

Inkrementorientierte Vorgehensmodelle

Die inkrementale Entwicklung von Software eignet sich besonders, wenn die zukünftigen Anwender nicht genau spezifizierbare Vorstellungen über die Funktionalität des Endprodukts (Finalziel) haben. Die folgende Aussage mag das illustrieren: "Ich kann Ihnen nicht genau sagen, was ich benötige, aber wenn ich es sehe, dann weiß ich es". Man sollte sich allerdings davor hüten, dies generell dem Unvermögen der Anwender zuzuschreiben, denn gerade wenn kreative Akte gefordert sind, hat selbst der Spezialist selten eine konkrete Anforderungsspezifikation vor Augen bzw. in der Hinterhand.

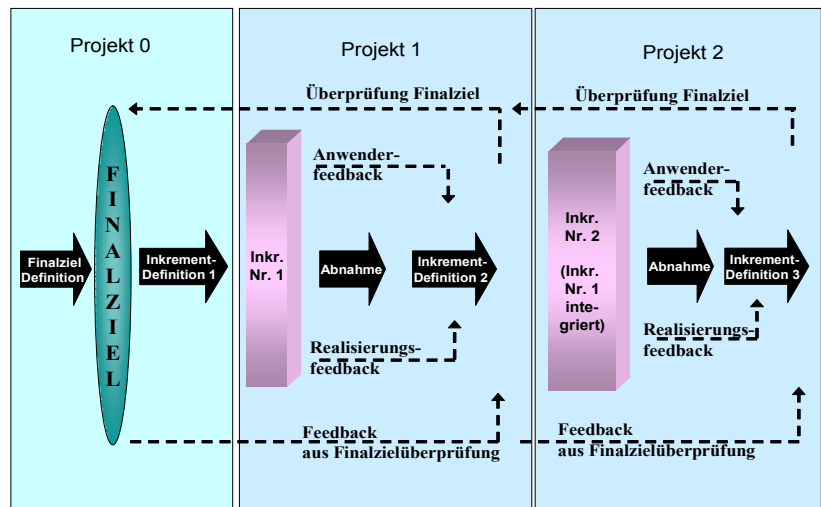


Bild 2: Das inkrementorientierte Modell.

Die inkrementorientierte Vorgehensweise zielt darauf ab, dem Anwender sehr früh im Entwicklungszyklus lauffähige Versionen für den Test bereitzustellen. Damit muss keine Produktionsübergabe verbunden sein. Der Vorteil ist, dass der Anwender seine Vorstellung über das Produkt (typische Inkrementdauern liegen je nach Projekttyp zwischen ein und sechs Monaten) anhand eines ersten lauffähigen Inkrements überprüfen kann. Aufgrund seiner konkreten Erfahrung, die er beim Test des ersten Inkrements gewinnt, wird in Absprache mit dem Entwicklungsteam die Funktionalität des nächsten Inkrements geplant. Zusätzlich besteht nach jeder Inkrementfertigstellung die Möglichkeit, Änderungen des Umfelds oder der hauseigenen Strategie in die Planung für das nächste Inkrement aufzunehmen.

Ein weiterer unbestrittener Vorteil der Methode ist, dass vermutete technische Risiken zusammen mit den Anwenderfunktionen in den ersten Inkrementen adressiert werden können, um die unangenehme Überraschung zu vermeiden, dass man am Ende eines langen Konzeptionswegs vor unüberwindlichen technischen Problemen steht und die ganze Investition abgeschrieben werden muss. Ein Problem des rein inkrementalen Vorgehens tritt dann auf, wenn die Softwarearchitektur mit der Weiterentwicklung nicht Schritt halten kann und eine komplette Überarbeitung der bis zu diesem Zeitpunkt erstellten Inkremente notwendig wird. In diesem Fall haben die bisher erstellten Inkremente den Status eines Wegwerfprototyps, der allerdings den großen Vorteil hat, dass er die verifizierten und sehr wahrscheinlich auch die validierten, funktionalen Anforderungen auf dem konkretesten aller möglichen Level enthält.

Das Modell in Bild 2 stellt die inkrementorientierte Entwicklung eines Softwareprodukts dar. Es zeigt, dass nach jeder Inkrementfertigstellung die Ziele für das jeweils nächste Inkrement gemeinsam von Anwendern und Entwicklern definiert werden und eine Überprüfung des Finalziels stattfindet. Im Unterschied zum Wasserfallmodell wird das Inkrementergebnis innerhalb kurzer Zeitspannen und nicht erst am Ende des Entwicklungszyklus zur Verifikation und Validierung bereitgestellt. Somit lässt sich der Aufwand für Änderungen und Fehler senken. Für den geringeren Aufwand gibt es zwei Gründe: Es müssen weniger bzw. in geringerem Umfang Entwicklungszwischenprodukte (Artefakte) geändert werden und der Qualitätssicherungsaufwand für Artefakte kann aufgrund der geringeren Fehlerbehebungskosten wesentlich niedriger gehalten werden (Jens Coldewey).

Da Inkremente iterativ entwickelt werden und auch laufend mit ihrer Softwarebasis integriert werden, kommt einer effizienten werkzeuggestützten Testunterstützung eine wesentliche Bedeutung zu.

Die Zutaten der Inkrementorientierung

1. Eine mehr oder weniger genaue Finalzielbeschreibung muss zu Projektbeginn angefertigt werden. (Was soll das Produkt in der Endausbaustufe leisten können?)

2. Ein Inkrement ist so zu spezifizieren, dass es einen Anwendernutzen hat und vom Anwender im Test abgenommen werden kann (ob eine Produktionsübergabe nach dem Test erfolgt, hängt von weiteren Bedingungen ab).
3. Zu Beginn einer Inkrementserstellung wird unter Berücksichtigung des zur Verfügung stehenden Zeitrahmens das Inkrementziel spezifiziert. Für jedes Inkrement (außer dem ersten) wird eine Finalzielüberprüfung vorgenommen und die Anwender- wie die Realisierungserfahrung der vorausgegangenen Inkremententwicklungen ausgewertet. Diese Erfahrungen fließen in die Spezifikation für das nächste Inkrement ein.
4. Ein Vertrag wird immer nur für das nächste zu erstellende Inkrement geschlossen (nicht bezogen auf das Finalziel!).

Während die ersten Punkte relativ plausibel klingen, hat es der letzte Punkt in sich. Warum soll der Auftraggeber auf "Planungssicherheit" verzichten, wenn er möglicherweise zu einem Festpreis die Gesamtfunktionalität "risikolos" geliefert bekommen kann? Um diese Frage zu klären, sollen zunächst kurz die Hauptrisiken der Softwareentwicklung betrachtet werden.

Laufzeit und Funktionsumfang als Hauptrisiken der Softwareentwicklung

Projektumfang ist größter Risikofaktor für Softwareprojekte

Eine empirische Untersuchung von T. Capers Jones (T. Capers Jones, "Estimating Software Costs") auf Basis von ermittelten Function Points, die als ein standardisiertes Maß für den funktionalen Umfang von Applikationen verwendet werden, zeigt Bild 3:

Die Abbruchwahrscheinlichkeit steigt überproportional mit zunehmender Funktionalität der zu realisierenden Software. Eintausend Function Points entsprechen nach Capers Jones' Daumenregel einem Aufwand von 106 Personenmonaten, die einen kalkulierten Zeitplan von 16 Monaten ergeben (Capers Jones, "By Popular Demand: Software Estimating Rules of Thumb"). Über Kostenüberschreitungen und Leistungsumfangskürzungen bzw. Qualitätsprobleme macht Capers Jones in diesem Zusammenhang keine Aussage.

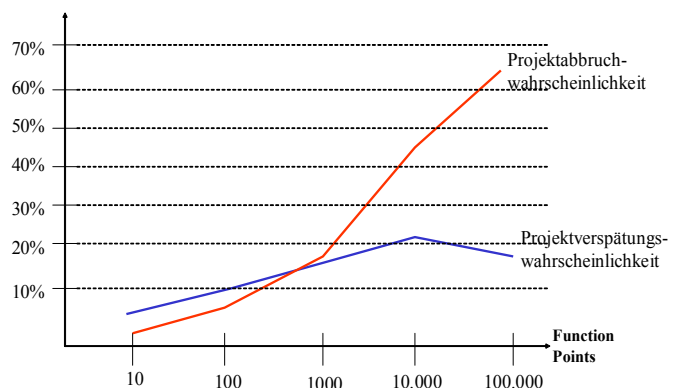


Bild 3: Abbruchwahrscheinlichkeit in Abhängigkeit von Function Points.

Das Duo infernale: Unklare Ziele und großer Leistungsumfang

Je länger die Entwicklungsarbeit dauert, desto schwieriger wird es, eine produktive Anwendung mit den erwarteten Erträgen zu realisieren. Unklare Ziele und Anforderungen bei kurzem Planungshorizont (die linke, blaue Säule in Bild 4) sind unter risikotechnischen Erwägungen gut vertretbar. Treten aber zu einem langen Planungshorizont unklare bzw. im Zeitablauf veränderliche Anforderungen hinzu, steigt das Realisierungsrisiko exorbitant. Oft geht ein langer Planungshorizont bzw. eine lange Projektlaufzeit mit ungenauen Zielvorstellungen Hand in Hand.

Unklare oder veränderliche Ziele

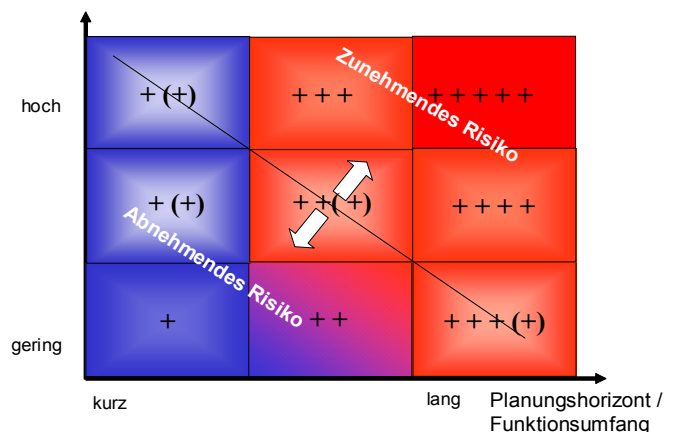


Bild 4: Projekt-Realisierungsrisiko in Abhängigkeit von Zielunklarheit und Planungshorizont.

Ob sich die Probleme der Softwareentwicklung auf schlecht geschätzte Vorgaben zurückführen lassen oder auf ungeplante Schwierigkei-

ten, die sich bei der Abwicklung einstellen, ist hier unerheblich – es ist unmittelbar einsichtig, dass in einer nicht stillstehenden Umwelt mit zunehmendem Projektionszeitraum bzw. Funktionsumfang Prognosen immer unzuverlässiger werden.

Der doppelte Hebel der Risikoreduzierung durch Inkrementorientierung

Das Risiko besteht bekanntermaßen aus dem Produkt der Risikoeintrittswahrscheinlichkeit und der Folgekosten bei Eintritt des Risikos. In den folgenden Abschnitten wird die Wirkung der Inkrementorientierung auf die beiden Komponenten des Risikos dargestellt.

Wie Inkrementorientierung die Risikoeintrittswahrscheinlichkeit senkt

1. Inkrementorientierte Entwicklung führt zu realistischen Zielen und Sollwerten

Gescheiterte Projekte zeigen sich normalerweise an einer nicht zu tolerierenden Abweichung von Zielen des magischen Projektdreiecks (Leistungsumfang/Qualität, Termin, kalkulierte Kosten). Letztlich ist es für einen Projektfehlschlag oft gar nicht mehr feststellbar, ob

- die Ursache in einer unrealistischen Schätzung trotz guter Projektarbeit liegt oder
- eine realistische Schätzung vorliegt und die Projektausführung schlecht ist bzw. auf unvorhersehbare Hindernisse stößt.

Der geringere Funktionsumfang für eine Inkrementschätzung ermöglicht eine realistische(re) Schätzung. Von großer Bedeutung ist auch, dass sich die Schätzung für einen begrenzten Leistungsumfang/Planungshorizont wesentlich schlechter für politische Zwecke instrumentalisieren lässt (Reinhard P. Oechtering, "Das Mautprojekt: Ein Stück aus dem Toll-Haus oder die Eigendynamik unrealistischer Terminpläne").

2. Frühzeitige Verifikation und Validierung der Anforderungen senken den Nachbearbeitungsaufwand

Durch die laufende Auslieferung von Inkrementen an den zukünftigen Anwender der Software werden die Fehler der Software frühzeitig entdeckt – ob sie nun auf einer Fehlinterpretation der Anwenderspezifikation beruhen (Verifikation) oder ob der Anwender feststellt, dass er, auch wenn die Spezifikation erfüllt ist, dennoch andere Funktionen benötigt (Validierung). Weil die Fehler frühzeitig entdeckt werden, sind Korrekturen wesentlich weniger aufwändig im Vergleich zum Wasserfallmodell, wo eine einmalige Auslieferung am Ende des Konstruktionsprozesses das Durchlaufen aller vorgelagerten Phasen erfordert. Das ständige Feedback spätestens nach einer Inkrementauslieferung wirkt auch einem schleichendem oder politisch gewollten Bedeutungswandel der Anforderungen entgegen, so dass teure Missverständnisse frühzeitig entdeckt werden können.

3. Effektivere Fortschrittmessung bei geringerem Administrations- und Controllingaufwand

Der Aufwand für das Projektcontrolling nimmt tendenziell mit der Höhe des zu verausgabenden Budget (Verlustrisiko) und langen Entwicklungszyklen (Wasserfallmodell) zu. Dies liegt unter anderem darin begründet, dass gerade in der Softwareentwicklung bei langen Entwicklungszyklen virtuelle Zwischenprodukte (Dokumente) definiert und bewertet werden müssen, da sonst der Projektfortschritt nicht messbar ist. Eine solche Bewertung, z.B. von Entwicklungsspezifikationen, ist nicht nur aufwändiger, sondern auch wesentlich weniger aussagefähig und somit risikobehafteter als die Bewertung einer testbaren Software.

4. Vermeidung von Fehlentwicklungen aufgrund von Umwelt- und Strategieänderungen

In einer sich immer schneller ändernder Umwelt müssen Entwicklungsziele laufend überprüft werden. Die inkrementorientierte Entwicklung bietet mit jeder Auslieferung eines Inkrements einen natürlichen Checkpunkt, an dem die Entwicklungsrichtung hinsichtlich des zu Beginn angepeilten Entwicklungsziels überprüft wird. Hat sich das Finalziel geändert, kann schon die Funktionalität für das nächste Inkrement auf das neue Finalziel justiert werden. Teure Fehlentwicklungen lassen sich so frühzeitig und mit geringem Aufwand korrigieren.

5. Mehr Vertrauen zwischen Auftraggeber und Auftragnehmer

Die Motivation und das Vertrauen wird durch eine kontinuierliche Auslieferung von Inkrementen und das damit verbundene Feedback sowohl auf Auftraggeber- als auch Auftragnehmerseite gestärkt. Durch die

laufende Präsentation von testbaren Ergebnissen wird Fehlinterpretationen und Spekulationen weitgehend der Nährboden entzogen. Dies stärkt die Kooperation und die Konzentration auf die sachliche Arbeit.

6. Bessere Qualität bzw. mehr Innovation durch verstärkte interdisziplinäre Zusammenarbeit auf Basis konkreter Ergebnisse

Die inkrementorientierte Entwicklung erfordert eine engere Zusammenarbeit zwischen Anwendern und Entwicklern. Hierdurch verstehen Anwender die Möglichkeiten der Software besser, und Entwickler bekommen ein tieferes Verständnis der zu lösenden Anwenderprobleme. Die Diskussion kann auf Basis ausgelieferter Inkremente (testbarer Software) wesentlich konkreter geführt werden als auf abstrakt definierten Anforderungen.

7. Kontinuierliches Realisierungsfeedback vermindert das Risiko von Realisierungsproblemen

Das Entwicklungsteam hat während der Realisierung kontinuierlich Zugriff auf die Erfahrungen mit den Problemen und Herausforderungen der Realisierung. Technisch nicht realisierbare Vorstellungen können möglicherweise frühzeitig durch alternative Lösungen im Einvernehmen mit den Anwendern umgangen werden.

Wie Inkrementierung die Risikofolgekosten begrenzt

Die inkrementorientierte Vorgehensweise mit einer risikogeleiteten Inkremententwicklung, bei der die größten Risiken unter Abwägung der geforderten Anwendungsfunktionalität in dem oder den ersten Inkrement(en) adressiert werden, führt dazu, dass auch die Risikofolgekosten niedrig gehalten werden, die in Form einer Fehlinvestition bei Eintritt des Risikos entstehen. Im Idealfall bildet jedes Inkrement eine funktionale Einheit, deren Nutzen vom Anwender höher bewertet wird als die Kosten. Falls es dann zu einem Abbruch der Inkremententwicklung kommt, weil zum Beispiel

- ein technischer Umstand die sinnvolle Weiterentwicklung verhindert,
- keine Nutzensteigerung durch verfeinerte oder erweiterte Funktionalität erzielt werden kann,
- eine Strategieänderung die Weiterentwicklung nicht sinnvoll erscheinen lässt,
- die Opportunitätskosten bei beschränktem Budget zu hoch sind,

müssen nur die Kosten für das letzte Inkrement abgeschrieben werden (siehe Bild 5). Kann die Anwendung ihre Erträge nur einspielen, wenn das Finalziel erreicht wird, so kann es auch bei einer inkrementalen Vorgehensweise zur Fehlinvestition über die gesamten Inkrementkosten kommen. Allerdings ist die Wahrscheinlichkeit dafür sehr viel geringer, da ja testbare Inkremente frühzeitig ausgeliefert werden und dadurch die Risiken zur Erreichung des Finalziels der Anwendung laufend neu bewertet und geeignete Maßnahmen eingeleitet werden können.

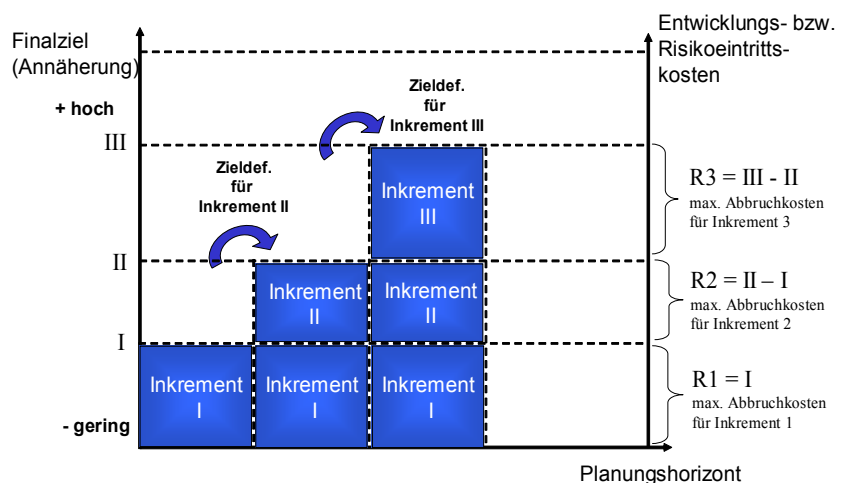


Bild 5: Idealfall der Verlustbegrenzung: eine Inkrementhöhe.

Inkrementorientierung als Mittel gegen Planungssillusionen

Um die Früchte der inkrementalen Entwicklung in vollem Umfang zu ernten, darf das Leistungsversprechen nur für das jeweils nächste Inkrement und nicht für das möglicherweise anvisierte Finalziel gegeben werden. Auftraggeber, die für ein großes Projekt mit dem Gedanken spielen, mit dem Auftragnehmer einen Festpreisvertrag auf Basis der Finalzielbeschreibung abzuschließen, sollten sich noch einmal an sämtliche Voraussetzungen bzw. Implikationen eines solchen Vertrags erinnern:

- Die Anforderungen (Finalziel) müssen klar und eindeutig beschrieben werden können.

- Der Auftragnehmer hat die Fähigkeiten zur Erledigung der Aufgabe.
- Der Auftragnehmer ist in einer wirtschaftlich stabilen Lage.
- Das zu erstellende Objekt ist (in Grenzen) gut kalkulierbar.
- Der Auftraggeber kann seinen vertraglich definierten Mitwirkungsverpflichtungen nachkommen.
- Gutes Vertragsmanagement ist gewährleistet.

Probleme des Festpreisvertrags bei umfangreichen Projekten

Das schwerwiegendste Problem für größere Softwareprojekte liegt darin, dass es nahezu unmöglich ist, die Anforderungen an das Endprodukt vollständig und so eindeutig zu definieren, dass der Interpretationsspielraum bei den Beteiligten möglichst gering ist. Wenn man nun noch berücksichtigt, dass sich die zum Zeitpunkt des Vertragsabschlusses fixierten Rahmenbedingungen im Projektverlauf wahrscheinlich ändern und das Projektteam kontinuierlich seine Fach- und Realisierungskompetenz erweitert und somit zu neuen Einsichten kommt, wird der Vertrag mit hoher Wahrscheinlichkeit bei länger laufenden Projekten zu einem realitätsfernen Korsett mutieren. Um dies zu verhindern, muss der Vertrag laufend mit der Realität synchronisiert und aktualisiert werden – hierzu bedienen sich Auftraggeber und gegebenenfalls auch Auftragnehmer des so genannten Änderungsverlangens (Change Request), das nicht nur in aller Regel kostenpflichtig und terminwirksam ist, sondern auch noch weitere Nachteile im Vergleich zu einem inkrementorientierten Vorgehen in sich birgt:

- Lange Wartezeit und hoher Administrationsaufwand, bis die notwendigen Genehmigungen vorliegen, die häufig aufwändig und mit großem Formalismus vorbereitet, bewertet und nachgehalten werden müssen.
- Die intuitive Gleichsetzung, dass Planabweichungen Probleme anzeigen, führt oft zu ausuferndem Rechtfertigungszwang.
- Politische Gründe: Entscheidungsträger möchten nicht hinter ein "öffentlich" gemachtes Versprechen zurückfallen und verhindern die Genehmigung von Change Requests, die zur Reduzierung ursprünglich geforderter, aber inzwischen überholter Leistungsmerkmale dienen.

Ein zugesagter auf das Finalziel bezogener Festpreisvertrag hat daher für große Projekte oft den Charakter einer Planungssillusion, der man sich scheinbar, wie die Negativschlagzeilen über gescheiterte Projekte zeigen, immer wieder entgegen aller Vernunft hingibt. Treten schwerwiegende Probleme bei der Abwicklung von Projekten auf Festpreisbasis zu Tage, so tragen in der Regel Auftraggeber und Auftragnehmer den Schaden, denn gerichtliche Auseinandersetzungen erzeugen keine funktionstüchtige Software, aber ziemlich sicher enttäuscht reagierende Stakeholder. Eine inkrementorientierte Softwareentwicklung verzichtet also im Vorhinein auf fixe am Finalziel orientierte Planungszusagen, da solche Zusagen für größere Projekte als unseriös betrachtet werden.

In der Tat verliert die aufwändige, zeitraubende und möglichst detaillierte Ausspezifizierung eines Vertrags bei einer inkrementorientierten Vorgehensweise an Bedeutung. Die inkrementorientierte Vorgehensweise beruht auf einer kooperativen Arbeitsweise. Da die Auftragsvergabe nur für das jeweils nächste Inkrement erteilt wird, besteht für den Auftragnehmer ein permanenter Anreiz zur Erzeugung guter Ergebnisse, um einen potenziellen Folgeauftrag für das nächste Inkrement zu erhalten. Einer möglichen Preiserhöhung durch den Auftragnehmer für die notwendige Entwicklung eines Folge-Inkrementes kann der Auftraggeber effektiv entgegenwirken, wenn er die anzuwendenden Entgeltsätze innerhalb eines Rahmenvertrags zu Beginn der Entwicklungsarbeiten festlegt.

Voraussetzungen für den erfolgreichen Einsatz inkrementorientierter Vorgehensmodelle

Das inkrementorientierte Vorgehensmodell kann nur zum Zuge kommen, sofern sich das gewünschte Softwareprodukt überhaupt in auslieferbare Inkremente mit einem eigenem Anwendernutzen zerlegen lässt. Einige Integrations- und Migrationsprojekte im Infrastrukturbereich mit vielen Applikationsschnittstellen können erst einen Nutzen erbringen, wenn sie als Ganzes in Betrieb genommen werden können. Bedingt durch Fusionen und Outsourcing gibt es derzeit besonders im Finanzdienstleistungsbereich viele

Infrastruktur- und Integrationsprojekte, die sich nur mit hybriden oder den herkömmlichen spezifikationsgetriebenen Vorgehensmodellen entwickeln lassen.

Ein weiteres Problem für das inkrementale Vorgehen sind Anwendungen, die auf das Fundament einer ausgefeilten Architektur angewiesen sind, wie zum Beispiel ein weltweites Buchungssystem für Fluglinien. Entwickelt man derart architekturensensible Anwendungen, ist das Risiko groß, dass beim n-ten Inkrement die Entwicklungsarbeit stecken bleibt, weil die Architektur den steigenden Anforderungen nicht standhält. Eine Alternative kann ein hybrides Vorgehensmodell sein, das die phasenorientierte Architekturentwicklung vorsieht und das Konzept über Prototypen testet, während die Anwenderfunktionen inkremental hinzugefügt werden.

Als letzte Voraussetzung sei hier noch eine weitgehend automatisierte Testunterstützung genannt, da alle Iterationen und Inkrementintegrationen einen vollständigen Durchlauf aller Tests notwendig machen.

Zusammenfassung

Neben historischen Gründen, unflexiblen Administrationsprozessen und Gewohnheiten ist vor allem das ungerechtfertigte hohe Vertrauen in Verträge (scheinbare Planungssicherheit durch Festpreisverträge) dafür verantwortlich, dass das Wasserfallmodell noch die Landschaft der Vorgehensmodelle beherrscht (Reinhard P. Oechtering, "Trügerischer Schutz: Festpreise für Software-Entwicklungsprojekte"). Dies betrifft sowohl Auftragnehmer, die lieber aus Umsatz- bzw. Auslastungsgründen den ganzen Kuchen unter Inkaufnahme höchster Risiken akquirieren wollen, als sich zunächst nur mit einem Stück zu begnügen, als auch Auftraggeber, die fälschlicherweise meinen, die Verantwortung und das Risiko mit einem Vertrag auf den Auftragnehmer transferiert zu haben.

Die Möglichkeiten, die Entwicklungszyklen ganz pragmatisch über Inkrementbildungen zu verkürzen, werden nach meiner Auffassung von den Projektverantwortlichen vor allem auf der Auftraggeber- –aber auch auf der Auftragnehmerseite zu wenig wahrgenommen. Hier besteht erheblicher Nachholbedarf, wenn man die Projektrisiken nachhaltig senken und Innovationschancen ergreifen will: Es kann gar nicht genügend Kreativität aufgewendet werden, ein offensichtlich lang laufendes Großprojekt in eine inkrementorientierte Vorgehensweise mit kürzeren Entwicklungszyklen zu überführen. Dabei ist es entsprechend den obigen Ausführungen die wahre Kunst, nicht nur einen Stufenrealisierungsplan zu entwickeln, sondern das vertragliche Leistungsversprechen nicht über das jeweils nächste Inkrement hinaus zu verhandeln und der scheinbaren Planungssicherheit eine Absage zu erteilen.

An dieser Stelle sei Kent Beck, ein Vertreter der agilen Methoden, aus seinem Manifest zum Extreme Programming zitiert: "Jedes Projekt, an dem ich mitarbeitete und das einen festen Preis und einen festen Umfang hatte, endete damit, dass beide Parteien behaupteten: Die Anforderungen waren nicht klar".

Literatur

- Reinhard P. Oechtering, "Best Practice im Projektmanagement", Tagungsband 21. Internationales Deutsches Projektmanagement Forum 2004, S. 167ff)
- Colin J. Neill, Phillip A. Laplante, "Requirements Engineering: The State of the Practice", IEEE Software, Nov. / Dez. 2003, (siehe auch Fraunhofer Umfrage: PM bei der Entwicklung kritischer Softwaresysteme, in Projektmanagement aktuell, 2/2004 S. 33ff)
- Jens Coldewey, "Beständig ist nur der Wandel – Agile Entwicklung und Änderbarkeit", Objektspektrum 6/2001
- T. Capers Jones, "Estimating Software Costs", McGraw-Hill, 1998, S. 113 ff
- T. Capers Jones, "By Popular Demand: Software Estimating Rules of Thumb", IEEE Computer, Vol. 29, March 1996
- Reinhard P. Oechtering, "Das Mautprojekt: Ein Stück aus dem Toll-Haus oder die Eigendynamik unrealistischer Terminpläne", Projekt Magazin 22/2003
- Reinhard P. Oechtering, "Trügerischer Schutz: Festpreise für Software-Entwicklungsprojekte", Projekt Magazin 6/2003
- Kent Beck, "Extreme Programming, Das Manifest", Addison Wesley 2000, S. 159

Mehr zu diesem Thema in unserer Rubrik Firmen & Produkte

[EDV- und IT-Beratung](#) ►►