests angepasst werden. Im schlimmsten Fall übersteigen die Änderungsaufwände für die Tests die Änderungsaufwände am Produktivcode um ein Vielfaches. Dann machen die Tests die Entwicklung nicht schneller und agiler, sondern langsamer und schwerfälliger. Daher gelten o.g. Aussagen zu Modularisierung ebenfalls für Tests. Deshalb ist Modularisierung für Tests wichtig, damit diese unabhängig voneinander sind und wenig Überschneidungen haben. Idealerweise führt Code-Änderung in einem Modul dazu, dass genau ein Test angepasst werden muss.

Refactoring

Refactoring beschreibt die Verbesserung von Programm-Code unter Beibehaltung seines Verhaltens. Refactoring ist eine notwendig Voraussetzung dafür, dass Code trotz vieler Änderungen langfristig wartbar bleibt. Das Ziel ist, kontinuierlich kleine Refactorings durchzuführen, damit die Änderungsnotwendigkeiten sich nicht kumulieren und irgendwann einen großen Umbau oder gar Neubau erzwingen. Viele Entwicklungsumgebungen bieten Refactoring-Unterstützung an. Allerdings haben viele Teams nicht gelernt, diese effektiv zu nutzen.

Spätere Entwicklungstechniken

Es existieren eine ganze Reihe weiterer mächtiger Entwicklungspraktiken, die das Entwicklungsteam auf jeden Fall in Betracht ziehen sollte. Nach unserer Erfahrung benötigen sie z. T. aber deutlich größere zeitliche Investitionen, um sie zu erlernen und sollten daher in den meisten Fällen erst in Angriff genommen werden, wenn das Team die o. g. frühen Entwicklungstechniken bereits beherrscht. Zu diesen Techniken gehören:

- testgetriebene Entwicklung (Test Driven Development, TDD)
- kontinuierliche Auslieferung (Continuous Deployment, CD)
- Programmieren in Paaren (Pair Programming, PP)
- Programmieren im Mob (Mob Programming)

Mit diesen Techniken ist es insbesondere möglich, jederzeit in hoher Qualität neue Versionen an Endkunden auszuliefern. Beispielsweise für E-Commerce-Anwendungen ist das ein entscheidender Wettbewerbsvorteil. In anderen Bereichen ist der Anpassungsdruck des Marktes geringer und die Investitionen in die o.g. Techniken sollten im Verhältnis zum Nutzen explizit betrachtet werden. ▶

Einführung der frühen Entwicklungstechniken

Die beschriebenen frühen Entwicklungstechniken lassen sich am Besten in einer Mischung aus Schulungen, Workshops und „learning on the job" einführen. Zunächst sollten die Techniken in Schulungen vermittelt und in einer „Laborumgebung" eingeübt werden. Entkopplungstechniken und eine erste modulare Architektur lassen sich gut in Workshops definieren. Zuletzt ist der Transfer der Entwicklungspraktiken in die Praxis notwendig: Die Lernerfahrungen aus der Laborumgebung reichen den Teams nicht aus, um z. B. Testautomatisierung mit existierendem Code durchzuführen. Daher braucht es praxisbegleitendes Mentoring, damit die Entwickler lernen, wie die „theoretischen Ansätze" in ihrer Praxis funktionieren können.

Zusammenfassung und Abschluss

In vielen Unternehmen werden die technischen Herausforderungen agiler Entwicklung nicht gesehen oder unterschätzt. Folgerichtig werden dann nicht die passenden Techniken verwendet, um diesen Herausforderungen zu begegnen. Sowohl die Entwickler als auch die Stakeholder stellen nach einer anfänglichen Euphorie fest, dass sie nicht die Vorteile aus agiler Entwicklung ziehen können, die sie sich erhofft hatten.

Dabei gibt es jahrzehntelange Erfahrung damit, wie dieses Problem zu lösen ist. Wenn die Herausforderung in Angriff genommen und gemeistert wird, dann schaffen es agile Teams tatsächlich, flexible Priorisierung nach Geschäftswert zu ermöglich und verlässlich schnell zu liefern. ■

Treten Sie gern mit den Autoren in Verbindung:
steven.collins@it-agile.de, stefan.roock@it-agile.de

STEVEN COLLINS

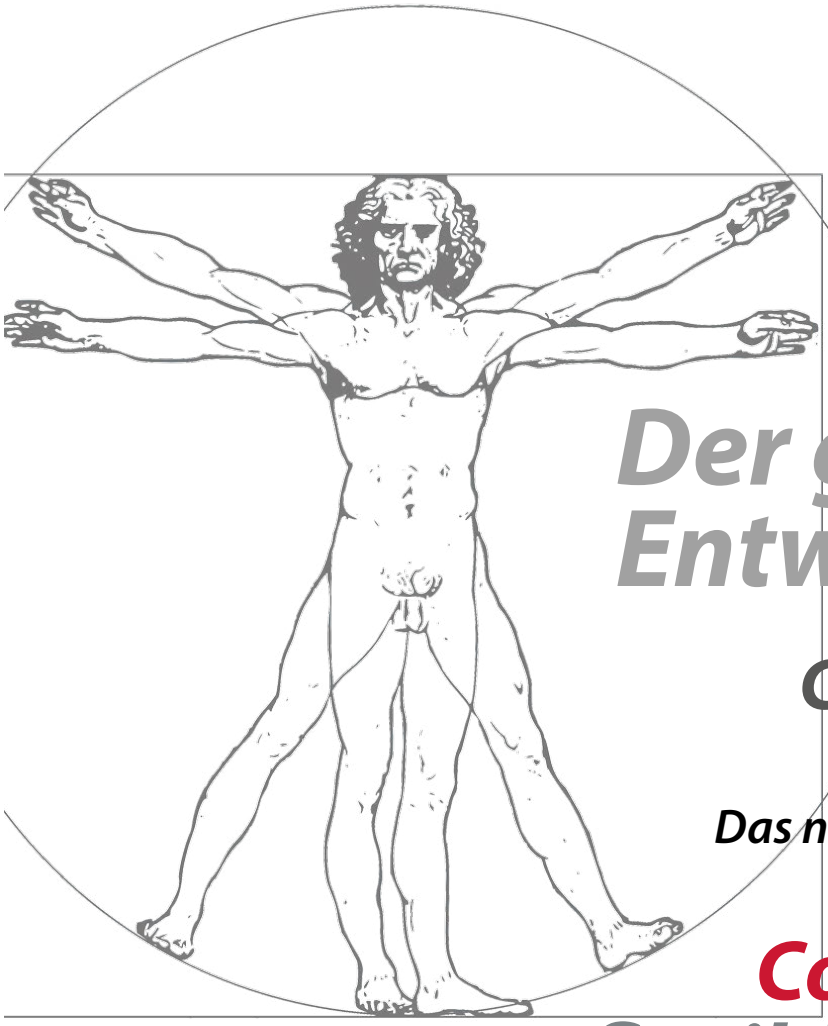
Steven Collins ist Agile Engineering Coach und weist seit über 10 Jahren im Auftrag von it-agile nach, dass agile Entwicklungspraktiken nicht nur langfristig zu wartbarer Software, sondern auch kurzfristig zu hoher Entwicklungsgeschwindigkeit führen. Sein Hauptaugenmerk liegt dabei auf dem Design von simplen und dennoch skalierbaren Softwaresystemen.

STEFAN ROOCK

Stefan Roock zählt zu den agilen Urgesteinen in Deutschland. Schon 1999 führte er erste Projekte mit Scrum und eXtreme Programming durch. Heute berät er Führungskräfte bei agilen Transitionen für einzelne Projekte, Organisationseinheiten oder ganze Unternehmen.
Er ist Autor mehrerer Bücher zu agilen Themen und regelmäßiger Sprecher auf Konferenzen und Tagungen.
Er lebt mit seiner Frau, zwei Söhnen und einer Tochter in Geesthacht und schreckt auch im Dezember nicht davor zurück, sich beim Windsurfen in der Ostsee nasse Füße zu holen.

it-agile

Entwicklung, Beratung, Schulung



Der ganzheitliche Entwickler

Certified Scrum Developer (CSD)

Das neue Zertifizierungsprogramm der ScrumAlliance

Collaboration

flexible Architektur

kontinuierliche Integration

Refactoring

testgetriebene Entwicklung

Java C# Python Cobol C++ PHP

Schulungen in Hamburg und München und bei Ihnen im Haus.

www.it-agile.de/csd info@it-agile.de

Team Liftoff
Agile Teams richtig zum Fliegen bringen

Bereits im Jahr 2012 haben Diana Larsen und Ainsley Nies die erste Ausgabe ihres Liftoff Buchs veröffentlicht [LarsenNies2012]. Darin beschreiben die beiden Coaches, wie man agile Teams zum Fliegen bekommen kann. Unsere Kollegen Marco Bonk und Markus Gärtner haben dieses Konzept bei einem Kunden an einem Tag für zwei unterschiedliche Teams verwendet und dabei lehrreiche Erkenntnisse erlangt, wann sich so ein Lift-Off lohnt und welche Voraussetzungen vorher gegeben sein sollten, damit das Team wirklich abheben kann. Von Marco Bonk und Markus Gärtner

it-agile

Entwicklung, Beratung, Schulung

Mit dem Latein am Ende?

Damit dein Team abliefert –
it-agile zeigt wie.

www.it-agile.de/coaching info@it-agile.de

Einfach mal machen und machen lassen

Ein Gespräch über das Ausprobieren

Das Interview führte Ilja Preuß, Redaktion Myriam Troester

it-agile

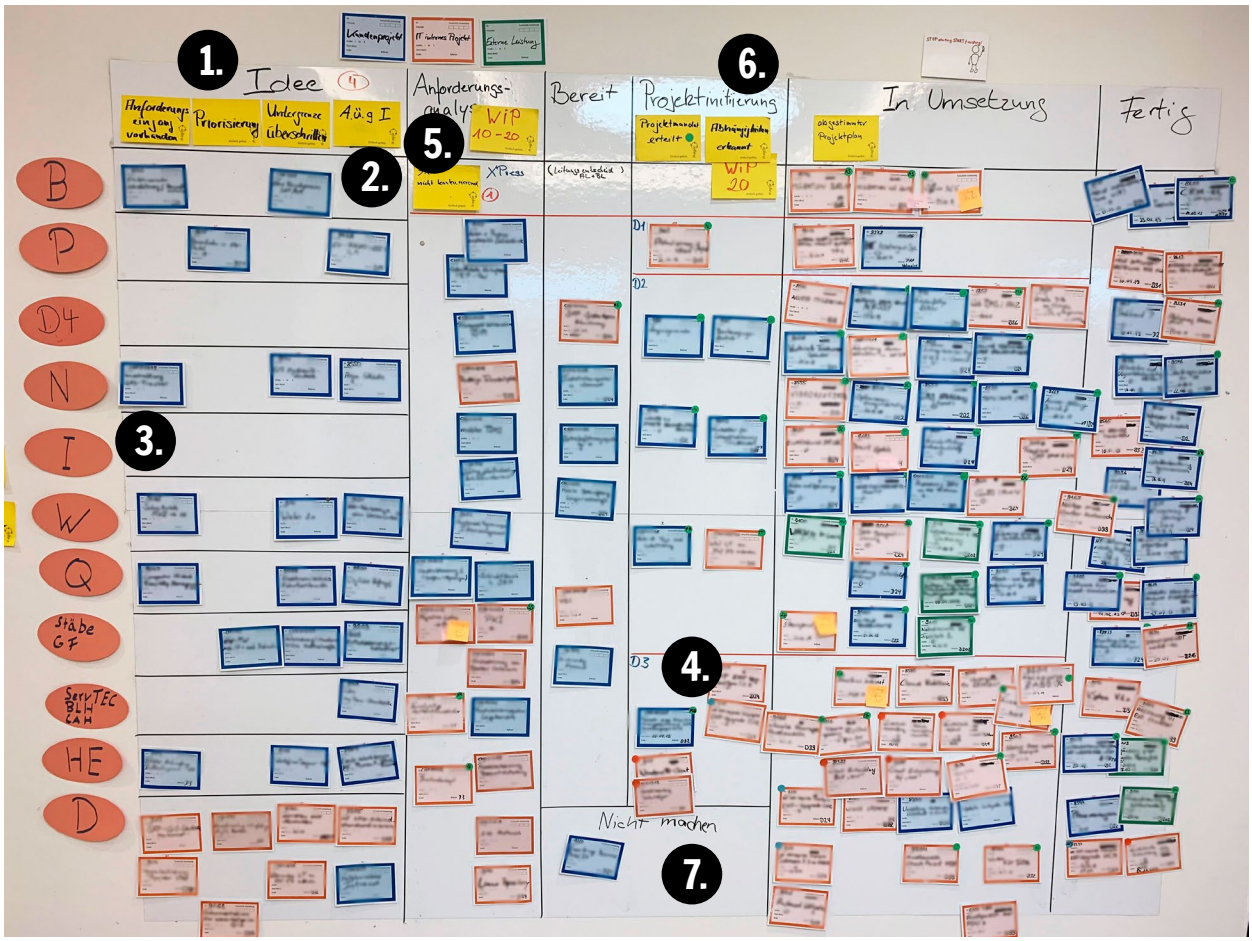
Agile Transition

http://www.it-agile.de/transition

Behind the Board – Portfoliomanagement bei HAMBURG WASSER

HAMBURG WASSER ist im Großraum Hamburg zuständig für die Stadtentwässerung und die Wasserversorgung. Dazu betreibt das Unternehmen eine IT-Abteilung, die die internen Projekte durchführt. Die Kunden der IT-Abteilung sind verschiedene Fachabteilungen des Unternehmens, z. B. Netze, Personalabteilung oder Betriebswirtschaft.
Um die Transparenz über laufende oder demnächst zu startende Projekte zu erhöhen, wurde ein Portfolio-Kanban-System erstellt. Von Sven Günther

Das Board befindet sich frei zugänglich in einem Flur des Unternehmens. Die exponierte Lage lässt immer wieder Interessenten anhalten und Fragen stellen. Der Aufbau sieht folgendermaßen aus:



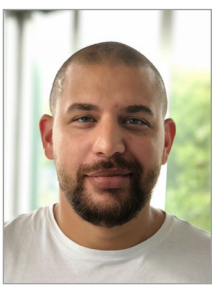
it-agile

Entwicklung, Beratung, Schulung

Das Kanban Maturity Model

Das KMM bietet eine bewährte Roadmap mit Anleitungen zu empfohlenen Vorgehensweisen zur Erreichung von Verbesserungszielen.
Es bildet typische Kanban-Praktiken sowie kulturelle Werte anhand von 7 Reifegraden der Organisation ab. Dies wird zu einem leistungsstarken Instrument für Trainer und Berater, die Kanban-Initiativen leiten und Unternehmen dabei helfen, ihre Agilität zu verbessern.

Lernen Sie dazu mehr in unserer Schulung zum Kanban Maturity Model.



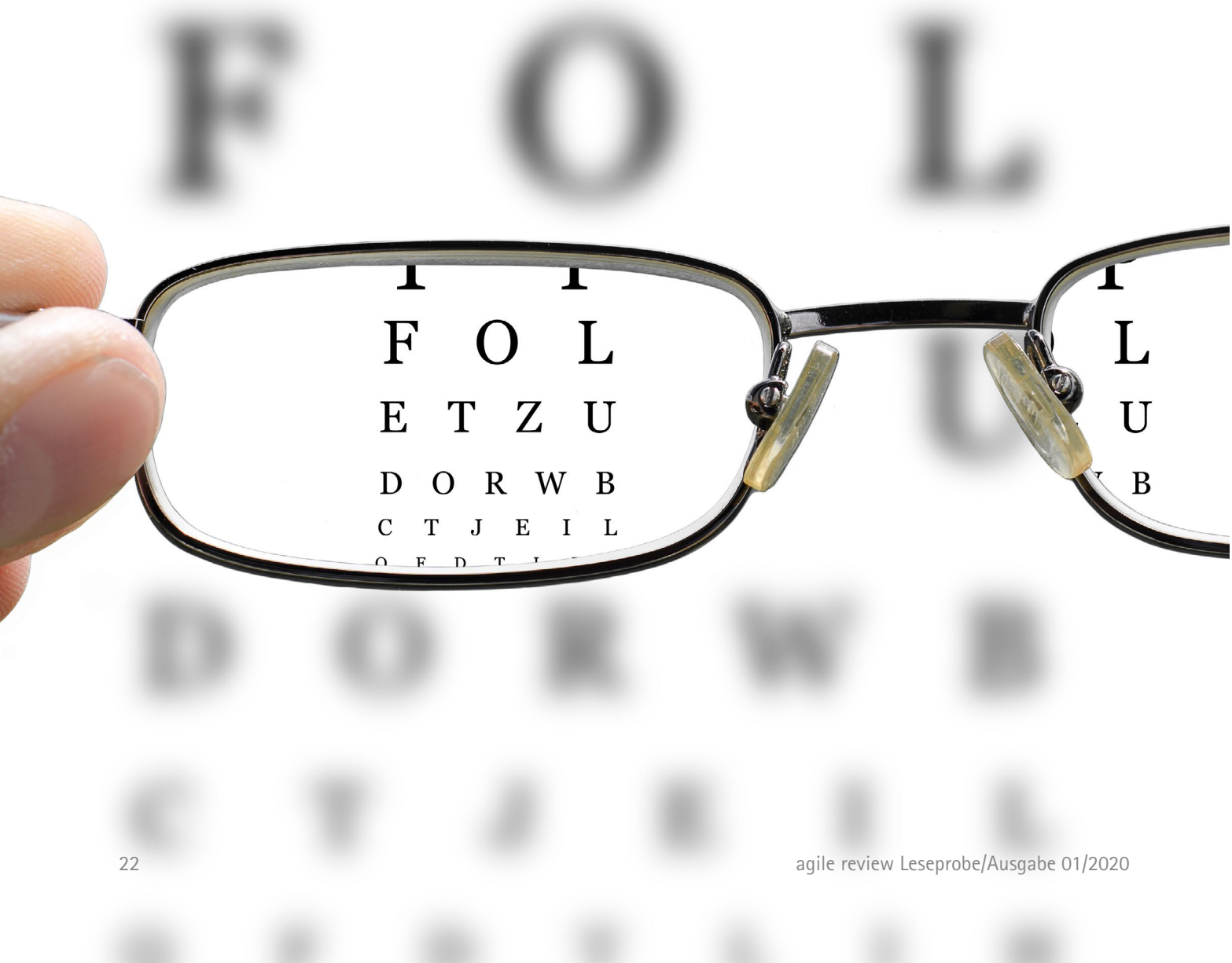
Unsere Trainer:
Sven Günther und Wolfgang Wiedenroth

http://www.it-agile.de/transition

Strategie für Klarheit

Fehlende Klarheit, fehlende Motivation

Strategie leidet in vielen Organisationen an einem von drei Symptomen: Entkopplung („die da oben"), Starre („Der Plan ist das Ziel") oder fehlende Kommunikation und blinde Flecke. In diesem Artikel stelle ich einen Ansatz vor, mit dem ich bei Unternehmen Strategie wieder zum Leben bringe. Strategie heißt jetzt Machen und ist stetig an Entscheiden und Arbeiten gekoppelt. Ich erkläre die unterschiedlichen Ebenen von Markern (die Identität des Unternehmens), Optionen (Entscheidungen und Commitments) und Arbeit und wie man diese miteinander koppelt, ohne ihre wertvollen, unterschiedlichen Eigenschaften zu beeinträchtigen. Von Markus Andrezak



Rahmen setzen mit Idealized Design und Produktvisionen

Dieser Artikel verdeutlicht, wie mächtig das Setzen von Rahmen darin ist, um Innovationen zu fördern, ohne selbstorganisierten Teams konkrete Lösungen vorzugeben.
Idealized Design ist ein Methodenklassiker von Russel L. Ackoff. Er schlägt vor, Probleme von der optimalen Lösung her rückwärts zu betrachten: „Stelle dir die ideale Lösung vor – was müssten wir heute tun, damit diese real wird?“ Eine solche inverse Denkrichtung lädt dazu ein, über den Tellerrand zu schauen, statt sich von hausgemachten Hindernissen aufhalten zu lassen. In diesem Artikel wird Idealized Design mit dem Konzept der Produktvision kombiniert. Während die Produktvision eine Abstraktion ist, die den Lösungsraum öffnet, konkretisiert Idealized Design ihn. Beide Methoden erreichen dies durch das Setzen von Rahmen und Randbedingungen.

Von Ralf Lethmate

it-agile shop

Der Online-Shop für agiles Lernen und Lehren.

STANDUP-BÄLLE

RETROSPEKTIVENBECHER

SCRUM-BIERDECKEL

PLANNING-POKER-KARTEN

AGILE REVIEW-AUSGABEN

POSTKARTENSERIE

Weitere Angebote finden Sie unter www.itagileshop.de

Auf einen Kaffee mit ...
... Laura Austermann

In dieser Reihe lernen Sie it-agile-Kollegen und ihre Sicht auf die (agile) Welt kennen. Beim letzten Mal hat Jan Ernsting erzählt, was Kaffeehandwerk und Software Crafting miteinander zu tun haben, dieses Mal stellt sich Laura Austermann den Fragen der Kollegin.

Das Gespräch mit Laura führte Nadine Wolf

it-agile

Immer noch nicht genug?

Dann abonnieren Sie doch den **Newsletter** von it-agile!

it-agile Newsletter Februar 2020

Agile Rollen, Ausbildungscoaching, Zeitmanagement, Liefern, Agiler Tipp, Leadership des Monats

Guten Tag,

traditionell rufe ich im Februar gerne den Fokus-Monat aus, schließlich stehen meist nur 28 T
zur Verfügung, aber dieses Jahr wird es ja viel entspannter. Oder doch nicht? Ich habe noch
hand vor in diesem Monat und bin ganz froh, dass ich unsere Broschüre mit Zeitmanagemen
wiedergefunden habe, die du unten zum Download findest. Mein Lieblingstipp: Nein-Sagen (mit
Varianten).

In diesem Newsletter findest du die folgenden Themen:

- Artikel-PDF zum kostenlosen Download: Agile Rollen
- Ausbildungscoaching
- Zeitmanagement-Tipps
- Dauerhaft & verlässlich schnell liefern
- Agiler Tipp: Umgang mit individuellen Zielen
- Aktuelle Schulungstermine
- Leadership des Monats: Wann zur Rechenschaft ziehen?

Viel Spaß und Lernen beim Lesen und einen fokussierten Februar,
Henning

P.S.: Am 7./8. April bieten wir wieder ein Forecasting-Training mit Troy Magennis an (auf Englisch). Das ist super-spannend und sehr empfehlenswert. Hier findest du die Beschreibung.

informativ

aktuell

auf den Punkt

Leadership des Monats

Einfach anmelden auf: www.it-agile.de

Erfolgslieferanten

it-agile