1 Die Architektur von APM

Agile Vorgehensweisen sind im grundsätzlichen Ansatz einfach zu verstehen, jedoch wird die innere Komplexität ihrer Umsetzung sofort deutlich, wenn wir versuchen, detailliert zu erklären, wie ein agil durchgeführtes Projekt konkret abläuft. Bevor wir in die Tiefen von *APM* eintauchen, finden Sie hier einen kurzen Überblick über die zentralen Elemente von *APM*. Danach klären wir Begriffe aus dem agilen Projektmanagement, die wir später noch genauer definieren werden, aber bereits für die ersten Darstellungen der Zusammenhänge in ihrer grundlegenden Bedeutung benötigen.

1.1 Was ist APM?

APM steht für **A**giles **P**rojekt**m**anagement und beschreibt ein Framework, um Softwareprojekte mit der notwendigen Flexibilität umzusetzen und der hohen Dynamik des Projektumfelds angemessen begegnen zu können. Es fußt auf fünf Säulen und nutzt eine Vielzahl einzelner Best Practices aus unterschiedlichen methodischen Vorgehensweisen (Abb. 1.1):

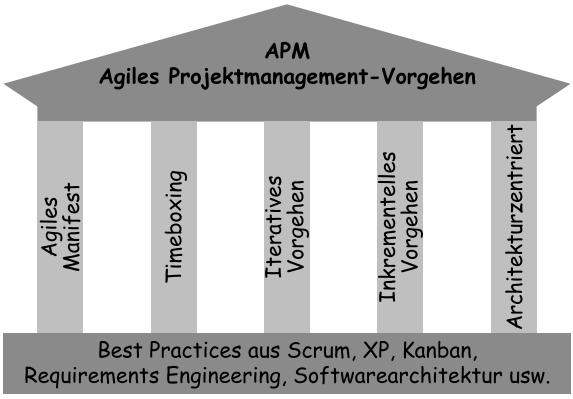

Abbildung 1.1: Das Fundament und die fünf Säulen von *APM*

- *APM* orientiert sich am Agilen Manifest, also an den beschriebenen vier Wertepaaren, und fußt auf den dort genannten zwölf Prinzipien als grundlegende Basis für das Vorgehen, die Führung und die Zusammenarbeit innerhalb des Projekts sowie mit den Stakeholdern [12].
- *APM* implementiert ein durchgängiges Timeboxing zur Planung und Steuerung von Projekten. Eine *Timebox* ist ein fester, vorab definierter Zeitabschnitt, in dem wir planen, ein inhaltliches Ergebnis zu erreichen, das wir am Ende überprüfen können. Dieses Konzept durchdringt *APM* von kleinen Abschnitten der täglichen Zusammenarbeit wie in täglichen kurzen Meetings über feste Zeitabschnitte von wenigen Wochen, Iterationen genannt, bis auf die Ebene der Auslieferungen.
- *APM* ist ein iteratives Vorgehen, d. h., der zeitliche Ablauf ist in gleich lange Iterationen von wenigen Wochen unterteilt, was einem zeitlichen Prüfraster mit festem Rhythmus entspricht.
- *APM* beinhaltet ein inkrementelles Vorgehen. In einer Iteration erarbeiten wir ein prüfbares Ergebnis von direktem Nutzen für den Kunden bzw. die Anwender, also ein Stück funktionierender Software. Das Ergebnis eines Projekts wird schrittweise erarbeitet. Ein Zwischenergebnis bezeichnen wir als Inkrement. Die Inkremente bauen aufeinander auf, sodass wir uns mit jedem weiteren Inkrement dem Projektergebnis konkret nähern. Damit ein inkrementell-iteratives Vorgehen funktioniert, strukturieren wir unser angestrebtes Projektergebnis derart, dass wir ausreichend kleine, sinnvolle Teilstücke als Inkremente innerhalb einer Iteration umsetzen können.
- *APM* ist ein architekturzentriertes Vorgehen. Die Arbeit an der Softwarearchitektur findet kontinuierlich statt, wobei regelmäßig im Projektverlauf grundlegende Architekturfragen im Vordergrund stehen.

Diese fünf zentralen Säulen von *APM* sind eingebettet in eine Vielzahl von Best Practices aus unterschiedlichen Frameworks und Methoden. Im Wesentlichen sind das Scrum, eXtreme Programming (XP), Kanban, agiles Requirements Engineering und Softwarearchitektur (Abb. 1.1). In *APM* wenden wir nicht nur neueste Techniken an, sondern nutzen die Erfahrung aus mehreren Jahrzehnten mit agilem Projektmanagement, Lean Management und inkrementell-iterativen Vorgehensweisen.

Damit ist *APM* besonders dafür geeignet, komplexe Projekte mit einem oder auch mehreren parallel arbeitenden Teams durchzuführen, in denen eine besonders hohe innere und äußere Qualität der Softwareprodukte gefordert ist. Ein wesentlicher Erfolgsfaktor liegt darin, dass die konkrete Vorgehensweise und die Ausprägung von *APM* im laufenden Projekt regelmäßig betrachtet und verbessert werden. Erfahrungen aus dem laufenden Projekt wirken so direkt auf das Projekt selbst und können sofort genutzt sowie den Projektteams kommuniziert werden.

APM stellt ein flexibles Rollenmodell bereit, über das wir weitgehend unabhängig vom Spezialisierungsgrad der einzelnen Teammitglieder crossfunktionale *Featureteams* bilden können, die jeweils gemeinsam verantwortlich einen eigenständig nutzbaren Teil des Softwareprodukts durchgängig und vollständig umsetzen [99]. So entkoppeln wir in *APM* die einzelnen Teams voneinander, sodass gegenseitig induzierte Wartezeiten zwischen den Teams minimiert werden und sich ein kontinuierlicher Projektfortschritt einstellt.

APM kann man auch als umfangreichen Werkzeugkasten um einen im Kern an Scrum angelehnten agilen Ablauf verstehen. Der Werkzeugkasten ist randvoll gefüllt mit zueinander kompatiblen, sich ergänzenden bzw. aufeinander aufbauenden Techniken, aus denen Sie sich zur Lösung bestimmter Aufgaben, Probleme oder Fragestellungen bedienen können. Die einzelnen Zutaten stammen ihrerseits aus unterschiedlichen Ansätzen. So finden sich z. B. auch traditionelle Techniken z. B. in der Projektvorbereitung oder im Risikomanagement wieder. Was zueinander passt, sich ergänzt, funktioniert und nicht den agilen Werten widerspricht, findet sich in *APM* wieder.

Daher passt *APM* erfahrungsgemäß besonders gut in Umfelder mit einer breiten Erfahrung in traditionellem Projektmanagement oder wo zusätzliche Anforderungen wie gesetzliche Auflagen oder andere Regularien zu erfüllen sind. Wenn Sie mit Scrum und Kanban an Grenzen gestoßen sind, weil die laufende Entwicklung stark gewachsen und an Komplexität zugenommen hat und Sie dafür eine Reihe ergänzender Praktiken etablieren möchten, kann Ihnen *APM* ebenfalls weiterhelfen.

Das offene Werkzeugkastenkonzept spiegelt auch die aktuelle Realität wider, in der die Gräben zwischen der agilen und der traditionellen Projektwelt langsam zugeschüttet werden und beeinträchtigende Dogmen beiseite geschoben werden. Wir können nur voneinander lernen.

Auch wenn *APM* auf den ersten Blick durch seinen umfangreichen Werkzeugkasten besticht, so ist es doch wesentlich mehr. *APM* basiert komplett auf der Umsetzung der agilen Werte und Prinzipien aus dem Agilen Manifest. *APM* ist kein traditionelles Vorgehen, das um ein paar Techniken aus Scrum erweitert wurde. Es ist ein durchweg agiles Vorgehen, bei dem vorurteilsfrei geschaut wird, was bei bestimmten Fragestellungen hilfreich sein kann. Was passt, wurde assimiliert.

APM zeichnet sich durch eine Reihe von Merkmalen gegenüber den meisten agilen Frameworks aus. Im Wesentlichen sind das:

- *APM* integriert die aktuellen Ansätze zu einem agilen Requirements Engineering sowie zu einer Use-Case-Analyse und -Modellierung und nutzt damit bewährte und verbreitete Techniken.
- Es ist kompatibel zu verbreiteten Rollenmodellen und kann daher ohne organisatorische Probleme schnell aufgesetzt werden. Es bietet damit

auch für die Mitarbeiter fachliche Karrieremöglichkeiten, die bei anderen agilen Vorgehensmodellen oft vermisst wird.
- Softwarearchitektur ist in ihrer Umsetzung und in die Entwicklungsprozesse vollständig integriert, sodass sie den Stellenwert erhält, der für langfristig erfolgreiche Produkte notwendig ist.
- *APM* beinhaltet neben einem testgetriebenen Vorgehen das agile Testkonzept nach Crispin und Gregory [37], das weit über ein rein testgetriebenes Vorgehen hinausgeht und in dem die Entwicklung begleitende Tests von Anfang an für eine hohe Qualität sorgen.
- Wege zur angemessenen Skalierung wachsender Projekte werden mitgeliefert und müssen nicht mühsam entwickelt und integriert werden.

1.2 Ein paar grundlegende Begriffe vorab

Beim Erklären und Darstellen agilen Vorgehens stoßen wir, wie wir bereits im vorherigen Abschnitt gesehen haben, schnell auf ein rekursives Problem. Wir brauchen einerseits definierte Begriffe, um das Zusammenspiel in agilen Projekten zu erklären, und andererseits das Zusammenspiel in agilen Projekten, um diese Begriffe zu definieren.

Wir versuchen auch für dieses Problem eine iterative Lösung zu finden. Daher beginnen wir mit kurzen Definitionen einiger Begriffe, um danach gleich in die Grundlagen von Agilität vorzudringen.

Eine **Iteration** ist ein vorab festgelegter, fester Zeitraum, in dem wir ein Zwischenziel erreichen wollen. Iterationen dienen uns dabei, ein Projektergebnis schrittweise zu erreichen und diese Schritte zeitlich zu gliedern. Den Teil des Projektergebnisses, der im Laufe einer Iteration neu hinzukommt, nennen wir **Inkrement**. Typische Längen einer Iteration liegen in *APM* zwischen zwei und vier Wochen. In der Regel gibt es in einem Projekt mehrere Iterationen, die direkt aneinander anschließen.

Die Anforderungen an ein Projektergebnis halten wir in einer Liste fest, dem **Backlog**. Wörtlich übersetzt bedeutet Backlog »Auftragsbestand« oder »Arbeitsrückstand«. Das Besondere an einem agilen Auftragsbestand ist seine Ordnung. Die einzelnen Aufgaben, die im Backlog stehen bzw. dort referenziert sind, haben wir eindeutig geordnet. Das wichtigste Ordnungskriterium ist dabei die Bedeutung des Eintrags für den Wert des späteren Projektergebnisses aus Kunden- bzw. Anwendersicht. Des Weiteren ist jeder Eintrag im Backlog geschätzt. Ein Backlog ist also eine geordnete Liste von geschätzen Aufgaben bzw. Anforderungen für ein Projektergebnis.

Die einzelnen geordneten Einträge im Backlog beschreiben die Anforderungen an das Projektergebnis bzw. seine fachlichen oder technischen Pro-

duktmerkmale. Eine weitverbreitete und praktikable Form der Beschreibung eines Eintrags im Backlog ist die sogenannte **User Story**. Um eine solche Anforderung umzusetzen, ist eine Reihe von Tätigkeiten durchzuführen, die **Tasks** genannt werden. Dieser Unterschied zwischen User Stories im Backlog und Tasks ist deshalb wichtig, weil wir formale Kriterien an einen Backlog-Eintrag stellen, die nicht für die Tasks gelten. Wir benötigen beide Sichten, Backlog-Eintrag und Task, um ein agiles Projekt steuern zu können oder das Projektcontrolling durchzuführen.

Einen Zwischenstand unserer kompilierten und zusammengebauten Software wird **Build** genannt. Mindestens einmal pro Tag bzw. Nacht erstellen wir einen Build auf Basis der aktuellen Versionsstände in unserer Konfigurationsverwaltung (Daily bzw. Nightly Build). Noch vorteilhafter sind kontinuierliche Build-Managementsysteme wie z. B. Jenkins. Lässt sich ein Build nicht erstellen, liegt ein sogenannter Build Breaker vor. Auf einem der letzten Builds innerhalb einer Iteration beruht dann das Inkrement. Dieser Build dient damit als Besprechungsgrundlage im Ergebnisreview. Ausgewählte Builds werden in einem **Release** freigegeben und an den Kunden als **Lieferung** (engl.: Delivery) ausgeliefert.

Am Ende einer Iteration finden zwei Meetings statt, das Ergebnisreview und die Retrospektive. Im **Ergebnisreview** demonstrieren wir Mitarbeitern der Auftraggeber- bzw. Kundenseite und anderen Stakeholdern den Fortschritt aus der gerade ablaufenden Iteration. Diese Gelegenheit für Rückmeldungen zum aktuellen Projektergebnis sind essenziell für die konkrete Planung der nächsten Iteration und darüber hinaus für die Möglichkeit, das Projektergebnis für den Kunden bezüglich Kosten, Termin oder anderer Kriterien maßzuschneidern.

Die Durchführung einer **Retrospektive** ist die letzte Aktion in einer Iteration. Wir nutzen sie, um auf den Projektverlauf und die Zusammenarbeit zurückzublicken, oft auftretende Muster zu erkennen, daraus Erkenntnisse abzuleiten und kurzfristig umsetzbare Maßnahmen zu initiieren, um den Projektverlauf zur nächsten Iteration weiter zu verbessern. Während im Review das außen sichtbare Projektergebnis im Zentrum der Betrachtung steht, liegt der Fokus der Retrospektive auf den inneren Abläufen im Projekt.

Zur Steuerung kommen noch zwei verwandte, aber doch in Wirkung und Einsatzbereich unterschiedliche Konzepte hinzu: Timebox und Meilenstein. Beide verknüpfen einen gepanten Inhalt mit einer Dauer bzw. einem Endtermin. Bei einer **Timebox** ist dabei der Termin fest. So definieren wir feste Prüfpunkte, an denen wir den erreichten Inhalt bewerten und ggf. für fehlende Teile den Restaufwand schätzen und deren Bearbeitung einplanen. Eine Iteration ist ein Beispiel für eine Timebox.

Ein **Meilenstein** definiert das Ende einer Phase oder eines Schrittes, zu dem ein bestimmter inhaltlicher Umfang erreicht sein soll. Ist der

gewünschte Umfang nicht erreicht, wird der Restaufwand geschätzt und der Meilensteintermin entsprechend verschoben.

Das wird erst einmal für die Begriffserläuterung genügen. Einen ersten Überblick über das zeitliche Zusammenspiel der Begriffe sowie der dahinter stehenden Konzepte finden Sie in Abbildung 1.2. Die Vorgänger- und Folgeiterationen sind angedeutet und grau eingezeichnet. Als Vorgriff finden Sie eine Verteilung der Aufgaben bzw. Verantwortlichkeiten auf Rollen in *APM*, die links in Abbildung 1.2 dargestellt sind.

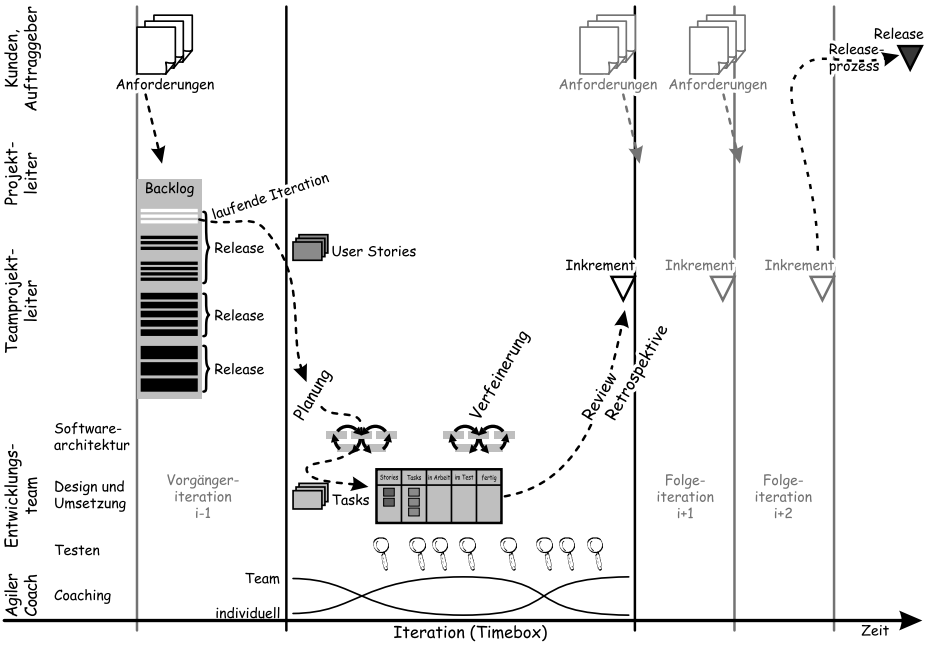

Abbildung 1.2: Das inkrementell-iterative Vorgehen in *APM*

Die **Planung** einer Iteration (engl. Planning) erfolgt in einer Timebox von wenigen Stunden. Sie kann und soll damit auch nicht vollständig sein. Um eine ausreichende Flexibilität zu erhalten, auch innerhalb einer Iteration auf unvorhergesehene Aspekte angemessen reagieren zu können, erfolgt in der zweiten Hälfte der Iteration meist eine weitere **Verfeinerung** der Planung (engl. Refinement) (Abb. 1.2).