Zeitschrift: Comtec : Informations- und Telekommunikationstechnologie = information and telecommunication technology

Herausgeber: Swisscom

Band: 76 (1998)

Heft: 5

Artikel: Dompteur, Coach oder Mädchen für alles?

Autor: Ludewig, Jochen

DOI: https://doi.org/10.5169/seals-877302

Dompteur, Coach oder Mädchen für alles?

Foto H. R. Bramaz

Softwareentwicklung ist allen gegenteiligen Behauptungen zum Trotz ein schwieriges Unterfangen, so schwierig, dass sie bei weitem nicht in allen Projekten gelingt. Die Erfahrungen zeigen, dass Projekte seltener an technischen Gründen scheitern als an Gründen, die in ihrer Planung und Leitung liegen. Damit spielt der Projektleiter – oder die leider seltenere Projektleiterin – eine Schlüsselrolle; keine andere Einzelperson hat soviel Einfluss auf den Erfolg eines Softwareprojektes.

Die wesentlichen Aspekte eines Projekts sind, dass es für bestimmte Zeit läuft, dass bestimmte Ziele damit verfolgt werden, dass es instabilen Rahmenbedingungen unterworfen ist, dass es darin Menschen und Ressourcen gibt, Strukturen im Innern und Schnittstellen nach aussen.

JOCHEN LUDEWIG, STUTTGART

Was ist ein Softwareprojekt?

Wir stellen fest:

- Hinter jedem Projekt steht ein «Erzeuger», also eine Person, eine Gruppe oder eine Institution, die dieses Projekt initialisiert hat. In den meisten Fällen ist der «Erzeuger» das Management der Firma, in der die Software entwickelt wird.
- Jedes Projekt hat einen Zweck, das heisst, mit dem Projekt wird ein Ziel verfolgt. In aller Regel ist das zentrale Ziel eines Softwareprojekts, eine Software herzustellen oder zu verändern; diese Software ist also das Resultat des Projekts. Weitere Ziele sind beispielsweise die Erweiterung des Know-hows, die Auslastung der Mitarbeiter, die Bereitstellung von Softwarebausteinen für spätere Projekte.
- Das Produkt hat einen Abnehmer oder wird in der Erwartung entwickelt, dass sich, wenn es fertig ist, Abnehmer bzw. Kunden dafür finden werden. Meist gehören die späteren Benutzer der Software entweder zur Organisation des Kunden, oder der Kunde verkauft seinerseits ein System, das die Software enthält.
- Zu jedem Projekt gehören Menschen und Hilfsmittel (Ressourcen), die dem Projekt ganz oder teilweise zur Verfügung stehen. Durch die spezielle Organisation sind die Rollen und Beziehungen der Menschen vorgegeben.
- Die Abgrenzung des Projekts zur Umgebung ist unscharf und veränderlich, weil es interne oder externe Mitarbeiter gibt, die im engeren Sinne nicht zum Projekt zählen, aber darin mitwirken; die Zugehörigkeit zum Projekt ist nicht konstant.

Der Prozess

Ein (Softwarebearbeitungs-)Prozess ist diejenige Folge von Handlungen, durch die ein Softwaresystem entsteht, das heisst, ein Prozess wird immer wieder angewandt und dabei fortentwickelt.
Wenn beispielsweise mit dem CMM (Capability Maturity Model) der Reifegrad einer Institution festgestellt wird, steht der Prozess auf dem Prüfstand.
Wenn durch Abstraktion die konzeptionellen Gemeinsamkeiten verschiedener Prozesse (z. B. in verschiedenen Firmen) freigelegt sind, haben wir ein Prozessmodell vor uns. Wasserfall- und Spiralmodell sind Beispiele solcher Prozessmodelle (Bild 1).
Prozesse und Prozessmodelle sind konservierte Erfahrungen. Ihre bewusste Anwendung kann daher die Arbeit des Projektmanagers erheblich erleichtern. Eine Gefahr besteht darin, dass ein ungeeigneter Prozess angewendet wird. Allerdings bedeutet der Verzicht auf eines der Standardmodelle in der Regel die Anwendung des «Steinzeitmodells» Code & Fix: Codieren, Ausprobieren und Verbessern, bis es zu funktionieren scheint. Dieses Vorgehen hat aber besonders viele Nachteile.
Der extreme Gegensatz zum Code & Fix ist das (ursprüngliche) Wasserfallmodell, das gekennzeichnet ist durch eine geplante Gliederung des Projekts in Schritte, die im wesentlichen sequentiell durchlaufen werden: Analyse, Spezifikation, Systementwurf, Modulspezifikation, Codierung, Integration und Test, Abnahme, Installation, Betrieb und Wartung. Aber auch damit gibt es Probleme, und es ist schwer zu entscheiden, wie weit diese durch erlernbares Verhalten beseitigt werden können.

Organisationsstrukturen

Conway hat gelehrt, dass Organisationen dazu neigen, Artefakte zu schaffen, in denen sie ihre eigene Struktur reproduzieren. Das bedeutet beispielsweise, dass ein Entwicklungsteam aus zwei Personen eine Software entwickeln wird, die aus zwei ähnlich grossen Modulen besteht. Man kann einen Schritt weitergehen und vermuten, dass die entstehenden Komponenten die Ziele und Interessen derer spiegeln, die sie entwickelt haben.

Für die Organisation eines Softwareprojekts hat diese Regel einige Bedeutung: Einerseits ist es zweckmässig, personelle Strukturen zu schaffen, die mit einer sinnvollen Produktstruktur korrespondieren. Andererseits müssen Strukturen daraufhin geprüft werden, ob sie wirklich technisch gerechtfertigt sind. Und auf allen Ebenen muss die Frage gestellt werden, ob die Interessen der Individuen und Gruppen miteinander verträglich sind und die gewünschte Resultierende haben.

Menschliche Aktivitäten, also auch Softwareprojekte, sind meist hierarchisch organisiert. Eine Person übernimmt oder erhält die Vollmachten und die Verantwortung für das Projekt, wird zum Kapitän der anstehenden Reise, als Projektleiter oder als Projektmanager.

Wer auf irgendeiner Stufe der Hierarchie zwischen ganz unten und ganz oben steht, muss sich entscheiden, wessen Interessen er vertritt. Die beiden extremen Positionen, die ein Projektleiter theoretisch einnehmen kann, sind einerseits die des «Räuberhauptmanns», der uneingeschränkt die Interessen seiner Gruppe vertritt, andererseits diejenige des «Landvogts», der die Weisungen eines fernen Herrschers ausführt. Offensichtlich ist der «Landvogt» als Projektmanager ungeeignet, weil er nicht das Vertrauen der übrigen Mitarbeiter geniesst. Der «Räuberhauptmann» ist dagegen in Gefahr, die Interessen des Unternehmens aus den Augen zu verlieren, beispielsweise in Fragen der Standardisierung oder der Wiederverwendung. Der Softwareprojektmanager muss also diese beiden Rollen im richtigen Verhältnis mischen.

Mit der Rolle des Projektleiters oder Projektmanagers verbinden wir also die Vorstellung eines hierarchischen Teams. Diese Struktur entspricht nicht nur dem hierarchischen Aufbau der Organisation insgesamt, sondern stellt auch eine einfache Kommunikation und ein klares Verständnis der Zuständigkeiten sicher.

Auch das Chief Programmer Team ist hierarchisch organisiert, aber es funktioniert ganz anders. Denn hier ist der Chief Programmer eine Art Vorarbeiter, also jemand, der selbst entwickelt und nur delegiert, was nicht extrem wichtig ist. Die übrigen Gruppenmitglieder haben teilweise spezielle Rollen, etwa die des Junior Chiefs oder des Bibliothekars (Librarian). Diese Organisationsform wurde – wenigstens in Europa – nirgends übernommen; viele Softwareprojektmanager verhalten sich aber so, als ob sie sich nicht zwischen ihrer hierarchischen Funktion und der Rolle des Chief Programmers entscheiden könnten.

Diese vom Operationsteam entlehnte Idee kann auch variiert werden, beispielsweise in der Form, dass jemand die Architektur der Software festlegt und dann die Realisierung leitet, ohne selbst zu implementieren. In diesem Falle steht das Team unter der Führung des Architekten; wir haben so etwas wie ein «Chief Architect Team» vor uns.

Vielfach kommt in der Praxis eine andere Form vor, die aber kaum je so klar wie hier benannt wird, nämlich das anarchische Team. Darin arbeiten die Entwickler im wesentlichen autonom, nach eigenen Vorgaben und Massstäben. Hierarchische Beziehungen fehlen oder werden faktisch ignoriert, weil der Wille, die Zeit oder die Fähigkeit zur Führung fehlt. Auch diese Struktur hat Vorteile: Die Entwickler sind selbstbestimmt, erleiden keine Hierarchieprobleme und kaum bürokratische Hemmnisse. Aber Standards und Normen lassen sich nicht durchsetzen. Die Entstehung der erforderlichen Resultate ist Glückssache (d. h. gewisse Dokumente entstehen in aller Regel nicht). Die Organisation insgesamt ist nicht lernfähig, Planung, Einführung neuer Methoden und Werkzeuge sind von der Laune der Mitarbeiter abhängig. Glaubt man den Organisationsdiagrammen, dann gibt es in der Wirtschaft nur hierarchische Teams. Die Realität sieht anders aus: Zum einen sind viele Entwicklergruppen in Wahrheit anarchisch, weil die Führung schwach ist. Zum anderen gibt es viele Projektleiter, die sich am wohlsten fühlen, wenn sie selbst ent-

Kommentar

Softwareprojektleiter zwischen Technik und Management

Softwareprojektleiter kommen zum überwiegenden Teil aus dem Entwicklungsbereich. Sie verfügen über gute Kenntnisse in der Technik und/oder in der Anwendung, besitzen aber oft nur unzureichende Kompetenz in ihren Managementaufgaben. Trotzdem widerstrebt es ihnen, sich ganz von der Entwicklung zu lösen, denn das war der Beruf, für den sie sich vor einigen Jahren entschieden und qualifiziert und innerhalb dessen sie auch ihre ersten beruflichen Erfolge erzielt hatten.

Persönlichkeiten mit der für diese Aufgabe notwendigen Qualifikation sind nicht sehr häufig, und die Hochschulen tragen kaum dazu bei, dass ihre Zahl genügend rasch steigt. Sie konzentrieren sich traditionellerweise auf die Heranbildung der technischen Kompetenz und weniger auf die Vermittlung solcher Kenntnisse und Fähigkeiten, wie sie in der untersten Führungsebene gebraucht werden. Aus diesem Grund wird die Kluft zwischen dem Angebot und der steil ansteigenden, die Bedeutung der Software in Produkten und Dienstleistungen widerspiegelnden Nachfrage immer breiter. Projektleiter müssen somit auch in Zukunft ganz überwiegend aus dem Kreis der Softwareentwickler herangebildet werden.

Der frisch gebackene Projektleiter sucht sich in der Regel einen Weg zwischen den Extremen. Er nimmt die ungewohnte Leitungsfunktion wahr, so gut und soweit es geht. Er bleibt aber im Herzen noch lange Zeit Entwickler und übt diese Tätigkeit auch noch gern aus, wenn sich die Gelegenheit dazu bietet. Damit kommen einige Aspekte der Projektleitung leider oft zu kurz, beispielsweise die Planung, das Risikomanagement oder die Qualitätssicherung. Der Projektleiter muss lernen, seine eigentliche Aufgabe zu erkennen und solche Tätigkeiten mit Priorität auszuüben, die dem Projekt insgesamt dienen, die Gefahren für das Projekt mindern und darüber hinaus für das gesamte Unternehmen positiv wirken. Da die Arbeit immer unter grossem Zeitdruck steht, ist es für ihn wichtig, zwischen zentralen und peripheren Problemen, zwischen unaufschiebbaren und weniger drängenden Fragen, zwischen delegierbaren Aufgaben und solchen, um die er sich selbst kümmern muss, unterscheiden zu können.

Jochen Ludewig

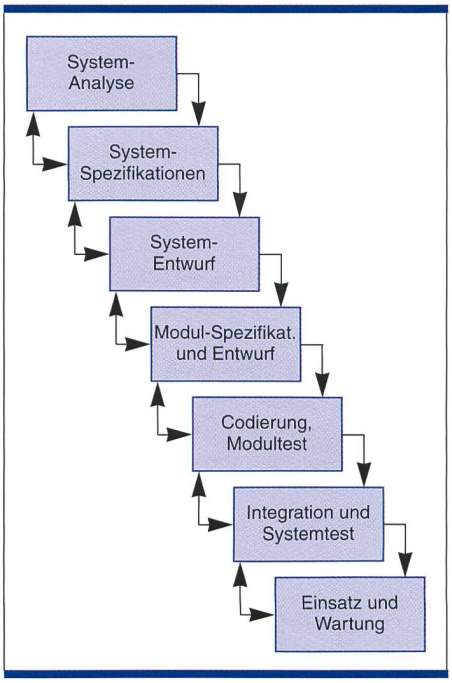

Bild 1. Das Wasserfallmodell.

wickeln. Sie sind verhinderte Chief Programmers, haben aber keine Chance, die Stärken des Chief Programmer Teams zu entfalten, weil sie von ihrer Umgebung auf die Funktion des Vorgesetzten festgelegt sind. Auf diese Weise entsteht eine besonders unglückliche Konstellation.

Die Einbettung des Projekts

Wir gehen hier von der Situation aus, dass das Projekt in der klassischen Projektorganisation durchgeführt wird; hierbei gehören dem Projekt die Mitarbeiter an, solange das Projekt läuft (oder wenigstens so lange sie gebraucht werden). Bei der Organisation von Projekten in einer funktionalen oder in einer Matrixstruktur kommen Führungsprobleme hinzu.

Der Kapitän eines Schiffes, der Regisseur eines Films: Das sind keine absoluten Herrscher, die über sich nur den gestirnten Himmel und unter sich nur willige Sklaven haben. Sie sind vielmehr in ein Geflecht von Abhängigkeiten eingebunden.

Offensichtlich gibt es in der Hierarchie höheres Management/«Projektleiter»/Projektmitarbeiter eine Abhängigkeit von unten nach oben (d.h. die Mitarbeiter sind vom Softwareprojektmanager abhängig), aber auch in der umgekehrten Richtung, denn das Management braucht den Erfolg des Projektleiters, so wie dieser die Kooperation seiner Mitarbeiter braucht. Schliesslich besteht auch mit dem Kunden eine wechselseitige Abhängigkeit.

In den beiden Situationen «Kapitän» und «Regisseur» hat sich eine Dreiecksstruktur entwickelt, die seit einiger Zeit auch im Bereich der Software Bedeutung gewinnt (Bild 2).

Resultate einer Analyse

Lichter und Mandl-Striegnitz haben 1996 die Probleme von Projektleitern in einer ganz bestimmten Umgebung untersucht. Die Essenz ihrer Ergebnisse steckt in den folgenden Feststellungen. Sie sind nicht wörtlich zitiert, sondern teilweise anders formuliert, auch anders gedeutet.

– Je weniger Zeit ein Projektleiter für das Projektmanagement hat, desto eher kommt es zu Termin- und Kostenschwierigkeiten.

Wenig Zeit (d. h. 5–35%) bedeutet sehr eingeschränkte Fortschrittskontrolle; Projektleiter, die sich für das Projektmanagement Zeit nehmen, benötigen deutlich weniger Kommunikationsaufwand mit ihren Mitarbeitern als andere (systematische Führung statt Krisenmanagement).

– Es besteht eine enge Beziehung zwischen der Person des Projektleiters und der Qualität seiner Leitungstätigkeit.

Projektleiter können als «technikorientiert» oder als «managementorientiert» eingestuft werden.

– Das höhere Management der Unternehmen ist sich der Wichtigkeit der Projektleitertätigkeit nicht bewusst.

Hat ein Unternehmen überdurchschnittlich viele «technikorientierte Projektleiter», lässt sich daraus schliessen, dass die Bedeutung der Projektleitertätigkeiten für den gesamten Projekterfolg nicht ausreichend bekannt ist.

– Die Projektleiter verwenden kaum Planungs- und Managementtechniken.

Die Untersuchung hat ergeben, dass die Projektleiter generell keine Risikomanagementverfahren und keine oder nur wenige Schätzverfahren verwenden. Die Planungstechnik des Risikomanagements wurde von keinem der befragten Projektleiter angewendet. Risikofaktoren, wie beispielsweise die Zuordnung der Mitarbeiter zu verschiedenen Projekten oder Abhängigkeiten von Zulieferungen, sind den Projektleitern zwar durchaus bewusst, finden jedoch bei der Planung der Projekte nur wenig Beachtung. Dynamisches (iteratives) Vorgehen bei der Durchführung der Kosten- und Aufwandschätzung ist vorteilhaft.

– Wichtige quantitative Daten stehen der Projektplanung nicht zur Verfügung.

Die meisten Projektleiter kennen wichtige Daten, die sie zur Planung der Projekte benötigen würden nicht, beispielsweise den durchschnittlichen Anteil einer Phase am Gesamtaufwand, die einplanbare Produktivität ihrer Mitarbeiter, die Kommunikationsaufwände.

– Verteiltes Entwickeln führt zu erheblichen Projektverzögerungen.

Verteilte Entwicklung, also die Kooperation mit Subunternehmen oder der Einsatz externer Mitarbeiter, bewirkt Steigerungen des Kommunikationsaufwands (über 30%) und Informationsverluste. Diese Probleme werden bei der Planung der Projekte nicht berücksichtigt.

– Der Aspekt der Qualitätssicherung wird vernachlässigt.

Qualitätssicherung existiert nicht oder ist Aufgabe der Entwickler selbst; sie wird

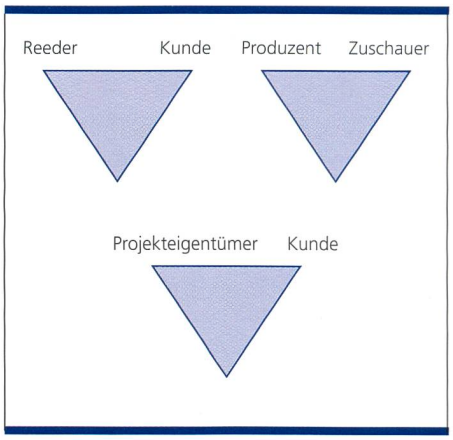

Bild 2. Reeder, Kapitän und Kunde. Rechts steht in allen Fällen der Kunde. Unten steht derjenige, der sichtbar Verantwortung trägt. Links oben steht der Eigentümer des Projekts, der Reeder, der Produzent. Seine Rolle ist in allen Fällen, die Mittel für die Reise, den Film, das Projekt bereitzustellen, dabei aber nicht selbst an der Überfahrt, den Dreharbeiten oder dem Projekt teilzunehmen, sondern von aussen eine Aufsichtsfunktion wahrzunehmen und seinerseits das Projekt in der höheren Hierarchie zu vertreten (oder das Schiff gegenüber den Behörden, den Film gegenüber den Banken).

Guter Rat

Ratschläge erfahrener Fachleute haben etwas ermüdendes, weil sie als Stimmen aus dem Off zu hören sind. Bei näherer Betrachtung der folgenden Regeln (aus Metzger, 1981, S. 191) zeigt sich aber, dass Metzger keineswegs auf der Wolke thront, sondern die Probleme sehr konkret anspricht.

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

Software-Engineering zu lehren ist immer schwierig, weil Software-Engineering so einfach ist. Beim Thema Softwareprojektmanagement gilt dies doppelt: Die Beachtung von Metzgers Regeln löst bereits die meisten Probleme, doch erst die Beachtung löst die Probleme, nicht ihre Kenntnis.

bei Terminproblemen reduziert. Zusätzliche Mitarbeiter werden bewilligt, wenn der Termin gefährdet ist, nicht aber, wenn die Qualität flau herauskommt.

– Eine intensive und regelmässige Kommunikation mit dem Auftraggeber und dem Benutzer hat in der Regel positive Effekte auf den Projektverlauf.

Auf diese Weise entstehen vollständigere und klarere Anforderungen, selbst bei Anforderungsänderungen, und das Projekt wird transparenter.

– Die Projektleiter wünschen sich einen intensiveren Kontakt zum Senior Management.

Besonders von den höheren Ebenen fühlen sich die Projektleiter bei der Durchführung der Projekte alleingelassen.

Jochen Ludewig, *Studium der Elektrotechnik in Hannover, Informatik in München. Sechs Jahre staatliche Forschungseinrichtung, Promotion. Bis 1985 BBC-Forschungszentrum, Baden-Dättwil. Drei Jahre Assistenzprofessor an der ETH Zürich. Seit 1988 Lehrstuhl Software-Engineering an der Universität Stuttgart.*

Thesen zur Situation vieler SPM

Die wiedergegebenen Feststellungen von Lichter und Mandl-Striegnitz sind zusammen mit eigenen Beobachtungen zu den folgenden Thesen verdichtet:

– Viele Softwareprojektmanager sind fehlqualifiziert.

Viele Softwareprojektmanager waren zwar gute Entwickler, haben aber den Beruf des Softwareprojektmanagers nicht erlernt und bekommen dazu auch keine Chance, suchen sie auch nicht.

– Mitwirkung verdrängt Leitung.

Die Tätigkeit eines Softwareprojekt-Managers besteht faktisch aus den drei Komponenten Planung und Überwachung, Aussenvertretung, Mitwirkung im Projekt. Die Mitwirkung sollte die Ausnahme sein, ist aber vielfach die Regel. Darunter leidet die Planung und Überwachung. Oder anders gesagt, ein Kapitän, der eine Nebentätigkeit als Maschinist hat, wird in der Regel den Eisberg zu spät entdecken.

– Viele Softwareprojektmanager fühlen sich vom oberen Management im Stich gelassen.

Dieses Gefühl ist in vielen Fällen begründet, denn das Management bietet zu wenig Führung und Sicherheit.

– Viele Softwareprojektmanager fürchten die Fakten, statt sich ihrer zu bedienen.

«Wissen ist Macht.» Dieser Satz gilt ohne Zweifel auch und besonders für den Softwareprojektmanager. Wenn aber die Realität allzuoft ausgeblendet wird, damit Probleme weniger bedrohlich erscheinen, verkommen Fakten zu Bedrohungen. Das zeigt sich vor allem in der Haltung zu Metriken und zum Risikomanagement.

– Ein Softwareprojektmanager, der die Wahrheit sagt, fällt auf.

Natürlich fällt er in den meisten Firmen leider negativ auf. Das ist falsch und die Schuld des höheren Managements. Spricht man mit einem typischen Softwareprojektmanager, so hört man fast sicher Sätze der Art: «Wir müssten eigentlich ...» oder: «Im Grunde wissen wir, dass ...». Es ist fatal, wenn sich Menschen daran gewöhnt haben, ihre Einsichten zu ignorieren.

– Die gängige Auffassung von Helden und Langweilern ist Gift für die Softwareprojekte.

Es liegt in der Natur des Menschen, Lebensretter zu feiern, Unfallverhüter aber auszulachen. Ein vernünftiges Management müsste durch sorgfältige Datenerfassung, Projektverfolgung und Retrospektive dafür sorgen, dass diese Neigung nicht das Wertesystem in der Firma prägt, dass also nicht die aus der Hüfte feuernden Desperados, sondern die zuverlässigen «Langweiler» die Projekte leiten. [11]

Referenzen

M.E. Conway: How do committees invent? DATAMATION, April 1968, 28–31.

P. Mandl-Striegnitz, H. Lichter (1996): Softwareprojektmanagement in der Industrie – Erfahrungen und Analysen. Bericht SL-2196, Software-Labor Stuttgart.

R.H. Thayer (ed.) (1988): Software Engineering Project Management. Computer Society Press of the IEEE, Washington D.C. (2nd ed. 1997).