Projektführung in Kundenprojekten

Nachlässigkeiten-Irrtümer-Methoden

Hubert Surrer

sd&m AG software design & management
Carl-Wery-Str. 42
81739 München
hubert.surrer@sdm.de

Abstract: In der Vergabe und Abwicklung von Kundenprojekten beobachtet man oft, dass Erkenntnisse aus der Informatik den Weg in die Köpfe der Kunden nicht finden - oder sich sogar als Irrtümer manifestieren.

Gleichzeitig beobachten wir immer wieder Nachlässigkeiten in der Projektführung. Beide Erkenntnisse zusammen bringen uns dazu, bei der Führung von Projekten nach immer mehr Professionalität zu streben.

Der Beitrag vergleicht Paradigmen aus dem Software-Engineering mit Software-Management und skizziert unter den oben genannten Einflüssen, wie man Software-Projekte professionell plant und steuert. Der Beitrag möchte zum Nachdenken anregen und die Fachwelt etwas näher an die IT-Welt rücken.

1 Motivation

Dass die Entwicklung von Individualsoftware ein schwieriges Geschäft ist, ist weiß Gott keine neue Erkenntnis. Nur zur Erinnerung: die CHAOS-Studie der Standish Group [Stan01] bescheinigt uns zwar eine Verbesserung in Termin- und Budgetüberschreitungen in den letzten zehn Jahren, trotzdem scheitern immer noch 23% aller untersuchten Projekte, d. h. diese Projekte werden nie produktiv! Die Ursachen dafür sind vielfältig. Unvollständige Anforderungen, fehlende Anwenderbeteiligung und "moving targets" werden als Top Ursachen genannt. McKinsey hat 1999 die Geheimnisse erfolgreicher IT-Projekte in ihrem Buch „Secrets of Software Success" veröffentlicht. Daran haben in den letzten 10 Jahren auch Werkzeuge, Prozesse und Vorgehensmodelle wenig geändert. Ich erinnere an 4GL-Sprachen, CASE-Tools, das AD/Cycle-Konzept der IBM oder Programmiersprachen wie Smalltalk. Es bleibt dabei: den Stein der Weisen hat die Informatikbranche noch nicht gefunden. Laut Dirk Berensmann [Ber05] hat die IT seit Anfang der Neunziger sogar wieder einen Rückschritt in Richtung „Handwerk" gemacht - ausgelöst durch Technologiehype, New Economy und nachfolgende Budgetstopps.

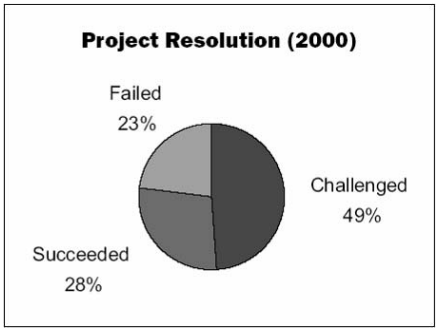


Abbildung 1: Chaos Studie 2001 an der Standish Group (aus [Stan01])

Zwei Beobachtungen scheinen dem zu widersprechen:

IT does not matter [Car04]

Informationstechnologie ist zur commodity geworden, zur Ware also, die man einkaufen kann wie Strom aus der Steckdose. Von einem gewissen Standpunkt aus hat Nicholas Carr sogar Recht: IT ist kein Alleinstellungsmerkmal mehr, sondern „Business enabler". Allein daraus zu schließen, dass jetzt alles ganz einfach wird, dass es keine Budget- oder Terminüberschreitungen mehr gibt, ist eine gefährliche Vereinfachung der Aussage „It is commodity". Wer hat schon von Termin- oder Budgetproblemen bei der Lieferung von Strom gehört? Unser Handwerkzeug wird nicht einfacher, sondern komplexer: SUN veröffentlicht ständig neue Java-Releases, wir werden beglückt mit neuen Versionen und Upgrades von Datenbanken, Middleware, Application-Servern usw. Einfacher wird es dadurch nicht.

Zwang zur Reduktion von Kosten

Märkte werden globaler, Unternehmen stehen im internationalen Wettbewerb, der Zwang zum Sparen wächst und wächst. Kostensparprogramme begegnen uns überall, manchmal haben sie wohlklingende Namen wie CORE, save-for-growth oder First Choice. Die Kostensparprogramme werden intern umgesetzt, schlagen aber auch auf die Lieferanten durch, was sich letztlich am Einkaufsverhalten widerspiegelt.

Wie wird heute IT-Dienstleistung eingekauft:

- *Skill Level soll in großen Unternehmen oftmals Mitarbeiter normieren*

Natürlich ist es erstrebenswert, Preise von Dienstleistern miteinander zu vergleichen. Alle Berater werden in ein Raster so genannter Skill-Level´s gesteckt, die versuchen, eine vermeintliche Normierung herzustellen. Dabei lehrt uns schon DeMarco, [DeMar, Lis91] dass die Performance zwischen dem besten und dem schlechtesten Mitarbeiter einen Faktor von 1:10 betragen kann, um nur einen Aspekt zu nennen, der diese Praxis fragwürdig erscheinen lässt.

- *Internetauktionen als Ersatz zu persönlichen Vertragsverhandlungen*

Nachdem IT ja offensichtlich commodity ist, und Berater über Skill-Level normiert sind, kann man auch den nächsten Schritt tun, und Stundensätze in einer online Auktion verhandeln. So zu sagen "ebay anders herum". Bleiben da die Qualität und Alleinstellungsmerkmale der Lieferanten auf der Strecke?

- *Risiko der Schätzunsicherheit beim Anbieter*

Große Vorhaben werden oft über alle Phasen von der Spezifikation bis zur Einführung und Hardwarebeschaffung ausgeschrieben. Dass ohne Vorliegen einer Spezifikation eine seriöse Aufwandsschätzung nicht machbar ist, bleibt das Risiko des Lieferanten (siehe dazu auch Kapitel 2). Oft werden solche Vorhaben unterschätzt. Das wird dann auch ein Problem für den Auftraggeber, wenn Mehraufwände nicht mehr im geplanten Terminrahmen abzuleisten sind.

- *Offshore Entwicklung*

Wir befinden uns in einem globalen Markt, weshalb also nicht auch die niedrigen Löhne und Kosten anderer Industrieländer wie China, Indien, Philippinen oder auch der Nearshore Länder wie Weißrussland, Polen und Ukraine ausnutzen? Wir satteln also auf die ohnehin schon komplexe Software-Entwicklung noch die Komplexität wie fremde Muttersprache, andere Kultur und Zeitverschiebung oben drauf. Bemerkenswert ist, dass es keine veröffentlichte Studien über gescheiterte Offshore Projekte gibt.

Die Kausalkette: „Software ist commodity, muss also immer günstiger einzukaufen sein" ist nur bis zu einem bestimmten Punkt richtig. Möglicherweise lohnt es sich, ein paar Euro mehr für Software auszugeben und Folgeschäden zu reduzieren. Gescheiterte Softwareprojekte haben US-Amerikanischen Firmen im Jahr 1995 immerhin 81 Mrd. USD [Stan01] gekostet.

2 Das Unglück beginnt mit der Schätzung

Prognosen sind besonders dann schwierig, wenn sie sich auf die Zukunft beziehen. Schätzung von Individualsoftware ist in der Informatik leider noch immer unterentwickelt. Es gilt noch immer: „Our techniques of estimating are poorly developed" [Bro75]. Alle Schätzmethoden basieren letztlich auf unscharfen Basisgrößen wie z. B. die Einordnung in leichte, mittlere und schwere Usecases, in Funktionspunkte oder in Lines of Code. Je früher die Projektphase, umso höher die Schwankungsbreite der Schätzung. Ein Faktor 4 zwischen zwei Schätzergebnissen ist in frühen Phasen normal [DeMar94], wenn eine Schätzung überhaupt möglich ist: „The cone of uncertainly implies that it is not only difficult to estimate a project accuractly in the early stages, it is theoretically impossible" [McCo98]

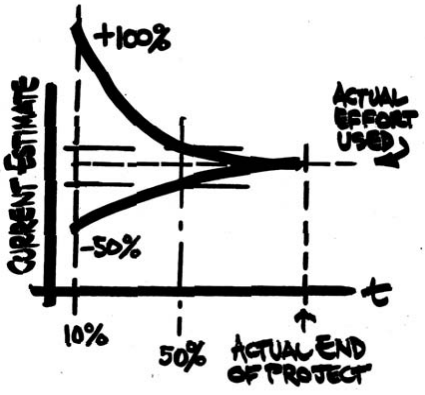


Abbildung 2: Schätzunsicherheit nach Projektverlauf (aus [DeMa94])

Versetzen wir uns in die Lage eines Auftraggebers. Die Ausschreibungsunterlagen sind verschickt. Vier Softwarelieferanten geben die Angebote ab: der Teuerste 3,4 Mio €, der Billigste 950 T€, dazwischen zwei mit 1,9 Mio € und 2,5 Mio €.

Ist die Aussage dieser Angebote, dass der Billigste die fachlichen Anforderungen nicht verstanden hat oder der Teuerste nicht wettbewerbsfähig ist, „sich zu warm angezogen hat“?

Hat wirklich jeder alles verstanden? Kennt jeder die Komplexität der eigenen Betriebsumgebung (deployment, Betrieb, Rechenzentrum etc.)? In dem Bewusstsein, dass eine Schwankung um Faktor 4 normal ist, macht man einen Fehler, wenn man sowohl den billigsten wie auch den teuersten Anbieter nicht ernst nimmt.

Aber was ist die Lösung?

Schätzgrundlagen schaffen

Ein Umsetzungsprojekt sollte ausschließlich auf Basis einer ordentlichen (!) und vollständigen (!) Spezifikation erfolgen. Wir beobachten häufig Ausschreibungen auf Basis von Spezifikationen, die diese Kriterien nicht erfüllen oder auf Basis von Requirements durchgeführt werden.

Machen Sie sich selbst ein Bild:

Auftraggeber sollten unabhängig von den Softwarelieferanten eine eigene Schätzung durchführen oder von Experten durchführen lassen.

Ist möglicherweise das eine oder andere Projekt von vorn herein zum Scheitern verurteilt, weil das Fundament einer ordentlichen Schätzung und damit Projektplanung gefehlt hat?

3 Struktur? Struktur!

Die Randbedingungen für Dienstleiter sind also strenger geworden: Dienstleistung wird als austauschbare Ware angesehen, der Preis steht im Vordergrund, Risiken aus großen Schätzunsicherheiten trägt häufig der Lieferant. Was bleibt also? Professionalität in Schätzung und Durchführung!

Im Folgenden gehe ich auf das Thema Durchführung ein und beginne mit einem Plädoyer für Struktur. Dies gilt sowohl für Kundenprojekte, als auch für interne Projekte.

Softwaresysteme sind die komplexesten Gebilde, die Menschen erschaffen haben. Für die Umsetzung solch komplexer Gebilde streben wir in der Informatik spätestens seit dem Nato-Gipfel 1968 nach Software-Engineering: Friedrich L. Bauer: „There is so much tinkering with software…..what we need is software engineering“ [Nato68]

Seit dem versuchen wir uns mit Struktogrammen, Nassy-Shneidermann, endlichen Automaten und vielem mehr. Letztlich aber haben wir nur “Boxes and arrows“, Prosa und Sourcecode.

Einige wenige ewige Wahrheiten des Software-Engineering sind unbestritten und allgemein anerkannt: Über Datenkapselung (zu Beginn als ADT, heute als Objekte) oder Nutzung von Interfaces wird heute niemand mehr streiten. Standardarchitekturen (SOA, Quasar) [Sid04] haben ihren Platz gefunden, und dass „goto´s“ gefährlich sind, wissen wir auch spätestens seit Edsger W. Dijkstra [Dijk68].

Die gleichen Paradigmen, die wir für die Umsetzung von komplexen Softwaresystemen wie selbstverständlich verwenden, sollten auch für die Projektplanung gelten:

Kapselung

Gute Pläne fassen verschiedene Arbeitspakete zu größeren Einheiten zusammen und kapseln den jeweils darunter liegenden Inhalt. Eine geordnete Darstellung dieser Kapselung sollte in Form von Projektstrukturplänen (work breakdown structures) erfolgen.

Interfaces

Was für die Software selbst Interfaces oder Schnittstellen sind, sind für Pläne Abhängigkeiten zwischen Arbeitspaketen. Die Planung sollte so geschnitten sein, dass sich möglichst einfach darstellbare Abhängigkeiten ergeben.

Architektur

Architektur hat etwas mit Ästhetik zu tun, genau wie ein Plan eine gewisse Ästhetik ausdrücken sollte.

goto

Haben Sie schon einmal einen Plan mit einem goto gesehen? Kennen Sie Pläne, die eine Abhängigkeit des Arbeitspaketes 563 zum Arbeitspaket 2 und vom Arbeitspaket 17 zum Arbeitspaket 623 darstellen? Na bitte!

Für die Steuerung von großen Softwareprojekten ist also professionelle Planung unerlässlich, ich gehe deshalb kurz auf die wichtigsten Themen ein.

4 Steuerung von Softwareprojekten

Es scheint so einfach, viele Bücher sind darüber geschrieben worden. Und trotzdem scheitern noch immer 23% aller Softwareprojekte. Gute Planung ist notwendige Grundlage, aber nicht hinreichend für die erfolgreiche Steuerung von Software Projekten. Trotzdem muss man mit den Grundlagen beginnen, denn nicht einmal da sprechen wir die gleiche Sprache.

Gleiche Sprache

Im Maschinenbau ist eine M6 Schraube eine M6 Schraube. Die ist genormt, darüber gibt es keine Diskussion. In der Informatik ist die Definition eines Arbeitspaketes zwar auch genormt,[13] die Norm hat aber in der Industrie leider noch keinen Einzug gefunden.

Ebenso verhält es sich mit den Begriffen wie: Aktivität, Aufgabe, Vorgang, Stufe und Release.

Weitere Beispiele sind Begriffe wie: Modultest, Integrationstest, Entwicklertest und Systemtest. Auch hier beobachten wir in verschiedenen Projekten verschiedene Interpretationen.

Selbst Rollen und Titel sind nicht normiert. Hat ein Projektleiter oder ein AP-Leiter mehr Verantwortung, gibt es einen Projektmanager und was macht ein Teilprojektleiter?

Wir begegnen bei jedem Kunden in jedem Projekt neuen Begriffen. Nun ist es Gott sei Dank nicht wie beim Turmbau zu Babel, dass daran Projekte scheitern, aber mindestens die ersten Wochen sind schwierig und führen zu Missverständnissen.

Handwerkliche Fehler bei der Projektplanung

Es gibt einige Mindestanforderungen, die wir an gute Pläne stellen. Eine fehlende Umsetzung davon ist manchmal möglicherweise durch die Auftraggeber-Auftragnehmerbeziehung verursacht. Die folgende unsortierte Liste soll einen Eindruck über handwerkliche Fehler vermitteln, die uns in der Projektplanung häufig begegnen.

[13] [DIN69901 „Projekt management systeme“]

- Arbeitspakete werden zu klein (ein oder zwei Tage) oder zu groß (zwei oder drei Monate) geplant

- Planung ohne Puffer: ein Plan mit Puffer ist kein Zeichen von Unsicherheit oder Schwäche sondern von Professionalität

- Nicht jeder Termin ist ein Meilenstein (Milestone oder Millstone) [Bro75]

- Projektteams mit viel Teilzeitkräften (trau keinem unter 70)[14]

- Fehlendes Change Management

- Fehlende Definition von "was ist fertig": kennen Sie Statusberichte, bei denen Arbeitspakete 4 Wochen lang den Status zu 90% fertig haben?

- Fehlende Projektstrukturpläne

- Endtermine die nicht geplant, sondern vom Management vorgegeben sind

- Das Einfachste wird zuerst eingeplant (weil es am Einfachsten ist): Ein Projekt gerät nicht in Zeitverzug, weil die letzte Stammdatenpflege nicht fertig ist, sondern weil die kritischen Funktionen nicht funktionieren. Diese plant man zu erst ein.

- Teamaufbau zu stark. Man kann es nur immer wieder betonen: das „Chinesenprinzip" funktioniert nicht ewig, und „Adding manpower to a late software project makes it later". [Bro75]

Der Projektplan

Wir schlagen vor, die Projektplanung in Anlehnung an die PMI Arbeitsgebiete 5, 6 und 9 [PMBOK04] als die Summe folgender Einzelprojekte zu definieren:

- 1. Projektstrukturplan (der zeigt das „was")

- 2. der Aufwandsplan (der zeigt, „wie aufwändig")

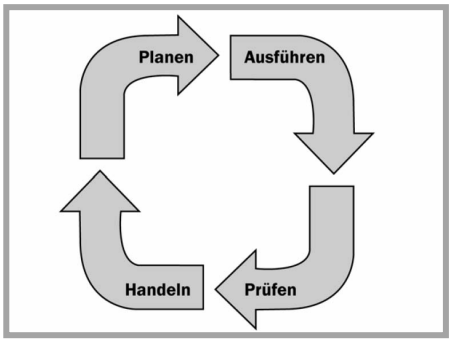
- 3. der Terminplan (der zeigt, das „wann" und die Abhängigkeiten)

- 4. der Personalplan (der zeigt das „wer")

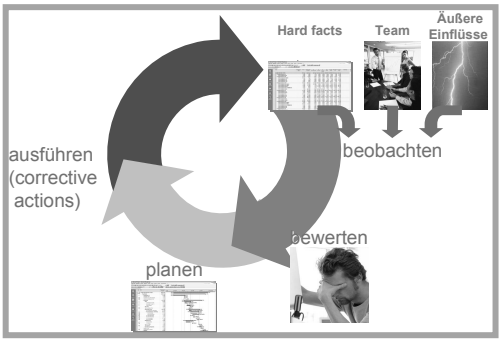
[14] 70 meint in diesem Zusammenhang weniger als 70 % Kapazität

Softfacts

Neben diesen Hardfacts gibt es eine Reihe von Softfacts, die für die erfolgreiche Steuerung von Softwareprojekten zusätzlich berücksichtigt werden müssen. In Erweiterung zu PMBOK ® plädieren wir hier für einen erweiterten Regelkreislauf, der die Schritte „beobachten“ und „bewerten“ als explizite Prozessschritte definiert. Wir trennen in Anlehnung an die Medizin den *Befund* (Fieber = beobachten) vom Schritt *Diagnose* (Grippe = bewerten) und *Therapie* (Bettruhe = Planung). Das Ausführen könnte man als *Anamnese* des Folgeschritts betrachten.



PMBOK Regelkreislauf



sd&m Regelkreislauf

Literaturverzeichnis

[Ber05] Dirk Berensmann, Deutsche Postbank AG, Vortrag auf der sd&m-Konferenz Software-Industrialisierung am 16.06.2005, IT matters – but who cares?

[Bro75] Frederick P. Brooks: The Mythical Man Month, Addison Wesley 1975

[Car04] Nicholas G. Carr: Does IT matter? Harvad Business School Publishing, Boston 2004

[DeMa94] Tom de Marco: Peopleware and beyond: Workshop mit sd&m, München 1994

[DeMa,Lis91] Tom DeMarco, Timothy Lister: Wien wartet auf Dich!

[Dijk68] Edsger W. Dijkstra: Letters to the editor: go to statement considered harmful, Communications of the ACM, v.11 n.3, p.147-148, 1968

[McCo98] Steve Mc Connell: Software Project Survival Guide, Microsoft Press, Redmond, 1998

[Nato68] P. Naur and B. Randell (Eds.): Report on a Conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch 7.-11.10.1968, NATO, Brüssel 1969

[PMBOK04] Projekt Management Institute: A guide to the Projekt Management of Knowledge (PMBOK ® Guide) Dritte Ausgabe, Pennsylvania 2004

[Sid04] Johannes Siedersleben: Moderne Software-Architektur. Umsichtig planen, robust bauen mit Quasar, Dpunkt Verlag 2004

[Stan01] Standish Group www.standishgroup.com