Agil++: Ein systematischer Ansatz für das Tailoring hybrider IT-Projektmethoden

So wenig Methode wie möglich – so viel Methode wie nötig

Thomas Greb[1]

Abstract: Die Verbreitung agiler Ansätze nimmt stetig zu. Aktuelle Daten zeigen jedoch, dass die Zufriedenheit mit rein agilen Ansätzen tendenziell zurückgeht, während gleichzeitig der Nutzungsanteil hybrider Methoden deutlich wächst. Hybride IT-Projektmethoden entstehen in der Praxis meist nach dem „trial-and-error"-Prinzip. Dieser Beitrag stellt einen systematischen Ansatz für das Tailoring hybrider IT-Projektmethoden vor. Ausgangsbasis bilden dabei erprobte agile Ansätze, die situativ um etablierte Projektmanagement- und Software-Engineering-Methoden erweitert werden.

Keywords: Agil, Projektmanagement, Software Engineering, Tailoring, hybride IT-Projektmethode

1 Plan-getrieben, agil, hybrid: „state of the art" 2021

Agile Ansätze sind im Mainstream angekommen, wobei dies stärker für agile Managementansätze wie Scrum oder Kanban, als für agile Software-Engineering-Ansätze wie Extreme Programming (XP) oder Test Driven Development (TDD) gilt [Ko20, S. 51]. Agilität hat sich in den letzten 35 Jahren ausgehend von einem Ansatz zur Produktentwicklung für Autos, Drucker etc. [Th86], über eine Methode zur Weiterentwicklung objektorientierter Softwaresysteme [Sc97, S. 3] hin zu einem allgemeinen Organisationsparadigma entwickelt. Wer will, wer darf heutzutage nicht agil sein?

Aktuelle Daten aus der Praxis belegen jedoch auch, dass die Erfolgsquote rein agiler Ansätze seit einiger Zeit zurückgeht [Ko20, S. 39]. Parallel nimmt die Nutzung hybrider - also gemischt Plan-getriebener und agiler - Methoden deutlich zu: von 27 % in 2012 auf 43 % in 2019 [Ko20, S. 14].

Vieles deutet darauf hin, dass der Einsatz hybrider IT-Projektmethoden kein lästiger Zwischenschritt auf dem Weg zu vollständiger Agilität, sondern für die meisten Situationen ein logischer, wünschenswerter Zielzustand ist. Eine Studie zu Softwareentwicklungsmethoden zeigt jedoch, dass hybride Methoden aktuell zu 80 % evolutionär entstehen [Kl19]. Bisher hat sich weder in der Wissenschaft noch in der Praxis ein systematischer Ansatz zum Tailoring hybrider IT-Projektmethoden etablieren können.

[1] Dr. Thomas Greb, Thomas Greb Consulting, Rischgraben 51, 28832 Achim, th.greb@thomas-greb-consulting.com, https://www.thomas-greb-consulting.com

2 Plan-getrieben, agil, hybrid: Kernideen, Stärken und Schwächen

„Erst denken, dann handeln" ist die Kernidee Plan-getriebener Entwicklungsansätze. Auf Basis einer systematischen Anforderungsanalyse werden ein Lösungsdesign sowie ein Umsetzungsplan entwickelt, um Komplexität zu reduzieren, Abhängigkeiten zu managen, Aktivitäten zu organisieren sowie konstruktiv Fehler zu vermeiden.

Für Plan-getriebene Ansätze gibt es sowohl im Projektmanagement als auch im Software-Engineering umfangreiche und erprobte Methodensets, etwa den Project Management Body of Knowledge® [AG21] oder den Software Engineering Body of Knowledge® [BF14], die teilweise Eingang in nationale und internationale Normen gefunden haben. Charakteristisch für die Projektmanagementstandards ist ihr universeller methodischer Anspruch: von der Softwareentwicklung bis zum Bau der Kindertagesstätte. Dem gegenüber spezialisieren sich Software-Engineering-Methoden stärker auf Themen und Technologien wie z.B. Architekturmanagement (TOGAF® [Th21] etc.) oder die Unterstützung objektorientierter Entwicklungsparadigmen (RUP® [Kr03], UML® [Bo17] etc.). Kritikpunkte an beiden Methodensets sind unter anderem die zu hohe Komplexität, der zu starke Dokumentenfokus sowie die zu geringe Flexibilität und Geschwindigkeit.

„Lernen durch Erfahrung" ist der agile Ansatz. Für priorisierte Anforderungen wird in selbstorganisierenden Teams möglichst schnell „working software" erzeugt. Auf Anforderungsänderungen bzw. Designfehler wird mit „refactoring" reagiert. Im Vergleich zu Plan-getriebenen Ansätzen sind agile Ansätze wesentlich flexibler, eher menschenorientiert und bewusst methodisch inhärent unvollständig. „Lernen durch Erfahrung" ist jedoch in Fällen prohibitiv, in denen der Verlust von Leben (Flugzeugsteuerung, Medizintechnik etc.) wahrscheinlicher wird [BT04].

Eine systematische Analyse der Stärken und Schwächen agiler Ansätze liefert Bertrand Meyer's "Agile! The Good, the Hype and the Ugly" [Me14]. Meyer legt insbesondere dar, warum die weitestgehende Ablehnung systematischer Planungs-, Analyse- und Designaktivitäten durch die agile Community - „no Big Upfront anything" - schädlich für gute Systementwicklung ist [Me14, S. 149]. Ernst Denert als einer der prominentesten Verfechter des Software Engineerings in Deutschland formuliert es gewohnt deutlich und kritisiert an der Agilität die unstrukturierte und wenig ingenieurmäßige Herangehensweise [De21, S. 124]. Im Rahmen der „Agilisierung" der Software- und Systementwicklung wurde viel Überflüssiges über Bord geworfen, aber in der immer noch andauernden „Hype"-Phase leider wohl auch einiges Sinnvolle und Notwendige.

Tabelle 1 fasst die wesentlichen Eigenschaften, Stärken und Schwächen von Plan-getriebenen und agilen Ansätzen im Hinblick auf das Design hybrider Methoden zusammen.

Plan-getrieben **„Erst denken, dann handeln“**	**Agil** **„Lernen durch Erfahrung“**
Systematische Analyse, Planung und korrektes Lösungsdesign sind - falls dies denn möglich ist - effektiver und effizienter als Lernen am Einzelfall	Iterative Vorgehensweise mit kurzem Planungshorizont ist fast die einzige Option bei initial unklaren Zielen und geringer Anforderungsstabilität
„Divide et impera“-Prinzip reduziert Komplexität und ermöglicht die Parallelisierung von Aktivitäten	Priorisierte Lösung von Einzelthemen kann die „time to market“ verkürzen
Viele Dokumente (Spezifikationen, Pläne etc.) mit teils zweifelhaftem Nutzen	„Lernen durch Erfahrung“ ist prohibitiv wenn Leben in Gefahr geraten
Projekt- statt Product-Life-Cycle-Orientierung	(Kurzfristige) Kunden- und Produkt- statt Plan-Orientierung
Stärkere personelle Trennung von Planung, Ausführung und Kontrolle	„Empower people“: Teamgedanke und persönliche Kommunikation
(Zu) umfangreiche Methodensets, aus denen ausgewählt werden muss: Tailoring	Minimalistische Methoden, die nur bei vollständiger Umsetzung wirksam sind

Tab. 1: Plan-getrieben vs. agil: Eigenschaften, Stärken und Schwächen

Hybride Ansätze bieten die Chance, die Stärken eines Ansatzes zu nutzen und gleichzeitig dessen Schwächen durch Elemente des jeweils anderen Ansatzes zu kompensieren. Hybride Methoden können in unterschiedlichster Form konfiguriert werden:

- Parallele Nutzung von Plan-getriebenen und agilen Ansätzen.
- Sequentielle Nutzung der Ansätze, z.B. „Wasser-Scrum-Fall“ [Ti17, S. 241 ff.].
- „Kreuzung“ von plan-getriebenen und/oder agilen Ansätzen, z.B. „ScrumBan“.
- Erweiterung agiler Ansätze bzw. „Agilisierung“ Plan-getriebener Ansätze.

Erfolgreiches Tailoring hybrider IT-Projektmethoden kann nur in Kenntnis der Stärken und Schwächen sowie der (In-)Kompatibilitäten der zu konfigurierenden Methoden-elemente beider Ansätze entstehen. Den unter Zeitdruck und Informationsdefiziten handelnden Praktikern kann dabei ein Tailoring-Framework[2] wie Agil++ helfen, das erfolgreiche Designprinzipien für hybride IT-Projektmethoden umsetzt und gleichzeitig einen situativen, erfahrungsbasierten Tailoring-Lernprozess ermöglicht.

[2] Projektmanagementstandards wie PMBoK® [AG21, S. 131] bieten zunehmend Handlungsempfehlungen für das Tailoring, die jedoch wenig konkret für hybride Projektkonstellationen sowie für IT-Vorhaben sind.

3 Agil++: Systematisches Tailoring hybrider IT-Projektmethoden

Die Kernidee von Agil++ ist es, systematisch und situationsspezifisch eine hybride IT-Projektmethode zu tailoren, indem eine passende agile Methode ausgewählt und um erprobte Methoden des Projektmanagements sowie des Software-Engineerings ergänzt wird. [3] Da die durchgängige, effiziente Nutzung der meisten Methoden nur mit geeigneter Werkzeugunterstützung gelingt, sollte eine solche Unterstützung direkt mit betrachtet werden. Der grundlegende Ablauf des iterativen Tailoring-Prozesses ist in Abbildung 1Abb. 1 dargestellt und wird in seinen idealtypischen Schritten im Folgenden beschrieben.

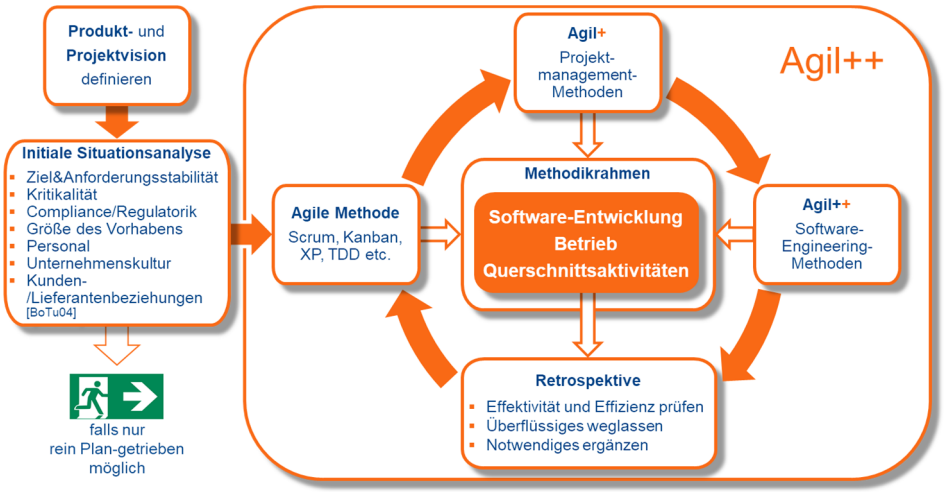

Abb. 1: Iterativer Agil++ Tailoring-Prozess

3.1 Produkt- und Projektvision

Um eine Entscheidungsgrundlage für das Tailoring zu haben, sollte zusammen mit dem Sponsor des Vorhabens sowie dem späteren „Product Owner“ ein gemeinsames Verständnis hinsichtlich der inhaltlichen und geschäftlichen Ziele und Rahmenbedingungen des Vorhabens erarbeitet und auf maximal zwei Seiten in Formen einer Produkt- und Projektvision dokumentiert werden.

[3] Agil++ unterstützt methodisch auch die Nutzung mehrerer, unterschiedlicher agiler Methoden parallel, die dann über die Agil+ bzw. Agil++ Ergänzungen miteinander synchronisiert werden.

3.2 Initiale Situationsanalyse

Auf Basis der Produkt- und Projektvision ist zu prüfen, ob die Voraussetzungen für den Einsatz einer agilen Methode gegeben sind. Die Entscheidungskriterien basieren auf den Überlegungen von Boehm und Turner [BT04, S. 55]:

Kriterium	Bedeutung und Konsequenzen
Ziel- und Anforderungsstabilität	Agile Methoden sind die Lösung, wenn damit instabile Anforderungen bis zum „return-on-investment“ umgesetzt werden können, während Plan-getriebene Ansätze bei stabilen Zielen und Anforderungen effizienter sind.
Kritikalität	Das agile Prinzip „Lernen durch Erfahrung“ ist prohibitiv, wenn Leben, Gesundheit, die Organisation, erhebliche Finanzmittel etc. in Gefahr geraten.
Compliance/ Regulatorik	In vielen Bereichen wie Luftfahrt, Medizin etc. gibt es gesetzliche und regulatorische Anforderungen an Dokumentation und „traceability“, die den Einsatz agiler Methoden einschränken oder verhindern können.
Größe des Vorhabens	Agile Ansätze funktionieren gut bis zur Teamgröße (8-12 Personen), darüber hinaus nur mit zusätzlichen Skalierungsmechanismen.
Personal	Agile Ansätze stellen hohe Anforderungen an die Qualifikation (Cockburn Level 2 und 3 people [BT04, S. 48]), Motivation und Verfügbarkeit von Personal.
Unternehmenskultur	Agiles Arbeiten setzt voraus, Unsicherheit sowie rational begründbare Scope-, Termin- und Budgetänderungen zu akzeptieren. Belohnt die Unternehmenskultur nur „predictability“, stellt dies ein Risiko für den Erfolg agiler Methoden (und die Karriere der Beteiligten) dar.
Kunden- und Lieferantenbeziehungen	Agile Ansätze erfordern direkte, schnelle Kommunikation mit dem Kunden bzw. Auftraggeber sowie flexible Lieferantenbeziehungen.

Tab. 2: Situative Erfolgsfaktoren für agile Methoden

Einige der Kriterien - etwa im Bereich Kritikalität und Regulatorik - haben K.O.-Charakter für agile Methoden, während sich andere Defizite beseitigen oder durch Gegenmaßnahmen kompensieren lassen. Wichtig ist, dass alle am Tailoring Beteiligten, zu einer realistischen Einschätzung und Gesamtbewertung hinsichtlich der Einsetzbarkeit

agiler Methoden kommen[4]. Sind die Voraussetzungen für eine agile Methode nicht gegeben, bleibt nur der „Plan-getriebene Notausgang“.

3.3 Agile Methode

Agile Methoden unterscheiden sich wesentlich hinsichtlich ihres methodischen Umfangs: Scrum hat sich im Laufe der Zeit zu einem universellen Management-Framework ohne Software Engineering-Unterstützung entwickelt[5], während XP [Be04] konkrete methodische Anforderungen (z.B. automatisierte Unit-Tests) im Bereich Softwareentwicklung stellt, dafür aber weniger Unterstützung im Bereich Management bietet.

Daher sollte die agile Methode so gewählt werden, dass sie den Bereich mit dem größten methodischen Handlungsbedarf am besten abdeckt [BT04, Appendix A, S. 165ff.]. Für wichtige und zeitkritische Vorhaben sollte zudem eine Methode gewählt werden, die in der Organisation praktiziert und zumindest einmal erfolgreich pilotiert wurde, da sonst inhaltliche Aufgabenstellung und Einführung der neuen Methode vom Team gleichzeitig bewältigt werden müssen.

Agile Methoden enthalten ein reduziertes, dafür aber wohldurchdachtes und in sich abgestimmtes Methodenset, das mit hoher Disziplin umzusetzen ist. In nahezu allen Fällen wird das Team nicht in der Lage sein, diese agile Methode ohne schädliche Seiteneffekte zu vereinfachen.[6] Daher sollte die gewählte agile Methode in sich strukturell nicht mehr verändert werden. Auf die Nutzung von Ansätzen wie „ScrumBan“ verzichtet Agil++ daher bewusst.

3.4 Agil+: Ergänzende Projektmanagementmethoden

Ergänzende Plan-getriebene Projektmanagementmethoden werden primär benötigt für

- Planungs-, Organisations- und Steuerungsaufgaben, welche die kurzfristige Detail- und Einzelthemensicht agiler Ansätze um eine „high-level“ Gesamtsicht erweitert.
- wichtige Einzelthemen, für die agile Ansätze so gut wie keine Unterstützung bieten.

[4] Vielleicht wird beim Leser an dieser Stelle der Wunsch nach einem Agil++ Kriterien-Template aufkommen. Dem Praktiker wird jedoch einsichtig sein, dass die Dokumentation von Personalqualifikationen und Einschätzung der Unternehmenskultur in der Realität wenig ratsam ist. Daher muss der Autor hier auf die Kreativität des Tailoring-Teams setzen.

[5] In der Erstveröffentlichung 1997 beschreibt Ken Schwaber Scrum noch wie folgt: „SCRUM is a management, enhancement and maintenance methodology for an existing system or prototype. It assumes existing design and code …“ [Sc97, S. 3] Der Scrum Guide 2020 bietet für Softwareentwicklung keine spezifische Unterstützung mehr. Das Wort Software taucht im aktuellen Scrum Guide gar nicht mehr auf [Th20].

[6] Schwaber und Sutherland beschreiben das im Scrum Guide 2020 wie folgt: „The Scrum framework, as outlined herein, is immutable. While implementing only parts of Scrum is possible, the result is not Scrum. Scrum exists only in its entirety and functions well as a container for other techniques, methodologies, and practices.“

Bei der folgenden Betrachtung dieser Methoden wird sich zeigen, dass einige dieser Methoden nicht nur getailored, d.h. ausgewählt und ggf. angepasst, sondern bereits im Rahmen des initialen Tailorings - bis zu einem gewissen Detaillierungsgrad - durchgeführt werden müssen, da das Ergebnis wieder Rückwirkungen auf das Tailoring selbst haben kann.

Gesamtaufwandsschätzung: Agile Methoden betrachten Inhalt und Aufwand in der Regel nur für die nächsten ein bis zwei Iterationen - etwa im Rahmen eines Scrum-„sprint-plannings“ - was u.a. mit der Instabilität zukünftiger Anforderungen begründet wird.

Insbesondere zur Beantwortung folgender Fragen, wird jedoch - zumindest auf „high-level“ Ebene - eine Gesamtaufwandsschätzung benötigt:

- Ist die gewählte agile Methode so einsetzbar, oder werden ggf. mehrere agile Teams und weitere Skalierungsmechanismen benötigt, um in der gewünschten Zeit Ergebnisse zu erzielen?
- Wie ist die Aufgabenverteilung und das Zusammenarbeitsmodell bei mehreren agilen Teams zu organisieren?
- Wieviel Personal mit welchem Qualifikationsprofil wird für wie lange benötigt?

Als Schätzmethoden kommen sowohl Plan-getriebene Verfahren wie 3-Punkt-Expertenschätzungen, die auf initialen Arbeitsstrukturbäumen basieren können, als auch agile Schätzmethoden wie „planning poker“, die ein initiales high-level „product backlog“ nutzen, in Betracht. Der Vorteil klassischer Schätzverfahren ist der breite Methodenfundus, der Vorteil des agilen Schätzansatzes ist es, dass das Ergebnis methodisch nahtlos für die weitere agile Detailplanung verwendet werden kann. Zur Beantwortung der oben genannten grundsätzlichen Fragen reicht zum Zeitpunkt des initialen Tailorings meist eine Schätzgenauigkeit von plus minus 75 % aus. Ein wesentlich höherer Genauigkeitsanspruch wäre in einem agilen Umfeld mit instabilen Anforderungen unrealistisch und bei unbekanntem Projektteam ohnehin zum Scheitern verurteilt.

Termin- und Abhängigkeitsmanagement: Agile Methoden bieten keine ausreichende methodische Unterstützung für das Management von Terminen und Abhängigkeiten, sofern diese zeitlich außerhalb einer Iteration bzw. organisatorisch außerhalb eines agilen Teams liegen [Me14, S. 150]. Der Königsweg, Abhängigkeiten vorab zu beseitigen, ist aus fachlichen, technischen und organisatorischen Gründen oft nicht möglich. Inhaltlich nicht auflösbare Abhängigkeiten, nicht berücksichtigte Vorgängerbeziehungen und ignorierte Vorlaufzeiten tauchen dann urplötzlich als „Impediments“ [7] auf, machen agile Teams beliebig unproduktiv und gefährden Sprintziele. Mechanismen wie „Scrum of Scrums“ bieten nur eine Teillösung, da dezentrale Planungsmechanismen weder

[7] Durch Ignorieren der zugrundeliegenden Ursachen dieser „Impediments“ wird die Schuld dann oft in unzureichender Arbeit des Scrum-Masters, fehlendem „agilem Mindset“ des „Zulieferers“ oder mangelnder Agilität der Gesamtorganisation gesucht.

Vorlaufzeiten noch Überlastungseffekte durch mehrere hochpriorisierte Anfragen, die kurzfristig auf ein agiles Team treffen, wirksam managen können.

In der Plan-getriebenen „Welt“ sind bereits seit längerer Zeit wirksame Methoden zum Zeit- und Abhängigkeitsmanagement etabliert (Netzplantechnik, kritische-Pfad-Analyse, Gantt-Charts etc.), deren Nutzung durch leistungsfähige Projektmanagementsoftware unterstützt wird.

Im Rahmen des Tailorings ist daher das Ausmaß von Abhängigkeiten zu analysieren und je nach Ergebnis sind eine oder mehrere Plan-getriebene Methoden für das „Scheduling“ auszuwählen. Diese Methoden sollten dann so konfiguriert werden, dass in die autonome, dezentrale Arbeitsweise agiler Teams so wenig wie möglich eingegriffen wird, während gleichzeitig ein Terminmanagement auf Ebene des Gesamtvorhabens ermöglicht wird.

Business Case: Auch agile Vorhaben müssen sich rechnen. Selbst wenn der „product pwner“ zu jedem Zeitpunkt die „user story“ mit dem höchstem Business Value auswählt, kann es sein, dass das Gesamtvorhaben, z.B. unter Berücksichtigung der Betriebskosten, nicht rentabel ist oder Budgetvorgaben nicht einhält. Daher macht es Sinn, eine „Business Case“-Methodik“ zu tailoren, diesen Business Case auf Basis der initialen Gesamtschätzung initial zu rechnen und in der Folge mittels der zunehmend detaillierten Informationen aktuell zu halten.

Risikomanagement: Risiken - also der potenzielle Eintritt eines negativen Ereignisses - können den Erfolg eines Vorhabens verhindern, egal ob es Plan-getrieben, agil oder hybrid organisiert ist. Risikoquellen können fachlicher, technischer, interner, externer Natur etc. sein. Risiken können auch aus der Nutzung agiler Methoden selbst entstehen, etwa das Risiko der nicht durchgängigen Verfügbarkeit des agilen Entwicklungsteams oder das Risiko, dass der „product owner“ das Unternehmen verlässt. Umgekehrt bieten agile Methoden gute Möglichkeiten, fachliche und technische Risiken frühzeitig und flexibel zu attackieren, wenn sie denn identifiziert werden.

Leider beinhalten agile Ansätze keine Risikomanagementmethodik, die über informelle Mechanismen hinausgeht. Daher macht es so gut wie immer Sinn, Risikomanagement als Methode auf „ja“ zu tailoren und direkt eine initiale Risikoanalyse durchzuführen, deren Ergebnis wertvollen Input für das weitere Tailoring geben kann. Hinsichtlich des Risikomanagementansatzes kann aus dem reichen Fundus der Projektmanagementstandards ausgewählt werden.

Stakeholdermanagement: Stakeholder sind Personen oder Organisationseinheiten, die Einfluss auf den Erfolg oder Misserfolg des Projekts oder Produkts haben. Die methodische Situation im Stakeholdermanagement ist ganz ähnlich wie im Bereich Risikomanagement: Agile Methoden bieten hier keine systematische Unterstützung, Plan-getriebene Ansätze schon. Eine passende Stakeholdermanagement-Methode zu tailoren, ist daher ratsam. Eine Stakeholderanalyse im Rahmen des initialen Tailorings kann z.B. wertvolle Hinweise geben, welche Personen in agile Teams einzubinden sind. Abhängig

vom Ergebnis wird dann über die weitere Vorgehensweise entschieden. Ist die Stakeholdersituation statisch und kundenfokussiert, kann das Stakeholdermanagement vom „product owner“ übernommen werden, gibt es Stakeholder in vielen Bereichen so ist das ganze Team gefragt.

Personal- und Skillmanagement: Der Erfolg agiler Methoden hängt noch wesentlich mehr als bei plan- und prozessbasierten Ansätzen von der fachlichen und persönlichen Qualifikation der Teammitglieder sowie deren möglichst durchgängiger Verfügbarkeit ab. Umso verwunderlicher ist es, dass agile Ansätze für die Themenfelder Personalauswahl, Weiterqualifikation und Optimierung der Teamzusammensetzung keine Unterstützung bieten. Die methodische Unterstützung sowie die persönliche Verantwortung für diese Themen sind im Rahmen des initialen Tailorings zu klären und dann fortwährend zu optimieren. Die Vorstellung, dass die mangelnde Qualifikation eines Softwareentwicklers durch sein „agiles Mindset“ ersetzt werden kann bzw. dass nicht qualifizierte Teammitglieder sich selbstständig aus dem Team entfernen, scheint dem Autor zu naiv zu sein.

Vertrags- und Lieferantenmanagement: Sind wesentliche Komponenten oder Leistungen extern zu beschaffen, so sind - falls „agilisierte“ Zusammenarbeitsmodelle auf „time-and-material“-Basis nicht in Frage kommen - hierfür etablierte Methoden des Vertrags- und Lieferantenmanagements zu nutzen, die es in agilen Ansätzen nicht gibt.

Weitere Projektmanagementmethoden: Über die diskutierten Projektmanagement-Methoden hinaus sollte die Notwendigkeit weiterer Methoden katalogartig - z.B. auf Basis eines Projektmanagementstandards wie PMBoK® [21] oder eines organisationsspezifischen Methodenfundus´ - überprüft, aber nur in sehr gut begründeten Fällen bejaht werden, da man ansonsten irgendwann zu viel Agilität verliert und bei einem methodisch zu schwergewichtigen Ansatz endet.

3.5 Agil++: Ergänzende Software Engineering- und IT-Operations-Methoden

Während Projektmanagementmethoden technologieneutral sind, unterscheiden sich die Software Engineering- und IT-Operations-Methoden je nach technischer Plattform (Mainframe, Client-Server etc.), Entwicklungsparadigma (strukturiert, objektorientiert etc.) und Betriebsmodell (on-the-premise, Software-as-a-Service etc.) deutlich. Insbesondere bei größeren Vorhaben sind diverse Technologie-Stacks gleichzeitig zu unterstützen.

Das Tailoring der Software-Engineering- und IT-Operations-Methoden ergänzt die ausgewählte agile Methode hinsichtlich verschiedener Aspekte:

- Stärkung des Software-Engineering-Prinzips in Form übergreifender Analyse- und Designaktivitäten, um Komplexität zu reduzieren sowie Lernen durch negative Erfahrungen - euphemistisch „refactoring“ genannt - soweit möglich zu vermeiden.
- Stärkung und Beschleunigung der agilen Lernschleife durch Automatisierung sowie durch Feedback- und Qualitätssicherungs-Mechanismen.

- Berücksichtigung von IT-Operations-Aspekten bereits im Entwicklungsprozess und Integration von Betriebserfahrungen in die agile Lernschleife.

Anforderungsanalyse und fachliches Systemdesign: Agile Methoden haben meist das Ziel, ein „minimal viable product“ (MVP) in kurzer Zeit zu liefern und durch Nutzer-Feedback zu verbessern. Insbesondere bei größeren Vorhaben gibt es gute Gründe, Methoden zur systematischen Anforderungsanalyse und für ein fachliches Systemdesign zu tailoren und bereits beim initialen Tailoring zumindest auf „high-level“-Ebene auszuführen:

- Wichtige nichtfunktionale Anforderungen und technische Risiken wie Performance, Skalierbarkeit, IT-Sicherheit und Verfügbarkeit können frühzeitig identifiziert und bei der Entwicklung berücksichtigt werden. Ein vollständiges „refactoring“ des MVP kann damit oft vermieden und somit letztendlich Zeit und Budget gespart werden.
- Bei Großvorhaben hilft ein gutes fachliches Systemdesign mit definierten Komponenten, Komplexität zu reduzieren und Aufgaben so auf mehrere „product owner“ und agile Teams zu verteilen, dass der Kommunikationsbedarf minimiert wird.
- Bei kritischen Anforderungen, kann es notwendig sein, die Anforderungsspezifikation nicht mittels „user stories“ am Einzelfall, sondern systematisch und vollständig durchzuführen. Oder würden Sie mit einem Auto fahren wollen, dessen Bremsensteuerungssoftware nur wesentliche, aber nicht alle Betriebssituationen berücksichtigt?

Um die Flexibilität agiler Methoden zu erhalten, sollten systematische Anforderungsanalyse und fachliches Systemdesign - abgesehen von der beschriebenen Ausnahme - nur auf „high-level“-Ebene betrieben, dafür aber über den kompletten Produktlebenszyklus hinweg genutzt werden. Welche Methoden im Detail getailored werden, hängt vom konkreten Technologie-Stack ab.

Technische Systemarchitektur: Geeignete Methoden zur Entwicklung einer übergreifenden technischen Systemarchitektur zu tailoren und diese Methoden im initialen Tailoring zumindest auf „high-level“-Ebene auch anzuwenden, unterstützt folgende Ziele und Aktivitäten:

- Frühzeitige Identifikation und Bearbeitung kritischer technischer Risiken.
- Gezielte Aufgabenverteilung technischer Themen auf agile Teams.
- Schnittstellenmanagement zur Integration des neuen Systems in die IT-Landschaft.
- Monitoring und Beseitigung von technischen Schulden im Entwicklungsprozess. [8]

[8] In einer Fallstudie eines DevOps-Handbuchs [KI16, S. 71] wird beschrieben, dass beim Karrierenetzwerk „linked-in“ direkt nach dem Börsengang für 2 Monate jegliche fachliche Weiterentwicklung eingestellt werden musste, weil technische Schulden und Skalierungsanforderungen im agilen Entwicklungsprozess nicht hinreichend berücksichtigt wurden und erst beseitigt werden mussten.

- Förderung der Entwicklung von wiederverwendbaren Komponenten.
- Bearbeitung von Themen, bei denen „Lernen durch Erfahrung“ prohibitiv ist: IT-Security etc.

Continuous Integration und automatisierte Regressionstests: „Lernen durch Erfahrung“ ist das agile Prinzip. Nach der Entwicklung neuer Funktionalität bzw. nach dem „refactoring“ existierenden Codes, stellt sich die Frage, welche Funktionalität die Software tatsächlich enthält. Automatisierte Regressionstests mit stetig wachsender Testabdeckung – funktional wie nicht nichtfunktional - sind hier fast die einzige Möglichkeit, schnell, beliebig wiederholbar und wirtschaftlich vertretbar eine belastbare Qualitätsaussage zu erhalten. Nach einigen „sprints“ mit vielen zwischen „product owner“ und Team - ggf. nur mündlich - kommunizierten Detailanforderungen werden diese automatisierten Testfälle auch gleichzeitig zur möglicherweise einzigen belastbaren Spezifikation, der tatsächlich in der Software enthaltenen Funktionalität.

Während einige agile Ansätze, wie z.B. Test-Driven-Development und XP, auf diesen kontinuierlichen Integrations- und Test-Mechanismen basieren, so bieten andere weit verbreitete agile Ansätze wie Scrum oder Kanban hierzu keine methodischen Vorgaben. „Agile Lernschleifen“ ohne reproduzierbare, automatisierte Systemintegrationsmechanismen und automatisierte Regressionstests sind aus Qualitätssicht kaum zu verantworten, in der Praxis jedoch leider öfters anzutreffen. Daher sind entsprechende Methoden beim Tailoring unbedingt zu berücksichtigen.

DevOps-Methoden: Der Vorteil agiler Entwicklungsansätze ist die schnelle „time-to-market“. Kurz nach dem Start der agilen Entwicklung ist dann bereits die erste Betriebseinführung vorzubereiten, und das Zusammenspiel von Betrieb und gleichzeitiger agiler Weiterentwicklung ist zu organisieren. Produktionsfehler sind zu fixen. Inhalte von Betriebslogs und Betriebstelemetriedaten sind in die agile Lernschleife zu integrieren. In der Welt von Continuous Integration und Delivery wird man auch feststellen, dass vermeintlich agile Methoden wie Scrum mit 2-Wochen-Sprintzielen im Hinblick auf die Beseitigung kritischer Produktionsfehler unerträglich langsam erscheinen.

Im Rahmen des Tailorings sollte daher die Integration diverser DevOps-Mechanismen (Continuous Integration and Delivery, Canary Releasing, Feature Toggles, Infrastructure as Code etc. [Ki16]) in das hybride IT-Methodenset systematisch evaluiert werden. Hierzu ist es sinnvoll und notwendig, Betriebspersonal frühzeitig mit in das Tailoring und in die agilen Teams einzubinden.

Weitere Software Engineering- und IT-Operations-Methoden: Darüber hinaus gibt es im Bereich Software Engineering und IT-Operations diverse technologie- und situationsabhängige „good practices“ etwa Code Reviews für kritische, schlecht testbare Komponenten. Analog zu den Projektmanagement-Methoden sollte ihr Einsatz im Rahmen des Tailorings katalogartig überprüft werden.

3.6 Retrospektiven: Tailoring als Teil des kontinuierlichen Lernprozesses

Tailoring ist nur erfolgreich, wenn es Bestandteil eines gemeinsam getragenen kontinuierlichen Verbesserungsprozesses wird:

- Die idealtypisch sequentiell dargestellten Schritte sind auch im initialen Tailoring-Prozess bei Bedarf mehrfach zu durchlaufen, bis die Qualität des Methodensets so gut ist, dass man mit der inhaltlichen Arbeit starten kann.
- Nach einer „iteration" bzw. nach jedem „sprint" wird im Rahmen der Retrospektiven überprüft: Was hat methodisch gut funktioniert und was ggf. warum nicht? Bei der Bewertung des Tailorings sind die tatsächlich gelebten und nicht die dokumentierten Methoden entscheidend. Notwendiges ist zu ergänzen. Überflüssiges, das sich in der Praxis nicht bewährt hat, ist zu entfernen.
- Methoden haben die Eigenschaft, dass sie sich in Anzahl und Umfang ständig vermehren, da auf auftretende Probleme oft mit einer methodischen Gegenmaßnahme reagiert wird. Von Zeit zu Zeit ist es daher sinnvoll, jede eingesetzte Methode im Hinblick auf ihren Ergebnisbeitrag kritisch zu hinterfragen und ggf. zu eliminieren.
- Methoden finden nur dann Akzeptanz, wenn ihre Nutzung effizient durch Werkzeuge (Projektmanagement-Tools, Testautomatisierungslösungen, Build-Tools etc.) unterstützt wird. Die Optimierung der Werkzeugunterstützung sollte daher ebenfalls integraler Bestandteil jeder Retrospektive sein.

3.7 Agil++ Designentscheidungen: Warum so und nicht anders?

Agil++ trifft einige grundsätzliche Designentscheidungen hinsichtlich des Tailoring-Vorgehens, deren Hintergründe nun erläutert werden sollen:

- „Bottom-up" statt „top-down"-Tailoring [BT04, S. 152]: Es wird mit einem kleinen Methodenset gestartet, das um das Notwendige ergänzt wird, statt aus einem vollständigen Methodenset das Unpassende zu streichen. Letzteres mag methodisch und wissenschaftlich systematischer sein, ist aber zeitaufwändiger und löst bei methodenunerfahrenen Praktikern den Reflex aus, im Zweifel lieber eine Methode zu viel zu tailoren. Zudem korreliert die „bottom-up"-Strategie besser mit dem Wachstum von „Startups" und passt zum stufenweisen „Hochfahren" von Großprojekten.
- Das Tailoring von Projektmanagement- sowie Software-Engineering- und IT-Operations-Methoden wird gleichermaßen betrachtet, da die Planungs- und Steuerungsmethoden zu den Entwicklungs- und Betriebsprozessen passen müssen.
- Agil++ wählt als Detaillierungslevel für das Tailoring die Methoden- und nicht die Prozessebene[9]. Agile Prinzipien priorisieren bewusst Menschen über Prozesse, daher

[9] Eine Prozessbeschreibung ist detaillierter und besteht aus Input, Output, Verarbeitungsschritten, Rollen etc.

wird in Agil++ den für die Methode Verantwortlichen die Art der Umsetzung überlassen. Wenn eine Organisation aus guten inhaltlichen Gründen (Regulatorik etc.) detaillierte Prozesse nutzt, so können diese selbstverständlich verwendet werden.

- Im Gegensatz zu proprietären agilen Skalierungsframeworks, wie z.B. SAFe® [Le16], verfolgt Agil++ den Ansatz, erprobte agile und plangetriebene Ansätze und Standards situationsspezifisch zu kombinieren, um methodische Defizite „minimalinvasiv" zu beheben, statt noch eine Methodik zu erfinden. Ein SAFe® 4.0 Reference Guide [Le16], der auf 508 Seiten zusammenfasst, wie man „agil" und „lean" arbeitet, und doch wieder nicht alle methodisch notwendigen Ergänzungen beinhaltet, macht die Fragwürdigkeit dieser Skalierungsansätze transparent.

4 Umsetzungsschritte und Erfolgsfaktoren

4.1 Bisherige Erfahrungen mit dem Ansatz

Agil++ basiert auf Erkenntnissen, die der Autor als Projektleiter von großen hybriden Releasemanagement-, Systemintegrations- und Digitalisierungsprojekten gewonnen hat, und bei denen diverse agile und Plan-getriebene Methodenelemente (Scrum, Kanban, DevOps, Risikomanagement, Termin- und Abhängigkeitsmanagement, Architekturmanagement etc.) genutzt wurden. Die Organisationen, in denen die Projekte durchgeführt wurden, hatten teilweise CMMI®-Level 3 basierte „Top-Down"-Tailoring Prozesse[10] bzw. etablierte Projektmanagementstandards und ein Projektmanagement-Office sowie jeweils parallel dazu agile Methoden.

Agil++ adressiert die in den Projekten empfundenen Reibungspunkte, z.B. ein zu komplexes Top-Down-Tailoring, und systematisiert gleichzeitig das Tailoring von Methodenelementen, die sich systematisch in der Praxis als erfolgreich erwiesen haben.

4.2 Organisatorische Voraussetzungen schaffen und Umsetzung sicherstellen

Um bei neuen Vorhaben schnell mit Agil++ produktiv zu werden, ist es sinnvoll, in der Organisation einen Tailoring-Prozess zu etablieren, d.h. zu dokumentieren, mit Templates und Werkzeugen zu unterstützen und zu schulen.

Die konkrete Durchführung des Tailorings sollte primär durch die Personen erfolgen, welche die Methoden als Scrum-Master, Entwicklerin etc. später nutzen werden. Im

[10] Bei CMMI® (Capability Maturity Model Integration, hier Dev. v1.3) werden projektspezifische Prozesse Kriterien-basiert aus der Menge der organisationsspezifischen Standardprozesse getailored. Obwohl dieses Tailoring-Vorgehen Vorteile hat, so hat es doch auch gravierende Einschränkungen: Es setzt voraus, dass die Organisation nach einem stabilen Prozessmodell arbeitet und zusätzlich auch noch für die gegebenenfalls fachlich und technisch neuartige Projektsituation bereits passende Standardprozesse entwickelt hat. In geschäftlich dynamischen Umfeldern mit Wachstumspotenzial wird man oft Beides nicht vorfinden.

Hinblick auf eine möglichst objektive und gute Methodenauswahl sollten diese Personen Kenntnisse in agilen und Plan-getriebenen Methoden haben. Das Tailoring bei Know-how-Defiziten durch einen Methodenspezialisten zu unterstützen, ist sinnvoll, solange die Ergebnisverantwortung für das Tailoring bei den inhaltlich Umsetzungsverantwortlichen verbleibt. Typischerweise wird man das Tailoring in Workshops jeweils für priorisierte Themenschwerpunkte, z.B. initiale Situationsanalyse, durchführen. Das erarbeitete Tailoring ist geeignet zu dokumentieren. Abbildung 2 zeigt beispielhaft ein Agil++ Tailoring-Template für ein größeres hybrides IT-Projekt mit mehreren agilen Teams, das die Erweiterung einer bestehenden Software aufgrund einer gesetzlichen Anforderung zum Inhalt hat.

Agil++ Tailoring-Template							
	Nutzung	**Durchführung im Intial-Tailoring**	**Geltungsbereich**	**Anmerkungen**	**Methodische Konkretisierung**	**Tool**	**Verantwortlich**
Agile Methoden							
SCRUM	ja	nein	Back-End Team		SCRUM-Guide 2020, ohne Anpassungen	JIRA	SCRUM-Master Back-End-Team
SCRUM	ja	nein	Front-End Team		SCRUM-Guide 2020, ohne Anpassungen	JIRA	SCRUM-Master Front-End-Team
Kanban	ja	nein	Operations Team			Kanban-Board	Operations Team Lead
Projektmanagement-Methoden							
Aufwandsschätzung	ja	ja	Gesamtprojekt	ggf. Bildung zus. Front-End-Team erforderlich	Top-Down-Analogie-Schätzung		PL zusammen mit Team
Termin- und Abhängigkeitsmanagement	ja	ja	Gesamtprojekt	kritische Pfad-Analyse erforderlich	Gantt-Chart	MS-Project	PL
Risikomanagement	ja	ja	Gesamtprojekt		Risikomanagement-prozess XY-AG	Risk-Tool XY-AG	PL zusammen mit Team
Stakeholder-Analyse	nein	nein	Gesamtprojekt	bestehendes System wird funktional weiterentwickelt			
Business Case	nein	nein	Gesamtprojekt	gesetzliches Muss-Projekt			
Software Engineering-Methoden							
Systemarchitektur	ja	ja	Gesamtprojekt	technische Risikobetrachtung erforderlich			Architektin
Unit-Tests	ja	nein	Gesamtprojekt			JUnit	tbd.
Last- und Performancetests	ja	nein	Gesamtprojekt	Erweiterung ggf. performancekritisch		Silk-Performer	tbd.
Coding-Standards	ja	nein	Back- und Front-End-Team				tbd.

Abb. 2: Agil++ Tailoring-Template

Aufgrund der Bedeutung des initialen Tailorings für die weitere Projektarbeit macht es zudem Sinn, das Ergebnis durch qualifizierte Peer Reviews in den jeweiligen Bereichen qualitativ abzusichern. Aufgrund der in den Retrospektiven gewonnenen Erkenntnisse wird das Tailoring dann kontinuierlich aktualisiert.

4.3 Menschliche und methodische Erfolgsfaktoren für das Tailoring

Die wesentlichen Erfolgsfaktoren für das Tailoring hybrider IT-Projektmethoden sind **Objektivität**, **Minimalismus** und **Balance**.

Objektivität: Dies bedeutet, sich beim Tailoring die Vor- und Nachteile agiler bzw. Plan-getriebener Methodenbausteine vorurteilsfrei in der konkreten Situation bewusst zu machen. Daher erfordert das Tailoring hybrider Methoden Wissen und Erfahrungen in agilen und klassischen Methodensets. Am Tailoring Beteiligte sollten sich zudem vorab ihrer persönlichen Präferenzen und deren möglichen Auswirkungen auf ihr Tailoring-Verhalten bewusst werden. Dass inzwischen Schulungen für „agile evangelists" angeboten werden zeigt, dass es hier möglicherweise Erkenntnisbedarf gibt.

Minimalismus: Der Nutzen jeder zusätzlichen Methode ist im Verhältnis zu ihrem Aufwand kritisch zu hinterfragen. Solange keine lebens- oder unternehmensbedrohenden Konsequenzen zu erwarten sind - und nur dann - kann es für die Akzeptanz des Tailorings besser sein, anfangs lieber eine Methode zu wenig zu tailoren und erst bei dadurch auftretenden Problemen methodisch nachzusteuern, als umgekehrt.

Balance: Im Laufe der Zeit sind immer detailliertere Projektmanagement- und Software Engineering-Methoden entstanden, deren ausgiebige Nutzung von Experten dieser Methoden natürlich dringendst empfohlen wird. Eine Methodenkette ist jedoch nur so stark wie ihr schwächstes Glied: Eine weitgehende Testautomatisierung hilft letztlich nur wenig, wenn der Deploymentprozess nicht automatisiert ist und unbeabsichtigt eine andere als die getestete Version ausgerollt wird. Daher ist beim Tailoring über alle Methoden hinweg hinsichtlich Detaillierung und Umfang eine Balance zu finden, die das Verhältnis von Aufwand und Nutzen optimiert und ein im Gesamten weiterhin agiles Vorgehen beibehält.

4.4 Zusammenfassung und Vorteile von Agil++

- Agil++ basiert auf erprobten agilen Methoden sowie ergänzenden Plan-getriebenen Projektmanagement- und Software Engineering-Standards, die auch unterschiedliche „Technologie-Stacks" unterstützen.
- Das Tailoring wird in agilen Lernschleifen regelmäßig und situationsspezifisch in Retrospektiven optimiert und an die konkreten Bedürfnisse angepasst.
- Der Formalisierungsgrad des Tailorings ist situationsspezifisch wählbar. Minimal erforderlich ist nur das Tailoring auf Methoden- und nicht auf Prozessebene.
- Agil++ verfolgt einen minimalistischen Tailoring-Ansatz. Außer einem überschaubaren Tailoring-Prozess muss nichts neu entwickelt, eingeführt und geschult werden.

Dabei gilt: So wenig Methode wie möglich – so viel Methode wie nötig.

Literaturverzeichnis

[Be04] Beck, K.: Extreme Programming – Das Manifest, Addison-Wesley, Boston, 2004.

[Bo17] Booch, G.: The Unified Modeling Language User Guide, Addison-Wesley Object Technology, Boston, 2017.

[BF14] Bourque, P.; Fairley, R.E.: Guide to the Software Engineering Body of Knowledge, 3. Auflage, IEEE Computer Society, 2014.

[BT04] Boehm, B.; Turner, R.: Balancing Agility and Discipline, Addison-Wesley, Boston, 2004.

[De21] Denert, E.: Software Engineering. Informatik Spektrum 44/21, S. 122–125, 2021.

[Ki16] Kim, G. et.al.: DevOPS Handbook - How to Create World-Class Agility, Reliability and Security in Technology Organizations, IT Revolution Press, Portland, 2016.

[Ko20] Komus, A. et.al.: Status Quo (Scaled) Agile 2019/2020, Hochschule Koblenz, 2020.

[Kl19] Klünder, J. et.al.: Catching up with method and process practice - An industry informed baseline for researchers. In (IEEE, Hrsg.): Proceedings of the International Conference on Software Engineering - Software Engineering in Practice (ICSE-SEIP2019). Montréal, S. 255-264, 2019.

[Kr03] Kruchten, P.: The Rational Unified Process - An Introduction, 3. Auflage, Addison-Wesley, Boston, 2003.

[Le16] Leffingswell, D.: SAFe® 4.0 Reference Guide - Scaled Agile Framework® for Lean Software and Systems Engineering, Addison-Wesley-Professional, Boston, 2016.

[Me14] Meyer, B.: Agile! The Good, the Hype and the Ugly, Springer-Verlag, Berlin, 2014.

[Th21] The Open Group: The TOGAF® Standard - Version 9.2, Van Haren Publishing, 's-Hertogenbosch, 2021.

[AG21] A Guide to the Project Management Body of Knowledge (PMBOK® Guide) - Seventh Edition and The Standard for Project Management, Project Management Institute, 2021.

[Sc97] Schwaber K.: SCRUM Development Process. In: (Sutherland J. et.al. Hrsg.): Business Object Design and Implementation. London, S. 117- 134, 1997.

[Th20] The Scrum Guide, https://Scrumguides.org, Stand: 13.08.21.

[Th86] The New New Product Development Game, https://hbr.org/1986/01/the-new-new-product-development-game, Stand: 12.08.21

[Ti17] Timminger, H.: Modernes Projektmanagement - Mit traditionellem, agilem und hybridem Vorgehen zum Erfolg, Wiley-Verlag, Hoboken, 2017.