Uwe Baumann
(ubaumann@artiso.com)
ist ausgewiesener Experte für Microsoft Visual Studio, ALM und .NET. Seit 1999 begleitet er die verschiedenen Technologien zur Softwareentwicklung in der Microsoft-Welt. Als ALM Consultant und strategischer Technologieberater bei der artiso solutions GmbH unterstützt er mit seiner umfassenden Erfahrung Kunden dabei, den Herausforderungen der modernen Softwareentwicklung zu begegnen.

Thomas Fischer
(kontakt@thomasfischerconsulting.de)
ist Betriebswirt (VWA) und begann 1995 seine Karriere im Bereich Human Resources bei einem großen und renommierten Handelsunternehmen. 2006 wechselte er zur Firma KUKA Roboter GmbH als Director Human Resources und verantwortete für die KUKA AG den Bereich Human Resources Development (HRD), bevor er 2014 seine eigene Firma, thomasFISCHERconsulting, gründete. Seit dem Jahr 2011 begleitet ihn beruflich und privat das Thema der Agilität. Im Jahr 2014 beginnt er in Zusammenarbeit mit Microsoft Deutschland, Vorträge zu diesem Thema zu halten. Er ist ein Verfechter der modernen Personalarbeit, sein Credo lautet „Gemeinsam die Zukunft gestalten".

Thomas Schissler
(tschissler@artiso.com)
ist Geschäftsführer der artiso solutions GmbH. Er beschäftigt sich seit dem Jahr 2001 intensiv mit der Softwareentwicklung auf Basis von .NET und hat sich auf die Themen Team Foundation Server, moderne Architekturkonzepte, Prozessdesign und verschiedene Technologien im Umfeld von Microsoft .NET spezialisiert. Er zählt in diesen Bereichen zu den anerkannten Top-Experten. Als zertifizierter Professional SCRUM Trainer und Professional SCRUM Master unterstützt Thomas Schissler Unternehmen bei der Einführung agiler Softwareentwicklungsmethoden und effektiver Prozesse mit Visual Studio. Hierbei kann er sich neben seinem theoretischen Wissen auf umfangreiche Praxiserfahrungen aus der Tätigkeit mit den eigenen Teams bei artiso sowie aus zahlreichen im Kundenauftrag durchgeführten Trainings, Consultings und Coachings on the job stützen.

Scrum für Manager: Warum lohnt sich Agilität aus betriebswirtschaftlicher Sicht?

Aktuell ist Agilität in der IT ein wichtiger Trend, der immer mehr Anhänger findet. Für Unternehmer, Personalleiter, Produktions- und IT-Leiter war das Vorgehen in Software- & IT-Projekten noch nie so sehr ein gemeinsames Thema wie im Moment. Dies ist nicht verwunderlich, denn aus betriebswirtschaftlicher Sicht bietet agile Softwareentwicklung entscheidende Wettbewerbsvorteile und trägt zu einer effektiven Risikominimierung bei.
Im vorliegenden Whitepaper werden die betriebswirtschaftlichen Hintergründe agiler Softwareentwicklung erläutert. Es wird aufgezeigt, wie auch Ihr Unternehmen profitieren kann und wie Sie sich auf die Transformation zu mehr Agilität vorbereiten können.

Die Herausforderung

Softwareprojekte sind komplex und schlagen deshalb sehr oft fehl – das heißt, sie überschreiten die festgelegte Zeit, das Budget oder liefern sogar gar kein verwertbares Produkt aus. Diese Binsenweisheit über die Softwareentwicklung wird wiederholt durch Studien bestätigt. Der bekannte *CHAOS-Report* des Beratungsunternehmens *The Standish Group* geht in der aktuellsten Version davon aus, dass etwa zwei von drei Softwareprojekten scheitern – eine etwas optimistischere Umfrage der renommierten Fachzeitschrift *Dr. Dobb's Journal* [Dobbs] schätzt, dass etwa die Hälfte der konventionell durchgeführten Projekte nicht erfolgreich abgeschlossen wird.

Abb. 1: Traditionelles Phasenmodell der Softwareentwicklung

Die Gründe dafür sind vielfältig und reichen von genereller Ineffizienz im Projektmanagement über mangelhaftes Anforderungs- und Qualitätsmanagement, unzureichende Fähigkeiten der Beteiligten bis zu einem fehlenden Fokus auf den Kundennutzen.

Geht es auch anders?

Auf den ersten Blick scheint es, als ob all diese Herausforderungen einfach akzeptiert werden müssten – schließlich hat die gesamte Branche mit den gleichen Problemen zu kämpfen. Dabei setzt gerade in der Softwareentwicklung ein Umdenken ein, das zu einer kompletten Umgestaltung der Entwicklungsprozesse in vielen Unternehmen geführt hat. Am Anfang steht dabei die Frage: Was macht ein erfolgreiches Softwareprojekt aus?

Aus betriebswirtschaftlicher Sicht ist diese Frage leicht zu beantworten: Entscheider erhoffen sich von neuer Software

Abb. 2: Iteratives Modell der Softwareentwicklung

Kosten- und Wettbewerbsvorteile. Dass die Investition in die Entwicklung einer neuen Software auch immer ein Risiko bedeutet, ist den meisten Auftraggebern klar – aber wie bei jeder Produktentwicklung gilt es, das Risiko und damit die potenziellen erhöhten Entwicklungskosten soweit wie möglich zu minimieren.

Kurze Zyklen, große Vorteile

Stellt man diese grundlegenden Überlegungen in den Mittelpunkt der Analyse, zeigt sich, warum der traditionelle Ansatz der Softwareentwicklung in der Krise steckt. Der konventionelle Entwicklungsprozess beruht auf einem Phasenmodell – auch „Wasserfall-Methode“ genannt (siehe Abbildung 1). Dabei werden zunächst alle Anforderungen erfasst. Basierend auf den Anforderungen wird die System- und Softwarearchitektur entworfen und anschließend in einer Implementierungsphase umgesetzt. Schließlich wird das Produkt getestet und nach erfolgreicher Abnahme ausgeliefert und in Betrieb genommen.

Moderne Ansätze in der Softwareentwicklung setzten hingegen auf ein iteratives Modell. Anstatt einen Makro-Zyklus zu durchlaufen, besteht ein Projekt hier aus vielen Mikro-Zyklen. Jede dieser Iterationen enthält dabei genau die Elemente, die auch aus dem traditionellen Prozess bekannt sind – auch hier werden Anforderungen erfasst, umgesetzt, es wird getestet und am Ende ein Produkt ausgeliefert (siehe Abbildung 2). Dieses Produkt hat dabei prozessbedingt noch nicht den am Projektende gewünschten vollen Umfang. Anders als bei einem „Meilenstein“ im klassischen Phasenmodell ist dieses Inkrement jedoch bereits voll getestet und die darin enthaltene Funktionalität ist schon voll einsetzbar.

Die kürzeren Zyklen machen dabei den entscheidenden Unterschied: Nach jedem Zyklus werden Anforderungen neu bewertet. Erkenntnisse aus dem letzten Zyklus können in die nächste Iteration einfließen und am Ende jedes Zyklus steht *Working Software*.

Dies bedeutet, dass erste Funktionen bereits zu einem früheren Zeitpunkt ausgeliefert werden können und die Time-to-Market reduziert wird. Nach ein paar Iterationen steht eine einsetzbare Software (Release) bereit, die zwar noch einige Funktionen nicht enthält, aber bereits ersten Kundennutzen generiert (siehe Abbildung 3).

Die Flexibilität des iterativen Modells zahlt sich auch aus, wenn die Marktgegebenheiten sich während des Projektverlaufs ändern – heute eher die Regel als die Ausnahme. Die nötigen Kurskorrekturen während des Projekts sind mit vertretbarem Aufwand realisierbar.

Und nicht zuletzt kann - und soll - nach jeder Iteration Anwenderfeedback berücksichtigt werden: Auch dies trägt entscheidend dazu bei, den Kundennutzen und die Akzeptanz des Produkts zu erhöhen.

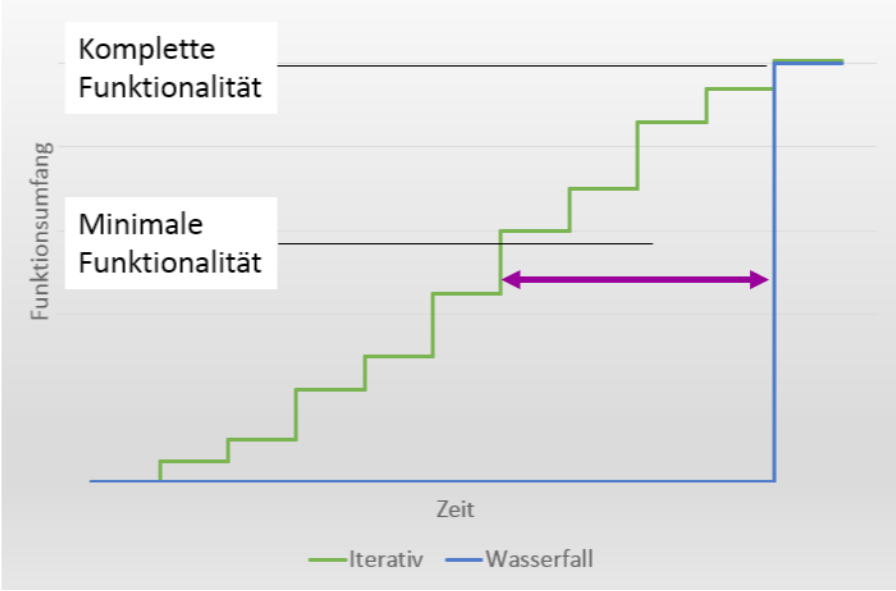

Abb. 3: Iterative Entwicklung stellt früher verwertbare Funktionalität bereit

Reduziertes Risiko statt „Projekt-Kidnapping“

Der erfolgreiche Einsatz von speziell entwickelter Software bedeutet signifikante Wettbewerbsvorteile und erhebliches Einsparpotenzial für Unternehmen. Dennoch ist für Entscheider nicht nur der Blick auf die potenziellen Vorteile wichtig, sondern auch ein nüchterner Blick auf die Risiken, welche sich bei einer Software-Produktentwicklung ergeben.

Beginnen wir beim persönlichen *Worst Case* für viele Entscheider: Welcher Auftraggeber eines Softwareprojekts fürchtet sich nicht vor dem berüchtigten „Projekt-Kidnapping“? Die Deadline ist erreicht, das geplante Budget ist verbraucht - aber das Softwareprojekt ist noch nicht fertig und damit unbrauchbar. Jetzt heißen die Alterativen Abschreiben oder Nachfinanzieren, mit allen unschönen Konsequenzen für die verantwortlichen Entscheider.

Was ist passiert? Beim Phasenmodell wird versucht, das Projektrisiko durch Monitoring zu reduzieren. Das Monitoring basiert aber auf Status-Reports der Entwicklung – und diese Reports sind unter Umständen fehlerhaft, missverständlich, verzögert oder im Extremfall sogar gefälscht. Für Verantwortliche ist es zudem schwer, den tatsächlichen Projektstand realistisch zu bewerten.

Im traditionellen Modell treten viele Probleme zudem erst spät im Projekt auf und machen das Risiko damit schwer kalkulierbar. Wer kann schon vorhersagen, wie viele Fehler beim Testen gefunden werden und wie lange es dauern wird, bis eine stabile Software verfügbar ist?

Beim iterativen Modell wird der Status durch die einsetzbaren Zwischenstände dokumentiert – ein überraschend effektives Verfahren, das wenig aussagekräftige Metriken und schwer nachprüfbare Berichte durch einen konkreten Funktionsbeweis ersetzt. So reduziert sich das Risiko eines unvorhergesehenen Totalausfalls für den Entscheider erheblich.

Projekte, die nach Verbrauch des Budgets oder nach Ablauf der Projekt-Deadline noch nicht komplett fertiggestellt sind, enthalten dennoch einen Großteil der Funktionalität, denn diese wurde ja bereits in den vorangegangenen Phasen implementiert und ist – getreu den Regeln

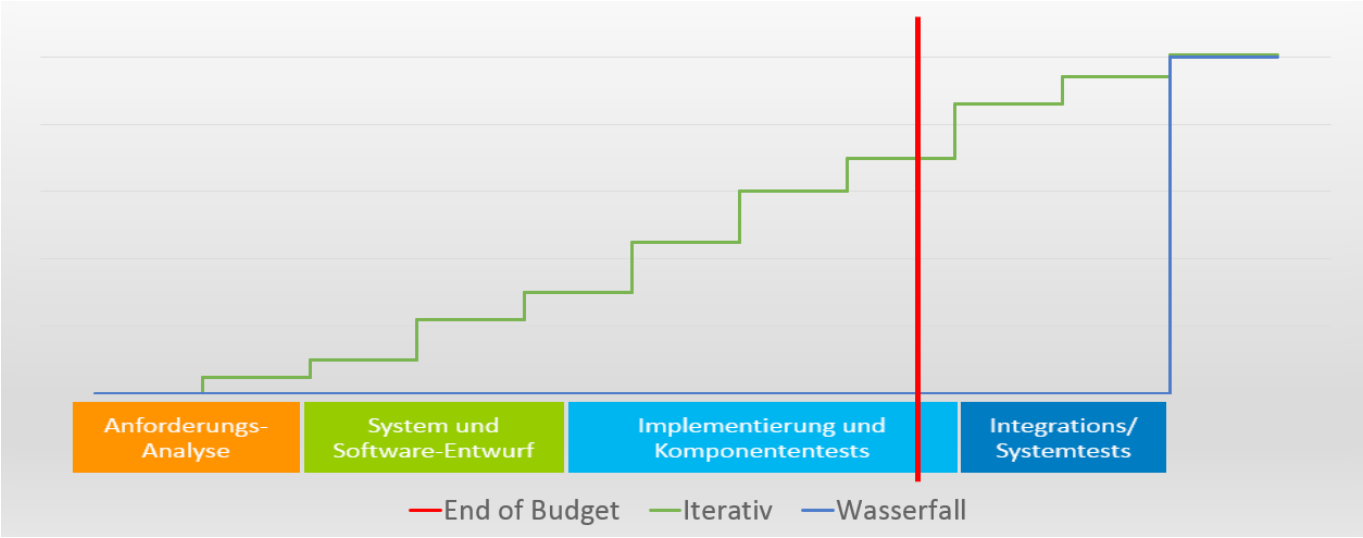

Abb. 4: Verwertbare Funktionalität bei verbrauchtem Budget

für iterative Entwicklung – auch einsetzbar. Statt der oben erwähnten Entscheidung „Alles oder Nichts" hat der Verantwortliche in diesem Fall eine bessere Option – mit dem etwas eingeschränkten Funktionsumfang zu leben oder einen in Relation geringeren Betrag für die Fertigstellung der noch ausstehenden Funktionalität zu investieren (siehe Abbildung 4).

Business Value statt Features

Eine weitere Form des Risikos in der Softwareentwicklung ist nicht so offensichtlich wie die Gefahr des teilweisen oder totalen Scheitern des Projekts: Es besteht die Möglichkeit, dass die Software zwar fertig wird, aber nicht die Erwartungen der Nutzer erfüllt und damit einen begrenzten Wert für das Business hat.

Beim iterativen Modell wird versucht, dieses versteckte Risiko durch ständiges Feedback der späteren Nutzer und eine ständige Überprüfung der Prioritäten während des Projektverlaufs zu reduzieren. Die Erfahrung zeigt zwar, dass Nutzer am Anfang des Projekts nicht sauber zwischen wichtigen und optionalen Funktionen unterscheiden können. Sobald jedoch nach den ersten Iterationen „Software zum Anfassen" zur Verfügung steht, ist es für alle Beteiligten viel einfacher, die Relevanz einzelner Anforderungen objektiv zu beurteilen. Es fällt deshalb in der Regel auch leichter zu beurteilen, welche Funktionalität aus der Anforderungsliste als Nächstes implementiert werden sollte.

Oft entstehen so Einsichten, die im weiteren Projektverlauf verwertet werden können – sei es, dass überflüssige Anforderungen gestrichen werden oder dass neue, bisher übersehene Anforderungen hinzugefügt werden. Die weitere Arbeit wird neu priorisiert, sodass zu jeder Zeit immer an den wichtigsten Funktionalitäten gearbeitet wird. Oder, anders ausgedrückt: Beim Phasenmodell müssen die Beteiligten alle guten Ideen zu Beginn des Projektes haben – beim iterativen Modell sorgt ein ständiger Lernprozess dafür, dass die Software sich immer mehr an die wirklichen Anforderungen annähert.

Die finanziellen Implikationen

Die bisher erwähnten Vorteile der iterativen Vorgehensweise erscheinen plausibel – aber halten die Behauptungen auch einer kühlen betriebswirtschaftlichen Kalkulation stand?

Wir haben für unsere Kalkulation ein fiktives, aber realistisches Projekt als Beispiel gewählt:

- Eine Maschinenbaufirma benötigt eine *Software zur Maschinenkonfiguration*.
- Die *Projektdauer* beträgt 12 *Monate*, bei *Projektkosten* von *5 Millionen Euro*.
- Die voraussichtliche *Lebensdauer* der Software beträgt *48 Monate*.
- Die Software realisiert einen *Effizienzgewinn von 200.000 Euro pro Monat*.

Etwas vereinfacht gesprochen sind für eine Betrachtung drei Faktoren nötig:

- Die *Kapitalrendite* (Return on Investment, ROI): Die Rendite auf das eingesetzte Kapital
- Der *Kapitalwert* (Net Present Value, NPV): Die Summe der Barwerte aller durch diese Investition verursachten Zahlungen inklusive Verzinsung
- Die *Gewinnschwelle* (Break-even-Point): Der Zeitpunkt, an dem Erlös und Kosten gleich groß sind

Übrigens: Falls Sie die Kalkulation mit Ihrer eigenen Projektkonfiguration durchrechnen wollen: Unter http://1drv.ms/1tUWFC2 haben wir eine Excel-Datei dafür vorbereitet.

Business Case: Phasenmodell gegen iterativen Prozess

Bei einem klassischen Projekt nach Phasenmodell („Wasserfall-Methode") ergibt sich mit diesen Daten ein Break-even-Point nach 37 Monaten bei einer Kapitalrendite von 44 Prozent und einem Kapitalwert von 1.481.000 Euro. Wichtig hier: Erst nach 12 Monaten greift der durch die Software realisierte Effizienzgewinn, vorher entstehen lediglich Kosten (siehe Abbildung 5, blaue Kurve).

Bei einem iterativen Modell mit einer Iterationsdauer von 2 Wochen und einem Releasezyklus von 6 Monaten lässt sich der Effekt beobachten, der entsteht, wenn zu einem früheren Zeitpunkt (nach 6 Monaten) ein Teil der Funktionalität zur Nutzung bereitgestellt wird:

Die rote Kurve zeigt bereits nach 6 Monaten erste Einsparungen, denn ab diesem Zeitpunkt lässt sich die Software nutzen und realisiert erste Effizienzgewinne. Der Break-even-Point wird nach 34 Monaten (statt 37) erreicht, die Kapitalrendite steigt auf 56 Prozent (statt 44), und der Kapitalwert steigt um 39 Prozent auf 2.058.000 Euro (von 1.481.000 Euro). Das Investitionsvolumen wird aufgrund

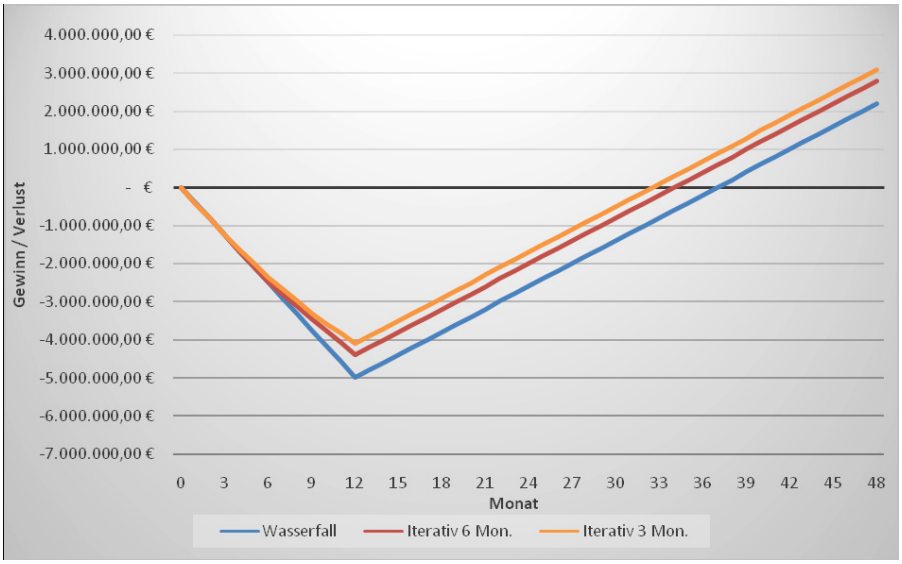

Abb. 5: Wasserfall vs. Iteration

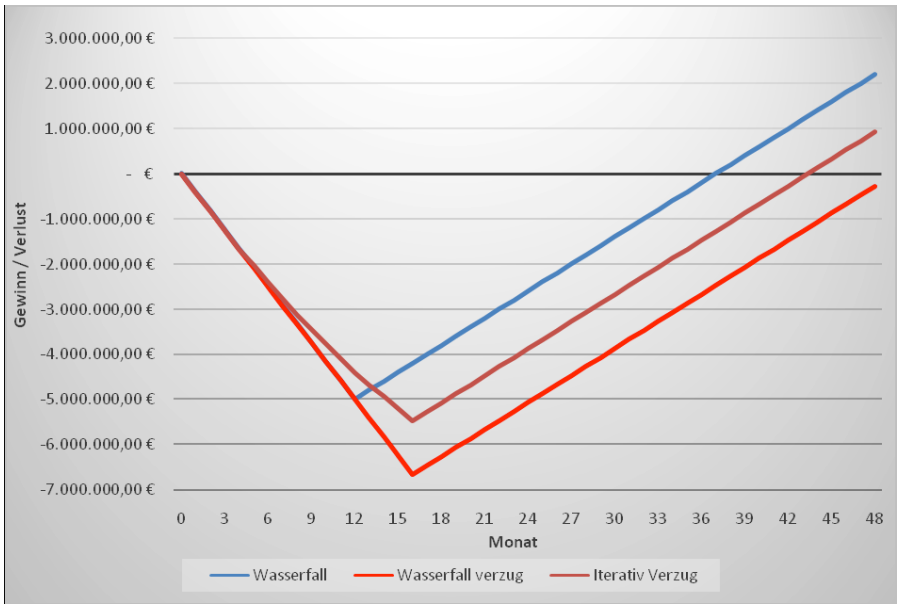

Abb. 6: Wasserfall vs. Iterativ, Iteration = 3 Monate, Verzögerung 4 Monate

dieser Einsparungen nicht komplett ausgeschöpft!

Eine weitere Verkürzung der Releasezyklen auf drei Monate bringt eine nochmalige Verstärkung der relevanten Effekte, denn die Software steht nun noch früher zur Verfügung (orangefarbene Kurve).

In diesem Fall wird der Break-event-Point bereits nach 33 Monaten (statt 37) erreicht, die Kapitalrendite liegt jetzt bei 62 Prozent (vorher 44) und der Kapitalwert beträgt 2.348.000 Euro.

Wichtig: Nach der Projektdauer von 12 Monaten steigen alle Kurven steil an, da ab diesem Zeitpunkt keine Entwicklungskosten mehr entstehen und die Software sich fortan amortisiert. Die Kennlinien der iterativen Variante fallen jedoch schon vorher nicht so stark ab – es ist ja bereits Teilfunktionalität nutzbar und damit ein erster Return on Investment sichtbar.

Business Case: Kosten von Projektverzögerungen

Verzögerungen sind bei Softwareprojekten an der Tagesordnung – und deshalb ist unser kleines Rechenmodell nur realistisch, wenn wir diesen Fall ebenfalls betrachten. Was passiert, wenn wir von einer Projektverzögerung um 4 Monate ausgehen (siehe Abbildung 6)?

Durch die zusätzliche Projektdauer ergibt sich, dass vier Monate lang weitere Investitionskosten entstehen, somit erhöht sich das Investitionsvolumen entsprechend. Der Break-even-Point wird jetzt erst nach mehr als 49 Monaten (gegenüber 37) erreicht (rote Kurve). Die Kapitalrendite ist negativ – wir verlieren Geld – und liegt bei -4 Prozent. Der Kapitalwert liegt bei -841.00 Euro. Ein drastischer finanzieller Effekt für „nur“ 4 Monate Verzug!

Beeindruckend ist die Performance der iterativen Prozessvariante in diesem Fall (braune Kurve): Mit einem Break-even-Point bei 43 Monaten, einer Kapitalrendite von 14 Prozent und einem Kapitalwert von 301.000 Euro ergibt sich ein überraschendes Bild: Das fiktive Projekt wird mit dem herkömmlichen Phasenmodell ein finanzielles Desaster, bei einem iterativen Prozess sind die Konsequenzen weit weniger drastisch: Wir realisieren immerhin noch eine positive Kapitalrendite.

Iterative Methoden im klaren Vorteil

Es ergibt sich ein klares Bild: Iterative Methoden bringen klare Vorteile gegenüber den traditionellen Methoden des Software-Projektmanagements. So ist es nur konsequent, dass sich in den letzten Jahren dieses Vorgehensmodell zunehmend in der Branche durchgesetzt hat – zusammen mit dem Begriff der „Agilen Entwicklung“, der für unterschiedliche Varianten der iterativ gesteuerten Entwicklung stehen kann. Dabei sticht ein Vorgehensmodell deutlich heraus: „Scrum“ ist der momentan verbreitetste Vertreter der iterativen Entwicklung und wurde über die Jahre ständig verbessert.

Agilität = Scrum

Scrum arbeitet nach einem einfachen Konzept: Anforderungen werden fortlaufend in eine priorisierte Liste erfasst, dem „Product Backlog“. Ein Produktverantwortlicher, der „Product Owner“, priorisiert die Anforderungen und das Entwicklungsteam setzt die jeweils wichtigsten Anforderungen in kurzen, zeitlich begrenzten Iterationen („Sprints“) um. Dabei organisiert das Entwicklungsteam die Arbeit selbsttätig und bestimmt auch, wie viele der Anforderungen im aktuellen Sprint umsetzbar sind. Am Ende jedes Sprints steht eine potenziell einsetzbare Version der Software. Vor jedem neuen Sprint wird neu priorisiert, und nach jedem Sprint erfolgt eine Phase der Analyse und Adaption. Das Scrum-Vorgehensmodell ermöglicht die in diesem Paper be-

schriebenen Vorteile der iterativen Entwicklung auf überzeugende Weise.

Der Weg zu Scrum

Das Potenzial, das sich für Sie mit einem funktionierenden agilen Prozess wie Scrum erschließt, ist enorm. Entscheidend ist jedoch, dass zusätzlich zur Organisation der Entwicklungsaufgaben auch das Umfeld betrachtet wird. Denn: Einführung von Agilität ist nicht nur die Einführung eines neuen Prozesses, sondern bedeutet auch eine Veränderung der Menschen und der Unternehmenskultur – eine anspruchsvolle Aufgabe für das Management und alle Mitarbeiter.

Falls Sie von den Vorteilen agiler Entwicklung profitieren wollen, kann es hilfreich sein, den ersten Schritt mit Unterstützung eines kompetenten Partners zu tun, der die Transformation begleitet und moderiert. Professionelle Scrum-Trainer können auf jahrelange Erfahrung zurückgreifen – ein unschätzbarer Vorteil auf dem Weg zum Erfolg. ■

Literatur & Links

[artiso] NET-Entwicklung – Individuelle Softwarelösungen, artiso Solution GmbH, siehe: http://www.artiso.com

[Dobbs] Sc. W. Ambler, The Non-Existent Software Crisis: Debunking the Chaos Report, 4.2.2014, siehe: http://www.drdobbs.com/architecture-and-design/the-non-existent-software-crisis-debunki/240165910

[Standish] Café CHAOS, siehe: http://blog.standishgroup.com