8 Projektleitung und Projektleiter

Nachdem in Kapitel 7 die statischen Strukturen betrachtet wurden, die in Software-Projekten vorkommen, geht es in diesem Kapitel um die Durchführung der Projekte und damit auch um die Rolle, die der Projektleiter innehat. Wir gehen nachfolgend von einem starken Projektleiter aus, wie er vor allem in der reinen Projektorganisation auftritt.

8.1 Ziele und Schwerpunkte des Projektmanagements

Projektmanagement hat immer zum Ziel, das Projekt erfolgreich durchzuführen und abzuschließen. »Erfolgreich« bedeutet, dass das Projekt die definierten Resultate in der geforderten Qualität innerhalb der vorgegebenen Zeit und mit den vorgesehenen Mitteln erzielt. Weitere sekundäre Ziele sind in der Regel

- der Aufbau oder die Verstärkung eines guten Rufs durch einen positiven Eindruck auf den Kunden und auf den Markt,
- die Aneignung von Kenntnissen, die zukünftig benötigt werden,
- die Entwicklung wiederverwendbarer Komponenten,
- die Wahrung eines attraktiven Arbeitsklimas für die Mitarbeiter.

Damit diese Ziele erreicht werden, muss das Projekt aktiv geleitet werden. Die Aktivitäten des Projektmanagements haben folgende Schwerpunkte:

- *Planen*
  Ohne Pläne kann ein Projekt nicht geführt werden; Planungsfehler lassen sich später selbst durch hochklassige Technik nicht kompensieren.
- *Bewerten und kontrollieren*
  Die Arbeitsergebnisse und der Projektfortschritt müssen bewertet werden; es muss überwacht werden, ob sich die Beteiligten an Vereinbarungen und Absprachen halten.
- *Kommunizieren*
  Der Projektleiter steht zwischen dem Management, dem Kunden oder dem Marketing und den Mitarbeitern. Für das Management repräsentiert er das

Projekt, für den Kunden die Herstellerfirma, für das Marketing die Technik, für die Mitarbeiter die Leitung der Firma. Entsprechend unterschiedlich sind die Erwartungen, denen er genügen soll. Nur wenn er auf allen Seiten zuhört und nach allen Seiten Informationen weitergibt, hat er eine Chance.

- *Günstige Rahmenbedingungen schaffen und erhalten*
  Ein Projekt gedeiht am besten, wenn die Mitarbeiter, ausgestattet mit der notwendigen Infrastruktur, konzentriert und ungestört stabile Ziele verfolgen können. Aber um das Projekt herum gibt es wankelmütige Kunden, unklare Zielsetzungen, Restrukturierungen, Sparmaßnahmen, enge Büros, lange Wege und andere Projektleiter, die versuchen, Mitarbeiter zu »stehlen«. Es ist Aufgabe des Projektleiters, das Projekt vor diesen störenden Einflüssen zu schützen.
- *Mitarbeiter führen und motivieren*
  Führen heißt: vorangehen, den Weg zeigen, auch die Gruppe mitziehen. Die meisten Entwickler wollen gute Leistungen erbringen, brauchen aber auch Orientierung und Bestätigung.
- *Schwierigkeiten möglichst früh erkennen und bekämpfen*
  Unvorhergesehene Schwierigkeiten und Probleme sind im Projekt nicht die Ausnahme, sondern die Regel. Darum muss der Projektleiter den Horizont regelmäßig nach Eisbergen absuchen und, wenn er einen entdeckt hat, rasch und wirksam reagieren. Er muss also systematisches Risikomanagement betreiben.

8.2 Das Vorprojekt

Manche Software-Projekte sind von Beginn an scharf konturiert. Das ist bei Systemprojekten (siehe Abschnitt 7.3.4) der Fall, wenn die Software Bestandteil eines größeren Systems sein wird, über dessen Entwicklung schon entschieden wurde. Ist zudem klar, dass man die benötigte Software nicht einkaufen kann, so gibt es an der Durchführung des Software-Projekts keine Zweifel. Meist ist dann nur noch zu entscheiden, ob die Arbeit nach außen vergeben oder im eigenen Hause durchgeführt werden soll.

In sehr vielen Fällen gibt es aber zu Beginn weit mehr Unklarheiten, auch die, ob das Projekt überhaupt zustande kommt. Typisch sind folgende Situationen:

- Ein Software-Haus bemüht sich in Konkurrenz mit anderen Anbietern, einen Auftrag zu erhalten.
- Software-Entwickler schlagen vor, ein Werkzeug zu entwickeln, das bestimmte oft durchzuführende Arbeiten erheblich erleichtern würde.
- Die Marketing-Abteilung weist auf einen Trend hin, durch den der Bedarf an bestimmten Software-Produkten erheblich steigen wird, und schlägt vor, rechtzeitig ein solches Produkt zu entwickeln.

- Mitarbeiter einer Bank wünschen eine neue Software für die Kundenberatung, weil die vorhandene veraltet und unflexibel ist.
- Die mit der Software-Wartung befassten Mitarbeiter schaffen es kaum noch, die notwendigen Anpassungen vorzunehmen, und drängen darauf, die alte Software nicht weiter zu flicken, sondern durch eine neue zu ersetzen.

Allen diesen Situationen ist gemeinsam, dass zunächst niemand wirklich weiß, welche Entscheidung richtig ist.

- Natürlich müssen Aufträge akquiriert werden, aber durch ein zu billiges Angebot kann sich der Anbieter ruinieren; ein zu teures Angebot hat keine Chance.
- Wird der Nutzen des Werkzeugs die Kosten übersteigen? Ist überhaupt die personelle Kapazität für eine Werkzeugentwicklung verfügbar?
- Haben wir das Know-how, um das neue Produkt zu entwickeln, und können wir es rechtzeitig auf den Markt bringen?
- Liegt das Problem wirklich in der Software, oder brauchen die Mitarbeiter einfach neue Rechner, moderne Bildschirme oder eine bessere Vernetzung?
- Könnte man die Probleme der Wartung auch dadurch entschärfen, dass man Änderungswünsche der Anwender sachlich prüft, statt sie grundsätzlich zu akzeptieren?

Wie man sieht, ist die Vorstellung, dass ein Projekt einfach beschlossen und begonnen werden könnte, allzu naiv. Meist brauchen wir ein *Vorprojekt*.

Das Vorprojekt führt von der Projektidee (»Wir könnten dieses Projekt machen«) zum Beschluss oder zur Feststellung über das Projekt (»Wir machen dieses Projekt« oder »Wir machen es nicht«).

Das Projektmanagement erfordert immer wieder Kompromisse, weil sich keines der Projektziele in reiner Form verwirklichen lässt. Das gilt ganz besonders für das Vorprojekt. Wenn am Ende eine negative Entscheidung steht, ist der investierte Aufwand verloren. Das spricht für eine enge Begrenzung des Aufwands.

Andererseits ist es notwendig, alle Informationen zu beschaffen und zu verarbeiten, die für eine leidlich zuverlässige Aufwandsschätzung benötigt werden. Eine zu hohe Schätzung kann eine in Wahrheit sinnvolle, rentable Entwicklung verhindern. Eine zu niedrige Schätzung ist noch schlimmer: Die personelle Kapazität wird länger gebunden, als vertretbar ist, die Kosten werden durch den zu niedrig angesetzten Preis nicht hereingeholt, und im Fall eines Auftragsprojekts wird der Ruf des Anbieters beschädigt.

8.2.1 Die Organisation des Vorprojekts

Im (nicht erreichbaren) Idealfall hat man am Ende des Vorprojekts alle Informationen, die für die Entscheidung über das Projekt benötigt werden. Bei einem

Auftragsprojekt weiß man, wie weit man sich bei den Terminen und beim Preis aus dem Fenster lehnen darf, beim EDV-Projekt kennt man die Kosten und die Dauer des Projekts, und bei einem Entwicklungsprojekt kann man vorhersagen, wann und in welchen Schritten die Markteinführung möglich und welcher Erfolg zu erwarten ist.

Dazu ist vor allem eine Aufwandsschätzung nötig, verbunden mit der Analyse, ob die richtigen Leute zum richtigen Zeitpunkt verfügbar sein werden. Das erfordert eine ungefähre Vorstellung von der Struktur des Systems und der eingesetzten Technologie, und diese hängen wiederum von den Anforderungen ab. Wir benötigen also:

- einen Überblick über die wichtigsten Anforderungen, vor allem solche, die den Aufwand erheblich beeinflussen,
- eine Konzeption, die Architektur des Systems, freilich ohne Details, aber mit den wichtigsten Entscheidungen über die einzusetzende Technologie,
- einen Entwicklungsplan,
- eine Aufwandsschätzung.

Es hat sich bewährt, das Vorprojekt in einer kleinen Gruppe (das sind bei einem kleinen bis mittelgroßen Projekt zwei bis fünf Leute) durchzuführen, in der sowohl Kenntnisse des Anwendungsbereichs als auch der Technik vertreten sind. Die Zusammenarbeit ist eng und kaum reglementiert. Der insgesamt investierte Aufwand liegt typischerweise bei wenigen Prozent der Entwicklungskosten.

8.2.2 Das Angebot

Soll ein Auftrag akquiriert werden, dann ist das Ergebnis des Vorprojekts das *Angebot*. Darin beschreibt der Anbieter, was er wann zu welchem Preis liefern kann. Ein Angebot enthält also:

- eine Beschreibung des zu liefernden Systems,
- Alleinstellungsmerkmale des Angebots,
- einen Plan mit den externen Meilensteinen des Projekts (Abschnitt 8.3.2),
- Hinweise zum weiteren Vorgehen (Kontaktpersonen).

Das zu liefernde System sollte so beschrieben werden, dass sich der Auftraggeber darin wiederfindet. Das schließt in der Regel eine rein technische Beschreibung aus. Stattdessen sollte der Auftraggeber erkennen, wie er später das System benutzen und von diesem profitieren kann. Der Anbieter muss dazu die Perspektive des Auftraggebers einnehmen. Ein guter Prototyp ist in vielen Fällen ein überzeugendes Argument.

Das bedeutet: Es ist falsch, das System als Maschine zu beschreiben. (»Wir liefern Ihnen eine Software auf der Basis der Technologie XYZ mit den folgenden technischen Merkmalen: ...«) Viel besser ist es, ein Szenario zu entwerfen, in dem

der Auftraggeber vorkommt. (»Sie können auf diese Weise Ihre alten Kundendaten ohne weitere Bearbeitung übernehmen und an dieser Bedienschnittstelle jederzeit abfragen, wo sich eine bestimmte Lieferung gerade befindet.«)

Die Alleinstellungsmerkmale können ganz unterschiedlicher Art sein. Es kann sich um eine besonders elegante Bedienung handeln, um die garantierte Zuverlässigkeit oder um die sehr bescheidene Inanspruchnahme der Ressourcen, z. B. des Speichers. Auch die Art der Projektdurchführung kann ein Alleinstellungsmerkmal sein. In jedem Fall muss es sich um Merkmale handeln, die über die Anforderungen hinausgehen, für den Auftraggeber attraktiv erscheinen und für den Anbieter keinen erheblichen Mehraufwand verursachen.

Bei der Akquise eines Auftrags spielt natürlich auch die Erwartung eine Rolle, ob und in welchem Maße man sich bei der Wartung der zu entwickelnden Software schadlos halten kann. Es ist offensichtlich, dass in dieser Phase oft mehr gepokert als technisch argumentiert wird.

8.2.3 Weitere Ergebnisse des Vorprojekts

Das Vorprojekt schafft die Grundlagen für die Entscheidung über das Projekt. Kommt das Projekt tatsächlich zustande, sollte der im Vorprojekt eingesetzte Aufwand weiter genutzt werden, denn es sind ja bereits einige Dokumente entstanden – natürlich nur rudimentär:

- Anforderungen wurden erhoben.
- Die Architektur ist wenigstens grob entworfen.
- Eine erste Aufwands- und Kostenschätzung wurde angefertigt.
- Ein Plan für die Durchführung wurde erstellt.

Alle diese Dokumente sind für die weitere Entwicklung nützlich, wenn sie ausreichend strukturiert, nachvollziehbar formuliert und verfügbar sind. Besonders wichtig ist es, die Quellen aller Anforderungen zu notieren; in vielen Fällen ist es später nötig, Quellen zu überprüfen, bei den Gesprächspartnern nachzufassen oder mit ihnen über Alternativen zu verhandeln.

Vor allem, wenn kein externer Auftraggeber beteiligt ist, besteht eine starke Neigung, das Vorprojekt gleitend ins Projekt übergehen zu lassen. Da das eigentliche Projekt wesentlich weiter gehenden Regeln unterliegt, sollte sein Beginn (und damit das Ende des Vorprojekts) sehr deutlich signalisiert werden. Eine Vermischung der Phasen beschädigt den Entwicklungsprozess.

8.3 Start des Projekts und Projektplanung

Ein Software-Projekt konkretisiert sich in einem Zeitraum zwischen zwei Ereignissen: der Entscheidung des Auftraggebers, ein Software-System entwickeln zu lassen, und dem eigentlichen Startschuss des Projekts. Dieser fällt, wenn der Pro-

jektplan in Kraft gesetzt wird. Dazwischen liegt die Definition des Projekts, die sogenannte *Startphase*. In dieser Phase wird der Projektleiter ernannt. Er verkörpert das Projekt, vertritt es nach außen und innen und bildet die Keimzelle des Projektteams.

Seine erste und wichtigste Aufgabe ist es, auf der Grundlage eines mehr oder weniger klar und vollständig formulierten Projektauftrags einen Vorgehensplan zu erstellen und die Rahmenbedingungen zu gestalten, unter denen das Projekt durchgeführt wird. Wichtige Teilaufgaben der Planung sind:

- Abgrenzen des Liefer- und Leistungsumfangs
- Definieren und Gliedern der Aufgaben
- Festlegen des Berichtswesens innerhalb des Projektteams sowie an der Schnittstelle zum Auftraggeber
- Abschätzen des Personalbedarfs (Anzahl und Qualifikation der benötigten Mitarbeiter)
- Erstellen des Budgets und der Terminpläne
- Verteilen der Aufgaben über die Rollen
- Bereitstellen der notwendigen Unterstützung (Dienstleistungen)
- Identifizieren von Risiken

Das Resultat dieser Tätigkeiten ist der Projektplan (siehe Abschnitt 8.3.3). Darin beschreibt der Projektleiter, wie das Projekt durchgeführt werden soll. Nachdem der Projektplan durch den Projekteigentümer und den Auftraggeber bestätigt und in Kraft gesetzt wurde, beginnt der produktive Teil des Projekts, die *Durchführungsphase*.

Ein Plan ist die geistige Vorwegnahme des zukünftigen Handelns. Wenn wir ein Projekt planen, dann legen wir auf der Basis des aktuellen Wissensstandes fest, wie und mit welchen Mitteln wir das Projekt durchführen wollen. Da wir das nicht exakt tun können, stimmen die Pläne nie exakt mit dem realen Verlauf überein. Eine gewisse Spannung zwischen Soll und Ist lässt sich nicht vermeiden. Das gilt insbesondere, wenn weit in die Zukunft geplant werden muss.

Diese Schwierigkeit ist aber kein Grund, eine brauchbare Planung als Utopie abzutun. Indem wir Risiken und Schwankungen der Bedingungen bereits in der Planung berücksichtigen, sorgen wir dafür, dass wir insgesamt im Plan bleiben. Ein Forschungsreisender, der Probleme mit dem Kompass hat, sollte den Kompass nicht wegwerfen, sondern ihn mit mehr Bedacht gebrauchen.

8.3.1 Planungsaspekte

Planung ist die zentrale Aktivität in der Startphase, zudem ist sie Bestandteil des Regelkreises der Projektdurchführung (siehe Abschnitt 8.8.1). Weil sich Pläne von der Realität lösen können, muss während der Projektdurchführung kontinuierlich geprüft werden, inwieweit der Stand des Projekts (Ist-Werte) mit der Pla-

nung (Soll-Werte) übereinstimmt. Wird eine erhebliche Abweichung festgestellt, dann müssen Korrekturmaßnahmen ergriffen werden. Oft muss auch die Planung korrigiert werden. In Abschnitt 8.8 stellen wir Ansätze vor, um den Projektstand zu bestimmen. Prinzipiell muss die Planung die folgenden Aspekte abdecken:

- *Planung der Aufgaben*
  Dabei wird festgelegt, was zu tun ist, um die Projektziele zu erreichen. Das Ergebnis der Aufgabenplanung sind Arbeitspakete. Die Aufgabenplanung ist eine Voraussetzung, um Termine und Ressourcen planen zu können.
- *Planung der Termine*
  Es ist festzulegen, wann welche Resultate verfügbar sein sollen. Dazu gehört die Terminplanung für die Meilensteine und die Releases, wenn das System in Ausbaustufen entwickelt wird. Natürlich ist auch der Endtermin anzugeben.
- *Planung der Ressourcen*
  Um die definierten Arbeitspakete zu bearbeiten, werden Ressourcen (im weitesten Sinne) benötigt, vor allem der Aufwand der Mitarbeiter (in Personentagen). In Abschnitt 8.4 gehen wir näher darauf ein, wie dieser Aufwand geschätzt werden kann. Auf der Basis dieser Schätzungen kann geplant werden, wann wie viele Mitarbeiter mit welchen Qualifikationen benötigt werden. Weiterhin müssen die Kosten für Hardware, Software und andere Investitionen eingeplant werden, die das Projekt benötigt. Sind alle diese Zahlen ermittelt, kann geplant werden, welches Budget zu welchem Zeitpunkt im Projekt sinnvoll und erforderlich ist.

Die Planung der Aufgaben: Arbeitspakete

Bevor Termine, Zeiten und Aufwände geplant werden können, müssen die Aufgaben identifiziert werden, die zu lösen sind, um die Projektziele zu erreichen. Dies geschieht natürlich auf verschiedenen Granularitätsstufen. Zuerst werden die großen Aufgabenbereiche ermittelt, dann werden sie verfeinert. Die Aufgaben werden folgendermaßen identifiziert: Die Gesamtaufgabe wird so lange baumartig in Unteraufgaben zerlegt, bis die Blätter des so entstehenden Aufgabenbaums nicht weiter aufgeteilt werden müssen, weil sie Arbeitspakete definieren, die jeweils separat geplant, durchgeführt und kontrolliert werden können.

Ein Arbeitspaket ist eine Aufgabe, die ein Entwickler oder ein kleines Team in überschaubarer Zeit, höchstens einem Monat, mit gut planbarem Aufwand lösen kann. Die Arbeitspakete sind die Atome der Projektplanung, eine weitere Verfeinerung ist aus Sicht der Planung weder sinnvoll noch notwendig.

Eine Aufgabe ist als Arbeitspaket geeignet, wenn

- sie durchgängig erledigt werden kann, ohne dass es Koordinationszwänge gibt,
- der Fortschritt und das Ende objektiv feststellbar (messbar) sind,

- es Ereignisse gibt, die Anfang und Ende bestimmen, und
- der Aufwand und die Termine schätzbar sind.

Auch wenn diese Kriterien in der Praxis nicht immer zu erfüllen sind, sollte man sich von ihnen leiten lassen. Das Ergebnis der Aufgabenplanung ist immer eine Liste von Arbeitspaketen. Es hat sich bewährt, die Gliederung der Aufgaben in einer grafischen Baumdarstellung oder in einer strukturierten Liste zu visualisieren (siehe Abb. 8–1). Als Ergebnis erhalten wir den sogenannten *Projektstrukturplan* (engl. *work breakdown structure*).

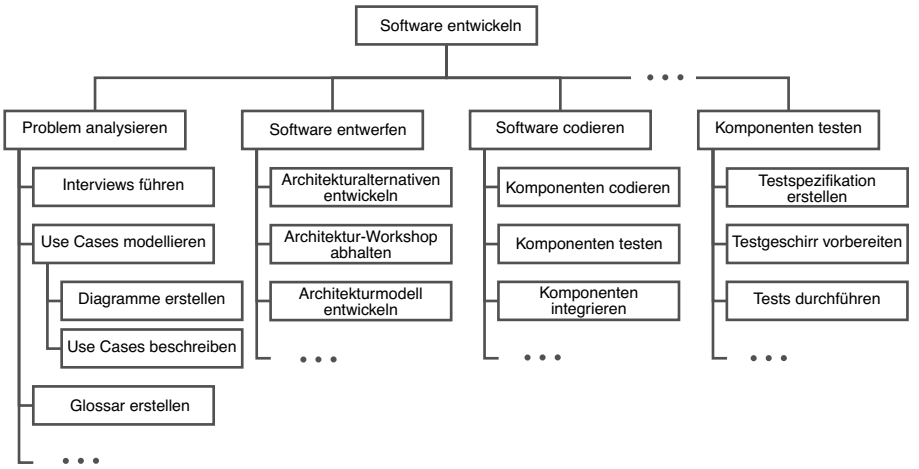

Abb. 8–1 *Schema eines Projektstrukturplans*

Prozessmodelle wie das V-Modell (Abschnitt 10.3) oder der Unified Process (Abschnitt 10.4) erleichtern die Aufgabenplanung, da sie Phasen und die durchzuführenden Entwicklungsaktivitäten vorgeben und somit eine generische Struktur für die Aufgabenplanung bieten.

8.3.2 Projektphasen

Eine *Phase* ist ein zusammenhängender, also nicht unterbrochener Zeitraum, in dem bestimmte Arbeiten durchgeführt und abgeschlossen werden. Am Ende der Phase steht ein *Meilenstein*. Die Phase ist erfolgreich beendet, wenn die Kriterien, die der Meilenstein vorgibt, erfüllt sind. Typischerweise beginnt dann die nächste Phase. Die Phasen überlappen sich also nicht; allerdings kann es in größeren Projekten verschiedene Entwicklungsstränge geben, die parallel ablaufen und durch unterschiedliche Meilensteine gegliedert sind.

Die Aufteilung eines Projekts in Phasen erleichtert die Kontrolle, da nicht das gesamte Projekt auf einmal betrachtet und finanziert werden muss, sondern

jeweils nur ein einzelner Abschnitt. Am Meilenstein, d. h. an der Prüfung der Zwischenergebnisse, ist der Kunde beteiligt.

Die Granularität der Phasengliederung ist kritisch; sind die Phasen sehr kurz, so wird der Kunde laufend in Anspruch genommen, und es gibt keine Freiräume für eine ruhige Entwicklung. Sind die Phasen zu lang, so kann es innerhalb der Phasen zu erheblichen Verzögerungen kommen, die erst am Ende der Phase auffallen. Darum werden, wenn nötig, neben den oben beschriebenen externen Meilensteinen interne Meilensteine eingeführt, an denen der Kunde nicht beteiligt ist.

- *Externe Meilensteine* definieren Ergebnisse, die aus Sicht des Auftraggebers von Bedeutung sind, z. B. das Vorliegen bestimmter Dokumente, eines Prototyps oder einer ersten Ausbaustufe. Bei externen Meilensteinen entscheidet der Auftraggeber nach Bewertung der erzielten Ergebnisse, ob die Arbeitspakete der nächsten Phase in Angriff genommen werden dürfen. Je nach Vertrag können mit dem Erreichen von Meilensteinen auch Zahlungen des Auftraggebers verbunden sein.
- *Interne Meilensteine* sind dann sinnvoll, wenn der Zeitraum zwischen zwei externen Meilensteinen so groß ist, dass Zwischenkontrollpunkte eingeführt werden sollten. Diese müssen aus Sicht des Projektleiters geeignet sein, den Fortschritt des Projekts sichtbar und messbar zu machen.

Wichtig ist, dass es für das Erreichen eines Meilensteins Kriterien gibt, die keinen Raum für Subjektivität (und damit Willkür) lassen. Es muss sich also objektiv feststellen lassen, ob alle Voraussetzungen zum Erreichen eines Meilensteins erfüllt sind. Zur Definition eines Meilensteins gehören demnach

- die Definition der Ergebnisse, die vorliegen müssen,
- die geforderten Qualitätseigenschaften dieser Ergebnisse,
- die Instanz (Person oder Gremium), die entscheidet, ob der Meilenstein erreicht ist, und
- der vorgesehene Zeitpunkt für das Erreichen des Meilensteins.

Damit die in einem größeren Unternehmen durchgeführten Projekte nach einem einheitlichen Schema organisiert und durchgeführt werden, verwenden diese Unternehmen oft eigene Prozessmodelle, die allen Projekten Standardphasen und Meilensteine vorgeben.

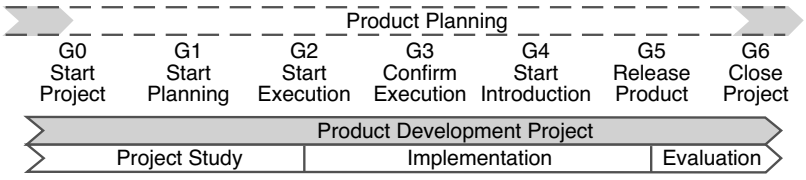

Abb. 8–2 *Beispiel eines Gate-Modells (aus Fröhlich, Lichter, Zeller, 2001)*

Abbildung 8–2 zeigt ein solches Beispiel: Das Prozessmodell definiert drei Projektphasen und sieben externe Meilensteine, die *Gates* genannt werden (Cooper, 2001). An jedem dieser Meilensteine sind spezifische Ergebnisse vorzulegen; dann wird entschieden, ob das Projekt weitergeführt werden kann oder abgebrochen werden muss.

Damit die definierten Arbeitspakete zeitlich sinnvoll angeordnet werden können, müssen die folgenden Einflüsse berücksichtigt werden:

- *Logische Abhängigkeiten zwischen Arbeitspaketen*
  So muss der Entwurf eines Moduls vorliegen, bevor es codiert werden kann.
- *Aufwand für die Arbeitspakete*
- *Einflüsse von außen*
  Lieferungen der Hardware und zugekaufter Software, Bereitstellung von Daten, Algorithmen usw. durch den Auftraggeber sowie die Bestätigung der Abnahme von Zwischenresultaten sind wichtige Ereignisse, die der Projektleiter nicht direkt beeinflussen kann.
- *Verfügbarkeit von Mitarbeitern und Betriebsmitteln*
  Die Verfügbarkeit von Spezialisten, der Zugriff zu Rechen- oder Testeinrichtungen, eventuell konkurrierend mit anderen Projekten, schaffen zusätzliche Einschränkungen.

Werden die logischen Abhängigkeiten zwischen den Arbeitspaketen und den externen Ereignissen ermittelt und dargestellt, dann wird ersichtlich, welche Arbeitspakete von der Sache her nur nacheinander und welche auch gleichzeitig ausgeführt werden können. Es gibt in der Praxis die folgenden drei Anordnungsbeziehungen zwischen Arbeitspaketen:

- *Normalfolge* (Ende-Anfang-Beziehung)
  Das Ende eines Arbeitspakets ist Voraussetzung für den Anfang eines anderen Arbeitspakets. Beispiel: Die Testumgebung muss installiert sein, bevor mit dem Test begonnen werden kann.
- *Anfangsfolge* (Anfang-Anfang-Beziehung)
  Der Anfang eines Arbeitspakets ist Voraussetzung für den Anfang eines anderen Arbeitspakets. Beispiel: Die Codierung muss begonnen haben, bevor mit der externen Code-Dokumentation begonnen werden kann.
- *Endfolge* (Ende-Ende-Beziehung)
  Das Ende eines Arbeitspakets ist Voraussetzung für das Ende eines anderen Arbeitspakets. Beispiel: Die Evaluierung des Prototyps muss abgeschlossen sein, bevor die Anforderungsanalyse abgeschlossen werden kann.

Terminpläne werden mit Balkendiagrammen (Gantt-Charts) und Netzplänen (PERT-Charts) dargestellt. Die Gantt-Charts sind nach dem amerikanischen Ingenieur Henry L. Gantt (1861–1919) benannt (Abb. 8–3), der sie 1910 eingeführt

hat. PERT steht für Program Evaluation Review Technique; diese Netzpläne sind in den Fünfzigerjahren in der US-Militärindustrie entstanden. In Netzplänen werden im Unterschied zu Balkendiagrammen die Abhängigkeiten zwischen den Arbeitspaketen explizit dargestellt. Wenn zusätzlich deren Dauer und der Starttermin des Projekts bekannt ist, kann für jedes Arbeitspaket ermittelt werden,

- wann es frühestens begonnen und frühestens beendet werden kann,
- wann es im Hinblick auf den Endtermin des Projekts spätestens begonnen und beendet werden muss.

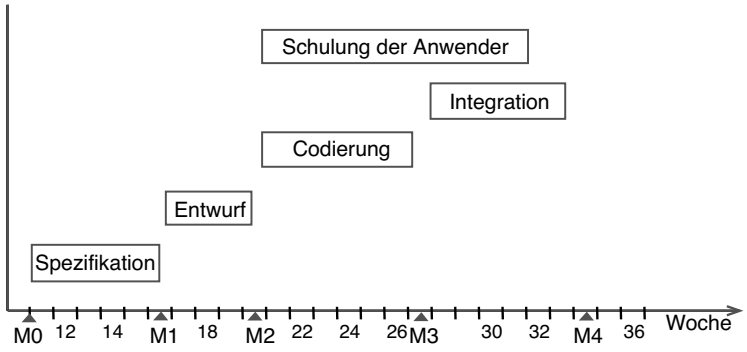

Abb. 8–3 *Gantt-Chart mit Meilensteinen*

Arbeitspakete, deren Dauer (d. h. auch: deren Verzögerung) direkt auf den Endtermin durchschlägt, werden *kritische Arbeitspakete* genannt. Sie bilden den *kritischen Pfad*, der besonders überwacht werden muss. Im Beispiel in Abbildung 8–4 sind die Arbeitspakete auf dem kritischen Pfad farblich hervorgehoben.

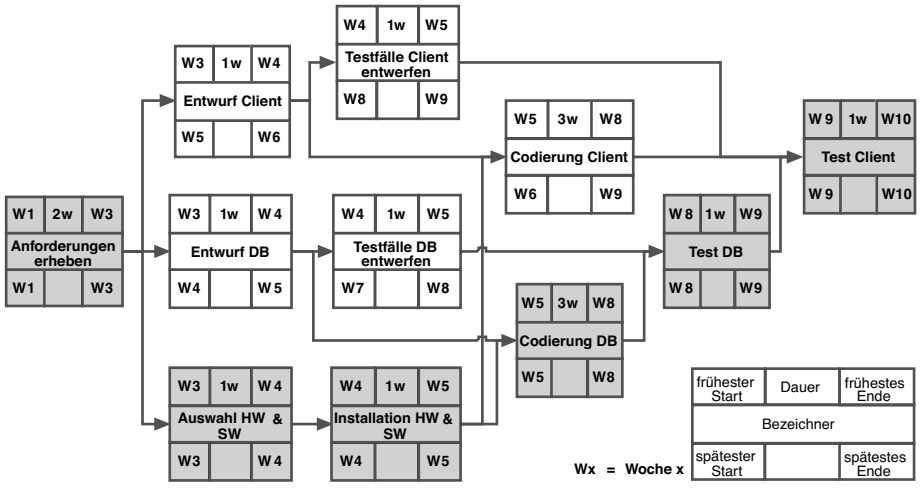

Abb. 8–4 *PERT-Chart (Netzplan) mit kritischem Pfad*

Die Erstellung und Nachführung von Netzplänen wird durch Planungswerkzeuge unterstützt, die in steigender Zahl angeboten werden. Sie erleichtern dem Projektleiter die Arbeit, bergen aber zwei Gefahren: Je ansprechender, »fertiger« die Darstellung erscheint, umso eher ist man geneigt, sie leichtfertig für sinnvoll zu halten (der bekannte Hochglanzeffekt). Und Abweichungen vom Plan lassen sich mit Werkzeugunterstützung leicht beseitigen, indem man die Pläne laufend der Realität anpasst. Da wedelt der Schwanz mit dem Hund. Das ist natürlich nicht sinnvoll.

Die Netzplantechnik wird z. B. in Kerzner (2017) eingehend beschrieben.

8.3.3 Der Projektplan

Die vorangehenden Abschnitte haben gezeigt, dass es eine Reihe von Plänen gibt (oder geben sollte), die zusammen die Planung bilden. Offensichtlich ist es sinnvoll, alle zum Projekt gehörenden Teilpläne in einem Dokument (oder Dokumentensystem), dem *Projektplan*, zusammenzuführen.

Abbildung 8–5 auf Seite 119 zeigt ein Muster für einen Projektplan. Natürlich kann ein Projektplan auch anders strukturiert werden. In jedem Fall ist es aber sinnvoll, bei der Planung von einem Muster auszugehen, das dann an die spezielle Situation angepasst wird. Dabei werden vor allem solche Punkte weggelassen, die für das konkrete Projekt keine Bedeutung haben.

8.4 Aufwands- und Kostenschätzung

Könnte der Projektleiter in die Zukunft sehen, so wüsste er schon vor Projektbeginn, welche Aufwände und damit auch welche Kosten entstehen werden, um das Projekt erfolgreich abzuschließen. Da die Projektleiter nicht hellsehen können, bleibt ihnen nur die Möglichkeit, durch Aufwandsschätzung die Unsicherheit der Zukunft in den Griff zu bekommen.

8.4.1 Ziele und Schwierigkeiten der Aufwandsschätzung

Eine der wichtigsten Fragen bei der Vorbereitung einer Software-Entwicklung lautet: Wie hoch wird der Aufwand sein? Sie ist eng verbunden (aber nicht gleichbedeutend) mit den Fragen:

- Wie lange wird die Entwicklung dauern?
- Wie viele Leute werden benötigt?

1. Einleitung
- 1.1 Zweck, Abgrenzung
- 1.2 Projektüberblick, Motivation

2. Formale Grundlagen
- 2.1 Vertragliche Anforderungen an die Projektdurchführung
- 2.2 Vertragliche Anforderungen an das Produkt
- 2.3 Vertragliche Anforderungen an die Konformität mit Normen
- 2.4 Gesetzliche Auflagen

3. Leistungen der Vertragspartner
- 3.1 Lieferumfang (Software, Dienstleistungen, ...)
- 3.2 Resultate, die nicht zum Lieferumfang gehören
- 3.3 Leistungen des Auftraggebers (Beistellungen)
- 3.4 Externe Meilensteine
- 3.5 Abnahmeprozedur
- 3.6 Änderungsverfahren

4. Entwicklungsprozess
- 4.1 Strategie für die Entwicklung und Integration
- 4.2 Projektspezifische Abweichungen vom Standardprozess
- 4.3 Phasen der Entwicklung
- 4.4 Dokumentationsplan
- 4.5 Prüfungen (Reviews und Tests)

5. Risiken
- 5.1 Risiken und ihre Bewertung
- 5.2 Maßnahmen zur Reduktion der Risiken
- 5.3 Notfallpläne

6. Richtlinien für die Entwicklung
- 6.1 Konfigurationsmanagement
- 6.2 Design- und Programmierrichtlinien
- 6.3 Einsatz von Werkzeugen

7. Anforderungen an die Umgebung
- 7.1 Infrastruktur (Büros, Rechnersysteme, Software)
- 7.2 Leistungen Dritter im Unternehmen
- 7.3 Leistungen externer Lieferanten

8. Projektorganisation
- 8.1 Schnittstelle zum Auftraggeber
- 8.2 Schnittstellen zur eigenen Organisation
- 8.3 Schnittstellen zu anderen Projekten
- 8.4 Schlüsselpersonen
- 8.5 Organisation (differenziert für die einzelnen Projektphasen)
- 8.6 Berichtswesen

9. Entwicklungsplan
- 9.1 Projektstrukturplan (Arbeitsgliederung)
- 9.2 Terminplan
- 9.3 Kostenplan

Abb. 8–5 *Muster für einen Projektplan (modifiziert nach Frühauf, Ludewig, Sandmayr, 2002)*

Aufwands- oder Kostenschätzverfahren sollen möglichst frühzeitig Antworten auf diese Fragen liefern. Die Resultate werden benötigt für die

- Kalkulation und Angebotserstellung,
- Personalplanung und mittelfristige Disposition,
- Vorbereitung einer Entscheidung »make or buy«,
- Nachkalkulation.

Zunächst ist festzustellen, dass es auf keinem Gebiet eine Möglichkeit gibt, *wirklich neue* Projekte präzise abzuschätzen; das unerwartet teure Münchner Olympiadach und das Opernhaus von Sydney (Gesamtkosten von 102 statt der geschätzten 7 Mio. AUS-$) sind zwei spektakuläre Beispiele, das Grimm'sche Wörterbuch (von den Gebrüdern Grimm als Arbeit für einige Jahre bei A begonnen, aber erst Generationen später mit Z abgeschlossen) ist als eigentliches Software-Projekt ebenfalls typisch. Da in der jungen Informatik neue Anwendungen noch viel öfter vorkommen als auf anderen Gebieten, ist die Zahl der unplanbaren Projekte besonders hoch. Der Trend geht jedoch in Richtung Routine-Entwicklung.

Aber auch wenn wir nichts völlig Neues planen, bleibt eine Schätzung eine Schätzung, wir kennen den tatsächlich benötigten Aufwand erst dann (oder könnten ihn kennen), wenn wir das Projekt abgeschlossen haben. Je früher wir schätzen, desto mehr Unsicherheit müssen wir in Kauf nehmen, weil vor uns entsprechend viele Unsicherheiten und Unklarheiten liegen. Je weiter das Projekt fortgeschritten ist, desto sicherer können wir schätzen, weil wir den Aufwand für die bereits abgeschlossenen Arbeitspakete kennen und auf das Fehlende hochrechnen können.

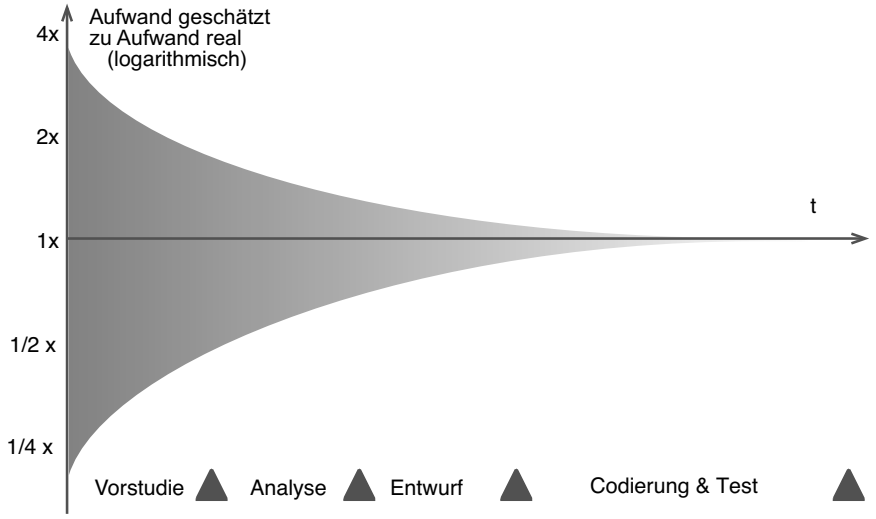

Abb. 8–6 *Der Schätztrichter, die Unsicherheit beim Schätzen (nach Boehm et al., 2000, S. 10)*

Abbildung 8–6 zeigt die abnehmende Unsicherheit im Projektverlauf. Da wir gerade aufgrund dieser Unsicherheit bei einer Schätzung nie wissen können, ob wir zu hoch oder zu niedrig schätzen, hilft uns der »Schätztrichter« nicht für die konkrete Prognose. Aber er macht deutlich, dass es sinnvoll ist, immer wieder zu schätzen. Sollte das Projekt mehr Aufwand benötigen als zu Beginn erwartet, muss der Projektleiter das so früh wie möglich erkennen.

Der Aufwand für kleine Einheiten lässt sich genauer schätzen als der für große, die nicht auf erkennbare Weise aus kleinen zusammengesetzt sind. Darum kann man erst brauchbare Schätzungen erwarten, wenn die Grobstruktur des Systems festgelegt ist. Vorher hat man nicht mehr als eine Daumenpeilung.

8.4.2 Ansätze der Aufwandsschätzung

Der Aufwand für ein Projekt oder Teilprojekt ist abhängig vom Umfang der Aufgabe, der Schwierigkeit und der Produktivität der Entwickler. Die folgende Formel zeigt den Zusammenhang:

$$\text{Aufwand} = \frac{\text{Umfang} \times \text{Schwierigkeit}}{\text{Produktiviät}}$$

Da die Kosten unmittelbar vom Aufwand abhängen, wird nicht zwischen Aufwandsschätzung und Kostenschätzung unterschieden. Keine der drei Größen ist zu Beginn genau bekannt.

Die Schätzverfahren, die in den letzten fünfzig Jahren entstanden sind, unterscheiden sich in verschiedenerlei Hinsicht:

1. Wird der Aufwand (im engeren Sinne des Wortes) geschätzt oder wird er auf der Basis geschätzter Größen berechnet? Im ersten Fall sprechen wir von *Expertenschätzungen*, im zweiten von *algorithmischen Schätzungen*.
2. Wird der geschätzte Aufwand direkt in den gesuchten Zahlen ausgedrückt (also in Geldbeträgen oder in Entwicklermonaten) oder in irgendwelchen Punkten, die vorerst offen lassen, welche konkreten Werte aus der Schätzung folgen. Inzwischen wurden viele solcher Punkte definiert, beispielsweise Function Points, Object Points, Class Points, Use Case Points, Story Points. Wir sprechen dann von *Bewertungspunkten* im Gegensatz zu *absoluten Zahlen*.
3. Wird die Kostenschätzung in kurzen Abständen wiederholt oder wird nur einmal (eventuell zwei-, dreimal) zu Beginn des Projekts geschätzt? Wir bezeichnen die ersten als *Routineschätzungen*, die zweiten als *Projektschätzungen*.

Die in der Praxis angewandten Schätzverfahren können grob nach den oben beschriebenen Kriterien geordnet werden. Tabelle 8–1 gibt eine Übersicht.

	Experten-schätzung	algorithm. Schätzung	Routine-schätzung	Projekt-schätzung	Punkt-bewertung	Absolute Zahlen
Delphi	X			X		X
Function Points		X		X	X	
COCOMO		X		X		X
Agile Schätz-verfahren	X		X	X	X	

Tab. 8–1 *Einordnung von Schätzverfahren*

In der Literatur findet man auch das Begriffspaar »Top-down-« und »Bottom-up-Schätzung«. Die Bottom-up-Schätzung geht von elementaren Programmbestandteilen wie Code-Zeilen aus. (*Analogie: Ich will im Restaurant essen und frage, was das Essen kostet, wenn ich die einzelnen Speisen vorgebe.*) Umgekehrt wäre ein Vorgehen, das von den vorgegebenen Gesamtkosten auf die erreichbare Funktionalität und Qualität schließt, ein Top-down-Verfahren. (*Was alles kann ich für diesen Betrag essen?*) Planungen in großen Unternehmen können mit einer Top-down-Schätzung beginnen. Anschließend wird bottom-up geschätzt. Bei Software-Projekten ist aber zu Beginn der Planung nicht oder nicht ausreichend klar, was entwickelt werden soll. Erst die Spezifikation liefert den Input für die Bottom-up-Schätzung. Der Top-down-Schätzung fehlt die Substanz. »*Wie viel Software kann ich für 10.000 € bekommen?*« Das funktioniert nicht.

8.4.3 Kalibrierung der Verfahren

Wenn man innerhalb einer Organisation eine gewisse Kontinuität erreicht hat, kann man viele Erfahrungen von Projekt zu Projekt übertragen. Doch das neue Projekt hat Besonderheiten: Das Arbeitsgebiet ist neu oder der Kunde ist kaum erreichbar oder es gibt Sprachprobleme usw. usw. Um nicht die Aufwandsschätzung wieder bei null zu beginnen, wird das Modell der Aufwandsschätzung kalibriert. Dadurch hat man die Möglichkeit, mit einem stabilen Modell mehrere (Teil-)Projekte durchzuführen.

Die Kalibrierung leistet ein simpler Faktor. Natürlich stellt sich die Frage, woher er kommt. Bei algorithmischen Verfahren sind dafür spezielle Faktoren vorgesehen, die Kalibrierung ist quasi eingebaut. Bei Verfahren mit Bewertungspunkten versteckt sich die Kalibrierung meist im letzten Schritt der Schätzung, wenn aus Punkten reale Kosten und Aufwände werden. Wie auch immer kalibriert wird: Die Beteiligten sollten diesen Schritt bewusst gehen und dokumentieren.

Nachfolgend stellen wir populäre Ansätze für die Expertenschätzung und für die algorithmischen Kostenschätzung vor.

8.5 Einige Verfahren zur Aufwandsschätzung

8.5.1 Die Expertenschätzung

Die Kosten eines Projekts durch einen oder besser mehrere Experten schätzen zu lassen, ist ein naheliegender, bewährter Ansatz. Er wird oft angewendet, allerdings meist nicht ausreichend systematisch; die präzise Dokumentation der Daten unterbleibt, sodass am Ende oft der schwarze Peter herumgeschoben wird: *»Du hast mit deinen Zahlen weit danebengelegen.« »Als ich geschätzt habe, sah die Planung noch völlig anders aus.« »Das war ja nur die Schätzung für ein Teilprojekt.«* Damit haben die Schätzungen kaum Nutzen. Jede Schätzung sollte mit den Voraussetzungen, dem Zeitpunkt, den schätzenden Personen und allen Vorbehalten und Einschränkungen dokumentiert werden. Das wirkt auch allzu leichtfertigen Prognosen entgegen.

8.5.2 Das Delphi-Verfahren

In den Sechzigerjahren entstand in der RAND Corporation (Santa Monica, CA) das *Delphi-Verfahren* als Methode der Zukunftsforschung (Brown, 1968). Es handelt sich um eine verfeinerte Variante der Expertenschätzung, die den Zweck hat, einerseits soziale Effekte der gegenseitigen Beeinflussung auszuschließen, andererseits Schätzergebnisse rückzukoppeln und die Gelegenheit zu Korrekturen zu bieten. Dazu wird eine moderierte Sitzung mit den Experten abgehalten, die folgendermaßen strukturiert ist:

1. Der Moderator erläutert zu Beginn die Aufgabe, deren Aufwand geschätzt werden soll, und verteilt, wenn sinnvoll, begleitende Unterlagen.
2. Die Experten schätzen anschließend getrennt und geben ihre Schätzungen dem Moderator.
3. Der Moderator fasst die Schätzungen zusammen und anonymisiert damit die Aussagen und Zahlen, die Ergebnisse teilt er den Experten mit.
4. Die Experten können nun – weiterhin separat – ihre Schätzungen, vor allem die Abweichungen von den anderen Schätzungen, begründen oder korrigieren.

Dieser Prozess wird wiederholt, bis die Abweichungen nach Meinung der Experten akzeptabel sind. Das Ergebnis der Schätzung ist der Median der Einzelschätzungen. Fairley (1985) beschreibt die Anwendung des Delphi-Verfahrens zur Aufwandsschätzung.

Boehm und Farquhar entwickelten dieses Verfahren weiter und nannten es Wideband-Delphi-Verfahren (Boehm, 1981), da es deutlich mehr Interaktionen und auch mehr Kommunikation zwischen den Teilnehmern einer Sitzung vorsieht.

8.5.3 COCOMO

COCOMO (Constructive Cost Model) ist ein algorithmisches Verfahren zur Kostenschätzung von Barry W. Boehm. Allen Rechnungen liegt die geschätzte Programmgröße zugrunde. Mit Formeln, die Boehm aus großen Mengen archivierter Projektdaten abgeleitet hat, können daraus der Aufwand und die Entwicklungsdauer errechnet werden. Spezielle günstige oder ungünstige Umstände, z. B. extreme Sicherheitsanforderungen oder ein überdurchschnittlich erfahrenes Projektteam, können durch Einflussfaktoren berücksichtigt werden, deren Werte man umfangreichen Tabellen entnimmt.

Die erste Version von COCOMO (Boehm, 1981), die inzwischen als COCOMO 81 bezeichnet wird, geht von einer Entwicklung nach dem Wasserfallmodell aus (siehe Abschnitt 9.1.3). Das zu schätzende Projekt wird einer von drei Schwierigkeitsklassen zugeordnet, für die jeweils angepasste Berechnungsformeln zur Verfügung stehen.

COCOMO 81 war Ende des 20. Jahrhunderts in zweierlei Hinsicht veraltet: Zum einen stimmten einige Parameter nicht mehr, zum anderen waren einige Einflussfaktoren bedeutungslos geworden, während andere fehlten. Ein Beispiel für einen obsoleten Faktor ist die *turn around time*, die Wartezeit, bis ein Programm im Batch-Betrieb ausgeführt wird. Dagegen war 1981 an Projekte, die an mehreren Standorten kooperativ durchgeführt werden (*multi-site development*), nicht zu denken. Neu war auch die Vielfalt der Vorgehensmodelle und die Idee der Prozessbewertung und -verbesserung. Zusammen mit einer langen Reihe von Koautoren hat Boehm das ursprüngliche Verfahren revidiert, aktualisiert und erweitert (Boehm et al., 2000). Das Ergebnis nannte er COCOMO II. Hier ist nachfolgend mit COCOMO immer COCOMO II gemeint.

COCOMO definiert drei verschiedene Modelle zur Aufwandsabschätzung, die in unterschiedlichen Entwicklungssituationen und zu unterschiedlichen Zeitpunkten im Projekt eingesetzt werden können:

- das *Application Composition Model* für Systeme, die weniger programmiert als konfiguriert oder – etwa mithilfe eines GUI-Builders – generiert werden,
- das *Early Design Model* als Adaption des Function-Point-Verfahrens (siehe Abschnitt 8.5.4), das sich bereits einsetzen lässt, wenn die Anforderungen geklärt sind,
- das *Post-Architecture Model*, das erfordert, dass die Software-Architektur definiert ist und dass der Umfang der Software-Komponenten abgeschätzt werden kann.

COCOMO basiert auf einer Schätzung der Programmgröße S (size), angegeben in kSLOC (kilo Source Lines of Code), das sind alle Zeilen bis auf solche, die leer sind oder nur Kommentar enthalten. Wer bereits Größenschätzungen in Function Points besitzt (siehe Abschnitt 8.5.4), kann diese in Source Lines of Code umrechnen, wie in Boehm et al. (2000) beschrieben.

Die COCOMO-Grundgleichung

COCOMO stellt wenige Formeln zur Verfügung; die beiden wichtigsten sind die, um den Aufwand E (effort) und darauf aufbauend die Entwicklungsdauer T zu bestimmen. Dies ist die Basisgleichung, um die geschätzten Personenmonate (PM) zu berechnen.

$$E = \pi \cdot S^{\varepsilon} \cdot \mu$$

Darin repräsentiert π die Produktivität, der Faktor μ repräsentiert die Schwierigkeit des zu entwickelnden Systems.

Anmerkung: Diese Formel entspricht dem bereits diskutierten Zusammenhang: Aufwand = Umfang · Schwierigkeit / Produktivität. Die Konstante π ist also der Kehrwert des genutzten Produktivitätswerts.

Welche Interpretation können wir mit dieser Berechnung verbinden? Die beiden Zahlen π und ε (und die vielen Kennwerte, auf die wir noch kommen) wurden durch Auswertung der Daten von 161 Projekten ermittelt, wobei die Mehrzahl der Projekte (101) zwischen 1995-1999 abgeschlossen wurden (Nguyen, Huang, Boehm, 2011). Die Analyse dieser Projektdaten lieferte für π den Wert 2,94; dieser entspricht einer Produktivität von 340 SLOC/PM. Dieser Wert ist heute sicherlich höher, er kann im Sinne der Kalibrierung durch einen aktuellen Wert ersetzt werden. Für den Exponenten ε wurde der Wert 1,1 ermittelt. Hinter dieser Analyse steht also keine Theorie, sondern eine gewaltige Menge von Beobachtungen. Wir verwenden in den nachfolgenden Beispielen die Originalwerte aus Boehm et al. (2000) für π und ε.

Wenn wir beispielsweise geschätzt haben, dass das geplante Programm 12000 Code-Zeilen haben wird, hat S den Wert 12 kSLOC. Setzen wir diesen Wert in die Formel ein und nehmen wir an, dass die Schwierigkeit des Programms den Aufwand nicht erhöht ($\mu = 1$), so ergibt sich für den geschätzten Aufwand

$$E = 2{,}94 \cdot 12^{1{,}1}\ \text{PM} = 2{,}94 \cdot 15{,}4\ \text{PM} = 45{,}2\ \text{PM}.$$

Einzig die Entscheidung, den Umfang S mit einem Exponenten knapp über 1 zu versehen, drückt eine Vermutung aus: Wenn wir am Parameter S drehen, ändert sich der Aufwand nicht proportional, sondern progressiv. Verdoppeln wir im Beispiel den Umfang, setzen also 24 kSLOC ein, dann erhöht sich der Aufwand von 15,4 nicht nur auf 30,8 PM, sondern auf 33 PM. Das bedeutet: Größe schafft zusätzliche Komplexität, und die macht Aufwand.

Erweiterung der COCOMO-Grundgleichung

Was wir bis hierher gezeigt haben, ist der Kern von COCOMO. Aber für die Praxis taugt er nicht, weil er blind ist für fast alle Einflüsse, denen das Projekt unterliegt. Ob das Entwicklerteam besonders erfahren oder unerfahren ist, ob das Projekt unter besonderem Zeitdruck steht, ob das Team spezielle Kompetenzen für

dieses Projekt hat, welche Rolle die Wiederverwendung spielt: Diese und viele andere Einflüsse müssen berücksichtigt werden.

Das geschieht, indem der lineare Korrekturfaktor für die Schwierigkeit μ sowie der Exponent ε variabel werden. Wenn ein Projekt in jeder Hinsicht durchschnittlich ist, behält ε den Wert 1,1, und μ bleibt mit dem Wert 1 neutral.

In den Exponenten ε geht die Summe von fünf Einflussfaktoren ein. Das COCOMO-Buch enthält dazu die Tabelle 8–2, die einerseits die Einstufung der Faktoren unterstützt, andererseits die den Stufen zugeordneten Kennwerte liefert. Es wird also für jeden Einflussfaktor die Beschreibung gesucht, die am besten passt. Beispiel: »Der Einflussfaktor TEAM steht auf Stufe *very low*, wenn ...«. Darauf folgt eine Charakterisierung. Trifft sie zu, wird der Tabelle der Kennwert 5,48 entnommen.

Faktor (F^{ε})		very low	low	nom.	high	very high	extra high
PREC	Erfahrung mit ähnlichen Projekten	6,20	4,96	**3,72**	2,48	1,24	0,00
FLEX	Flexibilität bei der Wahl des Entwicklungsprozesses	5,07	4,05	**3,04**	2,03	1,01	0,00
RESL	Kompetenz im Risikomanagement und Architekturentwurf	7,07	5,65	**4,24**	2,83	1,41	0,00
TEAM	Höhe des teaminternen Kommunikationsaufwands	5,48	4,38	**3,29**	2,19	1,10	0,00
PMAT	Fähigkeit, den Entwicklungsprozess effektiv und effizient umzusetzen	7,80	6,24	**4,69**	3,12	1,56	0,00

Tab. 8–2 *Werte der Einflussfaktoren für den Exponenten ε*

Die Summe dieser Einflussfaktoren liegt zwischen 0 und 31,62. Diese wird durch 100 geteilt, das Ergebnis ist der Progressionswert δ. Vermehrt um die empirisch ermittelte Progressionsverschiebung ω = 0,91 wird daraus der Exponent ε. Dementsprechend kann ε Werte zwischen 0,91 und 1,23 annehmen. Wenn alle Faktoren Nominalwerte haben, hat ε den Standardwert 1,1.

$$\delta = 0,01 \cdot \sum_{j=1}^{5} F_j^{\varepsilon}$$

$$\varepsilon = \delta + \omega \qquad \text{mit} \quad \omega = 0,91$$

Ganz ähnlich geht man vor, um die Werte der 17 Einflüsse zu bestimmen, deren Produkt den linearen Korrekturfaktor für die Schwierigkeit μ ergeben. Tabelle 8–3 zeigt einen Auszug dieser Einflüsse.

Faktor		very low	low	nom.	high	very high	extra high
RELY	Zuverlässigkeit des Systems	0,82	0,92	**1,00**	1,10	1,26	
CPLX	Komplexität des Systems	0,73	0,87	**1,00**	1,17	1,34	1,74
PCAP	Fähigkeiten der Programmierer	1,34	1,15	**1,00**	0,88	0,76	
SCED	Bedingungen an die Entwicklungsdauer	1,43	1,14	**1,00**	1,00	1,00	

Tab. 8–3 *Kennwerte im Post-Architecture Model (4 von 17 Faktoren)*

Man sieht, dass hier »nominal« gleichbedeutend ist mit dem neutralen Faktor 1. Die meisten Faktoren steigen, wenn die Einstufung steigt. Das zeigt der Faktor RELY. Wenn die Zuverlässigkeit hoch sein muss, ist es auch der Aufwand. Einige Faktoren sinken mit höherer Einstufung, z. B. PCAP: Gute Leute kommen mit weniger Aufwand aus.

Und dann gibt es noch den speziellen Faktor SCED, die verfügbare Zeit. Die sogenannte nominale Zeit T_{nom}, angegeben in Kalendermonaten (KM), ist für das Projekt optimal. Sie wird mit der Formel

$$T_{nom} = \tau \cdot E^{\frac{\delta + 1,4}{5}}$$

berechnet. Darin ist τ wiederum eine empirisch ermittelte Konstante mit dem Wert 3,67. Auch die Verrechnung des Progressionswerts δ im Exponenten ist ein Ergebnis der von Boehm analysierten Projekte.

In unserem Beispiel, einem Programm der Größe 12 kSLOC, hat der nominale Wert für den Exponenten $\varepsilon = 1,1$ zu einem geschätzten Aufwand von 45,2 PM geführt. Da wir ε kennen, berechnet sich der Progressionswert δ nach der Formel $\delta = \varepsilon - \omega$. Wir erhalten den Wert $\delta = 1,1 - 0,91 = 0,19$. Setzen wir diesen und die anderen Werte in die Formel ein, dann ist die nominale Entwicklungsdauer

$$T_{nom} = 3,67 \cdot 45,2^{\,0,318}\ \text{KM} = 3,67 \cdot 3,36\ \text{KM} = 12,3\ \text{KM}.$$

Demnach müssen wir durchschnittlich etwa 3,5 Entwickler einsetzen. Mehr Zeit als die nominale Dauer nützt nichts. Aber weniger Zeit schadet erheblich. Was 3,5 Entwickler in 12 Monaten entwickeln können, lässt sich mit 7 Entwicklern nicht in 6, sondern etwa in 9 Monaten fertigstellen. (Und noch schneller geht es, wie Boehm lehrt, überhaupt nicht. Noch mehr Leute behindern sich nur noch.)

Natürlich ist es mühsam und umständlich, die Kennwerte aus dem Buch zu holen und das Modell von Hand durchzurechnen. Dafür gibt es COCOMO-Rechner. Der Schätzer muss die Einstufungen vornehmen, den Rest macht das Werkzeug. Wenn man unter »COCOMO Calculator« sucht, findet man solche Rechner.

Wiederverwendung und instabile Anforderungen

Die Wiederverwendung hat zunehmend große Bedeutung. Wird ein erheblicher Teil des zu entwickelnden Systems aus einem älteren System oder aus Bibliotheken übernommen, so fällt als Aufwand nur die Anpassung der übernommenen Teile an.

COCOMO verwendet in diesem Fall nicht den Umfang S, sondern einen entsprechend angepassten Umfang S‘. Beispiel: Für ein Softwareprojekt ist der Umfang auf 12 kSLOC geschätzt. Ein Viertel davon kann übernommen werden, also 3 kSLOC. Die Anpassung erfordert 20 % des Aufwands, der für die Neuentwicklung erforderlich gewesen wäre, also den Aufwand für 0,6 kSLOC. Damit reduziert sich der Umfang auf

$$S‘ = 12\ \text{kSLOC} - 3\ \text{kSLOC} + 0{,}6\ \text{kSLOC} = 9{,}6\ \text{kSLOC}.$$

Analog werden Projekte behandelt, bei denen die Anforderungen instabil sind. In diesem Fall gibt es einen Aufschlag, der die Mehrarbeit durch schwankende Anforderungen nachbildet. Beispiel: Wird geschätzt, dass 10 % der Anforderungen instabil sind, dann erhöht sich der Wert von S‘ um 10 % auf

$$S‘ = 9{,}6\ \text{kSLOC} \cdot (1 + 0{,}1) = 10{,}56\ \text{kSLOC}.$$

Dieser Wert wird dann in der Formel E für S verwendet.

Was-wäre-wenn-Analysen mit COCOMO

COCOMO definiert ein explizites Modell, das der Rechnung zugrunde liegt. Dieses kann für Was-wäre-wenn-Analysen genutzt werden, indem gezielt Werte von Parametern verändert und die berechneten Zahlen bewertet werden. Nachfolgend zeigen wir das exemplarisch.

Wir gehen wieder von unserem Beispielprojekt aus, dessen Umfang wir mit 12 kSLOC abgeschätzt hatten. Wenn dieses Projekt ohne besondere Einflüsse realisiert werden kann, dann liefert COCOMO den Aufwand E = 45,2 PM. Da die Anforderungen an unser System bereits spezifiziert sind und nicht mehr geändert werden können, haben wir folgende Fragen:

1. Welchen Einfluss haben die Qualifikationen des Teams auf den Aufwand?
2. Wie stark verändert sich der Aufwand, wenn das Team stark verstreut oder eng konzentriert am Projekt arbeitet?

Tabelle 8–4 zeigt die Einflussfaktoren ACAP, PCAP, PCON und APEX, die in COCOMO die Qualifikationen des Teams repräsentieren. Wir wählen die low-Werte für ein Team, das wenig Erfahrung und Qualifikationen mitbringt, die high-Werte für ein sehr erfahrenes Team. Der Einflussfaktor SITE wird verwendet, um die zweite Frage zu beantworten. Hier nutzen wir für die Analyse den low- und den very-high-Wert.

Faktor		very low	low	nom.	high	very high	extra high
ACAP	Fähigkeiten der Analytiker	1,42	1,19	**1,00**	0,85	0,71	
PCAP	Fähigkeiten der Programmierer	1,34	1,15	**1,00**	0,88	0,76	
PCON	Kontinuität der eingesetzten Mitarbeiter	1,29	1,12	**1,00**	0,90	0,81	
APEX	Erfahrungen im Anwendungsbereich	1,22	1,10	**1,00**	0,88	0,81	
SITE	Grad der Verteiltheit der Entwicklungs-standort	1,22	1,09	**1,00**	0,93	0,86	0,80

Tab. 8–4 *Kennwerte für eine Was-wäre-wenn-Analyse*

Für jede Frage berechnen wir zwei Szenarien; deren Aufwandswerte sowie die Abweichungen zum nominalen Aufwand (E = 45,2) sind in der Tabelle 8–5 enthalten.

Szenario	E_S	Diff. zu E
Es wird ein unerfahrenes Team eingesetzt. (low-Wert für: ACAP, PCAP, PCON und APEX)	76,2 PM	+ 68 %
Es wird ein sehr erfahrenes Team eingesetzt. (high-Wert für: ACAP, PCAP, PCON und APEX)	26,8 PM	- 41%
Das Team arbeitet räumlich sehr verteilt. (low-Wert für SITE)	49,3 PM	+ 8 %
Das Team arbeitet ein einem Ort zusammen. (very-high-Wert für SITE)	38,9 PM	- 16 %

Tab. 8–5 *Ergebnisse einer Was-wäre-wenn-Analyse*

Die Zahlen in der letzten Spalte zeigen sehr deutlich, dass die Fähigkeiten der Teammitglieder einen weit stärkeren Einfluss haben als die Konzentration des Teams an einem Ort. Wenn also die personelle Situation so ist, dass die erfahrenen Leute nicht an einem Ort versammelt werden können, dann sollte man nicht auf weniger erfahrene Leute zurückgreifen, sondern sich mit der räumlichen Verteilung abfinden.

8.5.4 Das Function-Point-Verfahren

Wie alle algorithmischen Verfahren basiert auch das Function-Point-Verfahren auf einem Modell, das die relevanten Einflussgrößen miteinander verbindet, die den Aufwand und damit auch die Kosten eines Projekts beeinflussen. Der Umfang eines Programms wird je nach Modell in verschiedenen (geschätzten) Kennzahlen angegeben. So kann beispielsweise die geschätzte Anzahl der Code-Zeilen oder die Anzahl der zu realisierenden Funktionen (Function Points, FPs) verwendet werden.

Die Idee der Function Points wurde 1979 von A. Albrecht, IBM, publiziert (Albrecht, 1979) und seitdem von verschiedenen Autoren und Institutionen weiterentwickelt (z. B. Albrecht, Gaffney, 1983). Das Verfahren unterscheidet sich von COCOMO prinzipiell wie folgt:

- Basis der Schätzung ist nicht die Zahl der Code-Zeilen, sondern der Umfang des Programms, ausgedrückt in den zu implementierenden Funktionen. Die Software wird also nicht aus der Sicht der Implementierung, sondern aus der des Benutzers gesehen.
- Da die Software-Architektur nicht vorausgesetzt wird, können die Einflussfaktoren nur global, nicht nach Modulen differenziert berücksichtigt werden.
- Das Verfahren ist arithmetisch einfacher, ein exponentieller Zusammenhang wie in COCOMO kommt nicht vor; stattdessen muss eine progressive Kennlinie (Aufwand über FPs) experimentell ermittelt werden.

Vereinfacht geht man dabei wie folgt vor: Alle für das Problem relevanten Daten, d. h. die logischen Datenbestände (Dateien) und alle Ein- und Ausgaben der zu realisierenden Vorgänge, werden erfasst und den folgenden Kategorien zugeordnet:

- Externe Eingabe
- Externe Ausgabe
- Externe Abfrage
- Interne Anwenderdaten
- Externe Referenzdaten

Eine Eingabe kann von der Tastatur, von einem Datenträger oder aus einer anderen Quelle kommen. Entsprechend gehen Ausgaben auf den Bildschirm, auf einen Drucker oder an ein anderes Gerät. Abfragen sind spezielle Ausgaben, die keine weitere Bearbeitung (z. B. eine Mittelwertbildung) der gespeicherten Informationen erfordern und diese auch nicht verändern.

Anschließend wird der Schwierigkeitsgrad jedes Datums als niedrig, mittel oder hoch bewertet. Durch Tabellen ist für jeden Typ und für jede Schwierigkeitsstufe (z. B. für »externe Eingabe, mittel«) eine Punktzahl gegeben. Diese Punkte werden addiert; das Ergebnis liefert die sogenannten *Unadjusted Function Points* (siehe Abb. 8–7).

Nun werden 14 Einflussfaktoren (General System Characteristic, GSC) wie Verflechtung mit anderen Projekten, dezentrale Datenverwaltung, Transaktionsrate, komplexe Verarbeitungslogik, Wiederverwendbarkeit, Konvertierungen oder Benutzerfreundlichkeit betrachtet und mit 0 bis 5 bewertet (0 = kein Einfluss, 5 = starker Einfluss), um einen Korrekturfaktor (Value Adjustment Factor, VAF) zu bestimmen. Dieser wird wie folgt berechnet:

$$\mathrm{VAF} = 0,65 + \frac{\sum_{i=1}^{14} \mathrm{GSC}_i}{100}$$

Dadurch ergibt sich ein Wert zwischen 0,65 und 1,35, der zu einer Korrektur von -35 % bis +35 % führt.

Typ	Komplexität			Summe
	niedrig	mittel	hoch	
Eingabe	__ · 3 =	__ · 4 =	__ · 6 =	
Ausgabe	__ · 4 =	__ · 5 =	__ · 7 =	
Abfrage	__ · 3 =	__ · 4 =	__ · 6 =	
Anwenderdaten	__ · 7 =	__ · 10 =	__ · 15 =	
Referenzdaten	__ · 5 =	__ · 7 =	__ · 10 =	
Unadjusted Function Points		UFP		
Value Adjustment Factor		VAF		
Adjusted Function Points		AFP = UFP · VAF		

Abb. 8–7 *Function-Point-Berechnungsschema*

Das Produkt aus Unadjusted Function Points und Korrekturfaktor liefert die *Adjusted Function Points*. Tabelle 8–7 zeigt ein einfaches Schema, um nach dieser Vorgehensweise die Function Points zu bestimmen.

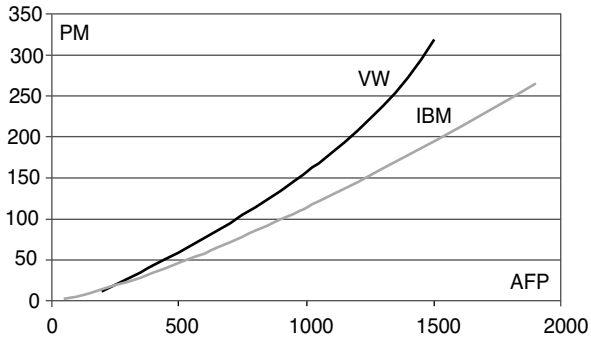

Abb. 8–8 *IBM- und VW-Kurve für die Umrechnung von AFPs in PM gemäß Noth, Kretzschmar (1984) und Knöll, Busse (1991)*

Der so ermittelte Function-Point-Wert wird schließlich mithilfe einer Kurve oder Tabelle, deren Grundlage die Nachkalkulation alter Projekte bildet, in Personenmonate umgesetzt (siehe Abb. 8–8).

Auf Basis der Erfahrungen mit neuen Projekten wird die Umrechnungstabelle stetig aktualisiert. Dazu können die neuen Wertepaare (FP/PM) zusätzlich in die Tabelle aufgenommen oder gegen die ältesten Wertepaare ausgetauscht werden.

8.5.5 Die Suche nach einem Standard für die funktionale Größe

Schon in den Neunzigerjahren wurde das Function-Point-Verfahren weiterentwickelt und modifiziert, um den Anwendungsbereich zu erweitern (von Informationssystemen auf Echtzeitsysteme und technisch-wissenschaftliche Software) und um die Genauigkeit der Schätzungen zu verbessern.

Inzwischen ist eine ganze Reihe von Varianten entstanden. Einige davon, z. B. die »feature points«, die »full function points«, die »Mark II function points« und die »3D function points«, werden in der neueren Literatur kaum noch erwähnt, sind also mutmaßlich ohne Bedeutung. Die unbefriedigende Literaturlage veranlasst uns, hier keine einzelnen Quellen anzugeben. Denn vielfach sind Urheberschaft und Status unklar. Auch ist kaum erkennbar, wie weit es wirklich unterschiedliche Konzepte gibt oder nur Konkurrenz zur Abgrenzung führt. Bei Bedarf findet man im Web mehr oder minder viele Spuren der Verfahren.

Andere Varianten genießen einige Aufmerksamkeit und haben auch namhafte Vertreter. Die ursprüngliche Idee wird vor allem von der IFPUG vertreten, der International Function Point Users Group. Das Buch von Poensgen und Bock (2005) basiert auf dieser Sichtweise.

Gleichzeitig hat sich der Schwerpunkt der Methoden von der Kostenschätzung zur Größenbeschreibung (»*Sizing*«) verschoben. Nun geht es darum, ein Verfahren zu definieren, das eine neutrale, objektive Charakterisierung des Software-Umfangs (*functional size*) liefert; die Kostenschätzung wird nur noch als Zusatzfunktion betrachtet. Natürlich wäre eine allgemein anerkannte Metrik der Software-Größe außerordentlich nützlich.

Das zuständige Normungskomitee der ISO hat 1998 einen (Meta-)Standard für die Größenbeschreibung verabschiedet, ISO 14143 *Functional Size Measurement* mit sechs Teilen (*Functional Size Measurement Concepts*, *Compliance*, *Verification, Reference Model*, *Software Domains* und *Guide to 14143 & related standards*). Dieser Standard legt fest, welche Anforderungen ein Verfahren zur Größenmessung erfüllen muss. Eine ganze Reihe von Vorschlägen wurde dann als kompatibel mit ISO 14143 ebenfalls standardisiert:

- ISO/IEC 19761, die COSMIC-Methode (Common Software Measurement International Consortium)
- ISO/IEC 20926, die IFPUG-Methode
- ISO/IEC 20968, die Mark-II-Methode (MkII)
- ISO/IEC 24570, die Methode der Nesma (Nederlandse Software Metrieken Associatie)

Bei Verfahren, die (wie die IFPUG-Methode) auch die Kostenschätzung einschließen, erfasst die Normung nur den Teil, der den funktionalen Umfang betrifft.

Offensichtlich sind wir von einem einzigen allgemein akzeptierten Standard für die Beschreibung der funktionalen Größe noch weit entfernt.

8.5.6 Einsatz der Schätzverfahren

In der Literatur gibt es eine Reihe von Studien, die verschiedene Aspekte der Schätzverfahren untersuchen, beispielsweise deren Genauigkeit, aber auch deren Einsatz in der Industrie. Auf Basis ihrer umfangreichen Untersuchung folgern beispielsweise Jørgensen und Shepperd (2007), dass algorithmische Schätzverfahren im Vergleich zu Expertenschätzungen deutlich seltener angewendet werden. Sie sehen auch keine Anhaltspunkte dafür, dass die Schätzgenauigkeit durch die Verwendung algorithmischer Schätzverfahren verbessert würde.

Von den algorithmischen Verfahren ist das Function-Point-Verfahren in der Praxis am weitesten verbreitet, vor allem im Bereich der betriebswirtschaftlichen und kaufmännischen Software. Entsprechend ist es in der Literatur (z. B. Poensgen, Bock, 2005) und im Web sehr präsent. COCOMO hat sich dagegen in der Industrie nicht wirklich etablieren können. Das mag zum einen daran liegen, dass der Aufwand und auch die Dauer für eine unternehmensspezifische Kalibrierung erheblich sind. Zum anderen gibt es, im Gegensatz zum Function-Point-Verfahren, keine Community, die COCOMO als Schätzverfahren gefördert und vermarktet hat. Darum gibt es weder Experten in Unternehmen noch freie Berater, die dieses Verfahren einsetzen und empfehlen.

8.6 Schätzung in agilen Projekten

Agile Prozesse zur Software-Entwicklung werden in vielen Unternehmen eingesetzt; sie haben die klassischen nicht agilen Entwicklungsprozesse zu einem großen Teil verdrängt. Diese Prozesse und deren Terminologie werden in Abschnitt 10.6 eingeführt. In Zusammenhang mit der Aufwandsschätzung reicht es aus, drei Begriffe zu erklären:

- *Scrum* ist ein sehr populäres Managementmodell für Software-Projekte.
- Ein *Sprint* ist eine inhaltlich und zeitlich scharf abgegrenzte Phase in einem Scrum-Projekt, die typischerweise nach zwei, maximal nach sechs Wochen beendet ist.
- Das *Backlog* verzeichnet die Menge aller Teilaufgaben, die noch nicht bearbeitet wurden. Für das Schätzen sollte das Backlog in Form eines Kastens mit Karteikarten vorliegen. Jede Karte repräsentiert dann eine Teilaufgabe.

In agilen Projekten, insbesondere wenn sie nach Scrum durchgeführt werden, sind die Teams dafür verantwortlich, für einen erfolgreichen Abschluss zu sorgen. Die Aufwandsschätzung ist darum eine der elementaren Tätigkeiten in agilen Teams. Geschätzt werden immer Einträge, die im Backlog enthalten sind. Achtung: Wenn wir davon sprechen, dass Einträge geschätzt werden, dann ist die Formulierung so zu verstehen, dass der für Einträge des Backlogs erforderliche

Realisierungsaufwand geschätzt wird. Folgende Fragen soll die Schätzung beantworten:

- Welcher Aufwand wird benötigt, um alle Einträge des Product Backlogs oder des Projekt-Backlogs zu implementieren?
- Welche Backlog-Einträge kann das Team im nächsten Sprint realisieren?

Es gibt eine Vielzahl von Anleitungen für agile Schätzverfahren im Internet. Eine gute Einführung in die wichtigsten Verfahren geben Wolf und Roock (2021), auch Wiechmann und Röpstorff (2022).

8.6.1 Agile Routine-Schätzungen

Die meisten agilen Verfahren für Routineschätzungen schätzen den relativen Schwierigkeitsgrad der Backlog-Einträge in *Story Points*. Story Points sind eine teamspezifische abstrakte Größe. Der Wert 5 kann beispielsweise für Team A bedeuten, dass es im Schnitt sechs Einträge dieses Schwierigkeitsgrads in einem zweiwöchigen Sprint realisieren kann. Team B kann im Unterschied dazu sieben Einträge, die mit 5 geschätzt wurden, in einem zweiwöchigen Sprint umsetzen.

Der Schwierigkeitsgrad eines Eintrags bezieht sich auf den Umfang und die Komplexität, aber auch auf die technischen Risiken. Es geht also im Kern darum, die zu schätzenden Einträge relativ zueinander zu bewerten, bezogen auf einen Referenzwert. Dieser wird festgelegt, indem ein repräsentativer und nicht allzu schwieriger Eintrag des Backlogs ausgewählt und (willkürlich) bewertet wird (z. B. mit 3 oder 5). Gegen diesen Referenzeintrag werden alle anderen Backlog-Einträge geschätzt. Wird ein Eintrag als einfacher angesehen, erhält er einen niedrigeren Wert, ist er schwieriger, erhält er einen höheren Wert.

Nachfolgend stellen wir exemplarisch das Planning-Poker-Verfahren vor, das genutzt wird, um die Backlog-Einträge für den nächsten Sprint zu schätzen.

Planning Poker[1] ist eine agile Variante einer Expertenschätzung; sie wurde 2002 von Grenning (2002) vorgestellt, später detailliert in Cohn (2005) beschrieben. Der Name suggeriert, dass es sich um ein Kartenspiel handelt. In der Tat werden spezielle Karten verwendet; ein Spiel ist es aber natürlich nicht.

Ein Planning-Poker-Kartensatz enthält Karten mit nichtlinear steigenden Zahlen, die bei den niedrigeren Werten den Fibonacci-Zahlen entsprechen. Häufig werden Karten mit den folgenden Werten verwendet: 1, 2, 3, 5, 8, 13, 20, 40, 100. Die steigende Schrittweite trägt der Tatsache Rechnung, dass große Einträge nicht so genau geschätzt werden können wie kleine, überschaubare. Darum ist es unerheblich, ob ein Eintrag mit dem Wert 42 oder 44 bewertet wird. Es gibt keine allgemein akzeptierte Interpretation für diese Werte. Eine mögliche ist in Tabelle 8–6 angegeben.

1 Planning Poker ist ein eingetragenes Warenzeichen der Firma Mountain Goat Software, LLC.

Wert	Interpretation
1	Eine einfache Anforderung, kaum der Rede wert.
2	Eine etwas komplexere Anforderung, die wir schnell umsetzen können.
3	Eine deutlich komplexere Anforderung, die aber immer noch gut überschaubar ist.
5	Eine »normale« Anforderung, die wir sicher umsetzen können.
8	Hui, diese Anforderung benötigt schon eine Menge Arbeit.
13	Diese Anforderung könnte für einen Sprint zu umfangreich sein. Es bestehen Risiken und Unklarheiten.
20	Diese Anforderung birgt zu viele Risiken und unbekanntes Terrain. Wir müssen sie detaillierter besprechen und wahrscheinlich aufteilen, bevor sie geschätzt und geplant werden kann.
40	Wir erahnen in etwa, worum es gehen könnte. Der Umfang ist aber viel zu groß. Die Anforderung muss auf jeden Fall aufgeteilt werden.
100	Diese Anforderung ist noch nicht einmal ansatzweise sinnvoll bewertbar.

Tab. 8–6 *Mögliche Interpretation von Story-Point-Kennwerten*

Manchmal werden Zahlenkarten auch durch Spezialkarten ersetzt. So bedeutet beispielsweise ein Fragezeichen, dass der Backlog-Eintrag so unklar formuliert ist, dass man ihn nicht schätzen kann. Der Ablauf einer Planning-Poker-Sitzung entspricht schematisch dem des Delphi-Verfahrens; die Rolle des Moderators übernimmt der Scrum Master.

1. Der Product Owner stellt den zu schätzenden Backlog-Eintrag vor und beantwortet Nachfragen dazu.
2. Anschließend schätzen die Teammitglieder den Schwierigkeitsgrad relativ zur Bewertung des gewählten Referenzeintrags, d. h., jeder wählt eine seiner Karten.
3. Dann werden die gewählten Karten gleichzeitig und nicht nacheinander aufgedeckt. Dies ist wichtig, damit die Schätzungen nicht von einzelnen Meinungsträgern beeinflusst werden. Stimmen die Schätzungen überein, wird dieser Wert festgehalten, und die Schätzung ist beendet. Andernfalls begründen die einzelnen Teammitglieder ihre Schätzungen und diskutieren die Unterschiede. Wichtig ist, dass alle Meinungen gehört und ernst genommen werden und in die Schätzung einfließen.

Die Schätzung wird so lange wiederholt, bis ein Konsens gefunden wurde. Oder man bricht ab und nimmt den Durchschnitt der abgegebenen Werte als Schätzwert. Falls ein Eintrag mehrheitlich als extrem schwierig bewertet wird, muss er aufgeteilt werden. Wenn ein Eintrag nicht abgeschätzt werden konnte, weil er zu ungenau formuliert war, muss er überarbeitet und präzisiert werden.

8.6.2 Agile Einzelprojekt-Schätzungen

Soll ein gesamtes Projekt oder ein großes Teilprojekt geschätzt werden, dann liegen die Backlog-Einträge häufig in einer gröberen Form vor, die erst später in User Stories umgeformt werden müssen. Um schon zu diesem frühen Zeitpunkt den Aufwand schätzen zu können, kann beispielsweise das *Team Estimation Game* angewendet werden. Dabei werden alle Backlog-Einträge nacheinander vorgelesen und dann auf einer Pinnwand angebracht.

Der erste Eintrag, d. h. die entsprechende Karteikarte, wird auf der Pinnwand fixiert. Ist der Eintrag auf der nächsten Karte ähnlich anspruchsvoll wie der vorige, so wird er auf gleicher Höhe angepinnt; ein einfacherer Eintrag wird tiefer, ein schwierigerer höher angebracht. Wer eine Karte anbringt, kann ab der dritten Karte zusätzlich eine Karte auf der Pinnwand umhängen. Eine so verschobene Karte wird markiert. Alle Karten, die mehrfach verschoben wurden, werden am Ende von der Wand entfernt. Diese Einträge müssen weiter verfeinert werden, da sie sehr unterschiedlich interpretiert und bewertet werden. Schließlich werden die so entstandenen Gruppierungen von Backlog-Einträgen auf die vom Team in den Routineschätzungen verwendeten Story-Point-Werte abgebildet.

Verfahren dieser Art liefern nur eine erste grobe Abschätzung, sie können dafür aber für große Mengen von Einträgen genutzt werden. Sie können auch verwendet werden, um ein zu groß gewordenes Backlog aufzuräumen. Wichtig ist dabei, dass während der Schätzung möglichst keine Diskussionen entstehen, um die große Menge an Backlog-Einträgen in sehr kurzer Zeit zu bewerten. Das Ergebnis einer solchen Schätzung benötigt der Product Owner beispielsweise für die Release-Planung.

Schließlich müssen die ermittelten Story Points auf klassische Aufwandsgrößen (z. B. Personentage) abgebildet werden. Dazu muss die Produktivität des Teams bekannt sein; sie wird in agilen Projekten als Entwicklungsgeschwindigkeit (*Velocity*) bezeichnet. Velocity ist die durchschnittliche Anzahl der Story Points, die ein Team auf Basis bereits abgeschlossener Sprints pro Sprint realisieren kann. Hat ein Team einen sehr stabilen Velocity-Wert, dann kann recht gut abgeschätzt werden, wie viele Sprints benötigt werden, um alle Backlog-Einträge zu implementieren.

Wenn beispielsweise der Aufwand für das gesamte Backlog eines Projekts mit 350 Story Points bewertet wurde und das Team einen Velocity-Wert von 15 für zweiwöchige Sprints hat, dann kann das Team in ca. 24 Sprints das aktuelle Backlog oder eine vergleichbare Menge an zukünftigen, noch nicht bekannten Einträgen realisieren. Die Laufzeit beträgt entsprechend 48 Wochen. Besteht das Team aus fünf Mitgliedern, dann liegt der Aufwand für dieses Projekt bei ca. 1200 Personentagen.

8.6.3 Agile Multiprojekt-Schätzungen

Muss ein sehr großes Projekt oder sogar ein Projektportfolio auf mehrere Teams aufgeteilt werden, dann kann das Backlog nicht mehr nur von einem Team geschätzt werden. In diesem Fall schätzen die Stellvertreter aller Teams das Backlog; das können Teamleiter oder Product Owner sein. Für diese Schätzung kann auch nicht mehr die teamspezifische Einheit Story Point verwendet werden, man braucht eine teamübergreifende standardisierte Schätzeinheit. Diese Einheit muss so gewählt werden, dass sie für eine effektive Release- und Ressourcen-Planung, aber auch für die Kostenkontrolle genutzt werden kann. Dementsprechend muss sie die Anzahl der vorhandenen Entwickler und deren Produktivität pro Sprint berücksichtigen. Natürlich kann man diese standardisierte Einheit wieder als Story Point bezeichnen, wie das beispielsweise vom Scaled Agile Framework (SAFe) vorgeschlagen wird (SAFe, o. J.). Dies birgt allerdings die Gefahr, dass es zu Missverständnissen kommt. Deshalb ist es durchaus sinnvoll, für die groben Backlog-Einträge eine andere Einheit zu verwenden (siehe das folgende Praxisbeispiel).

8.6.4 Ein Beispiel aus der Praxis

Seit etwa 2010 haben viele Unternehmen den Weg der sogenannten agilen Transformation eingeschlagen oder sind bereits am Ziel, im Unternehmen sukzessive agile Prozesse und Praktiken einzuführen. Dementsprechend sind agile Schätzverfahren mittlerweile weit verbreitet. In den allermeisten Fällen werden auf Basis der publizierten agilen Schätzverfahren unternehmensspezifische Schätzverfahren und Prozesse entwickelt, erprobt und kontinuierlich verbessert. Dies zeigt das folgende Praxisbeispiel:

Die IVU Traffic Technologies AG bietet unter anderem ein Portfolio von Software-Produkten für den Anwendungsbereich Öffentlicher Verkehr an. Die IVU hat viele nationale und internationale Kunden, die diese Produkte langfristig nutzen. Diese Kunden stellen sehr unterschiedliche neue Anforderungen, aber auch die Product Owner entwickeln kontinuierlich neue Ideen. Die Software-Produkte werden von vielen Teams gepflegt und weiterentwickelt, die agil in zweiwöchigen unternehmensweit getakteten Sprints arbeiten.

Um die Releases zu planen – es werden vier Releases pro Jahr erstellt –, schätzen alle Verantwortlichen, also Teamleiter und Product Owner, falls notwendig unterstützt von Experten aus den Teams, in Schätzklausuren die noch sehr groben Anforderungen. Dazu nutzen sie die unternehmensweit standardisierte Einheit »Feature Point«. Ein Feature Point entspricht in etwa dem, was ein Entwickler in einem Sprint leisten kann.

Die Zahl der Entwickler ist bekannt, ebenso die Verantwortlichkeit jedes Teams. Damit kann auf Basis der Feature-Point-Bewertung und einer Priori-

sierung der Anforderungen entschieden werden, wie viele und welche Anforderungen in welchen Releases umgesetzt werden sollen. Die Anforderungen, die ein Team bis zum nächsten Release umsetzen soll, werden im folgenden Schritt in User Stories heruntergebrochen und dann vom Team in Story Points bewertet. Zeigt sich dabei, dass die Realisierung einer Anforderung durch ein Team deutlich aufwendiger wird, als initial in Feature Points geschätzt wurde, dann greifen entsprechende Prozesse, um z. B. die Release-Planung anzupassen. In regelmäßigen Retrospektiven werden die zur Schätzung und Planung genutzten Verfahren und Prozesse bewertet und angepasst.

Fazit: Aufwandsschätzung ist – und bleibt auch mit den vorgestellten Verfahren – schwierig. Aber die Verantwortlichen haben keine Alternative! Schätzen, dokumentieren, besser schätzen, das ist der einzige Weg zu brauchbaren Prognosen. Nur wenn wir die Erfahrungen aufzeichnen und für künftige Projekte zur Verfügung stellen, können wir aus den unvermeidbaren Fehlern lernen und die Qualität unserer Schätzungen über die Zeit verbessern. Mängel der verfügbaren Verfahren (die es zweifellos gibt) sind kein Alibi für den Verzicht auf die systematische Kostenschätzung.

8.7 Risikomanagement

In jedem Projekt treten Probleme auf, die das Projekt behindern, verzögern oder im schlimmsten Fall scheitern lassen. Mit dem Risikomanagement verfolgen wir das Ziel, möglichst frühzeitig denkbare Probleme zu identifizieren, um geeignete Gegenmaßnahmen einzuleiten, damit diese Probleme entweder gar nicht erst entstehen oder das Projekt nicht ernsthaft bedrohen. Risikomanagement ist eine kontinuierliche Tätigkeit, die in der Startphase besonders wichtig ist, aber danach, bis zum Projektende, systematisch fortgesetzt werden muss.

Präzises Risikomanagement ist notwendig, um Projekte erfolgreich abzuschließen. Umso erstaunlicher ist es, dass es in der Praxis oft vernachlässigt wird oder ganz fehlt (oder, was die gleiche Wirkung hat, zu einer Formalübung wird). Denkbare Gründe dafür sind, dass der Begriff Risiko vorbelastet ist, dass man nicht gerne über Risiken spricht, weil man alles »im Griff hat« oder glaubt, haben zu müssen. Außerdem sind die Verfahren für ein systematisches Risikomanagement oft nicht bekannt.

8.7.1 Definition und Beschreibung der Risiken

Was ist ein Risiko? In Anlehnung an Hindel et al. (2009) definieren wir:

Risiko — Ein Problem, das noch nicht eingetreten ist, aber wichtige Projektziele oder Projektergebnisse gefährdet, falls es eintritt. Ob es eintreten wird, kann nicht sicher vorausgesagt werden.

Risikomanagement umfasst alle Aktivitäten, die darauf abzielen, dass aus einem Risiko kein schweres Problem für das Projekt wird. Im Einzelnen zählen dazu:

- Identifikation von Risiken
- Analyse und Bewertung der Risiken
- Planung von Gegenmaßnahmen
- Verfolgen der Risiken und der gewählten Gegenmaßnahmen

Wichtig ist, dass Risiken explizit dargestellt und kommuniziert und dass geeignete Gegenmaßnahmen frühzeitig eingeleitet werden.

Um Projektrisiken zu identifizieren, hat es sich bewährt, Checklisten für Risikoquellen zu benutzen. Sie stellen sicher, dass die erfahrungsgemäß kritischen Themenfelder betrachtet und die identifizierten Risiken dokumentiert werden. Ziel ist es, eine möglichst vollständige Sicht auf die Projektrisiken zu erhalten. Beispiele für solche Themenfelder sind die verwendete Technik, die Beziehungen zum Kunden, die Verfügbarkeit von Mitarbeitern und Ressourcen und die Stabilität der Anforderungen. Carr et al. (1993) geben eine Taxonomie für Risikoquellen und Themen vor, die als Grundlage für Checklisten verwendet werden kann. Um eine möglichst vollständige Liste der Projektrisiken zu erhalten, müssen alle Experten beteiligt werden. In der Praxis haben sich dazu moderierte Workshops bewährt. Es liegt aber in der Natur der Risiken, dass wir nicht sicher sein können, alle Risiken entdeckt zu haben. Umso wichtiger ist es, dass gesammelte Erfahrungen verfügbar sind, die zum Beispiel in Form von Abschlussberichten anderer Projekte vorliegen.

8.7.2 Risikobewertung

Nachdem die Risiken identifiziert und beschrieben sind, müssen sie analysiert und bewertet werden. Dazu werden für jedes betrachtete Risiko die *Eintrittswahrscheinlichkeit* (p) und die im Schadensfall entstehenden Kosten (K) für das Projekt abgeschätzt. Das Produkt dieser beiden Werte ergibt den *Risikowert*, der für die weitere Analyse benutzt wird.

$$\text{Risikowert} = p \cdot K$$

Natürlich sind die Schätzwerte für die Eintrittswahrscheinlichkeit und den verursachten Schaden immer mit erheblicher Unsicherheit behaftet. Trotzdem sollte man versuchen, p und K für jedes Risiko abzuschätzen; es wird sich später zeigen, dass Fehler, die dabei unvermeidlich sind, keine schlimmen Folgen haben. Mit p und K kann man jedes Risiko als Punkt in einer doppelt logarithmischen Darstellung eintragen (Abb. 8–9). Geraden von links oben nach rechts unten kennzeichnen einen bestimmten Risikowert, der rechts abgelesen werden kann. In der Abbildung ist unterstellt, dass Risikowerte bis 10 000 € akzeptabel sind; das ist natürlich von der jeweiligen Umgebung abhängig, in einer kleinen Firma dürfte

dieser Wert niedriger sein. Für Punkte, die unter der Diagonalen liegen, sind keine Aktionen erforderlich.

Punkte über der Diagonalen müssen nach links oder nach unten bewegt werden. Eine Senkung der im Problemfall entstehenden Kosten verschiebt den Punkt nach unten (Pfeil a), eine Senkung der Wahrscheinlichkeit verschiebt ihn nach links (Pfeil b). In der Praxis haben alle Maßnahmen zur Folge, dass zusätzliche, sichere Kosten entstehen, die aber wesentlich geringer sind als der mögliche Schaden.

Zwei Bereiche (oben und rechts) sind von besonderer Bedeutung: Oben liegen extreme Risiken. Wenn ein solches Risiko eintritt, ist der Schaden nicht mehr zu beherrschen (z. B. ein Bankrott der Firma). In diesem Fall ist zu erwägen, ob eine Versicherung abgeschlossen werden kann. Versicherungen senken den Risikowert nicht, sondern machen aus einem wenig wahrscheinlichen sehr hohen Schaden eine sichere kleine Zahlung (Pfeil c).

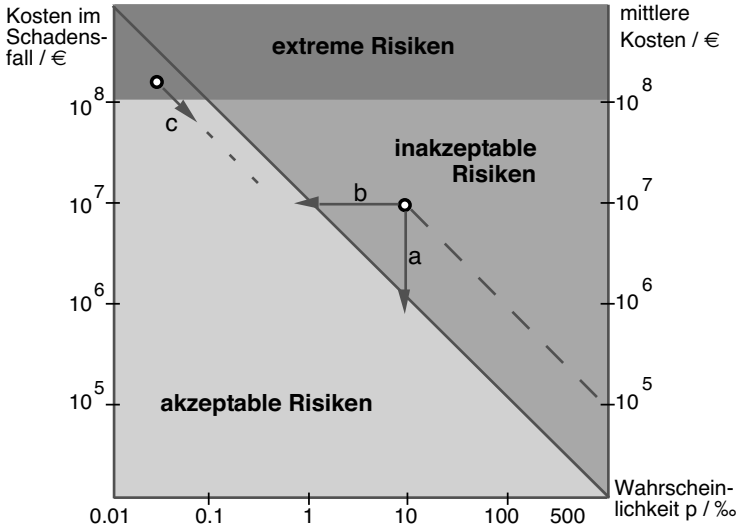

Abb. 8–9 *Risikodiagramm*

Risiken mit einer Eintrittswahrscheinlichkeit über 50 % sollten als sichere Ereignisse behandelt werden. Dass der Kunde seine Anforderungen ändert, dass auch nach dem Test noch Fehler im Programm sind, dass es Änderungen im Personal geben wird, das ist so sicher wie der Schneefall im kommenden Winter. Wir wissen noch nicht, wann, wo und wie viel Schnee fällt, aber wir können sicher damit rechnen, *dass* er fällt.

Alle geschätzten Werte sind unvermeidlich vage; die doppelt logarithmische Darstellung mildert aber den Effekt der Fehler ganz erheblich. Und wir erleben hier ähnlich wie bei der Aufwandsschätzung, dass die Zahlen genauer werden,

wenn wir das Verfahren einige Male angewandt haben: Wer ein Instrument spielen will, muss es schlecht gespielt haben, um es eines Tages gut spielen zu können.

Wer die Abschätzung der Werte scheut, kann eine vereinfachte Variante des Verfahrens anwenden. Tabelle 8–7 zeigt eine dreiwertige Skala für die beiden zu schätzenden Werte. Die Produkte der beiden Werte (1, 2, 3, 4, 6 oder 9) kennzeichnen das Risiko.

Wert	Eintrittswahrscheinlichkeit	Schadensausmaß
1	Es ist wenig wahrscheinlich, dass das Risiko eintritt.	Der Schaden wird für das Projekt kaum merklich sein: – geringe Einschränkung der Leistungen – geringe Zeitverzögerung – geringe Überschreitung der Kosten
2	Es wäre nicht überraschend, wenn das Risiko eintritt.	Der Schaden ist beträchtlich: – merkliche Einschränkung der Leistung – merkliche Zeitverzögerung – merkliche Überschreitung der Kosten
3	Es ist damit zu rechnen, dass das Risiko eintritt.	Der Schaden für das Projekt ist groß: – zentrale Funktionen sind betroffen – lange Zeitverzögerung – starke Überschreitung der Kosten

Tab. 8–7 *Eine dreiwertige Skala für die Risikobewertung*

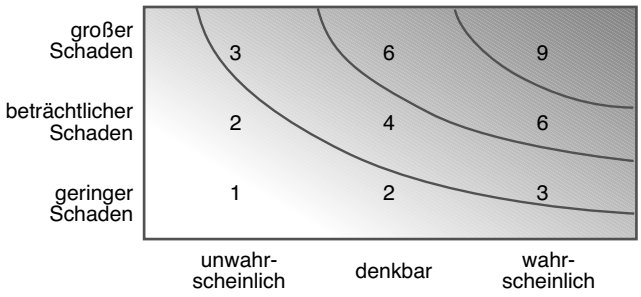

Abb. 8–10 *Klassifikation von Risiken*

8.7.3 Prävention und Notfallmaßnahmen

Präventivmaßnahmen sollen die Eintrittswahrscheinlichkeit eines Risikos senken (im günstigsten Fall auf null) oder die Bedingungen so verändern, dass der eintretende Schaden erträglich bleibt. Beispielsweise könnte das Risiko darin bestehen, dass ein für das Projekt sehr wichtiger Mitarbeiter die Firma verlässt. Indem man ihm eine Erfolgsprämie oder eine andere Belohnung für den erfolgreichen Abschluss in Aussicht stellt, kann man die Wahrscheinlichkeit, dass er geht, senken. Indem man ihm einen Kollegen zur Seite stellt, der in der Zusammenarbeit

wichtige Kenntnisse aufnimmt, verringert man den Schaden, der eintreten kann. Natürlich schließen sich die beiden Strategien nicht gegenseitig aus.

Notfallmaßnahmen werden reaktiv ergriffen, sie dienen dazu, den Schaden zu begrenzen, wenn ein Schadensfall eingetreten ist. Auch solche Maßnahmen können geplant werden. Beispielsweise könnte man sich im Fall des Mitarbeiters, dessen Firmentreue unsicher ist, frühzeitig vergewissern, wo und wie ggf. ein Nachfolger gefunden werden kann. Auf diese Weise hat man eine Problemlösung in der Schublade, wenn sie gebraucht wird.

Gegenmaßnahmen müssen, seien sie nun präventiv oder reaktiv, in den Projektplan aufgenommen werden, denn sie müssen – mehr als alle anderen Maßnahmen – überwacht werden, damit sie sicher zum Erfolg führen. Wenn nötig muss die Maßnahme abgeändert oder durch weitere Maßnahmen verstärkt werden. Da die Gegenmaßnahmen Kosten verursachen, müssen diese im Projektbudget eingeplant werden.

8.7.4 Risikoüberwachung

Da sich die Risiken während der Projektdurchführung verändern – vorhandene Risiken schwächen sich ab oder verstärken sich, neue Risiken kommen hinzu –, muss die Risikosituation eines Projekts während der gesamten Laufzeit immer wieder bewertet werden. Dies geschieht häufig in Form regelmäßiger Projektsitzungen. Es hat sich bewährt, die Risikosituation kondensiert darzustellen, sodass sie von den Managern schnell eingeschätzt werden kann. Häufig werden deshalb die zehn wichtigsten Risiken zusammen mit den Gegenmaßnahmen aufgelistet, und die Gesamtbewertung wird in Form einer Ampelskala visualisiert (mit Grün für alles, was glatt läuft, Gelb für drohende Probleme und Rot für dringenden Handlungsbedarf).

Ein solches systematisches Risikomanagement hat folgende Vorteile:

- Risiken werden frühzeitig identifiziert, dokumentiert und für alle Beteiligten sichtbar gemacht.
- Risiken werden nicht verschwiegen, sondern offen kommuniziert.
- Gegenmaßnahmen können frühzeitig geplant und durchgeführt werden. Es wird nicht erst dann gehandelt, wenn das Kind bereits in den Brunnen gefallen ist, d. h., wenn die Risiken zu Problemen geworden sind.
- Die Kosten und der Nutzen von Gegenmaßnahmen können bewertet werden.
- Das Ergebnis der Risikobewertung fließt in die Termin- und Kostenplanung ein.
- Auf die sich verändernde Risikosituation kann angemessen reagiert werden, weil die Risiken regelmäßig neu bewertet werden.
- Die Fähigkeiten zur Risikobewertung werden laufend verbessert, weil die Risikodaten gesammelt, analysiert und in Folgeprojekten genutzt werden.

Auch hier gilt, was in vielen Bereichen des Software Engineerings gilt: Es kommt vor allem darauf an, dass man das Risikomanagement ernsthaft, systematisch und projektbegleitend betreibt und laufend verbessert. Welche Techniken und Werkzeuge verwendet werden, ist sekundär.

8.8 Projektkontrolle und -steuerung

Die Projektkontrolle und -steuerung hat die Aufgabe, den Fortschritt eines Projekts zu beobachten und steuernd einzugreifen, wenn Abweichungen festgestellt werden, die durch Störungen verschiedenster Art verursacht sein können.

Für eine wirksame Projektkontrolle eignen sich solche Maßnahmen am besten, die Abweichungen vom Plan oder andere Probleme möglichst früh anzeigen. Nur dann können Korrekturen so früh eingeleitet werden (auch Korrekturen des Plans, wenn er ungeeignet war), dass noch Mittel für Maßnahmen bereitstehen und Zeit bleibt, damit die Maßnahmen wirken können.

8.8.1 Der Regelkreis der Projektdurchführung

Die Projektkontrolle und -steuerung kann als eine Art Regelkreis modelliert werden. Damit er nicht nur intuitiv, sondern explizit umgesetzt wird, ist es erforderlich, dass

a) Vorgaben und Erwartungen festgehalten werden: das Soll, der Plan;
b) erfasst und bewertet wird, was im Projektverlauf erreicht wird: das Ist;
c) Erwartungen und Bewertungen verglichen werden: Soll-Ist-Vergleich;
d) aus allen Abweichungen Konsequenzen gezogen, also korrigierende Maßnahmen eingeleitet werden.

Wird eine dieser Tätigkeiten nicht ausgeführt, so ist der Regelkreis nicht geschlossen, und das Projekt wird nicht geführt, sondern entwickelt sich nach seinen eigenen Gesetzen. Abbildung 8–11 zeigt, wie diese Tätigkeiten zusammenhängen.

Der Ist-Zustand wird anhand der bearbeiteten oder fertiggestellten Arbeitspakete festgestellt. Darauf gehen wir im nächsten Abschnitt genauer ein. Zeigt der anschließende Soll-Ist-Vergleich (Bewertung des Projektfortschritts) Abweichungen von der Planung, dann ist zu entscheiden, ob mithilfe geeigneter korrektiver Steuerungsmaßnahmen das Ist wieder an das Soll herangeführt werden kann oder die Planung geändert werden muss, also der Plan der korrigierten Einschätzung der Gegebenheiten angepasst werden muss. Bei der Auswahl von Steuerungsmaßnahmen muss immer berücksichtigt werden, welche Auswirkungen die erkannten Abweichungen auf die Projektziele haben; Leistungen, Kosten und Termine sind dabei zu unterscheiden. Steuerungsmaßnahmen (technische und organisatorische Maßnahmen) sind in vielen Fällen, Planänderungen unbedingt mit dem Auftraggeber abzustimmen.

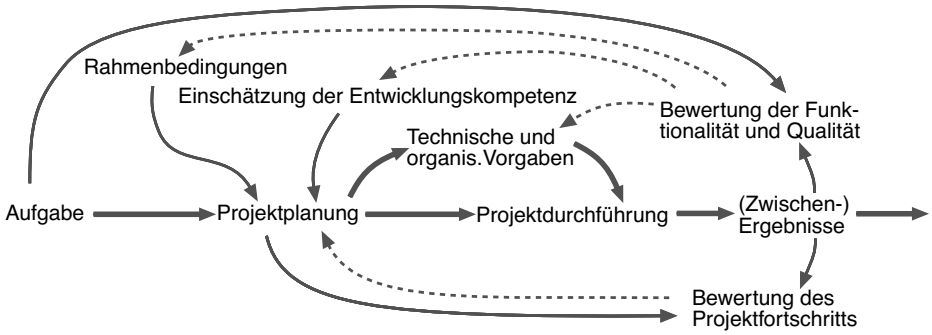

Abb. 8–11 *Projektkontrolle und -steuerung als Regelkreis; fette Pfeile kennzeichnen die konstruktive Entwicklung, punktierte Pfeile korrigierende Eingriffe. Die übrigen Pfeile dienen der Projektplanung und -verfolgung.*

Die Abbildung 8–11 ist vereinfacht, wenn sie auch auf den ersten Blick kompliziert erscheinen mag; die Bewertung des Projektfortschritts hat in vielen Fällen zusätzlich die Wirkungen, die bei der Bewertung der Funktionalität und der Qualität eingezeichnet sind. Zudem ist die Spezifikation, die aus der Aufgabe folgt, nicht explizit dargestellt.

8.8.2 Die Bewertung des Erreichten

Im Mittelpunkt der Projektkontrolle steht das Arbeitspaket. Jeder Mitarbeiter ordnet seine Arbeitsstunden den Arbeitspaketen zu, mit denen er sich befasst hat. Dieser Ist-Aufwand wird dem Soll, der Aufwandsschätzung für das Arbeitspaket, gegenübergestellt. Abweichungen entstehen,

- wenn die Schätzung falsch war,
- wenn die Arbeit ineffizient durchgeführt wurde,
- wenn eine andere oder größere Aufgabe gelöst wurde, als im Arbeitspaket vereinbart war,
- wenn die Arbeitszeit falsch verbucht wurde.

Zeiterfassung

Die Zeiterfassung wird in der Regel auch in Bezug auf die einzelnen Mitarbeiter ausgewertet. Wer für 40 h pro Woche bezahlt wird, sollte auch 40 h Arbeit nachweisen. Durch einige Stunden Toleranz berücksichtigt man die Tatsache, dass es Zeiten gibt, die nicht sinnvoll zugeordnet werden können. Trotzdem entstehen scheinbare Fehlzeiten, wenn bestimmte notwendige Arbeiten nicht in den Arbeitspaketen berücksichtigt waren, weil sie in der Planung vergessen wurden oder weil sie *gar nicht auftreten dürfen*. (Ein Beispiel sind in vielen Unternehmen die zusätzlichen Arbeiten, die entstehen, wenn die Projekte international durchge-

führt werden. Das Management, das mit dieser Entscheidung seinen Willen zur Kostensenkung demonstriert, kann nicht zugeben, dass dadurch Kosten entstehen.)

Die Mitarbeiter reagieren darauf, indem sie die Zeiten *irgendwo* buchen. Ein ähnlicher, weitverbreiteter Effekt ist zu beobachten, wo die Budgets der einzelnen Projekte oder Projektphasen hart beschränkt sind, die Mitarbeiter aber typischerweise in mehreren Projekten arbeiten. Dann gilt das Prinzip: Dieser Topf ist leer, also esse ich aus einem anderen. Damit verliert die Aufwandserfassung natürlich ihren Sinn und Nutzen. Nachfolgend unterstellen wir, dass wir korrekte Zahlen für den erbrachten Aufwand haben.

Fertigstellungsgrad

In einer idealen Welt könnte man (wenn es dort nicht ganz überflüssig wäre) den Fertigstellungsgrad nach der folgenden Formel ermitteln:

$$\text{Fertigstellungsgrad} = \frac{\text{Ist-Aufwand}}{\text{Geschätzter Gesamtaufwand}}$$

Sie entspricht dem optimistischen Prinzip: »Mein Geld ist fast alle. Also ist heute der 28.« Selbst wenn die Aufwandsschätzung relativ gut war, ist der entstehende Fehler gegen Ende des Projekts groß. Betrachten wir ein Beispiel: Nach neun Monaten eines Projekts sind etwa 90 % des geschätzten Aufwands erbracht. Wir folgern, dass der Fertigstellungsgrad 90 % ist, nach einem weiteren Monat werden wir vermutlich fertig sein. Wenn aber die Schätzung um (nur) 11 % zu niedrig lag, haben wir tatsächlich noch zwei Monate vor uns; wir werden also von der Verzögerung überrascht.

Um den Fertigstellungsgrad einer Software zu beurteilen, sollten wir nicht den geschätzten Gesamtaufwand, sondern den Restaufwand, den noch notwendigen Aufwand bis zur Fertigstellung, zugrunde legen. Mit anderen Worten: Wir verwenden nicht die ursprüngliche, sondern die aktuelle Aufwandsschätzung. Diese ergibt sich als Summe des bereits erbrachten Aufwands und des Restaufwands:

$$\text{Fertigstellungsgrad} = \frac{\text{Ist-Aufwand}}{\text{Ist-Aufwand} + \text{Restaufwand}}$$

Diese Größe hat den Vorteil, dass eine Übertreibung in irgendeine Richtung dem Mitarbeiter schadet: Ist der prognostizierte Restaufwand zu groß, wird die bereits geleistete Arbeit abgewertet, ist der Restaufwand zu klein, sind Probleme bei den nächsten Fortschrittskontrollen programmiert.

Auf ein mögliches Missverständnis sei hier vorsorglich hingewiesen: Der prognostizierte Restaufwand oder der prognostizierte Gesamtaufwand wird nicht automatisch zum neuen geplanten Aufwand und somit zu einer neuen Zielgröße.

Der Planwert gilt weiterhin, und eine abweichende Prognose ist eine Aufforderung, Maßnahmen zur Beseitigung dieser Abweichung zu ergreifen.

In Organisationen, in denen der erfasste Ist-Aufwand erst nach Wochen zur Verfügung steht, kann diese Größe nicht zur Vorhersage des Restaufwands herangezogen werden, denn sonst liegt die Bewertung des Fertigstellungsgrads erst zu einem Zeitpunkt vor, zu dem sie bereits überholt ist.

Ein anderer Ansatz umgeht dieses Problem, indem der Ist-Aufwand gar nicht verwendet wird. Als Referenz dient nicht der prognostizierte Gesamtaufwand, sondern der geplante Aufwand. Dabei wird der *erarbeitete Wert* nach folgender Gleichung bestimmt:

$$\text{Erarbeiteter Wert} = \text{Geplanter Aufwand} - \text{Prognostizierter Restaufwand}$$

Der Fertigstellungsgrad lässt sich auf dieser Basis durch folgende Formel berechnen:

$$\text{Fertigstellungsgrad} = \frac{\text{Erarbeiteter Wert}}{\text{Geplanter Aufwand}}$$

Achtung, der erarbeitete Wert und damit der Restaufwand kann negativ werden, wenn das Projekt sehr sorglos geplant worden war, sodass während der Durchführung der Restaufwand höher ist als der zu Beginn geschätzte Gesamtaufwand.

Kennt man auch den Ist-Aufwand, so erlaubt der erarbeitete Wert Aussagen wie: »Wir haben 90 % des budgetierten Aufwands geleistet, aber erst 80 % der Aufgabe erledigt.«

Den Fertigstellungsgrad des gesamten Projekts kann man sehr einfach auf Basis der bereits abgeschlossenen Arbeitspakete nach folgender Beziehung ermitteln:

$$\text{Fertigstellungsgrad} = \frac{\text{Anzahl abgeschlossener Arbeitspakete}}{\text{Anzahl der Arbeitspakete}}$$

In diese Rechnung gehen die gerade bearbeiteten Arbeitspakete nicht ein. Das liefert Ergebnisse, die etwas zu niedrig sind. Bei einer geringen Gesamtzahl ist dieser Fehler möglicherweise inakzeptabel hoch; dann kann man Pakete, die gerade in Arbeit sind, mit 0,5 oder einem noch genauer abgeschätzten Gewicht berücksichtigen.

Die Earned-Value-Analyse

Das bisher beschriebene Verfahren ist für kleinere Projekte angemessen. Bei größeren und großen Projekten reicht es jedoch nicht aus. In solchen Projekten wird oft die sogenannte Earned-Value-Analyse (EV-Analyse) genutzt, die auch als Earned Value Management bezeichnet wird. Dieses Verfahren wurde in den Sechzigerjahren vom US-Verteidigungsministerium entwickelt, um den Fortschritt

eines Projekts zu bewerten und um Prognosen über den voraussichtlichen Endtermin und die zu erwartenden Gesamtkosten zu erhalten. Als Ergebnis der Analyse erhält die Projektleitung eine kleine Menge von Kennzahlen, die sie als Steuerungsgrößen nutzen kann. Eine einführende Beschreibung der EV-Analyse stellt das »Project Management Institute« auf seiner Webseite zur Verfügung (PMI, o. J.). Das Verfahren wird ausführlich in Kerzner (2017) vorgestellt, seit 2018 liegt die EV-Analyse in der Norm ISO 21508 vor.

Als Datenlieferanten der EV-Analyse dienen die Kostenplanung durch Angabe der geplanten Kosten über die Projektlaufzeit und der geschätzten Gesamtkosten (Budget At Completion, BAC) sowie die Buchhaltung, die die bis zum Berichtszeitpunkt aufgelaufenen Projektkosten kennt. Natürlich stützt sich die EV-Analyse auch auf die Bewertung der einzelnen Arbeitspakete. Für diese müssen die folgenden Werte ermittelt werden, die dann auf Projektebene zusammengefasst werden:

1. *Geplante Kosten (Planned Value, PV)*
   Das sind die bis zum Berichtszeitpunkt laut Planung vorgesehenen Kosten für ein Arbeitspaket.
2. *Tatsächliche Kosten (Actual Cost, AC)*
   Das sind die bis zum Berichtszeitpunkt tatsächlich entstandenen Kosten für ein Arbeitspaket.
3. *Erarbeiteter Wert (Earned Value, EV)*
   Das sind die geplanten Kosten für den Wert der bis zum Berichtszeitpunkt geleisteten Arbeit. EV gibt also die geplanten Kosten für ein Arbeitspaket an, gemessen am tatsächlichen Fertigstellungsgrad.

Betrachten wir dazu das folgende sehr einfache Beispiel: Ein Arbeitspaket soll mit der Kalenderwoche 37 beginnen und fünf Arbeitstage erfordern; jeder Arbeitstag kostet 1000 € (1 k€). Die geplanten Kosten sind somit 5 k€. Zum Berichtszeitpunkt am Ende der KW37 stellt sich die Situation so dar: Das Arbeitspaket konnte nicht wie geplant am Montag, sondern erst am Dienstagmittag in Angriff genommen werden; damit sind Kosten in Höhe von 3,5 k€ entstanden. Durch unerwartete Probleme wurde in diesen dreieinhalb Tagen nur die Arbeit geschafft, für die drei Tage (entsprechend 3 k€) vorgesehen waren. Wir haben damit als Kennwerte PV = 5 k€, AC = 3,5 k€, EV = 3 k€.

Mithilfe dieser Basisgrößen lassen sich nun die Kosten- und die Zeitabweichung relativ zur Planung bestimmen.

- *Kostenabweichung (Cost Variance, CV = EV - AC)*
  Sie ergibt sich aus der Differenz zwischen erarbeitetem Wert und tatsächlichen Kosten. Ist sie negativ, dann hat die Arbeit nicht so viele Werte geschaffen, wie sie gekostet hat.

- *Planabweichung (Schedule Variance, SV = EV - PV)*
  Sie berechnet sich aus der Differenz von erarbeitetem Wert und den zum Berichtszeitpunkt geplanten Kosten. Die Planabweichung wird also durch die finanzielle Differenz zur Planung ausgedrückt. Ist der SV-Wert negativ, dann liegt das Projekt gegenüber der Planung zurück, und zwar um Arbeit im Umfang des SV-Wertes.

In unserem Beispiel erhalten wir die folgende Werte:

$$\begin{aligned} CV &= 3\,k€ - 3{,}5\,k€ &&= -0{,}5\,k€ \\ SV &= 3\,k€ - 5\,k€ &&= -2\,k€ \end{aligned}$$

Sie bedeuten, dass das, was bisher erarbeitet wurde, 500 € teurer war als geplant und dass wir einen Arbeitsrückstand äquivalent zum Wert von 2000 € haben.

Wie bereits erwähnt, wird die EV-Analyse auf Projektebene durchgeführt. Die PV-Werte aus der Kostenplanung kann man in einem Diagramm über der Zeit auftragen. Fügt man regelmäßig die ermittelten EV-Werte und die aktuellen AC-Werte aus der Buchhaltung hinzu, dann entspricht der vertikale Abstand der EV-Kurve von den PV- und AC-Kurven den Größen SV und CV (Abb. 8–12).

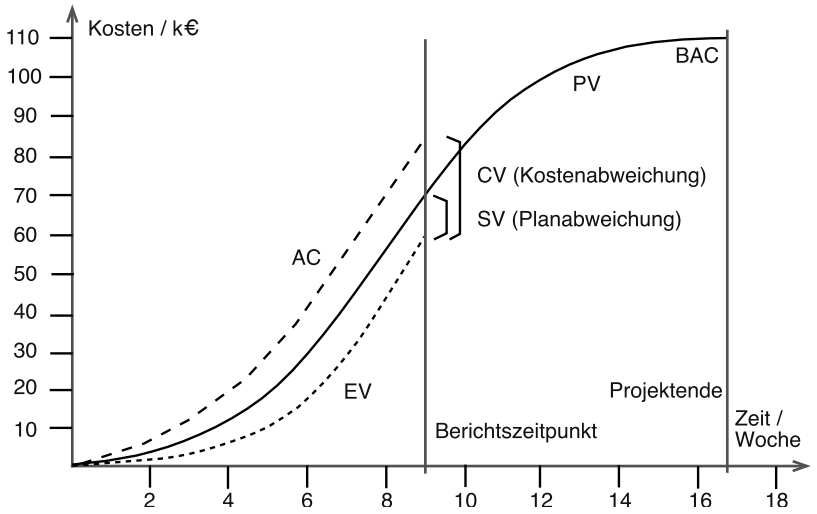

Abb. 8–12 *Darstellung von Kosten- und Planabweichung*

Die EV-Analyse kann auch genutzt werden, um die Gesamtkosten und den Fertigstellungstermin vorherzusagen. Dazu dienen zwei Leistungskennzahlen:

- *Cost Performance Index (CPI = EV / AC)*
  Dieser setzt den Wert der tatsächlich geleisteten Arbeit ins Verhältnis zu den tatsächlichen Kosten. Ist der Quotient größer als eins, dann war die geleistete Arbeit günstiger als geplant; ist er kleiner, dann war sie teurer.

- *Schedule Performance Index (SPI = EV / PV)*
  Hier wird die tatsächlich geleistete Arbeit zu den geplanten Kosten ins Verhältnis gesetzt. Ist der Wert größer als eins, dann konnte mehr erreicht werden als geplant. Ist er kleiner als eins, dann ist der Arbeitsfortschritt geringer als angenommen.

Dementsprechend sollten die Werte für CPI und SPI in einem gut geführten Projekt nur in einem schmalen (und definierten) Intervall um 1 liegen.

Unser Beispielprojekt liefert für den Berichtszeitpunkt nach Woche 9 folgende Werte: PV = 70 k€, AC = 85 k€, EV = 60 k€ (Abb. 8–12). Daraus berechnen wir als Kennzahlen CPI = 0,71 und SPI = 0,86. Sie bedeuten, dass wir bis jetzt 14 % weniger Leistung erbracht haben als geplant. Da wir gleichzeitig mehr Aufwand hatten, als vorgesehen war, liegen wir sogar 29 % unter der Leistung, die nach den Kosten zu erwarten wäre. Das Projekt erfordert also massive Eingriffe oder gar den Abbruch.

Wenn aufgrund von Erfahrungen angenommen werden kann, dass die ermittelten Leistungskennzahlen typisch sind und sich im weiteren Projektverlauf nicht wesentlich ändern werden, dann können Prognosen gestellt werden:

- *Wie viel wird das Projekt am Ende kosten?*
  Die auf Basis der aktuellen Bewertung prognostizierten Projektkosten (Independent Estimate At Completion, IEAC) berechnen sich als Verhältnis vom anfangs geschätzten Gesamtaufwand (BAC) zum Cost Performance Index.

  IEAC = BAC / CPI

- *Wie viel Zeit werden wir für das Projekt benötigen?*
  Die voraussichtliche Projektdauer (Independent Schedule At Completion, ISAC) kann auf Basis der geplanten Dauer und des Schedule Performance Index bestimmt werden.

  ISAC = Geplante Dauer / SPI

Statt nach einer Prognose können wir auch fragen, ob es durch höhere Leistung möglich ist, das Projekt noch zu den geplanten Kosten fertigzustellen. Über das ganze Projekt hinweg hätte ein CPI = 1 ausgereicht. Bislang wurden aber nur 71 % (CPI) der erwarteten Leistung erreicht.

Die noch ausstehende Arbeit ist BAC - EV, das noch verfügbare Budget BAC - AC. Bis zum Projektabschluss müsste also der To Complete Performance Index (TCPI) gelten:

TCPI = (BAC - EV) / (BAC - AC)

Mit den bereits ermittelten Kennzahlen erhalten wir für unser Beispielprojekt die folgenden Prognosen:

IEAC = 110 k€ / 0,71 = 155 k€ 41 % Mehrkosten!
ISAC = 17 Wochen / 0,86 = 19,8 Wochen 16 % Zeitüberschreitung!

Die Frage nach dem erforderlichen TCPI ergibt:

TCPI = (110 k€ - 60 k€) / (110 k€ - 85 k€) = 50/25 = 2

Ein TCPI von 2 bedeutet, dass das Projektteam ab dem Berichtszeitpunkt 200 % der Leistung bringen müsste, die ursprünglich gefordert war, um das Projekt zu den geplanten Kosten abzuschließen. Gegenüber der bisherigen Leistung (71 %) ist fast eine Verdreifachung notwendig. Das ist offensichtlich irreal. Selbst ein wesentlich kleineres »Über-Soll« wie 120 % ist nur schwer und nicht auf Dauer zu schaffen.

8.8.3 Termindrift-Diagramme (Meilenstein-Trend-Analyse)

Durch die Projektverfolgung, wie sie im vorigen Abschnitt beschrieben ist, werden Daten gesammelt, die den Stand des Projekts charakterisieren und später die Chance bieten, Schwächen des Projekts zu erkennen, sodass sie bei weiteren Projekten vermindert oder vermieden werden können. Dazu müssen die Informationen möglichst anschaulich visualisiert werden; eine simple Tabelle reicht nicht aus, auch wenn sie rein formal alle Informationen enthält.

Eine besonders eingängige Darstellung des Projektverlaufs und der aufgetretenen Abweichungen von der Planung erhält man durch das Termindrift-Diagramm. In der Literatur wird es meist als *Meilenstein-Trend-Analyse* bezeichnet, das suggeriert aber eine Methode, wo es sich nur um eine einfache Darstellungstechnik handelt. Leider finden wir nirgends eine Angabe, die den Urheber würdigt.

Die Grundidee besteht darin, in einem Koordinatensystem aus zwei Zeitachsen einzutragen, wie sich die Termine für das Erreichen bestimmter Meilensteine im Laufe des Projekts verändern (oder nicht verändern). Prinzipiell ist es natürlich gleichgültig, ob man die vertikale Achse nach oben oder nach unten zeigen lässt; wir ziehen (wegen der Schreib- und Leserichtung von links nach rechts und von oben nach unten) die hier gezeigte Form vor.

Die Abbildung 8–13 zeigt ein einfaches (und untypisches) Beispiel. Zum Anfang des Projekts (links) gibt die Planung die vertikale Achse mit den geplanten Meilensteinen vor. Mit fortschreitender Zeit wächst das Diagramm von links nach rechts. Immer, wenn der Stand des Projekts neu eingeschätzt und die Planung eventuell verändert wird, können unter dem entsprechenden Punkt der horizontalen Achse die geplanten Meilensteine markiert werden. Im Beispiel wird M1 planmäßig erreicht, und zum selben Zeitpunkt stellt sich heraus, dass M2 früher als ursprünglich erwartet erreicht werden kann. Dadurch entsteht die Stufe nach oben. Wenn die Meilenstein-Linie die Diagonale trifft, ist der Meilenstein erreicht; das Gebiet rechts über der Diagonalen ist Vergangenheit, es hat keinen Sinn, die Linien dort weiterzuführen.

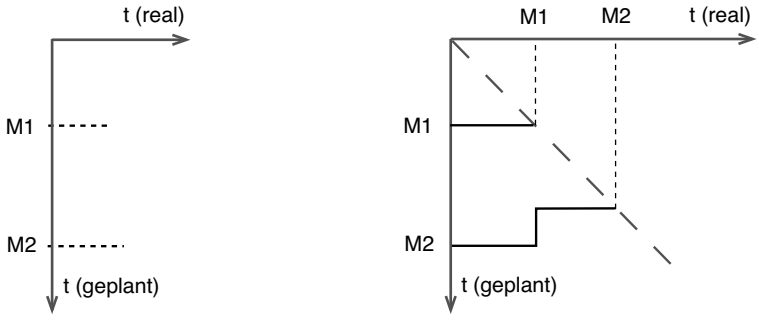

Abb. 8–13 *Termindrift-Diagramm zu Beginn und nach Ende des Projekts*

Offensichtlich werden in einem perfekt geplanten Projekt die Meilensteine genau zum geplanten Zeitpunkt erreicht (M1 in Abb. 8–13). War die Planung allzu ängstlich oder wurde der Aufwand überschätzt, so werden Meilensteine früher erreicht, als geplant war (M2); im sehr viel häufigeren Fall einer allzu sorglosen Planung geschieht das Gegenteil (alle Meilensteine in Abb. 8–14).

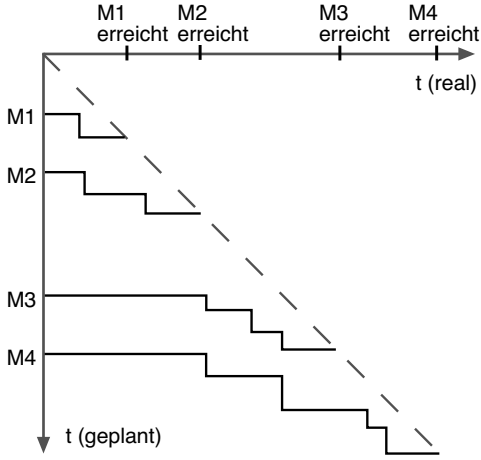

Abb. 8–14 *Termindrift-Diagramm, Beispiel mit Verzug beim Erreichen aller Meilensteine*

Man sieht auf einen Blick,

- wie die ursprüngliche Planung war (die Zieltermine der Meilensteine),
- wann die Planung korrigiert wurde,
- wann die Meilensteine tatsächlich erreicht wurden.

Am Ende eines schlecht geplanten und entsprechend schlecht gelaufenen Projekts ähnelt das Diagramm den Schleuderspuren eines Autos, dessen Fahrer die Schwierigkeiten und Risiken der Fahrt unterschätzt hatte. So erkennt man in Abbildung 8–14, dass sich der Projektleiter auch nach frühen Anzeichen für Pla-

nungsfehler nicht zu einer Totalrevision entschließen konnte. Stattdessen hat er wider besseres Wissen angekündigt, die späteren Meilensteine wie geplant zu erreichen, und an der falschen Prognose festgehalten, bis die Termine jeweils kurz bevorstanden.

8.9 Der Projektabschluss

Oft enden Projekte ohne eigentlichen Abschluss. Viele Mitarbeiter verlassen das Projekt schon vor der Auslieferung, andere haben noch darüber hinaus mit Nacharbeiten zu tun. Auf diese Weise fehlen sowohl die objektive Dokumentation der mit diesem Projekt erreichten Erfolge und Misserfolge als auch das subjektive Bewusstsein der Beteiligten, auf etwas Abgeschlossenes zurückblicken zu können.

8.9.1 Abschlussarbeiten

Zu den administrativen Arbeiten gehört zunächst die Archivierung der Projektresultate und aller Projektunterlagen.

Organisatorisch ist es notwendig, die Projektorganisation aufzulösen und alle Projektmitarbeiter in neuen Aufgaben unterzubringen. Dies macht vor allem bei einer reinen Projektorganisation Arbeit. Oft müssen noch einige Restarbeiten erledigt werden, auch wenn das Projekt schon abgeschlossen ist. Diese müssen identifiziert und Personen zugewiesen werden.

Der Abschluss eines Projekts bietet wie kein anderer Moment die Möglichkeit, durch Aufzeichnung von Projektkennzahlen aus verschiedener Sicht ein plastisches Bild des Projekts festzuhalten, das für die Planung weiterer Projekte benutzt werden kann. Es sind mindestens die Standpunkte des Projektleiters, eines erfahrenen Entwicklers und der Kosten-/Terminplanstelle zu berücksichtigen.

8.9.2 Rituale

Traditionell vollziehen die Menschen Rituale, wenn eine größere Veränderung stattgefunden hat. Meist sind diese Rituale religiös geprägt. Typische Beispiele sind die Begrüßung eines Neugeborenen, die Aufnahme in den Kreis der Erwachsenen, die Heirat oder die Totenfeier. Auch im Beruf gibt es eine Reihe solcher Rituale, beispielsweise die Aufnahme unter den Handwerksgesellen oder die Priesterweihe. Offensichtlich ist der wesentliche Zweck des Rituals, die Veränderung für alle Menschen deutlich zu machen.

Auch Projekte durchlaufen verschiedene Zustände. Auf dem Bau kennen wir die Grundsteinlegung, das Richtfest und die Schlüsselübergabe. Mehr noch als bei einem Haus, das gut sichtbar aus dem Fundament wächst, sind bei der Entwicklung eines Software-Systems Rituale extrem nützlich. Mindestens die drei

Meilensteine Projektstart, Fertigstellung der Konzeption und Abschluss sollten Anlass sein, die Veränderung durch ein Fest im Bewusstsein der Beteiligten (Projektleiter und Mitarbeiter, Kunden, Projekteigentümer, Management) zu verankern.

8.9.3 Die Dokumentation der Erfahrungen

Die Rückschau wird am besten in einem Dokument mit einer dem Projektplan ähnlichen Gliederung zusammengefasst (Abb. 8–15).

Die ersten vier Kapitel dienen einer kompakten Darstellung der gesammelten Informationen. Die beiden letzten Kapitel sollen Interpretation und Umsetzung in Maßnahmen enthalten.

Um vergleichbare Erfahrungswerte zu bekommen, insbesondere Werte, die für die Planung des nächsten Projekts nützlich sind, ist es wichtig, die wesentlichen Projektstrukturen, z. B. die Phasenfestlegung oder die Gliederung in Arbeitspakete, über einen längeren Zeitraum konstant zu halten.

Abb. 8–15 *Inhaltsverzeichnis der Projektgeschichte*

8.10 Projektmanagement als Führungsaufgabe

Projektmanagement ist und bleibt eine sehr komplexe Aufgabe, die sich nicht durch einige einfache Regeln beschreiben oder verbessern lässt. Aber in der Praxis sieht man immer wieder, dass sehr simple Fehler gemacht werden. Zumindest diese kann man durch Beachtung einiger Regeln vermeiden. Besonders kritisch ist die Besetzung der Projektleiter-Rolle.

8.10.1 Die Wahl des Projektleiters

Die Wahl des Projektleiters ist von großer Bedeutung, denn nur, wenn der Projektleiter das ihm übertragene Projekt angemessen leiten kann, hat es eine Erfolgschance. Ein geeigneter Projektleiter ist eine notwendige Voraussetzung für ein erfolgreiches Projekt.

Natürlich gibt es den idealen Projektleiter nicht. Bei der Auswahl sollte jedoch immer beachtet werden, dass jedes Projekt individuelle Randbedingungen und Probleme hat, denen die Qualifikationen und Fähigkeiten des Projektleiters entsprechen sollten. Je größer ein Projekt ist, desto mehr Management- und Organisationstalent ist gefragt. Technisches Verständnis ist hier weniger wichtig, dies kann von einem Mitglied des Projektteams ergänzt werden.

In der Praxis beginnt die typische Projektleiterlaufbahn damit, dass jemand aufgrund guter technischer Leistungen zum Projektleiter ernannt wird. Diese Person kann und will einen solchen Karriereschritt in der Regel nicht ablehnen, auch wenn die neue Rolle nicht unbedingt den eigenen Neigungen und persönlichen Zielen entspricht. Eine Absage würde in vielen Fällen das Ende der Karriere bedeuten, da Karriere oft ausschließlich über Projekt- und Mitarbeiterverantwortung definiert ist. Eine technische Karriere (zum Chef-Entwickler oder Senior Software Engineer) gibt es nur selten.

Als Konsequenz dieser Karrierepolitik leiten Mitarbeiter Projekte, obwohl sie in technischen Bereichen viel bessere Leistungen erzielen könnten. Andere arbeiten technisch, obwohl sie in organisatorischen Bereichen deutlich mehr Erfolg hätten.

Aufgrund der zentralen Rolle im Projekt und der vielseitigen Tätigkeiten, die er durchführen muss, werden von einem guten Projektleiter viele Qualifikationen und Fähigkeiten gefordert. Dazu zählen:

- *Fachliche Qualifikation*
  Der Projektleiter benötigt kaufmännisches Know-how, Wissen über die Anwendung, die erstellt werden soll, und natürlich auch Software-technisches Wissen.
- *Projektmanagement-Qualifikationen*
  Der Projektleiter muss gängige Techniken der Planung, der Schätzung, der Planverfolgung und des Zeitmanagements beherrschen. Aber auch Verhandlungsgeschick im Umgang mit dem Auftraggeber und dem eigenen Management ist gefragt.

- *Führungsfähigkeiten*
  Als Leiter des Entwicklungsteams benötigt der Projektleiter Offenheit, Kommunikations- und Entscheidungsfähigkeit, Bereitschaft zum Delegieren, Fähigkeit zur Motivation und zur Vermittlung zwischen den beteiligten Personen.
- *Persönliche Fähigkeiten*
  Dazu zählen beispielsweise Verantwortungswille, Belastbarkeit, Beharrlichkeit, Teamgeist, Geduld, Fähigkeit zum Umgang mit unterschiedlichen Typen von Menschen und Konfliktfähigkeit.

8.10.2 Woran scheitern Projekte?

Wer in einem Projekt eine leitende Rolle hat, steht in der Regel unter Druck. (Wir sprechen hier der Einfachheit halber vom Projektleiter, auch wenn seine Rolle anders benannt ist.) Die zugesagten Mitarbeiter stehen nicht zur Verfügung, der Kunde ändert seine Anforderungen, zugelieferte Komponenten kommen nicht zum vereinbarten Termin, neue Mitarbeiter sind weniger produktiv als erwartet, es gibt Konflikte im Team usw. usw.

Natürlich muss sich der Projektleiter um alle diese Probleme kümmern, er muss sie erkennen, beobachten und bekämpfen. Dafür gibt es keine einfache Lösung. Trotzdem sollte sich der Projektleiter von Zeit zu Zeit zurücklehnen und prüfen, ob die Prioritäten richtig gesetzt sind. Dabei können die Erfahrungen helfen, die in zahlreichen Publikationen dokumentiert sind. Ein Musterbeispiel solcher Ratgeber ist die Metastudie von Nasir und Sahibuddin (2011). Die Autoren haben 43 Publikationen ausgewertet, die sich mit den kritischen Erfolgsfaktoren in Software-Projekten befassen. Sie haben daraus 26 kritische Erfolgsfaktoren extrahiert und diese nach der Zahl der Nennungen in einer Rangliste geordnet.

Vor allem auf den ersten Plätzen gibt es keine Überraschungen; mehr als zwanzig Nennungen hat jeder der folgenden fünf Erfolgsfaktoren bekommen:

1. Clear requirements and specifications
2. Clear objectives and goals
3. Realistic schedule
4. Effective project management skills/methodologies (project manager)
5. Support from top management

Wer als Projektleiter, als Entwickler oder als Berater viele Projekte aus der Nähe beobachten konnte, wird ohne Zweifel zustimmen: Diese Erfolgsfaktoren sind definitiv kritisch für den Projekterfolg. Trotzdem werden sie in der Praxis oft vernachlässigt. Darum sollte man sie immer wieder in Erinnerung rufen und im Team diskutieren.

Die übrigen 21 Faktoren sind nicht unwichtig. Aber man kann ihre Reihenfolge diskutieren. Zudem ist im konkreten Projekt nicht klar zu entscheiden, ob dieser oder jener Punkt ausreichend Beachtung gefunden hat. Das betrifft die meisten Themen. Wann haben wir »realistic budget« (Rang 8) oder »adequate resources« (Rang 16)? Wann können wir sagen, dass wir ein »committed and motivated team« haben (Rang 22)? Harte Kriterien kann man daraus nicht ableiten.

Fazit: Metastudien zu den Erfolgsfaktoren eignen sich als Checklisten, um vor allem die Projektleiter auf die regelmäßig auftretenden Probleme hinzuweisen, bevor sie sich von selbst zeigen. Ein typisches Beispiel ist das Risikomanagement (Rang 18). Faustregel: Wenn man nicht sicher ist, genug für das Risikomanagement getan zu haben, hat man nicht genug getan.

Aber es gibt auch – nicht nur in gescheiterten Projekten – schwerwiegende Probleme; die Ergebnisse der Studie von Mandl-Striegnitz und Lichter (1999) belegen das. Wir sehen dafür vor allem die folgenden Gründe:

- *Mangelnde Ausbildung*
  In vielen Fällen übernehmen Personen die Leitung eines Projekts, ohne vorher die notwendigen Managementkonzepte, Methoden und Techniken zu erlernen. Lernen durch Fehlermachen ist die logische Konsequenz. Das funktioniert aber bei Weitem weniger gut als allgemein angenommen. Denn es setzt voraus, dass der Lernende die Möglichkeit hat, Fehler zu erkennen, zu reflektieren und zu korrigieren. In einem Projekt mit hartem Termindruck und reichlich Problemen hat er diese Möglichkeit kaum.
- *Unrealistische Erwartungen des oberen Managements*
  Oft werden vonseiten des oberen Managements ungünstige Randbedingungen für Projekte vorgegeben. Trotzdem sollen die Projekte erfolgreich durchgeführt werden. So ist es beispielsweise unmöglich, hochqualitative Software zu entwickeln, wenn die Termine viel zu eng sind und zu wenig und häufig noch unzureichend geschultes Personal zur Verfügung steht.
- *Implizite Kundenerwartungen*
  Der Kunde erwartet, dass das Projekt Anforderungen realisieren soll, die nie klar formuliert wurden.
- *Verhalten der Mitarbeiter*
  Sie erwarten Informationen vom Projektleiter, geben aber eigene Informationen, z. B. über den bereits geleisteten Aufwand, selbst nicht korrekt weiter.
- *Eigener Anspruch*
  Übernimmt der Projektleiter zusätzlich auch noch technische Aufgaben, so wird ab einer bestimmten Belastung einer der beiden Bereiche darunter leiden. Da die Ergebnisse der technischen Aktivitäten besser sichtbar und bewertbar sind als die Managementaufgaben, werden die technischen Aktivitäten vorgezogen.

Aus diesen Feststellungen folgern wir speziell für die Projektleiter:

- Projektleiter müssen ausgebildet sein! Wie jedes andere Teammitglied muss auch der Projektleiter die notwendigen Fähigkeiten mitbringen.
- Projektleiter müssen ernst genommen werden! Dies gilt insbesondere für die vom Projektleiter erstellten Planungen, Schätzungen und Risikobetrachtungen.
- Die Zusammenarbeit mit dem Management muss reibungslos funktionieren. Nur dann können die notwendigen Entscheidungen im Projekt abgestimmt und zielführend getroffen werden.
- Die Auftraggeber müssen erkennen, dass ohne ihre eigene intensive Beteiligung keine brauchbaren Systeme entstehen können.
- Die Mitarbeiter müssen lernen, systematisch und nach Vorgaben zu arbeiten und ihre Ergebnisse angemessen zu dokumentieren.
- Die Projektleiter müssen sich auf die Aufgaben des Projektmanagements konzentrieren.

8.10.3 Führungsprobleme

Bei Entwicklungsprojekten treten immer wieder Führungsprobleme auf. Aus der Sicht des Projektleiters (Manager-Perspektive) sind mögliche Ursachen dafür:

- Die Mitarbeiter sind nicht zielstrebig.
- Die Eigeninitiative der Mitarbeiter ist unzureichend, sie lassen sich kaum fördern.
- Die Mitarbeiter haben kein Verständnis oder gar Interesse für »höhere« Ziele, also die Ziele der Organisation oder Firma.
- Den Mitarbeitern fehlt bei ihren Entscheidungen das Kostenbewusstsein.
- Die Spezialisten genießen ihre Macht und sind entsprechend arrogant.

Aus Sicht der Mitarbeiter (Frosch-Perspektive) stellt sich das natürlich anders dar. Als Ursachen werden hier gesehen:

- Die Mitarbeiter erhalten keine Informationen über den Projektstand, über Änderungen der Rahmenbedingungen und über Entscheidungen, die das Projekt betreffen.
- Den Mitarbeitern werden keine Anreize geboten, sie werden nicht motiviert.
- Die persönlichen Interessen der Mitarbeiter werden ignoriert.
- Aufgabe, Vollmachten und Verantwortung werden voneinander getrennt.
- Die Führung fehlt; Zielvorgabe, Kontrolle, Beurteilung und Entscheidung werden voneinander getrennt.
- Die Manager sind fachlich inkompetent, treffen aber dennoch fachliche Entscheidungen.
- Die Beurteilung der erzielten Ergebnisse fehlt oder bleibt ohne Konsequenzen.
- Die Rahmenbedingungen sind schlecht und werden nicht verbessert.

- Das Management und – vor allem – der Verkauf treffen ohne Rücksprache Entscheidungen, die den Projekterfolg gefährden.

Die häufigsten Fehler des Projektleiters sind:

- *Keine Führung*
  Der Projektleiter ist nicht präsent, er kommuniziert nicht oder zu wenig.
- *Keine Beobachtung*
  Der Projektleiter nimmt die Arbeit der Mitarbeiter nicht wahr.
- *Keine Bewertung*
  Der Projektleiter bewertet die Arbeitsergebnisse nicht und bildet sich kein Urteil.
- *Keine Konsequenzen*
  Der Projektleiter trifft keine Entscheidungen als Folge von Bewertungen.
- *Technokratisches Verhalten*
  Der Projektleiter respektiert nicht das Bedürfnis der Mitarbeiter nach Information, Bewertung und Anerkennung; er berücksichtigt die persönlichen Interessen, z. B. nach Fortbildung, nicht oder zu wenig.

8.10.4 Regeln für das Projektmanagement

Auswahl und Tätigkeit der Projektleiter

- Projektleiter müssen so ausgewählt werden, dass sie aufgrund ihrer Fähigkeiten in der Lage sind, das Projekt zu leiten. Gute Entwickler sind nicht immer auch gute Projektleiter; wenn auch eine technische Karriere angeboten wird, sinkt der Druck, etwas zu tun, was manchem nicht liegt.
- Projektleiter müssen geschult, ernst genommen und kontrolliert werden.

Kompetenz und Verteilung

- Aufgabe, Vollmacht, Verantwortung und Erfolg dürfen niemals voneinander getrennt werden.
- Bei der Arbeitsteilung müssen die Zuständigkeiten für beide Seiten immer klar geregelt sein; es gilt das Vertragsprinzip.
- Die Ziele bezüglich Terminen, Kosten, Ergebnissen und deren Qualität müssen klar definiert sein.

Kontrolle und Beurteilung

- Die erreichten Zwischen- und Endergebnisse, auch der festgestellte Projektfortschritt, müssen laufend gegen die definierten Ziele kontrolliert werden.

- Zuerst müssen die Ergebnisse beurteilt werden, dann sind die Mitarbeiter darüber zu informieren, anschließend müssen die notwendigen Konsequenzen (Belohnung, Schulung oder sonstige Änderung) daraus gezogen werden.

Zeitlos gültige Regeln

Metzger (1981) gibt in seinem Buch die folgenden Regeln an, um ein Projekt erfolgreich abzuschließen.

Rules of behavior for successful project management

1. *Think people first, the business second. All a business is, is its people. Take care of them.*
2. *Establish a clear definition of your project's development cycle and stick to it.*
3. *Emphasize the front-end of the project so that the rear-end won't be dragging.*
4. *Establish baselines early and protect them from uncontrolled change.*
5. *State clearly the responsibilities of each person on the project.*
6. *Define a system of documents clearly and early.*
7. *Never give an estimate or an answer you don't believe in.*
8. *Never forget Rule 1.*