Blind-Bird Tickets!
Die große IT-Konferenz
08. - 11.12.2025
in Frankfurt am Main
IT-Tage 2025

*Daniel Dubbel* *04. Februar 2025*

# Agile: Hype, Hoffnung oder Hindernis?

Agile Software-Entwicklung: Hype oder Hindernis?
© Adobe: Bojan / Generiert mit KI

**Agilität hat uns einst die große Freiheit versprochen – Flexibilität, Eigenverantwortung und Nähe zu den Kunden. Wer heute aber mit Entwickler:innen spricht, hört oft Ernüchterung: Daily Stand-ups, Retrospektiven und Backlog Refinements werden als zusätzliche Belastung empfunden, als Routine ohne echten Mehrwert. Die Frage liegt auf der Hand: Was bleibt von der Agilität übrig, wenn wir *Scrum*, *SAFe* und *Jira* loslassen? Die provokante Antwort: Vielleicht genau das, was wir brauchen!**

## Agilität: Vom Ideal zur Belastung

Erinnert Ihr Euch noch an die ersten agilen Projekte vor 10 oder 20 oder gar 30 Jahren? Diese aufregende Zeit, als man begeistert neue Wege gegangen ist, statt im ständig gleichen Trott endlose Meetings über Prozesse zu führen und sich in Gantt-Chart-Anpassungen zu verlieren? Nur, was ist aus dieser aufregenden Zeit geworden? Der Traum ist aus. Verfangen in einem Netz aus Regeln, Zertifizierungen und akribisch geplanten Transformationen mit Meetings über Meetings hat es seinen Zauber längst verloren.

Viele Software-Entwickler:innen rigider Anwendungen empfinden agile Methoden als hinderlich für ihre Produktivität und haben den Eindruck, mehr Zeit mit der Pflege agiler Prozesse zu verbringen als mit der eigentlichen Entwicklungsarbeit. Ein Beispiel aus dem Alltag: Ein crossfunktionales Team investiert Stunden in die Perfektionierung von Akzeptanzkriterien für User Stories, nur um später festzustellen, dass die Kundenanforderung falsch verstanden wurde. Der Dialog mit dem Kunden? Fehlanzeige. Warum? Weil der Sprint-Plan wichtiger erscheint als pragmatische Flexibilität. Weil es nicht um die Lösung des Problems, sondern um die Einhaltung der Regeln geht. Dieses Szenario ist symptomatisch für ein System, das Agilität über alle Maßen formalisiert und dabei ihre Essenz verloren hat.

## Warum agile Frameworks oft scheitern

Eine Studie aus dem Jahr 2023 ergab, dass 65 Prozent der Software-Projekte, die agile Methoden anwenden, nicht rechtzeitig, nicht innerhalb des Budgets und nicht mit einem hohen Qualitätsstandard abgeschlossen wurden [1]. Eine andere Untersuchung stellte fest, dass 31 Prozent der befragten Unternehmen ihre agile Transformation für abgeschlossen halten [2]. Abgeschlossen? Ist Agilität nicht ein fortwährender Prozess, der letztendlich nie abgeschlossen sein kann?

Frameworks wie Scrum und SAFe sollen Orientierung bieten und Zusammenarbeit vereinfachen, doch in der Praxis ersticken sie oft an ihrer Starrheit. Entwickler:innen erleben nicht selten, dass die Methoden wichtiger erscheinen als die Ergebnisse.

> “ Ist Agilität nicht ein fortwährender Prozess, der letztendlich nie abgeschlossen sein kann?

Nehmen wir typische Daily Stand-ups. Jeden Morgen um Punkt neun Uhr tauscht sich das Team aus – doch der eigentliche Nutzen bleibt aus. Dabei könnten die meisten Updates asynchron über ein Chat-Tool geteilt werden. Plötzlich wäre wertvolle Zeit für echte technische Diskussionen verfügbar. Warum bleibt man trotzdem dabei? Weil es eben dazugehört.

Oder wie ist es mit *Jira*-Workflows? Das Team verbringt Wochen damit, komplexe Workflows in *Jira* zu erstellen. Ergebnis: Frust über endlose Pflichtfelder und eine erdrückende Prozesstiefe. Der eigentliche Zweck – die Zusammenarbeit zu erleichtern – geht dabei verloren. Menschen hängen oft an bestehenden Prozessen, ganz egal ob sie funktionieren oder nicht. Dieses Phänomen ist bekannt als "Sunk Cost Fallacy" und einer der Gründe, warum Teams trotz offensichtlicher Probleme an Frameworks – auch an agilen – festhalten [3]. Inspect & Adapt wird hier schon aus psychologischer Sicht zu einer Herausforderung.

Betrachtet man in einem größeren Kontext agile Transformationen, sieht es nicht besser aus. Unternehmen investieren oft erhebliche Ressourcen in agile Transformationen und scheuen sich hinterher, diese Investitionen als Fehlschläge anzuerkennen [4]. Gleichzeitig fürchten viele Führungskräfte den Verlust von Kontrolle, wenn Prozesse gelockert werden.

## Vom Agilen Manifest zum Massengeschäft

2001 setzten sich 17 Entwickler zusammen und schrieben das Agile Manifest – ein radikaler

## Autor

**Daniel Dubbel**

Daniel Dubbel ist Experte für Organisationsentwicklung, moderne Führung und agile Transformationen mit über 20 Jahren Berufserfahrung in… >> Weiterlesen

Blind-Bird Tickets!
IT-Tage 2025
08. - 11. Dezember 2025
Kap Europa, Frankfurt am Main

## Newsletter

**Unser Newsletter informiert Sie regelmäßig und kostenlos über Neuigkeiten, Artikel und Veranstaltungen zu aktuellen IT-Themen.**

Newsletter abonnieren

## Nachrichten

**10.01.2025**

**Konferenz für Software-Architektur 2025: Software-Architektur-Tage**

Vom 08. bis 11.12.2025 finden die Software-Architektur-Tage in Frankfurt am Main statt. Das Besondere daran: Du kannst an allen anderen…

>> Weiterlesen

**10.01.2025**

**IT-Tage 365: Online-Konferenzen des Fachmagazins Informatik Aktuell**

Flexible Fortbildung zu Software-Entwicklung, Datenbanken, DevOps, Cloud, IT-Security, Software-Architektur mit einem Ticket! Online…

>> Weiterlesen

**03.01.2025**

**Konferenz für Java-Entwicklung 2025: Java-Tage in Frankfurt am Main**

Eine der wichtigsten Konferenzen für Java-Entwicklung in Deutschland: Mehr als 250 Sessions zu Java, JVM, Spring Boot, Java EE, Datenbanken,…

Gegenentwurf zu damals gängigen Arbeitsweisen: Zusammenarbeit statt Bürokratie, funktionierende Software statt endloser Dokumentation, Anpassungsfähigkeit statt rigider Pläne. Die Idee war simpel und befreiend. Doch was einst ein flexibler Denkansatz war, wurde schnell zum Geschäftsmodell.

Heute gibt es unzählige Zertifikate, Schulungen und Frameworks. Jeder kann sich "Scrum Master" nennen und Firmen zahlen Unsummen für Begleitung, Transformation, Rezertifizierungen oder die Lizenzierung von Tools wie *Jira* und Co. Statt echter Agilität dominieren Methoden, die wie in Stein gemeißelt wirken. Ganze Skalierungs-Frameworks verkaufen sich als die Lösung, während sie häufig nur alte Probleme neu verpacken, Menschen neue Rollen bescheren und dabei Agilität ad absurdum führen.

Agilität wurde kommerzialisiert. Agile wurde ein Trend und mehr und mehr optimiert für Umsatz, nicht für die Nutzung durch Entwickler in ihrer wertschöpfenden Arbeit. Wenn es scheitert – und das passiert oft – wird die Schuld auf die Firmenkultur, die Organisation oder die Mitarbeitenden geschoben. Kaum jemand fragt dabei, ob denn das dogmatische Durchdrücken agiler Methoden und Praktiken, das Beharren, es "by the book" und damit vermeintlich "richtig" zu machen, die falsche Lösung für das Problem war.

Für Entwickler:innen bleibt die Frage: Wie können wir bei all der agilen Bürokratie wieder etwas verändern, um Wert zu schaffen und um ein System zu verlassen, das mehr Schein als Sein ist?

## Software-Entwicklung ohne Agile: Ein radikaler Ansatz

Wir sollten den Begriff Agile hinterfragen und uns auf das Wesentliche konzentrieren. Zusammenarbeit, Kundenfokus und kontinuierliche Verbesserung sind keine exklusiven Bestandteile agiler Frameworks – sie sind Grundprinzipien guter Arbeit.

Ein paar beispielhafte Ansätze:

1. Feature Driven Development ohne Meetings: Ein Team arbeitet in kleinen Einheiten, stimmt sich direkt mit Stakeholdern ab und holt Feedback über kurze Demos ein – ohne formalisierte Meetings. Die Ergebnisse sind oft schneller und näher an den Kundenbedürfnissen.
2. Pragmatische Fehlerkultur: Statt Retrospektiven durchzuführen, analysiert ein Team Fehler direkt, wenn sie auftreten. Die Erkenntnisse werden sofort umgesetzt, ohne sich in Diskussionen über Post-its und Diagramme zu verlieren.
3. Empirie statt Dogmatismus: Ein Team testet gezielt verschiedene Arbeitsweisen in einer kontrollierten Pilotphase und etabliert daraus angepasste Methoden, die im spezifischen Kontext funktionieren.

Die Beispiele zeigen: Die besten Teams kommen vielleicht ganz ohne "Agile" zu echter Agilität.

## Scrum, Kanban oder SAFe: Flexibilität und Pragmatismus statt Dogmatismus

Nicht jedes Team braucht *Scrum*, *Kanban* oder *SAFe*. Oft reicht ein simples Whiteboard oder eine To-Do-Liste. Entscheidend ist, dass die Methode dem Team und seiner Arbeit dient – nicht umgekehrt.

Stellt Euch ein kleines Entwicklungsteam vor, das an einem Prototypen arbeitet und statt *Jira* ein einfaches Whiteboard zur Aufgabenverwaltung nutzt. Meetings finden nicht regelmäßig mit immer gleichem Ablauf statt, sondern nur dann, wenn sie notwendig sind. Das Ergebnis schlägt sich positiv auf die Zeit für die Entwicklung und eine schnellere Lieferung des Prototypen nieder.

Auch große Unternehmen profitieren von solcher Flexibilität. So könnten in hybriden Ansätzen eine Kombination aus Kanban Boards und Feature Driven Development eingeführt werden, um den unterschiedlichen Bedürfnissen von Teams gerecht zu werden.

## Agilität aus der Perspektive von Software-Entwickler:innen

Was bedeutet echte Agilität für Software-Entwickler:innen? Weniger Mikro-Management, mehr Eigenverantwortung. Teams sollten selbst entscheiden, welche Arbeitsweisen für sie funktionieren. Das erfordert Vertrauen und den Mut, Kontrolle abzugeben. Warum nicht ein "No-Agile-Experiment" wagen? Teams legen alle agilen Frameworks beiseite und entwickeln ihre eigene Arbeitsweise mit hohem Fokus auf Wertschöpfung und direktem Kontakt zum Kunden. Nach drei Monaten könnten die Ergebnisse ausgewertet und neue, kontextspezifische Ansätze abgeleitet werden.

Scrum Master können sich vom Prozess-Kontrolleur zum "Ermöglicher" entwickeln. Product Owner profitieren von näherem Kundenkontakt und schnelleren Anpassungsmöglichkeiten und die Kunden sehen eine schnellere und flexiblere Berücksichtigung ihrer Bedürfnisse. Klar ist dabei auch, dass ein solches Vorgehen ein hohes Engagement aller Beteiligten erfordert.

## Von der Theorie in die Praxis: Implementierungsstrategien

Schöne Ideen sind das eine, aber wie setzt man sie um? Hier ein paar Ideen, wie Teams den Übergang zu einer flexibleren Arbeitsweise gestalten können, ohne dabei völlig ins Chaos zu stürzen und ohne sich in Agile-Diskussionen zu verlieren:

1. **Think big, start small:** Nichts spricht gegen ein Pilotprojekt, gegen einen ersten Testballon. Wählt ein überschaubares, aber repräsentatives Vorhaben aus und experimentiert dort mit neuen Ansätzen.
2. **Definiert Eure eigenen Metriken:** Story Points zählen für eine "agile Metrik"? Überlegt besser, was bedeutet Erfolg für Euer Team? Kundenzufriedenheit? Weniger Überstunden? Messt das, was für Euch wichtig und richtig ist und Euch in einer kontinuierlichen Verbesserung nach vorne bringt!

>> Weiterlesen

Informatik Aktuell auf Facebook

IT-Jobs auf Informatik Aktuell

3. **Rituale auf dem Prüfstand:** Hinterfragt jedes Meeting, jedes Ritual. Bringt es echten Mehrwert oder ist es nur Gewohnheit? Seid radikal. Schafft ab, was euch offensichtlich nichts bringt. Und wenn Ihr es doch wieder braucht, dann wird es wiederkommen.
4. **Kommunikation ist alles:** Erklärt Eurem Management, warum Ihr etwas ändern wollt und was das ist. Zeigt Zwischenergebnisse. Nehmt Ängste ernst, aber lasst Euch nicht ausbremsen.
5. **Lernen, lernen, lernen:** Führt ein Logbuch eurer Experimente. Was klappt? Was nicht? Und vor allem: Warum? Diese Erkenntnisse sind Gold wert für Eure nächsten Veränderungen.

Und nicht vergessen, dass dieser Veränderungsprozess flexibel bleiben sollte. Plant nicht alles im Voraus, sondern tastet Euch vor, lernt und passt an. Es ist eine Reise und auf dem Weg gibt es viel zu entdecken.

## One size fits non: Branchenspezifische Betrachtungen

Wer schon mal versucht hat, ein XL-T-Shirt einem Kleinkind anzuziehen, weiß: Eine Größe passt nicht allen. Genauso ist das auch mit Arbeitsweisen in der Software-Entwicklung. Deswegen hier ein Blick auf verschiedene Branchen und Unternehmensgrößen:

**Startups (1-20 Mitarbeiter):** Hier ist Flexibilität Trumpf. Formale Prozesse? Eher hinderlich. Stattdessen geht es um direkte Kommunikation, schnelle Entscheidungen, enge Kundenbindung. Ein Whiteboard und tägliche Stand-ups reichen oft völlig aus.

**Mittelständische Softwarehäuser (20-200 Mitarbeiter):** Die Herausforderung hier ist, Struktur zu schaffen, ohne die Flexibilität zu verlieren. Hybride Ansätze können gut funktionieren. Das können Kanban Boards für den Überblick sein, kombiniert mit Feature Driven Development für die tägliche Arbeit.

**Konzerne (200+ Mitarbeiter):** Hier braucht es mehr Koordination. Die Kunst bleibt, sich vor einem Bürokratie-Monster zu schützen! Angepasste Frameworks wie SAFe können sinnvoll sein für große Vorhaben. Dabei darf der Spielraum für die Teams selbst für eigene Anpassungen nicht verloren gehen. Dezentralisierung und gutes Nahtstellenmanagement sind ein Schlüssel. Denn auch in Großunternehmen kann man in kleinen, relativ unabhängigen Einheiten arbeiten.

**Regulierte Branchen (z. B. Finanzsektor, Gesundheitswesen):** Compliance-Anforderungen machen es nicht leichter, aber unmöglich ist nichts. Ein Trick könnte sein, smart und nicht umfangreich zu dokumentieren. Freiraum kann durch ein hohes Maß an Automatisierung entstehen. Und findet auch hier Wege, um trotz Regularien nah am Kunden zu bleiben.

Was für alle Größen und Branchen gilt: Hört auf Eure Teams. Sie wissen oft ziemlich genau, was die Kunden wollen oder brauchen und was nicht. Und wenn nicht, dann brauchen sie den Raum, genau das herauszufinden, am besten mit den Kunden gemeinsam.

## Jenseits von Agile und Scrum: Neue Trends in der Software-Entwicklung

Während wir uns von starren agilen Frameworks verabschieden, tauchen neue, spannende Ansätze am Horizont auf. Einer davon lautet *Impact Engineering*. Klingt fancy. Was steckt dahinter?

Im Grunde dreht Impact Engineering den Spieß um. Statt sich – wie in Frameworks wie *SAFe* oder *Scrum* üblich – auf Prozesse zu fokussieren, steht der tatsächliche Impact im Mittelpunkt. Teams in der Software-Entwicklung stellen sich die Fragen, welchen messbaren Unterschied sie für ihre Nutzer machen und wie sie diesen Impact maximieren können.

Ein Team könnte beispielsweise kontinuierlich Nutzerdaten analysieren, statt stur an einem definierten Feature Set zu arbeiten. Dabei entdecken sie vielleicht, dass eine kleine Änderung im Checkout-Prozess die Konversionsrate um fünf Prozent steigern könnte. Und direkt nach der Erkenntnis wird unkompliziert priorisiert, umgesetzt und wieder gemessen. Echter Impact statt Feature-Checklisten.

Andere Trends gehen in Richtung "*Minimal Process Engineering*" – der Versuch, mit so wenig Prozess wie möglich auszukommen. Oder "*Adaptive Planning*", bei dem Pläne nicht für Wochen, sondern täglich angepasst werden. Die gemeinsamen Nenner der Ansätze sind Flexibilität, Kundenfokus und die Bereitschaft, ständig zu lernen und sich anzupassen. Ist kein *Scrum*, kein *SAFe*. Und es klingt irgendwie... agil, oder?

## Zahlen, bitte! Daten zur Effektivität agiler und nicht-agiler Methoden

Wir lieben ja alle anekdotische Evidenz, aber manchmal braucht es harte Fakten. Also, was sagen die Zahlen? Eine globale Studie des *Project Management Institute* aus dem Jahr 2024 liefert interessante Einblicke [4]:

- 64 Prozent der Projekte mit flexiblen, teamspezifischen Methoden wurden als erfolgreich bewertet, verglichen mit 56 Prozent bei strikt agilen und 49 Prozent bei traditionellen Wasserfallmethoden.
- Teams, die ihre Arbeitsweise regelmäßig angepasst haben, berichteten von einer um 23 Prozent höheren Mitarbeiterzufriedenheit.
- Überraschenderweise gab es keinen signifikanten Unterschied in der Projektdauer zwischen agilen und nicht-agilen Projekten. Der entscheidende Faktor war die Fähigkeit, schnell auf Veränderungen zu reagieren.

Spannend ist auch, dass Unternehmen, die einen Mix aus verschiedenen Methoden verwenden, in puncto Innovation und Kundenzufriedenheit am besten abschneiden. Was das bedeutet, ist klar. Flexibilität schlägt Dogmatismus. Und es gibt nicht den einen richtigen Weg. Es kommt darauf an, was für euer Team, euer Projekt und eure Kunden am besten funktioniert.

## Agile Software-Entwicklung trifft Regulierung: Ein Balanceakt

Wie gut es geht, ohne “Agile” flexibel zu arbeiten, zeigt ein tieferer Blick in sehr regulierte Branchen, zum Beispiel die Finanzbranche. Hier gibt es strenge Compliance-Regeln, Audits und Dokumentationspflichten. Klingt erstmal bürokratisch und schwergewichtig. Das muss es aber nicht sein. Unternehmen mit Fokus auf Anpassbarkeit und Kundennähe zeigen, wie es auch in einem solchen Umfeld ohne agile Methoden funktionieren kann:

- Sie automatisieren Compliance-Checks und integrieren sie direkt in den Entwicklungsprozess.
- Sie arbeiten mit "Regulatory Sprints", in denen Compliance-Experten direkt im Entwicklungsteam sitzen.
- Sie nutzen "Compliance as Code"–Ansätze, bei denen Regulierungsanforderungen direkt in den Code eingebaut werden.

Auch im Gesundheitswesen, in dem Patientensicherheit oberste Priorität hat, finden sich Wege:

- Kontinuierliche Validierungen ersetzen große, seltene Audits.
- Risikomanagement wird Teil des täglichen Entwicklungsprozesses, nicht eine separate Phase.
- Dokumentation entsteht nebenbei, nicht als zusätzliche Belastung am Ende.

> “ Die Zukunft der Software-Entwicklung liegt in der Fähigkeit, sich kontinuierlich anzupassen, um sich bestmöglich aufzustellen.

Der Punkt ist: Wenn man sich von Dogmatismen löst und nicht das Ziel verfolgt, Agile einzuführen und Frameworks zu etablieren, dann wird der Blick frei für das, was trotzdem gehen kann. Dann kann der Schlüssel darin liegen, Regulierung nicht als Hindernis, sondern als integralen Teil der Entwicklung zu betrachten. Es erfordert Kreativität und manchmal auch Mut, aber es ist möglich und oft sogar effizienter als traditionelle Ansätze, auch als "klassisch agiles" Vorgehen.

## Fazit: Agile Software-Entwicklung – Zeit für eine Neudefinition

Der Agile-Hype liegt längst im Sterben. Die wichtigen Prinzipien wie Zusammenarbeit, Kundenorientierung, kontinuierliche Verbesserung sind zeitlos und ihre Umsetzung muss flexibel bleiben. Teams brauchen den Freiraum, ihre eigenen Lösungen zu finden, und Unternehmen müssen den Mut haben, diesen Prozess zu unterstützen. Die Zukunft der Software-Entwicklung liegt nicht in starren Prozessen, weder in klassischen, noch in klassisch agilen, sondern in der Fähigkeit, sich kontinuierlich anzupassen, um sich für den Zweck von Unternehmen und Team bestmöglich aufzustellen. Der erste Schritt? Den dogmatischen Agile-Hype in den Schrank zu anderen verstaubten Ideen zu packen und neu zu denken mit der Bereitschaft, das Beste aus theoretischen Konzepten und Erfahrungen aus der Praxis zu vereinen.

Die Diskussion um post-agile Arbeitsweisen ist nicht neu, steckt aber vielerorts noch in den Kinderschuhen. Und auch wenn in vielen anderen Bereichen jenseits der Software-Entwicklung gerade erste Schritte in Agilität gegangen werden, für die Software-Entwicklung ist das ein mindestens 25 Jahre alter Hut. So wie damals agile Software-Entwicklung den Stein ins Rollen gebracht hat, ist es jetzt vielleicht an der Zeit, nicht-agile, zweckorientierte und wirklich flexible Software-Entwicklung anzugehen.

### Agile auf den diesjährigen IT-Tagen

Spannende Vorträge und Workshops zum Thema **Agile** erwarten Euch auch auf den **IT-Tagen**, der Jahreskonferenz von Informatik Aktuell. Die IT-Konferenz findet jedes Jahr im Dezember in Frankfurt statt – dieses Jahr vom 08.-11.12.

## Quellen

[1] CIO: Ist agile Softwareentwicklung nutzlos?
[2] BearingPoint: Studie Agilität: Die Mehrheit der Organisationen hat ihre Transformationsziele noch nicht erreicht
[3] The Decision Lab: Sunk Cost Fallacy
[4] Project Management Institute

mitteilen teilen teilen teilen mail

## Autor

**Daniel Dubbel**

Daniel Dubbel ist Experte für Organisationsentwicklung, moderne Führung und agile Transformationen mit über 20 Jahren Berufserfahrung in verschiedenen Branchen und Unternehmensgrößen.

>> Weiterlesen

## Das könnte Sie auch interessieren

**DevOps und Agile: Synergien im IT-Projektmanagement**

**Konflikte lösen: Der gute Konflikt**

**Lean Startup im B2B-Kontext anwenden**

**Aufarbeitung von Mikrotraumata und struktureller Gewalt**

**Flaccid Scrum – woran Agile scheitert und wie Sie es verhindern**

**AI-Office in zehn Schritten: Eine Bauanleitung**

## Kommentare (0)

### Neuen Kommentar schreiben

Name:

E-Mail-Adresse:

Kommentar:

☐ Ich verstehe und akzeptiere die Datenschutzbestimmungen.

Absenden

**Newsletter**

- ► *Aktuelle Artikel*
- ► *Aktuelle Nachrichten*
- ► *Aktuelle Konferenztermine*

Newsletter abonnieren

**Kategorien**

Startseite
Management
Entwicklung
Betrieb

**Service**

Aktuelle Meldungen
Konferenzkalender
IT-Jobs
Sitemap

**Information**

Impressum
Über Uns
Media
Datenschutz
Nutzungsbedingungen
Kontakt

**Folgen Sie uns**

© Alkmene Verlag GmbH 2025