Wie Sie auf agile Software-Entwicklung umstellen – Agile Vorgehensweisen

Ausgehend vom Beitrag „Agile Konzepte“ (Kap. 08A01), wird in drei aufeinander aufbauenden Beiträgen die Umstellung auf eine agile Form der SW-Entwicklung beschrieben.

Im vorliegenden dritten und letzten Beitrag werden kontextabhängige Vorgehensmuster zur Umstellung auf agile Entwicklungsmethoden für diese Situationen beschrieben: Entwicklung und Unterhalt umfassender Individuallösungen; Entwicklung und Unterhalt von plattformbasierten Lösungen; SW-Entwicklung als Dienstleistung. Die Vorgehensmuster umfassen die Ebenen des Portfolio- und Programmmanagements sowie die Teamebene und folgen dem agilen (und nicht normierten) Einsatz agiler und traditioneller Methoden und Techniken.

Der erste der drei Beiträge (Kap. 08A02) zeigt, dass diese Umstellung Veränderungen der Unternehmenskultur voraussetzt und erst dann Methoden und Techniken der agilen (aber auch traditionellen) SW-Entwicklung sinnvoll genutzt werden können. Der zweite Beitrag (Kap. 08A03) beschreibt im Detail einige Methoden und Techniken für diese Kulturveränderung.

Autor: Hans-Peter Korn
E-Mail: hans-peter@korn.ch

1 Voraussetzungen für agile Vorgehensweisen in der Software-Entwicklung

Sie möchten Ihre SW-Entwicklung auf ein „agiles“ Vorgehen umstellen? Und zwar – so nehme ich an – als eine langfristig wirksame strategische Investition?

Teil 2

In zweiten Teil dieser Reihe (Kap. 08A03) wurde dargestellt, wie die unabdingbaren kulturellen Voraussetzungen verbessert werden können, um die für die jeweilige Aufgabenstellung der SW-Entwicklung am besten passende Vorgehensweise zu nutzen. Und zwar unabhängig davon, um welche agile (oder nicht agile) Vorgehensweise es sich handelt.

Teil 1

Im ersten Teil dieser Reihe (Kap. 08A02) wurde dies in den Abschnitten „Das agile Vorgehen als strategische Investition“ und „Wie beginnt ein Unternehmen seine ganz individuelle Reise zu seiner persönlichen Art von Agilität?“ und „Zuerst geht die Reise durch den Kontinent der Kultur, dann erst durch jenen der Vorgehensweisen“ begründet:

Wichtig!

Der etablierte Kontinent der Unternehmenskultur bestimmt, wie polymorph der Kontinent der Vorgehensweisen sein kann oder muss, nicht aber umgekehrt.
Investitionen in die Unternehmenskultur sind langfristig und strategisch. Sie bilden die tragfähige Basis für kurzlebige und stets anzupassende Vorgehensweisen. Investitionen nur in Vorgehensweisen sind Verschwendung.

Teil 3

Der hier vorliegende dritte Teil dieser Reihe (Kap. 08A04) geht davon aus, dass Ihr Unternehmen ein gutes Stück seiner Reise in Richtung agile Kultur durch die Landschaften der Kommunikations-, Kooperations- und Führungsmethoden (s. Kap. 08A03) hinter sich bringen konnte und nun über gute Fähigkeiten auf diesen Gebieten verfügt:

- Robustheit
- Belastbarkeit
- Reaktionsfähigkeit
- Flexibilität
- Innovationsfähigkeit
- Anpassungsfähigkeit

Kulturelle Voraussetzungen sind gegeben

Das sind die essenziellen Voraussetzungen, um die für die jeweilige Aufgabenstellung der SW-Entwicklung am besten passende Vorgehensweise nutzen zu können. Und zwar un-

abhängig davon, um welche agile (oder nicht agile) Vorgehensweise es sich handelt.

2 Muster zum Umstellen auf agile SW-Entwicklung

Gehen Sie jetzt bitte zum Abschnitt „Reiseführer durch den Kontinent der Vorgehensweisen“ im ersten Teil (Kap. 08A02). Rekapitulieren Sie dort bitte diese Abschnitte:

- Was sind die Reiseziele?
- Agiler Umgang mit agilen Vorgehensmodellen
- Einladung zur chaotischen Vielfalt?
- Generelle Muster zum Umstellen auf agile SW-Entwicklung
  - Agile Kultur
  - Craftsmanship: SW-Entwicklung als professionelles Handwerk

Das sind Ihre spezifischen Ziele

Rufen Sie sich jetzt jene von Ihnen mit der „Zeitmaschine“ (s. Kap. 08A02, 2) erarbeiteten Ziele Ihres Unternehmens in Erinnerung, die sich insbesondere auf den „Kontinent der Vorgehensweisen“ beziehen, und notieren Sie hier Stichworte dazu:

Anhand dieser Ziele können Sie entscheiden, welche der nachfolgend im Detail dargestellten Methoden und Techniken

der SW-Entwicklung am bes
gehensmuster Sie sich zu
möchten.

„Agil“ nicht unbedingt billiger oder besser

Werfen Sie jetzt nochmals
(wirtschaftliche) Nutzen von
wurde erläutert, dass das agil
zu keiner klaren Kostenred
führt.

Wenn also Ihre primären Zi
weniger SW-Fehler sind, da
agile SW-Entwicklung nicht
geeignete Mittel.

Craftsmanship und Clean Code

Eine handwerklich professio
manship“), beruhend auf d
(siehe Kap. 08A02, 7.4.2), i
Vorgehensweise als solche o

2.1 Vorgehensmuster zur
Entwicklung und Unt
Individuallösungen

Die Charakteristiken und
Unternehmen/Unternehmens
und dem Unterhalt von Indiv
wurden in „Wie reisen Sie
schrieben (Kap. 08A02, 3, P
ser Unternehmen/Unterneh
nehmensinternen IT-Bereich
Portfoliomanagement für d
unterstützenden individuelle
vices.

Wichtig!

Es müssen zwei Arten von Vorgehensweisen in Einklang gebracht werden:

1. Das Neuentwickeln oder grundlegende Überarbeiten kundenspezifischer Individuallösungen.
2. Das als „Arbeitsfluss mit unbestimmtem Ende“ gestaltete Pflegen und Weiterentwickeln der kundenspezifischen Lösungen.

Zusammenspiel mehrerer Ebenen

Unternehmen dieser Art umfassen bis zu mehreren Tausend oft weltweit kooperierender Informatikspezialisten in vielen verschiedenen Teams. Nicht das Vorgehen und die Methodik in einem kleinen Team ist dort der Knackpunkt, sondern die Koordination all dieser Spezialisten und Teams auf Programm- und Portfolioebene. Das erfordert ein Zusammenspiel mehrerer agiler Vorgehens- und Steuerungsebenen:

Portfoliomanagement

- **Agiles strategisches Portfoliomanagement**
  Es dient der Koordination, Priorisierung und mittelfristigen Ressourcenallokation über verschiedene Produktlinien (Applikationsfamilien) und technologische Initiativen (etwa grundlegende Architekturveränderungen) hinweg. Optionen dazu sind SAFe (Kap. 08A01, 3.3.3) oder andere Varianten von Portfolio-Kanban.

Programmmanagement

- **Agiles Programmmanagement**
  Die Kooperation vieler Teams innerhalb einer Produktlinie/Applikationsfamilie (eines „Value Stream“) kann als „Agile Release Train“ z. B. auf der Basis von SAFe (s. „Agile Release Train in [1]) auf der Programmebene koordiniert werden. Es ist „DER“ Schlüssel für die teamübergreifende Steuerung.

Projekte

- **Agiles projektartiges Management zur Neuentwicklung oder grundlegenden Überarbeitung**
  Die Menge an diesbezüglichen Projekten sollte möglichst

klein gehalten werden (s. Kap. 08A01, 3.2). Wann immer möglich, sollten auch Neuentwicklungen oder grundlegende Überarbeitungen im Rahmen der etablierten Agile Release Trains oder – bei einer neuen Produktlinie oder Applikationsfamilie – im Rahmen eines neuen Agile Release Train abgewickelt werden. Die wenigen dann noch wirklich nötigen Projekte müssen in geeigneter Art mit dem Programm- und Portfoliomanagement verknüpft werden. Das agile Management von Projekten ist mit PRINCE2 oder DSDM (s. Kap. 08A01, 3.2.1) möglich.

Teams

- **Agiles Arbeiten im Team**
  Im Vergleich zum agilen Programmmanagement für mehrere parallele Agile Release Trains und dem agilen Portfoliomanagement sind die Vorgehensweisen und Methoden für die (Selbst-) Steuerung der agilen Arbeit innerhalb eines Teams eher einfach. Schwierig wird die Arbeit im Team vor allem bei mangelhafter teamübergreifender Koordination und schlechter Abstimmung zwischen den Teams. Innerhalb der Teams kann z. B. mit Kanban (insbesondere bei fortlaufenden Wartungsarbeiten) oder Scrum oder Scrumban oder DSDM gearbeitet werden.

Altlasten-Bereinigung

- **Das Bereinigen der historisch gewachsenen und sehr heterogenen System- und Anwendungslandschaften** kann mit „Architectural Epics“ (s. „Architectural Epics“ in [1]) auf Portfolioebene und einem „Architectural Runway“ (s. „Architectural Runway“ in [1]) auf Programmebene agil vorangetrieben werden.

Mögliche aufeinanderfolgende Schritte für diese Konstellation sind nachfolgend in 2.1.1 bis 2.1.12 beschrieben.

2.1.1 Das ist bereits erreicht

„Agile Kultur“ Wesentliche Elemente der „agilen Kultur“ (vgl. Kap. 08A03) werden praktiziert, insbesondere:

1. Daily Standup
2. transparente Informationen
3. Sichtbarkeit, Verfolgbarkeit, Analysierbarkeit der Zusammenarbeit
4. Arbeitsplanung im Team durch das Team – nicht durch den Teamleiter
5. kontinuierliche Verbesserung der Kooperation im Team
6. Teamarbeit mit Elementen von Scrum
7. Optimierung der Teamstruktur
8. „agile“ Art der Führung
9. Führen von Teams
10. Teamleiter – neu definiert

Craftsmanship und Clean Code Craftsmanship (SW-Entwicklung als professionelles Handwerk) (vgl. Kapitel 08A02, 7.4.2) ist etabliert, insbesondere:

- Nicht nur funktionierende Software, sondern gut gefertigte Software ist das erklärte Ziel.
- Clean Code Development prägt die Arbeitsweise [2].
- Professionelles Requirements Engineering mit schrittweiser Verfeinerung auf der Basis eines JEDUF (Just Enough Design Up Front) (s. Kap. 08A01, 2.1) statt BDUF (Big Design Up Front) ist akzeptiert und wird weitgehend praktiziert.
- Test Driven Development (TDD) und Acceptance Test Driven Development (ATDD) und die Integration von Entwicklung und Test werden fortlaufend verbessert.

- Continuous Integration, C
tomation werden fortlauf
- Aktuelle Applikations- u
liert und wird schrittweis
- Serviceorientierte Archit

2.1.2 Produktlinien/Appli Stream

Langlebige Wertströme . . . „Value Streams“ sind langl
eines wirtschaftlichen Nutze
aller Anforderungen, Systen
und Auslieferungsschritte, d
tes Geschäftsthema (z. B. ei
sicherungsgeschäfts) ein int
und zu warten, das dem U
Kunden/Kundenkreis und d
grenzten Fluss von Mehrwe
[1]).

Ich benutze hier und im Folg
3.3.3) als Illustration für ein
wicklung, das auch das Progr
umfasst. Es ist dem Leser üb
Kontext nutzen möchte.

. . . statt nur einzelner Projekte „Value Streams“ sind keine
grenzten – Projekte. Bestim
ferungsschritte (etwa zur Ne
tegrierten Systems) können
nes Value Stream angesiedel

Value Streams und Portfolio-management

Value Streams sind abgeleitet aus und verbunden mit den längerfristigen und umfassenden strategischen Themen und Initiativen des unternehmensweiten Portfoliomanagements.

Die bisherige Form des Portfoliomanagements kann jetzt noch beibehalten werden. In der Regel ist dies auf die Budgetierungsperiode für ein ganzes Geschäftsjahr angelegt und daher nicht sehr flexibel. Es kann in einem späteren Schritt flexibilisiert werden. Allerdings:

Wichtig!

Das Portfolio ist grafisch (nicht als „Zahlenfriedhof“) für alle einsehbar. Und zwar so, dass der aktuelle und zukünftige ungefähre Arbeitsumfang pro Value Stream und Quartal erkennbar ist und somit auch angemessen begrenzt werden kann.

2.1.3 Value Streams als Agile Release Trains (ARTs) implementieren

ART als Rückgrat der SW-Entwicklung

Ein „Agile Release Train“ (s. „Agile Release Train“ in [1]) ist das Gefäß für die sich selbst managende Gemeinschaft all jener Teams, die Beiträge für einen bestimmten Value Stream leisten. Agile Release Trains (ARTs) sind das Rückgrat der agilen SW-Entwicklung umfassender Lösungen unter Beteiligung von hundert bis mehreren tausend Personen. Jeder ART ist ein organisatorisches Konstrukt zum kontinuierlichen Schaffen (Definieren, Entwickeln, Ausliefern, Warten) von Geschäftswert für einen Value Stream und ermöglicht die Ausrichtung aller zu ihm gehörenden Teams auf gemeinsame Ziele und das Management der Abhängigkeiten zwischen den Teams durch die Teams selbst.

Im Zug sitzen alle beteiligten Teams

Die zu einem ART beitragenden Teams umfassen typischerweise insgesamt 50 bis 100 Personen. Bei mehr als 150 Personen (entsprechend der Dunbar-Zahl, das ist die maximale Anzahl Personen, mit denen eine Einzelperson soziale Beziehungen pflegen kann) muss der Value Stream thematisch so unterteilt werden, dass sich dann mehrere ARTs bilden lassen.

In Unternehmen mit weniger als 100 Entwicklern sollten einzelne ARTs nicht weniger als je drei Teams (insgesamt etwa 20 Personen) umfassen. In Unternehmen mit weniger als etwa 25 Entwicklern können alle einen einzigen ART bilden, aber nur dann, wenn sie gemeinsam an einem einzigen funktionalen Themenbereich (einem Value Stream) arbeiten.

Keine übergreifenden Teams!

Entwicklungsteams müssen so gebildet sein, dass jedes Team Beiträge nur für einen einzigen ART leistet. Innerhalb jedes ART (und nicht ART-übergreifend!) gibt es ein „Deployment Operations Team“ (s. „DevOps“ in [1]).

Releases = Haltestellen des ART

Die Teams, die im selben ART, also „im selben Zug“, sitzen, sind auf ein gemeinsames Ziel und einen gemeinsamen Zeitplan und Lieferrhythmus auf der Basis von Releases – die Haltestellen des ART – ausgerichtet und unterstützen so den gleichmäßigen Fluss der Produktentwicklung.

2.1.4 Überprüfen und anpassen der Teamstruktur

Können alle Teams zu 100 % einem und nur einem ART zugeordnet werden?

Können die Teams innerhalb eines ART möglichst unabhängig von anderen Teams oder teamexternen Spezialisten arbeiten? Also so:

- Jedes Team verfügt über permanente, stets voll ausgelastete und daher ungeteilte Mitglieder/Spezialisten, die ins-

gesamt zum Definieren/Realisieren/Testen der Softwareprodukte oder -services des Teams nötig sind.

- Jedes Team kann pro Release (also innerhalb von zwei bis sechs Monaten, nicht unbedingt alle zwei bis vier Wochen) und allein (d. h. ohne umfassende Zulieferungen anderer Teams und nur gelegentlich unterstützt von temporär hinzugezogenen Spezialisten) eine ins Gesamtsystem integrierte und technisch und funktional getestete SW-Teillösung einer teamübergreifenden End-to-End-Funktionalität des ART herstellen.
- Die SW-Teillösung eines Teams ist abgegrenzt durch möglichst wenige und gut definierbare Schnittstellen zu den SW-Teillösungen der anderen Teams.
- Die SW-Teillösung jedes Teams ist zu Beginn der Release-Bearbeitung so weit definiert, dass die Schnittstellen als „Stubs“ realisiert werden können.
- Jedes Team realisiert innerhalb der Release-Abarbeitung seine SW-Teillösung so, dass innerhalb dieser Stubs Zwischenresultate möglichst früh technisch integriert und funktional getestet werden können, um den teamübergreifenden Integrations- und Testaufwand am Ende der Release-Entwicklung zu minimieren. Die Ergebnisse der regelmäßigen (idealerweise kontinuierlichen) Integration werden während der Release-Bearbeitung in mehreren teamübergreifenden „System Demos“ (s. „System Demo“ in [1]) vorgeführt.

Pro ART Feature- und Component-Teams

Wie in Kap. 08A03, 3.5.3 beschrieben, wird es in jedem ART sowohl Feature- als auch Component-Teams geben.

Sehr oft kann das alle Teams eines ART umfassende Deployment der Teamergebnisse nicht von jedem Team selbst geleistet werden. Dann wird es pro ART ein Deployment-Operations-Team (s. „DevOps“ in [1]) geben. Ein mehrere

ARTs übergreifendes Deploy
zu empfehlen.

2.1.5 „WAS“ vom „WIE“ Backlog Owner

Die Teamleiter sind herköm
wohl dafür, „WAS“ das Tea
„WIE“. Der Vorteil ist, dass
Teams betrachtet den Teaml
ner für dieses Team gibt. D
dann allzu sehr von den Mög
die Arbeit leisten kann (Kor
Spezialisten, Entwicklungsi
jedoch widerspricht dem Zie
zer den optimalen „Wert zu
Lean (s. Kap. 08S01, 2.2).

„WAS“ formulieren = Produkt-management

Deshalb ist es essenziell, die
und das „WIE“ auf zwei R
getrennt bleiben. Damit dies
die Verantwortung für das „
Teamleiter und auch niemand
sollte besser eine Person aus

„WAS“ mit Team erarbeiten . . .

Das „WAS“ beschreibt all d
künftig zu liefern hat: Use
technische Stories, Fehlerbe
Spikes, Refactorings und so
anlässlich der Release-Planu
des ART teil) vom Team erb
tiert ist das im „Team Backl
einer nach Abarbeitungsreih
ser Dinge.

. . . verantwortlich bleibt Backlog Owner

Verantwortlich jedoch für diese Liste und alles, was dort enthalten ist, sowie für die Reihenfolge ist ausschließlich der „Team Backlog Owner“, der zum Agilen Team gehörende „Product Owner“ (s. „Product Owner in [1]). Er ist als einzige Person für die Verwaltung des Team Backlog verantwortlich auch dann, wenn er einzelne seiner Aufgaben von anderen Personen (etwa im Team oder vom Produktmanagement) ausarbeiten lässt.

Team Backlog statt Product Backlog

Ich spreche hier ganz bewusst nicht (wie bei Scrum) von „Product Owner“ und „Product Backlog“, sondern vom „Team Backlog Owner“ und „Team Backlog“. Denn dann, wenn etliche Feature- und Component-Teams gemeinsam ein umfassendes SW-Produkt herstellen, ist es nicht realistisch, dass dieses Produkt von nur einem einzigen Product Owner verantwortet wird, der für jedes Team ein spezifisches Backlog verwaltet und mit jedem Team die Planungen und Reviews macht. Eingebürgert hat sich die Praxis, dass alle Teams eigene Owner für ihre Backlogs haben (allenfalls für zwei Teams einen), die dann ein „Team der Owner“ bilden, das dann vom „eigentlichen“ Product Owner repräsentiert wird. Außerdem ist es weit hergeholt, bei einem Component Team von einem „Product Backlog“ und einem „Product Owner“ zu sprechen, da dieses Team ja Systemkomponenten oder systemnahe Services und keine „Produkte“ im eigentlichen Sinn realisiert.

Nur der Team Backlog Owner verantwortet das „WAS“

Der Team Backlog Owner entlastet das Team vom Klären widersprüchlicher und vom Priorisieren vieler gleichzeitig vorliegender Anforderungen diverser Anspruchsgruppen. Jedes Team verhandelt das „Was“ seiner Arbeit ausschließlich mit seinem Team Backlog Owner, einschließlich der nichtfunktionalen Anforderungen. Die Teams sind für den von ihnen realisierten Lieferumfang nicht mehr gegenüber ihrem Teamleiter und diversen anderen Anspruchsgruppen (Auftraggeber, Nutzer, QA, Architektur . . .), sondern nur noch ihrem Team Backlog Owner gegenüber direkt verantwortlich.

Wichtig!

Der Team Backlog Owner kann jedoch gegenüber all diesen Anspruchsgruppen in größeren Organisationen nicht als der zuletzt allein Entscheidende auftreten. Das geschieht im Produktmanagement.

Der Interessenabgleich hat im Produktmanagement auf den Ebenen des Programm- und Portfoliomanagements stattzufinden (s. „Product Owner“, Tabelle 1 in [1]). Die Formulierung im Scrum Guide [3]:

„The Product Owner is one person, not a committee . . . For the Product Owner to succeed, the entire organization must respect his or her decisions.“

ist in größeren Organisationen nicht umsetzbar. Wichtig bleibt jedoch, dass nur der Team Backlog Owner die Ergebnisse des Interessenabgleichs durch das Programm- und Portfoliomanagement im Backlog festhält und sie nur von ihm allein gegenüber dem Team vertreten werden.

2.1.6 Die Teamleiter bleiben Rahmensetzer für das „WIE“

Wie-Verantwortliche

Die bereits vorhandenen Teamleiter werden „Wie-Verantwortliche“ im Sinn eines Scrum Master oder Agile Master (s. „Scrum Master“ in [1] und in [3], Abschnitt „The Scrum Master“). Sie sind nicht mehr verantwortlich für das „Was“. Sie bleiben die Vorgesetzten der Entwickler. Das ist auch kein Widerspruch zu Scrum (vgl. Kap. 08A03, 4.4 u. 4.5) oder eine

Behinderung der Selbstverwaltung des Teams, sondern eine für das Team wichtige Unterstützung [4].

Die Team Backlog Owner hingegen sollten zur Organisationseinheit „Produktmanagement“ oder „Servicemanagement“ gehören. Dass damit die Teamleiter (bzw. die Teams als solche) nicht mehr für das „Was“ zuständig sind, führt oft zu Irritationen bei den Kunden oder Nutzern. Bisher konnten sie ja direkt den Teamleiter oder ein Teammitglied ansprechen, um Änderungswünsche erfüllt zu bekommen.

Agile Teams als Org.-Einheiten der Linie mit Teamleitern

Die agilen Teams sind keine temporären Projektteams, sondern längerfristig stabile Feature- oder Component-Teams. Deshalb werden sie als Organisationseinheiten der Primärorganisation mit jeweils einem Teamleiter als Führungskraft gebildet. Nur so ist gewährleistet, dass die individuelle Förderung der Teammitglieder auf der Basis der Tag für Tag beobachteten Arbeit erfolgt und nicht „aus der Entfernung via Informationen aus dritter Hand“, wie es bei Matrixorganisationen oft der Fall ist.

Der Teamleiter setzt gegenüber dem Team und vertritt und verteidigt gegenüber außen den Rahmen für das „Wie“ zur Durchsetzung des Grundsatzes „Nicht nur funktionierende Software, sondern auch gut gefertigte Software“ und der Prinzipien des Clean Code Development. Das Team konzentriert sich innerhalb dieses Rahmens auf die professionelle Realisierung und das Testen des vom Team Backlog Owner vertretenen „Was“.

2.1.7 Pro „ART“ die Zeitpunkte der externen und internen Releases bestimmen

Produktinkremente in kurzen Intervallen oft nicht machbar

Idealerweise wird beim agilen Vorgehen in jeweils sehr kurzen Zeitabständen ein auslieferbares Software-Inkrement er-

zeugt. Beim iterativen Vorge
das alle zwei bis vier Woche
hen (etwa bei Kanban) imme
aber, insbesondere wenn et
Software-Inkrement arbeiten
genen Systemen, ist es schli
kurzen Abständen ein über
hinweg im Gesamtsystem g
higes Inkrement auszuliefern
gen Benutzer- und Betriebsd
Abb. 3).

Überforderung der Nutzer vermeiden

Und die Nutzer? Sind sie gew
Wochen oder noch häufiger
funktionen zu erhalten und
läufe darauf abzustimmen?

Zeitpunkte externer Releases

Vom Produktmanagement zu
Fragen:

- Wie groß ist die Dynamik
  Funktionalitäten (Nutzer
  werber)?
- Wie groß also ist pro ART
  der Anforderungsänderun
  Anforderungen pro Woch
- Wie häufig wünschen die

Die Antworten bestimmen di
effektiv ausgelieferten und pr
ART.

Wenn pro ART die Zeitpunk
ßigen) externen Releases mi

sich folgende weitere pro ART zusammen mit allen Teams des ART zu beantwortende Frage:

- Wie groß ist die optimale Dauer der INTERNEN Releases?

Zeitpunkte interner Releases

Interne Releases sind für die eigentlichen Nutzer unsichtbar. Sie dienen der Software- oder Produktentwicklung zur kurzfristigen und realitätsnahen Integration und Validierung der Resultate (s. 1 in „Release“ in [1] und Abb. 3 in „psi-release“ in [1]).

PSI

Der Takt interner Releases ist regelmäßig und nie länger als jene minimale Dauer, in der alle zum internen Release beitragenden Teams gemeinsam ein integriertes „Potentially Shippable Increment“ (PSI) liefern können.

Und zwar so, dass die Beiträge aller Teams in eine betriebsnahe Integrationsumgebung erfolgreich überführt und dort sowohl technisch (inklusive aller nichtfunktionalen Anforderungen) als auch fachlich erfolgreich getestet werden können. Ein Richtwert in großen Unternehmen mit heterogenen Systemlandschaften sind acht bis zehn Wochen.

2.1.8 Programmmanagement pro ART agil gestalten

Abschied vom „eisernen Dreieck“

Das Programmmanagement (s. „Program Portfolio Management“ in [1]) dient der teamübergreifende Koordination der Funktionalitäten umfangreicher Systeme pro ART als Abfolge stets gleich langer Iterationen (= interne Releases) von acht bis zehn Wochen mit fixierten Qualitätsanforderungen, jedoch mit mittel- und langfristig veränderlichem Umfang (kein „eisernes Dreieck“: fester Umfang → fixe Termine → feste

Kosten, s. Kap. 08S01, 3.2). Ergebnisse des ART sind regelmäßig vorhandene Potentially Shippable Increments (PSIs). Sie werden in einer dynamischen „Roadmap“ (s. „Roadmap“ in [1]) geplant.

Dynamische Roadmap

Pro ART zeigt eine dynamische Roadmap für die zukünftig geplanten Termine der INTERNEN Releases, das Thema, die jeweiligen Ziele und die geplanten priorisierten Features.

Nur nächstes PSI verpflichtend

Die Beschreibung des unmittelbar nächsten internen Releases der Roadmap stellt eine Verpflichtung gegenüber dem Unternehmen dar. Für die Ausgestaltung der späteren Releases gibt es keine verbindlichen Zusagen, deren Umfang ist bestenfalls unscharf.

Kunden wollen längerfristige und verbindliche Zusagen

Dass Pläne für zukünftige interne Releases nicht für unternehmensexterne Verpflichtungen – also gegenüber den Kunden – genutzt werden sollten, ist bei vielen Kundenaufträgen jedoch schwierig durchzusetzen. Gegenüber den Kunden wird es in der Regel eine längerfristige und verbindlichere Planung zukünftiger, jedoch EXTERNER Releases der Kundenlösungen geben müssen – mit dem damit verbundenen Risiko, dass sich diese längerfristigen Planungen nicht umsetzen lassen.

Option: Agiler Festpreis

Verträge auf der Basis des „Agilen Festpreises“ [5] sind eine Lösung, wenn es ein gutes Vertrauensverhältnis mit dem Kunden gibt.

Die Roadmap repräsentiert die jeweils aktuelle Sicht der Prognosen des Unternehmens für die zukünftigen Releases. Diese Sicht ist stets Änderungen unterworfen, da sich Entwicklungsergebnisse, Geschäftsprioritäten und Kundenbe-

dürfnisse ändern können. Deshalb sollen Pläne für zukünftige interne Releases nicht für unternehmensexterne Verpflichtungen genutzt werden. Verpflichtend gegenüber dem Unternehmen ist nur der unmittelbar bevorstehende interne Release. Inwieweit daraus auch eine Verpflichtung für einen dem Kunden ausgelieferten (also externen) Release abgeleitet wird liegt in der Verantwortung des Produktmanagements.

Transparente Planung, Begrenzung der Arbeiten

Eine für alle einsehbare Grafik pro ART zeigt pro internen Release (PSI) den aktuellen und zukünftigen ungefähren Arbeitsumfang pro Team bezogen auf 70 % seiner Maximalleistung. Damit kann der anstehende Arbeitsumfang angemessen begrenzt werden.

Produktmanager im Programmmanagement

Auf der Ebene des Programmmanagements legen die Produktmanager (es gibt aber auch viele andere Bezeichnungen für diese Rolle) die Features des Systems als dynamische Roadmap fest. Die Rolle des „Product Owners“ gemäß Scrum Guide ist dafür dann gut anwendbar, wenn jeweils ein Team allein ein Produkt, vertreten von einem Product Owner, entwickelt. In größeren Unternehmen jedoch sind, wie bereits vorher in 2.1.5 beschrieben, die Verantwortlichkeiten eines Product Owner im Sinne von Scrum oft so breit, dass sie auf technologisch orientierte Team Backlog Owner und auf den Markt oder die Nutzer fokussierte Produktmanager verteilt sind. Letztere nehmen ihre traditionellen Verantwortlichkeiten sowohl bezüglich der Produktdefinition als auch der Präsentation der Lösung gegenüber dem Markt wahr. Unabhängig von der firmenspezifischen Bezeichnung nimmt der Produktmanager/das Produktmanagement vor allem folgende Verantwortungen wahr:

Verantwortung Produktmanagement

- Ownership der Programm
gramm-(Release-)Backlo
- Managen des Release-U
Ebene
- Pflegen der Produkt-Roa
- Aufbau eines effektiven
Owner-Teams pro ART

2.1.9 Planung des komme eines „ART“

Jede Timebox zur Realisieru
eines ART beginnt mit einem
Release-Erarbeitung. Diese
PSI dient der Klärung seines
der am ART beteiligten Te
Releases. Die Vorgabe für da
aktuelle Vision zusammen
Releases und der Zusammen
priorisierten Features im Pro

Planungsmeeting mit ALLEN Mitgliedern aller beteiligten Teams

Im Unterschied zu sonst übl
grammmanagements planen
beteiligten Teams (also typis
das Release gemeinsam an e
bewährter Techniken zur M
Abb. 3 in „Release Planning
für das nächste PSI in Bet
Programm-Backlogs in jene
beiten können. Das ergibt di
PSI (den anstehenden inte
Prozesses klären die Teams
Teams. Sie schätzen den Um
in Kenntnis ihrer Kapazität

sie leisten oder nicht leisten können und damit über Veränderungen des Release-Umfangs mit dem Team Backlog Owner und dem Produktmanagement. Zusätzlich zu dieser Planung verpflichten sich die Teams gegenüber ihrem Team Backlog Owner auf die zu erreichenden Release-Ziele und priorisierten Features. Die Ergebnisse der Release-Planung fließen in die Aktualisierung der Roadmap des ART ein.

Ist das nicht viel zu aufwändig und daher „unbezahlbar“?

Ungewöhnlich – aber hoch effizient!

Hier die Rechnung:

Zehn Teams (= 60 Personen) arbeiten gemeinsam an einem internen Release eines ART und es gibt fünf interne Releases pro Jahr:

→ 2 Tage × 5 Releases × 60 Personen = 600 Personentage/Jahr

→ Pro Person 180 „produktive“ Tage pro Jahr × 60 Personen = 10.800 Personentage/Jahr

Nur 5,5 % der Kapazität!

Diese kollaborative Release-Planung beansprucht nur 5,5 % der Entwicklungskapazität!
Das ist wesentlich weniger als der Aufwand für die Behebung der erst im Verlauf der Release-Erarbeitung erkannten Koordinationsmängel, die dann auch nur „notdürftig und on the fly“ bereinigt werden und oft zu Folgeproblemen führen.

Agil = Kooperation/ Konversation, nicht Auftragserteilung „von oben“

Wenn solche kooperativen Arbeitsweisen als viel zu teuer abgelehnt werden mit dem Hinweis, dass es ja genüge, wenn die Team Backlog Owner allein (also nur zehn statt 60 Personen) all das in nur einem halben Tag erledigen, ist das ein

Zeichen für eine immer noch stark auf „Auftragserteilung von oben“ und „Koordination durch Repräsentanten“ geprägte Kultur. Es hat sich dann noch nicht die Einsicht durchgesetzt, dass ein Unternehmen erst durch die fortlaufende Konversation aller das Unternehmen umfassenden Mitarbeitenden und Stakeholder entsteht.

„Agil“ als Fassade?

„Agile“ Vorgehensweisen werden dann bestenfalls nur als „Fassade“ sichtbar sein, nicht aber gelebt werden. Das jedoch erzeugt mehr Irritationen und Verunsicherungen als Nutzen. Wenn die Kultur nicht stimmt, sollten besser die alten Vorgehensweisen praktiziert werden.

2.1.10 Agile Arbeit auf Teamebene

Inwieweit einzelne Teams innerhalb dieser PSI-Erarbeitung in Anlehnung an Scrum in Sprints, kontinuierlich mit Kanban oder Scrumban, einer Kombination aus beidem, oder nach eigenem Muster arbeiten, kann den Teams überlassen werden. Für die Teams ist mit der Relase-Planung pro ART der Rahmen für das „WAS“ gesetzt. Der Teamleiter hat die Verpflichtung, den Rahmen für das „WIE“ zu setzen, um die Erreichung des „WAS“ zu erleichtern.

Jedes Team wählt seine Vorgehensweise selbst

Die passenden Details der Arbeitsweise – das Bild im Rahmen – können die Teams selbst wählen. Es liegt insbesondere an den sehr eng kooperierenden Teams, sich selbst gemeinsam angemessen zu organisieren und allenfalls synchronisierte Sprints zur besseren Kooperation und Transparenz zu vereinbaren. Teams mit einem hohen Anteil fortlaufend und unplanbar eintreffender Wartungs- und Supportaufgaben hingegen arbeiten besser auf der Basis von Kanban oder Scrumban mit regelmäßigen teamübergreifenden „System Demos“.

Entscheidend ist, dass jedes Team seine in der Release-Planung zugesagten Ziele erreicht. Das „WIE“ liegt in der Verantwortung der Teams.

Keine Methode um ihrer selbst willen

Schenken Sie „Methodendogmatikern“ keinen Glauben. Glauben Sie vor allem nicht daran, dass ein bestimmtes Vorgehen wie insbesondere Scrum zuerst 1:1 gemäß einem der beiden heute vorliegenden Regelwerke (dem „Scrum Guide“ der **scrum.org** und dem „Core Scrum“ der Scrum Alliance) praktiziert werden muss, bevor es teamspezifisch angepasst wird. Gehen Sie auch hier von Anfang an agil vor: Praktizieren und intensivieren Sie im Team all das, was passt, und verändern Sie Ihre Praktiken schrittweise. Die dokumentierten Regeln der diversen Rahmenwerke dienen der Orientierung, nicht als „Gesetze“. Und wenn sich eine dieser Regel als „schmerzhaft“ herausstellt, dann hören Sie damit auf und machen Sie etwas Passenderes. Ihr Unternehmen lebt nicht vom Einhalten fremder Regeln, sondern von den Ihre Kunden zufriedenstellenden und sie hoffentlich auch begeisternden Produkten und Dienstleistungen, die das Unternehmen „schmerzfrei“ und gewinnbringend realisieren kann.

Denken Sie auch stets daran, dass auf der Ebene des Teams die SW-Entwicklung als professionelles Handwerk (Craftsmanship) wichtiger ist als das Befolgen irgendwelcher teamextern definierten Vorgehensregeln, um nicht nur funktionierende, sondern gut gefertigte Software zu realisieren.

Tagesaktuell sichtbarer Arbeitsfortschritt

Für die Arbeit bis zur Lieferung des nächsten PSI (interne Releases) haben alle Teams unabhängig von ihrem Vorgehen jedoch diese Transparenz zu gewährleisten:

Der Arbeitsfortschritt der ein
alle einsehbar – aber nicht an
oder anhand erarbeiteter Kon
Verhältnis aus:

- Umfang der in der Integra
  gegenüber „stubs“) fun
  Backlog Owner akzeptier
  realisierten Prototypen

im Vergleich zum:

- Gesamtumfang der bis zu
  Softwareinkremente und

Gleichzeitig steht die Siche
der Menge der in jedem PS
dergrund. Deshalb gibt es di

Team Backlog Owner rechenschaftspflichtig für SW-Qualität

Der Team Backlog Owner ist
Ergebnisse zu akzeptieren, di
funktionalen Anforderungen,
Softwarequalität, vollständig
Er ist rechenschaftspflichtig
tigen Überprüfungen (evtl. au
vorgenommen wurden.

2.1.11 Projekte innerhalb

Projekte passen nicht zur agilen Entwicklung im ART

„Projekte“ in ihrer in der IT
hen im Widerspruch zur
Kap. 08A01, 3.2). Anderers
ganisatorisches und kommer
Fokussierung darauf, bestim

definierten Zeit mit definierten Kosten zu erzielen, und es schafft dafür spezielle temporäre Verantwortlichkeiten (Lenkungsausschuss, Projektleiter, Projektteam). Wenn jedoch in einem Unternehmen alles als „Projekt“ betrachtet und organisiert wird, führt das zu kaum zu bewältigenden Koordinationsproblemen, zu einem Übermaß an projektbedingten Meetings und bei den in der Regel gleichzeitig an mehreren Projekten beteiligten Mitarbeitenden zu einem die Produktivität beschränkenden Multitasking.

Mit Widerspruch pragmatisch umgehen

Mit dem Widerspruch zwischen der projektunabhängigen kontinuierlichen Arbeit in einem ART einerseits und der zeitlich begrenzten Fokussierung auf Projekte kann so umgegangen werden:

Fokussierte Releases statt Projekte

- Verhindern Sie eine ausufernde „Verprojektisierung“: Initialisieren Sie Projekte nur dann, wenn diese Fokussierung im Rahmen der üblichen Arbeiten eines ART sicher nicht erreicht werden kann. Initiieren Sie also kein Projekt, wenn z. B. ein oder mehrere Releases fast ausschließlich einem speziellen Fokus (etwa der Entwicklung eines neuen Funktionskomplexes innerhalb des ART) gewidmet werden können.

Massive Reduktion von Projekten

- Formulieren Sie enge Kriterien dafür, was „echte“ Projekte von eher „normalen“ Aufgaben und Vorhaben unterscheidet, so, dass höchstens 25 % ihrer aktuellen Projekte als „echte“ Projekte verbleiben.

Keine Großprojekte

- Projekte mit einer Durchlaufzeit von mehr als sechs Monate sollte es nicht geben. Unterteilen Sie umfassendere Vorhaben in kleinere Projekte, die auf der Ebene des Programmmanagements eines ART koordiniert werden. Denken Sie daran: Gemäß Abbildung auf Seite 4 im CHAOS-MANIFESTO 2013 der Standish Group [6] waren kleine Projekte von 2003 bis 2012 fast achtmal erfolgreicher als große – egal, ob sie (gemäß Abbildung auf S. 25 in [6]) „agil“ oder mit „Wasserfall“ abgewickelt wurden. Der Projekterfolg korreliert mit der Projektgröße, „agil“ oder „Wasserfall“ machen kaum einen Unterschied.
- Und: Projekte dürfen nur jeweils einen ART betreffen – andernfalls müssen sie in Teilprojekte pro ART unterteilt und auf der Ebene des Portfoliomanagements koordiniert werden.

Projekte sind Konstrukte im Programmmanagement

- Projekte sind wenn immer möglich Konstrukte auf der Ebene des Programmmanagements eines bestimmten ART. Deren „deliverables“ sind in das Programm-Backlog des ART integriert und dort speziell markiert. Es gibt keine vom Programm-Backlog des ART getrennten Projekt-Backlogs. Die für Projekte zu leistenden Arbeiten fließen anlässlich der Release-Planungen des ART in die Team Backlogs ein und sind dort speziell markiert.
- Auf Teamebene können die Arbeiten für einzelne Projekte in den Planungstafeln in speziellen „swimlanes“ erscheinen.
- Dass Projekte möglichst nur als Konstrukte im Programmmanagement behandelt werden, sollte auch bei Neuentwicklungen individueller Kundenlösungen praktiziert werden, auch wenn sie auf Kundenseite in der Regel als „Projekte“ gesehen und beauftragt werden.

Projektspezifische Teams als Ausnahme

- In speziellen Fällen kann innerhalb eines ART für ein Projekt ein temporäres Projektteam, bestehend aus Mitgliedern der schon bestehenden Teams, gebildet werden.
- Als absolute Ausnahme kann für ein Projekt ein weiterer – zeitlich begrenzter – ART mit mehreren Teams gebildet werden. Es ist dann jedoch zu vereinbaren, welche SW-Teile nach Projektabschluss von welchen permanenten ARTs zur weiteren Betreuung übernommen werden und welche Teams dieser permanenten ARTs vorher bereits in

die Arbeit des temporären „Projekt-ART“ einbezogen werden.

PM-Methoden müssen JEDUF und Produkt-inkremente unterstützen

- Für das Management von Projekten innerhalb einer agilen Kultur eignen sich PM-Methoden, die mit Prinzipien wie „Just Enough Design Up Front“ und „Inkrementelle Lieferung von Ergebnissen“ gut vereinbar sind. Das trifft insbesondere für PRINCE2 und DSDM zu. Die Rolle des Projektleiters ist dabei im Bereich des Programmmanagements/Team Backlog Owner anzusiedeln, nicht auf der Ebene des Teamleiters/Agile (Scrum) Master.

2.1.12 Agiles Portfoliomanagement

Portfolio-management zur Umsetzung der Geschäfts-strategie

Das Portfolio (alles zu „Portfolio“ in [1]) umfasst die Gesamtheit der strategischen Themen (s. Abb. 1 in „Portfolio Vision“ in [1]). Deren Prioritäten steuern die Investitionen auf Unternehmensebene. Aufgabe des Portfoliomanagements ist es, dass die zukünftig geleistete Arbeit tatsächlich der Umsetzung der Geschäftsstrategie dient. Die strategischen Themen bestimmen die Portfolio-Vision, dargestellt als umfangreiche Initiativen in Form von Epics. Sie fließen im Verlauf der Zeit in verschiedene Agile Release Trains ein.

Die Verantwortung für die Erarbeitung der Themen und Epics tragen die Portfoliomanager, das sind entweder die „Owners“ der Geschäftsbereiche oder die Produktgremien und andere Personen, die gegenüber ihren Stakeholdern verantwortlich sind.

„Themen“ des Portfolios sind langlebig

Zweck der vom Portfoliomanagement erarbeiteten Themen ist es, auf dem Markt differenzierende und konkurrenzfähige Kernprodukte anzubieten. Diese Themen haben eine weitaus längere Lebensdauer als die Epics. Gruppen solcher Themen können bis zu einem Jahr oder länger unverändert bleiben.

Es gibt zwei Arten von aus
leiteten Epics:

Business-Epics

- Business-Epics stellen d
zerbedürfnisse dar.

Architektur-Epics

- Architektur-Epics repräs
Vorhaben als Voraussetz
der Lösungen, die die la
nehmensbedürfnisse unt
Systemarchitektur in ei
mäßige Investitionsüberl

„Epics“ beschreiben wertschaffende Ergebnisse eines „Themas“

Epics sind Entwicklungsvor
fender Ergebnisse für ein In
Portfolio Backlog festgeha
laufend gepflegt. Vor der Re
sie vom Programmanagem
ners in spezifische Features

Epics können dargestellt w
nutzersprachlicher „Stories“
deo, als Prototyp oder in irg
eignet ist, die Absicht der
Ziel der Epics ist die Beschr
nicht die Genauigkeit. Mit an
so weit im Detail beschrieb
künftige Diskussion darüber
Features ein Epic impliziert.

Je ein Portfolio-Kanban für Business- und Architektur-Epics

Busines-Epics werden im Ka
(s. „Business Epic Kanban
Kanban-System der Architek
Kanban“ in [1]) erfasst. In b
der weitere Bearbeitungsflus
ban typischen „Work In Prog

Wichtig!

Damit wird die bisher übliche recht starre Jahresplanung durch eine rollende und flexible Planung ersetzt.

2.2 Vorgehensmuster zur Umstellung auf agile Entwicklung und Unterhalt von plattformbasierten Lösungen

Die Charakteristiken und Herausforderungen von großen Unternehmen/Unternehmensbereichen mit der Entwicklung und dem Unterhalt von plattformbasierten Lösungen als Hauptaufgabe wurden im Abschnitt „Wie reisen Sie mit Ihrem Unternehmen?“ (Kap. 08A02, 3, Punkt 1) beschrieben.

Die Hauptaufgaben auch dieser Unternehmen erfordern ein strategisches Portfoliomanagement für die verschiedenen je nach Service Agreements mehr oder weniger langfristig zu unterstützenden kundenspezifischen plattformbasierten SW-Lösungen, zusätzlich aber auch für die fortlaufende Pflege der kundenneutralen Plattform(en) für einzelne Produktlinien.

Abstimmen der Vorgehensweisen

Diese Arten von Vorgehensweisen müssen aufeinander abgestimmt werden:

1. Das Neuentwickeln oder grundlegende Überarbeiten kundenspezifischer plattformbasierter Lösungen, ohne dabei die generelle Plattform zu verkomplizieren oder mit separaten Speziallösungen zu unterlaufen.
2. Das als „Arbeitsfluss mit unbestimmten Ende“ gestaltete Pflegen und Weiterentwickeln dieser kundenspezifischen Lösungen.

3. Das Neuentwickeln oder grundlegende Überarbeiten der kundenneutralen Plattform(en) mit konsequenter Einhaltung und fortlaufender Optimierung einer modularen, serviceorientierten Gesamtarchitektur.
4. Das als „Arbeitsfluss mit unbestimmtem Ende“ gestaltete Pflegen und Weiterentwickeln dieser kundenneutralen Plattform(en).

Mehr noch als beim vorangehenden Vorgehensmuster zur Umstellung auf agile Entwicklung und Unterhalt plattformunabhängiger umfassender Individuallösungen sind hier nicht das Vorgehen und die Methodik in einem kleinen Team die Herausforderung, sondern die Koordination all dieser Spezialisten und Teams auf Programm- und Portfolioebene.

Hoch komplexes Gesamtgebilde ist nicht beherrschbar

Diese vier Arten von Vorgehensweisen zusammen bilden ein hochkomplexes Gebilde. Es kann nicht auf der Basis umfassend geplanter Gesamtkoordinationsprozesse und Verantwortlichkeiten „beherrscht“ werden. Zudem kooperieren in solchen Unternehmen in der Regel weltweit mehrere Hundert bis zu vielen Tausend Informatikspezialisten in Hunderten verschiedener Teams. Die heute auf diesem Gebiet tätigen Unternehmen haben daher gelernt, mit dieser Komplexität einigermaßen erfolgreich umzugehen (sonst gäbe es sie nicht mehr) – oft natürlich mit sehr unkoordiniert bis chaotisch oder bürokratisch-schwerfällig anmutenden Arbeitsweisen.

Das Funktionierende beibehalten und stärken

„Umstellung auf agiles Vorgehen“ bedeutet für diese Unternehmen, all das, was heute bereits situativ und empirisch begründet (statt auf der Basis starrer Prozesse und Verantwortlichkeiten) funktioniert, beizubehalten und zu stärken – und Schwachstellen schrittweise zu eliminieren.

Situativ adaptierbares Zusammenspiel

Erforderlich ist ein agiles (also situativ adaptierbares) Zusammenspiel mehrerer agiler Vorgehensweisen und Steuerungsebenen wie bereits vorher für die Entwicklung und den Unterhalt umfassender Individuallösungen beschrieben, jetzt aber mit der zusätzlichen Berücksichtigung kundenneutraler Pattformen. Das bedeutet:

Portfoliomanagement

- Ein agiles strategisches Portfoliomanagement einerseits für die kundenspezifisch zu entwickelnden und zu wartenden Lösungen und anderseits – zusätzlich – für die Produktlinien der Plattform. Optionen dazu sind auch hier SAFe oder andere Varianten von Portfolio-Kanban.

Programmmanagement

- Ein agiles Programmmanagement wie vorhin beschrieben, hier jedoch zusätzlich die Produktlinien der Plattform umfassend.

ART

- Die Kooperation vieler Teams innerhalb eines „Value Stream“ (= eines „Agile Release Train“) kann auch hier z. B. auf der Basis von SAFe auf der Programmebene koordiniert werden. Hier ist die Koordination auf der Basis von Agile Release Trains jedoch komplexer als vorhin, da diese sowohl die Arbeiten für kundenspezifisch zu entwickelnde und zu wartende Lösungen als auch die Arbeiten zur Weiterentwicklung und Wartung der Produktlinien der Plattform umfassen müssen.

Projekte vermeiden

- Projekte zur Neuentwicklung oder grundlegenden Überarbeitung der Produktlinien der Plattform sollten wenn immer möglich vermieden werden. Sie würden das Gesamtgebilde noch komplexer machen. Diese Arbeiten sollten stets im Rahmen von Agile Release Trains abgewickelt werden.

Kundenprojekte als Konstrukte des Programmmanagements

- Projekte zur Entwicklung individueller Kundenlösungen sollten möglichst nur als Konstrukte auf der Ebene des Programmmanagements existieren, auch wenn sie auf

Kundenseite in der Rege
auftragt werden.

- Die wenigen dann noch v
in geeigneter Art mit de
nagement verknüpft werd

Wenn „Projekte“, dann mit PRINCE2 oder DSDM

- Das agile Management d
oder DSDM möglich. O
(z. B. bei Aufträgen de
Vorgehen basierend auf e
führt zu etlichen Inkompa
adaptiven Vorgehen, aus
sollte dem Kunden entg
Sinn der „agilen Agilitä
Vorgehen so weit als mög

Teams wählen ihr Vorgehen selbst

- Das agile Arbeiten im T
Vergleich zum agilen Pro
parallele Agile Release T
management bezüglich d
den für die (Selbst-)Steue
eines Teams eher einfach
hier mit Kanban (insbes
tungsarbeiten) oder Scru
gearbeitet werden.

Mögliche aufeinanderfolgen
on sind nachfolgend in 2.2.1

2.2.1 Das ist bereits erreic

Wie in der vorherigen Kons
terhalt umfassender Individ
beschrieben. Dabei wird bes
dulare, serviceorientierte A
quent einzuhalten und fortla

folgen alle das Ziel, dass kundenspezifische Lösungen die eigentliche Plattform weder komplizierter machen noch mit separaten Speziallösungen unterlaufen.

2.2.2 Produktlinien/Applikationsfamilien als Value Streams

„Value Streams“ sind – wie ebenfalls in der vorherigen Konstellation beschrieben – auch hier als langlebige Wertetröme zur Erzielung eines wirtschaftlichen Nutzens zu verstehen.

Wertetröme für Plattform und Kundenlösungen

Nun aber gibt es zwei verschiedene Arten von Wertetrömen:

1. Weiterentwicklung und Wartung der Produktlinien der Plattform
2. Entwicklung und langfristige Wartung von plattformbasierten Kundenlösungen

Beide Arten umfassen die Gesamtheit aller Anforderungen, Systemdefinitionen und Entwicklungs- und Auslieferungsschritte, die dazu dienen, ein integriertes System (die Plattform oder eine umfassende Kundenlösung auf ihrer Basis) zu realisieren und zu warten.

„Value Streams“ sind auch hier keine einzelnen – stets zeitlich begrenzten – Projekte, auch nicht im Fall kundenspezifischer Lösungen.

Wertetröme (Value Streams) von Strategie bestimmt

Wie aber können die Value Streams beider Arten gebildet werden? In „Value Streams“ in [1] werden einige essenzielle Fragen als Orientierungshilfe gestellt. Es wird dabei auch klar, dass die Festlegung der Value Streams von der strategischen Ausrichtung des Unternehmens geprägt sein muss und nicht primär eine Frage nur der operativen Optimierung ist.

Zwei Strukturprinzipien bieten sich an:

Nur die Plattform bestimmt Value Streams

1. Die Value Streams repräsentieren die Produktlinien der kundenneutralen Plattform. Angemessen ist das dann, wenn sich die Einmaligkeit des Unternehmens vor allem aus der Einzigartigkeit der Plattform ergibt und nicht auf den spezifischen auf ihrer Basis realisierten kundenspezifischen Lösungen beruht und der Aufwand für die Entwicklung kundenspezifischer Lösungen im Vergleich zum Aufwand für die Plattform klein ist. Kundenlösungen werden dann innerhalb der Value Streams für die Produktlinien der kundenneutralen Plattform entwickelt und betreut.

Value Streams für Plattform und Kundenlösungen

2. Es gibt Value Streams einerseits für die Produktlinien der kundenneutralen Plattform und andererseits solche für charakteristische Arten kundenspezifischer Lösungen (z. B. Branchen, Unternehmensgröße . . .) und möglicherweise Value Streams für je einen sehr großen Kunden mit einer umfassenden spezifischen Lösung. Bei dieser Strukturierung in getrennte Value Streams für die Plattform einerseits und für Kundenlösungen andererseits ist das Risiko allerdings hoch, dass kundenspezifische Lösungen die generelle Plattform komplizierter machen oder mit kundenspezifischen Sonderlösungen unterlaufen.

Value Streams als ARTs implementieren

Beide Strukturprinzipien liefern Value Streams, die im Bereich der Entwicklung und Wartung als Agile Release Trains (ARTs) implementiert werden und Teams mit insgesamt höchstens 150 Personen pro ART umfassen sollten. In einem großen Unternehmen dieser Art mit 6.000 Mitarbeitenden wird es also etwa 50 bis 60 ARTs und ebenso viele Value Streams geben, die es auf der Ebene des Portfoliomanagements abzustimmen gilt.

Trotz der Neustrukturierung in Value Streams kann die bisherige Form des – in der Regel auf der Budgetierungsperiode für ein ganzes Geschäftsjahr und daher nicht sehr flexiblen – Portfoliomanagements eventuell noch beibehalten und erst in einem späteren Schritt flexibilisiert werden. Allerdings:

Wichtig!

Das Portfolio ist grafisch (nicht als „Zahlenfriedhof“) für alle einsehbar. Und zwar so, dass der aktuelle und zukünftige ungefähre Arbeitsumfang pro Value Stream und Quartal erkennbar ist und somit auch angemessen begrenzt werden kann.

2.2.3 Value Streams als Agile Release Trains (ARTs) implementieren

Das für die vorherige Konstellation (Entwicklung und Unterhalt umfassender Individuallösungen) in 2.1.3 Geschriebene trifft auch hier zu, jedoch mit diesen Erweiterungen:

Beim Strukturprinzip „Value Streams einerseits für die Produktlinien der kundenneutralen Plattform und andererseits für charakteristische Arten kundenspezifischer Lösungen“ gibt es zwei Gruppen von Teams: Einerseits die Teams in den Agile Release Trains für die Entwicklung der Plattform und andererseits die Teams in den Agile Release Trains für kundenspezifische Lösungen. Wenn hingegen die Value Streams nur von kundenunabhängigen Produktlinien der Plattform bestimmt werden, gibt es keine zwei Gruppen von Teams bzw. ARTs.

Jeder dieser ARTs umfasst Teams mit insgesamt 50 bis 100 (max. 150) Personen und jeder ART ist ein weitestgehend autonomes soziales Handlungssystem, um das kaum zu leistende zentralistische Management Hunderter Teams und

Spezialisten zugunsten des
rungsfähigkeit sozialer Syste

Abstimmung der Plattform-/ Kunden-lösungs-ARTs

Wie aber kann im Fall „Valu
duktlinien der kundenneutral
charakteristische Arten kun
chergestellt werden, dass die
Produktlinien der kundenne
der kundenspezifischen Lö
dass umgekehrt die ARTs de
die Neuerungen der Plattfor

2.2.4 Plattformgruppen al

Formalisierte Abstimmung via Repräsentanten unrealistisch

Die Abstimmung der Weite
der Plattform mit der Entw
sungen ist mit formalisierter
keiten in Gestalt von Repr
Koordination leisten sollen,
die für eine agile Unternehm
nikationsräume zu schaffen.

Themenzentrierte „Communities“ als Kommunikationsräume

In vielen Unternehmen bewä
in Form themenzentrierter „
struktur. Im vorliegenden Fal
kundenspezifischen Lösung
ARTs der kundenneutralen
weise so: Eine als ART i
Plattform befasst sich mit „
Spezialisten in den ARTs de
die sich dort mit „Fakturier
bilden zusammen mit den „
turierung und Zahlung“ meh
„Fakturierung und Zahlung
dieser Gruppen nicht mehr al

dieser Gruppen trifft sich monatlich zu einem „Führungs-Meeting“, wie im Abschnitt „Anpassbare Kreise statt starrer Hierarchien“ (Kap. 08A03, 4.1) beschrieben wurde. Die Erkenntnisse der einzelnen Plattformgruppen zum Thema „Fakturierung und Zahlung“ fließt in die Planungsmeetings des ART „Fakturierung und Zahlung“ ein. Dort sind ja alle Mitglieder dieses ART in den diversen Plattformgruppen zu diesem Thema vertreten.

Ungewohnt – aber sehr effektiv!

Wenn derartige Kommunikationsräume als „zu teuer“ abgelehnt werden, ist das auch hier – wie bereits vorhin beim Aufwand für die Planungsmeetings pro ART erwähnt (Abschnitt 2.1.9) – ein Zeichen einer noch stark auf „Auftragserteilung von oben“ und „Koordination durch Repräsentanten“ geprägten Kultur. Es hat sich dann noch nicht die Einsicht durchgesetzt, dass ein Unternehmen erst durch die fortlaufende Konversation aller das Unternehmen umfassenden Mitarbeitenden und Stakeholder entsteht.

2.2.5 Überprüfen und Anpassen der Teamstruktur

Hier ist das in der vorherigen Konstellation (Entwicklung und Unterhalt umfassender Individuallösungen) im Abschnitt 2.1.4 dazu Geschriebene ohne Einschränkungen oder Ergänzungen anwendbar.

Das gilt auch für diese nächsten Schritte:

2.2.6 „WAS“ vom „WIE“ trennen: pro Team ein Team Backlog Owner

Siehe Abschnitt 2.1.5

2.2.7 Die Teamleiter bleiben Rahmensetzer für das „WIE“

Siehe Abschnitt 2.1.6

2.2.8 Pro „ART“ die Zeitpunkte der externen und internen Releases bestimmen

Siehe Abschnitt 2.1.7

2.2.9 Programmmanagement agil gestalten

Siehe Abschnitt 2.1.8

2.2.10 Planung des kommenden „PSI“ (interner Release) eines „ART“

Siehe Abschnitt 2.1.9

2.2.11 Agile Arbeit auf Teamebene

Siehe Abschnitt 2.1.10

2.2.12 Projekte innerhalb eines „ART“

Siehe Abschnitt 2.1.11

2.2.13 Agiles Portfoliomanagement

Siehe Abschnitt 2.1.12

2.3 Vorgehensmuster zur Umstellung auf agile SW-Entwicklung als Dienstleistung

Die Charakteristiken und Herausforderungen von Firmen im Bereich der SW-Entwicklung als Dienstleistung wurden im Abschnitt 3 „Wie reisen Sie mit Ihrem Unternehmen?“ im ersten Teil dieser Reihe beschrieben (Kap. 08A02, 3).

Isolierte, ganz unterschiedliche Kundenprojekte

Die Hauptaufgaben dieser Dienstleister ergeben sich aus dem Anbieten von SW-Entwicklungskapazität (Spezialisten samt Entwicklungsinfrastruktur) im Rahmen isolierter und oft ganz unterschiedlicher Kundenprojekte bis zur Inbetriebnahme der Lösung. Wartungs-, Anpassungs- und Erweiterungsarbeiten für diese Lösungen werden nur gelegentlich und speziell beauftragt.

Strategisches Portfoliomanagement kaum machbar

In der Regel können diese Aufträge nicht längerfristig, etliche Monate im Voraus, abgeschätzt werden. Ein strategisches Portfoliomanagement erübrigt sich daher oder reduziert sich auf die Festlegung jener Branchen, Kundenarten, Technologien und Entwicklungsinfrastruktur, auf deren Basis das Unternehmen mit seinen Spezialisten heute und in Zukunft tätig sein kann und will.

Vorgehensweise nur bedingt selbst wählbar

Nur wenn die Entwicklung von Individualsoftware im Rahmen eines Werkvertrags mit dem zu erbringenden Funktionsumfang beauftragt wird, kann der Dienstleister das Vorgehensframework und die Entwicklungsmethodiken frei wählen. In aller Regel werden solche Aufträge dann als isolierte Projekte – oft im Haus des Dienstleisters – abgewickelt.

Ein bis wenige Teams pro Projekt

Diese Projekte erfordern beim Auftragnehmer in der Regel den Einsatz nur eines für dieses Projekt arbeitenden Teams, selten einer Handvoll Teams, manchmal auch nur einzelner

Personen. Ein ausgefeiltes I
nicht erforderlich.

Gesamtprojekt oft vom Kunden gemanagt

Oft aber werden nur SW-En
von größeren und vom Aufti
jekten zur Verfügung geste
dann das Vorgehen und Tei
muss sich daher in verschie
und Methodiken professione

Craftsmanship als Wettbewerbsfaktor

In jedem Fall kann (und sol
fessionelles Handwerk (Cra
Das steht kaum im Widersp
gebenen Frameworks. Diese
Beherrschen irgend eines Vo
Wettbewerbsfähigkeit des Di

Die Vorgehensweisen dieser
- vom projektbasierten Arb
eller Kundenlösungen mi
rären Projektteam. Das a
ist mit einer auf das jew
schnittenen flexiblen un
PRINCE2 oder DSDM m
- von der Fähigkeit, bei der
des Auftraggebers desse
anwenden zu können;
- von Methoden und Werkz
mittelfristigen Einsatzpl
möglichst einfach gestalt
selbst (und nicht von ein
pflegt werden können;
- von all dem, was zur „Cra

Mögliche aufeinanderfolgende Schritte für diese Konstellation sind nachfolgend in 2.3.1 bis 2.3.8 beschrieben:

2.3.1 Das ist bereits erreicht

Wesentliche praktizierte Elemente

Wesentliche Elemente der „agilen Kultur“ werden praktiziert, insbesondere:

- Daily Standup (mit den nicht gerade bei Kunden arbeitenden Personen) und etwa zweimal pro Woche mit allen Personen unter Nutzung virtueller Kanäle (wegen der in den Häusern diverser Kunden eingesetzten Personen)
- Transparente Informationen und Sichtbarkeit, Verfolgbarkeit, Analysierbarkeit der Zusammenarbeit unter Nutzung virtueller Kanäle
- Arbeitsplanung im Projektteam durch das Team – nicht durch den Projektleiter
- „Agile“ Art der Führung mit all jenen Einschränkungen, die sich aus der matrixartigen Organisation ergeben (die Spezialisten arbeiten in mittelfristig immer wieder wechselnden Teams des Unternehmens oder der Kunden)

Craftsmanship ist etabliert

Craftsmanship (SW-Entwicklung als professionelles Handwerk) ist etabliert, insbesondere:

- Nicht nur funktionierende Software, sondern gut gefertigte Software ist das Ziel.
- Clean Code Development prägt die Arbeitsweise.
- TDD und ATDD und die Integration von Entwicklung und Test, Continuous Integration, Continuous Delivery und Testautomation werden fortlaufend verbessert, soweit dies in den Entwicklungsumgebungen der Kunden möglich ist.

2.3.2 Überprüfen und Anpassen der Teamstruktur

Die in temporären Projektteams des Dienstleisters oder der Kunden arbeitenden Mitarbeitenden nach Möglichkeit so einsetzen, dass:

Temporäre Projektteams

- jedes temporäre Projektteam beim Dienstleister über permanente, stets voll ausgelastete und daher „ungeteilte“ Mitglieder/Spezialisten verfügt, die insgesamt zum Definieren/Realisieren/Testen der Softwareprodukte oder -services nötig sind;

Kein Projekt-Multitasking

- jeder Mitarbeitende des Dienstleisters immer nur in einem Team des Dienstleisters bzw. des Kunden eingesetzt ist (kein „Multitasking“).

2.3.3 Bei (temporären) Projektteams des Dienstleisters das „WAS“ vom „WIE“ trennen

Projektleiter = „WAS“-Verantwortlicher = Team Backlog Owner

Üblicherweise sind die Projektleiter Rahmensetzer sowohl dafür, „WAS“ das Projektteam zu leisten hat, als auch für das „WIE“. Der Vorteil ist, dass es pro Projektteam aus Kundensicht einen einzigen Ansprechpartner sowohl für das „WAS“ als auch für das „WIE“ gibt. Der Nachteil ist, dass das funktionale „WAS“ allzu sehr vom machbaren „WIE“ (Kompetenzen im Team, Auslastung einzelner Spezialisten, Entwicklungsinfrastruktur) bestimmt wird und umgekehrt der Projektleiter das „WIE“ zu sehr einschränkt. Deshalb ist es essenziell, die Rahmensetzung für das „WAS“ und das „WIE“ auf Rollen aufzuteilen, die personell getrennt sind. Damit diese klare Trennung gelingt, sollte die Verantwortung nur für das „WAS“ (nicht das „WIE“) der gegenüber dem Kunden in Erscheinung tretende Projektleiter übernehmen. Mit dem „WAS“ beschreibt der Projektleiter all das, „was“ das Team derzeit und künftig zu liefern hat: User Stories, zukünftige Features, technische Stories, Fehlerbehebungen, Infrastruk-

turarbeiten, Spikes, Refactorings usw. Auch wenn einzelne dieser Dinge im Team erarbeitet werden, bleibt er als Projektleiter allein verantwortlich für diese Liste und alles, was darin enthalten ist, und für die Reihenfolge der Einträge. Er ist als einzige Person für die Verwaltung des Team Backlog verantwortlich.

Der Projektleiter als Team Backlog Owner entlastet das Team vom Klären widersprüchlicher und vom Priorisieren vieler gleichzeitig vorliegender Anforderungen diverser Anspruchsgruppen. Das Team verhandelt das „WAS“ seiner Arbeit ausschließlich mit seinem Projektleiter als Team Backlog Owner, einschließlich der nichtfunktionalen Anforderungen. Das Team ist für den realisierten Lieferumfang nur gegenüber seinem Projektleiter als Team Backlog Owner und nicht diversen anderen Anspruchsgruppen direkt verantwortlich.

2.3.4 Rahmensetzer und Unterstützer für das „WIE“ bei temporären Projektteams des Dienstleisters

„WIE“-Verantwortlicher = Agile Master

Neben dem Projektleiter als Team Backlog Owner hat jedes Projektteam einen „Wie-Verantwortlichen“ im Sinn eines Scrum Master oder Agile Master. Im Unterschied zu den für längere Zeit existierenden Teams der vorherigen zwei Konstellationen sind sie keine Teamleiter. Im Einvernehmen mit der – hier aber von der tagtäglichen Arbeit „ihrer“ Mitarbeitenden entkoppelten – Führungsperson kümmert sich dieser Agile Master auch um die individuelle Förderung der Teammitglieder auf der Basis der von ihm Tag für Tag miterlebten Arbeit.

(Bei in Teams des Kunden entsandten Mitarbeitenden des Dienstleisters ist auch das nicht gegeben. Dort muss sich die von der tagtäglichen Arbeit „ihrer“ Mitarbeitenden entkop-

pelte Führungsperson bei ih
tionen des Kunden stützen.)

2.3.5 Agiles Vorgehen bei abgewickelten Proje

Inwieweit Projekte „agil“ a
umstritten (s. Kap. 08S01, 3

Wenn der Kunde mit dem B
reits zu Beginn klare und d
zeitliche und finanzielle Re
verstanden ist, dass die Ziele
im Projektverlauf im Rahm
Absprache mit ihm ohne ei
Management verändert werd
adaptives Vorgehen möglic
rungsphase wird dann ein
„Features“ (nicht bereits aus
ten Grundlagen der Projekts

Das agile Projektvorgehen in
– auf das unbedingt nötige M
PRINCE2 oder DSDM erfol

Die vertragliche Regelung so
[5] beruhen.

2.3.6 Pro im Haus des Die Projekt die Zeitpunk Releases bestimmen

Ideal alle zwei bis vier Wochen

Idealerweise sollte beim agi
Wochen, beim flussorientier
etwas fertig ist, ein auslief

zeugt werden. Das setzt jedoch voraus, dass das möglicherweise recht heterogene System des Kunden ein in so kurzen Abständen in seinem System getestetes und zuverlässig lauffähiges Inkrement auszuliefern erlaubt und die Nutzer beim Kunden gewillt sind, regelmäßig alle zwei bis vier Wochen oder noch häufiger neue und geänderte Softwarefunktionen zu erhalten und zu erlernen und ihre Arbeitsabläufe darauf abzustimmen.

Projektleiter klärt

Vom Projektleiter zu klären ist daher:

- Wie groß also ist die Volatilität der Anforderungsänderungen? Wieviel Prozent der Anforderungen ändern sich pro Woche oder Monat oder Quartal?
- Wie häufig wünschen die Nutzer neue Releases?

Die Antwort bestimmt die Häufigkeit der externen, also effektiv ausgelieferten und produktiv gesetzten Releases.

Wenn für das Projekt die Zeitpunkte der (nicht unbedingt regelmäßigen) externen Releases mittelfristig abgeschätzt sind, stellt sich folgende weitere zusammen mit allen Teams des Projekts zu beantwortende Frage:

- Wie groß ist die optimale Dauer der internen (für die eigentlichen Nutzer unsichtbaren) Releases?

Deren Takt ist regelmäßig und nie länger als jene minimale Dauer, in der alle zum internen Release beitragenden Teams gemeinsam ein „Potentially Shippable Increment“ (PSI) so erzeugen können, dass die Beiträge aller Teams in eine betriebsnahe Integrationsumgebung erfolgreich überführt und dort sowohl technisch (inklusive aller nichtfunktionalen Anforderungen) als auch fachlich erfolgreich getestet werden können.

2.3.7 Planung des kommenden „PSI“ eines im Haus des Dienstleisters abgewickelten Projekts

Hier ist das in Abschnitt 2.1.9 dazu Geschriebene sinngemäß anwendbar, wobei „Agile Release Train“ durch „Projekt“ ersetzt ist und das „Programmmanagement“ durch den „Projektleiter“.

2.3.8 Agile Arbeit in den Teams der im Haus des Dienstleisters abgewickelten Projekte

Inwieweit einzelne Teams innerhalb der Realisierung eines PSI in Anlehnung an Scrum in Sprints oder mit Kanban oder Scrumban oder nach eigenem Muster arbeiten, kann auch hier den Teams überlassen werden. Für die Teams ist mit der Release-Planung pro Projekt der Rahmen für das „WAS“ gesetzt. Der Agile Master hat die Verpflichtung, den Rahmen für das „WIE“ zu setzen, ohne dabei das „WAS“ zu erschweren.

Alles Weitere entspricht dem für die vorherigen zwei Konstellationen zu „Agile Arbeit auf Teamebene“ bereits Geschriebenen.

3 Rückblick auf alle drei Teile der Reihe „Wie Sie auf agile SW-Entwicklung umstellen“

Konzepte

Ausgehend vom Beitrag „Agile Konzepte“ (Kap. 08A01), wurde in drei weiteren aufeinander aufbauenden Beiträgen die Umstellung auf eine agile Form der SW-Entwicklung beschrieben.

Veränderung der Unternehmenskultur

Der erste Beitrag (Kap. 08A02) zeigte, dass diese Umstellung Veränderungen der Unternehmenskultur voraussetzt. Das sind langfristige und strategisch relevante Investitionen. Erst

auf dieser Basis können die jeweils angemessenen und vergleichsweise kurzlebigen Methoden und Techniken der agilen (aber auch „traditionellen“) SW-Entwicklung sinnvoll genutzt werden. Der erste Beitrag gab einen zusammenfassenden Überblick über einige Methoden und Techniken zur Kulturveränderung und schlug danach einen agilen Umgang mit den Methoden und Techniken der agilen SW-Entwicklung vor, wobei die „Craftsmanship“ und das „Clean Code Development“ unabhängig von der Vorgehensweise zentrale Bedeutung haben.

Methoden und Techniken zur Kulturveränderung

Der zweite Beitrag (Kap. 08A03) beschrieb im Detail einige Methoden und Techniken für diese Kulturveränderung im Bereich der Kommunikation, Kooperation und Führung, und zwar:

- Etablierte Konversationsräume
- Daily Standup
- Erfahrungs- und Praxisgemeinschaften
- Transparente Informationen
- Unternehmensinterne Kommunikation mittels Social Media
- Sichtbarkeit, Verfolgbarkeit, Analysierbarkeit der Zusammenarbeit
- Arbeitsplanung im Team durch das Team – nicht durch den Teamleiter
- Kontinuierliche Verbesserung der Kooperation im Team
- Teamarbeit mit Elementen von Scrum
- Optimierung der Teamstruktur
- Anpassbare Kreise (statt starrer Hierarchien) mit Führungsmeetings, wöchentlichen taktischen Meetings und Daily Standup Meetings

- Change Events: Nur den
lassen
- „Agile“ Art der Führung
- Führen von Teams
- Teamleiter – neu definie

Kontextabhängige Vorgehensmuster

Im dritten Beitrag (Kap. 08
zur Umstellung auf agile E
Situationen beschrieben:

- Entwicklung und Unter
sungen
- Entwicklung und Unterl
sungen
- SW-Entwicklung als Die

Diese drei Vorgehensmuster
folio- und Programmmanag
folgen dem agilen (und nicht
„traditioneller“ Methoden u

Das Vorgehensmuster für di
lung und Unterhalt plattfor
fassendste, bestehend aus di

- Voraussetzungen
- Produktlinien/Applikatio
- Value Streams als Agile
mentieren
- Plattformgruppen als Se
- Überprüfen und anpasse
- Das Was vom Wie trenn
Owner

- Die Teamleiter bleiben Rahmensetzer für das Wie.
- Pro ART die Zeitpunkte der externen und internen Releases bestimmen
- Programmmanagement agil gestalten
- Planung des kommenden PSI (interner Release) eines ART
- Agile Arbeit auf Teamebene
- Projekte innerhalb eines ART

Diskussionsforum zum Thema „agile SW-Entwicklung“

XING-Forum zur Beitragsserie

Dieser letzte und alle vorangehenden Beiträge der Serie zum Thema „agile SW-Entwicklung“ bieten viel Diskussionsstoff. Deshalb hat der Autor dazu in Xing ein Diskussionsforum eingerichtet:

tinyurl.com/py8wgke

Ihre Beiträge und Fragen sind willkommen!

Quellen

[1] **scaledagileframework.com/glossary**/

[2] Oberrath, Reick; Vollmer, Jörg: Clean Coding Cosmos. In: OBJEKTspektrum. 11/12.2013

[3] **scaledagileframework.com/**↩
agile-software-requirements-model/

[4] Korn, Hans-Peter: Agile Führung: Ein Oxymoron? In: OBJEKTspektrum, Ausgabe 05/2014

[5] Opelt, Andreas: Der agile Festpreis: Leitfaden für wirklich erfolgreiche IT-Projekt-Verträge. Carl Hanser, 2012

[6] Standish Group: CHAOS MANIFESTO 2013.
versionone.com/assets/img/files/↩
ChaosManifesto2013.pdf