Entscheiden in selbstorganisierten Teams

„Was machen wir denn jetzt?“

Immer mehr Unternehmen setzen bei Softwareprojekten auf Hierarchieabbau und selbstorganisierte Teams. Statt der erwarteten Leistungssteigerungen gibt es jedoch häufig ein neues Problem: In der gefühlten „Basisdemokratie“ fällt das Treffen von Entscheidungen schwer. Konflikte entstehen und der Entwicklungsfortschritt verzögert sich. Dieser Artikel zeigt auf, wie selbstorganisierte Teams diese Klippen umschiffen und einen guten Umgang mit Entscheidungen finden können.

Softwareprojekte sind nichts für entscheidungsscheue Menschen. So wie man heutzutage im Supermarkt die Qual der Wahl zwischen unzähligen Marmeladensorten hat, sind auch in einem Softwareentwicklungsvorhaben viele Entscheidungen zu treffen:

- Was soll in der Produktvision stehen?
- Welche Features sollen umgesetzt werden?
- Welche Qualitätsanforderungen gelten für das Produkt?
- Wie ist die Strategie zur Einbeziehung der Nutzer?
- Welche Architekturansätze werden verfolgt?
- Und so weiter und so fort.

Entscheidungen über Entscheidungen…

Theoretisch ist das in agilen Teams ganz einfach: Der Product Owner entscheidet über das „Warum“ und das „Was“, also über alles, was mit den Anforderungen zusammenhängt. Das Dev-Team entscheidet über das „Wie“, sprich über technische Details der Umsetzung.

In der Praxis läuft der Umgang mit Entscheidungen allerdings oft anders ab, als im Scrum-Guide vorgesehen [Scru]. Häufig ist der Product Owner nicht ermächtigt, alle entsprechenden Entscheidungen zu treffen. Und auch in der Zusammenarbeit mit den Stakeholdern lauern viele Fallstricke. Sprechen Stakeholder nicht „mit einer Stimme“, wird das Herbeiführen von Entscheidungen möglicherweise zu einem langwierigen und konfliktträchtigen Unterfangen.

Hinzu kommt in komplexen Umfeldern noch ein anderes Problem: Die Folgen von Entscheidungen sind nur schwer oder gar nicht vorhersehbar. Es gibt oft kein offensichtliches Richtig oder Falsch mehr, sondern es zeigt sich erst im Nachhinein, ob eine Entscheidung gut war. Man spricht in diesem Zusammenhang auch von „retrospektiver Kohärenz“. Entscheidungen müssen damit immer mehr auf Basis von unvollständigen Informationen getroffen werden.

Experimente gegen die Entscheidungsstarre

Ich habe es schon oft erlebt, dass die Angst vor dem Scheitern Entscheidungen in Softwareprojekten mehr oder minder komplett blockiert. Über mehrere Monate finden immer wieder Diskussionsrunden und Workshops statt, ohne dass es einen Schritt vorwärts geht. Und falls die Entscheidungsergebnisse wichtige Grundlagen für das weitere Entwicklungsvorgehen sind, stockt dann natürlich auch der Entwicklungsfortschritt.

Zum Glück gibt es im agilen Kontext ein gutes Instrument gegen Entscheidungsblockaden aller Art: das Durchführen von sogenannten Safe-to-Fail-Experimenten. Die Bezeichnung *Safe-to-Fail* ist dabei potenziell missverständlich: Es geht hier nicht um Experimente, bei denen es sicher ist,

Entscheidungsmodell	Details	Vorteile	Nachteile
Autokratische Entscheidung	Der Product Owner entscheidet allein.	Schneller Entscheidungsprozess, in Krisenzeiten gut geeignet.	Abweichende Perspektiven werden möglicherweise ignoriert; Motivation der Gruppe kann leiden, falls Widerstände missachtet werden.
Konsent	Die Entscheidung wird getroffen, wenn es dagegen keine gravierenden Einwände gibt.	Schnell und kollaborativ, passt zum iterativen Vorgehen (Lösungen müssen nicht perfekt, sondern nur gut genug sein).	Einwände werden evtl. nicht vorgebracht, da die Person den Aufwand scheut, den Einwand zu argumentieren und eine Lösungsalternative anzubieten.
Delegationsentscheidung	Der Product Owner benennt eine Person, die entscheidet (oft mit festgelegten Rahmenbedingungen).	Kapazität des Product Owners wird nicht durch die Entscheidung gebunden, positiv für Motivation des Teams.	Eventuell mehr Unterstützung des Teams notwendig, als gedacht; Product Owner ggf. mit der gefundenen Lösung nicht zufrieden.
Demokratische Entscheidung	Die Gruppe stimmt über die Entscheidung ab, die Abstimmungsmehrheit gewinnt.	Transparenter Prozess, wird als sehr fair wahrgenommen.	Kann negativ für das Gruppenklima sein (Teil des Teams fühlt sich als „Verlierer"), möglicherweise fehlende Unterstützung des Ergebnisses, wenn nicht dafür gestimmt wurde.
Konsens	Die Gruppenmitglieder einigen sich (entweder auf eine neue Lösung oder einen Kompromiss aus bestehenden Lösungen).	Perspektiven aller Stakeholder werden einbezogen, gut für den Gruppenzusammenhalt, große Unterstützung des Entscheidungsergebnisses.	Kann sehr zeitaufwendig sein, ungeeignet für große Gruppen oder bei Machtkämpfen innerhalb der Gruppe.
Konsultativer Einzelentscheid	Eine vorab vom Product Owner benannte Person entscheidet, nachdem sie bestimmte Stakeholder angehört hat.	Zusätzliche Perspektiven werden einbezogen, ermöglicht Einschätzung der politischen Auswirkungen der Entscheidung.	Nicht angehörte Personen fühlen sich möglicherweise ausgeschlossen.
Stochastische Entscheidung	Die Entscheidung wird dem Zufall überlassen (z. B. Münze werfen).	Schneller Entscheidungsprozess, geeignet bei vielen sehr ähnlichen Alternativen.	Nicht geeignet, falls Alternativen nicht ähnlich genug sind.

Tabelle 1: Übersicht über Entscheidungsmodelle

zu scheitern. Sondern um Experimente, bei denen es nicht schlimm ist, wenn man scheitert. Trotz des Scheiterns ist man danach immer noch „safe". Die Entwicklung von Wegwerf-Prototypen, A-/B-Testing oder das Ausprobieren von neuen Arbeitsweisen für einen vorerst kurzen Zeitraum sind Beispiele für solche Experimente.
Aber auch Experimente entbinden Product Owner und Dev-Team nicht von der Notwendigkeit, Entscheidungen zu treffen. Oft gibt es mehr Ideen für Experimente, als umgesetzt werden können. Hier gilt es also festzulegen, welche Experimente überhaupt realisiert werden sollen. Und auch nach Abschluss eines Experiments bleiben oft noch Fragen offen, über die entschieden werden muss. Soll ein Produkt oder ein Feature nach einem erfolgreichen Experiment realisiert werden? Und falls ja, bis wann? Kann ein nicht erfolgreiches Experiment möglicherweise mit geänderten Rahmenbedingungen doch noch zum Erfolg gebracht werden und sollte daher weiterverfolgt werden?

Entscheidungsmodelle: Decide how to decide!

Wer Entscheidungen im Team treffen will, sollte sich zuerst mit dem Thema Entscheidungsmodelle befassen. **Tabelle 1** gibt einen Überblick über die wichtigsten Modelle sowie deren Vor- und Nachteile. Grundsätzlich sind nicht-kollaborative Entscheidungsmodelle, wie die autokratische Entscheidung oder die Delegationsentscheidung, gut geeignet, wenn man schnell ein Ergebnis benötigt oder es bei der Entscheidung nicht um viel geht. Kollaborative Entscheidungsmodelle, wie der konsultative Einzelentscheid oder eine Konsensentscheidung, führen meist zu einem längeren Entscheidungsprozess, allerdings auch zu einer besseren Unterstützung des Ergebnisses durch das Team. Daher sind sie gut für Entscheidungen mit großen und langfristigen Auswirkungen geeignet.

Bevor das Team eine Entscheidung trifft, sollte es festlegen, welches Entscheidungsmodell es verwenden wird. Diese Art von Transparenz ist am Anfang meist etwas ungewohnt, kann aber mit etwas Übung gut gelingen. In diesem Zusammenhang sei auch ein Artikel von Ellen Gottesdiener empfohlen, der weitere Impulse zur Abstimmung von Entscheidungsmodellen beinhaltet [Medi].
Wer noch unsicher ist, welches Entscheidungsmodell am besten geeignet ist, kann dafür auch die Website „Decider App" konsultieren (**siehe Abbildung 1**). Diese empfiehlt nach der Beantwortung einiger

Abb. 1: Screenshot Decider App [Deci]

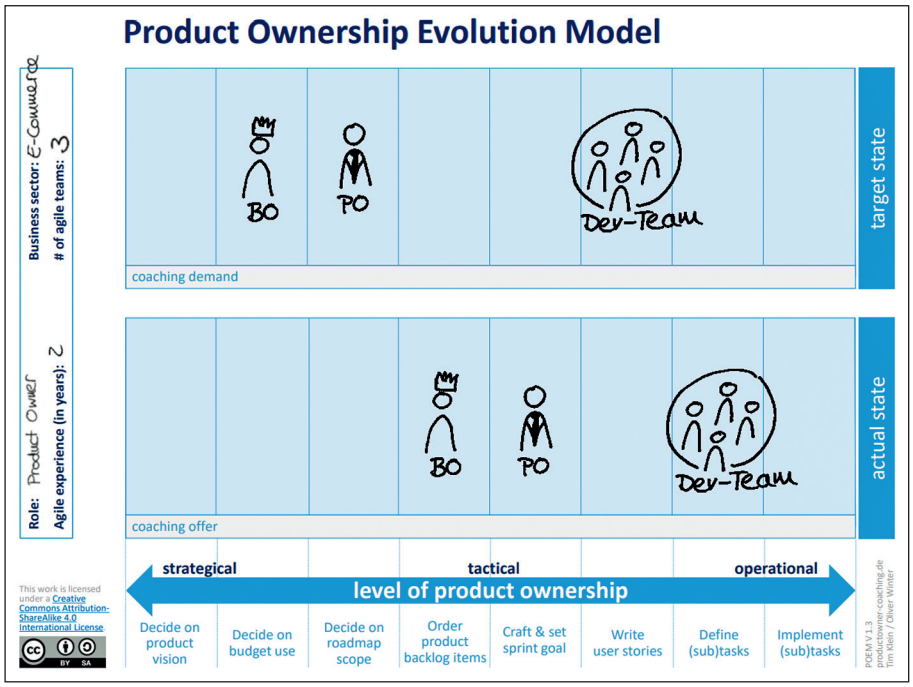


Abb. 2: Product Ownership Evolution Model (POEM) [Prod]

- *Marktanforderungen:* Kunden- und Nutzerbedürfnisse sowie Geschäftsziele; Antworten auf die Frage, warum das Produkt erstellt werden soll.
- *Produktanforderungen:* Geforderte Produkteigenschaften; Antworten auf die Frage, was das Produkt leisten soll.
- *Komponentenanforderungen:* Detaillierte Beschreibungen der geforderten Softwarefunktionalität; Antworten auf die Frage, wie das Produkt funktionieren soll.

Kasten 1: Anforderungsebenen, vgl. [Ebe14]

kurzer Fragen zur anstehenden Entscheidung ein passendes Entscheidungsmodell. Auch einen Slackbot gibt es dazu – momentan allerdings wie die Website leider nur auf Englisch.

Entscheidungsspielräume ausloten

Neben der Festlegung des Entscheidungsmodells ist es auch wichtig, die Entscheidungsspielräume des Teams und seiner Mitglieder zu klären. Bei Entscheidungen im privaten Bereich ist das Klären des Entscheidungsspielraums oft einfach: Was ich morgens anziehe, entscheide ich grundsätzlich selbst. Bei manchen Menschen redet dort allerdings auch der Arbeitgeber mit einem Dresscode oder ein Ehepartner mit einem etwas anderen Modegeschmack mit ...

Product Owner und Dev-Team können gemeinsam mit ihrem Business-Sponsor das Product Ownership Evolution Model (POEM) nutzen, um ihre Entscheidungsbefugnisse genauer auszuloten (**siehe Abbildung 2**). Es enthält die unterschiedlichen Entscheidungsbereiche, die typischerweise in den Bereich des Product Ownership fallen. Diese erstrecken sich von strategischen Themen wie der Festlegung einer Produktvision bis hin in den operativen Bereich wie die Implementierung von Subtasks.

Zunächst füllen Product Owner, Business-Sponsor und Dev-Team-Mitglieder das Modell getrennt voneinander aus. Jeder ordnet sich und seine Kollegen entsprechend den aktuellen Aufgaben und Befugnisse den Tätigkeiten zu. Unterschieden wird hierbei nach aktuellem Zustand (d. h. welche Aufgaben und Befugnisse hat jemand aktuell?) und Zielzustand (d. h. welche Aufgaben und Befugnisse sollte bzw. möchte jemand zukünftig haben?). In einem nächsten Schritt werden die individuellen Eintragungen in der Gruppe diskutiert und es werden bei Bedarf Anpassungen vorgenommen.

Auf dieser Basis können gemeinsam Entscheidungsspielräume festgelegt werden. So kann es beispielsweise ein Ergebnis der Diskussion sein, dass der Product Owner zwar den Scope des Produkts definieren, aber nicht über die Verwendung des Budgets entscheiden darf. Dafür ist der Business-Sponsor zuständig. So kann das Product Ownership Evolution Model dabei helfen, Entscheidungsbefugnisse transparent zu machen und Konfliktpotenziale zu verringern.

Konflikte können aber nicht nur zwischen Product Owner und Business-Sponsor entstehen. Gerade wenn der Product Owner selbst auch einen technischen Hintergrund hat, kann es dazu kommen, dass er dem Dev-Team in den User-Storys auch Implementierungsdetails vorgeben möchte. Diese Art von Entscheidungen sollte ein Product Owner aber tunlichst dem Dev-Team überlassen, das die technischen Möglichkeiten bedeutend besser einschätzen kann.

Eine klare Trennung zwischen Entscheidungen über den Problemraum (Product Owner) und über den Lösungsraum (Dev-Team) hilft, hier Klarheit zu schaffen. Bei der Frage, was in den Problem- und was in den Lösungsraum fällt, kann die Verwendung von expliziten Anforderungsebenen unterstützen. Gängige Modelle unterteilen Anforderungen in Markt-, Produkt- und Komponentenanforderungen (**siehe Kasten 1**).

Ein Team könnte beispielsweise explizit vereinbaren, dass der Product Owner sich in erster Linie um Marktanforderungen kümmert, während das Dev-Team für die Ableitung von Produkt- und Komponentenanforderungen zuständig ist. So kann ein mögliches „Kompetenzgerangel“ vermieden werden.

Entscheiden mit Delegation Poker

Eine weitere Möglichkeit zum Vereinbaren von Entscheidungsspielräumen bietet das Kartenspiel „Delegation Poker“ [Mana]. In einem Delegation Board werden als Erstes potenzielle Entscheidungssituationen aufgelistet (**siehe Tabelle 2**). Jedes Teammitglied enthält dann sieben Spielkarten, die die in **Kasten 2** aufgeführten Delegations-Ebenen repräsentieren.

Ähnlich wie beim Planning Poker werden nun die Entscheidungssituationen nacheinander abgearbeitet. Jeder überlegt zunächst für sich, welche Delegations-Ebene für die Situation angemessen wäre, und legt die entsprechende Karte verdeckt vor sich hin. Auf ein Signal decken alle Teammitglieder ihre Karten auf. Falls sich alle einig sind, wird die Delegations-Ebene für die Entscheidungssituation in das Delegation Board eingetragen. Falls es unterschiedliche Einschätzungen gibt, erfolgt der Austausch darüber und gegebenenfalls eine Einigung oder die Bildung eines Mittelwerts. Das ausgefüllte Delegation Board wird im Teamraum aufgehängt, sodass jederzeit transparent

ist, wie in welcher Situation entschieden werden soll.

Entscheidungsprozesse moderieren und Ergebnisse dokumentieren

Nachdem sich die Teammitglieder über Entscheidungsmodelle und -befugnisse verständigt haben, geht es nun daran, die Entscheidung auch zu treffen.
Bei kontroversen Entscheidungen kann es hilfreich sein, dass eine neutrale Person den Entscheidungsprozess moderiert. Dies kann ein Kollege aus einem anderen Team oder ein externer Moderator sein. Wichtig ist, dass diese Person kein eigenes persönliches Interesse an einem bestimmten Entscheidungsergebnis hat!
Im Entscheidungsworkshop werden zunächst die Entscheidungsalternativen herausgearbeitet und dann miteinander verglichen. Hierzu eignet sich die Vier-Felder-Tafel [Kan18], die eine systematische und neutrale Bewertung von Entscheidungsalternativen ermöglicht. Für jede Entscheidungsalternative bereitet der Moderator eine Tafel vor (**siehe Tabelle 3**). Hierzu können Flipcharts, Metaplanwände oder Post-its auf einer Wand, aber auch PC und Beamer genutzt werden.
Der Moderator geht nun mit den Teammitgliedern die Entscheidungsalternativen nacheinander durch. Im ersten Schritt bittet er darum, mögliche Vor- und Nachteile der Alternative zu nennen. Dabei sollte er darauf achten, dass sowohl positive als auch negative Aspekte genannt werden, und auch strittige Antworten notieren. Im zweiten Schritt sammelt das Team dann Ideen zum Ausgleich möglicher Nachteile, um im dritten Schritt notwendige Schritte für die Umsetzung der Entscheidungsalternative aufzulisten. Hierbei ergeben sich möglicherweise neue Nachteile, für die dann wiederum ein Ausgleich gefunden werden muss.
Abhängig davon, wie die Gruppe die gefundenen Vor- und Nachteile gewichtet, kann sich so schnell ein Lösungsfavorit herauskristallisieren. Oft finden die Teilnehmer jedoch auch neue Lösungswege als Kombination aus den betrachteten Alternativen. Diese sollten ebenfalls auf einer eigenen Vier-Felder-Tafel bearbeitet werden.
Neben dem Schaffen von Transparenz und Fairness bei der Entscheidungsfindung hat die Vier-Felder-Tafel einen weiteren Vorteil: Sie sorgt bereits während des Entscheidungsprozesses für eine Dokumentation desselben und macht die Entscheidung damit nachvollziehbar. So vermeidet man, dass im Nachhinein immer wieder Diskussionen um das Entscheidungsergebnis und verworfene Alternativen auftreten. Außerdem ist es einfacher, die Auswirkungen vorherzusehen, sollte es einmal notwendig werden, die Entscheidung zu revidieren.
Um bei einer Vielzahl von getroffenen Entscheidungen später nicht den Überblick über die Entscheidungsdokumentation zu verlieren, nutze ich mit meinem Team das Entscheidungsprotokoll des Enterprise-Wikis Confluence (**siehe Abbildung 3**). Wir erstellen für jede Entscheidung eine eigene Confluence-Seite mit dem Seiten-Template „Entscheidung“, welches wir individuell für unser Team angepasst haben. Confluence listet alle offenen und getroffenen Entscheidungen innerhalb unseres Teamspace auf einer automatisch generierten Wiki-Seite auf [Atla].

	1 Verkünden	2 Verkaufen	3 Befragen	4 Einigen	5 Beraten	6 Erkundigen	7 Delegieren
Product Backlog Management		X					
GUI-Design				X			
Softwarearchitektur					X		
Projektvorgehensmodell						X	
Auswahl von Tools (z. B. IDE)							X
Softwaredesign im Backend							X
Neue Teammitglieder einstellen			X				
Einkauf von HW / SW					X		
Urlaub						X	
Homeoffice							X

Tabelle 2: Delegation Board (erstellt mit [Mana])

- *1. Verkünden:* Autokratische Entscheidung durch eine Führungskraft.
- *2. Verkaufen:* Ebenfalls eine autokratische Entscheidung, allerdings muss der Entscheider versuchen, die Stakeholder von seiner Entscheidung zu überzeugen.
- *3. Befragen:* Konsultativer Einzelentscheid.
- *4. Einigen:* Konsensentscheid, bei dem sich die Gruppe gemeinsam auf eine Entscheidungsalternative einigt.
- *5. Beraten:* Das Team entscheidet, muss jedoch vorher die Beratung des Managers einholen.
- *6. Erkundigen:* Das Team entscheidet selbstständig, informiert jedoch über die getroffene Entscheidung.
- *7. Delegieren:* Das Team entscheidet komplett selbstständig und muss niemanden über seine Entscheidung informieren.

Kasten 2: Delegations-Ebenen aus dem Delegation Poker

Nicht zu entscheiden, ist auch eine Entscheidung

Und was ist, wenn man einfach nicht entscheiden kann oder will? Hier hilft es, sich

Welche Vorteile hat es, wenn wir uns für diese Alternative entscheiden? … … … …	Welche Nachteile hat es, wenn wir uns für diese Alternative entscheiden? … … … …
Welche nächsten Schritte wären zur Umsetzung der Alternative notwendig? … … … …	Wie könnten wir die möglichen Nachteile der Alternative ausgleichen? … … … …

Tabelle 3: Vier-Felder-Tafel als Tool für Entscheidungsprozesse

Literatur & Links

[Atla] Decisions Blueprint, siehe: **https://confluence.atlassian.com/doc/decisions-blueprint-339739406.html**

[Deci] Decide better together, siehe: **https://thedecider.app/**

[Ebe14] C. Ebert, Systematisches Requirements Engineering, dpunkt.verlag, 2014

[Kan18] A. von Kanitz, Crashkurs Professionell Moderieren, Haufe Lexware, 2018

[Mana] Delegation Poker & Delegation Board, siehe: **https://management30.com/practice/delegation-poker/**

[Medi] Decide How to Decide: Empowering Product Ownership, siehe: **https://medium.com/@ellengott/decide-how-to-decide-empowering-product-ownership-97a54be29e7**

[Prod] POEM – Product Ownership Evolution Model, siehe: **https://productownership.de/poem/**

[Scru] Der Scrum Guide, siehe: **https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-German.pdf**

[Tot19] S. Toth, Vorgehensmuster für Softwarearchitektur, Hanser, 2019

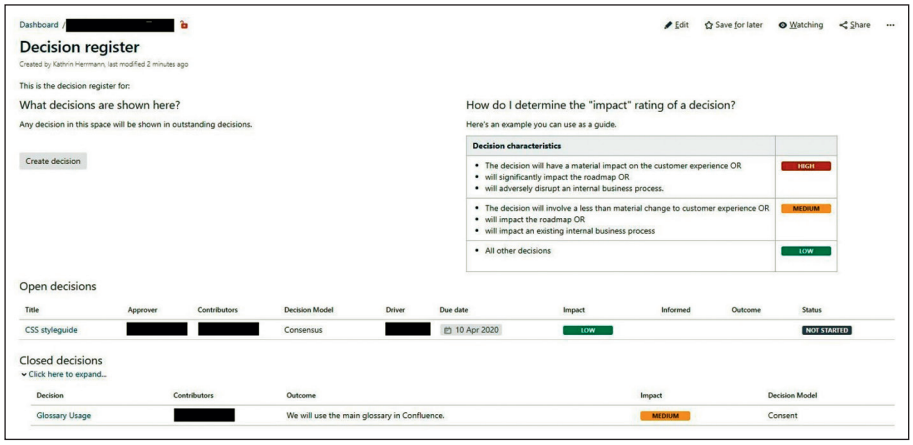


Abb. 3: Entscheidungsprotokoll in Confluence

klarzumachen, dass auch nicht zu entscheiden eine Entscheidung ist. Und das muss nicht immer schlecht sein. Manchmal erledigen sich die Dinge von selbst und mit großem Nachdruck verlangte Features stellen sich als gar nicht so wichtig heraus. So kann das „Aussitzen“ von Entscheidungen unter Umständen sogar dazu führen, Kosten einzusparen.

Im Lean Software Development ist diese Taktik auch als „Last Responsible Moment“ bekannt. Entscheidungen werden nach Möglichkeit hinausgezögert, um in der Zwischenzeit mehr über den Entscheidungsgegenstand zu lernen und so zu besseren Entscheidungen zu gelangen. Irgendwann wird jedoch die Entscheidung notwendig, weil sonst die Kosten für das weitere Hinauszögern der Entscheidung die Kosten für das Treffen der Entscheidung übersteigen würden. Beispielsweise könnte es sein, dass durch das Hinauszögern der Entscheidung, einen Client für mobile Endgeräte anzubieten, ein wichtiger Kunde abzuspringen droht. Den „Last Responsible Moment“ zu identifizieren, ist allerdings nicht trivial. Wichtig ist hierbei, den Wert und das Ablaufdatum der zur Verfügung stehenden Optionen regelmäßig zu bewerten und miteinander zu vergleichen [Tot19].

Eine andere Lösungsmöglichkeit für eine Situation, in der ein Produkt Owner nicht zwischen zwei Optionen entscheiden kann oder will, ist die Alternativenbildung. Möchte ein Stakeholder eine Funktionalität nutzen, ein anderer jedoch nicht, kann dies beispielsweise über Featureschalter realisiert werden. Die Funktion wird dann zwar implementiert, lässt sich aber über eine Einstellung (beispielsweise in einer Konfigurationsdatei) aktivieren oder deaktivieren.

Auch unterschiedliche Benutzeroberflächen für verschiedene Benutzergruppen können eine Lösung sein. In der Hafenwirtschaft habe ich es erlebt, dass die Anwender in unterschiedlichen Häfen abweichende Bezeichnungen für gleiche Dinge verwenden (so nennt man beispielsweise das befestigte Ufer im Hafen in Hamburg „Kai“ und in Bremerhaven „Kaje“). In einem Softwareprojekt wurde diesem Umstand so begegnet, dass das Team zwei separate Oberflächenübersetzungen für die Hafenstandorte realisiert hat. Über eine Konfigurationseinstellung kann die Spracheinstellung gewechselt werden.

Obwohl man so Stakeholder mit sehr unterschiedlichen Bedürfnissen glücklich machen kann, sollte man allerdings daran denken, dass dies meist zu einem höheren Entwicklungsaufwand und einer höheren Quellcodekomplexität führt. Dies hat wiederum Auswirkungen auf die Wartbarkeit der Software.

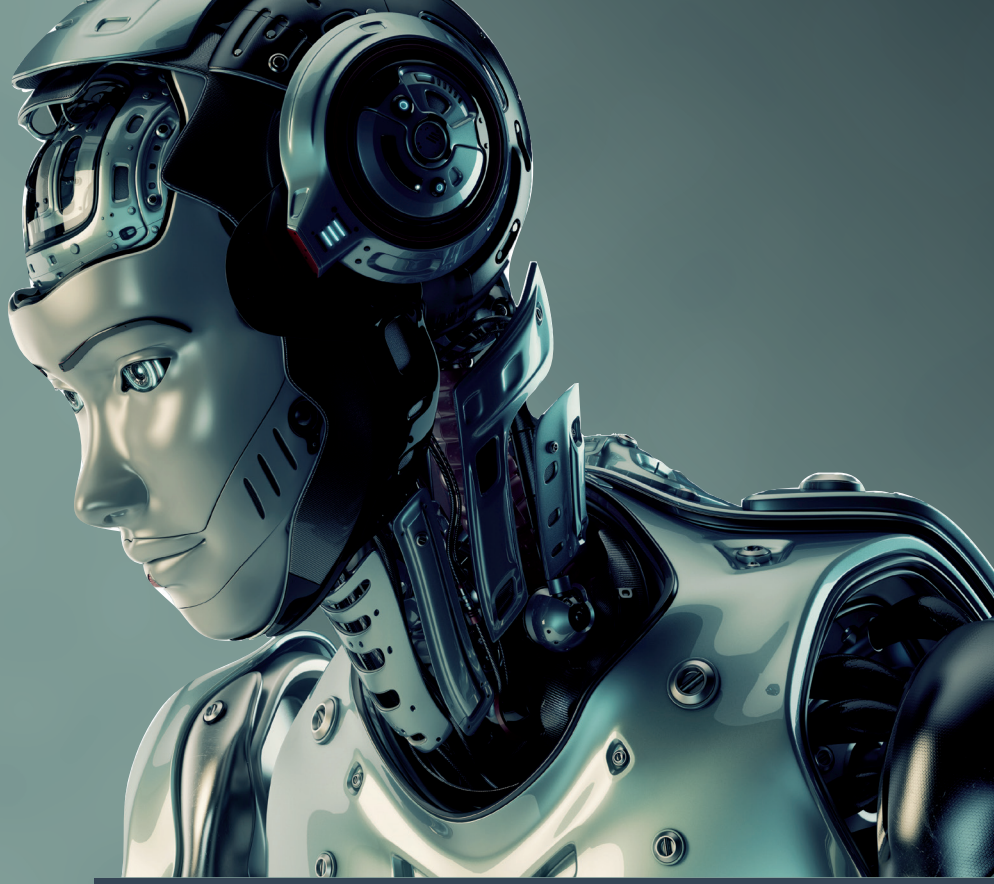
Fazit

In Softwareprojekten gibt es üblicherweise viel zu entscheiden. Ein strukturierter und konstruktiver Umgang mit diesen Entscheidungen kann die Arbeit enorm vereinfachen und Konflikte im Team oder mit externen Stakeholdern verringern oder zumindest versachlichen. Transparenz bei der Wahl eines Entscheidungsmodells und eine offene Kommunikation über Entscheidungsspielräumen verhelfen allen Beteiligten zu einem besseren Miteinander. Und Methoden wie die Vier-Felder-Tafel machen es möglich, dass auch zu einem späteren Zeitpunkt jeder nachvollziehen kann, warum die Welt heute so ist, wie sie ist.

Darüber hinaus sorgen sie so dafür, dass auch abweichende Meinungen gehört und wertgeschätzt werden. Entscheidungen können gemeinsam und bewusst getroffen werden, anstatt „von oben“ verordnet zu werden. So steigt durch einen guten Umgang mit Entscheidungen letztendlich die Motivation aller Beteiligten. ||

Die Autorin

Kathrin Herrmann
(mail@kathrin-herrmann.net)
ist freiberufliche Trainerin und Coach mit dem Schwerpunkt Requirements Engineering für agile Softwareprojekte. Sie verfügt über Zertifizierungen des IREB und der Scrum Alliance. Außerdem ist sie ausgebildeter systemischer Business Coach.



Der AI Trendletter

Jetzt anmelden!

AI Trendletter — fundierte, herstellerunabhängige Fachinformationen rund um Künstliche Intelligenz, Machine Learning, Deep Learning und Robotics.

Wer sich beruflich damit befasst — sei es als erfahrener Software-Entwickler, IT-Manager oder BI-Spezialist — hat es schwer, den Überblick über die schnelllebige Entwicklung zu behalten.

Für diese Fachleute bringen wir den AI Trendletter heraus, der alle zwei Monate über die wichtigsten AI-Facts informiert. Und das herstellerunabhängig, praxisorientiert und für Sie kostenfrei — in Form von tiefschürfenden Fachbeiträgen, fundierten Hintergrund-Interviews, innovativen Praxisbeispielen und prägnanten Überblicksartikeln.

Hier können Sie den AI Trendletter bestellen:

sigs-datacom.de/ai/register/