Agilität und planbare Projektbudgets

Widerspruch oder eine Frage der Herangehensweise?

UNSER KONTAKT

queo GmbH

Tharandter Str. 13
01159 Dresden

info@queo-group.com
Tel. 0351 213038-0
Fax 0351 213038-99

www.queo.com

facebook.com/queogroup
twitter.com/queo_group

1 Klassik und Agilität

Wer sich eingehender mit dem IT-Projektmanagement bzw. Softwareentwicklungsmodellen beschäftigt, wird zwangsläufig mit agilen Modellen konfrontiert. Verschiedene Studien[1] zeigen, dass sich agile Vorgehensweisen verstärkt durchsetzen. Zu dieser Entwicklung hat sicher auch die Tatsache beigetragen, dass klassische Ansätze wie das Wasserfallmodell im Verlauf der Zeit immer komplexer und bürokratischer wurden. Durch die starren Vorgaben kann ein solches Modell nicht flexibel auf Veränderungen im magischen Dreieck (Kosten – Qualität – Zeit) angepasst werden, ohne die Kontrolle über die Gesamtbetriebskosten zu verlieren.

Eine Herausforderung bringen agile Projekte jedoch mit sich: Durch die Bestimmung eines fixen Budgets zu einem variablen Leistungsumfang steht nicht selten ein Widerspruch zu Interessen und Zielvorgaben der Projektbeteiligten fest.

Gerade in größeren Konzernstrukturen trifft man häufig auf eine konservative Aufteilung von Kompetenzen im Softwarebeschaffungs- und Entwicklungsprozess. Der Fachbereich legt normalerweise den Umfang fest und stellt dafür einen Teil seines IT-Budgets bereit. Im Einkauf wird das Ziel verfolgt, die vom Fachbereich vorgegebenen Umfänge mit einem möglichst geringen Budget einzukaufen. Über allem steht das Management, für das der Fortschritt des Projekts entlang eines Plans transparent nachvollziehbar sein muss.

Aus unserer Erfahrung ist es in solchen Konstellationen aufgrund der unterschiedlichen Interessenlagen sehr schwierig, ein rein agiles Projekt zu platzieren. Der Einkauf sieht beispielsweise die Gefahr der Verwässerung seines Verhandlungsergebnisses durch eine Kürzung der umgesetzten Funktionen durch den Dienstleister. Zusätzlich wird es durch die stetigen Änderungen an dem zu realisierenden Funktionsumfang für das Management schwieriger, Aussagen über den Leistungsfortschritt zu treffen.

In diesem Whitepaper wird gezeigt, dass sich agile Projektvorgehen und klassische Projektmanagementverfahren nicht zwangsläufig widersprechen, sondern – korrekt angewendet und geschickt kombiniert – auch Synergiepotentiale haben können. Dazu werden die Begriffe „inneres und äußeres Projektmanagement“ eingeführt, um auf dieser Basis das mehrfach bewährte Vorgehen aus unserer Praxis als Anbieter individueller Softwarelösungen zu veranschaulichen. Zusätzlich nehmen wir im folgenden Kapitel eine historische Einordnung der iterativen Ansätze im Vergleich zu den vermeintlich klassischen Ansätzen vor, die überraschende Erkenntnisse mit sich bringt.

1 vgl. http://www.heise.de/developer/meldung/Studie-Agile-Softwareentwicklung-ist-Mainstream-912207.html und http://www.heise.de/developer/meldung/Studie-Agile-Methoden-im-Hoehenflug-2442650.html

2 Am Puls der Zeit arbeiten: Agile Softwareentwicklung

2.1 Das Wasserfallmodell, ein historischer Irrtum?

Wenn heute von klassischen Entwicklungsansätzen gesprochen wird, dann ist häufig das Wasserfallmodell gemeint, das auch immer noch eine reine Form des traditionellen, plangesteuerten Entwicklungsvorgehens im Gegensatz zu iterativen Vorgehensweisen darstellt.[2]

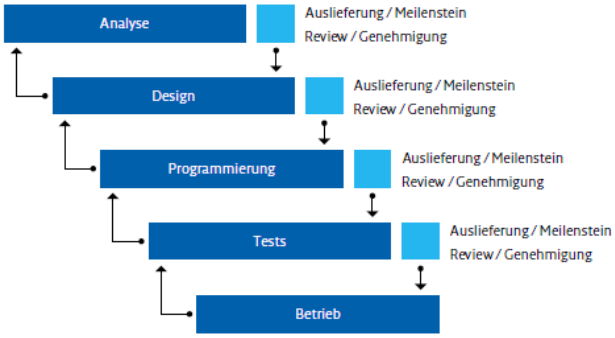

Abbildung 1: Wasserfallmodell[3]

Doch welcher Ansatz nun der klassische ist, bleibt bei genauerer Betrachtung der Entwicklungsgeschichte zweifelhaft, reichen doch die Wurzeln iterativer Projektmanagementverfahren so weit wie jene sequenzieller Prinzipien. Winston Royce gilt im Zusammenhang mit dem Wasserfallmodell als dessen Begründer, publizierte aber bereits 1970 ein erstes iteratives Modell, das für den US-Militär-Standard STD-2167 das geeignete Fundament bot. David Maibor, Mitautor dieses Militär-Standards, berief sich auf eine vereinfachte Variante des iterativen Modells und legte damit den Grundstein für das heute als Wasserfallmodell bezeichnete Vorgehen. Interessant ist hier ebenso, dass Royce den Begriff „Wasserfall" in seiner Veröffentlichung gar nicht verwendet hat, sondern lediglich viele Visualisierungen mit kaskadenartigen Phasen benutzte. Der Sohn von Royce behauptete später, dass sein Vater immer ein Vertreter iterativer, inkrementeller und evolutionärer Entwicklungsmodelle gewesen sei und, dass das aus seiner Veröffentlichung abgeleitete Wasserfallmodell nur die einfachste Variante als Empfehlung für sehr unkomplizierte Projekte gewesen wäre. Diesen Gedanken greift Maibor später auch auf und stellt klar, dass er iterative Vorgehen imSTD-2167 viel deutlicher empfohlen hätte, wenn er damit vertrauter gewesen wäre.[4]

So gesehen handelt es sich beim Wasserfallmodell also tatsächlich um einen historischen Irrtum, der sich in den 1980er und 1990er Jahren schnell verbreitete und zum Praxisstandard wurde. Betrachtet man die problematischen Effekte, die bei unreflektierter Anwendung des Wasserfallmodells auf große Projekte auftreten können, so ist dies schon eine kuriose Anekdote der Informatik.[5]

Parallel zur Verbreitung des Wasserfallmodells hatten es iterative Ansätze schwer, sich zu behaupten, auch wenn es beispielsweise durch IBM bei der Entwicklung der Software für die NASA Space-Shuttle beachtliche Erfolge gab.[6]

---

2 vgl. Rubin, Keneth S.: Essential Scrum – Umfassendes Scrum-Wissen aus der Praxis, MITP, 2014 – S. 64
3 in Anlehnung an:. Royce, W.: Managing the development of large software sytems, in: Proceeding of IEEE WESCON, 1970
4 vgl. Larman, C.: The historical accident of waterfal validity, in: Agile and iterative development, 2004
5 vgl. Oestereich, B.; Weiss, Chr.: APM - Agiles Projektmanagement - Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008 – S. 13 und Larman, C.: The historical accident of waterfal validity, in: Agile and iterative development, 2004
6 vgl. Oestereich, B.; Weiss, Chr.: APM - Agiles Projektmanagement - Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008 – S. 13

2.2 Agile Softwareentwicklung und agiles Projektmanagement

Ende der 1990er Jahre entstanden unter dem Schlagwort „Agil“ quasi als Gegenbewegung zu den existierenden überreglementierten und starren Ansätzen verschiedene iterative Vorgehen, von denen im deutschen Sprachraum „Extreme Programming“ (XP) [7] und „SCRUM“[8] heute immer noch zu den verbreitetsten Vertretern gehören.

Eine wichtige Basis für ihre Verbreitung war unter anderem auch die Veröffentlichung des agilen Manifestes im Jahr 2001 (siehe http://www.agilemanifesto.org), aus dem dann auch die späteren Prinzipien agiler Softwareentwicklung abgeleitet wurden.

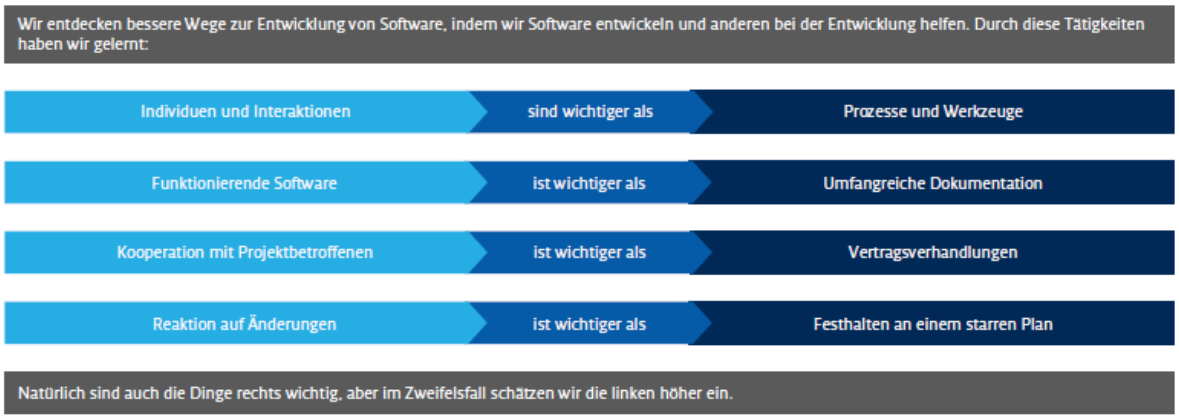

Abbildung 2: Wortlaut des agilen Manifestes[9]

Bis heute gibt es kein agiles Standardverfahren. Die meisten existierenden Ansätze haben über die Zeit ihre Berechtigung gefunden, wobei sich viele Elemente, Techniken und Werte der Verfahren ähneln. Darüber hinaus werden oft Abwandlungen der etablierten, sehr formalisierten agilen Ansätze eingesetzt, weil sie dadurch besser in die Organisation der Unternehmen passen.

Im Kundengespräch fallen beim Thema agiles Projektmanagement meist sehr schnell Begriffe wie Sprint oder Product Backlog, die der eigenen Terminologie von SCRUM entnommen sind. Dadurch entsteht oft der Eindruck, dass die SCRUM-Methodik und die agile Softwareentwicklung synonyme Begriffe sind. Das ist nicht der Fall, aber wie aktuelle Studien zeigen, ist SCRUM die verbreitetste Methode des agilen Projektmanagements[10] und so verwundert es nicht, dass diese Annahme ebenso weit verbreitet ist. Da sich unser Vorgehen ebenso der SCRUM-Methodik bedient, sei an dieser Stelle auf einen **Artikel auf der queo Website verwiesen**, der die Terminologie mit Hilfe eines Schaubilds erläutert.

7 vgl. Beck, K.: Extreme Programming, Das Manifest, 2. Auflage, Addison Wesley, 2001
8 vgl. Beedle, M.; Schwaber, K.: Agile Software Development with Scrum, Prentice Hall, 2002
9 Übersetzung entnommen aus: Oestereich, B.; Weiss, Chr.: APM - Agiles Projektmanagement - Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008 – S. 15
10 vgl. http://www.heise.de/developer/meldung/Studie-Agile-Softwareentwicklung-ist-Mainstream-912207.html

3 Das Fixe und das Variable: Agiles erweitert Klassisches

Unabhängig davon, ob der Kunde klassisch oder agil vorgehen möchte, für Anbieter individueller Softwarelösungen ist es gelebte Praxis: Das mit einem Projekt verfolgte Ziel liegt in einer Wolke und ist kaum sichtbar. Alle Projektbeteiligten wissen zwar grob, wie das Ergebnis aussehen soll, aber um das Ziel zu beschreiben, werden ausgehend von den bisherigen Erfahrungen aller Beteiligten Annahmen getroffen, die wenig später schon überholt sein können oder nur zum Teil zutreffen werden.

Selbst, wenn im Sinne eines klassischen Vorgehens diese Annahmen in einem Lastenheft präzise beschrieben sind, so wird dadurch nur ein Teil der Unschärfe aus dem Projekt genommen. Im Projektverlauf werden erfahrungsgemäß immer Fragestellungen auftauchen, die bisher nicht oder nur unzureichend betrachtet wurden.

Um solche Fragestellungen innerhalb des laufenden Projektprozesses zu vermeiden, sind Entwickler bereits zu Beginn bemüht, Ziele möglichst eindeutig und widerspruchsfrei zu beschreiben. Diese Maxime erscheint nachvollziehbar, sie kann aber im wirtschaftlichen Kontext einen hohen konzeptionellen Aufwand bedeuten, der den Projektstart verzögert. Des Weiteren kann die Antwort auf eine Frage vom Beginn divergieren von jener zum Projektende. Schnell wird offensichtlich: Ist man bereit zu akzeptieren, dass das Ziel in der Regel nicht glasklar ist, kann dieser Umstand einen schnelleren Projektfortschritt begünstigen.

Jener Gedanke ist die Basis aller iterativen Vorgehensweisen, bei denen das Gesamtsystem nicht mit einem Mal entsteht, sondern schrittweise in sogenannten Timeboxes[11] erstellt wird. Das Ziel ist dabei keine Konstante, sondern verändert sich mit der Zeit, wie in Abbildung 3 dargestellt.

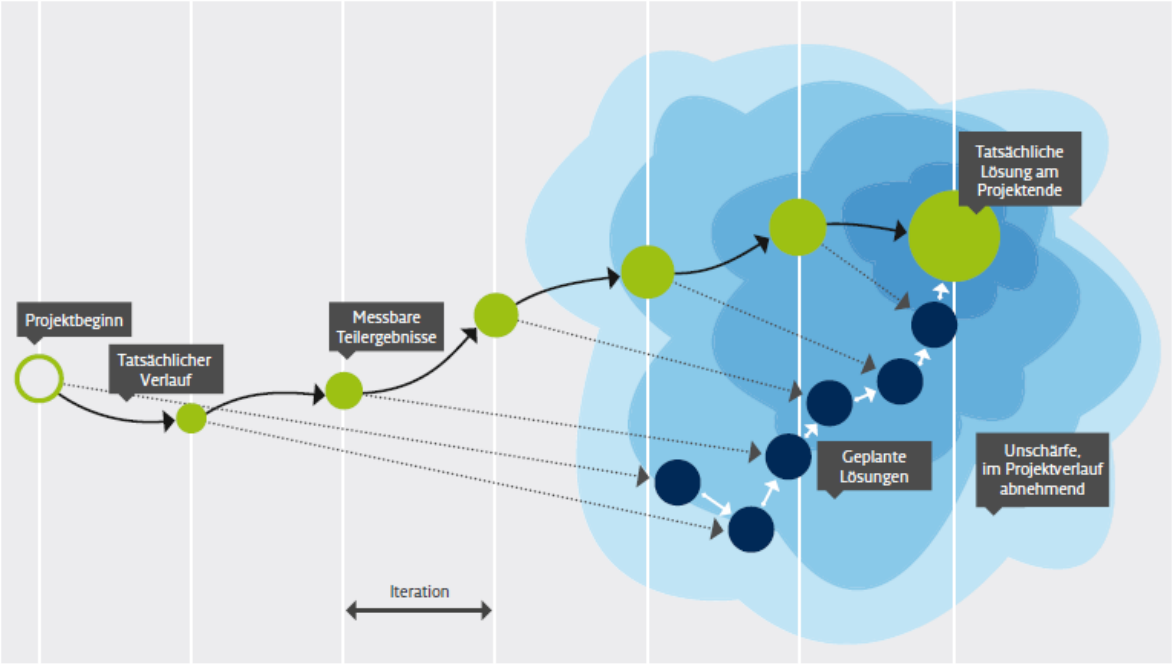

Abbildung 3: Iterations-Wolken-Metapher[12]

11 vgl. Oestereich, B.; Weiss, Chr.: APM - Agiles Projektmanagement - Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008 – S. 5
12 in Anlehnung an: Oestereich, B.; Weiss, Chr.: APM - Agiles Projektmanagement - Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008 – S. 3

Die Einteilung in Iterationen sind den einzelnen Stakeholdern in der Fachabteilung des Kunden, über die IT bis hin zur Beschaffung gut zu vermitteln, da sie dadurch frühzeitig auf die Ergebnisse im Sinne des Projekterfolgs einwirken können. Dass sich mit der Bekenntnis zur iterativen Projektgestaltung auch ursprüngliche Zielstellungen verändern können, da diese während des Prozesses als variabler Bestandteil akzeptiert werden, erscheint folgerichtig, für den Kunden aber nicht mehr akzeptabel. Aus der Kundensicht verliert man so doch tradierte Formen fixer Leistungskriterien, die niedergeschrieben werden können und idealerweise eine Beauftragung zum Festpreis ermöglichen. Diesem Druck, ein Ziel so konkret wie möglich formulieren zu können, begegnen Fachabteilungen meist mit der Ausarbeitung eines präziseren Konzepts, das sich im Tagesgeschäft aber nur schwer integrieren lässt. Die Realisierung eines solchen Pflichtenhefts wird daher oft einem Dienstleister übertragen.

So wird nicht nur der erste Schritt auf dem klassischen Weg der Softwareentwicklung beschritten, ferner wird auch die bestehende Ausgangssituation manifestiert, denn: Wie soll ein Dienstleister die Fragen allein klären, denen die Fachabteilung aus Zeitgründen ausweicht? Ein Dienstleister kann so auch bei engstem Kundenkontakt nur Annahmen treffen, die im Zweifel erst ganz am Ende durch die Anwender im Rahmen der Produkttests hinterfragt werden. Wichtige Entscheidungen basieren somit auf einer riskanten Stochastik aus dem Erfahrungsschatz des Dienstleisters. Das so entstandene Pflichtenheft und die darauf basierende technische Konzeption mit Aufwandsschätzung können zwar das Erreichen eines bestimmten Ziels zu einem fixen Budget garantieren, doch wo bleibt in diesem Gefüge der Aspekt kundenseitiger Zufriedenheit? Es darf deshalb, etwas spitz formuliert, durchaus hinterfragt werden, welches der folgenden Ideale im Ergebnis wichtiger erscheint:

Ein garantiertes unklares Ziel zu einem fixen Budget oder ein akzeptiertes klares Zwischenziel zum gleichen Budget?

Im Glauben an die Überlegenheit iterativer Vorgehensweisen und der agilen Softwareentwicklung könnte ein Dienstleister an dieser Stelle aber das Risiko eingehen und in der Umsetzung über Iterationen agil arbeiten. Die Begeisterung der Anwender mag ihm auch sicher sein, aber spätestens bei der Abnahme wird das Ergebnis auch anhand des Pflichtenheftes und des Vertrages bewertet. Sind dann einzelne Module durch das agile Vorgehen aus dem vermeintlichen Basisumfang herausgefallen, wird das zwangsläufig zu Diskussionen mit dem Auftraggeber führen. Da dies aber eine der Grundannahmen agiler Vorgehensweisen ist, wird auch dieser eher versteckte agile Entwicklungsansatz scheitern, wenn sich nicht alle Stakeholder von Anfang an darauf einigen und sich im Projektverlauf gegenseitig vertrauen. Um das zu verhindern, sollten sich alle Beteiligten zum Projektbeginn über die Antworten auf die folgenden Fragestellungen bewusst werden:

- Kann ein klares Ziel formuliert und abgegrenzt werden?
- Wie groß ist das Minimal-Set an Anforderungen / Funktionen für eine Basisanwendung?
- Welches Budget steht zur Verfügung und was sind die Minimalanforderungen wert?
- Warum muss bei einem unscharfen Ziel und fließenden Anforderungen ein Budget garantiert werden?
- Kann eine mangelnde Akzeptanz der Software aufgrund ihrer unzureichenden Nutzbarkeit vor dem Management vertreten werden?

Hat man sich auf Antworten zu diesen Fragen verständigt, ist das Problem noch nicht gelöst, aber man hat auf allen Seiten ein Bewusstsein für die Problemstellung und eine Basis geschaffen, um agil im Sinne der Zufriedenheit und Akzeptanz der Software in den Grenzen fixer Budgets vorzugehen.

Der im folgenden Kapitel dargestellte Lösungsansatz greift die beschriebenen Problemstellungen auf und formuliert Antworten auf Basis eines etablierten Vorgehens, dass sich an SCRUM orientiert und sich bei queo, einem Anbieter individueller Softwarelösungen, bewährt hat.

4 Über Division zur Fusion: Die zwei Seiten des Projektmanagements

4.1 Trennung in inneres und äußeres Projektmanagement

Um den Zwiespalt zwischen klassischen und agilen Ansätzen nicht nur auf Basis der historischen Irrtümer aufzulösen, sondern unseren Kunden auch Lösungen innerhalb klassischer Denkmuster anzubieten, haben wir gute Erfahrungen damit gemacht, dass wir unseren großen Softwareentwicklungsprojekten eine klassische Hülle verpassen. In dieser Hülle, die wir auch als das „äußere Projektmanagement" bezeichnen, bilden wir alle formalen und weniger inhaltlichen Aufgaben entlang des Entwicklungsprozesses einer individuellen Softwarelösung ab, die für uns als Projektdienstleister aus Sicht des Multiprojektmanagements essentiell sind und darüber hinaus auch für die Controlling-Mechanismen unserer größeren Kunden benötigt werden. Neben der Definition der vertraglichen Rahmenbedingungen und eines fixen Budgets werden im Projektverlauf auch die verbrauchten zu den verbliebenen Aufwänden dokumentiert oder der Finanzstatus und Gesamtprojektfortschritt vor entsprechenden Gremien vertreten. Hinzu kommen weitere allgemeine, kosmetische Aufgaben, wie z. B. das Einrichten von Buchungsmöglichkeiten für die Arbeitszeiterfassung oder die Pflege und Dokumentation der einzelnen Planungsstände in einem entsprechenden Projektmanagementtool, wie bspw. MS Project oder dem von uns entwickelten und verwendeten **Tool Barbuda.**[13]

Das „innere Projektmanagement" beschreibt in unserem Lösungsansatz im Vergleich zum „äußeren Projektmanagement" die konkreten Aufgaben zur Umsetzung der Software durch das Team in enger Zusammenarbeit mit der Fachabteilung des Kunden auf Basis agiler Vorgehensweisen in Anlehnung an SCRUM. Durch die direkte Kommunikation zwischen Fachexperten und Entwicklern rückt der agile Ansatz im Innern auch in das Bewusstsein des Kunden und durch den Fokus auf die schnellstmögliche Erstellung eines ausführbaren, aber ggf. nicht voll funktionsfähigen Product Increment, kann er zeitnah Feedback zu den umgesetzten Funktionen geben. Diese gemeinsame Interaktion und das gegenseitige Vertrauen sind dabei die wichtigsten Erfolgsfaktoren für das Gelingen agiler Projekte.

SCRUM beschreibt diesen zentralen Ausschnitt des gesamten Softwareentwicklungsprozesses über eine beliebige Anzahl von Sprints konstanter Länge mit einem sich wiederholenden Ablauf. Diese Sprints aus dem „inneren Projektmanagement" werden am Anfang in der Sprintplanung durch das Team in Teilaufgaben bzw. Arbeitspakete zerlegt und über geeignete Tools beschrieben, so dass diese Aufgaben über die Dauer des Sprints durch das Team abgearbeitet werden können. Ein solcher Sprint findet sich als einzelne Phase, an deren Ende die Auslieferung des erstellten Inkrements als Meilenstein steht, im Projektplan des „äußeren Projektmanagements" wieder. Über diese gemeinsame Schnittstelle ist auch die Bewertung des Fortschritts des Gesamtprojektes auf Basis der Ergebnisse eines Sprints bzw. des Umsetzungsstandes der dort definierten Teilaufgaben möglich.

13 http://barbuda.queo.de/

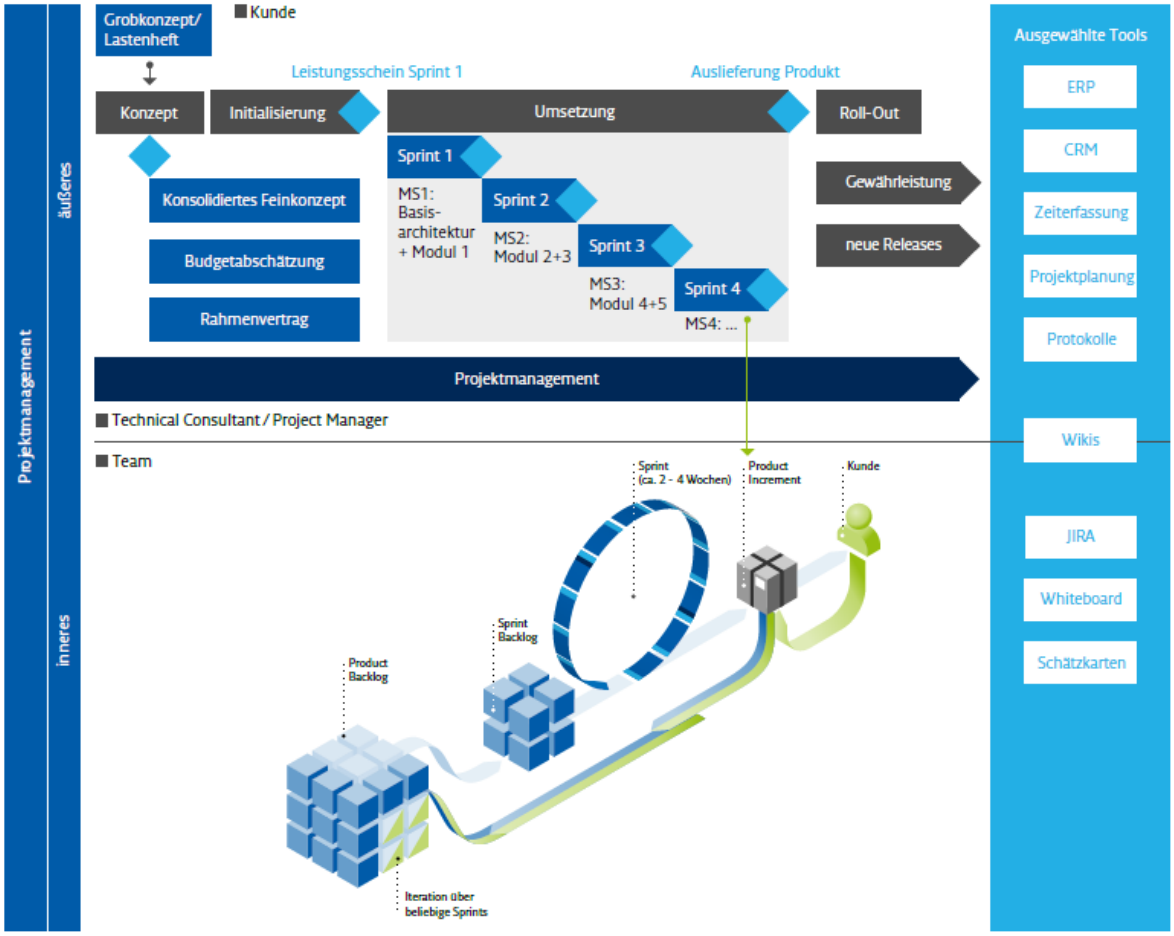

Abbildung 4: Abgrenzung "inneres" und "äußeres" Projektmanagement[14]

Dafür kommen in beiden Sichten auf das Projekt auch sehr unterschiedliche Tools mit voneinander getrennten Zielgruppen zum Einsatz, wie Abbildung 4 zeigt.

Durch diese Trennung der beiden Bereiche in „inneres“ und „äußeres“ Projektmanagement wird aber auch deutlich, was für unterschiedliche Welten dabei aufeinandertreffen. Während im inneren Projektmanagement möglichst hohe Freiheitsgrade für das Team die entscheidende Effizienz liefern, führen die Ziele des äußeren Projektmanagements, nämlich die Einhaltung von entsprechenden Regularien und Konventionen zur Beschneidung eben jener Freiheiten und damit zu einem Zielkonflikt auf beiden Seiten.

14 in Anlehnung an: Pichler, R, - Scrum - Agiles Projektmanagement erfolgreich umsetzen, dpunkt.verlag, 2008 – S. 7

4.2 Konzeption und Budgetabschätzung

Wie eingangs beschrieben, bekommt unser Gesamtprojekt eine klassische Hülle und wird im Inneren in (Teil-)Projekte („agile Inseln") zerlegt. Dabei bildet das äußere Projektmanagement die Hülle und sorgt für eine klare Kommunikation gegenüber den Verwaltungsorganen abseits der Fachabteilung, wie beispielsweise dem Projektcontrolling, dem Einkauf bzw. der Beschaffung, einem PMO oder der Öffentlichkeitsarbeit.

Um dies überhaupt so umsetzen zu können, muss im Rahmen des äußeren Projektmanagements und im Vorfeld der Realisierung der Teilprojekte und Module der grobe Umfang geklärt werden.

Dazu wird im Regelfall ein vom Kunden geliefertes Lastenheft zur Konsolidierung der Anforderungen in enger Zusammenarbeit mit dem Kunden so überarbeitet, dass es konkret alle Anforderungen der Basisfunktionalitäten sowie die potentiellen Ausbaustufen dokumentiert. Dieses konsolidierte Feinkonzept wird dann durch das Entwicklerteam in einzelne überschaubare Module zerlegt, für die ein grober Aufwand – auch auf Basis dokumentierter Annahmen – durch die Entwickler geschätzt werden kann.

Je präziser hier die Anforderungen beschrieben werden können und je weniger Annahmen getroffen werden müssen, desto genauer kann in diesem Schritt auch das zu erwartende Budget ermittelt werden. Dabei kann bei der Definition der Module auch schon darauf geachtet werden, dass in zwingend notwendige Basisumfänge und spätere Optionen unterschieden wird.

Das so zum gewünschten Realisierungsumfang ermittelte Budget kann jetzt gegenüber der Beschaffung und der IT eines Kunden genutzt werden, um die oft notwendige Verbindlichkeit über Abnahmekriterien zur späteren Bezahlung des Projektes für eine Beauftragung zu klären.
Das konsolidierte Feinkonzept ermöglicht es auch, die Umsetzung der Software unter verschiedenen Dienstleistern auszuschreiben, da die einzelnen Angebote aufgrund der so erreichten Klarheit auch wirklich miteinanderverglichen werden können.

Diese Phase könnte auch noch vor Projektstart durch einen Dienstleister wie queo im Rahmen eines separaten Beratungsprojektes realisiert werden, um zunächst eine belastbare Basis für interne Budgetplanungen zu erhalten und im Anschluss die Leistung auf Basis klarer Vorgaben ausschreiben zu können.

4.3 Projektinitialisierung und vertragliche Regelungen

Nachdem mit der groben Konzeption und der Aufteilung des Gesamtumfangs in einzelne Module eine belastbare Budgetabschätzung existiert, werden die groben Entwicklungsaufgaben, die zur Abschätzung des Umfangs der einzelnen Module dienen, in ein Product-Backlog überführt und in Sprints grob zusammengefasst. Dafür sollte ein geeignetes Tool zum Einsatz kommen, queo nutzt hierfür JIRA. 16

Um die Durchlaufzeit des Projektes zu bestimmen, muss zu diesem Zeitpunkt auch die Teamkonstellation geklärt werden. Je nach Größe des Projektes und den Terminvorgaben bzw. -wünschen des Kunden muss hier eine optimale Struktur gefunden werden. Auf Basis der dann durchschnittlich eingesetzten Mitarbeiter und ihres zeitlichen Einsatzes im Projekt kann eine durchschnittliche Kapazität an Entwicklerstunden für einen Sprint errechnet werden, wobei wir eine Sprintdauer von mindestens 2, besser 3 Wochen empfehlen.

Schätzung: 150 Entwickler-Manntage	Umfang des Minimal-Set an Anforderungen

Team aus 2 Softwareentwicklern (Vollzeit) und einem SW-Architect sowie Technical Consultant zur Unterstützung. Die Dauer eines Sprints beträgt 3 Wochen, wobei nur die Arbeitszeit der Entwickler mit 36 h / Woche, also 90 % einer 40 h-Woche, als verfügbare Kapazität angesetzt wird.

Kapazität pro Sprint: 216 h	3 Wochen x 2 Entwickler á 36 h

Die Anzahl der Entwickler-Manntage wird durch die Kapazität je Sprint geteilt, um die Mindestanzahl an Sprints für die Umsetzung des Minimalset an Anforderungen zu erhalten (Voraussetzung: Schätzung inkl. Aufwand für Modul- und GUI-Tests).

Mindestanzahl an Sprints: 6	150 Entwickler-Manntage á 8 h / 216 h = 5,56 Sprints

Auf Basis obiger Annahmen wird für das Team ein durchschnittlicher Preis je Sprint errechnet, wobei insgesamt 54 h für Projekt- und Anforderungsmanagement durch einen SW-Architect / Technical Consultant veranschlagt werden.

Projektbudget: 170.100,00 €	6 Sprints á 28.350,00 €
Durchlaufzeit: 18 Wochen	6 Sprints á 3 Wochen

Angenommene beispielhafte Tagessätze: 1000,00 € für SW-Architect / Technical Consultant und 800,00 € für Softwareentwickler

Abbildung 5: Herunterbrechen der geschätzten Entwicklerstunden auf die Anzahl der Sprints, das Budget und die Durchlaufzeit

Zum Projektende hin ist es auf Grundlage unserer Erfahrungen oft auch zielführender, die Durchlaufzeit der Sprints noch einmal zu verkürzen, da bei Anwendungen mit einem großen Fokus auf dem User-Interface spätestens dann auch das größte Feedback der Nutzer zu erwarten ist und auf diese Art und Weise eine schnelle Integration über die kürzeren Release-Zyklen gewährleistet ist.

Durch die einfache Teilung des erwarteten Gesamtaufwandes in Entwicklerstunden durch die zuvor errechnete durchschnittliche Kapazität eines Sprints erhält man die Anzahl an Iterationen, die notwendig ist, um in jedem Fall das vom Kunden avisierte Basisprodukt zu entwickeln.

Die Durchlaufzeit des Gesamtprojektes ergibt sich dann aus der Anzahl der errechneten Sprints und der konstanten Länge eines Sprints (z. B. 18 Wochen bei 6 Sprints á 3 Wochen). Ob dann später im Verlauf des Projektes die Durchlaufzeit der Sprints noch einmal verringert wird, ist dabei für die Ermittlung der Durchlaufzeit vernachlässigbar. Wichtiger ist es, dass die Zeit für die Planung des nächsten Sprints parallel zum laufenden Sprint bei der Kapazität der Entwickler zur Ermittlung der Entwicklerstunden eines Sprints berücksichtigt wird. In unserem Beispiel in Abbildung 5 wurde dafür extra ein Software Architect in Verbindung mit einem Technical Consultant eingeplant und aus der Kapazität des Sprints herausgerechnet. Wird in sehr kurzen Iterationen von nur 2 Wochen gearbeitet, dann empfiehlt es sich unter Umständen, entsprechende Pufferzeiten zwischen den Sprints bei der Ermittlung der Durchlaufzeit einzuplanen.

Im Regelfall liegen mit dem groben Ziel, dem Budget und der Durchlaufzeit bzw. dem Liefertermin die wesentlichen Kriterien vor, die für eine Beauftragung relevant sind bzw. die für Einkäufer als wesentlich erscheinen und die am Ende auch für eine Abnahme durch den Fachbereich notwendig sind. Diese Abnahme hängt am Ende auch sehr stark von der Zufriedenheit der Nutzer und somit dem Kunden ab. Hier spielen iterative Vorgehensweisen ihre Stärken aus, denn erreicht der Kunde mit der Software seine Ziele und fühlte er sich auf dem Weg dorthin jederzeit mitgenommen, dann wird er am Ende nicht auf eine 1:1-Abdeckung zwischen Software und initialer Spezifikation bestehen. Zusätzlich hat er aber durch die aktive Teilhabe jederzeit, so auch inmitten des Entstehungsprozesses, die Möglichkeit, die Passgenauigkeit der entstehenden Anwendung auf die für ihn notwendigen Kriterien zu überprüfen, die durch den Aspekt umfassender Zufriedenheit bereichert werden.

Trotzdem ist es wichtig, über Verträge Verbindlichkeit zu schaffen. Aus diesem Grund empfiehlt sich bei iterativen Vorgehensweisen die Verwendung von sogenannten Rahmenverträgen, die nur die generellen Bedingungen zur Zusammenarbeit, Themen wie Geheimhaltung und Nutzungsrechte oder die Überlassung von Quellcode regeln. Darüber hinaus ist es mit einem Rahmenvertrag möglich, dem Auftraggeber eine gewünschte Verfügbarkeit an Entwicklerkapazitäten zu garantieren, wenn dieser sich im Gegenzug zur Abnahme eines bestimmten Volumens verpflichtet. Bei einem ersten Projekt könnten das ermittelte Budget und ein Aufschlag für die Support- und Wartungsphase in Höhe von ca. 20 % eine gute Basis zur Annäherung an ein solches Volumen bieten.

Die konkreten Leistungen werden innerhalb solcher Rahmenverträge dann über sogenannte Leistungsscheine – dem eigentlichen Auftrag – spezifiziert. So können die zuvor beschriebenen Aufgaben zur Grobkonzeption und Initialisierung des Projektes über einen solchen Leistungsschein genau abgegrenzt werden. Im Anschluss ist dann die Erstellung eines Leistungsscheins für die erste Sprintplanung und Feinspezifikation der entsprechenden Entwicklungsaufgaben möglich, der dann wiederum die Basis für den Leistungsschein zur Umsetzung des ersten Sprints ist.

Hat man sich auf ein agiles Vorgehen nach SCRUM geeinigt, so stellt man im Rahmen eines Dienstleistungsvertrages über den Leistungsschein lediglich das Personal eines Teams mit den benötigten Entwicklerstunden bereit, die dann beispielsweise über eine Aufstellung der durch die Entwickler gebuchten Zeiten nachgewiesen und abgerechnet werden können. Der eigentliche Inhalt des Sprints bleibt entsprechend des SCRUM-Modell-Vorgehens offen und alle Aufgaben, die nicht realisiert werden konnten, werden entsprechend ihrer Priorisierung in darauffolgende Sprints verschoben.

Da sich einige Auftraggeber mit diesem Umstand erfahrungsgemäß schwer tun, empfiehlt es sich, die Sprintplanung und die Feinspezifikation als Leistungsbeschreibung für den Leistungsschein zu nutzen und so nicht nur die Durchlaufzeit, sondern auch den zu erbringenden Umfang des Teilprojektes festzusetzen. Diese „agile Insel“ ist im Regelfall sehr klar beschrieben, so dass das daraus resultierende Projektrisiko überschaubar bleibt. In jedem Fall wird das Ergebnis für den Auftraggeber so planbar und auch für die Freigabe von Zahlungen können klare Abnahmekriterien für einen Sprint definiert werden.

Wichtig ist auch, dass entlang des Prozesses ein erfahrener Projektmanager gemäß den Prinzipien des äußeren Projektmanagements den Weg strukturiert und die administrativen Aufgaben des Projektes im Blick behält. Je nach Organisation gibt es hier verschiedene Ansätze. Bei queo übernimmt dies im Regelfall der Technical Consultant zusätzlich zu seinen Aufgaben als Anforderungsmanager und Berater des Auftraggebers. Diese Rollen lassen sich aber auch problemlos trennen, dann beschränkt sich der Projektmanager auf die reine Administration des Projektes und verfügt über wenig inhaltliche Tiefe im Kontext des Projektes.

4.4 Umsetzung

In der Umsetzungsphase wiederholt sich das zuvor beschriebene Prozedere der Sprint-Planung und -Umsetzung über die einzelnen Sprints.

Innerhalb eines Sprints mit Planung und Umsetzung kann das Team selbstständig, gemäß den Prinzipien des agilen Manifestes, arbeiten und die sich daraus ergebenden Mehrwerte auch effektiv nutzen. Die Sprint-Planung kann je nach vereinbartem Modell der Anforderungsmanager des Auftragnehmers oder der Ansprechpartner des Auftraggebers gemeinsam mit dem Team beschreiten. Dieser kennt dann sehr genau die einzelnen Aufwandsschätzungen für die Aufgaben eines Sprints, da sie quasi in seinem Beisein entstanden sind, was zu einem gegenseitigen Verständnis und zu einem hohen Maß an Vertrauen in der Beziehung zwischen Auftragnehmer und -geber führt.

Am Ende einer jeden Iteration erfolgt die Auslieferung des ausführbaren Inkrements an den Auftraggeber, welcher parallel zur nächsten Sprint-Umsetzung dieses dann ausführlich testen kann. Das Feedback fließt bei hohen Prioritäten bei einem rein agilen Vorgehen im Idealfall sofort in den laufenden Sprint ein oder spätestens in die nächste Iteration.

Für die Auslieferung des Inkrements sollte gegen Ende des Sprints ein Feature-Freeze definiert werden, ab dem die Integration in die Gesamtanwendung getestet wird, so dass eine termingerechte Auslieferung am letzten Tag des Sprints erfolgen kann.

Zusätzlich sollte am Ende eines jeden Sprints Zeit für die Retrospektive eingeplant werden, so dass die Lessons Learned des abgeschlossenen Sprints ggf. in die Planung, zumindest aber in die Umsetzung des nächsten Sprints mit einfließen können.

4.5 Roll-Out, Wartung und Support der Endanwendung

Da das Produkt mit dem Kunden entlang der gewünschten Basisfunktionalitäten entsteht, ist auch der Roll- Out einer agil entwickelten individuellen Softwarelösung in verschiedenen Einsatzstufen möglich. Das sollte in der Planung des äußeren Projektmanagements im Vorfeld berücksichtigt werden.

Das zuvor schon beschriebene Konstrukt der Rahmenverträge bietet auch nach dem erfolgreichen Roll-Out einer Anwendung flexible Möglichkeiten, um die dann aufkommenden Anforderungen der Anwender aufzunehmen und in konkrete Weiterentwicklungen der Software zu überführen.

Hier empfiehlt sich ein laufendes Support-Budget für die Validierung von Anfragen. Entsprechend klassifizierte Bugs werden im Rahmen der Gewährleistung behoben und neue Features werden in regelmäßigen Abstimmungen mit dem Kunden definiert und dann wieder zu Sprints gebündelt, die über einen entsprechenden Leistungsschein in ein neues Release überführt werden können. Gibt es noch weitere größere geplante Entwicklungsschritte können die auch nahtlos über ein iteratives Vorgehen an den Roll-Out angeschlossen werden.

5 Unser Weg: Planbare Budgets in agilen Projekten

Agile und klassische Vorgehen haben jeweils ihre eigenen Vorteile und damit auch ihre Berechtigung. Projekte werden nicht automatisch erfolgreicher, weil sie agil bzw. iterativ umgesetzt werden. Ihren großen Mehrwert gewinnen sie aus der schrittweisen Annäherung an das Ziel. Ist dieses klar abgrenzbar und kann widerspruchsfrei beschrieben werden, so ist ein möglichst ausführliches Lastenheft des Kunden und ein darauf konsolidiertes Feinkonzept als Basis für die Aufwandsschätzung die klare Budgetgarantie.

Die unreflektierte Anwendung beider Ansätze – klassisch und agil – kann im Projektverlauf zu Problemstellungen führen, die durch die in diesem Whitepaper beschriebene Kombination minimiert werden sollen. Da dadurch auch ein höherer Abstimmungsaufwand entsteht, ist es ratsam, dass Projekte, die nach diesem Vorschlag umgesetzt werden, eine Projektgröße von mindestens 150 Entwickler-Manntagen aufweisen und auch in einer Teamkonstellation von mindestens zwei, besser drei Entwicklern bearbeitet werden. Erst dann sind Teilprojekte, in denen agile Methoden erfolgversprechender sind als der klassische Ansatz, sinnvoll abgrenzbar. In dieser Domäne des „inneren Projektmanagements“ kommunizieren und verhandeln die Teilprojektleiter bzw. -teams selbst direkt mit dem Kunden bzw. Endnutzer und können so inhaltliche Chancen und Grenzen besser ausloten. Sie setzen die Anforderungen agil um und erhalten so zügig eine Resonanz zum Erfolg der vergangenen Arbeit.

Die Organisation muss in diesem Fall die richtigen Rahmenbedingungen bereitstellen, damit klassische und agile Teilprojekte koexistieren können. Im beschriebenen Ansatz wird das durch regelmäßige Stand-Ups der eigenständigen Entwicklerteams mit dem „äußeren Projektmanager“ gewährleistet. Seine Sicht auf das agile Projekt ist der Projektplan mit den im äußeren Projektmanagement geplanten Meilensteinen. Er sorgt für die Synchronisation des Gesamtprojektplans mit den Statusmeldungen aus den einzelnen Projektteams bzw. Sprints. Die Verteilung einzelner Aufgaben und die genaue Gestaltung der Umsetzung verbleiben im inneren Projektmanagement und bei dem Projektteam – der agile Kern des Projekts bleibt daher unberührt.

Budgetkonformität erreicht man auch im von uns beschrieben Ansatz nur, wenn vor Projektstart im Rahmen einer Konsolidierung des Lastenheftes im äußeren Projektmanagement das Minimal-Set des noch unscharfen Ziels so präzise wie möglich beschrieben ist. Auf Basis dieser Umfangs- und Funktionsbeschreibung kann der Kunde den festgelegten Projektpreis bei weiteren Anbietern am Markt, z. B. im Rahmen einer Ausschreibung, validieren lassen.

Das so entstehende Produkt erfüllt die Minimalanforderungen des Kunden, so dass er seine gewünschten Ziele erreicht und mit dem Ergebnis zufrieden ist. Das sollte als Abnahmekriterium schwerer wiegen als die sklavische Abarbeitung von Akzeptanzkriterien, die zu einem Zeitpunkt definiert wurden, als das Ziel noch unscharf war und im Nebel lag. Jener Begriff von kundenseitiger Zufriedenheit speist sich innerhalb des agilen Projekts aus einem gegenseitigen Vertrauensverhältnis zwischen Kunde und Dienstleister. Vertrauen wiederum kann in

diesem Zusammenhang nicht weiter aufgebaut werden, es liegt schlichtweg schon mit der Entscheidung des Kunden für einen Dienstleister bei Vertragsunterzeichnung vor. Mit Beginn der Zusammenarbeit bestehen so nur noch zwei Szenarien in Bezug auf den Vertrauensvorschuss, den der Kunde initial zu investieren bereit war: Die Konsolidierung oder Enttäuschung dieses Verhältnisses. Als erfahrener Dienstleister im Bereich der individuellen Softwareentwicklung verstehen wir unser Ziel in der Konsolidierung und dem Ausbau des Vertrauensverhältnisses zwischen uns und unseren Kunden.

Mit der Realisierung gewünschter Projekte verpflichten wir uns nicht nur zu den vertraglich zugesicherten Leistungen, sondern pflegen einen gleichermaßen selbstreflektierten Umgang mit dem bereitgestellten Budget. Trotz variabler Projektziele sehen wir das bereitgestellte Budget innerhalb des Projektrahmens als Fixum, mit dem das eigenständig agierende Projektteam umgeht, als ob dieses in eine eigene Software investiert wird.

Über uns

Das nächste Level erreichen! Mit diesem Anspruch entwickelt queo Markenkommunikation und Kampagnen, Websites und Portale sowie individuelle Softwarelösungen. Im Spannungsfeld aus Kommunikation und Technologie arbeiten 150 queos. Sie setzen sich für den Erfolg ihrer Kunden ein. Dazu gehören öffentliche Verwaltungen, die AOK, der Zentralverband Deutsches Kfz-Gewerbe, Koenig & Bauer, Credit Suisse, Volkswagen und andere mehr. Zusammen erreichen sie den Status queo. Seit 15 Jahren. In Dresden, Berlin und Solothurn (Schweiz). www.queo.de

Literaturverzeichnis

Heise:

http://www.heise.de/developer/meldung/Studie-Agile-Softwareentwicklung-ist-Mainstream-912207.htmlhttp://www.heise.de/developer/meldung/Studie-Agile-Methoden-im-Hoehenflug-2442650.html
Beck, K.: Extreme Programming. Das Manifest, 2. Auflage, Addison Wesley, 2001.
Beedle, M.; Schwaber, K.: Agile Software Development with Scrum, Prentice Hall, 2002.
Pichler, R.: Scrum – Agiles Projektmanagement erfolgreich umsetzen, dpunkt.verlag, 2008.
Schwaber, K.: Agile Project Management with Scrum, Microsoft Press, 2004
Oestereich, B.; Weiss, Chr.: APM – Agiles Projektmanagement: Erfolgreiches Timeboxing für IT-Projekte, dpunkt.verlag, 2008.
Larman, C.: The historical accident of waterfal validity, in: Agile and iterative development, 2004
Rubin, Keneth S.: Essential Scrum – Umfassendes Scrum-Wissen aus der Praxis, MITP, 2014.
Royce, W.: Managing the development of large software sytems, in: Proceeding of IEEE WESCON, 1970.