

FOM Fachhochschule für Oekonomie & Management Berlin
Studiengang Wirtschaft, 5. Semester

Seminararbeit im Schwerpunkt Controlling

CONTROLLING VON SOFTWAREPROJEKTEN

Autor: Hendrik Saly

Betreuer: Prof. Dr. J. N. Stelling
Abgabe: 13.01.2007

Inhaltsverzeichnis

1	Einleitung	1
1.1	Softwareprojekte und ihre Problematik	1
1.2	Thematische Einordnung und Abgrenzung	3
2	Planung von Softwareprojekten	4
2.1	Ablauforganisation	4
2.2	Aufwandsschätzung	6
2.2.1	Function Point	8
2.2.2	SLIM	9
2.3	Termin- und Aufgabenplanung	10
2.4	Kostenplanung	12
3	Kontrolle von Softwareprojekten	14
3.1	Soll/Ist Vergleich	15
3.2	Kennzahlen in der Praxis	16
4	Zusammenfassung und Fazit	18
A	Literaturverzeichnis	19

Abkürzungsverzeichnis

COCOMO	Constructive Cost Model (nach Boehm)
FP	Function Point
LoC	Lines of Code
PT/PeT	Personentage

Abbildungsverzeichnis

1	Inhalt einer Iteration (Quelle: selbst erstellt)	5
2	Story-Card (Quelle: [Jef06])	6
3	Releaseplan (Quelle: Projekt)	11
4	Aufwand-Budget-Diagramm (Quelle: Projekt)	16
5	Feature-Burndown (Quelle: Vgl. [WRL05, S. 203])	17
6	Feature-Burnup (Quelle: Vgl. [WRL05, S. 204])	17

1 Einleitung

In dieser Seminararbeit sollen aus Sicht des Controllings (mit Schwerpunkt Planung) die speziellen Eigenheiten von Softwareprojekten aufgezeigt werden. Die vorliegende Arbeit greift als Ergänzung zu den theoretischen Betrachtungen auf Erfahrungen zurück, die der Autor in den Jahren 2003-2006 in einem großen Softwareprojekt bei einem Berliner Versorgerbetrieb gemacht hat. Der Projektumfang lag hier bei über 5000 Personentagen und einem Budget von mehreren Millionen Euro.

Das Projekt wurde dabei, aus Projektmanagementperspektive betrachtet, was die Ablauforganisationen (Methode im Sinne eines Vorgehensmodells) betrifft nach agilen oder auch inkrementellen Grundsätzen, in Anlehnung an das Spiralmodell¹, durchgeführt. Dieses Vorgehen hat besondere Auswirkungen auf den Planungs- und Kontrollprozess. In den nachfolgenden Kapiteln sollen diese Auswirkungen näher betrachtet werden.

1.1 Softwareprojekte und ihre Problematik

„Dem Chaos Report der Standish Group aus dem Jahr 2004 zufolge, verläuft etwa nur jedes dritte Softwareprojekt erfolgreich. Die Studie besagt weiterhin, dass etwa 70% der Software-Projekte nicht die ursprünglichen Projektziele erreichen, obwohl sie weder das Budget überschreiten, noch in Terminverzug geraten oder Abstriche in der Qualitätssicherung machen“².

Was ist die Ursache dafür, dass gerade Softwareprojekte immer wieder scheitern? Um auf diese Frage eine Antwort geben zu können, muss zunächst einmal definiert werden was genau unter einem Softwareprojekt verstanden werden soll und was die Ziele eines solchen Unterfangens sind.

Ein Softwareprojekt ist zunächst einmal ein Projekt. Die Definition für ein Projekt findet sich in der DIN 69901: „Unter einem Projekt wird allgemein eine einmalige, komplexe, zeitlich befristete Aufgabe verstanden.“ Die Einmaligkeit des Vorhabens wird durch die

- Definition der Projektziele
- zeitliche, personelle und finanzielle Restriktionen,
- Separationsmöglichkeit gegenüber anderen Vorhaben und
- der Notwendigkeit einer spezifischen Organisation

gekennzeichnet³.

¹ Vgl. [Fey04, S. 17]

² [o.V06]

³ Vgl. [Ste03, S. 175]

Projekte sind Vorhaben, die im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet sind⁴.

Ein Softwareprojekt ist nun zunächst ein Projekt im obigen Sinne, dessen Hauptzweck die Erstellung, Auswahl, Definition, Einführung, Ablösung oder Migration eines Softwaresystems ist. Im Rahmen dieser Arbeit wird unter einem Softwareprojekt die Erstellung individueller Softwarelösungen im Großkundenbereich verstanden.

Um am Ende des Projektes eine Bewertung des Erfolges vornehmen zu können, muss das Projektziel definiert werden: „Ziel eines jeden Projektes ist es, in einem definierten Zeitraum mit einem definierten Budget eine vom Kunden definierte Leistung zu erbringen“⁵. Das problematische an dieser Definition, nicht nur im Hinblick auf Softwareprojekte, ist die Präzisierung der Leistung, die erbracht werden soll. Bei der Entwicklung komplexer Softwaresysteme hat sich gezeigt, dass

- die zu erbringende Gesamtleistung⁶ nicht von Anfang an bekannt ist, sondern sich erst im Projektverlauf ergibt,
- der Kunde in frühen Projektphasen die eigentliche Aufgabe nicht genau beschreiben kann,
- durch exogene Einflüsse, z.B. politische Entscheidung, mitten im Projekt komplett umgedacht werden muss.

Aus diesen Gründen ist Anforderungsanalyse und die Art der Kommunikation mit dem Kunden von entscheidender Bedeutung für den erfolgreichen Projektverlauf. Da sich die Anforderungen beständig verändern, kommt der Planung eine besonderer Rolle zu. Es ist im Rahmen von Softwareprojekten nicht möglich, die Anforderungen in ihrer Gesamtheit und Vollständigkeit am Anfang zu definieren und dann im weiteren Projektverlauf rückkopplungsfrei umzusetzen.

Um diese Probleme zu lösen wurden in den letzten Jahren speziell für Softwarevorhaben neue Projektablauforganisationen⁷ geschaffen. Alle diese neuen Methoden haben die Gemeinsamkeit, dass sie ein schrittweises (auch iteratives oder zyklisches) Vorgehen empfehlen um elegant und flexibel (agil) auf Änderungen reagieren zu können.

⁴ Vgl. [PR04, S. 18]

⁵ [GJS03, S. 17]

⁶ Im Sinne von Gesamtfunktionalität/-umfang der Software

⁷ Projektmethoden

1.2 Thematische Einordnung und Abgrenzung

Im Rahmen dieser Arbeit wird nur der Planungs- sowie der Kontrollprozess unmittelbar vor Projektstart und während der Laufzeit des Softwareprojektes (Realisationsphase) betrachtet. Frühere oder spätere Phasen (wie z.B. Nachkalkulationen) werden nicht betrachtet. Aspekte des Projektmanagements werden, soweit sie aus Controllingsicht nicht relevant sind, nicht behandelt. Unter einem Softwareprojekt wird im Kontext dieser Arbeit ein externes Festpreisprojekt verstanden, dessen Ziel die Entwicklung kaufmännischer Individualsoftware für Großkunden ist. Dabei finden die gesamten Tätigkeiten vor Ort beim Kunden statt. Die Gesamtanzahl aller projektbeteiligten Personen liegt in der Regel zwischen fünfzehn und dreißig. Der Schwerpunkt liegt auf der Planung von Softwareprojekten. Nicht näher betrachtet werden ferner Qualitätsaspekte wie z.B. das Testen von Software.

2 Planung von Softwareprojekten

2.1 Ablauforganisation

Um der im Kapitel 1.1 erläuterten Problematik bei der Erstellung von Software gerecht zu werden, entstanden in den letzten Jahren eine Reihe neuer Projektmethoden speziell für die Durchführung von Softwareprojekten. Neben der bekanntesten und im Folgenden detailliert beschriebenen Methode „eXtreme Programming“ existieren noch Scrum, Feature Driven Development (FDD), den Eclipse-Entwicklungsprozess, Industrial XP (IXP), XP Version 2 und das V-Modell XT⁸. Dabei basieren⁹ alle diese agilen Methoden letztendlich auf dem Spiralmodell. Das Spiralmodell entstand aus der Kritik am Phasenmodell (auch Wasserfallmodell) die darin begründet liegt, dass der Kunde das Softwareprodukt erst nach der kompletten Fertigstellung erstmals präsentiert bekommt und dann häufig feststellt, dass etwas anderes realisiert wurde als er ursprünglich haben wollte.¹⁰

Am Ende des Projektes können jedoch kaum noch Korrekturen vorgenommen werden. Somit scheitern dann technisch sogar oft einwandfreie Produkte daran, dass die Anforderungen des Kunden missverstanden wurden und folglich etwas realisiert wurde, was eigentlich nicht gebraucht wird. Diese Problematik ist dem Umstand geschuldet, dass es nicht möglich ist ein komplexes Softwaresystem am Anfang vollständig zu spezifizieren. Hinzu kommen Kommunikationsprobleme zwischen Kunde und Softwareentwickler.

Der Kunde ist Träger des fachlichen Know-Hows, besitzt aber of kein technisches Wissen. Umgekehrt verhält es sich beim Softwareentwickler. Es bedarf also bestimmter Praktiken oder Artefakte, die die Kommunikation hier erleichtern. Diese Artefakte sind Bestandteil der agilen Methoden. Die zwei bekanntesten Beispiele sind Prototypen und Story-Cards.

Das Problem welches hier geschildert wird, ist dass der Anforderungsanalyse (auch Anforderungsermittlung). Das Ziel ist, dass Entwickler und Kunde (Anwender) übereinkommen, welche fachliche Funktionalität auf welche Art und Weise vom zu erstellenden Softwaresystem zur Verfügung gestellt werden soll. Dabei zeigt sich häufig, dass die Anwender diese Fragen zunächst selbst nicht vollständig beantworten können. Dies ist insbesondere der Fall, wenn Softwareprogramme vollständig neu entwickelt werden und es kein Altsystem gibt an dem eine Orientierung möglich wäre. Das Problem tritt auch auf, wenn im Unternehmen des Kunden neue Prozesse eingeführt oder neue Produkte hergestellt werden und die Anwender in den Fachbereichen mit den Neuerungen noch nicht vertraut sind.

⁸ Vgl. [WRL05, S. 133-159]

⁹ Vgl. [Küt05, S. 216]

¹⁰ Vgl. [Fey04, S. 17]

Bei der Art Softwareprojekten die im Rahmen dieser Schrift betrachtet werden (vgl. Kapitel 1.2), besteht eine enge Kopplung zu den Geschäftsprozessen des Kunden. Sind die Prozesse, die in der Softwarelösung abgebildet werden sollen nicht oder nur lückenhaft bekannt, so muss auf die Anforderungsermittlung besonderen Wert gelegt werden. Praktiken hierfür sind unter anderem das offene Interview und Arbeitsplatzbeobachtungen¹¹. Schließlich erfolgt eine Rückkopplung durch den Einsatz des Systems, entweder als (Oberflächen-)Prototyp oder als voll funktionale Lösung im Testbetrieb.

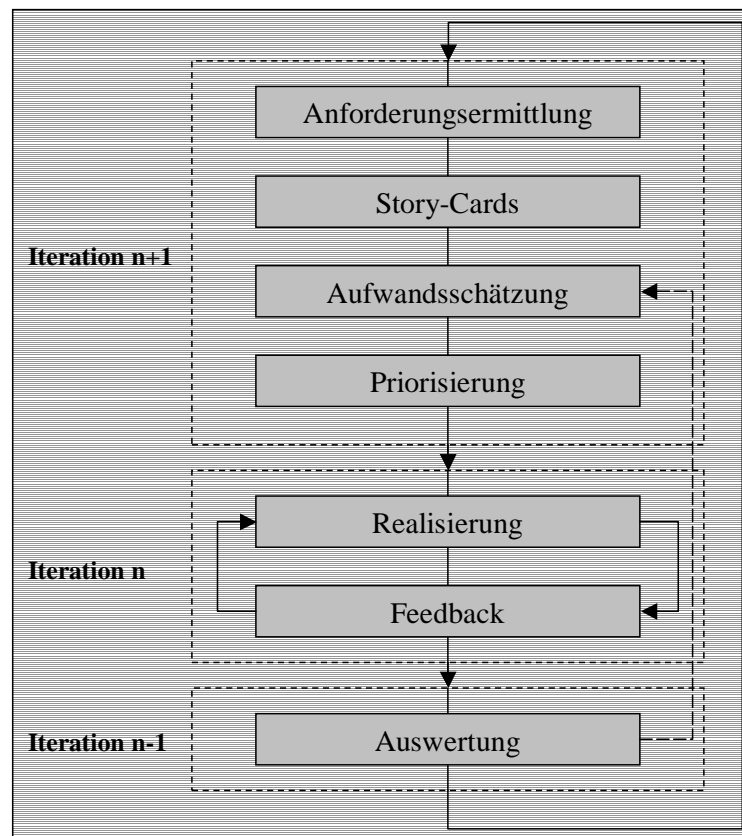


Abbildung 1: Inhalt einer Iteration (Quelle: selbst erstellt)

Ist eine Story-Card vom Entwickler abgearbeitet, so wird sie auf einen Stapel mit den anderen bereits erledigten Karten gelegt und der Entwickler nimmt sich die nächste Karte vor. Der benötigte Realisierungsaufwand (in PT) wird nach Erledigung auf der Karte vermerkt. Eine Story-Card ist nicht notwendigerweise eine wirkliche Karte aus Papier (am besten eignen sich jedoch linierte DIN A5 Karten), sondern kann auch digital als Excel-Datei vorgehalten werden. In der Praxis haben sich echte Papierkarten bewährt, da sie greifbar sind, schnelles Anbringen von Notizen erlauben und automatisch verhindern, dass mehrere Programmierer gleichzeitig an der selben Karte arbeiten.

Abbildung 2 zeigt exemplarisch eine Story-Card. Die genauen Inhalte sind nicht spezifiziert und können von Team zu Team variieren. Angaben über das Datum, den Autor, die

¹¹ Vgl. [WRL05, S. 191-202]

Customer Story and Task Card		BIW Development / COLA	
DATE: 3/19/98	TYPE OF ACTIVITY: NEW: <input checked="" type="checkbox"/> FIX: <input type="checkbox"/> ENHANCE: <input type="checkbox"/> FUNC. TEST: <input type="checkbox"/>		
STORY NUMBER: 1275 / 275	PRIORITY: USER: <input type="checkbox"/> TECH: <input type="checkbox"/>		
PRIOR REFERENCE: _____	RISK: _____ TECH ESTIMATE: _____		
TASK DESCRIPTION: SPLIT COLA: When the COLA rate chgs. in the middle of the BIW Pay Period we will want to pay the 1 ST week of the pay period at the OLD COLA rate and the 2 ND week of the Pay Period at the NEW COLA rate. Should occur automatically based on system design.			
NOTES: on system design. For the OT, we will run a m/frame program that will pay or calc the COLA on the 2 ND week of OT. The plant currently retransmits the hours data for the 2 ND week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA			
TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEEnt Express COLA			
Date	Status	To Do	Comments

Abbildung 2: Story-Card (Quelle: [Jef06])

Beschreibung der Programmieraufgabe, den geschätzten und tatsächlich benötigten Aufwand sowie das Datum der Erledigung sind aber zu empfehlen¹².

Die Projektaufbauorganisation wird im Rahmen dieser Arbeit nicht näher erläutert, da sie aus Controllingsicht von untergeordneter Bedeutung ist.

2.2 Aufwandsschätzung

Der wichtigste Punkt bei der Planung von Softwareprojekten ist die Aufwandsschätzung. Da ein Projekt per Definition¹³ eine Unsicherheit in sich birgt, kann der zu erwartende Aufwand zunächst nur geschätzt werden. Damit diese Schätzung belastbar ist und möglichst dicht an der Realität liegt, werden bestimmte systematische Verfahren angewendet, die nachfolgend kurz vorgestellt werden sollen.

Die grundsätzlichen Erwartungen an eine Schätzmethode können, im Sinne von Maximalforderungen, wie folgt beschrieben werden:¹⁴

- Einheitlich anwendbar: nachvollziehbar, auf unterschiedliche Projekte, in unterschiedlichen Organisationen, technologieunabhängig
- Objektiv: eindeutige Aussagen, Minimum/Maximum Simulation, realitätsnah

¹² Vgl. [WRL05, S. 35 f.]

¹³ Vgl. [PR04, S. 18]

¹⁴ Vgl. [Fey04, S. 138]

- Integrierbar: ins Projektmanagementsystem, früh einsetzbar
- Benutzerfreundlich: nicht aufwändig, einfach in der Anwendung, IT-gestützt

Dabei werden Schätzmethode und Schätzverfahren unterschieden. Die Schätzmethode bilden dabei die Basis für alle Schätzverfahren. Gängige Verfahren sind dabei die folgenden Methoden:¹⁵

- Analogiemethode
Vergleich der aktuellen Aufgabe mit einer ähnlichen oder gleichen bereits gelösten Aufgabe.
- Relationenmethode
Ähnlich der Analogiemethode. Formaler Ansatz durch Bewertung aufwandsverursachender Faktoren.
- Multiplikatormethode
Die ermittelten Aufwände aus der Anwendung der Analogie- oder Relationenmethode für einzelne Projektmodule/-Teile können mit der Multiplikatormethode zusammengefasst und auf der Ebene des Gesamtprojektes aggregiert werden. Dabei wird der Aufwand pro Projektteil mit der Anzahl gleichartiger Projektteile multipliziert.
- Gewichtungsmethode
Auf Basis eines Katalogs von Aufwandsfaktoren wird ein gewichteter Gesamtaufwand errechnet.
- Parametrische Schätzgleichung
Dieser Methode (auch Faktorenverfahren) liegt eine Korrelationsanalyse der relevanten Einflussfaktoren zugrunde.
- Prozentsatzmethode
Auf Basis vergangener Projekte werden hier einzelne Projektabschnitte aufwandsmäßig in Relation gesetzt. Nach Ende eines Abschnitts wird dann eine Hochrechnung des Gesamtaufwands vorgenommen.

Auf der Grundlage der genannten Schätzmethode wurden die Schätzverfahren entwickelt, die bestimmte Vorteile einiger Methoden kombinieren und einen standardisierten Rahmen bilden. Neben dem Function Point-Verfahren und SLIM, welche in den nachfolgenden Kapitel näher betrachtet werden, existieren u.a. noch das Data-Point-Verfahren, COCOMO und das IBM-Verfahren.

¹⁵ Vgl. [Fey04, S. 145]

2.2.1 Function Point

„Function Points sind ein Maß für den Umfang (engl. Size) - aus der Sicht des Nutzers - eines IT-Produktes und des Projektes, welches es entwickelt. Sie sind unabhängig von der Programmiersprache, der Entwicklungsmethode, der Technologie oder der Fähigkeit des Projektteams, welches das IT-Produkt entwickelt“¹⁶.

Das 1979 von Allan Albrecht (IBM) entwickelte Verfahren, ist das zur Zeit erprobteste und genaueste Verfahren. Es wird ständig optimiert und an neue Erfordernisse der Softwareentwicklung angepasst¹⁷.

Die Function Point-Methode beschreibt im engeren Sinne die Leistung bzw. die Funktionalität. Damit ist zumindest unter Anwendung dieser Methode eine konkrete Leistungsbestimmung möglich (vgl. Kapitel 1.1). Ausserdem wird die, aus Controllingsicht intransparente und kaum nachvollziehbare, Expertenschätzung vermieden.

Die FP-Methode funktioniert in ihren Grundzügen wie folgt:¹⁸

Nach einem bestimmten Verfahren (welches untenstehend näher erläutert wird) werden aus Sicht des Benutzers oder Anwenders¹⁹ die Anzahl der FPs²⁰ errechnet, die für die Erstellung des Softwaresystems benötigt werden. Dabei sind die FPs eine zunächst abstrakte Größe, die mittels einer FP-Aufwandstabelle bzw. FP-Erfahrungskurve auf eine konkrete Anzahl von Personentagen abgebildet wird. Diese Aufwandstabelle oder Erfahrungskurve ist je Projektteam unterschiedlich. Sie wird auch als Produktivitätstabelle bezeichnet. Zu Anfang wird mit Standardwerten gearbeitet die dann nach und nach verfeinert werden, je nachdem, wie produktiv das Team arbeitet. Die Kurve ändert sich immer dann, wenn Erfahrung gewonnen oder verloren (z.B. durch Änderung der Teamzusammensetzung) geht.

Produktivität = Produktumfang/Aufwand = FP/PT

Das Verfahren basiert auf den folgenden Schritten:

1. Function Point Zählung durchführen
2. Bestimmung der Geschäftsfunktionstypen
3. Geschäftsentitäten bestimmen

¹⁶ [GJS03, S. 160]

¹⁷ Vgl. [Fey04, S. 153]

¹⁸ Vgl. [GJS03, S. 160 ff.] und [Fey04, S. 152 ff.]

¹⁹ auch Personengruppen oder Organisationseinheiten

²⁰ Eine Alternative zu den FPs wäre eine einfache Zählung der Programmcodezeilen (engl. Lines of Code, LoC) welche aber deutliche Nachteile mit sich bringt²¹. Insbesondere ist hier keine Unabhängigkeit der Technologie bzw. der Programmiersprache gegeben, da die Abbildung einer Aufgabe in Quellcode von Sprache zu Sprache stark variiert.

4. Komplexitätsgrade bestimmen
5. Wertfaktor bestimmen
6. Justierten Function Point Wert bestimmen
7. Interpretation des Ergebnisses

2.2.2 SLIM

Das SLIM-Verfahren basiert auf der parametrischen Schätzgleichung. Es ist ein schnelles, aber dafür deutlich ungenaueres Schätzverfahren als Function Point. Es operiert lediglich mit groben Aufwandsgrößen, die nur den reinen Entwicklungsaufwand abbilden. Der Managementaufwand bleibt unberücksichtigt.

Auf Basis der Informationen aus der Bedarfsanalyse oder dem Sollkonzept wird die gewichtete Summe der Softwarekomponenten ermittelt. Softwarekomponenten können Datenobjekte, Nachrichtenobjekte oder Unterprogramme sein. Aus der Summe der Komponenten läßt sich über die SLIM-Produktivitätstabelle der Nettoaufwand ablesen. Der Nettoaufwand wird dabei in Aufwandsmonaten (AM) angegeben²². Die Produktivitätstabelle entsteht durch lineare Extrapolation, wobei der Aufwand pro 10 Komponenten um 0,9 AM wächst.

Um vom Nettoaufwand auf den Bruttoaufwand schließen zu können, müssen noch weitere Einflussfaktoren berücksichtigt werden. Dies sind einerseits allgemeine Faktoren wie z.B. Hard- bzw. Softwareausstattung und Organisationsstand. Andererseits spezielle Qualitätsfaktoren wie z.B. Zuverlässigkeit, Sicherheit oder Wartungsfreundlichkeit.

Die abgebildeten Tabellen sind als initiale Startwerte zu verstehen. Sie sollten in jedem Unternehmen durch eine Nachkalkulation iterativ mit jedem Projekt verfeinert werden. So kann es im Laufe der Zeit zu deutlichen Produktivitätssteigerungen kommen, wenn z.B. nur geringe Mitarbeiterfluktuation im Unternehmen herrscht und/oder die Mitarbeiter gute Weiterbildungsmöglichkeiten haben.

Bis die Tabelle an unternehmensinterne Gegebenheiten angepasst wurde, sollte noch eine letzte Korrektur am ermittelten Bruttoaufwand vorgenommen werden. Da die Initialtabellen schon etwas älter sind und mittlerweile die Produktivität bei der Entwicklung von Software aufgrund neue Programmiersprachen und -paradigmen erheblich gestiegen ist, wird der Bruttoaufwand durch den 4-GL-Faktor²³ geteilt und verringert sich somit. Dieser 4-GL-Faktor hat den initialen Wert fünf.

²² Ein Aufwandmonat entspricht 160 Stunden

²³ Alte Programmiersprachen wie z.B. COBOL werden als Sprachen der 3. Generation (3-GL, dabei steht GL für Generation Language) bezeichnet und waren lange nicht so produktiv einsetzbar, wie die neuen Sprachen der 4. Generation (4-GL), z.B. Java oder SQL.

Im Projekt des Berliner Versorgerbetriebes wird eine Mischung aus beiden Schätzverfahren verwendet. Damit wird versucht die Genauigkeit von Function Point mit der Leichtigkeit von SLIM zu verknüpfen. Das Function Point Verfahren ist aufgrund seiner zeitintensiven Komplexität, in Reinform, für agile Softwareprojekte nicht geeignet.

2.3 Termin- und Aufgabenplanung

Die Herausforderung bei der Terminplanung in inkrementellen (agilen) Projekten liegt darin, mit zunächst unvollständigen Anforderungen (vgl. Kapitel 1.1) einen Projektplan aufzustellen und die Aufwände zu schätzen. Dieses Problem existiert jedoch nur am Anfang des Projektes und kann im weiteren Verlauf dadurch gelöst werden, dass die Planung ebenfalls inkrementell betrieben wird (kontinuierliche Projektplanung). Die Schätzungen und Planungsaktivitäten für jedes weitere Inkrement basieren auf der Erfahrung des vorhergegangenen Zykluses. So nimmt im Projektverlauf die Planungssicherheit zu.

Das oben beschriebene Dilemma am Projektanfang wird dadurch abgemildert, dass in XP Projekten grundsätzlich nicht zu weit im Detail vorausgeplant wird. Einerseits weil dies der Philosophie des zyklischen Vorgehens widerspricht, andererseits weil umfangreiche Planungen und Schätzungen selbst sehr aufwändig und darüber hinaus im Ergebnis kaum realistisch sind. Statt dessen wird das Projekt hierarchisch aufgegliedert und je nach Detailstufe fein- oder grobgranular geschätzt. Die kleinste Ebene stellen die Story-Cards dar, die einzelne Funktionen des Systems beschreiben. Diese werden im engen Zusammenspiel mit dem Kunden iterativ entwickelt und sind daher am Projektanfang nicht vollständig bekannt. Sie können aber zu sogenannten Features (oder Business-Stories) zusammengefasst werden. Eine Zusammenfassung von Features wird als Subsystem bezeichnet.

Es hat sich in der Praxis als sinnvoll herausgestellt, einen festen Iterationszyklus von zwei Wochen zu wählen. Wie im Kapitel 2 erläutert, finden also die Planungsaktivitäten Anforderungsermittlung, Schreiben der Story-Cards, Aufwandsschätzung und Priorisierung in diesem Abstand statt. Es werden immer nur so viele Anforderungen ermittelt, wie in den nächsten zwei Wochen auch realisiert werden können. In der Regel befindet sich ein XP-Projekt, was die Anforderungsermittlung betrifft, zwei bis drei Iterationen im Vorlauf. Dies ist vor allen Dingen dem Umstand geschuldet, dass die Ermittlung der Anforderungen länger dauern kann als deren Umsetzung. Insbesondere die Analyse komplexer Prozesse, die Zusammenarbeit mit unmotivierten oder falsch gewählten Fachbereichsvertretern oder die Notwendigkeit mehrerer Rückkopplungsdurchgänge (Feedbackschleifen) kann die Arbeit in diesem Punkt erschweren²⁴.

²⁴ Vgl. [WRL05, S. 32 ff.]

Release	Iteration	Fertigstellung
	1 Mitarbeiterverwaltung	KW 10
	2 Mitarbeiterverwaltung	KW 12
	3 Mitarbeiterverwaltung	KW 14
	4 Anwesenheitsplaner	KW 16
	5 Anwesenheitsplaner	KW 18
	6 Fahrzeugverwaltung	KW 20
	7 Fahrzeugverwaltung	KW 22
	8 Einsatzplanung	KW 24
	9 Einsatzplanung	KW 26
<u>1</u> Einsatzplaner für Bereich 1	10 Einsatzplanung	KW 28
	11 Vertretungsregelung	KW 30
	12 Vorgangsverwaltung	KW 32
<u>2</u> Vorgangsverwalter für Bereich 2	13 Vorgangsverwaltung	KW 34
	14 SAP-Anbindung	KW 36
	15 SAP-Anbindung	KW 38
	16 SAP-Anbindung	KW 40
	17 SAP-Anbindung	KW 42
	18 SAP-Anbindung	KW 44
<u>3</u> SAP-Anbindung Vorgangsverwalter für Bereich 2	19 SAP-Anbindung	KW 46

Abbildung 3: Releaseplan (Quelle: Projekt)

Das Ergebnis einer jeden Iterationsplanung sind neben den neu entstandenen Story-Cards (inklusive Aufwandsschätzung) ein Releaseplan, der einen Gesamtüberblick über das Projekt gibt. In Abbildung 3 ist ein solcher Plan exemplarisch dargestellt.

Ein Release ist die Auslieferung von Software(-komponenten) und deren Inbetriebnahme. Eine bestimmte Anzahl von einzelnen Iterationen ergeben ein Release. Ein Release ist dabei im klassischen Projektmanagement meist auch ein Meilenstein.

2.4 Kostenplanung

Um sinnvoll Kostenplanung betreiben zu können, muss zunächst untersucht werden, wie die Verträge mit dem Auftraggeber gestaltet werden können. Für Softwareentwicklungsprojekte, wie sie im Rahmen der vorliegenden Arbeit betrachtet werden, existieren eine Reihe unterschiedlicher Vertragsmodelle²⁵. Die am häufigsten in der Praxis vorkommenden sind:

- Vergütung nach Aufwand
Die Bezahlung erfolgt monatlich nach vereinbartem Tagessatz und erbrachten Aufwand. Für den Kunden ist die flexible Steuerungsmöglichkeit ein Vorteil, gleichzeitig liegt die gesamte Verantwortung und das Schätzrisiko bei ihm. Für den Hersteller besteht kaum ein Risiko aber auch keine Planungssicherheit, da der Kunde das Projekt jederzeit beenden kann.
- Budget mit fixierter Funktionalität
Vor Projektbeginn werden Funktionalität, Endtermine und der Preis fest vereinbart. Die Funktionalität wird in einem Pflichtenheft beschrieben. Der Kunde gibt somit fast alle Risiken an den Hersteller weiter. Anforderungsänderungen während der Laufzeit können aber nicht mehr berücksichtigt werden. Fehler im Pflichtenheft gehen ebenfalls zu Lasten des Auftraggebers. Der Hersteller trägt das Schätzrisiko und erhält die Vergütung erst am Projektende. Kann das Projekt mit weniger Aufwand als geplant realisiert werden, so sind für den Hersteller hohe Gewinne möglich.
- Budget mit variablen Inkrementen
Ähnlich wie das Budget mit fixierter Funktionalität. Es können jedoch einzelne Anforderungen im Pflichtenheft ausgetauscht werden²⁶. Ausserdem werden während der Projektlaufzeit schon Softwarekomponenten an den Kunden ausgeliefert und von ihm bezahlt. Für den Hersteller besteht nur geringe Planungssicherheit, da der Auftraggeber das Projekt vorzeitig beenden kann.

Im Folgenden wird nur das Vertragsmodell „Budget mit variablen Inkrementen“ weiter betrachtet.

Die Kostenplanung muss, wie die Terminplanung, im agilen Projekt ebenfalls zyklisch erfolgen. Vor Projektstart wird auf Basis des Pflichtenheftes eine initiale Aufwandsschätzung vorgenommen die als Grundlage einer Basisplanung dient. Weiterhin sollte die unternehmensinterne Kostenstruktur bekannt sein, damit eine Kalkulation für das Preisangebot vorgenommen werden kann. Softwarehersteller (so wie in dieser Arbeit dargestellt)

²⁵ Vgl. [WRL05, S. 32 ff.]

²⁶ Featuretausch

haben fast ausschließlich fixe Kosten. Der größte Kostenblock sind dabei die Personalkosten. Um eine sinnvolle Kalkulation erstellen zu können, ist es notwendig die Tagessätze der beteiligten Mitarbeiter zu kennen, bei der gerade eine Kostendeckung erreicht wird. In der Praxis²⁷ läßt sich als ein Weg folgende Formel nutzen:

$$GK = G * LKS * GKZ \quad (1)$$

$$RAT = \frac{(260 - AF) * AUF}{12} \quad (2)$$

$$TS = \frac{GK}{RAT} \quad (3)$$

GK: Gesamtkosten pro Mitarbeiter G: Bruttogehalt
 LKS: Lohnnebenkostensatz GKZ: Gemeinkostenzuschlag
 RAT: Reale Arbeitstage pro Monat AF: Ausfalltage
 AUF: Auslastungsfaktor TS: kostendeckender Tagessatz

Für ein exemplarisches Gehalt von 2000 EUR und einem Lohnnebensatz von 20% sowie einem unternehmensspezifischen Gemeinkostenzuschlag von 40% errechnen sich aus $GK = 2000 * 1.2 * 1.4$ die Gesamtkosten pro Mitarbeiter auf 3360 EUR. Die realen Arbeitstage pro Monat errechnen sich aus $RAT = \frac{(260-50)*0.8}{12}$ bei 50 Ausfalltagen (Urlaub, Feiertage, Krankheit) und einer geplanten Auslastung von 80%. Sie betragen dann 14 Tage. Aus diesen beiden Werten ergibt sich nach $TS = \frac{3360}{14}$ ein Tagessatz von 240 EUR.

Dieser Tagessatz kann nun für die Festpreisberechnung herangezogen werden. Zur Vereinfachung wird über alle Tagessätze der zum Einsatz kommenden Mitarbeiter der Mittelwert gebildet und mit den geschätzten Aufwandstagen multipliziert. Auf diesen Wert wird dann der Gewinnaufschlag sowie ggf. ein Sicherheitsaufschlag addiert. Die resultierende Größe ist der Preis, den der Kunde für das Projekt bezahlen muss.

Die ermittelten Tagessätze sind auch Grundlage der iterativen Kostenplanung. Diese findet nach jeder Iteration statt. Die Tagessätze werden mit den geschätzten Aufwänden multipliziert und gehen so in die Budgetplanung ein.

²⁷ Gemeint ist hier das Unternehmen in welchem der Autor dieser Schrift tätig ist.

3 Kontrolle von Softwareprojekten

Um an einem bestimmten Stichtag die Kontrolle der geplanten Größen zu ermöglichen ist es notwendig zunächst alle Istdaten zu ermitteln. In agilen Projekten sollte die Kontrolle, genau wie die Planung, iterativ und kontinuierlich durchgeführt werden. In der Praxis hat sich ein wöchentlicher Rhythmus als geeignet erwiesen. In diesem Kapitel wird nicht auf die Kontrolle der Qualität eingegangen. Ebenfalls nicht näher beschrieben werden die Konsequenzanalyse und die Ableitung von Steuerungsmaßnahmen.

Im Rahmen der Kontrolle muss zwischen der Analyse und der Prognose unterschieden werden. Die Analyse gibt Aufschluss über die Vergangenheit. Die Bewegung auf der Zeitachse ist rückwärtsgerichtet. Die Prognose extrapoliert die Ergebnisse der Analyse in die Zukunft und ist somit auf der Zeitachse vorwärtsgerichtet.

Die Erfassung der Istzustände findet in einem agilen Softwareprojekt wie folgt statt:

- Leistung

Der Fortschrittsgrad (FGR) (auch innerhalb einer Iteration²⁸) errechnet sich aus dem Verhältnis der erledigten zu allen Story-Cards, Business-Stories und Subsysteme.²⁹

- Termine

Die Terminalsituation kann mit Hilfe des im Kapitel 2.3 vorgestellten Releaseplans erfasst werden. Darüber hinaus kann in agilen Projekten zusätzlich auch jede andere Art der Terminüberwachung installiert werden. In der Praxis ist die Verwendung von klassischen Balken- und Netzplänen für die grobgranulare Gesamtplanung durchaus üblich.

- Kosten

Die Istkostenerfassung basiert auf der Stundenaufschreibung des Personals. Bei den Softwareentwicklern geschieht dies für produktive Tätigkeiten automatisch auf den Story-Cards. Die Erfassung der Aufwände aller anderen Projektbeteiligten sowie die Erfassung unproduktiver Zeiten der Entwickler erfolgt über Excel-Tabellen. Die Istkosten können anhand der verbrauchten Stunden mal den zugrunde gelegten Tagessätzen (vgl. Kapitel 2.4) errechnet werden.

Im Folgenden wird gezeigt wie diese Istdaten für bestimmte Controllingelemente herangezogen werden können.

²⁸ Eine Iteration kann in der klassischen Projektmanagementterminologie als Arbeitspaket angesehen werden.

²⁹ $FGR = \frac{\text{erledigte Stories}}{\text{erledigte Stories} + \text{offene Stories}} = \frac{\text{erledigte Stories}}{\text{alle Stories}}$

3.1 Soll/Ist Vergleich

Ein Soll/Ist Vergleich der Kosten lässt sich in einem agilen Projekt wie folgt durchführen:

1. Bestimmung des Fortschrittsgrads wie im Kapitel 3 erläutert.
2. Ermittlung der Plankosten

Die Plankosten ergeben sich aus dem geplanten durchschnittlichen Tagessatz mal dem bis zum Stichtag geplanten Aufwand. Hier ist dies zunächst der geschätzte Aufwand angegeben in Aufwandspunkten, der mittels der geplanten Produktivität³⁰ in reale Personentage umgerechnet werden muss.

3. Fertigstellungswert

Die Sollkosten zum Stichtag (= Fertigstellungswert) errechnen sich aus Fortschrittsgrad mal Plankosten.

4. Vergleich

Aus der Subtraktion der Istkosten von den Sollkosten lässt sich nun die Abweichung (zum Stichtag) bestimmen. Je nach Betrag (und Ursache) der Abweichung können dann entsprechende Maßnahmen eingeleitet werden, um die Abweichung zu kompensieren.

Neben dem oben skizzierten Vergleich bietet sich ein weiteres Element für die Kostenkontrolle, das sogenannte Aufwand-Budget-Diagramm³¹ (siehe Abbildung 4), an. In diesem Diagramm wird auf der Abszisse die Zeit und auf der Ordinate die Anzahl PT (auch PeT) aufgetragen. Darin lassen sich nun zwei Kurven einzeichnen, um eine Kosten- bzw. Budgetkontrolle zu ermöglichen. Die Aufwand-Kurve stellt die kumulierten verbrauchten PT, die mittels Stundenaufschreibung ermittelt werden, dar. Die Budget-Kurve trägt das bisher „verdiente Budget“, gemessen in PT, ab. Letzteres entspricht dem kumulierten Aufwand der bereits erledigten Story-Cards.

Folgendes vereinfachtes Beispiel soll diesen Zusammenhang verdeutlichen:

In einem Softwareprojekt wird ein Aufwand von 600 PT geschätzt³². Dem Kunden wird ein Festpreisangebot unterbreitet, welches der Hersteller intern mit 600 PT mal seinem durchschnittlichen Tagessatz von 500 EUR bepreist. Nach einem Gewinnaufschlag von 10% beläuft sich das Angebot an den Kunden auf 330.000 EUR. Die 600 PT verteilen sich auf insgesamt zehn Story-Cards zu je 60 PT. Nach acht Wochen wird mittels Aufwand-Budget-Diagramm festgestellt, dass von den geschätzten 600 PT bisher 300 PT erledigt wurden (untere Kurve). Es wurden also bisher $300 \text{ PT} * 550 \text{ EUR}$ verdient und zum Beispiel über eine Abschlagszahlung auch wirklich vom Auftragnehmer bezahlt. Allerdings

³⁰ In der Praxis wird hier häufig die durchschnittliche Istproduktivität verwendet.

³¹ Dieses Diagramm hat sich aus Projektpraxis im genannten Projekt entwickelt.

³² Auf die Verwendung von Aufwandspunkten wird der Einfachheit halber an dieser Stelle verzichtet.

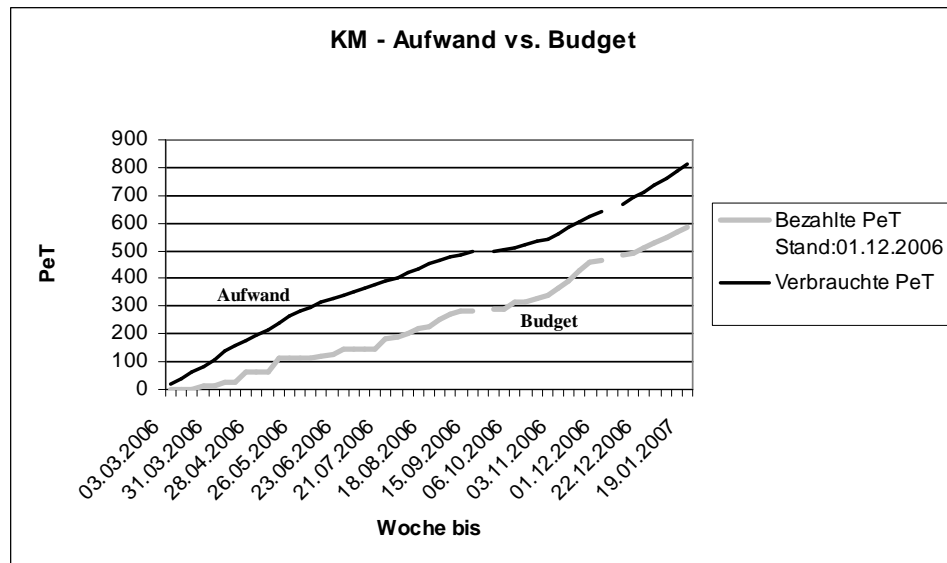


Abbildung 4: Aufwand-Budget-Diagramm (Quelle: Projekt)

zeigt die zweite (obere) Kurve, dass bisher aber 500 PT geleistet wurden. Das Budget ist also zum Betrachtungszeitpunkt um 200 PT überzogen. Nun müssen geeignete Steuerungsmaßnahmen eingeleitet werden, um diese Überziehung zu kompensieren.

3.2 Kennzahlen in der Praxis

In agilen Softwareprojekten wird oftmals das sogenannte Feature-Burndown Diagramm zur grafischen Visualisierung des Restaufwandes verwendet. Dabei ergibt sich der Restaufwand aus den noch offenen Story-Cards, Features oder Subsystemen. Gemessen wird er zunächst in Function Points oder Aufwandspunkten (AP), die aus der jeweiligen Aufwandsschätzung entnommen werden können. Auf der Basis einer aktuellen Produktivitätskurve können dann die Aufwandspunkte in reale Aufwände (Personentage) umgerechnet werden³³.

Anhand der oben dargestellten Abbildung ist erkennbar, wann die gewünschte Funktionalität komplett implementiert sein wird. Die Prognose wird unter der Annahme getroffen, dass die Produktivität konstant bleibt und dass die Schätzungen, aus denen der Restaufwand ermittelt wurde, einen linearen Zusammenhang haben was die Relationen der Aufwandspunkte betrifft. Letzteres bedeutet, dass ein Aufwand der mit drei AP angegeben wurde auch dem dreifachen Aufwand einem mit einem AP geschätzten Aufwand entspricht.

Eine Schwäche des Feature-Burndown Diagramms ist, dass plötzliche Produktivitätsab-

³³ Vgl. [WRL05, S. 202-203]

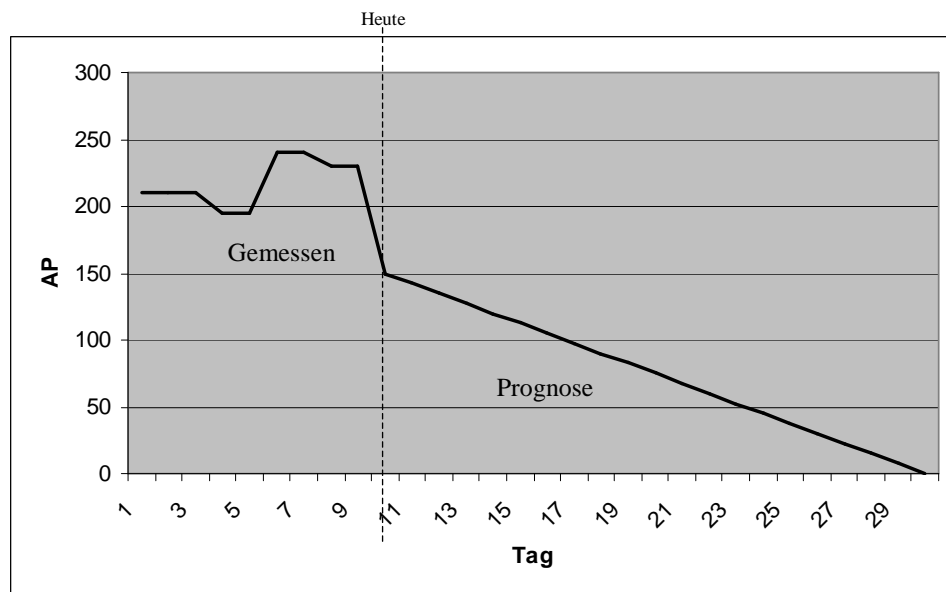


Abbildung 5: Feature-Burndown (Quelle: Vgl. [WRL05, S. 203])

fälle oder -zunahmen nicht ausreichend differenziert werden können. Es ist unklar, ob die Ursache der Produktivitätszunahme am Tag neun im obigen Diagramm auf einer tatsächlichen Effizienzsteigerung beruht oder, ob nicht einfach nur Aufwände gestrichen wurden. Um diese Situation grafisch besser darstellen zu können, wird das Feature-Burndown Diagramm auf den Kopf gestellt und in ein Feature-Burnup Diagramm (siehe Abbildung 6) umgewandelt. Ausserdem wird noch der Gesamtaufwand als Kurve eingetragen³⁴.

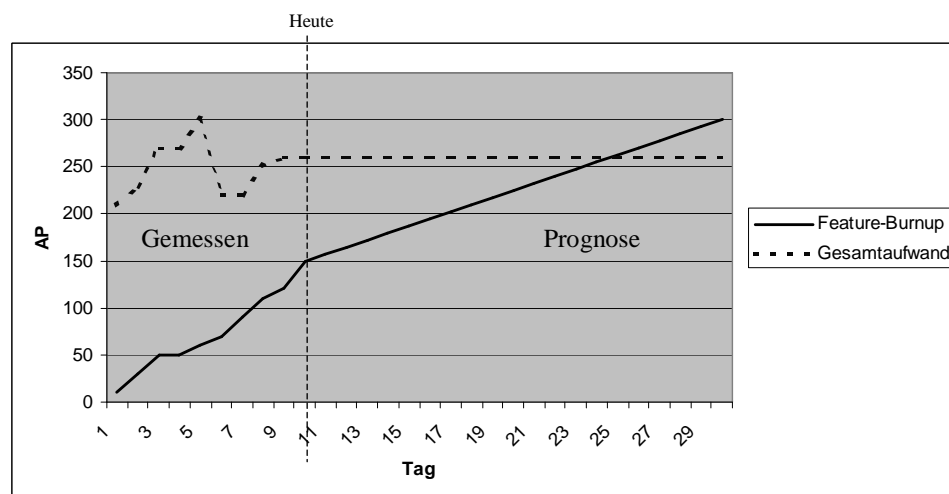


Abbildung 6: Feature-Burnup (Quelle: Vgl. [WRL05, S. 204])

Aus dem Diagramm lässt sich nun das voraussichtliche Projektende als Schnittpunkt beider Kurven auf der Abszisse ablesen.

³⁴ Vgl. [WRL05, S. 203-204]

4 Zusammenfassung und Fazit

Das Controlling von agil (iterativ) durchgeführten Softwareprojekten unterscheidet sich in einigen Punkten vom „normalen“ Projektcontrolling. Es wurde zunächst dargestellt, wo die Problembereiche der Softwareerstellung liegen. Dies ist vor allen Dingen im planerischen Bereich der Fall, da bei Softwareprojekten nicht davon ausgegangen werden kann, dass alle Anforderungen an ein System von vornherein bekannt sind. Dazu kommt, dass das vorherige Schätzen von Aufwänden eine komplexe Angelegenheit ist. Viele Anforderungsdetails und damit verbundene Abhängigkeiten ergeben sich sogar erst mitten im Projektverlauf. Darauf kann mit einer agilen Vorgehensweise reagiert werden. Dies bedeutet, dass im gesamten Projekt iterativ (zyklisch) vorgegangen wird. Das hier betrachtete Projektablaufmodell welches für so ein Vorhaben einen Rahmen bereit stellt, nennt sich „eXtreme Programming“. Allerdings müssen damit auch die Controllingmechanismen der Planung und Kontrolle angepasst werden. So ist es unabdingbar, sowohl die Planung, wie auch die Kontrolle ebenfalls iterativ zu gestalten. Dazu zählt insbesondere die Notwendigkeit Daten aus den vorigen Iterationen zur stetigen Verfeinerung der Genauigkeit weiterer Schätzungen heranzuziehen. Dies kann mittels einer Produktivitätsfunktion erreicht werden. Voraussetzung dafür ist die Einführung einer abstrakten Größe (Function Point bzw. Aufwandspunkt) und somit die Entkopplung von realen Aufwänden. Die Kostenplanung und Kostenkontrolle im Rahmen eines Festpreisprojektes sowie spezielle aber wichtige Kennzahlen aus der Praxis wurden ebenfalls erläutert. Dabei steht die Ermittlung von Kostensätzen sowie ein Soll/Ist Vergleich im Zentrum der Betrachtung.

In agilen Projekten ist das Controlling ein implizit eingebundener Vorgang, der oft als solcher kaum mehr selbständig wahrgenommen wird. Dies resultiert aus dem Umstand heraus, dass praktisch jeden Tag Controllingaktivitäten von fast allen Projektbeteiligten durchgeführt werden. Das Controlling hat also eine hohe Wichtigkeit und ist vor dem Hintergrund des stetig wachsenden Wettbewerbsdrucks und der damit verbundenen Produktivitätserwartung an Softwarelieferanten unverzichtbar.

A Literaturverzeichnis

Literatur

- [Fey04] FEYHL, Achim W.: *Management und Controlling von Softwareprojekten*. 2. Auflage. Wiesbaden : Gabler Verlag, 2004
- [GJS03] GRUNER, Katrin ; JOST, Christian ; SPIEGEL, Frank: *Controlling von Softwareprojekten*. 1. Auflage. Wiesbaden : Friedr. Vieweg & Sohn Verlag/GMV Fachverlage GmbH, 2003
- [Jef06] JEFFRIES, Ron. *Story and Task Cards*.
http://www.xprogramming.com/xpmag/story_and_task_cards.htm. Dezember 2006
- [Küt05] KÜTZ, Martin: *IT-Controlling für die Praxis*. 1. Auflage. Heidelberg : dpunkt.Verlag, 2005
- [o.V06] O.V. *Software Delivery Optimization*.
http://www.borland.com/de/company/software_delivery.html. Dezember 2006
- [PR04] PATZAK, Gerold ; RATTAY, Günter: *Projektmanagement*. 4. Auflage. Wien : Linde Verlag, 2004
- [Ste03] STELLING, Johannes N.: *Kostenmanagement und Controlling*. 2. Auflage. München/Wien : R. Oldenbourg Verlag, 2003
- [WRL05] WOLF, Henning ; ROOCK, Stefan ; LIPPERT, Martin: *eXtreme Programming*. 2. Auflage. Heidelberg : dpunkt.verlag, 2005