Ausgabe 04|2022

Deutschland € 12,90 Österreich € 13,90 Schweiz sfr 22,20

4 194164 112900 04

vormals OBJEKTspektrum

IT Spektrum

Digitaler Wandel & Software-Architektur für Profis

www.ITSpektrum.de

Teamzusammenarbeit gestalten

Selbstorganisation? Einfach mal ausprobieren!

– Ein Artikel von Melanie Brunnbauer und Maximilian Aulinger –

Teamzusammenarbeit gestalten: Selbstorganisation? Einfach mal ausprobieren!

DIP um IOSP ergänzen

Das Dependency Inversion Principle richtet Schaden an

Teamzusammenarbeit gestalten

Selbstorganisation – einfach mal ausprobieren

Neue Serie Architektur-Porträt

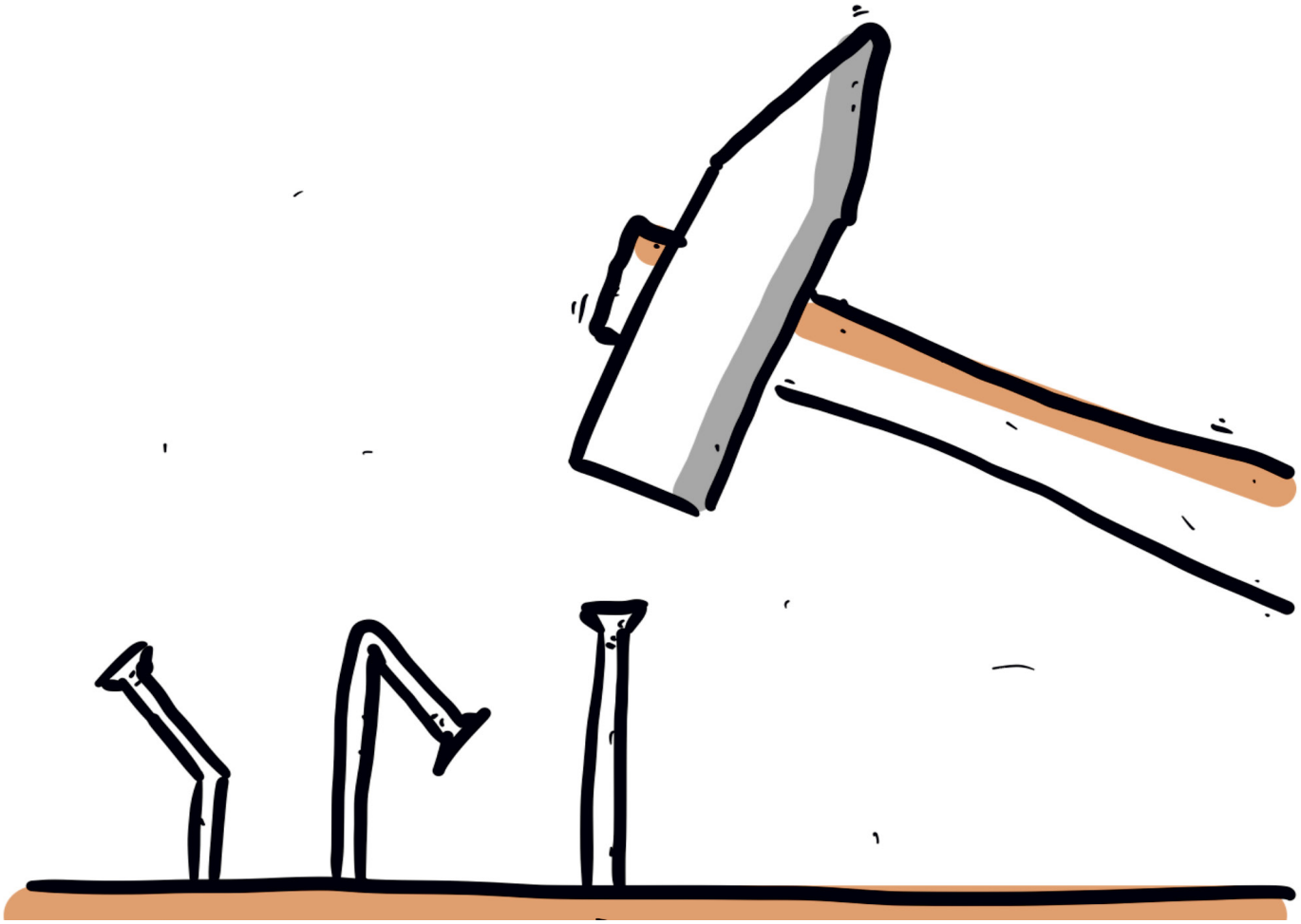
Was der Quelltexteditor Visual Studio Code kann

G 6540F

Teamzusammenarbeit gestalten

Selbstorganisation? Einfach mal ausprobieren!

Was hilft, den eigenen Verantwortungsfreiraum zu erkunden und zu nutzen? Nach unserer Erfahrung: Mutig ausprobieren! Wir stellen drei Experimente vor, die verschiedenen Entwicklungsteams geholfen haben, konstruktiv zusammenzuarbeiten und gemeinsam bessere Software zu liefern. Vom „Just-in-Time Task Breakdown“ über „Double Pairs“ bis zum „90 Minutely“ beleuchten wir die Motivation hinter den Experimenten, die Perspektiven der Beteiligten und was wir daraus gelernt haben.



Selbstorganisation organisiert sich nicht von selbst: Wir beobachten viele Teams in der Softwareentwicklung, für die Selbstorganisation eine Art luftleerer Raum ist. Was darf das Team entscheiden, was sollte es entscheiden, was möchte es überhaupt entscheiden? Aus unserer Sicht sind Mut, Verantwortungsbewusstsein und Kreativität notwendig, um erfolgreich und motiviert zusammenzuarbeiten. Nichts davon etabliert sich nach unserer Erfahrung „einfach so“ und ohne entsprechende Unterstützung.

Das zeigt sich bei einem Blick auf typische Abläufe in einem Entwicklungsteam. Der Sprint ist geplant, die Sprint Backlog-Elemente sind in Tasks heruntergebrochen, auf ein Sprint-Ziel wurde verzichtet. Das Team hat sich aufgeteilt und arbeitet parallel an den verschiedenen Backlog-Elementen, entweder einzeln oder in Kleingruppen. Im Laufe des Sprints kommt im Daily immer mal wieder zur Sprache, dass es bei einem Backlog-Element Schwierigkeiten mit der Umsetzung gibt oder dass sein Umfang doch größer ist als geschätzt. Viel Resonanz erzeugen solche Aussagen meist nicht. Manche Teammitglieder sind gedanklich bei anderen Themen und können die Problematik nicht im Detail nachvollziehen. Andere versuchen zwar, Ideen zu geben, ihre Kenntnisse der speziellen Materie reichen jedoch nicht aus, um wirklich hilfreich zu sein. Am Ende des Sprints ist dann mehr als die Hälfte des Sprint Backlog fertig umgesetzt, nur leider nicht die am höchsten priorisierten Backlog-Elemente.

Ein herzliches Schulterklopfen mit dem Kommentar „aber schaut mal, was wir sonst alles geschafft haben“ hilft, darüber hinwegzusehen. Wochen später, in einem nachfolgenden Sprint wird die Umsetzung eines Backlog-Elements infrage gestellt. Der Code liefert zwar die gewünschte Funktionalität, aber eigentlich hätte man es auch anders implementieren können, sollen oder vielleicht sogar müssen?

Genau betrachtet hat sich das Team formal gemeinsam mit den Backlog-Elementen befasst, genauso formal verlief die Umsetzung als Teamarbeit. Tatsächlich besteht das Team aber lediglich aus einer Summe einzelner Teile. Aus der Zusam-

menarbeit dann zu profitieren, wenn es am dringendsten nötig wäre, können diese „Teile“ schlecht oder schlimmstenfalls gar nicht.
In solchen Teams ermuntern und inspirieren wir bewusst durch das Einstreuen von Experimenten, die helfen können, die Zusammenarbeit in Teams zu stärken – hin zu einem gemeinsamen statt nur parallel verlaufenden Agieren. Wir stellen drei ausgewählte Experimente vor, die wir mit verschiedenen Teams bereits ausprobiert haben. Wir möchten damit ausdrücklich zur Nachahmung ermuntern – und dazu, mutig eigene Experimente zu erfinden!

Der Just-in-Time Task Breakdown

Der – aus unserer Sicht – große Mehrwert eines Task Breakdown im gesamten Team besteht in der gemeinsamen Lösungsfindung. Das Team hat dadurch die Chance, auf das Wissen und die Ideen aller mitdenkenden Köpfe zurückzugreifen und so einen Lösungsansatz zu finden, der vom gesamten Team getragen wird – eine wichtige Säule für die Wartungs- und Betriebsverantwortung einer Software.
Der Begriff „Task Breakdown“ beschreibt, dass ein Entwicklungsteam ein Backlog-Element zunächst in konkrete technische Aufgaben zerlegt, die zur Umsetzung erledigt werden müssen. Das Scrum-Framework [Scru] sieht diese Detailplanung als Abschluss des Sprint Planning vor. Üblicherweise bricht das Team dann sämtliche Backlog-Elemente des Sprint Backlog in Tasks herunter. In

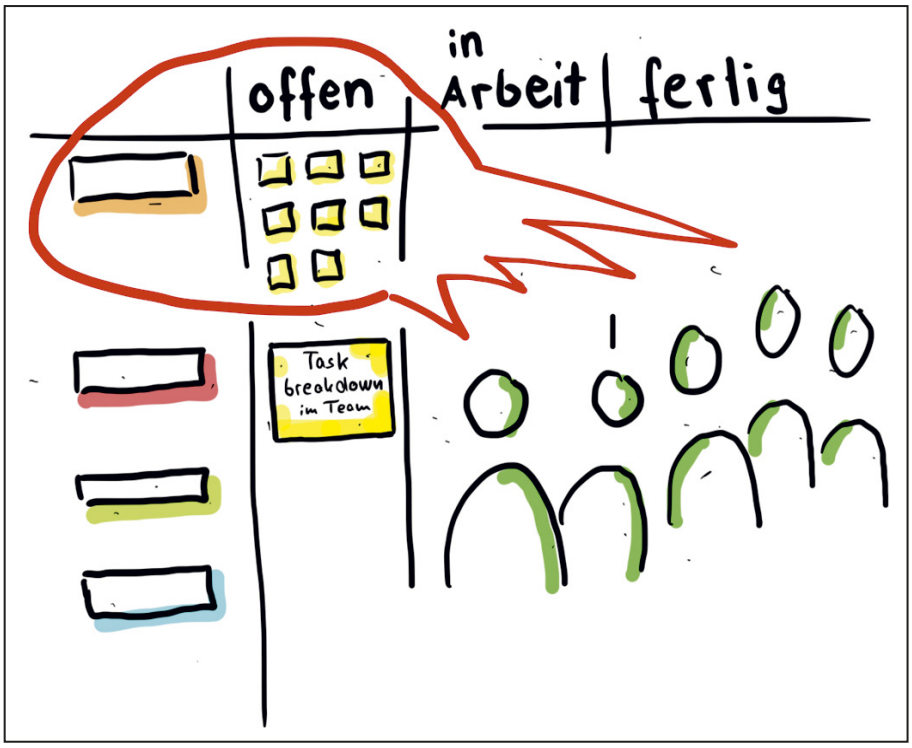


Abb. 1: Der Just-in-Time Task Breakdown: Das Sprint Backlog zu Beginn eines Sprints

der Praxis fällt es vielen Teams jedoch schwer, sich für jedes einzelne Backlog-Element bis ins Detail in die konkrete Umsetzung einzudenken – besonders bei den weiter unten stehenden Elementen im Sprint Backlog, die erst später im Laufe des Sprints umgesetzt werden. In der Folge entstehen dann oft generische und wenig aussagekräftige Tasks (implementiere Backend, implementiere Frontend, teste).
Im Laufe des Sprints stehen die Teammitglieder dann vor Fragen wie „Was haben wir noch mal mit diesem Task hier gemeint?“, oder auch „Wie sollen wir das konkret implementieren?“. Im Idealfall setzt sich das Team jetzt noch mal zusammen und klärt offene Punkte gemeinschaftlich. In weniger idealen Fällen treffen jedoch Einzelpersonen im Alleingang wichtige Entscheidungen, die andere Teammitglieder dann im Nachhinein für nicht tragfähig halten.
Unser Experiment setzt nun einen Just-in-Time Task Breakdown anstelle der oben beschriebenen Herangehensweise. Wir haben das mit einem Team ausprobiert, das sich von dieser Just-in-Time-Variante mehr Effizienz und Effektivität erhoffte. Im Sprint Planning haben wir uns bewusst darauf beschränkt, nur die ersten Backlog-Elemente in Tasks herunterzubrechen, und uns die Zeit genommen, im gesamten Team ausführlich über die konkrete Umsetzung zu diskutieren. Das nächste Backlog-Element haben wir mit dem Task „gemeinsamer Task Breakdown im Team“ versehen, als Erinnerung, dass wir uns vor der Umsetzung wieder gemeinsam eine Detailplanung machen wollen (**siehe Abbildung 1**).
Die Fokussierung auf ein oder wenige Backlog-Elemente im Task Breakdown hat zu einem intensiven Austausch geführt, bei dem unser Team ein sehr konkretes Bild entwarf, wie es die Implementierung gestalten wollte. Die Tasks konnten wir oftmals so schneiden, dass sie sogar zeitlich parallel bearbeitet werden konnten. Nach dieser Detailplanung waren alle gedanklich tief im Thema, daher war es dann nur folgerichtig, auch gleich die Ärmel hochzukrempeln und als Team mit möglichst viel Menschenstärke die Implementierung in Angriff zu nehmen. So blieb die Anzahl der gleichzeitig in der Umsetzung befindlichen Backlog-Elemente klein und der Fokus im Team scharf – es gelang uns leichter, die am höchsten priorisierten Backlog-Elemente erfolgreich umzusetzen (**siehe Abbildung 2**).
Dieses Experiment haben wir kurz vor der Corona-Pandemie ausprobiert, als wir noch in einem Raum zusammengearbeitet

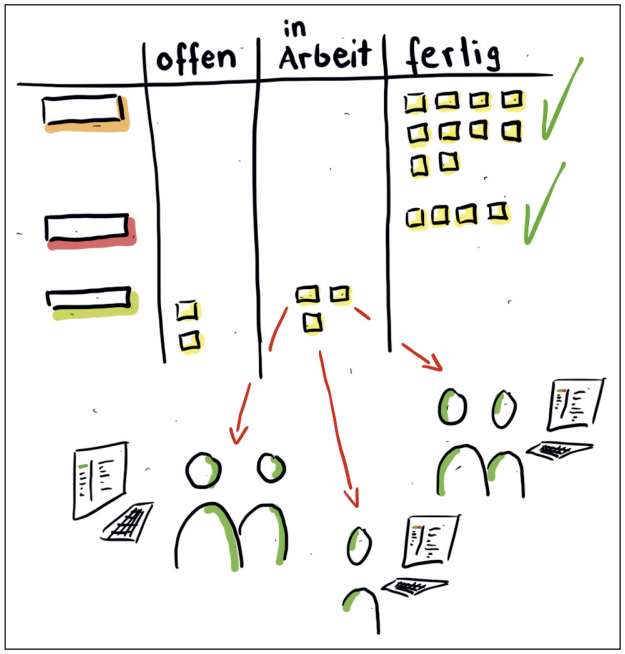


Abb. 2: Fokussierte Zusammenarbeit auf Basis eines gemeinsamen Lösungsverständnisses

haben. Dadurch hatten wir zum einen eine starke gemeinsame Taktung (z. B. im Hinblick auf Arbeitspausen), zum anderen war für jeden ersichtlich, wer gerade mitten in einer konzentrierten Arbeitsphase steckte und besser in Ruhe gelassen werden sollte. Das hat es uns erleichtert, uns spontan bei jedem neuen Backlog-Element für den gemeinschaftlichen Task Breakdown zusammenzufinden, ohne gerade ablaufende Arbeitsprozesse unnötig zu stören.

Die Erfahrung im Team *zusammen* zu arbeiten – und der gemeinsam erlebte Erfolg, Backlog-Elemente schneller fertigstellen zu können, hat sich ausgezahlt, die Zufriedenheit im Team stieg deutlich.

Dann verschob sich die Zusammenarbeit in unserem Beispielteam mit Beginn der Pandemie vollständig auf Remote, wodurch die gemeinsame Taktung und der direkte Draht zueinander ein Stück weit verloren gingen. Zeitgleich hatte sich auch die Zusammensetzung des Teams stark geändert. Beides führte dazu, dass die Kultur des Just-in-Time Task Breakdown aus dem Team weitgehend verschwand – nicht aufgrund einer gewollten Entscheidung im Team, sondern als schleichende Folge der Vielzahl an Veränderungen.

Wir haben daraus gelernt, dass der Just-in-Time Task Breakdown insofern ein fragiles Konstrukt ist, als er von zwei Dingen abhängt: Zunächst muss es mindestens einen Kern an Teammitgliedern geben, die auf diese Weise zusammenarbeiten wollen und sich aktiv für seinen Erhalt einsetzen. Außerdem muss es dem Team grundsätzlich möglich sein, sich mehr oder weniger spontan zu treffen. Das kann, je nach den Rahmenbedingungen, deutlich erschwert sein, etwa aufgrund häufiger Zu- oder Abgänge im Team, stark variierender Anfangs- oder Pausenzeiten der einzelnen Teammitglieder oder einer generellen räumlichen Trennung, vielleicht sogar über verschiedene Zeitzonen hinweg. Dennoch sind solche Herausforderungen lösbar, eventuell mittels Team-spezifischer Varianten des Experiments. Ganz wichtig ist, dass sich die Menschen im Team bewusst dafür entscheiden, die Kultur einer gemeinschaftlichen Lösungsfindung im Task Breakdown etablieren zu wollen.

Wann eignet sich das Experiment?

- Hoch-priorisierte Backlog-Elemente werden oft nicht fertig, andere aber schon.
- Die Teammitglieder haben eine hohe Überschneidung ihrer Arbeitszeit, um sich spontan zusammenfinden zu können.

Was kann durch das Experiment erreicht werden?

- Das gesamte Entwicklungsteam weiß, wie ein Backlog-Element umgesetzt wird, und trägt die Lösung mit.
- Backlog-Elemente werden so zügig wie möglich fertig, weil das Team fokussiert mit maximal möglicher Menschenstärke daran arbeiten kann.

Wie kann das in der Praxis aussehen?

- Im Sprint Planning wird nur das erste Backlog-Element in präzise formulierte Tasks heruntergebrochen, die möglichst unabhängig voneinander bearbeitet werden können.
- Das Team setzt mit möglichst vielen Mitgliedern die Tasks dieses Backlog-Elements um.
- Wenn an diesem Backlog-Element keine Arbeit mehr übernommen werden kann, entscheidet das Team, ob es mit der Umsetzung des nächsten Backlog-Elements beginnt. Dazu trifft sich das Team erneut zum gemeinschaftlichen Task Breakdown.

Abb. 3: Die Double Pairs: Ausloten verschiedener Lösungsansätze mit zwei parallel arbeitenden Pairs

Stimmen aus dem Team

„Im Nachhinein würde ich sagen, es ist besser, konsequent zu sein und wirklich jede Story gemeinsam im Team zu besprechen."

„Zusammenfassend war das Beste, dass wir versucht haben die Storys so zu schneiden, dass möglichst alle im Team an derselben Story arbeiten können."

„Es war uns wichtig, dass wir das Wissen aller Teammitglieder abzapfen können, um zu einer möglichst guten Lösung zu kommen."

Double Pairs: Wir treffen uns zum Duell im Morgengrauen

Konkurrenz und Wettbewerb erscheinen uns in einem Team eher deplatziert. Bei den Double Pairs bezieht sich der Wettbewerb dementsprechend keineswegs auf die Mitglieder des Teams, sondern auf konkurrierende Lösungsansätze für die gleiche Aufgabe. Dabei muss sichergestellt sein, dass ein Team autonom entscheiden darf, wie es vorgehen möchte, und der „Wettbewerb" eine Team-interne Aktion ist, die keinerlei Bewertung von außen erfährt.

Dann kann das Experiment sehr hilfreich sein, wenn es schwierig ist, einen genauen Lösungsweg für die Umsetzung eines Backlog-Elements zu skizzieren und die Detailplanung anzugehen. Konkurrierende Umsetzungsmöglichkeiten machen es unmöglich, im Vorfeld zu entscheiden, welche davon die beste Wahl ist. Vielleicht kann sich das Team auch nicht auf einen Lösungsansatz einigen. In diesem Fall besteht der erste Schritt zur Umsetzung darin zu klären, welche Richtung das Team einschlagen möchte.

Genau das ist die Schwierigkeit? Dann sollte die Entscheidung zugunsten einer Lösung vertagt werden zugunsten des parallelen Ausprobierens verschiedener Varianten. Das ist der Ansatz der Double Pairs. Darauf kamen wir nach der scherzhaft gemeinten Frage „Duelliert ihr euch jetzt? Zwei gegen Zwei?", die Max als Scrum Master vor Jahren in „seinem" Entwicklungsteam stellte.

Das Team hat diese Idee prompt aufgegriffen und ausprobiert, bei Themen mit viel Unsicherheit mit mehreren konkurrierenden Pairs im Wettbewerb zu arbeiten. Die Zielsetzung dabei ist, schnell zu lauffähigen, vergleichbaren und damit bewertbaren Lösungsvorschlägen zu kommen (**siehe Abbildung 3**).

Mit einem Pair ist in Anlehnung an den Begriff „Pair Programming" [Bec04] in diesem Fall eine Kleingruppe aus zwei Personen gemeint, die in vertrauensvoller Zusammenarbeit gemeinsam Software implementieren.
Die Lösungsvorschläge sollten nur so weit implementiert werden, wie zur Entscheidungsfindung nötig. Dazu ist es erforderlich, mit klaren Abbruch- oder Akzeptanzkriterien zu arbeiten. Fragen wie „Was wollen wir lernen?" und „Woran erkennen wir, dass wir unser Lernziel erreicht haben?" können helfen, um diese Kriterien zu definieren. Sonst könnte ein Pair im Flow aus Versehen doch mal den Ansatz fertig bauen. Das jedoch wäre pure Verschwendung, wenn dieser Ansatz anschließend verworfen wird. In diesem Fall würde aus dem Experiment eine Methode, um Geld zu verbrennen!
Klar definierte Kriterien ermöglichen den Fokus auf das Wesentliche, sodass vergleichbare Minimallösungen entstehen. Im Idealfall kann das sogar eine kleine automatisierte Testsuite sein. Finden sich keine passenden Kriterien, lohnt es sich in jedem Fall, mit einem Abbruchkriterium zu arbeiten, beispielsweise einem vorgegebenen Zeitfenster.
Eine zentrale Fragestellung bei der Anwendung der Double Pairs ist selbstverständlich, ob ich es mir leisten kann – oder besser, ob es sich lohnt, es sich zu leisten –, eine sowieso schon teure Entwicklung doppelt zu machen. Ein paar Gedanken zu entstehenden Kosten und Möglichkeiten eines Lösungsansatzes sollen bei der Entscheidung helfen. Wenn ein Team mehrere Ansätze in der Lösungsfindung parallel verfolgt, ermöglicht das

- *eine objektive Gegenüberstellung der Lösungsvorschläge:* Implementierte Lösungsvorschläge lassen sich bewerten (z. B. Messen der Performance), testen und sind dadurch vergleichbar.
- *eine beschleunigte Lösungsfindung:* Wenn sich ein Lösungsansatz als Sackgasse entpuppt, verfolgt das Team zeitgleich noch mindestens einen vielversprechenden weiteren Ansatz.
- *eine vertrauensvolle Zusammenarbeit:* Durch die Bearbeitung jedes Lösungsansatzes in Zweiergruppen ist die Wahrscheinlichkeit, dass jeder bei der Lösungsfindung mitdenkt und sein Wissen einbringt, höher als in einer größeren Gruppe. Das begünstigt, dass sowohl das zur Verfügung stehende Wissen als auch die kreative Lösungsfindung aller Beteiligten eingesetzt werden kann.
- *eine größere Akzeptanz der Lösung im umsetzenden Team:* Durch den konkreten Vergleich der verschiedenen Lösungsvorschläge lassen sich Vorbehalte leichter ausräumen.
- *eine bessere Argumentationsfähigkeit für die gewählte Lösung auch außerhalb des Teams:* Falls der gewählte Lösungsansatz beispielsweise von üblichen Architekturmustern oder Unternehmensvorgaben abweicht und deutliche Vorteile in der Erfüllung der Anforderung bietet.

Aus dem Experiment wurde eine lebendige Teampraxis, die bis heute weiterlebt.
Aus Sicht des Teams bleiben Entscheidungen, die auf einer validen, bestätigten Basis getroffen wurden, die Erfolge der umgesetzten Lösungen und auch die Erkenntnis, dass das Experiment manche frühere Fehlentscheidung hätte verhindern können, die später ein teures Refactoring nach sich zog.

Wann eignet sich das Experiment?

- Konkurrierende Lösungsansätze – auch scheinbar abstruse – sind denkbar und es ist unklar, welcher besser geeignet oder leichter umzusetzen ist.
- Das Team hat mit einem Lösungsansatz bereits begonnen und ist in irgendeiner Form steckengeblieben.

Was kann durch das Experiment erreicht werden?

- Ein belastbarer Lösungsvorschlag in kurzer Zeit.
- Ein Teamkonsens zur Auswahl des Lösungswegs.

Wie kann das in der Praxis aussehen?

- Das Team skizziert die Lösungsoptionen.
- Es werden gemeinsam geeignete Abbruch- oder Akzeptanzkriterien festlegt. Beispielsweise das Erreichen eines Durchstichs (der Code muss nicht Produktionsreife haben, sondern lediglich beweisen, dass der Ansatz funktioniert), oder eine Zeitbeschränkung.
- Das Team teilt sich in kleine Gruppen (z. B. in Pairs) und beginnt mit der Umsetzung der verschiedenen Ansätze.
- Optional, falls es um Geschwindigkeit geht: Der erste funktionierende Lösungsansatz „gewinnt" und wird dann fertig entwickelt.

Stimmen aus dem Team

„Für mich ist dieser Ansatz ein Schritt in Richtung evidenzbasiertes Arbeiten."
„Es hat sich wie eine positive Form eines Wettbewerbs angefühlt. Ich glaube, das kann man nicht in jedem Team machen. Für uns war es ein sportliches Messen."
„Durch die mehrmalige Durchführung des Experiments ist auch ein Eindruck an Gegenbeispielen entstanden, bei denen dieser Ansatz im Nachhinein betrachtet geholfen hätte."

Vom Daily zum 90 Minutely

Aus unserer Erfahrung liefert das Daily Scrum für das Team den höchsten Nutzen, wenn es als gemeinsames Planungstreffen verstanden wird. Geplant werden die nächsten Schritte, um einen signifikanten Fortschritt hin zu einem Ziel zu machen, sei es die Fertigstellung eines Backlog-Elements oder auch das Sprint-Ziel.
Alle, die einen Beitrag zu diesem Fortschritt leisten, sind Teilnehmer und nutzen das Daily Scrum, um erreichte Teilschritte, Erkenntnisse und erkannte Hindernisse miteinander zu teilen. Diese Informationen dienen als Grundlage für eine bessere Planung der nächsten Schritte zur Umsetzung der Lösung.
Der Planungshorizont reicht dabei bis zum nächsten Daily Scrum. In der Regel also 24 Stunden. In manchen Situationen kann ein Entwicklungsteam von deutlich kürzeren Abständen zwischen solchen Abstimmungsmeetings profitieren.
Jasmine Zahno und Joseph Pelrine machten in einem Vortrag auf der OOP 2019 [Zah19] aufmerksam auf ein mutmaßliches Muster in der menschlichen Aufmerksamkeitsspanne, das als „Basic rest-activity cycle" (BRAC) bezeichnet wird. Ganz grob und stark vereinfacht dargestellt, beschreibt dieser Zyklus eine Zeitspanne von 80 bis 120 Minuten, in der sich eine Phase mit hoher Leistungs- und Konzentrationsfähigkeit und Ruhephasen abwechseln. Nach ca. 90 Minuten ist eine Pause oder zumindest ein Wechsel der Aktivität notwendig. Der Körper holt sich diese Ruhephase oder Abwechslung in irgendeiner Form selbst, wenn er keine äußere Möglichkeit dafür erhält.
Auf dieser Basis entstand die Idee, die Teamarbeit entsprechend zu takten: ca. 90 Minuten stehen für das – konzentrierte – Implementieren zur Verfügung. Danach folgt durch die gemeinsame Reflexion und Planung ein Wechsel in eine andere Form der Aktivität.
Eines unserer Entwicklungsteams war zu diesem Zeitpunkt in einer Phase großer Unsicherheit – insbesondere in Bezug auf technisch mögliche und tragfähige Lösungsansätze. Die Bereitschaft im Team war dadurch groß, eine engere Taktung für Austausch und Planung nächster Schritte auszuprobieren. Dabei stellen die 90 Minuten, auf die sich das Team ei-

Literatur & Links

[BecO4] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 2004

[Scru] Scrum Guide, siehe: **https://scrumguides.org/scrum-guide.html**

[Zah19] J. Zahno, J. Pelrine, Seven (plus/minus two) ways your brain screws you up, Konferenzbeitrag OOP 2019

nigte, einen Richtwert da, der möglicherweise von individuellen Arbeitsweisen abweicht. Dennoch hoffte das Team, mit dem Experiment technologische Sackgassen schneller zu erkennen und bei der Implementierung neu gesammelte Erkenntnisse frühzeitig im Team zu teilen.

Dieses Experiment ist für das Team besonders hilfreich, wenn alle Teammitglieder den gleichen Kontext haben, also idealerweise vollständig an einem Backlog-Element arbeiten. Dann bildet das Experiment einen einfachen Startpunkt, um als Team enger zusammenzuarbeiten.

Ein beobachteter Nebeneffekt ist der starke Fokus, den die Teammitglieder aktiv aufrechterhalten. Das Team lässt sich durch Störungen von außen weniger leicht ablenken.

Im Zusammenspiel mit anderen Experimenten, wie dem Just-in-Time Task Breakdown und den Double Pairs, ist der sehr kurze Feedbackzyklus eine passende Ergänzung.

Wann eignet sich das Experiment?

- Wenn der im Daily gefasst Plan oft nicht eingehalten wird und das Team seine selbst gesteckten Ziele nicht erreicht.
- Wenn sich ein Team in einem Umfeld mit großer Unsicherheit bewegt und von stark verkürzten Rückmeldungszyklen profitieren kann.

Was kann durch das Experiment erreicht werden?

- Verhindert Tunnelblick bei Problemen und fördert frühe Kooperation im Team.
- Erleichtert gemeinsames Lernen, wenn sich das Team ein neues (technologisches) Gebiet erarbeitet.

Wie kann das in der Praxis aussehen?

- Zusätzlich zum „normalen" Daily bis zu vier weitere kurze Zusammenkünfte am Sprintboard im Abstand von ca. 90 Minuten einplanen (Dauer beschränkt auf max. 7 bis 10 Minuten).
- Schwerpunkt setzen, über den im 90 Minutely gesprochen wird, dies kann beispielsweise ein Task sein, der gerade in Bearbeitung ist.
- Das 90 Minutely als kurze Lagebesprechung gestalten und klären, welche Erkenntnisse man gewonnen hat und was in den nächsten 90 Minuten angegangen werden wird.

Stimmen aus dem Team

„Wir haben damit angefangen, weil wir mit vielen Tasks am Ende des Tages nicht fertig geworden sind. Viele Probleme, die während der Entwicklung frühzeitig aufgetaucht sind, wurden zu lange allein angegangen."

„Als Team haben wir das 90 Minutely auch wieder situativ angewendet, wenn wir der Meinung waren, dass es mit der gegebenen Aufgabe hilfreich ist."

„Bis heute geblieben ist ein ‚Lunchly' oder ‚Coffee-Daily' – ein zweites Daily mittags nach dem Essen. Das hilft jetzt auch in der Remote-Situation, wenn man sich sonst nicht persönlich sieht."

Fazit

Aus unserer Erfahrung sind Entwicklungsteams oft eine Ansammlung von Menschen, die zwar für einen fachlichen Kontext, aber nebeneinander her arbeiten. Teams können so das Potenzial ihrer Zusammenarbeit nicht voll ausschöpfen. Technologien entwickeln sich kontinuierlich weiter und übersteigen daher in der Regel das Wissen von Einzelpersonen. Damit ist Lernen im Team eine essenzielle Kompetenz, um gute Software zu entwickeln, vor allem um technische Möglichkeiten – und wie diese auf die fachliche Zielsetzung einzahlen können – zu bewerten und einzusetzen. Das schließt einen passenden und zeitgemäßen Werkzeugeinsatz (Infrastruktur, Pipelines, Testsysteme, IDE) ein.

Um eine solche Lernkultur zu entwickeln, haben uns die vorgestellten Experimente in den beschriebenen Situationen geholfen. Allerdings verstehen wir unsere Beispiele als reine Inspiration, nicht als 1-zu-1 übertragbare Patentrezepte – die gibt es aus unserer Sicht nicht. Wir möchten dazu anregen, selbst eigene Experimente zu gestalten, die zum jeweiligen Team und Umfeld passen. Um zu eigenen Experimenten zu kommen, können diese Fragestellungen weiterhelfen:

- Was möchten wir in der Zusammenarbeit erreichen?
- Was können wir ausprobieren, um dieser Zielsetzung näher zu kommen?
- Woran erkennen wir, dass unsere Zusammenarbeit erfolgreich ist?
- Wann ist es sinnvoll, das Experiment zu bewerten?

Wir wünschen viel Spaß und ein intensives Lernerlebnis beim Ausprobieren! ||

Die Autoren

Maximilian Aulinger
(maximilian.aulinger@andrena.de)
ist Agile Coach bei der andrena objects ag. Er begleitet Softwareentwicklungsteams mit Augenzwinkern und einer Prise Lösungsfokus. Dabei steht immer das Verlassen ausgetretener Pfade, um einen höheren Nutzen zu erzielen, im Vordergrund.

Dr. Melanie Brunnbauer
(melanie.brunnbauer@andrena.de)
ist agile Softwareentwicklerin bei der andrena objects ag. Bei ihrer Arbeit in Entwicklungsteams liegt ihr nicht nur die innere und äußere Qualität des Softwareprodukts am Herzen, sondern auch die kontinuierliche Verbesserung der Zusammenarbeit im Team.