

Wolfgang Fahl
Frank Hoffmann

Erfolgsfaktoren der Softwareentwicklung

**Best Practises für den Weg von den
Anforderungen zum richtigen Produkt**

HANSER



Fahl/Hoffmann
Erfolgsfaktoren der Softwareentwicklung

Wolfgang Fahl
Frank Hoffmann

Erfolgsfaktoren der Softwareentwicklung

**Best Practises für den Weg von den
Anforderungen zum richtigen Produkt**

HANSER

Wolfgang Fahl, Willich-Schiefbahn
wf@bitplan.com

Frank Hoffmann, Kaarst
fh@bitplan.com

www.hanser.de

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Information – oder Teilen davon – entsteht.

Ebenso übernehmen Verlag und Autoren keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Die Deutsche Bibliothek - CIP Einheitsaufnahme
Ein Titeldatensatz für diese Publikation
ist bei der Deutschen Bibliothek erhältlich

Dieses Werk ist urheberrechtlich geschützt.
Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) - auch nicht für Zwecke der Unterrichtsgestaltung - reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2002 Carl Hanser Verlag, München Wien
Lektorat: Margarete Metzger
Copy-editing:
Herstellung: Irene Weilhart
Umschlaggestaltung:
Datenbelichtung, Druck und Bindung:
Printed in Germany

ISBN 3-446-22305-3

Inhalt

Vorwort der Autoren	1
Die Brücke als Sinnbild	1
Die Umfrage.....	5
Literaturtipps.....	7
Das Projektverwaltungs-Projekt.....	8
1 Von Menschen und Computern.....	9
1.1 Menschen als Handelnde.....	9
1.2 Die zwei Blickwinkel - Mensch und Maschine.....	11
1.3 Die "Brücke" vom Wunsch zur Software.....	13
1.4 Welches Vorgehensmodell?.....	18
1.5 Gemeinsames Erfolgsverständnis	19
1.6 Die Sprache des Auftraggebers sprechen.....	24
1.7 Wie wird der Auftraggeber glücklich?.....	25
1.8 Wie sicher kann ich sein, dass die Software richtig ist?	27
1.9 Literatur	29
2 Was soll die Software leisten?	31
2.1 Projektauswahl	31
2.2 Anforderungen aufnehmen.....	32
2.3 Prüfbeispiele dokumentieren.....	40
2.4 Anforderungsprofil erstellen	42
2.5 Änderungen der Anforderungen.....	45
2.6 Der Wert von Software	48
2.7 Literatur	50
3 Vom Anforderungsprofil zum Modell.....	51
3.1 Modellieren	51
3.2 Objektorientierter Ansatz	53
3.3 Ableitung des Modells aus den Anforderungen	55
3.4 Einfach, kurz und klar - komplizierte Modelle sind falsch!.....	61
3.5 Die Abstraktionsfalle.....	63
3.6 Literatur	66
4 Vom Modell zum Code	69
4.1 Umsetzen.....	69
4.2 Kurze Entwicklungszyklen.....	77
4.3 Abnahme und Einführung	77
4.4 Literatur	78

5	Änderung, Wartung, Pflege	79
5.1	Schrittweises Vorgehen.....	79
5.2	Fehlermeldungen und Änderungswünsche	79
5.3	Literatur.....	81
6	Infrastruktur und Werkzeuge	83
6.1	Arbeitsumgebung.....	83
6.2	Werkzeugauswahl	87
6.3	Einzelteile und Versionen	92
6.4	Generierung.....	95
6.5	Literatur.....	96
7	Projektmanagement, Kommunikation, Politik	97
7.1	Zielorientierung.....	99
7.2	Entscheidungen fällen	101
7.3	Vom Umgang mit Eisbergen.....	103
7.4	Das Team	105
7.5	Führung und Förderung.....	110
7.6	Projektplanung und Steuerung	112
7.7	Umgang mit dem Zufall	114
7.8	Erfahrungen teilen.....	118
7.9	Literatur.....	120
8	Computerfachchinesischlexikon.....	121
9	Anhänge	123
9.1	Liste der Erfolgsfaktoren.....	124
9.2	Fragebogen.....	129
9.3	extreme Programming	131
9.4	Internet Referenzen	132
9.5	Literaturverzeichnis.....	133
9.6	Index.....	134

*"Es wäre gut Bücher kaufen, wenn man die Zeit, sie zu lesen, mitkaufen könnte, aber man verwechselt meistens den Ankauf der Bücher mit dem Aneignen ihres Inhalts."
Arthur Schopenhauer*

Vorwort der Autoren

Herzlichen Glückwunsch: Sie sind bereit, aus den Erfahrungen anderer zu lernen! Sie haben dieses Buch in die Hand genommen und begonnen zu lesen. Sie interessieren sich für Softwareentwicklung und wollen die wesentlichen Faktoren kennenlernen, die dazu führen, dass auch Ihr Softwareprojekt ein Erfolg wird.

Die Brücke als Sinnbild

Die Autoren haben auf der ICRE'98 in Colorado Springs, USA, erlebt wie es möglich ist, in kurzer Zeit das Wissen von vielen Dutzend Experten zu nutzen, um ein gemeinsames Ergebnis zu schaffen. Gary Weinberg sollte damals einen Vortrag zum Thema Anforderungsmanagement halten. Am Anfang des Vortrages sagte er jedoch zu den zweihundert Softwareexperten im Saal, unter denen viele angesehene Namen wie Larry Constantine und Ed Yourdon waren, dass sein Wissen nicht ausreichen würde, um für die Teilnehmer etwas wirklich Interessantes, Wichtiges oder Neues zu bieten. Daher bat er die Teilnehmer, den Vortrag selbst zu gestalten. Es wurden vier Gruppen gebildet, die jeweils eine halbe Stunde Zeit bekamen ein Thema zu bearbeiten. Das "Red Team" bearbeitete die Frage "Was wäre für Sie die wertvollste Fähigkeit, die eine Person erwerben müsste, um in der Softwareentwicklung übergreifend erfolgreich zu sein?" Als Antwort entstand die folgende Skizze:

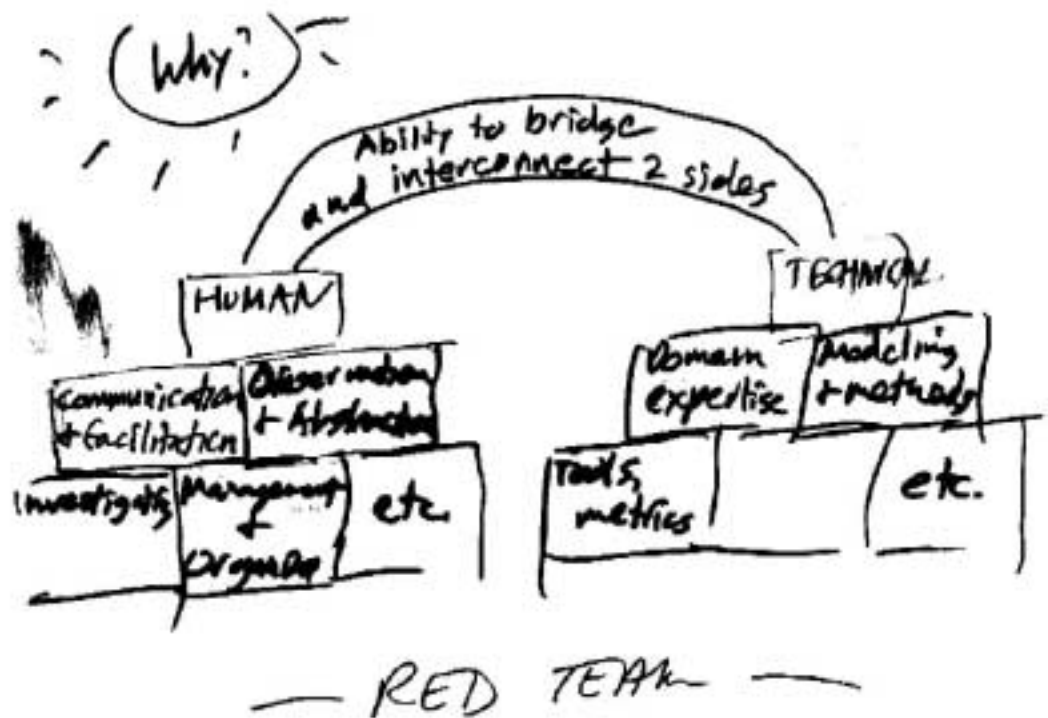


Abbildung 0.1: Die Brücke als Sinnbild für die Softwareentwicklung

Softwareentwicklung wurde von den etwa fünfzig Experten im roten Team als das Schlagen der Brücke von Menschen zur Computertechnik interpretiert. Die Fähigkeit, diese beiden Seiten miteinander zu verbinden, also menschliche Faktoren wie Kommunikation, Moderation, Forschen, Abstrahieren, Management und Organisation usw. mit technologischen Faktoren wie Fachwissen, Modellierung und Methoden, Werkzeugen usw. in Einklang zu bringen, wurde als Schlüssel zum sonnigen Leben unter der ewigen Frage "Warum" herausgestellt.

Die Brückenmetapher hat uns seither in vielen Projekten geholfen, die Herausforderung der Softwareentwicklung klarer zu sehen. Sie hat uns geholfen, Erfolgsfaktoren zu identifizieren.

Wie ist es Ihnen bisher in Ihren Projekten ergangen? Gehören Sie zu den wenigen Glücklichen, die bisher nur an erfolgreichen Projekten beteiligt waren? Wenn ja, dann sind wir daran interessiert zu erfahren, inwieweit die hier genannten Erfolgsfaktoren mit Ihren Erfahrungen übereinstimmen. Nehmen Sie Kontakt mit uns auf! Was konnten Sie Neues entdecken? Was haben Sie anders erlebt? Unsere Kontaktangaben finden Sie am Ende des Vorworts.

Wenn Sie wie wir jedoch zu denjenigen gehören, die gemischte Erfahrungen mit Softwareprojekten gemacht haben oder wenn Sie noch keine Erfahrungen haben, dann wird Ihnen dieses Buch praktische Hinweise geben, wie Sie Ihre Erfolgchancen erhöhen können.

Softwareprojekte sind unendlich komplexe Vorhaben. Es ist leicht, "den Wald vor lauter Bäumen nicht mehr zu sehen". Indem Sie sich die in diesem Buch genannten Faktoren zu eigen machen werden Sie sich erheblich sicherer in Ihrem Softwareprojekt fühlen - was immer auch Ihre spezielle Softwareaufgabe sein mag. Softwareprojekte bergen ein großes Frustrationspotential und oft führt dies zu persönlichen Folgen bei den Beteiligten. In den Projekten, die wir als Referenz heranziehen haben wir oft erlebt, wie das Wesentliche aus den Augen verloren wurde und Nebenkriegsschauplätze in den Vordergrund traten. Profitieren Sie vom Blick auf das Wesentliche! Sie werden feststellen, wie sich Ihre Herangehensweise an Softwareprojekte verändert. Ihre Auftraggeber werden eine höhere Sicherheit haben, dass die Software, die Sie liefern, ihre Wünsche erfüllt, und Sie werden voraussichtlich dafür gelobt werden. Ihren Mitarbeitern werden Sie besser vermitteln können, worauf es in Softwareprojekten ankommt. Der Beitrag Ihres Teams zu diesen Veränderungen wird für alle Beteiligten positiv spürbar werden. In den Organisationen wird Software immer mehr zum kritischen Faktor - Ihre Organisation wird also erfolgreicher werden, da sie mit größerer Zuverlässigkeit "die richtige Software" liefern kann oder erhält. Ihre Mitarbeiter werden begeistert und Ihre Kunden glücklich sein.

Wir wollen Ihnen mit diesem Buch:

- einen einfach verständlichen Weg zur Entwicklung von Software aufzeigen.
- Ihnen schrittweise vermitteln, was notwendig ist, um Software erfolgreich zu entwickeln.
- Ihr Verständnis dafür wecken, dass die Punkte, die wir nennen unabhängig vom Inhalt oder der Art Ihres Projektes anwendbar sind.
- erreichen, dass Sie die Erfolgsfaktoren in Ihrem Projekt zur Wirkung bringen.

In diesem Buch werden Sie lernen, die Softwareentwicklung aus zwei Blickwinkeln zu betrachten - vom Menschen her und vom Computer her. Durch diese Trennung wird es möglich, den "politischen" Teil der Softwareentwicklung - die menschliche Seite der Aufgabenstellung - so getrennt wie möglich vom inhaltlichen Teil zu betrachten und zu bearbeiten. Damit werden Sie bereits viele Probleme vermeiden können, die in Projekten durch die Mischung dieser Blickwinkel entstehen. Die "Brücke" wird als Sinnbild für die Umsetzung der Wünsche der Beteiligten in die richtige Software dienen. In fünf Kapiteln wird die Brücke schrittweise vorgestellt und zu jedem Schritt über die Brücke werden die wesentlichen Erfolgsfaktoren aufgezeigt. Damit legen wir ein solides Fundament, auf das Sie in Ihrer Projektgestaltung aufbauen können. Ein durchgängiges Beispiel für den Brückenschlag hilft Ihnen beim Verständnis und der Umsetzung in Ihrem Alltag - dieses Beispiel ist so angelegt, dass alle wesentlichen Aspekte dieses Buches daran nachvollziehbar sind. Jedes Kapitel ist mit einer Liste von

des Kapitel ist mit einer Liste von weiterführender Literatur versehen, die Sie für die Klärung von Detailfragen zu Rate ziehen können.

Wenn Sie dieses Buch gelesen haben und Ihre Mischung von Erfolgsfaktoren für die persönliche Umsetzung in Ihrem Umfeld ausgewählt und angewendet haben, hoffen wir, dass Sie unsere Webseite www.b-agile.de/Erfolgsrezepte/feedback.html !!! besuchen und uns an Ihren eigenen Erfahrungen teilhaben lassen. Dadurch werden wir erfahren, welchen Nutzen Ihre Erfolgsrezepte für Sie entfalten. Wir sind fest davon überzeugt, dass über 95% der Antworten positiv ausfallen werden, da Sie den sofortigen Nutzen in Ihrem Projekt spüren werden, den der Blick auf das Wesentliche Ihnen bietet.

Die Sammlung der Erfolgsfaktoren in diesem Buch lebt sowohl von unseren eigenen Projekterfahrungen als auch von denen zahlreicher anderer, die Softwareprojekte durchgeführt haben. Wolfgang Fahl war seit 1986 an über 50 Softwareprojekten persönlich beteiligt. Seine Leidenschaft sind Softwareprojekt-Desaster und die Analyse ihrer Ursachen. Frank Hoffmann hat seit 1992 über 110 Trainings zu Themen der Softwareentwicklung durchgeführt und war an Projekten bis zu mehreren hundert Personenjahren maßgeblich beteiligt. Die Essenz aus Dutzenden von Büchern mit den Erfahrungen von Autoren mit Weltrang sind in dieses Buch eingeflossen. Schließlich sind uns in unserem Berufsleben Hunderte von Projektleitern, Softwareentwicklern und Auftraggebern begegnet - alle diese Menschen haben ein komplexes Bild der Softwareentwicklung entstehen lassen, das wir so weit wie es vermögen vereinfachen.

Es gibt nur wenige Projekte, denen es gelungen ist, fast alle in diesem Buch genannten Erfolgsrezepte anzuwenden. Bei einem Finanzdienstleister haben wir es 1999 jedoch auf beeindruckende Weise erlebt. Das Team stand von Anfang an im Vordergrund und einigte sich darauf, wie die Brücke geschlagen werden sollte. Es wurde systematisch auf die Trennung von fachlichen und technischen Themen gesetzt. Anforderungen wurden basierend aus Szenarien mit detaillierten Abnahmekriterien aufgenommen. Es wurde ein gemeinsames Intranet-basierendes Projektstagebuch geführt, das von allen Teammitgliedern intensiv zum Austausch von Randergebnissen und offenen Punkten genutzt wurde. Entscheidungen wurden (nach anfänglichen Schwierigkeiten) systematisch vorangetrieben. Es wurde objektorientiert entwickelt, obwohl als Zielsprache lediglich klassisches COBOL zur Verfügung stand. Die fertige Software ist im April 2001 in Einsatz gegangen und hat bis heute wie gewünscht ihren Dienst getan. Die Software wird weiterentwickelt und ist äußerst anpassungs- und änderungsfreundlich - selbst der Wechsel der Programmiersprache nach Java ist bereits exemplarisch durchgeführt worden. Insbesondere sind alle Projektbeteiligten mit den Ergebnissen äußerst zufrieden und sprechen gerne über dieses Projekt. Sprechen Sie uns an: Wir bringen Sie mit diesem Team in Kontakt!

Die Umfrage

Im Sommer 2002 haben wir mit einer Umfrage begonnen, um festzustellen, welche Relevanz die in diesem Buch genannten Erfolgsfaktoren für reale Projekte haben. Dazu haben wir zunächst 45 Projekte im Umfang von 600 Personenjahren befragt. Für jedes Projekt haben wir erfahren, wie das Projekt seinen Erfolg beurteilt und was die Erfolgskriterien für das Projekt sind. Zu den in diesem Buch genannten Erfolgsfaktoren gaben die Projekte an, ob der Erfolgsfaktor angewendet wurde und welchen Einfluss der Faktor auf den Erfolg des Projektes hatte. Dazu stand eine Skala von „++/stark gefördert“ bis „--stark behindert“ zur Verfügung.

Die Ergebnisse dieser Umfrage sind in dieses Buch eingeflossen. Wir nennen hier die mehrheitlich als fördernd genannten Erfolgsfaktoren. Die Liste der Erfolgsfaktoren wird von uns mit Ihrer Hilfe ständig weiter gepflegt, und sie finden diese immer aktuell unter www.b-agile.de/Erfolgsrezepte. Beteiligen Sie sich dort an der Umfrage und erfahren Sie den aktuellen Stand der Ergebnisse!

CHAOS-Studie der Standish Group

Die CHAOS-Studie wird von der Standish Group [Url5], seit 1994 durchgeführt und hat bisher über 35.000 Projekte untersucht.

Die Chaos-Studie unterscheidet folgende Projektverläufe:

- **Erfolgreich:** Das Projekt wird im Zeit- und Kostenrahmen fertig und liefert alle Eigenschaften und Funktionen wie vereinbart ab.
- **Gefährdet:** Das Projekt wurde fertig und das Ergebnis läuft, aber das Budget oder der Termin wurde überschritten und weniger Eigenschaften und Funktionen wurden geliefert als vereinbart.
- **Gescheitert:** Das Projekt wird vor Fertigstellung abgebrochen, lieferte nie ein Ergebnis ab oder das Ergebnis wurde vor der Installation verworfen.




In der folgenden Tabelle sind die 10 Erfolgsfaktoren aufgeführt, die laut dieser Studie die wichtigsten sind. Wir betrachten diese Erfolgsfaktoren unter ihrer deutschen Bezeichnung in diesem Buch ebenfalls.

Rang	Erfolgsfaktor lt. CHAOS	Erfolgsfaktor in diesem Buch	Seite
1	Executive Support	Förderung durch die Verantwortlichen.	111
2	User Involvement	Identifizieren und beteiligen Sie alle Betroffenen, insbesondere die Anwender.	47
3	Experienced Project Manager	Erfahrener Projektleiter.	111
4	Clear Business Objectives	Definieren Sie die Projektziele klar anhand von messbaren Kriterien und fest-	80

Rang	Erfolgsfaktor lt. CHAOS	Erfolgsfaktor in diesem Buch	Seite
		gelegten Werten.	
5	Minimized Scope	Die richtig priorisierte Auswahl der Anforderungen legt das zu realisierende Anforderungsprofil fest.	42
6	Standard Infrastructure	Kümmern Sie sich rechtzeitig um eine adäquate und standardisierte Infrastruktur/Arbeitsumgebung für Ihr Softwareprojekt.	84
7	Firm Basic Requirements	Achten Sie darauf, dass die Rahmenanforderungen für Ihr Projekt stabil bleiben.	48
8	Formal Methodology	Einigen Sie sich auf ein für Ihr Projekt geeignetes Vorgehensmodell und wenden Sie dieses aus Überzeugung an.	18
9	Reliable Estimates	Je zuverlässiger Ihre Schätzungen sind, desto besser.	111
10	Skilled Staff	Fähige Mitarbeiter.	110

Kennzeichnung der Erfolgsfaktoren

Wir haben viel weggelassen. Das was übrig bleibt, sind unserer Ansicht nach die Punkte, die in praktisch jedem Projekt eine Rolle spielen. Entsprechend der Ergebnisse der Umfrage und der CHAOS-Studie haben wir die Relevanz der Erfolgsfaktoren wie folgt gekennzeichnet:

Symbol	Relevanz
	Wichtiger Erfolgsfaktor, der laut Umfrage/Studie den Erfolg der Projekte mehrheitlich stark gefördert hat.
	Erfolgsfaktor, der laut Umfrage/Studie den Erfolg der Projekte mehrheitlich gefördert hat.
	Praxistipp der Autoren, der unserer Erfahrung nach den Projekterfolg fördert..

Entscheiden Sie für sich selbst, welche Faktoren für Sie wichtig sind. Und achten Sie darauf, dass es wichtiger ist, die richtigen Dinge zu tun als die Dinge richtig zu tun!

Ihre Meinung ist uns wichtig

Unter www.bitplan.com/Erfolgsfaktoren/index.html finden Sie weiteres Hintergrundmaterial zu diesem Buch. Dort finden sie auch alle Details des Beispiels.

Ihre Meinung interessiert uns! Über die Web-Site können Sie Kontakt mit uns aufnehmen.

Danksagungen

Dieses Buch hat schon einige Jahre „gegärt“ bevor es geschrieben wurde. Viele der hier vorgestellten Ideen sind schon in der Zeit von 1996-1999 geboren worden, in der die Autoren noch für Martin Rösch tätig waren. Ihm danken wir für die Inspiration und das fundamentale Verständnis des Softwareengineering, das wir durch ihn gewonnen haben.

Den Teilnehmern an der Umfrage – insbesondere der ersten Version bei den Agility Days in Herrsching 2002 – danken wir für die spontane Bereitschaft, bei der Bewertung der Erfolgsfaktoren mitzuwirken.

Viele weitere Kunden, Lieferanten, Partner, Freunde und Verwandte haben an diesem Buch mitgewirkt und verdienen unseren Dank dafür.

Wolfgang Fahl (fahl@b-agile.de) !!!

Frank Hoffmann (hoffmann@b-agile.de) !!!

Literaturtipp

[Ber93] R. Berth: **Erfolg**, Econ, 1993

Das Buch heisst schlicht „Erfolg“. Es richtet sich an Manager und hat das vorliegende Buch inspiriert. Insbesondere gibt es eine Übersicht von Erfolgsfaktoren für die innovative Zukunftsbewältigung von Unternehmen zusammen mit einer durch Unternehmensdaten fundierten Bewertung. Erfolgsfaktor Nr. 1 für Unternehmen mit einer Wirkung von 347% gegenüber dem Durchschnitt ist demnach „Ergänzendes Aufeinanderzugehen“

Das Projektverwaltungs-Projekt

Die folgende fiktive Ausgangssituation wird in diesem Buch durchgängig als Beispielprojekt herangezogen:

Hr. Aufenberg ist Vorstand der Gründer Software AG. Dieses Software-Unternehmen wurde 1989 als GmbH gegründet und hat in seiner 13-jährigen Firmengeschichte mit kundenspezifischer Software für die unterschiedlichsten Branchen ein stürmisches Wachstum erreicht.

Die Gründermannschaft von fünf erfahrenen Softwareentwicklern ist inzwischen auf die stolze Größe von 178 Mitarbeitern gewachsen, davon sind 120 aktiv in der Softwareentwicklung im Kundenauftrag tätig. Vor einem Jahr erfolgte die Umwandlung in eine AG. Inzwischen wickelt das Unternehmen ca 20 Software-Projekte jährlich ab, die eine durchschnittliche Größe von sechs Person Jahren haben.

Das vor Jahren selbst im Hause (noch unter Beteiligung des Unternehmensgründers Dr. Gründer) entwickelte System PVW zur Projektverwaltung entspricht nicht mehr den Ansprüchen des Unternehmens, und Hr. Aufenberg beschließt daher selbst als Auftraggeber aufzutreten. Er möchte sich ein neues Projektverwaltungssystem entwickeln lassen.

Das vielversprechenste Angebot hat die PV-Soft GmbH gemacht. Fr. Probst ist dort die Projektleiterin. Das von ihr zusammengestellte Team besteht aus Hr. Anglet, Fr. Entrop und Hr. Teschner. Von den ursprünglichen Entwicklern des hauseigenen Projektverwaltungsystems der Gründer-Software AG ist nur noch Hr. Exner im Unternehmen. Er soll helfen, PV-Soft zu erklären, was für die neue Lösung unbedingt genauso gelöst werden muss wie im alten System.

Die Namen der Personen in diesem Beispiels sind so gewählt, dass die Rolle der Beteiligten durch die ersten Buchstaben des Nachnamens angedeutet wird:

- Hr. **A**ufenberg ist **A**uftraggeber
- Frau **P**robst ist **P**rojektleiterin
- Hr. **A**nglet kümmert sich um die **A**nforderungen
- Fr. **E**ntrop ist **E**ntwicklerin
- Hr. **T**eschner hat die Verantwortung für die **T**ests
- Hr. **E**xner hat das „**E**x“-System PVW mitentwickelt

„Wir alle haben aus der Welt nur zu sehr einen Computer gemacht und diese abermalige Erschaffung der Welt hat lange begonnen, bevor es elektronische Computer gab.“

Joseph Weizenbaum

1

Von Menschen und Computern

Wie können wir sicherstellen, dass die Computer dieser Welt das tun, was wir Menschen von ihnen wollen?

Wie kann ich sicherstellen, dass mein Computer tut, was ich von ihm will?

Alle Computer dieser Welt werden durch Software gesteuert. Das Verhalten von Computern lässt sich im Wesentlichen dadurch verändern, dass die zugehörige Software geändert wird. Der entscheidende Moment dafür ist das Softwareprojekt, in dem die Software erstellt bzw. gewartet wird. Im Softwareprojekt stellen die beteiligten Menschen die Weichen dafür, was der Computer tun kann und welche Möglichkeiten die späteren Anwender haben, um zu beeinflussen, was der Computer tun soll.

1.1 Menschen als Handelnde

Im Mittelpunkt aller Softwareprojekte stehen Menschen, die Software-Wünsche haben. Sie haben eine mehr oder weniger grobe Vorstellung vom neuen System. Sie sind die potentiellen Anforderungssteller. Ihre Vorstellungen müssen in die Software systematisch einfließen.

Ein Team von weiteren Menschen ist dafür verantwortlich, dass dies geschieht.

Wie Menschen den Erfolg beeinflussen

Wie schätzen Sie die Chancen Ihres Projektes ein, wenn "die richtigen Menschen" beteiligt wären? Wie sehr ist das Scheitern eines Projektes auf "das falsche Team" zurückzuführen?

Der Erfolg Ihres Projektes steht und fällt mit den beteiligten Menschen. Die sogenannten "weichen" Faktoren beeinflussen das Projekt weit mehr als die Bezeichnung "weich" andeutet.



Erfolgsfaktor:

Menschen bilden das Fundament des Erfolges

Wir stellen diesen Erfolgsfaktor als ersten und wichtigsten Faktor heraus. Die weiteren in diesem Buch genannten Faktoren können nur dann ihre Wirkung entfalten, wenn die beteiligten Menschen sich diesen ersten zu eigen machen und ihr gemeinsames Handeln danach ausrichten. Dazu ist viel Überzeugungsarbeit und eigene Überzeugung erforderlich.

In diesem Buch gehen wir bewusst auf die „weichen“ Themen der Softwareentwicklung vor allem im Rahmen weiterer konkreter Erfolgsfaktoren ein. Es ist unserer Ansicht nach im Zusammenhang mit Menschen unmöglich, allgemeine Ratschläge zu geben, die in jeder Situation helfen. Jede Begegnung von Menschen ist einzigartig.

Es hilft aber im Umgang mit Menschen grundsätzlich zu verstehen, wie Kommunikation „funktioniert“. Nach [Thu81] verläuft jede Verständigung auf vier Ebenen:

- Sachinhalt
- Beziehung
- Appell
- Selbstkundgabe

Der Auftraggeber Hr. Aufenberg und die Projektleiterin Fr. Probst sowie das vorgesehene Team Hr. Anglet, Fr. Entrop und Hr. Teschner kennen sich seit mehreren Jahren und haben schon mehrere Projekte in Kooperation für gemeinsame Kunden erfolgreich absolviert. Die „Chemie“ zwischen diesen Beteiligten stimmt. Hr. Exner betrachtet das Projekt jedoch skeptisch. Er hat Angst um seinen Arbeitsplatz und identifiziert sich vor allem mit der jetzigen Lösung, die er ja noch mit Dr. Gründer zusammen entwickelt hat. Er betrachtet den Versuch von Hr. Aufenberg, dieses System abzulösen als „Frevel“.

Wenn Hr. Exner also einen Satz spricht wie „Das PVW-System läuft doch noch prima!“, dann stecken darin mehrere Aussagen gleichzeitig:

- auf der Sachebene: „Das PVW-System ist im Einsatz und die Verfügbarkeit und Fehlerrate entspricht den Vorgaben“
- auf der Beziehungsebene kann etwas wie: „Ich mag den Hr. Aufenberg umso weniger, je mehr er versucht das PVW-System abzulösen.“

- auf der Appellebene könnte es „Lasst das PVW-System bestehen!“ bedeuten
- im Rahmen der Selbstkundgabe hört jemand vielleicht „Ich bin ein hervorragender Softwareexperte - das PVW-System ist von mir“.

Was beim Hörer einer Aussage ankommt entscheidet dieser viel mehr als derjenige, der die Aussage formuliert. Wichtiger noch, die Aussage in Textform spielt eine unbedeutende Rolle - der „Ton macht die Musik“. Gestik, Mimik und Haltung werden viel stärker wahrgenommen als der Inhalt einer Nachricht. Im Abschnitt 7.3 „*Vom Umgang mit Eisbergen*“ gehen wir noch näher darauf ein, wie 90% der Kommunikation von nicht direkt wahrnehmbaren Anteilen geprägt sind.

Die Art, wie wir mit anderen umgehen ist stark von einigen Grundstilen geprägt, die in [Thu89] geprägt sind. Unserer Erfahrungen und das Wertesystem in dem wir aufgewachsen sind prägt sehr stark, wie wir gegenüber Anderen auftreten. Manchmal kommt es bei der Begegnung von zwei Menschen mit gegensätzlichem Umgangsstil zu „Teufelskreisen“. Eine solche Situation entsteht zum Beispiel oft, wenn ein Mensch mit „Helfer“-Charakter auf jemanden trifft, der dazu neigt die Rolle des „Hilflosen“ einzunehmen. Die Abhängigkeit des Hilflosen vom Helfer verstärkt sich je mehr der Helfer versucht Unterstützung zu geben.

Alle diese Zusammenhänge spielen in der Softwareentwicklung deswegen eine so grosse Rolle, weil in den Anfängen der Informatik die Technik und Mathematik im Vordergrund stand. Es herrschte lange Jahre die Vorstellung, dass die Verbesserung der technischen Verfahren die Softwareentwicklung erheblich rationalisieren würde. In Wirklichkeit sind die Verbesserungen auf diesem Weg weit hinter den Erwartungen zurückgeblieben. Die Möglichkeiten, die in den Menschen stecken, wenn sie gemeinsam Software erstellen, sind noch lange nicht ausgeschöpft.

1.2 Die zwei Blickwinkel - Mensch und Maschine

Probleme, die mit dem Computer gelöst werden sollen, kann man von zwei Blickwinkeln aus betrachten:

- vom Menschen aus
- vom Computer aus

Woran liegt es, dass es so schwer ist, Probleme, die sich mit dem Computer lösen lassen, auch wirklich mit dem Computer zu lösen? Wie oft haben Sie schon erlebt, dass Computerlösungen, die Sie gekauft, in Auftrag gegeben oder erstellt haben, wirklich das tun, was Sie sich gewünscht haben oder was Ihnen beim Kauf versprochen wurde?

Eine der Ursachen für die Antworten, die Ihnen in den Sinn gekommen sind lautet: "Es ist schrecklich schwer, Ihr Problem zu lösen, wenn man es ausschliesslich oder vorwiegend aus Sicht des Computers betrachtet".



Erfolgsfaktor:

Betrachten Sie das zu lösende Probleme aus den zwei Blickwinkeln:

1. Vom Menschen aus

2. Vom Computer aus

Verbinden Sie dann die Sichten miteinander.

Normalerweise stossen bei der Aufgabe, ein Problem mit dem Computer zu lösen, zwei Parteien aufeinander:

■ Auftraggeber

Die eine Partei betrachtet das Problem aus der Sicht des Auftraggebers, Kunden bzw. Nutzers - das ist die vorwiegend menschliche Betrachtungsweise. Es geht hier um die "wirklichen Dinge des Lebens" - es kann um Grundbedürfnisse wie Wasser oder Brot oder alltägliche Güter wie Autos, Versicherungen oder Strom gehen. Manchmal sind Spezialgebiete betroffen, wie medizinische Geräte, Sprachwissenschaft oder Theologie. In unserem Beispiel ist die Gründer Software AG der Auftraggeber.

Der Auftraggeber weiss oder hofft, dass sein Problem aus der Alltagswelt, in der er sich bewegt, mit dem Computer lösen lässt. Er hat eine halbwegs klare Vorstellung davon, was sein Problem ist und wie die Lösung in seiner mehr oder weniger laienhaften Vorstellung ungefähr aussehen könnte. Er ist in der Lage, sein Problem zu beschreiben und zu sagen, wann er mit einer vorgeschlagenen oder wirklich erstellten Lösung zufrieden wäre. Neben dem sachlichen Problem, das zu lösen ist, "menschelt" es hier gehörig: Es sind Wünsche und Hoffnungen und Gefühle, Phantasien und Ängste im Spiel. Oft ist der "Auftraggeber" nicht eine einzelne Person sondern tritt als eine Gruppe von Personen auf, die sich erst einmal einigen müssen, was sie eigentlich gemeinsam wollen. Die menschlichen Faktoren vervielfachen sich nun, in vielen solchen Auftraggebersituation entsteht eine richtig "politische" Situation. Unterschiedliche Interessen treffen aufeinander und es müssen Entscheidungen getroffen, sowie Kompromisse ausgehandelt werden. Die obigen Aussagen bzgl. der "klaren Vorstellung, was das Problem ist, und wie die Lösung aussehen könnte" sind zunächst nicht sofort zutreffend, sondern es bedarf bereits einiger Anstrengung, um diesen Zustand zu erreichen.

■ Auftragnehmer

Die andere Partei betrachtet das Problem aus der Sicht eines Auftragnehmers, Lieferanten bzw. Lösungserstellers. Hier treten die Spezialisten aus dem Computerbereich auf - und wer wird ihnen verdenken, dass hier vorwiegend die Sichtweise aus dem Blickwinkel "Computer" anzutreffen ist. Hier geht es um die technische Seite des Problems und da hat die Computerei ja einiges an Kauderwelsch und Fachchinesisch zu bieten, um diese

Sichtweise auszudrücken. In unserem Beispiel ist die PV-Soft GmbH der Auftragnehmer. Der Auftragnehmer hat die Verantwortung dafür, das was der Auftraggeber verlangt, in eine funktionierende Computerlösung umzusetzen.

Hr. Aufenberg interessiert sich als Manager kaum für die technische Umsetzung des Problems. Ihm kommt es darauf an, den Überblick über die Projekte zu behalten und aus den Projektzahlen die Angaben für den Quartalsbericht zusammenstellen zu können, damit er seinen Aktionären Rede und Antwort stehen kann. Hr. Exner kennt noch jedes Bit der ursprünglich in Assembler und COBOL realisierten ersten Projektverwaltungslösung, die Hr. Gründer noch aus seiner Urzeit als Entwickler mitgebracht hatte. Die Techniker der Firma PV-Soft sind vor allem daran interessiert, die neuesten Webtechnologien bei diesem Projekt endlich ausprobieren zu können. Fr. Probst sieht es als Projektleiterin als ihre wichtigste Aufgabe an zwischen den Beteiligten zu vermitteln und einen Interessenausgleich zu schaffen.

Es ist eigentlich also ganz einfach: Der Auftraggeber sagt, was er verlangt und der Auftragnehmer sorgt dafür, dass der Computer das Verlangte leistet. Warum gelingt das nicht immer wie erhofft? Wie steigen die Chancen dafür, dass es gelingt?

In den folgenden Abschnitten finden Sie einige Antworten auf diese Fragen.

1.3 Die "Brücke" vom Wunsch zur Software

Im Vorwort haben wir Ihnen bereits die „Brücke“ als Sinnbild für die Softwareentwicklung vorgestellt. Die folgenden Abbildungen zeigen dieses Sinnbild mit einem typischen objektorientierten Vorgehen als Unterteilung der Schritte über die Brücke Abbildung 1.1 zeigt die grobe Einteilung in vier Abschnitte:

- Was - Was soll die Software leisten?
- Wie - Wie soll das Verlangte umgesetzt werden?
- Umsetzung
- Prüfen - Ist das Verlangte umgesetzt worden?

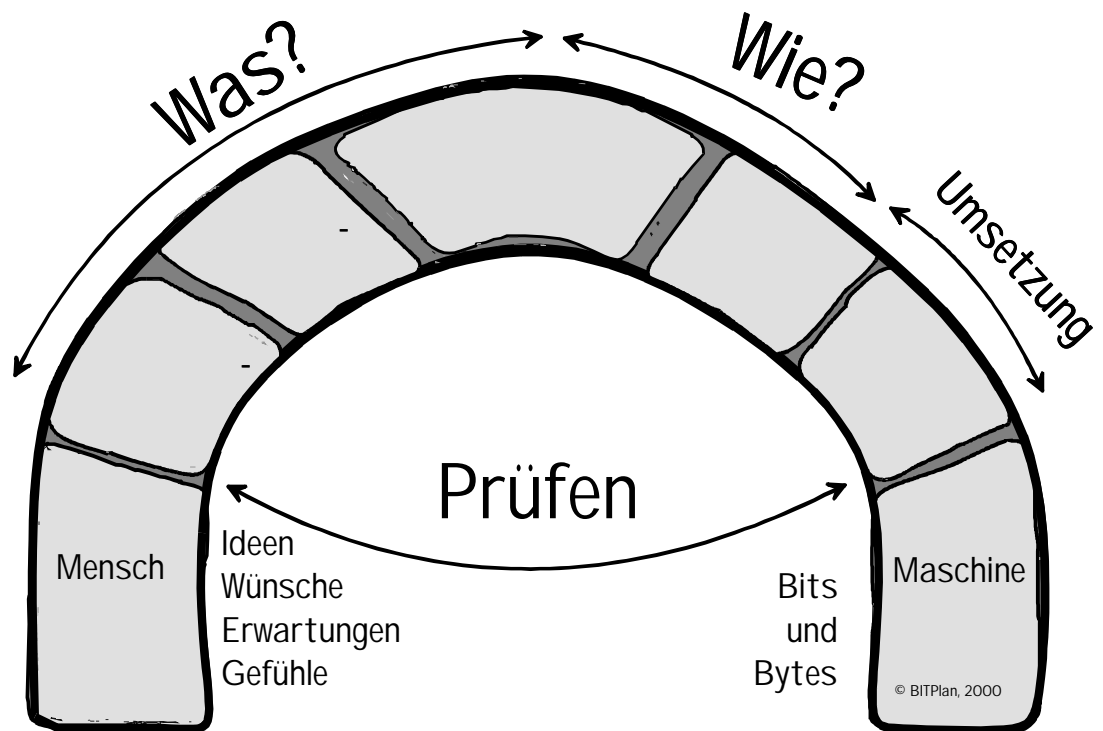


Abbildung 1.1: Die Brücke mit den wichtigsten Klärungsschritten

Ausgehend von den Ideen, Wünschen, Erwartungen und Gefühlen der beteiligten Menschen geht es darum, festzustellen:

■ Was - Was soll die Software leisten?

Klärung des konkreten Auftrags. Was will der Auftraggeber? Was soll die fertige Computerlösung leisten? Wie wird der Auftraggeber prüfen, dass das Verlangte umgesetzt wurde? Ausgehend von den Ideen, Wünschen, Erwartungen und Gefühlen der beteiligten Menschen dienen die Schritte Geschäftsprozessanalyse, Anforderungsaufnahme und objektorientierte Analyse (OOA) dazu Klarheit und Verbindlichkeit herzustellen. Aus Wünschen wird eine ausgehandelte, verbindliche Liste von Anforderungen.

Der Schlussstein der Brücke ist erreicht. Es herrscht Klarheit über den Auftrag. Das Softwareprojekt hat von nun an die Mission diesen Auftrag in funktionierende Bits und Bytes die auf den gewünschten Maschinen laufen umzusetzen. Als nächstes muss geklärt werden:

■ Wie - Wie soll das Verlangte umgesetzt werden?

Mit den Analyse- und Design-Mitteln Strukturieren, Konzipieren und Modellieren (siehe Kapitel 3) wird das Gesamtproblem in Einzelteile zerlegt und Schritt für Schritt der Problemlösung näher gebracht. Das Ergebnis ist eine vollständige Beschreibung der Problemlösung, die zur Umsetzung geeignet ist. In den Schritten objektorientierte Analyse und objektorientiertes Design wird die Umsetzung geplant und vorbereitet.

Die Klärung des „Was“ und „Wie“ ist Voraussetzung für die Umsetzung:

■ Umsetzung

Aus der Problemlösungsbeschreibung werden Computerprogramme und andere Lösungselemente hergeleitet. Die so entstehenden Lösungsteile werden zu einem Ganzen zusammengefügt. Der Schritt Implementierung leistet dies.

Die erstellte Software sollte nun mit den Vorstellungen des Auftraggebers übereinstimmen. Dies gilt es zu prüfen:

■ Prüfen - Ist das Verlangte umgesetzt worden?

Sicherstellen, dass die fertige Computerlösung dem entspricht, was die Software leisten soll. Innerhalb der (in der Abbildung nicht eingezeichneten) Schritte Abnahme, Test und Inbetriebnahme hat diese Prüfung zu erfolgen, damit sichergestellt ist, dass die an den Auftraggeber gelieferten Bits und Bytes, die von der Maschine ausgeführt werden, dem entsprechen, was Auftraggeber wollte.

Die einzelnen Schritte über die Brücke brauchen nicht strikt nacheinander zu erfolgen. Es ist möglich in Iterationen und Zyklen zu arbeiten, d.h. wie bei einer Springprozedur einen Schritte rückwärts zu gehen, bevor der nächste Schritt gemacht wird oder das Team aufzuteilen, so dass einige schon ein paar Schritte weiter vorne sind und jeweils Ergebnisse über die Brücke durchgereicht werden (siehe Abschnitt 4.2 „Kurze Entwicklungszyklen“).

Abbildung 1.2 zeigt die Brücke mit Details für ein objektorientiertes Vorgehen:

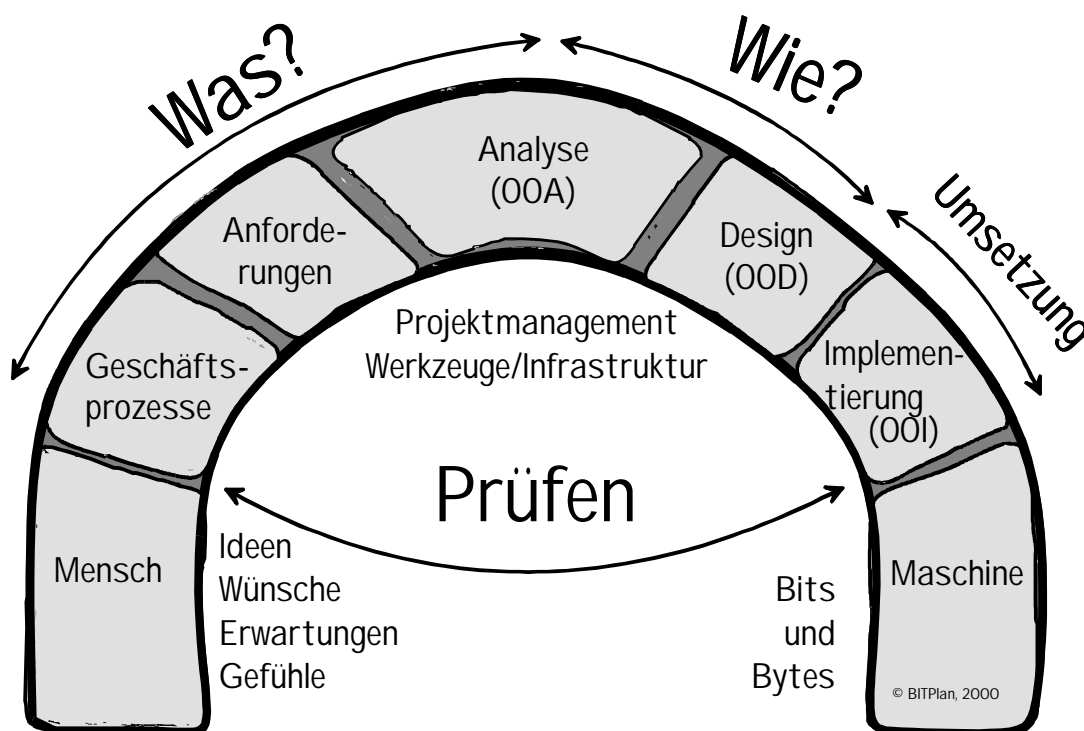


Abbildung 1.2: Die Brücke mit den Softwareprozessschritten für ein typisches objektorientiertes Vorgehen

Die Themen Projektmanagement sowie Werkzeuge und Infrastruktur spielen im gesamten Verlauf des Softwareprojektes eine Rolle und sind daher unterhalb der Brücke als begleitende Aufgaben dargestellt. Bitte lesen Sie dazu Kapitel 6 „*Infrastruktur und Werkzeuge*“, insbesondere wenn Sie interessiert, wie das Projektverwaltungsprojekt mit diesen Themen umgegangen ist.

Einstieg in ein neues Projekt

Wie können Sie schnell den inhaltlichen Einstieg in ein neues Projekt finden? Eine Möglichkeit ist es mit den relevanten Beteiligten des Auftraggebers jeweils ein Erstinterview zu führen. Jeder Beteiligte wird dabei einzeln für sich nach folgenden Regeln befragt:

- **Vorstellung**
Was ist die wahrgenommene betriebliche Rolle? Woran arbeitet der Beteiligte in seinem „normalen“ Arbeitsleben ausserhalb der Anforderungsaufnahme? Für die Vorstellung sind ca. 5 Minuten vorgesehen.
- **Ausgangslage**
Was ist die subjektive Sicht der Ausgangslage? Wie ist er persönlich Betroffenen? Welche Fakten kennt er? Welche Kompetenzen hat er? Welche Gefühle entstehen beim Gedanken an das was ist und kommen soll? Bis zu. 30 Minuten.
- **relevante Themen und Begriffe**
Nach der 7+/-2 Regel (siehe Abschnitt 3.4) werden die wichtigsten Dinge/Begriffe/Themen genannt, die im Zusammenhang mit dem geplanten Projekt eine Rolle spielen. Bis zu 10 Minuten.
- **Aktivitäten - Was soll mit den genannten Begriffen/Themen getan werden?**
- 3 bis 5 Zeilen zur Beschreibung je Aktivität - bis zu 45 Minuten.
- **Erste Anforderungen und konkrete Beispiele dazu - Woran wird erkannt, dass die Aktivitäten „richtig“ laufen? Welche Anforderungen müssen erfüllt sein? - Zeitrahmen je nach Verfügbarkeit.**

Bis auf Punkt Vorstellung und Ausgangslage werden alle Ergebnisse vollständig schriftlich erfasst. Zu Ausgangslage und Vorstellung werden nur die Punkte erfasst, die der Interviewte explizit schriftlich festgehalten haben möchte.

Fr. Probst führt zu Beginn des Projektes nacheinander zwei kurze Erstinterviews mit den Auftraggebertretern Hr. Aufenberg und Hr. Exner. durch Sie erklärt jeweils kurz die Regeln des Interviews:

Das Interview mit Hr. Exner verläuft so:

1. Vorstellung: Hr. Exner ist 54 Jahre - von der Ausbildung Hr. Industriekaufmann und seit Beginn bei der Fa. Gründer. Dr., Gründer und Hr. Exner hatten schon im Vorgängerunternehmen zusammen gearbeitet. Vor drei Jahren sollte Hr. Exner Leiter des Supportbereiches werden, aber eine mehrmonatige Erkrankung führte dazu, dass er die Gelegenheit nicht wahrnehmen konnte. Hr. Marx leitet diesen Bereich jetzt. Hr. Exner bekam nach seiner Rückkehr die Leitung

eines grossen Bankprojektes übertragen bekam. Dieses Projekt wird in Kürze beendet sein. Wie es nach diesem Projekt weitergeht weiss Hr. Exner noch nicht.

2. Ausgangslage: Hr. Exner beschreibt wie das bestehende System vor 13 Jahren entstanden ist und seitdem tadellos läuft. Die Mitarbeiter, die damals daran gearbeitet haben sind inzwischen bis auf Hr. Exner alle nicht mehr im Unternehmen. In den letzten drei Jahren hat Hr. Exner das System neben seinem Bankprojekt immer wieder unter hohem persönlichen Einsatz betreut und dabei weder Überstunden noch das Opfern von Freizeit gescheut. Hr. Exner bittet Fr. Probst dies nicht schriftlich festzuhalten und Bitte auch nicht den Umstand, dass er überhaupt nicht nachvollziehen kann, warum das schöne existierende System nun unbedingt von Hr. Aufenberg weggeschmissen werden soll. Fr. Probst möge das nicht persönlich nehmen.

3. relevante Themen und Begriffe:

Projekt, Mitarbeiter, Kunden, Zeiten, Verantwortlicher, Budget, Priorität, Projektstatus

4. Aktivitäten:

Projekt anlegen, Mitarbeiter erfassen, Kunden erfassen, Mitarbeiter Projekten zuordnen, Verantwortlichen festlegen, Zeiten erfassen, Restbudget ermitteln mit den entsprechenden Beschreibungen dazu.

5. Erste Anforderungen und Prüfbeispiele

Hier nennt Hr. Exner, dass es unbedingt möglich sein muss, das Zeitbudget zu überziehen. Als Beispiel nennt er ein Projekt mit 3000 Personentagen, bei dem tatsächlich 6500 Personentage angefallen sind. Das Verfügbare Restbudget war dann zum Schluss natürlich -3500 Personentage - aber das sei schon o.k. so. Für weitere Anforderungen reicht die Zeit von 90 Minuten, die für das Interview bereit stand dann leider nicht mehr.

In unserem Beispiel hat die Projektleiterin sich rechtzeitig darum gekümmert, die Weichen für den weiteren Projektverlauf zu stellen. Nach den beiden Interviews vermutet Sie, dass Hr. Exner um seinen Arbeitsplatz fürchtet und sie erfährt von Hr. Aufenberg, dass einer der Hauptgründe für die Ablösung des Altsystems darin besteht die Abhängigkeit von Hr. Exner zu beseitigen, die zudem von Hr. Aufenberg als Innovationshürde empfunden wird. Die vorliegende Mischung von menschlichen und inhaltlichen Problemen ist typisch für Softwareprojekte. Die meisten Softwareprojekte haben schliesslich zum Ziel Veränderungen in der Arbeitsweise der Beteiligten herbeizuführen. Frau Probst weiss durch Ihre Interviews sehr früh, wo Schwierigkeiten auf der fachlichen und menschlichen Seite auftauchen können. Durch die Fragen nach Begriffen, Aktivitäten und ersten Anforderungen hat sie sich rasch mit den wichtigsten *Geschäftsobjekten* und *Anwendungsfällen* vertraut machen können die ihr voraussichtlich im Rahmen der objektorientierten Analyse später begegnen werden ohne einen Begriff der Objektorientierung überhaupt in den Mund zu nehmen.

1.4 Welches Vorgehensmodell?

Welches Vorgehensmodell ist für mein Softwareprojekt das Beste? Welche Kriterien kann ich zur Entscheidung dafür heranziehen?

Zur Zeit streiten sich die Softwareexperten darüber, welche Vorgehensweise jeweils einer anderen überlegen ist. Folgende Punkte kristallisieren sich dabei heraus:

- Es gibt kein generell für alle Softwareprojekte gleichermassen geeignetes Vorgehensmodell
- Die richtige Balance zwischen den Extremen „chaotisches Vorgehen“ und „strikte Einhaltung eines vorgegebenen detaillierten Vorgehensmodells“ zu finden verspricht den größten Projekterfolg
- Auf der Suche nach neuen Vorgehensweisen sind in den letzten Jahren viele neue Begriffe für Vorgehensweisen eingeführt worden: extreme Programming, Personal Software Process, Agile Software Development, Rational Unified Process.



Erfolgsfaktor:

Einigen Sie sich auf ein für Ihr Projekt geeignetes Vorgehensmodell und wenden Sie dieses aus Überzeugung an.

Wichtig für die richtige Auswahl des Vorgehensmodells und der Elemente des Vorgehensmodells, die Sie einsetzen wollen ist es, den Zusammenhang zwischen den angewendeten Verfahren und Methoden und dem damit zu erreichenden Projekterfolg zu kennen. Dafür leistet dieses Buch Hilfestellung. Praktisch alle hier genannten Erfolgsfaktoren finden Sie unter gleichem oder anderem Namen in den einschlägigen Vorgehensweisen wieder. Prüfen Sie für sich selbst die Relevanz jedes Faktors anhand der Erfahrungswerte, die aus der Umfrage zu diesem Buch gewonnen wurden. Beachten Sie vor allem, dass die Einhaltung eines Vorgehensmodells an sich keinen Wert hat. Es kommt darauf an, dass alle Beteiligten von diesem Vorgehen überzeugt sind und es aus dieser Überzeugung heraus anwenden. Dazu gehört auch, dass die vorgegebenen Regeln bewusst gebrochen werden, dürfen, wenn es dem Projekt nützt¹.

Fr. Probst lädt zu Beginn des Projektes zu einer Anfangsbesprechung ein. Bei dieser Besprechung lernen sich alle Teammitglieder kennen und die Kernpunkte des Projektes werden besprochen. Hr. Aufenberg fragt, welches Vorgehensmodell angewendet werden wird. Fr. Probst erläutert, dass erst Aufgabe und Ziele geklärt werden und dann das Vorgehen. Sie bitte also die Teammitglieder bzgl. dieses Punktes noch um etwas Geduld.

Im Grundsatz gehört die Frage der Klärung des Vorgehensmodells für ein Softwareprojekt an die dritte Stelle der Klärung von Aufgabe, Zielen und Vor-

¹ Hewlett-Packard war Vorreiter auf diesem Gebiet, indem es in den Unternehmensgrundsätzen festlegte, dass jeder Mitarbeiter Regeln bewusst brechen darf, wenn er davon überzeugt ist, dass dies dem Unternehmen nützt.

gehen. Das Vorgehensmodell sollte also erst ausgewählt werden, wenn Aufgabe und Ziele des Projektes klar sind (siehe nächster Abschnitt 1.5 „Gemeinsames Erfolgsverständnis“). Bei der Auswahl des Vorgehensmodells hilft es sich die Grundsätze zu überlegen, die Sie an Ihr Vorgehen anlegen wollen. Wir vertreten zum Beispiel folgende Vorgehensgrundsätze, die sich in unseren Projekten bewährt haben:

- Modellgestützte Anwendungsentwicklung
- Hohe Wartbarkeit durch Nachverfolgbarkeit von der Anforderung bis zur Implementierung
- Frühzeitige Absicherung der Berücksichtigung der fachlichen Anforderungen schon in der Analysephase
- Iteratives Vorgehen in überschaubaren Zyklen
- Saubere Trennung zwischen Fachlichkeit und Technik

1.5 Gemeinsames Erfolgsverständnis

Wie kann in einem Projekt zügig ein gemeinsames Erfolgsverständnis herbeigeführt werden?

Das folgende schrittweise Vorgehen zur Klärung von Aufgabe, Ziele und Vorgehen hat sich für die Kooperation in Teams bewährt. Nacheinander werden dabei geklärt:

- Aufgabenverständnis
Worum geht es?
- Ziele
Was soll erreicht werden?
Bis wann soll es erreicht werden?
In welchem Umfang soll es erreicht werden?
- Vorgehen
Auf welchem Wege sollen die Ziele erreicht werden?
Welche Schritte sind erforderlich ?
In welcher Reihenfolge sollen die Schritte bearbeitet werden?
Wer macht was bis wann?

Diese Zerlegung ist gleichermaßen für das Gesamtprojekt wie auch für Teilaufgaben geeignet. In Verlaufe eines Projektes lohnt es sich also immer wieder, Aufgabenverständnis, Ziele und Vorgehen auf den verschiedensten Detaillierungsniveaus zu klären. Wichtig ist dabei, dass die Schritte konsequent nacheinander durchgeführt werden und unter den Beteiligten nach jedem Schritt eine Einigung herbeigeführt wird. Nach dem Schritt "Aufgabenverständnis" müssen also alle Teammitglieder bereit sein die Aussage "Ja - das ist unser gemeinsames Aufgabenverständnis!" zu unterschreiben. Erst dann geht

es mit der Klärung der Ziele weiter - bis hin zum dem Punkt "Ja - das sind unsere gemeinsamen Ziele!". Erst zum Schluß wird dann das Vorgehen abgestimmt bis auch darin Einigkeit erzielt wird.



Erfolgsfaktor:

Klären Sie zu Beginn des Projektes und bei Bedarf im weiteren Verlauf Aufgabenverständnis, Ziele und Vorgehen.

Die Trennung in diese drei Schritte bringt den großen Vorteil mit sich, dass die Bandbreite der zu untersuchenden Möglichkeiten viel kleiner ist. Es müssen nur noch die Ziele betrachtet werden, die zum Aufgabenverständnis passen und später nur die Vorgehensweisen diskutiert werden, die zu den Zielen führen, auf die sich das Team geeinigt hat. Zusatzaufwand, der durch Missverständnisse entsteht und langwierige Diskussionen aufgrund von falschen Annahmen lassen sich so vermeiden.

Fr. Probst bitte alle Teammitglieder, ihr Aufgabenverständnis in einem Satz auf einer Metaplankarte zu formulieren und die Karten werden dann an eine Pinwand geheftet. Die Karten lauten:

Hr. Anglet: „Gründer möchte ein webbasiertes Projektplanungssystem haben“

Hr. Aufenberg: „PVW wird so abgelöst, dass die Projektdaten direkt in die Unternehmensquartalsberichte einfließen können“.

Hr. Teschner: „Hr. Aufenberg möchtel ein Zeiterfassungssystem für die Mitarbeiter der Softwareprojekte“

Hr. Exner „PVW soll um eine Internetkomponente ergänzt werden“

Fr.. Probst: „Auf Basis der Funktionalität von PVW soll ein neues System entstehen, das Webfähig ist und mit dem System zum Erstellen der Quartalsberichte zusammenarbeitet“.

Fr. Entrop: „Gründer hat Interesse die Vorgehensweise von PV-Soft näher kennenzulernen und daher die Projektverfolgung als Pilotprojekt ausgewählt“

Gemeinsames Verständnis der Projektziele erreichen

Wie häufig sind Ihnen schon Projekte begegnet, bei denen die Beteiligten unterschiedliche Ziele verfolgt haben?

Die wichtigste Voraussetzung für den Erfolg eines Softwareprojektes ist, dass die Beteiligten von Beginn an Klarheit und Einigkeit darüber haben was sie als Projekterfolg ansehen. Ein Microsoftteam, das eine neue Word für Windows Version erstellt hat mit Sicherheit ein anderes Erfolgsverständnis als ein Team der European Space Agency, das Software für die nächste Ariane-Mission erstellt. Bei Microsoft wird viel Wert darauf gelegt, dass die Software schnell fertig wird und ausgeliefert werden kann. Ein Betatest mit einigen tausend Anwender dient dazu, die wichtigsten Fehler noch vor der Auslieferung zu beheben. Der Rest der Fehler wird später oder gar nicht beseitigt. Ausfälle der Software dürfen bei Microsoft vorkommen, solange nur wenige Anwender

betroffen sind. Das Arianeteam hat im Fokus eine Software zu erstellen, die auf Anhieb die für die Weltraummission entscheidenden Funktionen zuverlässig bereitstellt. Die Software darf an entscheidenden Stellen überhaupt nicht ausfallen.

In unserem Beispielprojekt haben die wenigen Beteiligten bereits relativ unterschiedliche Vorstellungen davon, was die Aufgabe des Projektes ist.

Die Erfolgchancen eines Softwareprojektes sinken dramatisch, wenn unter den Beteiligten unterschiedliche Vorstellungen konkurrieren, was als Erfolg des Projektes anzusehen ist. Die Erfolgskriterien bestimmen schliesslich die Ziele des Projektes. Klassische Konkurrenten für Projektziele sind z.B.:

- Termin versus Kosten versus Qualität
 - Der Auftraggeber möchte das Ergebnis möglichst schnell (Termin)
 - Der Auftragnehmer möchte ein möglichst hohen Gewinn erzielen (Kosten)
 - Der Endanwender möchte, dass die Software seine Bedürfnisse erfüllt (Qualität)
- Auftrag versus Produkt
 - Der Auftraggeber möchte gerne eine kundenspezifische Lösung
 - Der Auftragnehmer möchte die Lösung als Produkt vermarkten
- Form versus Funktion
 - Die Software soll hübsch aussehen und angenehm in der Handhabung sein (Form)
 - Die Software soll die geforderten Funktionen in allen Details realisieren

Je besser es den Beteiligten gelingt, so früh wie möglich einen Interessenausgleich bzgl. konkurrierender Zielvorstellungen zu erreichen, um so zufriedenstellender wird das Ergebnis werden.

Nach einiger Diskussion einigen sich die Teammitglieder auf folgende Aufgabenformulierung:

“Auf Basis der Funktionalität des bestehenden Projektverfolgungsystems PVW soll für Gründer Software AG eine neu zu erstellende Lösung entstehen, die webbasiert die Erfassung von Projekten und der von den Mitarbeitern dafür geleisteten Zeiten ermöglicht damit daraus für die Quartalsberichte die Auslastungs- und Budgetsituation sowie der Status der Projekte abgeleitet werden kann.“

Erst nachdem alle Beteiligten zugestimmt haben, dass dies das gemeinsame Aufgabenerständnis der Beteiligten ist, werden Ziele und Vorgehen geklärt. Im Verlauf der Diskussion um die Aufgabe werden die Ideen zu Zielen und Vorgehen bereits gesammelt. Die Einigung darüber findet jedoch erst danach statt.

Die Liste der Stichpunkte aus der Aufgabendiskussion umfasst: “automatische Erfassung am Arbeitsplatz-PC, Java, maximal 3 Monate Projektdauer, Enterprise Java Beans, JUnit, extremeProgramming, DB2-Datenbank

muss bleiben, Hr. Exner ist nur 1 Tag die Woche verfügbar,“

Auf folgende Ziele einigen sich die Anwesenden noch im Laufe der Sitzung:
- maximal 5 Monate Projektdauer, eine Projektdauer von 4 Monaten wird angestrebt.

- es soll das „moderne“ Entwicklungsvorgehen von PV-Soft ausprobiert werden- dabei soll das möglichst frühe Testen der Einhaltung der Wünsche des Auftraggebers eine grosse Rolle spielen- zwischen der Nennung eines Wunsches, der Implementierung und dem Test soll nicht mehr als drei Wochen Zeit vergehen. Dies bedeutet, dass die Wünsche schrittweise aufgenommen werden.
- es soll mit Java auf Basis einfacher Applets entwickelt werden - die Anwendung soll sowohl mit Netscape ab Version 4 als auch mit Internet-Explorer ab Version 5 laufen.

- die DB2-Anbindung entfällt zugunsten einer einfachen XML-Datenhaltung - die Reaktionszeit beim Start der Anwendung darf sich dafür auf bis zu 10 Sekunden verschlechtern, bei Abfragen auf bis zu 3 Sekunden.

- Hr. Exner steht jeweils Montags den ganzen Tag für das Projekt zur Verfügung



Erfolgsfaktor:

Definieren Sie die Projektziele klar anhand von messbaren Kriterien und festgelegten Werten.

Bei der Festlegung von Zielen kommt es darauf an, dass diese in nachprüfbarer Weise festgelegt werden. Sowohl der Inhalt muss klar bestimmt sein, als auch die Werte für die Kriterien anhand derer geprüft werden kann wie gut das jeweilige Ziel später erreicht wird.

Abgestimmtes Vorgehen

Wann ist der richtige Zeitpunkt das Vorgehen festzulegen? Welche Freiheitsgrade bestehen bezüglich der Einhaltung des Vorgehens? Wer bestimmt, wie vorgegangen wird?

Es ist nicht sinnvoll einem Softwareprojekt ein Vorgehen von aussen aufzuzwingen. Die Beteiligten müssen selbst vom Vorgehen überzeugt sein. Der beste Weg dazu, ist das Vorgehen gemeinsam im Team abzustimmen, nachdem Aufgabe und Ziel des Projektes klar sind. Erst dann kann festgestellt werden, was ein angemessenes Vorgehen im Verhältnis zu dieser konkreten Aufgabe und den konkret zu erreichenden Zielen ist. Zunächst sollten die Beteiligten eine Vorstellung entwickeln, wie das Vorgehen aussehen wird. Dabei muss noch nicht jedes Detail klar sein, lediglich die Linie muss soweit stimmen, dass deutlich zu erkennen ist, dass alle Probleme und Hindernisse die aus der Startsituation heraus bereits erkennbar sind auf dem vorgeschlagenen Weg ausgeräumt oder umgegangen werden können. Im Verlaufe des Projektes werden sich je nach Typ des Projektes Aufgabe, Ziele und Vorgehen unterschiedlich stark ändern. Versuchen Sie sich vorzustellen, wie gross die auf Sie zu-

kommenden Änderungen sein werden. Jetzt ist der Zeitpunkt das Vorgehen abzustimmen, damit für die ersten Schritte klar ist, wer wann was wie macht.

Am nächsten Montag trifft sich das Projektteam bei PV-Soft. Frau Probst zeigt anhand eines Projektes, dass PV-Soft vor einigen Monaten bei einem anderen Unternehmen erfolgreich abgewickelt hat das bei PV-Soft erprobte Vorgehen auf. Sie erläutert, dass der Kern des Vorgehens darin besteht, die Wünsche systematisch so zu erfassen, dass alle Beteiligten jeweils Einblick haben. Die Wünsche werden dann priorisiert und dort wo entschieden wird, dass die Umsetzung erfolgen soll müssen Prüfbeispiele gesammelt werden. Meistens werden diese jedoch schon bei der Nennung des Wunsches erfasst, um bessere Klarheit über den Wunsch zu bekommen. Frau Probst nennt ein Beispiel aus dem Vorprojekt, wo es um eine Bankanwendung ging:

Wunsch: Es sollen die fälligen Zinsen berechnet werden. Dafür ist tagesgenaue Zinsrechnung anzuwenden. Die Zinsformel lautet also: $Zinsen = Restschuld * Zinssatz / 100 * Zinstage / Anzahl\ Tage\ im\ Jahr$. Die Genauigkeit der Berechnung beträgt 5 Nachkommastellen.

Ausgangssituation des Prüfbeispiels: Die Restschuld eines Darlehens beträgt 10.000 Währungseinheiten. Der Zinssatz beträgt 6% per anno. Das Darlehen ist für 200 Tage im Jahr 2002 zu verzinsen.

Im Beispiel soll als Aktion die Zinsrechnung erfolgen.

In diesem Prüfbeispiel wird erwartet, dass die Zinsen als 328,76712 Währungseinheiten berechnet werden.

Die Textbeschreibungen der Anforderungen und Prüfbeispiele werden in einer webbasierten Datenbank gehalten. Aus den Textbeschreibungen lassen sich Rahmen für JUnit-Tests generieren. Während aus den Anforderungen ein Modell abgeleitet wird und daraus Java-Code entsteht wird parallel dazu oder oft auch schon vorher pro Prüfbeispiel ein Testprogramm erstellt. Frau Probst erläutert die Zusammenhänge anhand der Abbildung 1.3.

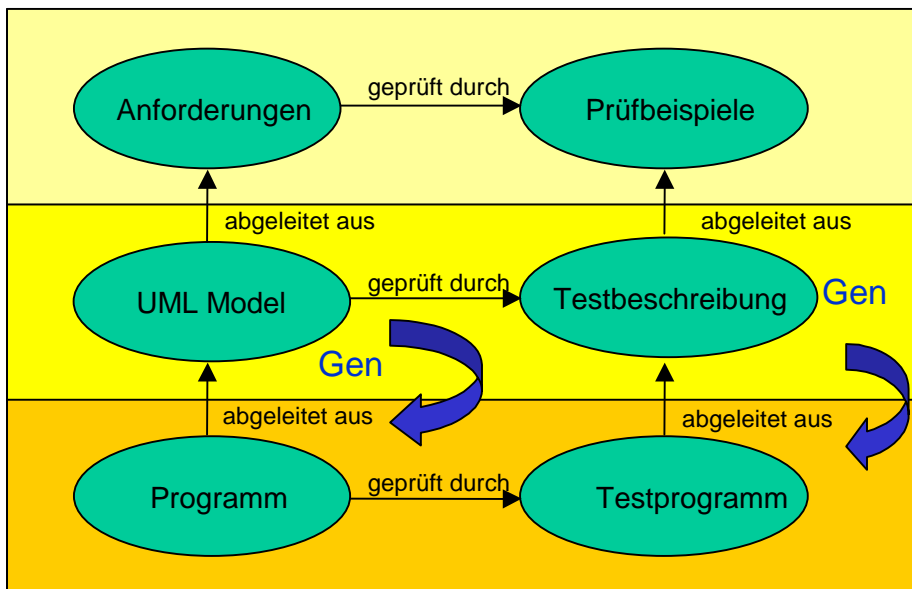


Abbildung 1.3: Das von PV-Soft verwendete Vorgehensmodell

Die Rückfragen, die im Rahmen dieses Vorgehens entstehend werden – soweit sie einigermaßen spannend sind – dokumentiert. Dies ist insbesondere nützlich,

wenig nicht gleich eine Antwort erhältlich ist und zunächst mit einer Annahme darüber, wie die Antwort ausfallen wird, weitergearbeitet werden muss. Bis auf Hr. Exner und Hr. Teschner kennen alle Teammitglieder dieses Vorgehen bereits und sind davon begeistert. Fr. Entrop berichtet, wie skeptisch sie selbst am Anfang des letzten Projektes war und wie sie nun selbst Feuer und Flamme dafür ist. „Es war einfach toll, wie gut alle Beteiligten auf dem Laufenden waren. Man konnte ruhig mal eine Woche in Urlaub sein - nach der Rückkehr reichte es sich im System Einblick über die neuen Themen zu verschaffen und dann die einzelnen Autoren jeweils anzusprechen, bei der nächsten Teamsitzung war man dann ohne grosse Rückfragen wieder voll im Boot!“

Hr. Exner fragt, wie teuer denn das Projekt bei all dem Aufwand geworden ist. Das müsste doch mindestens dreimal soviel Aufwand gewesen sein wie sonst. Fr. Probst beruhigt ihn und erläutert, dass es tatsächlich in den ersten Projekten mit diesem Verfahren Probleme gegeben hat, den Aufwand im Griff zu halten. Damals wurde von Anfang an zu jeder Anforderung sofort sehr viele Prüfbeispiele aufgenommen. Inzwischen haben die Teammitglieder ein Gefühl dafür entwickelt, zwischen „Wunsch“ und „Anforderung“ zu unterscheiden. Zum Schluss der Sitzung einigen sich alle darauf, dass Vorgehen anzuwenden. Hr. Exner erklärt jedoch, dass er gerne nach einem Monat auf Basis der bis dahin vorliegenden Erfahrungen noch einmal mit Hr. Aufenberg sprechen möchte, ob dieser an diesem Verfahren festhalten will.

1.6 Die Sprache des Auftraggebers sprechen

Wie können Sie sicherstellen, dass Sie als Auftragnehmer Ihren Auftraggeber verstanden haben? Woher wollen Sie wissen was die Spezialbegriffe und Ausdrucksweisen bedeuten, die Sie im Rahmen des Projektes zum ersten Mal kennenlernen?

Wenn Ihnen klar wird, wie wichtig es ist Missverständnisse, die bereits in den sprachlichen Unterschieden zwischen Auftraggeber und Auftragnehmer liegen zu vermeiden, dann ist der erste Schritt getan. Sie werden sich bemühen, die Sprache des Auftraggebers zu sprechen.



Erfolgsfaktor:

Sprechen Sie als Auftragnehmer die Sprache Ihres Auftraggebers!

Seien Sie sich bewusst, dass Ihr Auftraggeber Begriffe möglicherweise anders benutzt, als Sie selbst es gewohnt sind. Auch wenn der Auftraggeber es zunächst möglicherweise als lästig empfinden wird: Fragen Sie nach! Sammeln Sie die Bedeutung von unklaren Begriffen am Besten in einem systematischen Glossar, das für alle Projektbeteiligten zugreifbar ist.

Bereits nach den ersten Interviews hat Fr. Probst begonnen ein Glossar aufzustellen. Sie hat die Begriffe aus den Interviews in das Glossar eingetragen und zunächst Ihr eigenes Verständnis der Begriffe eingetragen:

Begriff	Bedeutung
---------	-----------

Begriff	Bedeutung
Projekt	Mit Ziel, Zeit- und Geldbudget, Mitarbeitern und Verantwortlichem ausgestattete benannte Aktivitätenfolge.
Mitarbeiter	Potentieller Projektbeteiligter / Angestellter der Gründer AG
Kunde	Auftraggeber eines Projektes, derjenige der durch das Projekt einen Nutzen erzielen soll
Zeiten	Arbeitszeit der Mitarbeiter die dem Projekt zugute kommt bzw. dem Projekt als Kosten zugerechnet wird.
Verantwortlicher	Pro Projekt übernimmt ein Mitarbeiter die Verantwortung für das Projekt. Er zum einen Ansprechpartner für Kunden und Geschäftsführung, zum anderen für die Mitarbeiter des Projektes.
Priorität	Die Priorität eines Projektes wird nach Dringlichkeit (dringend, weniger dringend, nicht dringend) und Wichtigkeit (wichtig, weniger wichtig, nicht wichtig) bemessen.
Projektstatus	Bearbeitungszustand des Projektes: Ein Projekt kann in Vorbereitung, aktiv oder beendet sein.

Beispieldokument 1.1: Entwurf des Glossars für das Projektverwaltungsprojekt

Das Projektteam wird das Glossar im Rahmen des Projekts ergänzen und Pflegen, insbesondere in Situationen in denen deutlich wird, dass es Verständnisunterschiede für Begriffe gibt.

1.7 Wie wird der Auftraggeber glücklich?

Wie glücklich macht Softwarequalität Ihren Auftraggeber?

1994 schrieb Martin Rösch einen Artikel mit der Überschrift „ISO9000: Qualität oder Zertifikat“ [Roe94b]. Ihm ging es darum den Unterschied zwischen ISO-9000 Zertifizierung und wirklicher Qualität in der Softwareentwicklung aufzuzeigen. Qualität ist die Übereinstimmung zwischen dem was Ihnen geliefert wurde mit dem was Ihnen versprochen wurde. Zufriedenheit ist die Übereinstimmung zwischen dem was Sie erhalten haben und dem was Sie sich gewünscht haben. Das ist häufig genug ein Unterschied – denn nicht immer wird Ihnen das versprochen, was Sie eigentlich haben wollen. Die folgende Version der Brücke veranschaulicht den Unterschied:

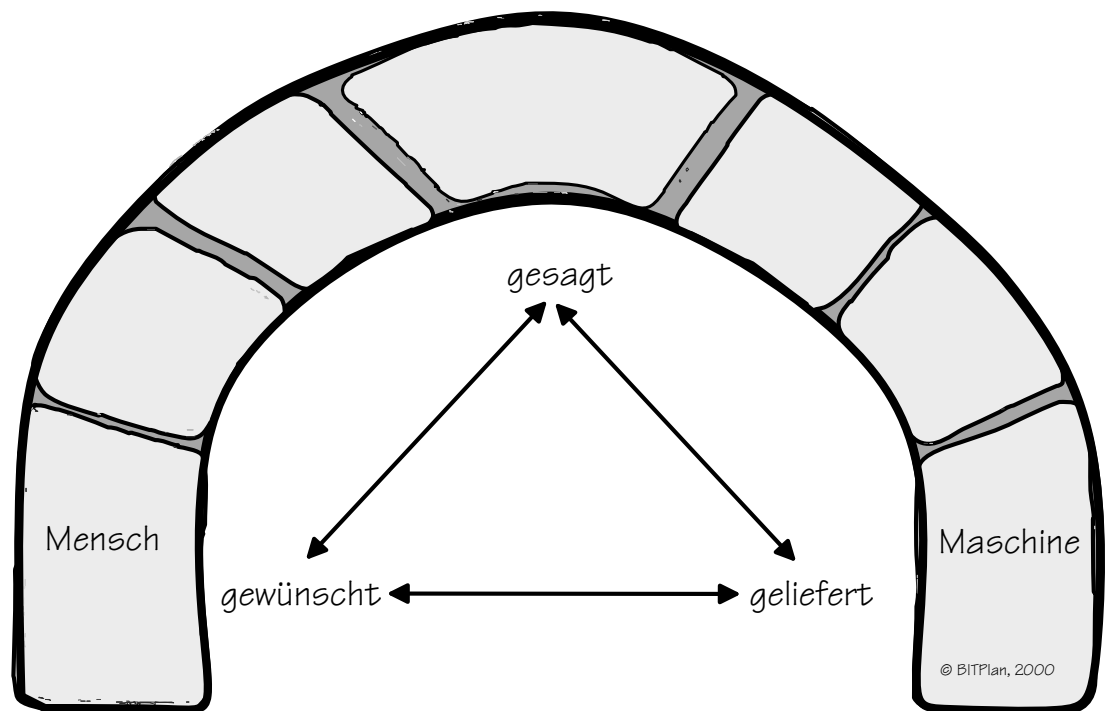


Abbildung 1.4 Woran macht der Auftraggeber seine Zufriedenheit mit der Software fest?

Im linken Teil der Brücke spielt der Auftraggeber die Hauptrolle. Seine Aussagen führen zu einem möglichst gut abgestimmten Bild, was die Software leisten soll. Nur die schriftlich hinterlegten oder unter Zeugen gemachten Aussagen sind ab jetzt rechtsverbindliche Grundlage der Umsetzung. Sollte es zu einem Streit vor Gericht kommen, spielt lediglich die Übereinstimmung zwischen Lieferung und dieser „gesagt“ vorliegenden Spezifikation eine Rolle. Die Qualitätsdefinition nach ISO 9000 geht von dem gleichen Prinzip aus. Ihre Organisation soll systematisch und nachprüfbar in der Lage sein die Übereinstimmung zwischen „geliefert“ und „gesagt“ herzustellen. Was nutzt es Ihnen aber, wenn Sie als Softwarelieferant zwar Gerichtsprozesse gewinnen können und ein ISO-9000 Zertifikat regelmässig bestätigt bekommen, Ihr Kunde aber nicht das geliefert bekommt, was er sich gewünscht hat?

Im linken Teil der Brücke spielt die Musik. Software nach Pflichtenheft abliefern können heute viele Unternehmen auf der ganzen Welt. Der Konkurrenzdruck durch Billiganbieter aus fernen Ländern, die über Internet prinzipiell spielend leicht mit potentiellen Auftraggebern kommunizieren können wird immer größer. Die Chancen liegen im Verbesserungspotential das in der Übereinstimmung von „gesagt“ und „gewünscht“ liegt. Denn wenn das Gesagte schon mit dem Gewünschten nicht übereinstimmt oder wenn die Wünsche gar nicht geäußert wurden (z.b. mangels Gelegenheit – niemand hat gefragt) - dann ist sicher, dass auch das Gelieferte nicht mit dem Gewünschten übereinstimmt.

Seien Sie sich bewusst, dass sie auf die Wünsche Ihres Auftraggebers Einfluss nehmen! Gerade wenn Sie sich gegenüber Mitbewerbern durchsetzen müssen, um ein Softwareprojekt für einen Auftraggeber durchführen zu können ist die Versuchung gross, Erwartungen zu wecken. Indem Sie Termine, Preise oder Inhalt versprechen beeinflussen Sie das gesamte Spektrum der Wünsche Ihres Auftraggebers. Der Zauberlehrling von Goethe „*Die ich rief, die Geisterwerd' ich nun nicht los.*“ mag es Ihnen oder Ihrem Team anschliessend durch den Kopf gehen. Tipps zu diesem Thema finden Sie im Abschnitt „Erwartungsmanagement und Verbindlichkeit“.

1.8 Wie sicher kann ich sein, dass die Software richtig ist?

Wie können Sie feststellen, dass Ihre Softwarelieferung mit dem Softwarewunsch Ihres Auftraggebers übereinstimmt? Im →Kapitel 2 Was soll die Software leisten? werden Sie einige hoffnungsvolle Antworten auf diese Frage finden. An dieser Stelle möchten wir jedoch eine kurze warnende Betrachtung einschieben. In den folgenden Kapiteln werden wir argumentieren, dass es möglich ist mit Computerunterstützung zu prüfen, ob die Wünsche des Auftraggebers erfüllt worden sind. Zumindestens die Projektleiterin aus unserem Beispiel, Frau Probst wird in diese Richtung agieren und damit passable Erfolge erringen. Grundlage dafür ist, dass wir nicht nur eine Software erstellen, die das umsetzt, was der Auftraggeber sich gewünscht hat. Entscheiden ist, dass wir auch Testsoftware erstellen, die Prüfbeispiel für Prüfbeispiel ermittelt, ob die Vorstellungen des Auftraggebers eingehalten wurden. Es gibt jedoch eine wichtige Grenze für diesen Ansatz. Diese Grenze liegt in der Wahrnehmung der Ergebnisse, die ein Computer liefert. Die Menschen, die später Nutzer unserer Software sind haben eine ggf. völlig andere Wahrnehmung der Ergebnisse, die unsere Software liefert, als unsere Prüf- und Testprogramme. Nehmen wir als klassisches Beispiel das Programm „Hello World!“. Eine Java-Version dieses Programm lautet:

```
public class Hello {  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

Ein Testprogramm dazu würde nun prüfen, dass tatsächlich die Zeichenfolge „Hello World!“ ausgegeben wird. Die Wahrnehmung eines Computers von Hello World! ist jedoch nur ein Bitmuster. Das Testprogramm testet die Zeichenkette in dieser Form. Zur Anzeige kommt die Zeichenkette dann jedoch ganz anders. Es kommt stark darauf an, auf welchem Computer und auf welchem Art die Anzeige erfolgt. Hier zwei Beispiele:



Abbildung 1.5: Ausschnitt aus der Anzeige von „Hello World!“ auf einem PC



Abbildung 1.6: „Hello World!“ auf einem Palm Taschencomputer

Auch wenn inhaltlich „Hello World!“ gleich „Hello World!“ ist, macht die Darstellung des Ergebnisses einen grossen Unterschied aus. Die Bandbreite ist dabei enorm. „Hello World!“ kann von einem Computer vorgelesen werden, als E-Mail verschickt, im Internet angezeigt oder als haushohe Reklame am New Yorker Time Square angezeigt werden. Die zugrunde liegende Digitaldarstellung als Bitmuster ist jeweils eine völlig Andere. Dieser letzte Teil der Übereinstimmung von „geliefert“ und „gewünscht“ spielt in der heutigen multimedialen Computerwelt eine immer größere Rolle. Wir menschen nehmen nicht die Bits des Computers wahr sondern nur das was unsere Sinnesorgane schliesslich als Umsetzung der Bits in wahrnehmbare Signale geboten bekommen.

1.9 Literatur

- [Bal96] H. Balzert: **Lehrbuch der Software-Technik**, Spektrum Akademischer Verlag, 1996
Lehrbuch mit umfangreicher Abhandlung aller Themen der Softwareentwicklung.
- [Bun02] C. Bunse, A. v. Knethen: **Vorgehensmodelle kompakt**, Spektrum Akademischer Verlag, 2002
Eine kurze und bündige Übersicht zu den verschiedenen Vorgehensmodellen, die bis heute für Softwareprojekte entwickelt worden sind.
- [Con01] L. L. Constantine: **The Peopleware Papers**, Prentice-Hall, 2001
Eine über Jahre gewachsene Sammlung von Aufsätzen über Menschen in Softwareprojekten.
- [DeM97] T. DeMarco: **Warum ist Software so teuer?**, Hanser, 1997
Warum ist Software eigentlich so teuer? Jerry Weinberg's Antwort „Im Vergleich zu was?“ ist die Basis für die Thesen dieses Buches.
- [DeM98] T. DeMarco: **Der Termin**, Hanser, 1998
Ein Roman: Ein Softwareprojektleiter wird entführt und findet sich in einer idealen Experimentierumgebung für Projekte wieder: es gibt reichlich Leute und Wissen. Die Zeit ist trotzdem knapp ...
- [DeM99] T. DeMarco, T. Lister: **Wien wartet auf dich**, Hanser, 1999
Der Originaltitel „Peopleware“ bringt den Inhalt dieses Buches auf den Punkt - sehr empfehlenswert!
- [DeM01] T. DeMarco: **Spielräume**, Hanser, 1999
DeMarco räumt mit dem Mythos auf, dass Erhöhung von Druck und das Entfernen von „Überflüssigem“ Projekte und Unternehmen erfolgreicher macht.
- [HUM95] W. S. Humphrey: **A Discipline for Software Engineering**, Addison-Wesley, 1995
Detaillierte Beschreibung eines systematischen und disziplinierten Ansatzes zur Softwareentwicklung, die sich an den einzelnen Softwareentwickler wendet. Durch beobachten der eigenen Fehlerquellen werden diese systematisch ausgemerzt.
- [Kru01] P. Kruchten: **Der Rational Unified Process - Eine Einführung**, Addison-Wesley, 2001
Eine Sammlung von Best Practices für Softwareprojekte, die von Rational unter dem Namen Rational Unified Process vermarktet wird.

- [Spre96] R. K. Sprenger: **Mythos Motivation - Wege aus einer Sackgasse**, Campus, 1996
Laut Sprenger gilt „Alles Motivieren ist Demotivieren“. Die Motivation des Menschen liegt in ihm selbst. Sie kann geweckt, aber nicht erzwungen werden.
- [Thu81] F. Schulz von Thun: **Miteinander Reden 1 - Störungen und Klärungen**, Rowohlt, 1981
Grundprinzipien der Kommunikation von einem Psychologen für Laien verständlich erklärt.
- [Thu89] F. Schulz von Thun: **Miteinander Reden 2 - Stile, Werte und Persönlichkeitsentwicklung**, Rowohlt, 1989
Auch ohne [Thu81] zu lesen äußerst aufschlußreich. Die wichtigsten Themen sind: ausgehaltene Balance zwischen extremen Ausprägungen von Werten innerhalb eines erträglichen Toleranzbereiches, z.B. Geiz - Verschwendung mit dem Toleranzbereich Sparsamkeit - Großzügigkeit und die Erklärung von acht typischen Kommunikationsstilen, wie z.B. dem „Sich beweisen“.
- [Url1] <http://www.systemsguild.com/GuildSite/TDM/June2002Computer.pdf>
Erfrischende Debatte zwischen Barry Boehm und Tom DeMarco über die Einführung von agilen Methoden.
- [Url2] <http://epic.onion.it/workshops/w10/slides02/index.htm>: **Pro-active Quality or Re-active Quality? - Workshop on ISO 9000 03/11/98**
Zurück zur ursprünglichen Bedeutung von Qualität - „Kundenzufriedenheit“.

„Nie aber weiß man genau, was man denkt,
bevor man nicht versucht hat, darüber zu
sprechen.“
Kurt Marti

2

Was soll die Software leisten?

Der Abstimmungs- und Einigungsprozeß was die Software leisten soll ist das für die Softwareentwicklung wichtiger als es scheint. Die Zufriedenheit der Auftraggeber hängt direkt davon ab, wie gut es gelingt ausgehend von den Wünschen zu einem schlüssigen Anforderungsprofil zu gelangen.

2.1 Projektauswahl

Wo soll als nächstes Zeit- und Geld in die Softwareentwicklung investiert werden?

Die Analyse der Geschäftsprozesse eines Unternehmen leistet für ein Softwareprojekt doppelte Vorarbeit:

- Die Geschäftsprozessanalyse stellt die Entscheidungsgrundlagen für die Auswahl des richtigen Softwareprojektes bereit, indem sie aufdeckt wo Geschäftsprozesse verändert und verbessert werden können.
- Die Geschäftsprozessanalyse liefert wichtige Grundlagen für die Rahmenanforderungen des Softwareprojektes - es werden bereits die Geschäftsprozesse, Geschäftsobjekte und die beteiligten Rollen festgestellt. Zudem werden bereits die Entscheidungen bestimmt, die den Ablauf der Geschäftsprozesse steuern.

Motivation für den Vorstand der Gründer AG, Hr. Aufenberg ein neues Projektverfolgungssystem entwickeln zu lassen ist die seit einem Jahr wirksame Rechtsform des Unternehmens „Aktiengesellschaft“. Zu den Pflichten einer AG gehört die regelmässige Veröffentlichung von Berichten. Die Projektsituation geht in die Quartalsberichte direkt ein. Dieser Geschäftsprozess wird von dem

alten System, das noch zu GmbH-Zeiten entstanden ist nicht ausreichend unterstützt. Zudem dauert der Weg von der Zeiterfassung auf Papier über die Erfassung im System bis zur Auswertung zu lange - die Ergebnisse liegen oft erst nach 6-8 Wochen vor. Von einer Änderung des Prozesses auf Direkteingabe durch die Mitarbeiter erhofft sich Hr. Aufenberg und sein Aufsichtsrat die Möglichkeit tagesaktueller Berichte.

In zweierlei Hinsicht ist unser Beispielprojekt typisch:

- Die Geschäftsprozessanalyse ist vor der Auftragsvergabe für die Software bereits erfolgt
- Eine formale Analyse, zudem unter Einsatz von Beratern und Werkzeugen, hat es gar nicht gegeben. Eine kurze Betrachtung der geschäftlichen Abläufe im Kreis der Betroffenen bringt bereits den Kern des zu lösenden Problems zu Tage.

In grossen Unternehmen mit vielen untereinander abhängigen Geschäftsprozessen ist die Analyse der Situation wesentlich aufwendiger und die Ableitung von Softwareprojekten eine eigenständige Aufgabe. Es gilt dann Softwarekomponenten und Geschäftsprozesse aufeinander abzustimmen.

2.2 Anforderungen aufnehmen

Anforderungen dokumentieren die Forderungen an die gewünschte Softwarelösung - klar verständlich in einfacher Sprache für Fachleute und Anwender ohne Computer-Fachchinesisch.

Würden Sie ein Haus kaufen oder bauen lassen, dessen Größe, Raumaufteilung, und verwendete Materialien erst nach Beginn der Bauarbeiten festgelegt würden? Wie oft würden Sie akzeptieren, dass die Bauarbeiter Entscheidungen für Sie fällen? Wie weit entspräche das fertige Haus später Ihren Bedürfnissen?

Für Softwareprojekte wird heute oftmals genau das absurde Vorgehen akzeptiert, dass die obigen Fragen nahelegen. Software hat scheinbar die Eigenschaft, dass sie jederzeit geändert werden kann, wenn etwas nicht so ist, wie es sein soll. Das ist nur scheinbar so, denn jeder Änderung hat Kosten und Aufwände zur Folge und da Geld und Zeit beschränkt sind werden immer nur die wichtigsten Änderungen durchgeführt werden. Im Hausbau hat sich bewährt, dass die Fragen nach Größe, Raumaufteilung und Verwendung von Materialien für den Rohbau am Anfang geklärt werden. Detailfragen nach Steckdosen, Fliesen, Tapeten usw. werden später geklärt und anfangs nur grob geschätzt.

Was ist bei Software der richtige Zeitpunkt für eine in Auftrag gegebene Software präzise festzulegen, was sie leistet? Die Angaben dazu müssen rechtzeitig vorliegen. Die Rahmenanforderungen sind bereits für die Planungsphase erforderlich, im Verlaufe des Projektes muss sichergestellt sein, dass die An-

forderungen rechtzeitig vorliegen. Eine Weg dies zu erreichen ist die enge Einbindung der Auftraggeber in das Projekt.



Erfolgsfaktor:

Nur wer rechtzeitig sagt, was er will, kann sicher gehen, dass er es auch bekommt.



Wer als Auftraggeber rechtzeitig klar sagt, was er will, legt damit das zentrale Fundament des Projekterfolges. Nur auf Basis dieser Aussagen können Aufwände, Kosten und Termine abgeleitet werden und der Auftraggeber kann dann entscheiden, welche seiner Wünsche er tatsächlich zu den genannten Bedingungen realisiert haben möchte.



Abbildung 2.1: Was haben die Auftraggeber des Eiffelturms über den fertigen Turm in diesem Moment der Bauphase gewusst?

Auftraggeber und Auftragnehmer müssen daher gemeinsam die Liste der Anforderungen klären. Jede Anforderung beschreibt für einen überschaubaren Teil des Systems das geforderte Verhalten. Die größte Klarheit wird erreicht, wenn alle Anforderungen schriftlich vorliegen und es zu jeder Anforderung mindestens ein Prüfbeispiel gibt.

Wenn die Auftraggeber dann sagen: "Ja, das ist das, was wir haben wollen" bedeutet dies für die Auftragnehmer: "Wir wissen, was wir liefern sollen. Ich

weiss, womit mein Kunde zufrieden sein wird. Ich weiss, mit welchen Inhalten das Projekt ein Erfolg wird."

Das Anforderungsprofil gibt in der Sprache des Auftraggebers wieder was die Software leisten soll. Damit ist der erste Schritt in Richtung auf die Realisierbarkeit mit dem Computer getan. Aus den zu Beginn unbekanntem und vielleicht auch unklaren und widersprüchlichen Gedanken, Wünschen und Ideen der Auftraggeber ist nachlesbarer Text geworden.

Vollständigkeit von Anforderungen

Wie können Auftraggeber und Auftragnehmer feststellen, ob eine vorliegende Liste der Anforderungen vollständig ist?

Die Problemkreise bei der Frage nach der Vollständigkeit von Anforderungen sind:

- Durch die Erklärung der Vollständigkeit erfolgt die Übernahme der Verantwortung durch den Auftraggeber
- Bei einer großen Menge von Anforderungen ist die Übersicht über das Anforderungsprofil nötig → Abschnitt „Gliederung von Anforderungen“.
- Die Verantwortlichen brauchen Klarheit über die Folgen des Anforderungsprofilinhaltes (z.B. für Kosten/Termine/Erfolgschancen der zu erstellenden Lösung).
- Übersicht über das Anforderungsprofil
Zur Gewinnung der Übersicht über das Anforderungsprofil muß zunächst einmal der Inhalt des Profils festgelegt werden → Abschnitt 2.4 Anforderungsprofil erstellen. Das Anforderungsprofil stellt den vollständigen Satz von Anforderungen dar, den ein zu erstellendes Informationssystem zu erfüllen hat.

Auftraggeber haben häufig Angst, dass sie im Anforderungsprofil etwas vergessen haben. Denn aus der Erfahrung der Vergangenheit wissen sie, dass spät abgelieferte Anforderungen teuer werden, zu Projektverzögerungen führen oder im schlimmsten Fall gar nicht mehr umgesetzt werden können (Abschnitt 2.5 „Änderungen der Anforderungen“).

Schliesslich ist die vollständige Berücksichtigung aller Quellen für Anforderungen zu prüfen:

- Alle Kundendokumente berücksichtigt?
- Alle Verantwortliche beteiligt?
- Alle Partnersysteme beteiligt?
- Sonstige vom Kunden benannte Quellen berücksichtigt?

Gliederung von Anforderungen

Gewöhnliche Menschen sind nicht in der Lage mehr als etwa 5-9 verschiedene Dinge gleichzeitig so wahrzunehmen, dass sie die einzelnen Dinge gedanklich auseinander halten können. Dies erkannte Miller [Mil56] und stellte die 7 ± 2 Regel auf, wonach die größte Verständlichkeit bei einer Gliederungstiefe von 7 plus oder minus 2 Elementen erreicht wird. Ein Buch sollte daher zwischen 5 und 9 Kapiteln haben und jedes Kapitel zwischen 5 und 9 Unterkapiteln und so weiter. Bei einer Gliederungstiefe von 7 liegt in etwa das Optimum. Aus dieser Regel lässt sich direkt ableiten, wie viele Gliederungsebenen erforderlich sind, um eine gegebene Anzahl von Anforderungen wohlstrukturiert bearbeiten zu können:

Anzahl Gliederungsebenen	Anzahl Anforderungen		
	ab 5 Teilungen pro Ebene	mittel 7 Teilungen pro Ebene	bis 9 Teilungen pro Ebene
keine	1	7	9
1	6	49	81
2	26	343	729
3	126	2401	6561
4	625	16807	59049

Mit nur 3 Gliederungsebenen lassen sich schon einige hundert Anforderungen strukturieren.

Mögliche Gliederungsebenen für Anforderungen sind:

- nach Themengebieten/Unternehmen/Anforderungsgebieten
- nach Geschäftsprozessen/Anwendungsfällen
- nach Geschäftsobjekten
- nach Partnersystemen
- nach Priorität
- nach Anforderungsdimensionen

Abbildung 2.2 zeigt wie ein Anforderungsgebiet nach Themen gegliedert werden kann.

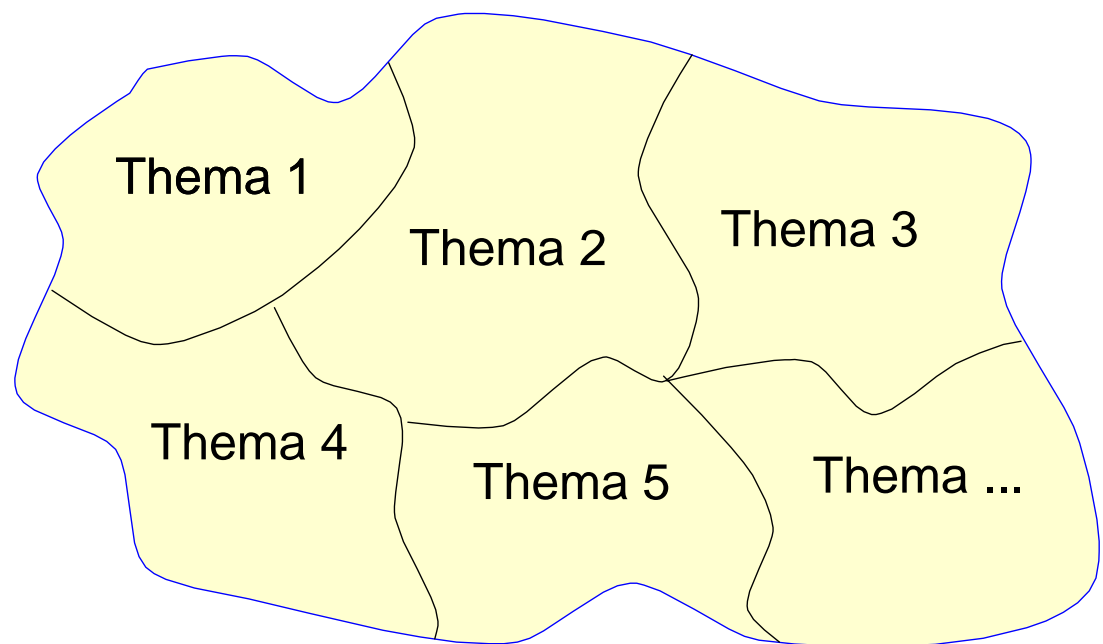


Abbildung 2.2: Gliederung des Anforderungsgebietes nach Themen

Im Gespräch mit Hr. Exner schlägt Frau Probst vor, ausgehend von den Stichpunkten, die Hr. Exner und Hr. Aufenberg in den Erstinterviews genannt haben, Themengebiete auszufällen entlang der die detaillierte Anforderungsaufnahme nun erfolgen soll. Hr. Exner ist einverstanden und beide stellen folgende Themenliste auf:

Projektdefinition, Mitarbeiter und Verantwortlicher, Kunden, Zeiterfassung, Projektberichte.

Auf der letzten Gliederungsebene werden die Anforderungen erfasst. Abbildung 2.3 zeigt wie z.B. auf der Gliederungsebene „Thema 2“ Anforderungen zu diesem Thema voneinander abgegrenzt werden können. So wird Thema für Thema bearbeitet, bis alle Anforderungen zu allen Themen bekannt sind.

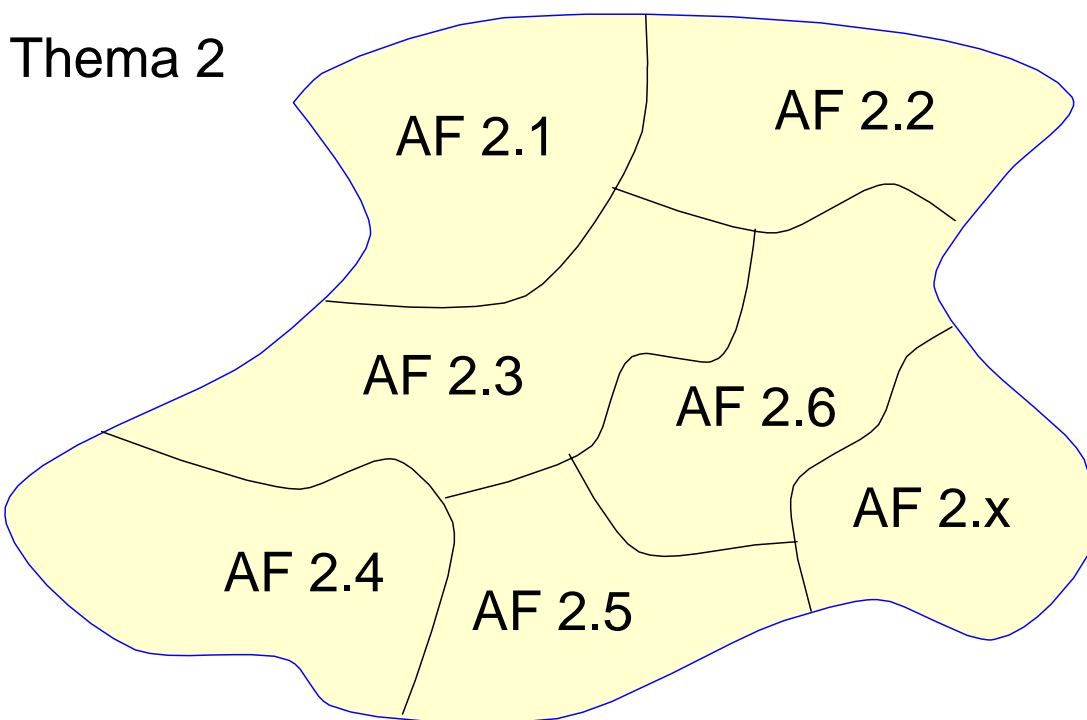


Abbildung 2.3: Gliederung eines Themas nach Anforderungen

Entlang der Gliederung nach Themen soll so schliesslich schrittweise das gesamte Anforderungsgebiet über feingranulare Anforderungen abgedeckt sein.

Feingranulare Anforderungen erfassen

Die schriftlich festgehaltene Anforderung ist der erste Schritt die Wünsche des Auftraggebers für die technische Umsetzung vorzubereiten. Die Klärung des Auftragsinhalts anhand schriftlicher Anforderungen liegt im gemeinsamen Interesse von Auftraggeber und Auftragnehmer, denn Mißverständnisse führen später zu unnötigen Kosten, Aufwänden und Verzögerungen die keinem nutzen. Deswegen sollte die Anforderungsaufnahme sehr ernst genommen werden und ständig geprüft werden, ob man sich gegenseitig verstanden hat. Der Detaillierungsgrad ist dabei dem erwarteten Nutzen anzupassen - wichtige Punkte mit hohen Risiken sollten sehr genau betrachtet werden, bei Randthemen ist es sinnvoll, sich kurz zu fassen.

Als erstes Thema gehen Hr. Exner und Frau Probst das Thema „Zeiterfassung“ an. Hr. Exner erklärt, dass es darum geht die Aufwände der Mitarbeiter auf die Projekte zu verteilen. Die Länge des Personentages hängt dabei von der Wochenarbeitszeit ab.

Beispieldokument 2.1 zeigt eine Anforderung, wie sie in unserem Beispielprojekt aufgenommen wird.

Zur Projektverfolgung der tatsächlichen Personalaufwände eines Projekts werden die Zeitblöcke je Mitarbeiter mit Tagesdatum und geleisteten Stunden ausgewiesen. Je Projekt können die bisher geleisteten Personentage ermittelt werden. Ein Personen-

tag besteht aus einem Fünftel der Wochenarbeitszeit der Person. Ein Zeitblock ist eindeutig einem Projekt zugeordnet.

Beispieldokument 2.1: Eine Anforderung

Die folgenden Abschnitte zeigen auf, wie der Klärungsprozess für Anforderungen im Detail durchgeführt und auf den Punkt gebracht wird.

Reihenfolge der Bearbeitung

Bei der Anforderungsaufnahme werden die relevanten Geschäftsprozesse, Anwendungsfälle, Geschäftsobjekte, Anforderungen und Prüfbeispiele gesucht bzw. verfeinert. Dabei entstehen offene Fragen, die geklärt werden müssen. Ein typischer Diskussionsstrang innerhalb einer Anforderungsaufnahme führt dabei kreuz- und quer durch die oben genannten Ergebnisse.

Frau Probst fragt Hr. Exner nach dem Geschäftsprozess „Quartalsbericht erstellen“. Im Rahmen der Diskussion kommen sie auf den Anwendungsfall „Projektstatus ermitteln“. Sie diskutieren darüber, wie der Projektstatus dem Geschäftsobjekt „Projekt“ zugeordnet werden soll - es stellt sich die offene Frage, ob ein Teilprojekt einen eigenen Projektstatus haben soll. Hr. Exner formuliert die Anforderung: „Teilprojekte können den Status fertiggestellt haben. Das Gesamtprojekt erhält den Status fertiggestellt, wenn alle seine Teilprojekte fertiggestellt sind“. Als Prüfbeispiel nennt er ein Projekt mit drei Teilprojekten A, B und C. Nur A ist fertiggestellt - das Gesamtprojekt gilt dann als nicht fertiggestellt“. Dieses Prüfbeispiel bringt den Gedanken auf, dass es einen Anwendungsfall geben muss, der den Projektstatus verfolgt ...

Da eine typische Diskussion oft scheinbar unsystematisch abläuft erscheint es schwierig, den Überblick darüber zu behalten, ob wirklich alle für das System relevanten Punkte beleuchtet worden sind. Die Versuchung ist gross, lieber der Reihe nach vorzugehen, also z.B. einen Geschäftsprozess nach dem anderen zu bearbeiten, und „Abschweifungen“ nicht zuzulassen. In unseren Projekten wurden die besten Ergebnisse erreicht, wenn der freie Lauf der Diskussion eine Weile lang zugelassen wurde und dabei Nachbarergebnisse entstehen. Schliesslich geht es dann wieder nach Plan weiter. Wichtig ist es, den Überblick zu behalten, welche Punkte bereits behandelt wurden. Dabei kann eine Mindmap wie in Abbildung 2.4 gezeigt helfen:

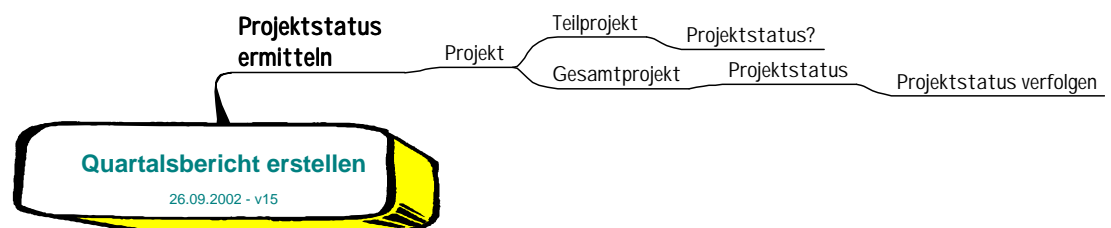


Abbildung 2.4: Mind-Map zu einer Diskussion innerhalb der Anforderungsaufnahme

Zum Schluss der Anforderungsaufnahme helfen folgenden Fragen bei der Klärung, ob alle Punkte angesprochen wurden:

- Sind zu diesem Geschäftsprozess alle Anwendungsfälle genannt worden?
- Sind zu diesem Anwendungsfall alle Anforderungen aufgenommen?
- Sind zu diesem Geschäftsobjekt alle Anforderungen aufgenommen?
- Sind zu dieser Anforderung alle Prüfbeispiele genannt worden?
- Sind alle Offenen Fragen beantwortet?
- Gibt es noch weitere Offene Fragen?

Dokumentation offener Fragen

Offene Fragen dokumentieren Unklarheiten, Auslassungen, Widersprüche, Verständnisprobleme und andere offene Punkte. Der englische Begriff TBD (to be defined) umschreibt solche offenen Punkte. Der richtige Umgang mit offenen Fragen ist ungemein wichtig für den erfolgreichen Verlauf des Projektes. Offene Fragen müssen ausgesprochen und dokumentiert werden. Dies kann bereits ein Widerspruch sein, denn das Verfolgen von Offenen Fragen ist mit Arbeit verbunden und provoziert die Notwendigkeit des Fällens von Entscheidungen (siehe 7.2 „*Entscheidungen fällen*“). Da behält so mancher Mitarbeiter seine Fragen doch lieber für sich. Unbeantwortete Offene Fragen und nicht getroffene Entscheidungen sind jedoch Gift für den Fortschritt des Projektes.

In unseren Projekten hat es sich bewährt, offene Fragen an zentraler Stelle zu erfassen, so dass alle Teammitglieder darauf zurückgreifen können. Zu jeder offenen Frage wird festgehalten:

- Inhalt der offenen Frage
- An wen die Frage gerichtet ist.
- Datum und Autor der Fragestellung
- Angenommene Antwort
- Tatsächliche Antwort
- Datum und Autor der Antwort

Die angenommene Antwort ist immer dann notwendig, wenn der Fluss der Arbeit ohne die Antwort unterbrochen werden würde. Voraussetzung ist, dass der Fragesteller das eigene Urteilsvermögen für ausreichend hält, um eine wahrscheinlich richtige bzw. sinnvolle Antwort zu nennen. In dieser Form angewendet ergibt sich eine besonders schnelle Form des Treffens von Entscheidungen: der Fragesteller fällt die Entscheidung und lässt sich diese vom Empfänger der Frage bestätigen. Gerade, wenn das Risiko einer Fehlentscheidung gering ist, ist dies der richtige Weg - abzuwarten bis die „offizielle“

Antwort kommt führt garantiert zu Zeitverlust und schadet bei geringem Risiko oft mehr, als die Entscheidung an Ort und Stelle unter Vorbehalt zu postulieren. Hat es sich eingespielt, dass nur in seltenen Fällen das Arbeiten in Richtung einer angenommenen Antwort „für die Katz“ war, gewinnt das gesamte Projekt.

Einige offene Fragen haben die Eigenschaft als „Dauerbrenner“ durch das Projekt geschleppt zu werden. Bei wichtigen Punkten ist dies ein dringendes Alarmzeichen, dass eine Entscheidung überfällig ist. Bei unwichtigen Randthemen ist es sinnvoller, das sich ein Verantwortlicher ein Herz nimmt und die Frage zu den Akten legt. Die simple Antwort „irrelevant, weil ...“ reicht in diesem Fall.

Durchführung mit Fachleuten

Wie wichtig die aktive Beteiligung der Anwender und Auftraggeber ist wird von uns im Kapitel 7 herausgestellt. Für die Anforderungsaufnahme ist es wichtig, dass Fachleute für das Thema beteiligt sind.

Wenn Sie bei den Erstinterviews oder im Verlauf der Anforderungsaufnahme das ungute Gefühl nicht loswerden, dass die Ansprechpartner des Auftraggebers ratlos sind, dann sollten Sie rechtzeitig reagieren. Es ist uns häufiger passiert, als uns lieb war, dass der Auftraggeber die Mitarbeiter für die Anforderungsaufnahme nach Kompetenz ausgesucht hat: diejenigen, die am wenigsten im Tagesgeschäft vermisst wurden waren unsere Gesprächspartner. Das waren dann manchmal Mitarbeiter die von dem Gebiet über das sie Auskunft erteilen sollten, keine Ahnung hatten - die kompetenten Mitarbeiter wurden ständig gebraucht und waren nicht verfügbar. Auf der Requirements Engineering Konferenz 1998 [ICRE98] wurde geschildert, wie eine Softwarefirma ein System für Aktienhändler erstellen sollte, diese aber auf keinen Fall bei der Arbeit unterbrechen durfte. Nur wenn sie sicherstellen, dass dem Auftraggeber klar ist, welchen Folgen solch verschlechterte Bedingungen der Anforderungsaufnahme haben, können sie den Erfolg des Projektes sicherstellen. Entweder indem der Auftraggeber die Bedingungen auf Ihr Anraten hin ändert oder die Folgen bzgl. Aufwand und Ergebnis akzeptiert. Zur Not müssen Sie sich ggf. auf Kosten des Auftraggebers selbst in das fachliche Thema einarbeiten.

2.3 Prüfbeispiele dokumentieren

Woher wissen Sie, ob das von Ihnen in Auftrag gegebene Haus Ihren Anforderungen entspricht? Sie haben für alle Ihre Anforderungen Prüfmöglichkeiten. Sie können die Grundstücksfläche auf den Zentimeter genau ermitteln, ebenso die Wohnfläche. Sie können die Anzahl der Fenster zählen und die Farbe des

Putzes mit Referenzfarbkarte überprüfen. Sie können vergleichen, ob Ihr Haus zu den veranschlagten Kosten termingerecht erstellt werden wird. Alle relevanten Prüfkriterien haben Sie vor Baubeginn mit Ihrem Auftragnehmer vereinbart.



Erfolgsfaktor:

Nur das, was ich prüfen kann, kann ich auch beurteilen.²



Wie beeinflusst die Anwendung dieses Prinzips den Erfolg Ihres Softwareprojektes? Die Erfüllung jeder Anforderung wird mittels eines oder mehrerer Beispiele prüfbar gemacht. Beispiele helfen erheblich zwischen Menschen und Maschine zu vermitteln. Die Verständlichkeit und Schlüssigkeit einer Anforderung erhöht sich mit jedem Prüfbeispiel, das für die Anforderung festgehalten wird. Prüfbeispiele werden in natürlicher Weise von Menschen bei der Erstellung von Modellen verwendet (siehe Abschnitt 3.1 „Modellieren“).

Fr. Probst fragt Hr. Exner nach einem konkreten Beispiel für die Berechnung des Gesamtaufwandes für ein Projekt.

Situation: Die Projekte „Autoteile Internet-Shop“ und „Vertragsmanager“ sind beauftragt. Herr Friesen, Frau Meidrich, Hr. Sturm und Frau Sauer haben bisher folgende Aufwände geleistet:

Mitarbeiter	Aufwände pro Projekt	
	Autoteile Internet Shop	Vertragsmanager
Herr Friesen	25,0 h	42,5 h
Frau Meidrich	18,0 h	-
Hr. Sturm	-	19,5 h
Frau Sauer	13,0 h	-

Aktion: Die Projekt- und Mitarbeiterangaben sowie die Aufwände werden erfasst und die Auswertung „Gesamtaufwand je Projekt“ angestossen.

Erwartetes Ergebnis:

Autoteile Internet Shop: 56 h, Vertragsmanager: 62 h.

Beispieldokument 2.2: Ein Prüfbeispiel

In den meisten Fällen brauchen Sie ihre Auftraggeber überhaupt nicht explizit um Prüfbeispiele zu bitten. Im Rahmen der Anforderungsaufnahme werden ihnen immer wieder konkrete Fälle und Dokumente genannt. Erarbeiten Sie aus diesen Ansatzpunkten die Prüfbeispiele.

Zwei einfache Fragen helfen, weitere Prüfbeispiele zu finden:

- Ist das immer so?
- Was könnte dazu führen, daß das Beispiel in bestimmten Situationen nicht stimmt?

² Vgl. auch “You can not control what you can not measure” - Tom DeMarco [DeM99]

Die Prüfbeispiele werden zur Beurteilung nicht nur der Anforderungen sondern auch aller folgenden Entwicklungsergebnisse herangezogen. Folgende Fragen sind mit dem Vorliegen von Prüfbeispielen jederzeit beantwortbar: Wie konsistent sind die Anforderungen mit ihren zugehörigen Prüfbeispielen? Wieviele der Prüfbeispiele werden von der bisher erstellten Software bzw. den dazugehörigen Modellen bestanden? Wie beurteilen Auftraggeber und Auftragnehmer die erstellten Ergebnisse im Verhältnis zur übrigen Projektsituation (bisher verursachte Kosten, abgelaufene Zeit usw.)?³

Am Ende des Projektes werden die Prüfbeispiele zu Abnahmekriterien für die ausgelieferte Software. Sie können Abnahmekriterien im Sinne von ISO 9000-3 sein, wenn sie als verbindlicher Test für die Erfüllung von Anforderungen deklariert werden.

2.4 Anforderungsprofil erstellen

Wie können die Software-Wünsche aller Beteiligten mit den verfügbaren Mitteln umgesetzt werden? Wo liegen die Grenzen? Was soll weggelassen werden? Wo kann der Umfang reduziert werden? Was wird später realisiert? Wo sind die Kosten im Verhältnis zum Nutzen einfach zu hoch?

Welche Anforderungen machen nach dem Pareto-Prinzip 80% des Erfolges aus und sind mit 20% des Aufwands umzusetzen? Welche Prioritäten ergeben sich daraus?



Erfolgsfaktor:

Die richtig priorisierte Auswahl der Anforderungen legt das zu realisierende Anforderungsprofil fest.

In den meisten Projekten ist es unmöglich alle vorliegenden Wünsche in vollem Umfang zu berücksichtigen. In der Liste der schriftlichen Anforderungen tauchen nur noch die Wünsche auf, bei denen sich die Beteiligten darauf einigen konnten zu untersuchen, welche Folgen (Kosten, Aufwände, technische Probleme) mit der Umsetzung in Abhängigkeit vom Umfang der Umsetzung verbunden sein werden. Für den Umfang der Umsetzung wird z.B. eine Skala von „fehlt“ bis „optimal gelöst“ verwendet.

Die Ergebnisse dieser Untersuchung fließen in die Festlegung der Anforderungsprofile ein. Auftraggeber und Auftragnehmer einigen sich pro Meilenstein auf ein Anforderungsprofil. Die Entscheidung über den Inhalt des Anforderungsprofil erfolgt, indem für jede potentielle Anforderung explizit zu fest-

³ Diese Transparenz macht die politische Brisanz dieses Erfolgsfaktors aus. Es ist eine selbsterfüllende Prophezeiung, wenn aus Angst den Erfolg zu gefährden, durch Verzicht auf diesen Erfolgsfaktor Probleme nicht offen gelegt werden. Die Probleme gehen davon nicht weg, nur der Erfolg des Projektes wird unwahrscheinlicher.

gelegt wird, ob sie ins Anforderungsprofil aufgenommen wird oder nicht. Diese Entscheidung kann nur von einem Verantwortlichen wahrgenommen werden. Ein Verantwortlicher muss die Folgen die sich durch die Aufnahme der Anforderung in das Profil ergeben (Kosten, Termine, Veränderungen) autorisieren können.

Ein potentielle Anforderung kann also

- ins Profil aufgenommen werden
- zurückgestellt werden bis sie verantwortet wird
- verworfen werden

Zur Klärung der Vollständigkeit von Anforderungen sollten in strittigen Fällen zurückgestellte und verworfene Anforderungen dokumentiert werden.

Das Anforderungsprofil hat folgende Dimensionen:

- Umfang/Aufzählung der Anforderungen
- Höhe/Bewertung der Anforderungen
- Detaillierung der Anforderungen

Im Einzelnen leisten die Dimensionen Folgendes:

- Umfang/Aufzählung der Anforderungen
In dieser Dimension geht es darum die Anforderungen zu benennen und in eine übersichtliche Struktur einzuordnen. "Das System soll die Erfassung der Projekt durch die Mitarbeiter geleisteten Zeiten unterstützen. Das System soll pro Projekt das noch verfügbare Restbudget ermitteln können ..."
- Höhe/Bewertung der Anforderungen
In dieser Dimension wird eine Anforderung bzgl. des Grades der nötigen Umsetzung bewertet. Es wird zum Beispiel entschieden ob die Anforderung besonders gut/schön/schnell umgesetzt werden soll. "Der monatliche Projektbericht ist in Farbe auszudrucken und soll ein ansprechendes Erscheinungsbild haben. Die Angaben im Projektbericht sollen tagesaktuell sein und eine Genauigkeit von einem halben Personentag pro Mitarbeitermonat haben."
- Detaillierung der Anforderungen
Die inhaltlichen Details der Anforderung stellen die dritte Dimension des Profils dar. Die Höhe/Bewertung ist prinzipiell eine besondere Form der Detaillierung - sie sollte jedoch unserer Ansicht nach von anderen Formen der Detaillierung abgegrenzt werden damit die Vollständigkeitsaussage klarer zu treffen ist.

Das Anforderungsprofil legt für die vorliegenden Anforderungsbereiche den Umfang fest, in dem die Anforderungen aus dem jeweiligen Bereich umzusetzen sind. Jedes Anforderungsprofilelement stellt damit einen verbindlichen Auftragsbestandteil dar.

Was soll die Software leisten?

Ein Mögliches System zur Bewertung von des Umfangs von Anforderungen nutzt folgende Stufen:

Symbol für Bewertung	Bedeutung
++	optimal gelöst
+	gut gelöst
o	vorhanden
-	vorhanden, aber unvollständig
--	fehlt

Die folgende Abbildung zeigt ein Beispiel für ein Anforderungsprofil, das mit diesen Symbolen erstellt wurde:

Um- fang	++							
	+							
	o							
	-							
	--							
		Zeit- blöcke	Projekt	Mit- arbeiter	Projekt- status	Kundenda- ten	...	
Anforderungsbereich								

Beispieldokument 2.3: Ein Anforderungsprofil

Ausschnitte der Wunschliste werden zu Anforderungsprofilen, wenn

- klar ist, was wichtig ist,
- die Beteiligten sich einigen, was sie fordern wollen,
- der Umfang der Forderungen festgelegt ist,
- das Anforderungspaket geschnürt ist und
- die Umsetzung in Auftrag gegeben wird.

Die Erstellung von Anforderungsprofilen hat sich bei der Entwicklung von Produkten außerordentlich bewährt. Nur die richtige Mischung von Eigenschaften eines Produktes bewährt sich am Markt. Die Übertragung dieses Erfolgsrezeptes auf Auftragsprojekte führt zum selben Effekt: die Zufriedenheit der Auftraggeber steigt mit dem Grad der Abdeckung des vereinbarten Anforderungsprofils durch die fertige Software.

Wie Anforderungsprofile für die Projektplanung optimal eingesetzt werden können, erfahren Sie unter

www.b-agile.de/Erfolgsrezepte/Anforderungsprofile.html !!!.

2.5 Änderungen der Anforderungen

Änderungswünsche und Anforderungen unterscheiden sich prinzipiell nur durch den Zeitpunkt der Nennung des Wunsches. Änderungswünsche entstehen nachdem bereits eine funktionierende Software ausgeliefert wurde. Zu diesem Thema finden Sie in Abschnitt 5.2 „*Fehlermeldungen und Änderungswünsche*“ weitergehende Hinweise.

Davon zu unterscheiden sind Änderungen an den Anforderungen während der Entwicklung noch bevor eine Anforderung in die funktionierende Software eingeflossen ist.

Anforderungen ändern sich

Es gibt viele Gründe warum sich Anforderungen an Software ändern. Der häufigste Grund ist, dass sich das Aufgabenfeld verändert in dessen Kontext die Software eingesetzt wird. Technische Anforderungen ändern sich heute sehr schnell, da die Lebenszykluszeit der eingesetzten Softwareprodukte heute um vieles geringer ist als zu Beginn des Computerzeitalters. Neue Versionen von Datenbanken, Netzwerken, Betriebssysteme und anderer Infrastruktur bringen ständige Veränderung mit sich. Der Wechsel von Personen und Organisationsformen führt ebenfalls zur Schnellebigkeit von Anforderungen.

Anforderungen ändern sich nicht

Es gibt eine ganze Reihe von Anforderungen, die über viele Jahre stabil bleiben. Dies betrifft Anforderungen an den grundlegenden Gang von Geschäften in Branchen, deren Verfahren sich über Jahrzehnte nicht geändert haben: Versicherungen und Banken haben zum Beispiel solche stabilen Basisverfahren. Denken Sie an Darlehen oder Lebensversicherungen – hier werden Verträge mit sehr langen Laufzeiten geschlossen – es muss sichergestellt sein, dass auch noch am Ende der Laufzeit die Vertragsinhalte von der eingesetzten Software korrekt umgesetzt werden. Aber auch im Umfeld dieser stabilen Basisanforderungen ändern sich einige Details sehr schnell, z.B. wenn sich gesetzliche Anforderungen verändern.

Projektlaufzeiten

Die Projektlaufzeit hängt vom Umgang der Anforderungen ab, die dem Projekt als Aufgabe gestellt werden. Es gilt, das „richtige“ Anforderungspaket zu

schnüren. Hier beginnt der Teufelskreis. Die Auftraggeber sind aus der Vergangenheit vielfach gewohnt, dass es lange Projektlaufzeiten gibt und das ein einmal eingesetztes System dann möglichst nicht mehr geändert werden soll, da dies mit zusätzlichen Kosten und Folgeproblemen verbunden ist. Aus dieser Erfahrung heraus entsteht zu Beginn eines neuen Projektes die Erwartung „dies ist für lange Zeit die letzte Gelegenheit, Anforderungen zu nennen“. So wird in so manchen Projekt möglichst noch alles an Wünschen hineingepackt, was politisch durchsetzbar ist. Der Anforderungsrahmen wird damit größer. Mehr noch – da ja klar ist das nun für Monate, meistens jedoch eher Jahre keine weiteren Anforderungen aufgenommen werden, muß auch für die Zukunft vorgesorgt werden. Alle Anforderungen die in absehbarer Zukunft auf die Auftraggeber zukommen werden bringen zusätzliche Brisanz. Bei diesen Anforderungen sind die Details noch nicht klar. Aber es ist sicher, wenn diese Anforderungen nicht mit in dieses Paket aufgenommen werden, dann werden die Anforderungen aufgrund der langen Projektlaufzeit nicht rechtzeitig erfüllt sein. „Sicherheitshalber“ nennt der Auftraggeber daher die unklaren Anforderungen und läßt sie – notgedrungen auch unscharf – in das Anforderungspaket aufnehmen. Nun nimmt das Schicksal seinen Lauf – ein Projekt, das mit unscharfen Anforderungen arbeitet hat einen erheblich höheren Kommunikationsaufwand und viel größere Änderungshäufigkeiten als dies mit klaren, scharfen Anforderungen der Fall wäre. Das Projekt verzögert sich zusätzlich. Am Ende der nun noch größeren Projektlaufzeit haben sich noch mehr Anforderungen geändert und das geschnürte Anforderungspaket stimmt nicht mit den dann tatsächlich vorhandenen Anforderungen überein.

Kleine Anforderungspakete

Offensichtlich bieten kleinere Anforderungspakete höhere Erfolgchancen. Die Effekte, die durch „aufgeschaukelte“ Projektlaufzeiten entstehen können durch kleinere Anforderungspakete vermieden werden.

Proaktivität

Wenn man Anforderungspakete als „Vektoren“ begreift, die einen Ausgangspunkt und eine Richtung haben, dann ist es möglich die zukünftigen Anforderungen vorherzuplanen. Dieses Vorgehen ist mit einer gewissen Unschärfe verbunden, da es in der Realität auch einmal zu heftigen Veränderungen von Anforderungsprofilen kommen kann. Aber die Chancen sind sehr gut bei diesem Vorgehen den tatsächlichen zukünftigen Anforderungsbedarf zu decken, wenn mit kleinen Anforderungspaketen gearbeitet wird.



Erfolgsfaktor:

Stellen Sie sich aktiv darauf ein, dass es zu Änderungen der Anforderungen kommen wird und berücksichtigen Sie diese so früh wie möglich!



Abbildung 2.5 erläutert das Vorgehen. Zu Beginn des Projektes (1) wird von einem bestehenden Anforderersprofil aus entlang des Vektors in Richtung (2) gearbeitet. Durch die kurze Projektlaufzeit kann rechtzeitig auf die veränderte Richtung auf (3) zugearbeitet werden. Zu diesem Zeitpunkt ist die erneute Richtungsänderung bekannt und das Projekt passt sich entsprechend an. Stellen Sie sich vor in der gleichen Zeit wäre einfach in Richtung des ersten Vektors weitergearbeitet worden – das Ergebnis würde zu weiten Teilen nicht mehr mit den wirklichen Anforderungen übereinstimmen.

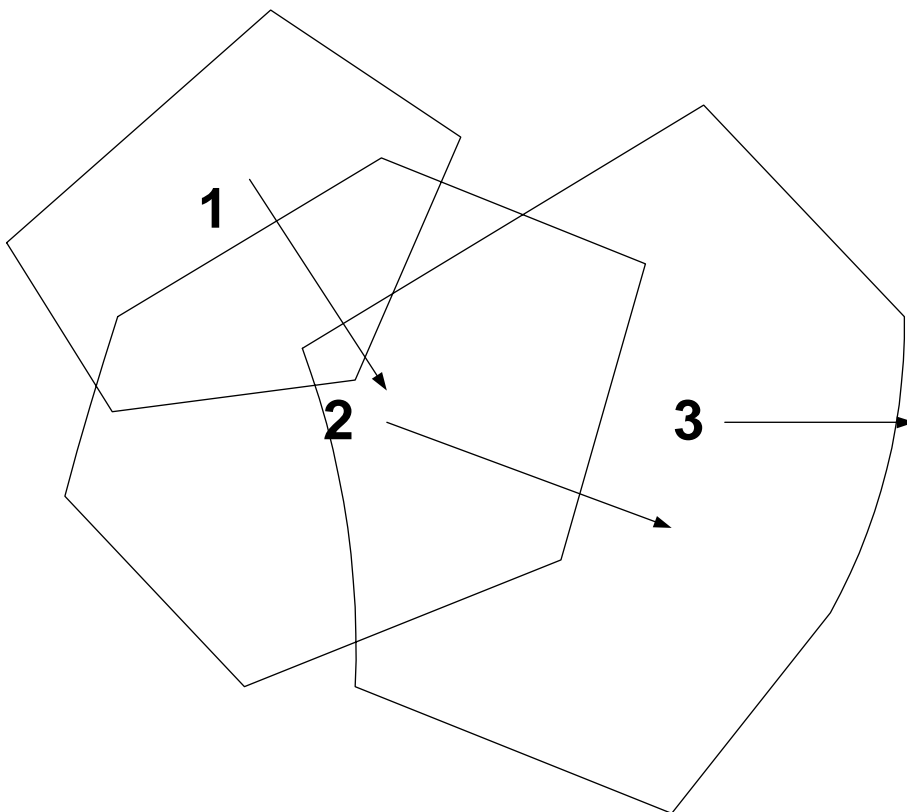


Abbildung 2.5 Änderung der Richtung von Anforderungen

Änderungsgeschwindigkeit von Anforderungen

Eine zu grosse Änderungsgeschwindigkeit von Anforderungen gefährdet den Erfolg ihres Projektes. Wenigstens die Rahmenanforderungen müssen so stabil bleiben, dass in jedem Entwicklungsschritt ein Vorankommen in Richtung dieser Rahmenanforderungen möglich ist.



Erfolgsfaktor:

Achten Sie darauf, dass die Rahmenanforderungen für Ihr Projekt stabil bleiben.

Bei der Auswertung der Umfrage hat der Erfolgsfaktor „Änderungen berücksichtigen“ zwei Gruppen von Projekten hervorgebracht, die diesen Faktor höchst unterschiedlich bewertet haben: Eine Gruppe von Projekten beurteilte diesen Faktor als fördernd, eine andere Gruppe von Projekten beurteilte diesen Faktor als hindernd. Es gab kein Mittelfeld! Es gibt also offensichtlich zwei grundsätzliche Ansätze, mit Änderungen umzugehen:

- Den Einfluss von Änderungen auf das Projekt grundsätzlich vermeiden.
- Mit Änderungen proaktiv umgehen.

Wir empfehlen Ihnen sich klar zu entscheiden, welchen der Ansätze sie wählen wollen. Egal wie Sie sich entscheiden - auf jeden Fall sollten Sie darauf bestehen, dass die Rahmenanforderungen für das Projekt stabil bleiben. Wenn sich die Rahmenanforderungen ändern bedeutet dies, dass ein komplett anderes Projekt von Ihnen verlangt wird. Es ist dann sinnvoll, auch explizit ein neues Projekt zu starten.

Damit Sie feststellen können, wann Ihre Rahmenanforderungen drohen instabil zu werden, ist es wichtig, diese bereits zu Beginn des Projektes ausdrücklich und schriftlich festzuhalten.

2.6 Der Wert von Software

Worin besteht eigentlich der Wert einer Software? Wie bei anderen Gütern auch gibt es natürlich verschiedene Wege den Wert von Software zu bemessen. Folgende Fragen sind zum Beispiel geeignet den Wert einer Software zu bemessen:

- Was ist der Preis, den die Software am Markt erzielt?
- Was ist der Nutzen, den die Software bei Ihrem Einsatz liefert?
- Was ist der Schaden, den die Software anrichtet, wenn sie nicht oder fehlerhaft funktioniert?

Software hat einen anderen Lebenszyklus als andere Güter. Betrachten wir einige Beispiele:

- Ein typischer Lebenszyklus für ein Autokauf besteht aus: Geeignetes Auto anschaffen, Autobetrieb mit Wartung/Versicherung/Kraftstoff/Steuer usw. sicherstellen, Auto verkaufen oder Reparaturen vornehmen solange es sich lohnt und dann verschrotten.

- Ein typischer Lebenszyklus für einen Fernseher besteht aus: Geeignetes Gerät anschaffen, Gerät betreiben, Reparaturen vornehmen solange es sich lohnt, Gerät verschrotten.
- Ein typischer Lebenszyklus für Software besteht aus: Software erstellen , Software wegen Änderung des Computers ändern, Software wegen Änderung der Infrastruktur ändern, Software wegen Änderung der Anwenderwünsche ändern, Software wegen neuer gesetzlicher Anforderungen ändern, ...

Der Wert von Software liegt in der ständigen Anpassbarkeit an neue technische und fachliche Gegebenheiten! Software verliert Ihren Wert wenn unter neuen Bedingungen die Änderung teurer als die Neuanschaffung wird. Software ist also so etwas wie ein unendlich reparierbares Auto.

2.7 Literatur

- [ICRE98] **Proceedings Third International Conference on Requirements Engineering**, IEEE Computer Society, 1998
Während dieser Konferenz entstand das im Vorwort gezeigte Urbild der von uns verwendeten Brücke.
- [Tha02] G. E. Thaller: **Software-Anforderungen für Webprojekte Vorgehensmodelle, Spezifikation, Design**, Galileo Press, 2002
- [Rob99] S. Robertson: **Mastering the Requirements Engineering Process**, Addison-Wesley, 1999
Alles was man über die Anforderungsaufnahme wissen muss. Im Anhang finden sich Vorlagen für eine gute gegliederte Anforderungsaufnahme nach dem „Volere“ - Prinzip.
- [Rup02] C. Rupp, SOPHIST GROUP: **Requirements Engineering**, Hanser, 2002
Frau Probst hat sicher das Buch von Chris Rupp und den Sophisten gelesen

„Eine (wahre) Theorie hat die Wirklichkeit als Modell.“
Mick Krippendorf

3

Vom Anforderungsprofil zum Modell

Aus dem Text des Anforderungsprofils gilt es im nächsten Schritt ein Modell zu machen, das bereits etwas mehr der Sicht des Computers entspricht. Auf der anderen Seite muss dieses Modell für die Projektbeteiligten verständlich bleiben. Modelle ermöglichen es, Ideen auf einfache Art auszutauschen und sie auf ihre Stichhaltigkeit hin zu überprüfen.

3.1 Modellieren

Nach Untersuchungen aus den 80er Jahren haben kreative Softwareentwickler folgenden Ansatz zur Problemlösung [Gla95]:

- Im ersten Schritt erstellt der Entwickler ein geistiges Modell der Lösung
- Dann prüft er sein Modell auf Richtigkeit, indem er es anhand konkret gedachter Situationen durchspielt. Wenn er Unstimmigkeiten entdeckt, wird das geistige Modell angepasst und erneut durchgespielt.
- Dieser Vorgang der mentalen Simulation wird mit verschiedenen Sätzen von Prüfbeispielen wieder und wieder durchgespielt, bis der Entwickler mit dem Modell soweit zufrieden ist, dass er es für eine machbare Lösung hält.
- Erst zum Schluss bringt der Entwickler seine Gedanken zum Ausdruck.

Vom Anforderungsprofil zum Modell

Modellieren bedeutet, das fertig gedachte Modell zum Ausdruck bringen zu können, bevor es zu Programmcode wird.



Erfolgsfaktor:

Erstellen Sie ein Modell der Realität.

In vielen Ingenieurwissenschaften ist es üblich Modelle zu entwickeln. Abbildung 3.1 zeigt als Beispiel den Originalplan des Eiffelturms in der frühen Phase der Planung.



Abbildung 3.1 Originalplan des Eiffelturms ca. zwei Jahre vor Baubeginn

Modelle haben folgende Vorteile:

- sie können sehr früh schon in der Planungsphase erstellt werden
- ein Modell zu verwerfen ist um vieles günstiger als eine erstellte Lösung ändern zu müssen
- Modelle veranschaulichen die Planung
- mit Modellen können Simulationen durchgeführt werden
- anhand eines Modells können Aufwände und Kosten abgeschätzt werden

- die Prüffälle, mit denen das Modell auf Tauglichkeit getestet wird, können später für das reale Ergebnis ebenfalls verwendet werden.

Wenn Sie modellieren können Sie also für recht geringe Kosten viele Vorteile erreichen. In unseren Projekten haben wir jedoch die Erfahrung gemacht, dass es stark darauf ankommt, wie modelliert wird und wie mit den Modellen umgegangen wird, ob diese Vorteile auch zum Tragen kommen. Folgende Punkte sind für den Erfolg entscheidend:

- Das Modell muss in beide Richtungen konkret sein - es muss eine hohe Übereinstimmung mit der Realität/den Anforderungen haben - es muss mit den für das Projekt vorgesehenen Mitteln realisierbar sein
- Das Modell muss entweder früh als solches an die Seite gelegt werden oder es muss durchgängig bis zum Projektende gepflegt werden⁴.

Ob Sie sich für Datenmodellierung, Modellierung nach der Strukturierten Analyse, Simulationsmodelle oder Modelle nach Ihren eigenen Vorstellungen entscheiden - in jedem Fall stehen Sie besser da, als wenn Sie überhaupt nicht modellieren. Unserer Ansicht nach hat den größten Nutzen jedoch ein durchgängig objektorientierter Ansatz.

3.2 Objektorientierter Ansatz

Welche drei Methoden der Einordnung, die unser Denken bestimmen wenden wir ständig an, um unsere Welt zu verstehen?⁵

1. bei der Erfassung unserer Beobachtungen trennen wir zwischen Objekten und ihren Eigenschaften – z.B. unterscheiden wir zwischen einem Baum und seiner Größe oder Lage im Verhältnis zu anderen Objekten,
2. wir unterscheiden zwischen ganzen Objekten und ihren Bestandteilen – z.B. wenn wir einen Baum im Gegensatz zu den Ästen betrachten, aus denen er besteht und
3. wir bilden und unterscheiden verschiedene Gruppen von Objekten indem wir sie klassifizieren – z.B. wenn wir die Klasse aller Bäume und die Klasse aller Steine bilden und diese unterscheiden.

„Objektorientierter Ansatz“ bedeutet lediglich auf diese Denkprinzipien aufzubauen und sie durchgängig anzuwenden und für alle Beteiligten sinnfällig zu machen. Damit sind die Beteiligten in einem Softwareprojekt in der Lage sich ein gemeinsames Modell der Wirklichkeit zu schaffen. Die zu erstellende Software basiert auf diesem Modell. Das Modell entspricht auf natürliche Weise den Denkstrukturen der beteiligten Menschen vom Auftraggeber bis

⁴ Bei generativen und round-trip Ansätzen (siehe 6.4 „Generierung“) ist diese Durchgängigkeit unabdingbar.

⁵ Nach [Coa90] - dort wird die Encyclopaedia Britannica „Classification Theory“ zitiert

zum Entwickler. Die einmal geprägten Begriffe für Klassen, Eigenschaften, Beziehungen zwischen Objekten bleiben dabei erhalten.



Erfolgsfaktor:

Verwenden Sie durchgängig den objektorientierten Ansatz.

Fr. Entrop stellt den Entwurf des objektorientierten Modells für die Projektverwaltung vor. Sie erläutert dabei die Klasse „Gesamtprojekt“ aller Projekte, die aus Teilprojekten bestehen. Die Eigenschaft „geplante Personentage“ hat sie dieser Klasse zugeordnet. Hr. Exner ist begeistert davon, dass heutzutage solch sprechende Namen benutzt werden können und das sogar Leerzeichen möglich sind. Frau Entrop erklärt, dass bei der Implementierung in Java später „geplantePersonentage“ als Feldname auftauchen wird. Hr. Exner erzählt, dass dieses Feld in der Altanwendung den Namen H47GMT01 hat.

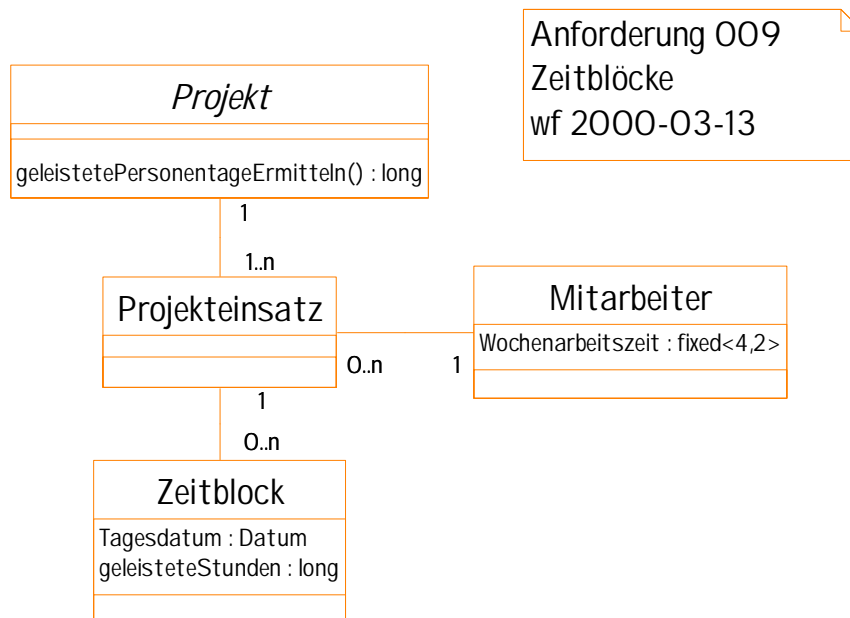


Abbildung 3.2 Sicht auf die Anforderung 009 - Zeitblöcke

Am leichtesten wird Ihnen die objektorientierte Modellierung fallen, wenn Sie die Objektorientierung als „natürliche Sicht“ auf die Realität betrachten. Versuchen Sie insbesondere nicht, mit mathematisch scharfen und exakten Ansätzen an die Objektorientierung heranzugehen. Wenn Sie das Gefühl haben, dass das Erstellen von Objektmodellen kinderleicht ist, dann liegen Sie richtig.

Im folgenden Abschnitt und dort insbesondere im Unterabschnitt „Die Welt als Bühne“ finden Sie Hinweise, wie Sie sich das Modellieren erleichtern können.

3.3 Ableitung des Modells aus den Anforderungen

Alle Analyseergebnisse basieren auf den Anforderungen - insbesondere soll nur modelliert werden, was gefordert ist. Für die Ableitung des Modells aus den Anforderungen gibt es keine strikten Regeln. Der Modellierer muss mit „Sinn und Verstand“ an die Arbeit gehen. Es ist jedoch möglich festzustellen, welche Analysergebnisse für welche Anforderungen benötigt werden. Das Modellieren entlang der Anforderungen und Prüfbeispiele ist zudem besonders leicht, weil sie dabei auf natürliche Weise modellieren und sich immer nur mit einem überschaubaren Teilaspekt des Problems beschäftigen.

Die Welt als Bühne - Objekte als Darsteller

Stellen Sie sich Objekte als handelnde Darsteller auf einer Bühne vor. Sie selbst übernehmen die Aufgaben für Drehbuch und Regie. Sobald Sie einem Darsteller einen Namen gegeben haben, sind sie in der Lage seine Eigenschaften zu beschreiben. Sie können festlegen zu welchen Handlungen er fähig ist und welche Kunststücke er beherrscht, was er weiss und wozu er aufgrund dieses Wissens fähig ist. Manchmal müssen Sie sich gedanklich in den Darsteller hineinversetzen, um entscheiden zu können, wie die Szene weitergehen kann. Lesen Sie sich nun aufmerksam eine Anforderung und die zugehörigen Prüfbeispiele durch – welche Darsteller können Sie entdecken? Fangen Sie mit der Situation an, stellen Sie Ihre Darsteller auf die Bühne. Betrachten wir dazu die Situation aus dem Beispieldokument in Abschnitt 2.3:

Die Projekte „Autoteile Internet-Shop“ und „Vertragsmanager“ sind beauftragt. Herr Friesen, Frau Meidrich, Hr. Sturm und Frau Sauer haben bisher folgende Aufwände geleistet ...

Um diese Situation auf Ihrer Bühne nachzustellen brauchen Sie sechs Darsteller: zwei der Sorte „Projekt“ und vier der Sorte „Mitarbeiter“. Die Projekte und Mitarbeiter sind einander wie in der Aufwandstabelle beschrieben zugeordnet. In der Situation sind die Namen der Projekte, die Namen der Mitarbeiter und die Zahlen für die geleisteten Aufwände beschrieben. Dieses Wissen gilt es nun den Darstellern sinnvoll zuzuordnen. Bei den Namen fällt die Entscheidung leicht: die Projekte „Autoteile Internet-Shop“ und „Vertragsmanager“ sollen ihren Namen kennen genauso wie jeder der Mitarbeiter seinen eigenen Namen wissen soll. In der UML⁶-Notation gibt es zwei Darstellungsmöglichkeiten für die Situation: Abbildung 3.3 zeigt eine passende Darstellung

⁶ Unified Modelling Language - eine standardisierte Form der Darstellung von objektorientierten Modellen

Vom Anforderungsprofil zum Modell

der Darsteller, Abbildung 3.4 zeigt eine Version die sich auf die Sorten von Darstellern beschränkt..

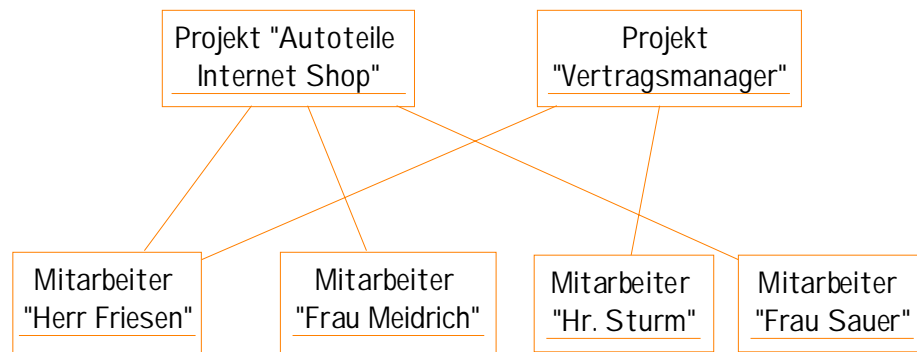


Abbildung 3.3: Darsteller für die vorliegende Situation⁷



Abbildung 3.4: Sorten von Darsteller für die vorliegende Situation⁸

Welchen der Darsteller sollen wir nun aber nach den Aufwänden für die Projekte fragen? Soll das Projekt jeweils wissen, wieviel Aufwand die Mitarbeiter die zu diesem Projekt gehören geleistet haben? Oder sollen die Mitarbeiter beantworten können, wieviel Aufwand sie für ein Projekt geleistet haben? Mit fünf weiteren Darsteller ist ein Mittelweg möglich: jede Projektbeteiligung wird als Bindeglied zwischen Projekt und Mitarbeiter in die Szene aufgenommen.

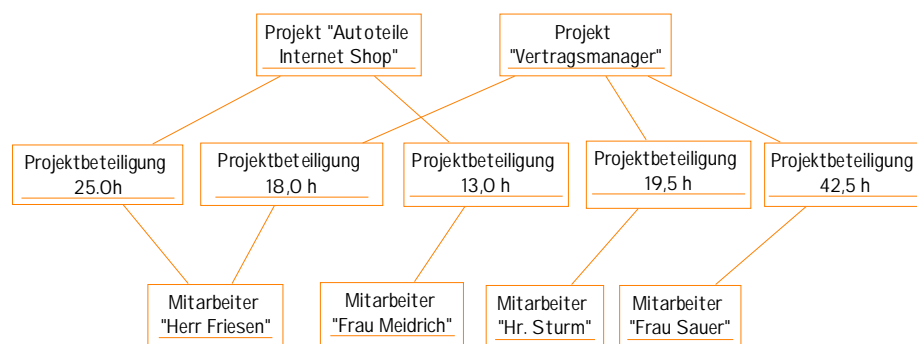


Abbildung 3.5: Die Projektbeteiligungen betreten die Bühne

⁷ Es handelt sich um ein „Kollaborationsdiagramm“. Unserer Erfahrung nach lohnt es sich nicht, die Kollaborationsdiagramme explizit zu zeichnen oder mit einem Werkzeug zu erfassen. Zur Veranschaulichung, wie die Angaben aus den Prüfbeispielen für die Modellierung genutzt werden können, ist ein solches Diagramm hier aber nützlich.

⁸ Ein Klassendiagramm

Abbildung 3.5 und Abbildung 3.6 geben die Situation jetzt vollständig wieder. Indem Sie die verschiedenen Sorten von Darsteller klassifizieren und ihre gemeinsamen Eigenschaften beschreiben, leisten Sie den wesentlichen Schritt der Modellierung - Sie abstrahieren.

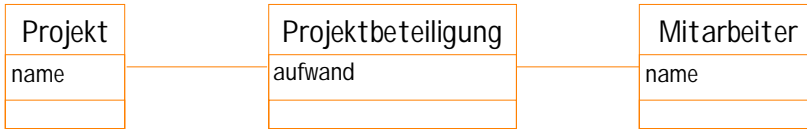


Abbildung 3.6: Charakterbild und Beziehungen der Darsteller

Indem Sie die Situation auf verschiedene Charaktere verteilen und nicht einfach nur eine Person „Situation“ auf die Bühne stellen, strukturieren Sie und Sie verteilen Verantwortung. Stellen Sie sich die Welt einfach als Bühne vor und diese Schritte werden Ihnen als selbstverständlich vorkommen. Zudem erreichen Sie dabei, dass die Strukturen, die Sie bilden den Strukturen ihres Fachproblems sehr nahe kommen. Die Aussagen aus den Prüfbeispielen stammen ja direkt von den Fachleuten und geben die Denkstrukturen der von Ihnen befragten Fachleute wieder. So kommen Sie auch nicht in Versuchung in technischen Strukturen zu denken, denn die Anforderungssteller werden normalerweise auch nicht in technischen Strukturen denken. Die Begriffe aus der Anforderungsaufnahme fließen so zudem ziemlich unverändert in Ihr Modell ein und Sie sprechen damit automatisch die Sprache Ihres Auftraggebers.

Vervollständigen wir nun das Modell, bis es das Prüfbeispiel ganz abdeckt. Betrachten wir als nächstes die Aktion:

Aktion: Die Projekt- und Mitarbeiterangaben sowie die Aufwände werden erfasst und die Auswertung „Gesamtaufwand je Projekt“ angestossen.

Nun soll Bewegung in die Szene kommen. Wie können die Darsteller durch ihr Zusammenspiel die gewünschte Aktion durchführen? Wer dafür welche Fähigkeiten haben? Wem wird die Verantwortung übertragen? Eine sinnvolle Lösung scheint zu sein, für Projekte, Mitarbeiter und Projektbeteiligungen jeweils einen Verwalter als Darsteller auftreten zu lassen. Die Verwalter übernehmen die Aufgabe der Erfassung. Zudem übernimmt der Projektverwalter die Aufgabe die Auswertung „Gesamtaufwand je Projekt“ durchführen zu können. Damit die Arbeit leichter wird, fragt er dazu jedes Projekt nach seinem Gesamtaufwand. Die folgenden Abbildungen geben das Ergebnis dieser Überlegungen wieder:

Vom Anforderungsprofil zum Modell

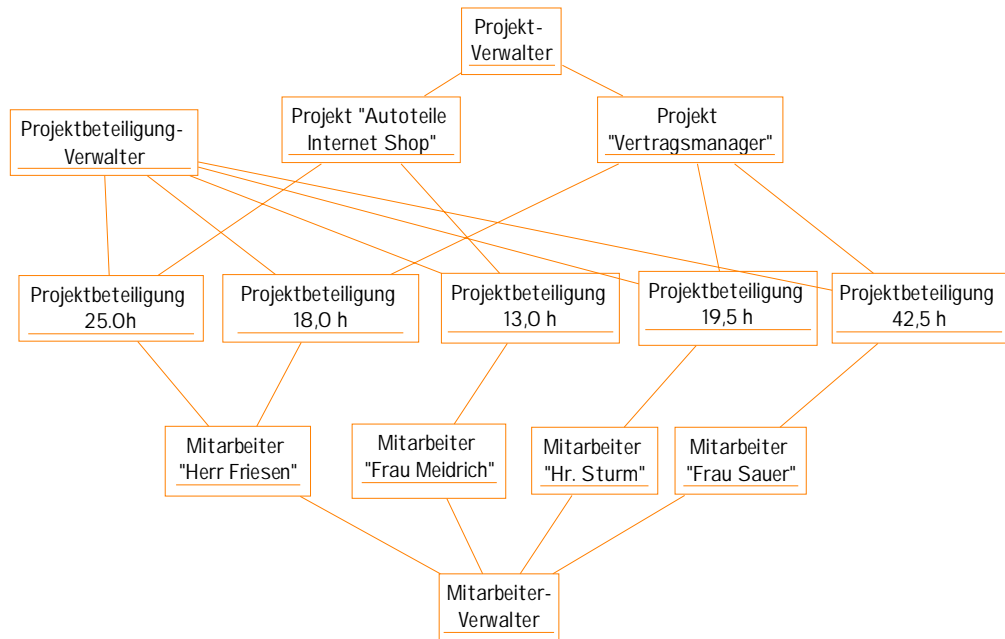


Abbildung 3.7: Die Verwalter betreten die Szene und treten in Aktion

In diesem Schritt haben Sie bereits einen Algorithmus konzipieren müssen. Sie haben sich überlegt, wie Projektverwalter und Projekt zusammenarbeiten, um die Auswertung „Gesamtaufwand je Projekt auswerten“ zu erreichen. Dazu haben Sie sich in die Rolle des Projektverwalters versetzt und sich überlegt „Wie würde ich das Problem lösen, diese Auswertung zu bekommen?“

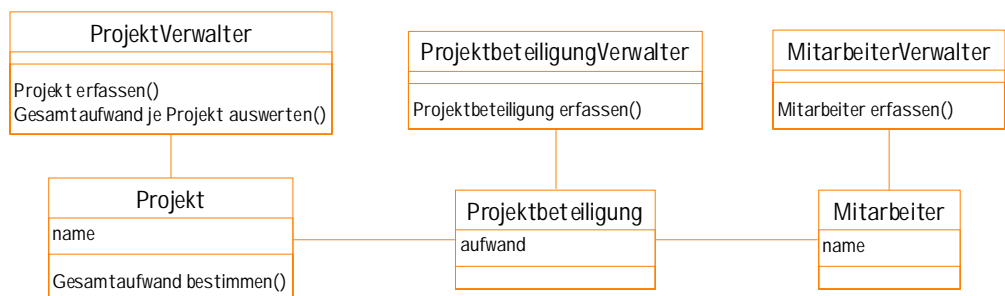


Abbildung 3.8: Einbeziehung der Aktion in das Modell

Eine mögliche Antwort in „ich-Form“ lautet „Ich bitte alle von mir verwalteten Projekte, Ihren Gesamtaufwand zu bestimmen und stelle anhand der Ergebnisse dann eine sortierte Liste der Gesamtaufwände je Projekt zusammen“. Dies ist bereits eine gute Dokumentation für „Gesamtaufwand je Projekt auswerten.“

Zum guten Schluss geht es darum, dass Modell auf Richtigkeit zu prüfen:

Erwartetes Ergebnis:

Autoteile Internet Shop: 56 h, Vertragsmanager: 62 h.

Wenn Sie der Überzeugung sind, dass das oben erstellte Modell dieses Ergebnis liefern kann, dann haben Sie jetzt gerade den Teil der Lösung modelliert, der dafür sorgt, dass unser Prüfbeispiel funktioniert. Da wir dabei bereits vom Prüfbeispiel abstrahiert haben, können wir prüfen, ob die Anforderung mit diesem Modellausschnitt grundsätzlich erfüllt wird. Wir können offene Fragen stellen oder weitere Prüfbeispiele hinzuziehen. Insbesondere erfolgte die Modellierung auf dem natürlichen Weg, denn wir zu Anfang dieses Kapitels im Abschnitt 3.1 beschrieben haben. Wenn Sie mit dem Modell noch nicht zufrieden sind, werden Sie demnach jetzt beginnen es zu verändern und erneut zu prüfen.

Bis jetzt haben wir beim Modellieren so gut es geht vermieden die Fachbegriffe der objektorientierten Analyse, wie Klasse, Objekt, Instanz, Operation, Attribut oder Beziehung zu verwenden. Wir glauben, dass diese Begriffe das Erlernen der Modellierung und die richtige Anwendung unnötig erschweren, denn diese Begriffe sind künstlich von Informatikern geprägt worden und gehören nicht zur Alltagssprache.

Nach unserer Erfahrung besteht die größte Hürde für Neulinge in der objektorientierten Modellierung darin, die Darstellung von einzelnen Darsteller und den Verwaltern dieser Darsteller in einem einzigen Symbol zusammenzufassen:

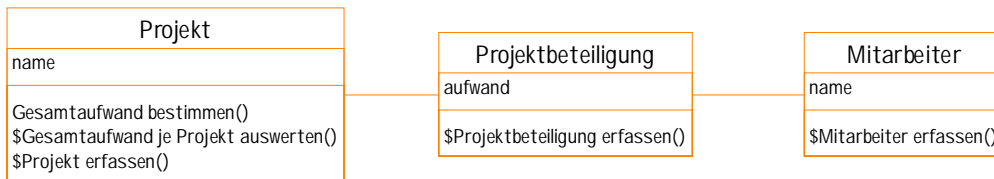


Abbildung 3.9: Weglassen der Verwalter

Verhalten und Wissen der Verwalter sind jetzt unter der Sortenbezeichnung der Darsteller zu finden zur Abgrenzung wird ein Symbol verwendet (hier ein „,\$“). Schliesslich ist es üblich, die Erfassungsmöglichkeit für Darsteller nicht explizit zu erwähnen:

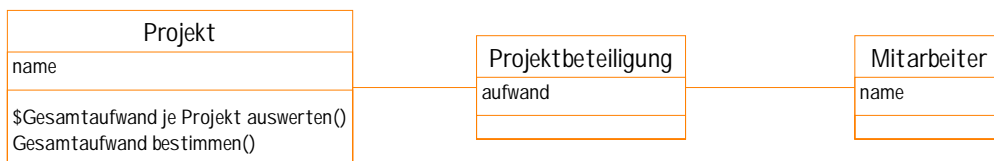


Abbildung 3.10: Weglassen der Erfassungsmöglichkeit

Abbildung 3.10 gibt nun das gesamte Prüfbeispiel in UML-Notation so wieder, dass damit das Prüfbeispiel zumindestens im Gedankenspiel richtig durchgespielt werden kann. Dies ist eine stabile Grundlage für die weitere Modellierung.

Sichtenbildung

Die konsequente Anwendung der Modellierung entlang von Anforderungen und Prüfbeispielen ermöglicht es, immer überschaubare Teile des Modells zu erstellen. Dazu werden Sichten verwendet. Um die Konsistenz der Sichten zu erhalten empfehlen wir dringend, ein Modellierungswerkzeug zu verwenden (siehe Abschnitt 6.2 „Werkzeugauswahl“). Die Sichten bieten für den Modellierer jeweils einen Ausschnitt zur Bearbeitung an, während das Modellierungswerkzeug im Hintergrund die volle Komplexität des Gesamtmodells berücksichtigt und auf Unstimmigkeiten automatisch aufmerksam macht. Abbildung 3.11 zeigt prinzipiell wie eine Sicht eine einzelne Anforderung umfasst, Abbildung 3.2 zeigt eine konkrete Sicht aus dem Projektverwaltungsbeispiel.

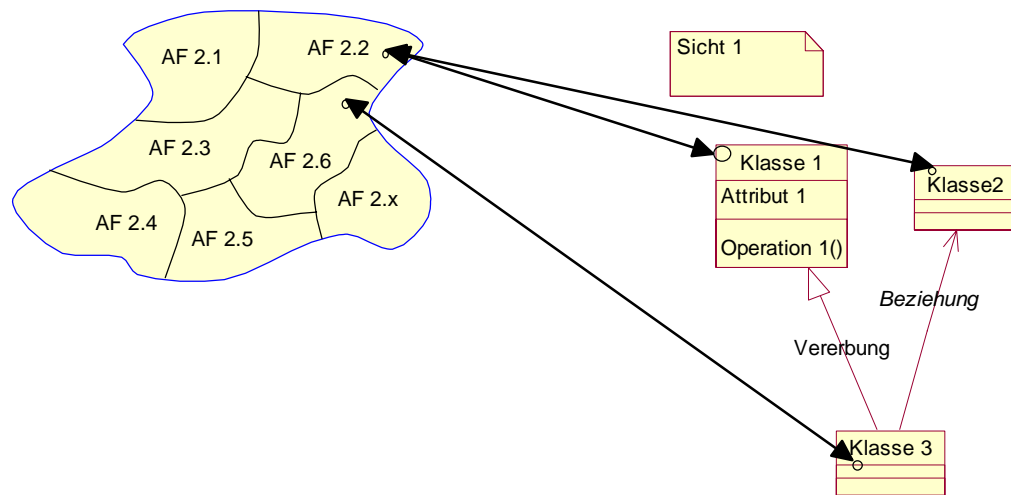


Abbildung 3.11: Modellierung von Anforderungen in Sichten

Durch das Modellieren von Anforderungen Sicht für Sicht gelingt es, schrittweise das gesamte Anforderungsprofil als Modell zu erfassen. Das systematische Vorgehen stellt sicher, dass keine Themen, Anforderungen und Prüfbeispiele vergessen werden und sich alle Beteiligten im Modell zu Recht finden. Abbildung 3.12 zeigt wie schliesslich alle Themen des Anforderungsprofils durch das Gesamtmodell abgedeckt werden. Das Modell erfüllt in diesem Zustand prinzipiell alle gestellten Anforderungen. Es muss nun noch der Nachweis erbracht werden, dass die aus dem Modell abgeleitete Realisierung ebenfalls diese Eigenschaft hat.

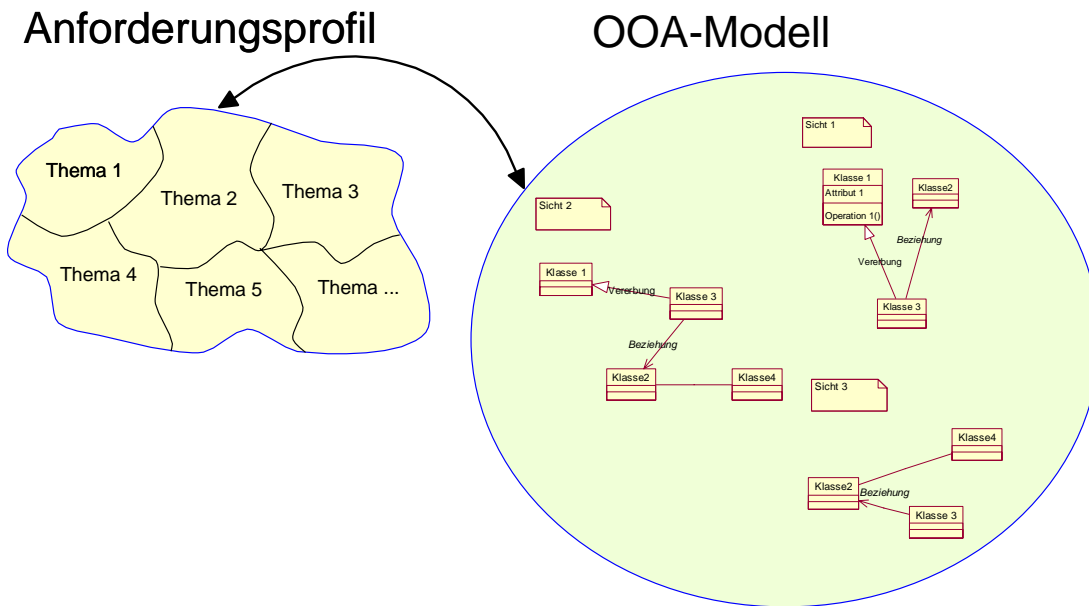


Abbildung 3.12: Schrittweise Erfüllung aller Anforderungen im Modell

3.4 Einfach, kurz und klar - komplizierte Modelle sind falsch!

In [Roe94] „Einfach, kurz, klar - komplizierte OOA Modelle sind falsch!“ beschreibt Martin Rösch wie leicht es in Softwareprojekten passieren kann, dass Modelle entstehen, die unverständlich sind. Der Inhalt der Modelle mag dabei durchaus richtig sein - die Unverständlichkeit an sich stellt bereits ein unakzeptables Hindernis dar. Die Negativen Folgen der Unverständlichkeit entwickeln sich dadurch, dass den Auftraggebern mit zunehmender Unverständlichkeit die Möglichkeit genommen wird, die Modelle auf ihre Richtigkeit hin zu prüfen. David Parnas beschreibt in [Par91] wie bei der Festlegung von Anforderungen für die sicherheitskritische Software eines Atomkraftwerkes auf formale Methoden gesetzt wurde. Die formale Vorgehensweise sollte die Beweisbarkeit der Sicherheit der Software erreichen. Tatsächlich konnte nachgewiesen werden, dass die Software der formalen Spezifikation entspricht. Die formale Spezifikation war jedoch in mathematischer Sprache verfasst. Das Prüfteam hat diese Spezifikation abgesehen, ohne die Spezifikation zu lesen - denn nach einem kurzen Blick darauf sahen die Prüfer, dass es sich um für sie unverständliche Formeln handelte. Sie vertrauten daraufhin „blind“ auf die Richtigkeit. Parnas zeigte anschliessend auf, dass die Spezifikation voller Widersprüche war.



Erfolgsfaktor:

Achten Sie auf die Verständlichkeit Ihrer Spezifikationen und Modelle.

Die drei Psychologen Langer, Schulz v. Thun und Tausch haben die Verständlichkeit von Texten wissenschaftlich untersucht [Lan99]. Sie fassten alle wesentlichen Texteigenschaften zusammen und gruppierten sie. Diese werden als:

Merkmale der Verständlichkeit

- Einfachheit
- Kürze - Prägnanz
- Gliederung - Ordnung - Klarheit
- Anregende Zusätze

bezeichnet. Diese Merkmale der Verständlichkeit lassen sich auf die Modellierung vollständig anwenden.

- Einfachheit

Einfachheit bezieht sich auf die Form, nicht den Inhalt eines Modells und der beschreibenden Texte. Ein einfacher Sachverhalt kann kompliziert dargestellt werden, genauso wie es gelingen kann, einen komplizierten Sachverhalt einfach darzustellen.

Die folgende Tabelle zeigt was Einfachheit im Gegensatz zur Kompliziertheit ausmacht:

Einfachheit	Kompliziertheit
einfache Darstellung	komplizierte Darstellung
kurze, einfache Sätze	lange, verschachtelte Sätze
bekannte Wörter	unbekannte Wörter
unbekannte Fachbegriffe erklärt	unbekannte Fachbegriffe nicht erklärt
konkret	abstrakt
anschaulich	unanschaulich

- Kürze - Prägnanz

Stimmt das Verhältnis von Inhalt und Aufwand der Darstellung? Klarheit bedeutet das Wesentliche mit geringen Mitteln zu modellieren. Eine zu knappe Darstellung behindert das Verständnis. Eine weitscheifige mit Ausschmückungen versehene Form lenkt vom eigentlich wichtigen Inhalt ab.

Kürze - Prägnanz	Weitschweifigkeit
(zu) wenig Inhalt	(zu) viel Inhalt
beschränkt auf das Notwendige	viel Unnötiges
gedrängt	breit
aufs Ziel konzentriert	abschweifend, ausschmückend

Kürze - Prägnanz	Weitschweifigkeit
knapp	ausführlich

■ Gliederung - Ordnung - Klarheit

Eine gute Gliederung eines Modells erleichtert das Verstehen und Merken des Inhaltes. Insbesondere ist es auch leichter, nach einem bestimmten Sachverhalt zu suchen. Entscheidend für die Verständlichkeit der Gliederung ist der logische Aufbau des Modells. Die Informationen werden in sinnvollen Zusammenhängen präsentiert. Durch Hervorheben und Zusammenfassen wird Wichtiges deutlich gemacht.

Gliederung - Ordnung - Klarheit	Zusammenhanglosigkeit
Gliederung in Sichten	ungegliederte Gesamtsicht
Gliederung in Gruppen	ungegliederter Haufen von Information
folgerichtig	zusammenhanglos, wirr
gute Unterscheidung von Wesentlichem und Unwesentlichem	schlechte Unterscheidung von Wesentlichem und Unwesentlichem
der rote Faden bleibt sichtbar	man verliert oft den roten Faden
alles erscheint im richtigen Zusammenhang	alles geht durcheinander

■ Anregende Zusätze

In der nüchternen Welt der Computerexperten fehlt dieses Merkmal der Verständlichkeit oft. Es dient dazu den Betrachter zu animieren und anzuregen, damit der Spass an der Sache erhalten bleibt. Als anregende Zusätze eignen sich die persönliche Ansprache des Betrachters, Ausrufe, praktische Beispiele, Fragen zum Mitdenken, humoristische Einlagen, Bilder, Animationen usw.

Anregende Zusätze	ohne anregende Zusätze
anregend	nüchtern
interessant, bunt	farblos, grau
abwechslungsreich	gleichbleibend neutral
persönlich	unpersönlich
bewegt	statisch

3.5 Die Abstraktionsfalle

Schon 1997 schrieb Brooks in [Bro87]: „Seit fast 40 Jahren hat man die ‚automatische Programmierung‘ oder die Generierung eines Programms zur Lösung eines Problems aus der Beschreibung der Problemspezifikation vorhergesehen oder darüber geschrieben ... kurz gesagt, automatische Programmierung war immer eine Umschreibung für dass Programmieren mit einer Sprache auf höherem Niveau als die bisher für den Programmierer verfügbare.“

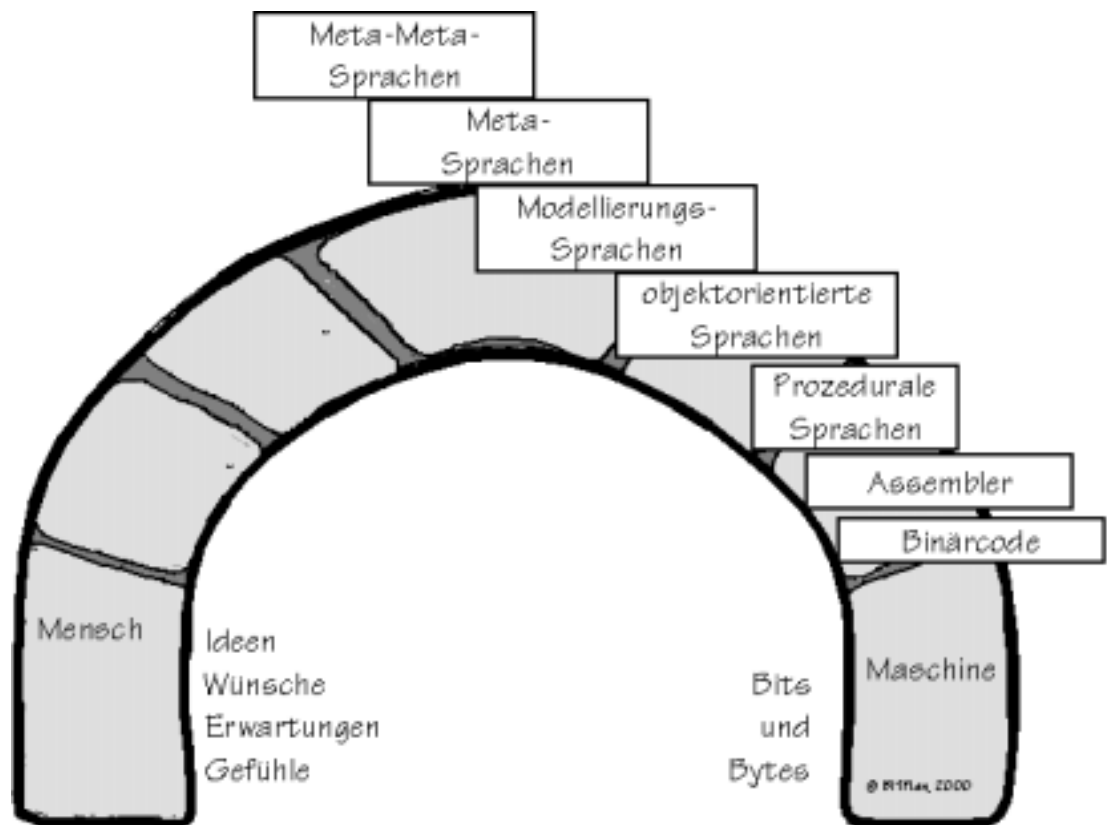


Abbildung 3.13 Die Abstraktionsfalle

Tatsächlich sind die größten Fortschritte der Softwareentwicklung der Weiterentwicklung der Ausdrucksmöglichkeiten auf der rechten Seite der Brücke zu verdanken. Es wird scheinbar immer einfacher, dem Computer mitzuteilen, was er leisten soll. Die Vereinfachung ging dabei in der Vergangenheit immer mit einer Erhöhung des Abstraktionsgrades einher:

- **Binärcode**
Das von-Neumann Prinzip ermöglichte es erstmals binären Programmcode wie Daten zu betrachten. Dies war die erste Abstraktion von der Maschine - Programme wurden nicht mehr durch Schaltereinstellungen oder festverdrahtete Schaltungen erzeugt, sondern wie Daten in die Maschine geladen. Dies erhöhte die Flexibilität von Programmen bereits erheblich.
- **Assembler**
Mit einer Textdarstellung von Maschinenbefehlen unter Nutzung von Kürzeln wie ADD, MOV und JMP wurde es möglich von der konkreten Maschine zu abstrahieren. Ein einmal in Assembler geschriebenes Programm konnte von nun an auf unterschiedlichen Maschinen (ähnlichen Typs) erneut in Binärcode umgesetzt werden⁹.

⁹ IBM lernte schmerzlich, wie wichtig es ist, dass Assemblercode auf neuen Maschinen lauffähig bleibt. Heute leisten weltweit noch Milliarden von Assemblercodezeilen aktive Dienste, die über 30 Jahre alt sind.

- **Prozedurale Sprachen**
Die Idee der ersten prozeduralen Programmiersprachen war es, möglichst nahe an die geschriebene „natürliche Sprache“ heranzukommen. Satzfolgen wie „COMPUTE DISCOUNT = DISCOUNT/100. DIVIDE 100 INTO PRICE. COMPUTE SALES = PRICE * QTY - (PRICE * QTY) * DISCOUNT. ADD SALES TO YTD-SALES.“ konnten von nun an zur Programmierung herangezogen werden.
- **Objektorientierte Sprachen**
Von nun an wird die Zusammenfassung von Daten und Funktionen zu Objekten die Verhalten zeigen und Wissen haben ermöglicht. Die Objekte können sich gegenseitig Nachrichten senden und so kann durch Zusammenarbeit von Objekten ein Problem gelöst werden.
- **Modellierungssprachen**
Die Unified Modelling Language (UML) ist ein Beispiel für die Abstraktion von Objektorientierten Sprachen. Über die üblichen objektorientierten Sprachkonstrukte Kapselung, Vererbung, Abstraktion, Polymorphismus, und Klassifizierung hinaus werden weitere Konstrukte angeboten. Dazu gehört die Modellierung von Beziehungen sowie die Beschreibung von dynamischem Verhalten¹⁰.
- **Meta-Sprachen**
Um noch weiter zu abstrahieren muss die nächst höhere Ebene zur Abstraktion genutzt werden - die Metaebene wird betreten. Dies ist zum Beispiel erforderlich, wenn Regelwerke umgesetzt werden sollen, die sich auf beliebige Objekte beziehen sollen. Nun ist es nur noch möglich mit einer Metasprache das Regelwerk zu beschreiben, denn die Objekte die später in den Regeln vorkommen sollen, sind ja noch gar nicht bekannt.
- **Meta-Meta-Sprachen**
Wenn eine Meta-Sprachen Lösung noch nicht einfach genug ist, dann muss halt eine Meta-Meta-Sprache her usw.

Und so besteht heute der Irrglaube, die weitere Erhöhung des Abstraktionsgrades würde es einfacher machen, richtige Software zu erstellen. Abbildung 3.13 **Die Abstraktionsfalle** verdeutlicht jedoch worin der Irrglaube besteht. Da die Abstraktion von der Seite der Maschine aus gedacht ist, wird es zwar möglich immer abstrakter zu spezifizieren, was die Maschine leisten soll. Schliesslich entfernt sich diese Beschreibung aber immer mehr von dem wie Menschen denken und fühlen, wenn lediglich weitere Abstraktion die Vereinfachung vorantreibt. Um entlang der Brücke in Richtung auf den Menschen „abzubiegen“ ist etwas anderes erforderlich. Dafür ist es zum Beispiel wichtiger, Widersprüchlichkeiten aufzulösen, Entscheidungen voranzutreiben und die Anpassbarkeit an die Bedürfnisse einzelner Anwender zu ermöglichen. Abstraktion verhindert jedoch derartige Vielfalt und Uneindeutigkeit eher.

¹⁰ Der Nachteil von UML besteht darin, dass es keine verbindliche Umsetzung von diesem abstrakten Niveau auf die objektorientierten Programmiersprachen gibt.

3.6 Literatur

- [Boo94] G. Booch: **Object-Oriented Analysis and Design with Applications**, Addison-Wesley, 1994
Grady Booch ist einer der „drei Amigos“ die gemeinsam bei Rational Software die UML entwickelt haben. Sein Buch ist einer der Klassiker zum Thema.
- [Bro87] F. P. Brooks, F.P.: **No Silver Bullet; Essence and Accidents of Software Engineering**, Computer, April 1987.
- [Bro02] M. Broy, J. Siedersleben: **Objektorientierte Programmierung und Softwareentwicklung - eine kritische Einschätzung**, Informatik Spektrum, Band 25, Heft 1/2002, pp. 3-11
Dieser kritische Artikel zeigt die Grenzen des Ansatzes der Objektorientierung auf und warnt vor dem Irrglauben mit der Objektorientierung sei der Weisheit letzter Schluss gefunden.
- [Brö00] P. Brössler, J. Siedersleben: **Softwaretechnik - Praxiswissen für Software-Ingenieure**, Hanser, 2000
Sammlung von Aufsätzen von Mitarbeitern der Firma sd&m zu praxisrelevanten Themen des Softwareengineering.
- [Coa90] P. Coad, E. Yourdon: **Object Oriented Analysis**, Yourdon Press, 1990
Coad und Yourdon sind Pioniere auf dem Gebiet der Objektorientierung. Sie sind in der Lage mit einfachen Mitteln zu erklären, was Objektorientierung nutzt und wie Analyse auf dieser Basis funktioniert.
- [Gla95] R. L. Glass: **Software Creativity**, Pearson Professional Education, 1995
- [Jac92] I. Jacobson: **Object-Oriented Software Engineering - A Use Case Driven Approach**, acmPress/Addison-Wesley, 1992
Ivar Jacobson bringt als zweiter Amigo die Anwendungsfälle (Use Cases) als Gliederungsmittel in die Modellierung von objektorientierten Systemen ein.
- [Jäh02] S. Jähnichen, S. Herrmann: **Was, bitte, bedeutet Objektorientierung?**, Informatik Spektrum, Band 25, Heft 4/ 2002, pp. 266-276
Gegenrede zum Artikel von Broy und Siedersleben [Bro02]
- [Lan99] I. Langer, F. Schulz von Thun, R. Tausch: **Sich verständlich ausdrücken**, Reinhardt, 1999
Verständlichkeit der zur Dokumentation von Modellen verwendeten Sprache ist Grundvoraussetzung für die Verständlichkeit von Mo-

dellen - dieses Buch zeigt die Ergebnisse einer wissenschaftlichen Untersuchung zum Thema Verständlichkeit – natürlich besonders verständlich – auf.

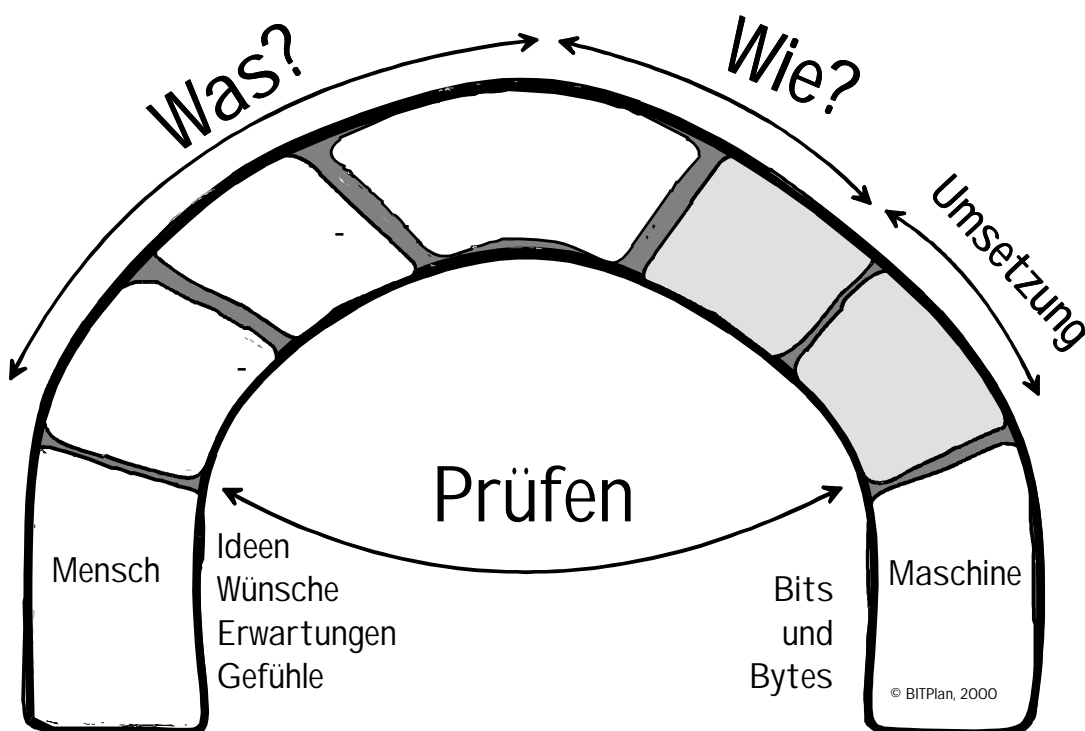
- [Oes98] B. Oestereich: **Objektorientierte Softwareentwicklung**, Oldenburg, 1998
Standardwerk für die Einführung in die Objektorientierung. Besonders als Einstieg in das Thema geeignet.
- [Roe94] M. Rösch: **Einfach, kurz, klar - komplizierte OOA-Modelle sind falsch!** in Objektspektrum 04/94, SIGS Datacom, 1994
Dieser Artikel aus dem Jahr 1994 hat schon früh das Dilemma der Verständlichkeit von OOA-Modellen aufgezeigt.
- [Rum91] J. Rumbaugh, W. Premerlani, W. Blaha: **Object-Oriented Modeling and Design**, Prentice Hall, 1991
Rumbaugh ist der dritte Amigo und dieses Buch sein Beitrag aus der Zeit vor UML.
- [Roe94b] M. Rösch: **ISO 9000: Qualität oder Zertifikat?** in Computerwoche 51/52, 1994

„Irren ist menschlich. Aber wenn man richtig Mist bauen will, braucht man einen Computer.“

Dan Rather, amerikanischer Fernsehreporter

4

Vom Modell zum Code



4.1 Umsetzen

Bei der Umsetzung des Modells in Code ist die wichtigste Entscheidung, wieviel Sie selber machen wollen und was sie als fertige Lösung dazukaufen.

Selber machen oder kaufen?

Was sind die Entscheidungskriterien für selber machen oder kaufen? Wo bekomme ich die Software her, die genau meinen Wünschen entspricht?

Der nächste Erfolgsfaktor ist ein Angriff auf den heiligen Gral so mancher Softwareentwickler. „Selber machen“ ist über Jahrzehnte die beliebteste Antwort gewesen. Wenn Softwareentwickler mit einer Lösung konfrontiert werden, die sie nicht selbst entwickelt haben, dann greift der Virus des „Not invented here Syndrom“ um sich. Rein wirtschaftlich betrachtet ist selber entwickeln inzwischen in den meisten Fällen die teuerste Variante. Manches Softwareunternehmen ist durch diesen Ansatz in den Ruin getrieben worden.

Die Erkenntnis, dass Kaufen sinnvoller ist, setzt sich durch – auf dem ersten Platz unserer Umfrage ist der folgende Erfolgsfaktor zu finden:



Erfolgsfaktor:

Der Einsatz von Frameworks, Komponenten und Klassenbibliotheken verkürzt ihre Entwicklungszeit und macht Ihre Lösung zukunftssicherer.

Im Gegensatz zur vollständigen Eigenentwicklung ist es heute möglich mit vorgefertigten Halbfabrikaten nach dem Baukastenprinzip eine Software zusammenzusetzen und zu erweitern. Die einsetzbaren Halbfabrikate sind:

- Frameworks
Eine Sammlung von kooperierenden Klassen die für das spezielle Problem angepasst und erweitert werden müssen. Das Framework gibt eine leer Hülle vor, die aber in sich bereits stabil ist. Ein Framework dient als Basis für eine Anwendung.
- Komponenten
Fertige Teillösungen die in die eigene Lösung eingebunden werden können und ebenfalls anpassbar oder erweiterbar sind. Fachliche Komponenten liefern Bausteine für ihr Fachgebiet. Technische Komponenten sorgen dafür, dass ihre Lösung in der vorgesehenen Computerumgebung lauffähig ist.
- Klassenbibliotheken
Allgemeine Sammlung von Basisklassen, insbesondere für die Lösung von technischen Standardprogrammen. Ein sehr bekanntes und beliebtes Beispiel ist die zum Java Development Kit (JDK) gehörende Klassenbibliothek.

Für den Einsatz solcher Zukaufteile spricht die schnelle Verfügbarkeit und die hohe Qualität der Komponenten. Zudem werden die Komponenten vom Anbieter weiterentwickelt und dieser ist Spezialist für das Thema. Wie stark diese Vorteile wiegen hängt von der Wahl des Anbieters, dem Grad der Standardisierung der Technologie und der Durchsetzung am Markt ab.

Die Anbieter von Frameworks, Komponenten und Klassenbibliotheken haben erkannt wie wichtig es ist, die eigene Lösung als Standard zu positionieren und den Käufern klar zu machen, dass diese Lösung über Jahre verfügbar sein wird und gepflegt wird. Die grossen IT-Firmen Microsoft, IBM, Sun, Oracle usw. führen dazu einen Balanceakt zwischen der Verfolgung eigener Interessen im Konkurrenzkampf und dem Überzeugen der Käufer bzgl. dieser Punkte auf.

In unseren Projekten haben wir mit Komponenten schon so einige Überraschungen erlebt - aber schliesslich stellte sich dennoch der Vorteil von Komponenten ein:

- In der Klasse `java.sql.ResultSet` für den Zugriff auf SQL-Datenbanken von Java gibt es die Methoden `moveToInsertRow` und `insertRow()`, mit denen neue Einträge in eine Tabelle geschrieben werden können. Diese Methoden funktionierten mit den standardmässig ausgelieferten JDBC/ODBC-Bridge Klassen nicht. Die Lösung brachte die gleiche Komponente von einem Drittanbieter.
- Eine Komponente zum versenden von Massenemails funktionierte plötzlich nicht mehr mit der neuesten Version der Datenbank zusammen. Eine neue Version des Herstellers stand aber bereits zur Verfügung - diese hatte zu dem noch neue nützliche Funktionen.

Hr. Angelet und Hr. Teschner stimmen sich über die XML-Schnittstelle für das Projektverwaltungssystem ab. Hr. Teschner schlägt vor, einen XML-Parser selbst zu schreiben, da ja nur ein Bruchteil der Funktionalität eines XML-Parsers gebraucht würde und dies sicherlich schneller ginge, als sich erst in die XML-Technologie einzuarbeiten. Hr. Angelet berichtet von seinen guten Erfahrungen mit den XML-Komponenten Xalan und Xerces in einem anderen Projekt - dort war es mit nur wenigen Zeilen Java Code gelungen die Java-Objekte aus einer XML-Datei heraus zu lesen und wieder zu schreiben. Als weiteres Argument führt Hr. Teschner an, dass in den neueren JDK-Versionen die XML-Unterstützung bereits enthalten ist und somit zum Java-Standard gehört. Er überzeugt Hr. Teschner schliesslich, möglichst wenig selbst zu entwickeln.

Softwarekomponenten sind am leichtesten im Internet zu finden. Die Suche über Suchmaschinen ist dabei immer noch effektiver als bei speziellen Komponentenbörsen nachzuschlagen. Bei einigen Komponentenanbietern und Standardisierungsgremien hat es sich bereits eingebürgert, dass die Prüfbeispiele bereits mitgeliefert werden. Dies ist sehr zu begrüssen und zum Teil bitter nötig, um Inkompatibilitäten zu vermeiden. Die J2EE Spezifikation enthält zum Beispiel eine „conformance suite“, die diesem Anspruch gerecht wird.

Teile und Herrsche

Das Prinzip der Modularisierung „teile und herrsche“ wurde in der Informatik schon früh eingesetzt, um komplexe Systeme erstellen zu können. Module kapseln Daten oder Verhalten und sind damit Vorläufer der Objekte. 1971 führte David Parnas zusätzlich das Geheimnisprinzip „Information Hiding“ ein. Er zeigte auf, dass zwischen den Modulen zu oft unsichtbare Abhängigkeiten bestehen, wenn nicht sichergestellt wird, dass die „Innereien“ eines Moduls verborgen werden. Daher wird zwischen der Spezifikation eines Moduls – seiner Schnittstelle und der Realisierung des Moduls unterschieden. Ein Modul kann somit durch ein anderes Modul mit gleicher Schnittstelle ersetzt werden.

Das Prinzip der Modularisierung erreichte in unserer Umfrage den 6. ten Platz:



Erfolgsfaktor:

Zerlegen Sie Problem und Lösung schrittweise in kleinere von einander möglichst unabhängige Teile.

Wir empfehlen Ihnen dringend, nicht auf Modularisierung nur deswegen zu verzichten, weil dieses Prinzip schon lange bekannt und damit „ein alter Hut“ ist.

Die richtige Modularisierung zu erreichen bedeutet nicht nur das Problem zu zerlegen, sondern auch eine Zerlegung zu finden, die möglichst optimal ist. Diese Aufgabe ist gar nicht so leicht:

- Wenn Sie wenig Module bilden können Sie die Vorteile der Modularisierung nicht genießen, die einzelnen Probleme bleiben zu groß, die Bindung in den Modulen ist zu gering.
- Wenn Sie viele Module bilden, bekommen Sie sehr viele Schnittstellen - der Kommunikationsaufwand zwischen den Modulen und die Pflege der Module wird aufwendig, die Kopplung zwischen den Modulen ist zu groß.

Regeln für die richtige Balance zwischen Bindung (cohesion) und Kopplung (coupling) von Modulen zu finden ist eine Wissenschaft für sich [Dar02]. Module sollen möglichst unabhängig voneinander sein und dennoch die Aufgaben untereinander aufteilen.

Unserer Erfahrung nach sind vor allem die Unterschiede in den Teams bzgl. der Neigung zur Modularisierung sehr hoch. Einige Mitarbeiter zerlegen Probleme in kleinste Schnipsel, bevor diese gelöst werden, andere bauen monströs große Funktionen, die über riesige Parameterlisten gesteuert werden müssen. Diesen Unterschieden lässt sich am Besten entgegenwirken, wenn die Ergebnisse gemeinsam erstellt werden bzw. die Bearbeitung rotiert.

Mit dem Einsatz von Werkzeugen zur Komplexitätsmessung haben wir keine Erfahrungen gemacht - es ist jedoch schade, dass in den heutigen integrierten Entwicklungsumgebungen diese Funktion nicht als Standard enthalten ist - dann würde sie sicherlich eher ihren Nutzen entfalten.

Trennung von Fachlichkeit, Design und Architektur

„Der Alptraum jeden Programmierers“ so wird in [Sie00] Software beschrieben, in der sich technische Probleme mit Anwendungsproblemen vermischen.

Die Trennung der Lösung von fachlichen und technischen Themen erlaubt es die Lösungen unabhängig voneinander wiederzuverwenden. Lösungen von fachliche Themen werden häufig in unterschiedlichen technischen Zusammenhängen benötigt. Ein Beispiel dafür ist die „Vertriebskanalunabhängigkeit“. Egal welche technische Infrastruktur ein Unternehmenspartner nutzt, ob Arbeitsplatzcomputer, Internet, Handy oder einen speziellen Automaten - die gleiche fachliche Funktion soll bereitstehen. Gelingt es nicht, auf die gleiche Lösung zurückzugreifen, entstehen oft Unstimmigkeiten - der Fahrkartenautomat nennt einen anderen Preis als die Auskunft - das Internet nennt einen Dritten - der Kunde ist verwirrt. Das Kreditangebot des Sachbearbeiters stimmt nicht mit den später zur Unterschrift vorliegenden Vertragsunterlagen überein - und dies wieder nicht mit dem Ergebnis des Kreditrechners auf der Werbe-CD der Bank.

Weitere Motivation für die Trennung von Fachlichkeit und Technik ist das unterschiedliche Tempo mit dem sich die Anforderungen an beide Bereiche ändern. Technische Themen, wie Datenhaltung, Netzwerk, Darstellung, Betriebssystem usw. ändern sich im Abstand von ein bis zwei Jahren pro Thema. Wenn also nur vier dieser Themen gleichzeitig in der Technik zu berücksichtigen sind, kommt es bereits alle drei bis sechs Monate zu Änderungen. Fachliche Themen können je nach Anwendungsgebiet sehr stabil sein - denken Sie z.B. an Lebensversicherungs- oder Darlehensverträge. Bei solchen Verträgen muss die Software auch 30 Jahre nach Vertragsabschluss noch korrekt entlang der ursprünglich vereinbarten Bedingungen arbeiten.



Erfolgsfaktor:

Lösen Sie technische und fachliche Probleme getrennt voneinander.



Optimal ist es, wenn existierende technische Lösungen sich mit neuen fachlichen Lösungen kombinieren lassen. Abbildung 4.1 zeigt wie drei fachliche und vier technische Lösungen miteinander verknüpft werden. Allgemein gilt, dass bei der Trennung von N fachlichen und M technischen Problem $n+m$ Lösungen erforderlich sind.

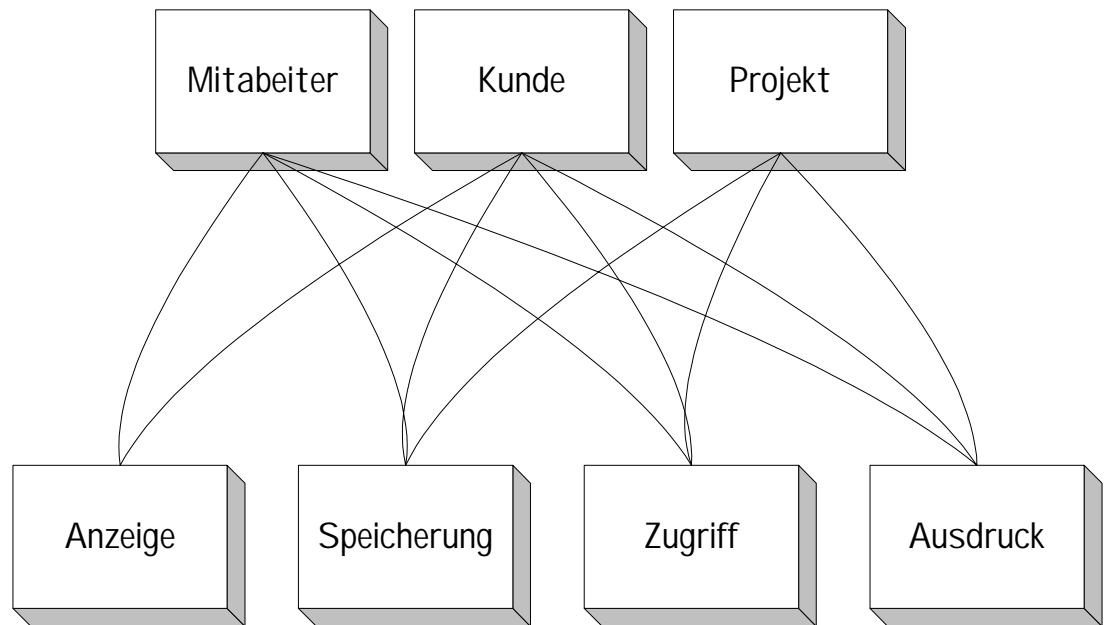


Abbildung 4.1: Kombination von 3 fachlichen und 4 technischen Lösungen (N+M)

Werden technische und fachliche Probleme jedoch gemischt, sind $N \times M$ Lösungen erforderlich (siehe Abbildung 4.2).



Abbildung 4.2: Kombination von 3 fachlichen und 4 technischen Lösungen (NxM)

Die folgende Tabelle stellt die beiden Ansätze anhand einiger Beispielzahlen für die Anzahl fachlicher und technischer Probleme gegenüber:

N Anzahl fachlicher Probleme	M Anzahl technischer Probleme	Trennung Fachlichkeit/ Technik N+M	Mischung Fachlichkeit/ Technik	Mehraufwand für Mischung als Faktor
4	3	7	12	1,7

20	3	23	60	2,6
20	6	26	120	4,6
40	10	50	400	8,0

Schon bei einem Projekt mittlerer Größe, bei dem 40 fachliche Probleme mit 10 technischen Problemen zu kombinieren sind, ist der Aufwand bei der Mischung der Lösungen 8 mal so hoch.

Ein Beispiel für einen guten Ansatz für die Trennung von Fachlichkeit und Technik liefert das Interface-Konzept der Programmiersprache Java¹¹. Mit Java-Interfaces ist es möglich, die Deklaration der Lösung von der Implementierung zu trennen.:

```
public interface Projekt
    implements Anzeige, Speicherung, Zugriff, Ausdruck {
}
```

Wenn im gesamten Code strikt „gegen“ Interfaces programmiert wird, dann kann nachträglich die Lösung verändert oder ausgetauscht werden. Die Code-teile, welche die Lösung verwenden können dabei völlig unverändert bleiben, solange die Beschreibung der Lösung – das Interface – sich nicht ändert.

Eine rigorose Trennung von Fachlichkeit und Technik erlaubt der Model Driven Architecture (MDA) - Ansatz (siehe Abschnitt 6.4. „Generierung“).

Richtiger Entwurf

Ein guter Entwurf ist einfach, kurz, klar und damit verständlich. Auf Basis des Modells in der Sprache des Auftraggebers gilt es nun einen Entwurf in der Sprache des Computers und der Programmierer zu erstellen. Es ist äußerst schwierig, auf Anhieb einen richtigen und tragfähigen Entwurf zu gestalten. Bei der Lösung von neuen und unbekanntenen Themen sind viele Randbedingungen zu beachten und ein Teil des Systems wirkt mit vielen anderen zusammen. Es hat sich gezeigt, dass bewährte Lösungsstrategien für wiederkehrende Probleme zu immer ähnlichen Entwurfsmustern führen.

Wenn Sie diese bekannten Muster nutzen, können Sie sich auf die Anwendbarkeit und Austauschbarkeit der Lösung weit mehr verlassen, als wenn Sie eine eigenen speziellen Ansatz erfinden.



Erfolgsfaktor:

Vereinfachen und standardisieren Sie Ihre Entwürfe durch die Verwendung von Entwurfsmustern.



¹¹ Die CORBA (Common Object Request Broker Architecture) bietet ebenfalls diese Möglichkeit.

Solche Entwurfsmuster werden inzwischen unter standardisierten Namen mit einheitlichen (englischen) Bezeichnungen für ihre Elemente verwendet. Beispielsweise sieht das „Iterator“ - Entwurfsmuster vor (siehe Abbildung 4.3), dass es eine Möglichkeit gibt, das nächste Element eines Iterators mit „next“ zu erhalten.

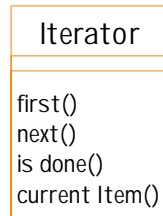


Abbildung 4.3 Das „Iterator“ Entwurfsmuster

Der klassische Katalog der Entwurfsmuster ist [Gam95] - die dort aufgezählten Entwurfsmuster haben inzwischen Einzug in viele Frameworks und Klassenbibliotheken gefunden. Die Java-Klassenbibliothek nutzt zum Beispiel mit jeder neuen Version mehr dieser Muster.

Entwurfsmuster sind keine fertigen Lösung - sie sind lediglich Beschreibungen eines Lösungswegs.

Bei der Verwendung von Entwurfsmustern ergeben sich folgende Fragen:

- Wie finde ich ein Entwurfsmuster für das vorliegende Problem bzw. wie erfahre ich davon, dass es ein Entwurfsmuster gibt?
- Wenn ich ein Entwurfsmuster gefunden habe, wie komme ich von diesem Muster zur fertigen Lösung?

Der Einsatz von Entwurfsmustern erfordert Erfahrung. Der erste Schritt dazu, ist, die bekanntesten Entwurfsmuster kennenzulernen und auszuprobieren. Dazu müssen Sie erneut das „Not invented here“ - Syndrom bekämpfen. Mit zunehmender Übung wird es Ihnen immer häufiger gelingen, zu einem Problem die passenden Entwurfsmuster auszuwählen. Um daraus eine komplette Lösung zu machen können Sie sich entweder eines Generators bedienen oder das Muster manuell mit Inhalt füllen.

Technischer Durchstich

Technik hat Ihre Tücken! In Softwareprojekten sorgt die Technik unserer Erfahrung für die größte Unsicherheit und Instabilität. Die Ursache liegt darin, dass die Technik von vielen verschiedenen Lieferanten geliefert wird. Sich in schneller Folge ändernde Produkten mit vielen Querabhängigkeiten dienen als Infrastruktur für die eigentlich zu realisierende Software.

Hr. Anglet hat gerade die neueste Version des XML-Adapters der Datenbank aus dem Internet geladen. Er will damit den Export bestehender PVW-Daten in das neue System ausprobieren. Schon die Installation scheitert jämmerlich. Das Installationsprogramm fordert ihn auf Servicepack 3 des Betriebssystems zu installieren. Nachdem er 120 MByte aus dem Internet geladen und installiert hat, wird er als Nächstes darauf aufmerksam gemacht, dass sein Internet-Browser ebenfalls mindestens Version 5.5 Servicepack 2 braucht. Weitere 20 MByte Download sind erforderlich - die Installation dauert noch einmal eine Stunde. Als die XML-Adapter-Software installiert ist, liest er in den Release-Notes, dass er unbedingt noch die Patches P437290 und P448859 für die Datenbank installieren soll und die Hinweise dazu zu beachten sind.

Ob die Produkte der verschiedenen Hersteller tatsächlich Ihren Ansprüchen genügen, alle Funktionen wie dokumentiert oder in der Werbung angepriesen funktionieren ist nicht sichergestellt. Die Situation ist um so kritischer, je weniger Erfahrung mit den technischen Komponenten aus anderen Projekten vorliegt. Gerade bei innovativen Projekten werden Sie in der Regel neue Komponenten ausprobieren müssen.

Wir empfehlen Ihnen einen „technischen Durchstich“ durchzuführen. Dazu dient ein minimaler Ausschnitt der Fachlichkeit, der so als Prototyp realisiert wird, dass zumindestens die wichtigsten technischen Komponenten durch diese Lösung beansprucht werden. Die Erfahrungen, die Sie mit dem Durchstich machen ermöglichen Ihnen besser zu beurteilen, ob das Gesamtsystem später in der beabsichtigten technischen Umgebung lauffähig sein wird.

Fachliche Prototypen

Fachliche !!!

4.2 Kurze Entwicklungszyklen

Erst testen dann codieren

4.3 Abnahme und Einführung

4.4 Literatur

- [Dar02] D. P. Darcy, C. F. Kemerer: **Software Complexity: Toward a Unified Theory of Coupling and Cohesion** Fall Workshop 2002, Management Information Systems Research Center, Carlson School of Management, University of Minnesota
http://misrc.umn.edu/workshop/spring2002/Darcy_020802.pdf
- [Gam95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: **Design Patterns - Elements of Reusable Object-Oriented Software**, Addison-Wesley, 1995
Standardwerk, das die bekanntesten Entwurfsmuster aufzählt und erläutert.
- [Knu97] D. E. Knuth: **The Art of Computer Programming Volume 1 - Volume 3**, Addison-Wesley, 1997
Klassisches voluminöses Werk, das die Grundlagen von Algorithmen und effizienten Lösungen für Standardprobleme beleuchtet: Vol 1: Sorting and Searching, Vol 2: Seminumerical Algorithms, Vol 3: Fundamental Algorithms. Die meisten der vorgestellten Algorithmen werden heute in Klassenbibliotheken, wie z.B. der Java-Klassenbibliothek angeboten. Hier erfährt man unter anderem detailliert, wie und warum ein Hashalgorithmus funktioniert (Vol 1. pp 514-558).
- [Sie00] J. Siedersleben, E. Denert: **Wie baut man Informationssysteme?** Informatik Spektrum, Band 23, Heft 4/2000, pp. 247-257
- [Url3] <http://www-st.inf.tu-dresden.de/hs/Vortraege/Thema5.pdf>
Vorstellung der Originalpapier von David Parnas zum Thema Modularisierung
- [Url6] Kurzweilige Beschreibung von David Parnas selbst, wie er auf das Information Hiding gekommen ist und wie dies noch heute Bedeutung behält.
- [Win96] T. Winograd: **Bringing Design to Software**, acmPress/Addison-Wesley, 1996
Sehr anregende Sammlung von Aufsätzen zum Thema Software-design mit vielen Bezügen zum Design in anderen Fachgebieten.



5

Änderung, Wartung, Pflege

5.1 Schrittweises Vorgehen

!!! Bezug auf CSMR Konferenz in Budapest

5.2 Fehlermeldungen und Änderungswünsche

Es gibt grundsätzlich drei Arten von Rückmeldungen, die vom Anwender Ihrer Softwarelösung an Sie herangetragen werden:

- Änderungswünsche
- Fehlermeldungen
- Bitten um Hilfestellung

Die Kontaktaufnahme kann dabei auf den unterschiedlichsten Wegen erfolgen z.B. per Telefon, E-Mail, Fax oder Brief. Die Rückmeldung gelangt vielleicht nicht direkt zu Ihnen sondern indirekt über Ihre Servicemitarbeiter, den Vertrieb, Partner oder andere Kunden.

- **Änderungswünsche**
werden unter der Kategorie „Changerequest“ mit der Abkürzung CR geführt. Sie betreffen Wünsche des Kunden, die über die zugesagten Eigenschaften der bisherigen Software hinausgehen. Ein Änderungswunsch kann vom Hersteller der Software angenommen, zurückgestellt oder abgelehnt werden. Bei Auftragssoftware muss der Auftraggeber für Änderungswünsche bezahlen. Ein Produktkunde muss darauf warten, dass der Änderungswunsch im Rahmen einer der nächsten Produktversionen umgesetzt wird.
- **Fehlermeldungen**
werden unter der Kategorie „Bugreport“ mit der Abkürzung BR geführt. Ein Fehler ist eine Abweichung des Verhaltens der Software von zugesagten Eigenschaften. Bei Fehlermeldungen liegen vom Anwender oft nicht genug Informationen vor, um eindeutig die Ursache feststellen zu können. Daher gehören zu den Fehlermeldungen auch vermutete Fehler – bei denen sich später herausstellen kann, dass es sich in Wirklichkeit um einen Änderungswunsch oder eine Bitte um Hilfestellung handelt.
- **Bitten um Hilfestellung**
werden unter der Kategorie „Supportrequest“ mit der Abkürzung SR geführt. Jede Meldung des Kunden, die keine Fehlermeldung oder Änderungswunsch darstellt wird hier eingeordnet. Typisch für diese Kategorie sind Fragen bzgl. der Installation, Konfiguration und Bedienung der Software.

Die Zufriedenheit Ihres Kunden mit Ihrer Software hängt sehr stark davon wie Sie mit seinen Rückmeldungen umgehen. Das Wichtigste ist, dass sich Ihr Kunde wahr- und ernstgenommen fühlt. Die erste Voraussetzung dafür ist, dass er eine „Quittung“ für seinen Kontakt erhält. Darin sollte enthalten sein

- Wann der Kunde sich gemeldet hat
- Mit welcher Aussage
- In welcher Kategorie Sie die Aussage eingeordnet haben (BR/CR/SR)

Damit Sie den Überblick behalten und Nichts in Vergessenheit gerät, sollten Sie den folgenden Erfolgsfaktor beherzigen:



Erfolgsfaktor:

Verfolgen Sie die Fehlermeldungen, Änderungswünsche und Bitten um Hilfe Ihrer Anwender systematisch.



Wir empfehlen Ihnen ein Werkzeug für diesen Zweck einzusetzen. Praktisch alle gängigen Konfigurationsmanagement-Werkzeuge haben geeignete Ergänzungen (siehe Abschnitt 6.3 „*Einzelteile und Versionen*“). Es gibt aber auch frei verfügbare Lösungen wie z.B. „Bugzilla“ [Url4].

5.3 Literatur

- [CSM02] **6th European Conference on Software Maintenance and Reengineering (CSMR 2002) 11–13 March 2002 - Budapest, IEEE, 2002**
- [Rad94] D. Radin: **Building a Successful Software Business**, O'Reilly, 1994
Alles was Sie wissen müssen, wenn Sie eine Softwarefirma gründen und betreiben wollen. Vom Marketing bis zur Kundenunterstützung finden Sie hier viele Tipps mit guten konkreten Beispielen.
- [Url4] <http://www.bugzilla.org/>
Opensource - Fehlerverfolgungssystem.

*"All parts should go together without forcing. You must remember that the parts you are reassembling were disassembled by you. Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer."
- IBM maintenance manual, 1925*

6

Infrastruktur und Werkzeuge

Ohne Computer unterstützte systematische Softwareentwicklung (CASE) ist heute kein Softwareprojekt mehr denkbar. Integrierte Entwicklungsumgebungen (IDE) sind inzwischen Standard.

Wie gut sind jedoch die einzelnen Werkzeuge in der Kette von Aktivitäten miteinander verbunden? Wie lassen sich einmal erzielte Ergebnisse durchgängig weiter nutzen? Wie arbeiten die für Softwareentwicklung verwendeten Werkzeuge mit den übrigen Programmen am Arbeitsplatzcomputer zusammen?

6.1 Arbeitsumgebung

Was brauchen Sie für die Arbeitsumgebung in einem Softwareprojekt? Die Umgebung sollte adäquat und standardisiert sein.

Sorgen Sie für eine adäquate Umgebung. Wir haben oft erlebt, dass in den Projekten gerade an den preiswerten Werkzeugen und an einfachen Arbeitsmitteln, wie Büchern, Computern und Druckern gespart wird. Die Personalkosten sind im Verhältnis um ein vielfaches höher. Sparen rechnet sich hier betriebswirtschaftlich nicht.

Vermeiden Sie Medienbrüche! Das Umformatieren von Ergebnissen von einem Werkzeug auf ein anderes ist mit Zeitaufwand und meistens sogar mit



Verlust von Inhalten verbunden. Zudem läßt wird dadurch die Möglichkeit der automatischen Verknüpfung von Ergebnissen durchbrochen.



Erfolgsfaktor:

Kümmern Sie sich rechtzeitig um eine adäquate und standardisierte Infrastruktur/Arbeitsumgebung für Ihr Softwareprojekt.

Mit dem Arbeitsplatz und der Arbeitsplatzausstattung der Projektmitarbeiter steht und fällt ihre Leistungsfähigkeit- und bereitschaft. Die ganze Bandbreite der Leistung von 0 bis 100% ist über die Eigenschaften des Arbeitsplatzes beeinflussbar. Geben Sie einem Mitarbeiter keinen eigenen Platz, keinen oder einen schlecht funktionierenden Computer und lassen Sie ihn „Mädchen für alles“ sein und von seiner Arbeitskraft wird fast nichts übrig bleiben. Im folgenden geben wir einige Tipps, wie sie nahe an die 100% kommen können. Wir empfehlen Ihnen die Investitionen für unsere Vorschläge im Verhältnis zu den Personalkosten durchzukalkulieren - sie werden überrascht sein wie die Verhältnisse liegen. Ein Beispiel dazu: Bürofläche macht in den Organisationen etwa 3% der Kosten aus, Personalkosten mehr als die Hälfte. Tom De Marco hat in „Peopleware“ [DeM99] nachgewiesen, dass die Arbeitsplatzausstattung jedoch massiv auf die Produktivität der Softwareentwickler wirkt. Eine Einsparung von 0,5% kann also leicht 5% „versteckte Kosten“ erzeugen. Umgekehrt kann eine Investition von zusätzlichen 0,5% genauso leicht 5% „versteckte Produktivität“ freisetzen. Im folgenden finden Sie einige Tipps, wie mit absolut gesehen geringen Investitionen die Arbeitsplatzausstattung verbessert werden kann.

Arbeitsplatz

Vorschläge, wie die Arbeitsplätze angeordnet sein können, finden Sie bei [Coc01]. Wir gehen hier von Einzelarbeitsplätzen unabhängig von der Größe und Aufteilung der zugehörigen Räume aus:

- Mindestens gebraucht wird ein Computergerechter Arbeitsplatz mit entsprechendem Tisch, Stuhl und Beleuchtung. Besser sind drei Schreibtische - einer als Computerarbeitsplatz, einer zum Schreiben und einer zum Experimentieren/Ablegen von halbfertigen Ergebnissen.
- Whiteboards (im Gruppenraum mit Kopierfunktion), alternativ Whiteboard/Flipchart mit Digitalkamera
- Telefon/Telefonanlage
Wichtig sind Funktionen, die es ermöglichen zeitweise ungestört zu arbeiten, also Rufweiterleitung, Mailboxfunktion, Anzeige der Rufnummer / des Namens des Anrufers im Display bevor abgenommen wird.
- Kaffeeküche
Es darf natürlich auch eine Teeküche sein. Da wo es zu Essen- und zu

Trinken gibt ist der natürliche Treffpunkt der Projektbeteiligten. Dieser Bereich sollte so gestaltet sein, das mehr möglich ist als nur das Zubereiten der Überlebensration.

- Infrastruktur für Post usw.

Was müssen die Mitarbeiter nicht unbedingt selbst machen? Post eintüten und frankieren? Büromaterial beschaffen? Manche Organisationen sorgen dafür, dass die Mitarbeiter private Besorgungen nicht selbst machen müssen, in der Hoffnung das Mitarbeiter für solche Dinge ihre Arbeit nicht unterbrechen brauchen.

Arbeitsplatzausstattung

Die folgende Ausstattungsliste erhebt keinen Anspruch auf Vollständigkeit, zeigt aber wichtige Punkte auf, die an einem Softwareentwicklungsarbeitsplatz vorhanden sein sollten, um effizient arbeiten zu können¹²:

- Adäquater Arbeitsplatzcomputer

Steht hier an erster Stelle, da in vielen Projekten noch unsinnigerweise genau hier gespart wird. Gute Rechner kosten heute weniger als eine Entwicklerwoche. Schlechte Rechner kosten so manches Projekt den Erfolg.

- Drucker/Fotokopierer

Ein schneller und guter Drucker sowie ein Fotokopierer mit Einzelblatteinzug, Sortier- und Heftfunktion muss mit wenigen Schritten erreichbar sein.

- Dokumentation

Werkzeuge zur Textverarbeitung, Tabellenkalkulation, und Präsentation sowie einfache Datenbankerstellung sollten an jedem Arbeitsplatz vorhanden sein. Ein einheitliches Office-Softwarepaket wird diesem Anspruch bereits gerecht.

- Editor

Ein eigenständiger guter Editor neben der später noch genannten integrierten Entwicklungsumgebung (IDE) ist Gold wert und kostet im Verhältnis wenig. Größtes Hindernis ist die Einigung auf einen Standard - den gerade im Bereich Editoren entwickeln so manche Entwickler ihre Vorlieben.

- Screen Painter/Prototyping

Minimum ist hier der Zugriff auf eine Möglichkeit zur digitalen Erfassung von Handzeichnungen. Ein Screenpainter gehört ansonsten heute gewöhnlich zur IDE dazu.

- Compiler

Gehört heute gewöhnlich zur IDE.

- Debugger

Gehört heute gewöhnlich zur IDE.

¹² Ausnahmesweise verwenden wir hier die üblichen Begriffe aus dem Englischen

- **Anforderungsaufnahme**

Bei kleinen Projekten wird hier kein spezifisches Werkzeug benötigt - dann reicht die Textverarbeitung bzw. zur systematischen Erfassung die Datenbank oder das Projektdiskussionsforum. Für größere Projekte ist es sinnvoll, dass die an der Anforderungsaufnahme beteiligten Mitarbeiter, Werkzeuge zur Geschäftsprozessanalyse und zur Anforderungsaufnahme an ihrem Arbeitsplatz vorfinden
- **Modellierung**

Bei kleineren Projekten genügt ein „Malwerkzeug“, das auch Modellieren kann. In größeren Projekten sind Werkzeuge zur Datenmodellierung, Funktionsmodellierung bzw. objektorientierten Modellierung sinnvoll.
- **Konfiguration Management**

Auf das Thema Konfiguration Management gehen wir im Abschnitt 6.3 „Einzelteile und Versionen“ noch detailliert ein. Zur Arbeitsplatzausstattung der Entwickler gehört der Zugriff auf mindestens das eigentliche Versionsmanagement sowie ein Werkzeug zum Nachverfolgen von Fehlern, Änderungswünschen und Supportanfragen der Kunden/Auftraggeber. Zudem wird ein Build-Management-Werkzeug benötigt.
- **Testen**

Werkzeuge für das Testen sind in drei Bereichen sinnvoll: Unit Test, Regressionstest und Performancetest.
- **Projektmanagement**

Für kleinere Projekt genügt sicherlich eine Tabellenkalkulation oder das Projektdiskussionsforum - für größere Projekte ist eine eigene Softwarelösung sinnvoll.
- **Zusammenarbeit**

Für die Zusammenarbeit ist zunächst mal der Zugang zu allen Papierdokumenten des Projektes erforderlich, d.h. zu den Projektunterlagen, den Büchern und Handbüchern zum Thema. Beim Internet-Zugang mit E-Mail, Web und News-Groups lohnt sich ähnlich wie oben beim Arbeitsplatzcomputer erwähnt das Sparen nicht. Die Themen Viren- und Datenschutz sollte jedoch auf jeden Fall gelöst sein.
Ein Projekttagbuch/Entwicklerforum dient dem Austausch von informellen Ergebnissen, Ideen, Vorschlägen und tagesaktuellen Informationsschnipseln, die sonst verloren gehen würden.
Ein Kunden-Forum dient dazu, dass Kunden neben dem formellen Weg über Änderungswünsche, Fehlermeldungen und Support-Anfragen ebenfalls ein informelles Forum erhalten.
- **Utility-Sammlung**

Entwickler auf Unix-Betriebssysteme haben hier die Nase vorn - dort sind kleinen „Helferlein“ zu dutzenden installiert. In anderen Umgebungen lohnt es sich Scriptsprachen wie awk/perl/python zur Verfügung zu haben, um Aufgaben zu automatisieren, die sonst mühsam mit der Hand erledigt

oder aufwendig mit „grossen“ Programmiersprachen gelöst werden müssten.

■ Verbindung der Werkzeuge/IDE

Der Abdeckungsbereich für integrierte Entwicklungsumgebungen wird immer größer. Vom Grundumfang Editor/Compiler/Debugger/Screen-Painter aus sind die Einbindungen von Modellierung, Anforderungsaufnahme, Änderungsverfolgung, Versionsmanagement, Projektverwaltung, und Testwerkzeug immer häufiger zu finden. Lediglich die Güte der Anbindung unterscheidet sich. Dabei ist nicht gesagt, dass die Anbindung der Werkzeuge am Besten ist, wenn alle Werkzeuge von einem Hersteller stammen. Eine ganze Reihe von Herstellern können die komplette Werkzeugpalette nur deswegen anbieten, weil sie in den letzten Jahren verschiedene andere Unternehmen aufgekauft haben. Die Integration dieser zugekauften Softwarepakete zu einer einheitlichen Linie dauert oft Jahre.

Das Team von Fr. Probst hat für die Dauer des Projektes zwei eigene Räume bei PV-Soft bezogen. Ein Raum dient als Gruppenarbeitsraum, der andere Raum ist ein Großraumbüro, das lediglich durch Trennwände aufgeteilt ist. Kaffeeküche und Kopierer befinden sich direkt nebenan und werden gemeinsam mit anderen PV-Soft Mitarbeitern genutzt. Der Gruppenarbeitsraum hat eine Computer-Projektionsmöglichkeit, und zwei Computerarbeitsplätze mit Netzwerkanschluss. An den zwei übrigen freien Wänden befinden sich Whiteboards und es ist ein Flipchart vorhanden. Im Schrank ist eine Digitalkamera verfügbar. Die einzelnen Arbeitsplätze im Großraumbüro sind in L-Form ausgelegt. Auf der Ecke befindet sich jeweils der Computer - leicht erhöht in Augenhöhe. Als Entwicklungsumgebung wird die gleiche Umgebung verwendet, die beim Auftraggeber üblich ist: alle Werkzeuge stammen vom gleichen namhaften Hersteller Omnitool¹³. Im Team hat diese Entscheidung für Aufregung gesorgt, insbesondere Hr. Teschner mag sich nicht an den Editor der Omnitool-IDE gewöhnen.

6.2 Werkzeugauswahl

Wie finden Sie unter der grossen Auswahl möglicher CASE-Tools das richtige Werkzeug für Ihren Zweck? Wir empfehlen Ihnen:



Erfolgsfaktor:

Wählen Sie Werkzeuge systematisch nach denen von Ihnen festgelegten Kriterien aus.



Es ist nicht unbedingt nötig, dass Sie bei der Werkzeugauswahl nach dem 55 Seiten starken ISO-Standard vorgehen [ISO95], in diesem Abschnitt finden Sie bereits eine Liste wichtiger Kriterien für die CASE-Tool Auswahl. Für grössere Organisationen empfiehlt es sich zudem die Toolauswahl als eigen-

¹³ Nicht zu verwechseln mit dem gleichnamigen Hersteller von Formen für die Industrie

ständiges Projekt aufzusetzen. Durch die Investition von einigen tausend Euro können Sie so ggf. Millionen einsparen.

Der folgende Ansatz führt zu einer systematischen Bewertung und Auswahl eines Werkzeugs:

- Stellen Sie Ihren eigenen Kriterienkatalog zusammen und gewichten Sie die Kriterien prozentual entsprechend ihrer Wichtigkeit. Die Gewichte aller Kriterien zusammen sollen dabei 100% entsprechen.
- Stellen Sie eine Liste aller Kandidaten-Werkzeuge zusammen, die potentiell in Frage kommen. Beschaffen Sie zunächst einfaches Informationsmaterial zu den Werkzeugen per Internet oder durch Anforderung beim Hersteller.
- Bewerten Sie die Kandidaten zunächst nach dieser „Papierform“ bzgl. der Kriterien und vergeben Sie dazu einen Erreichungsgrad von 0-100%.
- Bilden Sie eine Gesamtbewertung indem Sie pro Kriterium das Gewicht mit dem Erreichungsgrad multiplizieren und die so gebildeten Werte pro Kandidat addieren. Das Ergebnis ist wieder eine Prozentzahl. Diese Gesamtbewertungszahl gibt an, wie das Werkzeug insgesamt mit Ihren Ansprüchen übereinstimmt. Sie werden in der Praxis kaum ein Werkzeug antreffen, dass hier 100% erreicht.
- Treffen Sie nun eine engere Wahl und testen Sie einige der Werkzeuge bzw. lassen Sie sich diese vom Hersteller vorführen. Führen Sie dann erneut die Bewertung durch - diesmal aber nicht nach Papierform, sondern anhand Ihrer konkreten eigenen Erfahrungen mit dem Werkzeug. Für eine faire Bewertung ist es hilfreich - wenn alle Werkzeuge anhand des gleichen „Beispielszenario“ getestet werden.

Beispieldokument 6.1 zeigt eine Entscheidungsmatrix die dem dargestellten Auswahlverfahren entspricht. Im Beispiel wurde Kandidat D nach der Vorauswahl gestrichen und die Kandidaten A, B und C getestet. Die Entscheidung ist jetzt zwischen Kandidat B und Kandidat C zu treffen - der Zahlenwert alleine sollte für die Schlussentscheidung nicht ausschliesslich eine Rolle spielen sondern lediglich die Entscheidungsdiskussion leiten.

Nr.	Kriterium	Gewicht	Kandidat A		Kandidat B		Kandidat C		Kandidat D
			Papierform	Test	Papierform	Test	Papierform	Test	Papierform
1	Funktion	40%	80%	65%	60%	70%	90%	85%	50%
2	Hersteller	30%	60%	50%	75%	80%	50%	60%	40%
3	Preis	20%	70%	90%	80%	80%	100%	90%	100%
4	Sonstiges	10%	90%	80%	80%	90%	50%	40%	30%
	Summe	100%	73%	67%	71%	77%	76%	74%	55%

Beispieldokument 6.1: Entscheidungsmatrix für eine Werkzeugauswahl

Im Folgenden nenne wir einige geeignete allgemeine Kriterien und dann jeweils mögliche spezifische Kriterien für die einzelnen Werkzeugarten.

Allgemeine Kriterien

- **Lebenszyklus**
Ab wann und wie lange soll das Werkzeug voraussichtlich eingesetzt werden?
- **Verfügbarkeit**
Gibt es das Werkzeug auf dem deutschen Markt? Gibt es eine aktuelle Version? Wird das Werkzeug vom Hersteller noch gepflegt?
- **Referenzen**
Welcher Kunde des Herstellers in ähnlicher Situation setzt die Lösung bereits ein? Wie ist der Kunde mit der Lösung zufrieden?
- **Hersteller**
Wie ist die wirtschaftliche Lage und Wettbewerbsfähigkeit des Herstellers - wird es ihn über den gesamten Lebenszyklus voraussichtlich noch geben?
- **Wartung/Support durch den Hersteller**
Welche Reaktionszeiten und welche Wartungsmöglichkeiten bietet der Hersteller? Welche Qualität der Leistungen wird sichergestellt?
- **Lizenzbedingungen/Preis**
Was kostet die Ausstattung der betroffenen Arbeitsplätze? Welche Lizenzmodelle werden angeboten?
- **Unterstützung durch den Hersteller**
Welche Dienstleistungen rund um das Werkzeug, wie Training und Einführungsberatung bzw. kundenspezifische Anpassung werden angeboten?
- **Anforderungen an Werkzeug-Umgebung**
Welche Ausstattung müssen die Arbeitsplatzrechner haben? Ist ggf. ein spezieller Serverrechner erforderlich? Welche andere Software muss verpflichtend vorhanden sein?
- **Einbindung in die übrige Infrastruktur/Schnittstellen**
Wie arbeitet das Werkzeug mit anderen CASE-Tools zusammen?
- **Import/Exportmöglichkeiten/API¹⁴ - Migrationspfade**
Von welchen anderen Werkzeugen gleicher Art können Daten übernommen werden? An welche anderen Werkzeuge gleicher Art können Daten übergeben werden? Welche APIs werden zur Verfügung gestellt?
- **Dokumentation**
Wie ist die Dokumentation für das Werkzeug gestaltet? Wie gut ist die On-

¹⁴ API=Application Program Interface

line-Hilfe? Wie gut sind die mitgelieferten Beispiele für den Einstieg geeignet?

■ Betreuungsaufwand

Wieviel Betreuungsaufwand ist für dieses Werkzeug zu erwarten?

IDE: Editor/Compiler/Debugger/Screen Painter

■ IDE

Mindestens wird eine Projektverwaltung benötigt - es muss leicht möglich sein, Dateien dem Projekt hinzuzufügen oder wieder daraus zu entfernen. Die Projektverwaltung soll den Status der Dateien anzeigen und die Abhängigkeiten der Dateien verfolgen. Auf Knopfdruck soll die Neuübersetzung der veränderten und der davon abhängigen Dateien ausgelöst werden können.

■ Editor

- Volltext-Editor mit den üblichen nützlichen Funktionen
- Anpassbar an Coding-Konventionen - d.h. Anzahl von Leerzeichen für Tabs, Speicherformat usw.
- Syntax highlighting für die gewählten Sprache, d.h. der Editor erkennt automatisch Schlüsselworte, Bezeichner und Kommentare und stellt diese farbig dar.

■ Screen Painter

Grafischer Editor mit dem Dialoge und Fenster erstellt werden können, ohne dafür Code zu schreiben. Die Steuerelemente wie Felder, Menüs und Schaltflächen und statischen Teile wie Text, Linien und Rahmen können nach dem WYSIWYG¹⁵ - Prinzip plaziert werden.

■ Debugger

Mindestens müssen Haltepunkte gesetzt werden können und der Zustand von Variablen angezeigt werden. Der Debugger muss im Einzelschritt und funktionsweise den Code abarbeiten können. Mehrere Threads müssen möglich sein.

¹⁵ WYSIWYG=What you see is what you get - Die Anzeige während der Arbeit entspricht dem später zu erwartenden Ergebnis

Geschäftsprozessanalyse

- Werden alle gängigen Prozessmodellelemente wie Prozesse, Aktivitäten, Übergänge, Ressourcen, Rollen, Dokumente, Geschäftsobjekte usw. unterstützt?
- Wie können Prozesse simuliert, analysiert und optimiert werden?
- Wie können definierte Abläufe an Workflow-Werkzeuge weitergereicht werden?

Anforderungsaufnahme

- Strukturierte Erfassungsmöglichkeit von Anforderungen
- Erfassungsmöglichkeit für Abnahmekriterien
- Ausgabemöglichkeit als geschlossener Text
- Testfallgenerierung
- Einbindung externer Dokumente und Referenzen
- Glossar

Modellierung

- Unterstützung der aktuellen UML

Konfiguration Management

- Versionsverwaltung
- Nachverfolgung von Fehlermeldungen, Änderungswünschen und Hilfefragen
- Automatische Erzeugung von Quittungen für den Anfrager

Weitere Kriterien finden sie in Abschnitt 6.3 „Einzelteile und Versionen“.

Testen

- Unit Test
- Regression Test
- Performance Test

Projektmanagement

- Projektplanung
- Projektverfolgung

Zusammenarbeit

- Projekttagbuch/Entwicklerforum
- Kunden-Forum

6.3 Einzelteile und Versionen

Wie gelingt es innerhalb der Softwareentwicklung den Überblick über tausende von Einzelergebnissen zu behalten, insbesondere wenn sehr viele Entwickler gleichzeitig daran arbeiten?

Die modulare Softwareentwicklung (siehe „*Teile und Herrsche*“ in Abschnitt 4.1) führt dazu, dass zur Auslieferung einer Software viele Einzeldokumente (insbesondere Quellcode) aufeinanderabgestimmt werden müssen. Zu verschiedenen Auslieferungsanlässen werden *Softwareversionen* gebildet (vgl. Abbildung 6.2). Basis der Versionen sind gemeinsame Dokumente, die es in verschiedenen Änderungszuständen gibt. Abbildung 6.1 illustriert dies indem die Teile „Haare“, „Augen“ und „Mund“ eines Kopfes als Stellvertreter für drei Teildokumente einer Software stehen.

Jedes Teildokument ändert sich im Laufe der Zeit und kann mit den anderen Teilen zu einer Vielzahl von Versionen kombiniert werden. Nur einige dieser Versionen dienen einem Auslieferungszweck, z.B.:

- Entwurf
- Prototyp
- Messe
- Betatest
- Auslieferung an den Kunden

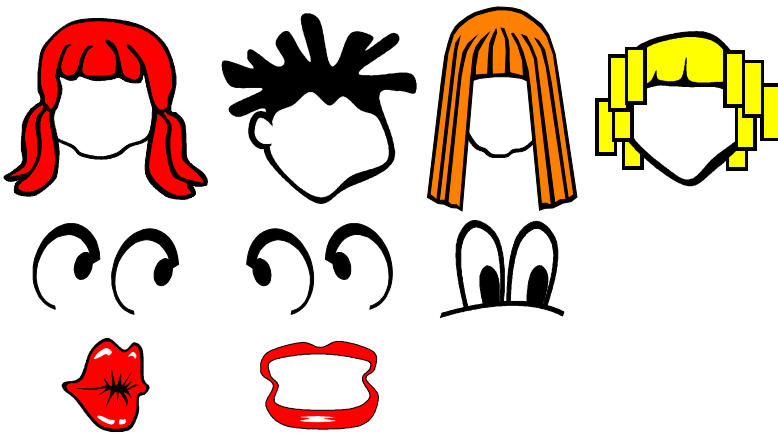


Abbildung 6.1 Gemeinsame Teile mit verschiedenen Änderungszuständen.

Die Softwarekonfigurationsverwaltung beugt dem Risiko vor, das durch falsch zusammengestellte Versionen entsteht. Ein einziges Teil, das im falschen Zustand einer Version hinzugefügt wird, kann bereits massiven Schaden anrichten. Fehler, die durch die falsche Konfiguration von Software entstehen, sind zudem besonders schwer zu finden.

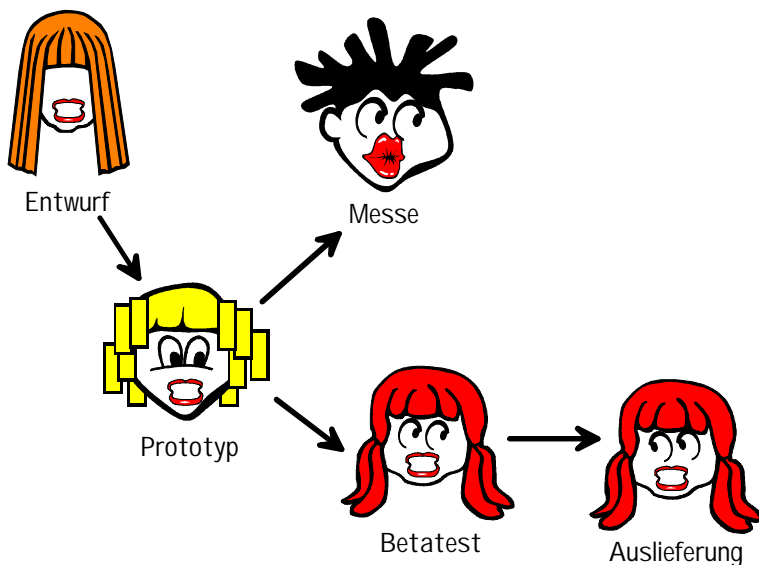


Abbildung 6.2: Versionen

Wegen der hohen Zahl von Kombinationsmöglichkeiten ist es nur mit Computerunterstützung sinnvoll möglich, die Kontrolle über die Softwarekonfiguration zu behalten. Sehr gute Werkzeuge zu diesem Zweck gibt es in allen Preislagen. Achten Sie aber darauf, sich nicht nur auf das Werkzeug zu verlassen. Es ist erforderlich, die Abläufe um die Versionsverwaltung herum in den folgenden Bereichen zu gestalten

- Konstruktion
- Mitverfolgen

- Team
- Prozess
- Steuerung
- Komponenten
- Struktur

Die Minimalanforderungen sind:

- Konstruktion
Es muss möglich sein, Versionen aus einem Archiv heraus zu reproduzieren. Dies sollte auch für Momentaufnahmen möglich sein (z.B. tägliche „Schnappschüsse“).
- Mitverfolgen
Der Status der Einzeldokumente und der Versionen muss ersichtlich sein. Es muss möglich sein, die Spur einer Änderung bis zum Autor und Datum der Änderung zu verfolgen. Es muss eine möglichst automatisierte Buchführung über Änderungen geben. Statistiken und Berichte helfen den Überblick zu behalten
- Team
Entwickler und Gruppen von Entwicklern müssen unabhängig voneinander arbeiten können und dürfen durch die Versionsverwaltung nicht behindert werden. Konflikte zwischen widersprüchlichen Änderungen müssen bereinigt werden können. Es muss möglich sein, zum Test Änderungen vorzunehmen, die in die endgültige Version nicht einfließen.
- Prozess
Die Dokumentation von Änderungen, der Austausch von Informationen über Änderungen und die Verteilung von Änderungsaufgaben soll unterstützt werden.
- Steuerung
Die Verfolgung und Änderungen und Fehler muss mit der Versionsverwaltung koordiniert erfolgen. Der Zugriff auf das Archive muss geregelt sein.
- Komponenten
Es muss ein gemeinsames Archiv geben, aus dem heraus Versionen gebildet werden können. Die Beschreibung von Versionen anhand von Stücklisten muss möglich sein und selbst wieder unter Versionskontrolle genommen werden. Versionen müssen in einen Projektkontext gestellt werden können.
- Struktur
Es muss möglich sein, Schnittstellen und Verbindungen zwischen Einzeldokumenten anzugeben und daraufhin einen konsistenten Zustand zu bilden.

Die Softwareversions- und Konfigurationsverwaltung wird mit zunehmender Projektgröße immer wichtiger. Während sich kleine Teams von drei bis fünf

Entwicklern noch mit einfachen Mitteln abstimmen können, ist die Verantwortung für dieses Thema bei Teams schon ab zwanzig Leuten leicht ein Vollzeit-job.



Erfolgsfaktor:

Setzen Sie eine Ihren Bedürfnissen entsprechende Softwareversions- und Konfigurationsverwaltung ein.



Schätzen Sie das Risiko für Ihr Projekt ein, und wählen Sie danach eine geeignete Lösung für sich aus. Dazu gehört sowohl eine Software, als auch die Verfahren und die Besetzung des Themas mit einer verantwortlichen Person.

Hr. Teschner ist schon seit 7 Uhr im Büro und begrüßt Hr. Angelet „Peter meine Testfälle laufen seit heute nicht mehr - ich bekomme die Fehlermeldung ``java.lang.NoClassDefFoundError: org/w3c/dom/Element`` - um den Fehler zu finden habe ich schon die Version von gestern wiederhergestellt, aber der Fehler bleibt“. Nach einiger Zeit stellen beide gemeinsam fest, dass die XML-Bibliotheken `xalan.jar` und `xerces.jar` nicht im Konfigurationsverwaltungssystem erfasst sind und gestern das gemeinsam genutzte Verzeichnis im Netzwerk umbenannt wurde. Sie informieren Fr.

Besonders wichtig für die Konfigurationsverwaltung sind:

- Die Vollständigkeit - neben den eigenen Ergebnissen sollten Sie auch alle verwendeten und zugekauften Komponenten so wie nach Möglichkeit auch Werkzeuge und Konfigurationsdateien unter Konfigurationsverwaltung nehmen.
- Die Beherrschung der Komplexität - teilen Sie die zu verwaltenden Einheiten in kleinere Bereiche auf die unabhängig voneinander gesteuert werden. Auch hier gilt wieder das Prinzip des „teile und herrsche“.

6.4 Generierung

!!!

Model Driven Architecture

!!! [Hub02]

Platform Dependent Model (PDM)

Platform Independent Model (PIM)

6.5 Literatur

- [Bab86] W. A. Babich: **Software Configuration Management - Coordination for Team Productivity**, Pearson Higher Education, 1986
- [Coc01] siehe [Coc01] in Abschnitt 7.9
- [DeM99] siehe [DeM99] in Abschnitt 1.9.
- [Hub02] R. Hubert: **Convergent Architecture - Building Model-Driven J2EE Systems with UML**, Wiley, 2002
- [ISO95] International Organization for Standardization: **ISO/IEC DIS 14102 - Evaluation and Selection of Computer-Aided Software Engineering (CASE Tools)**, ISO, 1995 (siehe auch <http://www.iso.ch>)

„Gute Vorsätze sind wie Schecks, die man auf eine Bank ausstellt, bei der man gar kein Konto hat.“
Oscar Wilde

7

Projektmanagement, Kommunikation, Politik

Worin unterscheiden sich der Teufel und die Familie? Warum finden die meisten Menschen Personal Computer besser als Mainframes obwohl Mainframes doch viel leistungsfähiger sind? Warum ist die Atomkraft unbeliebter als die Windkraft, obwohl doch mehr Atomstrom produziert wird?

Akzeptanz entsteht aus dem Zusammenwirken von Problemlösungskompetenz und Vertrauen. Dem Teufel traut zwar niemand über den Weg aber man traut ihm alles zu. Deswegen hat selbst der Teufel eine gewisse Akzeptanz. Die Familie genießt hohes Vertrauen und sorgt dabei für die wichtigsten Probleme Grundbedürfnisse, wie Essen, Schlafen und Kleidung. Bei Atomkraft denken viele Menschen eher an die Folgen von Tschernobyl als an den Strom aus ihrer Steckdose - Windkraft dagegen gilt als saubere und zukunftsweisende Lösung. Problemlösungskompetenz und Vertrauen alleine genügen nicht - sie brauchen von beidem die richtige Menge, um Akzeptanz zu genießen. Abbildung 7.1 zeigt die Einordnung der gezeigten Beispiele in eine grafische Darstellung:

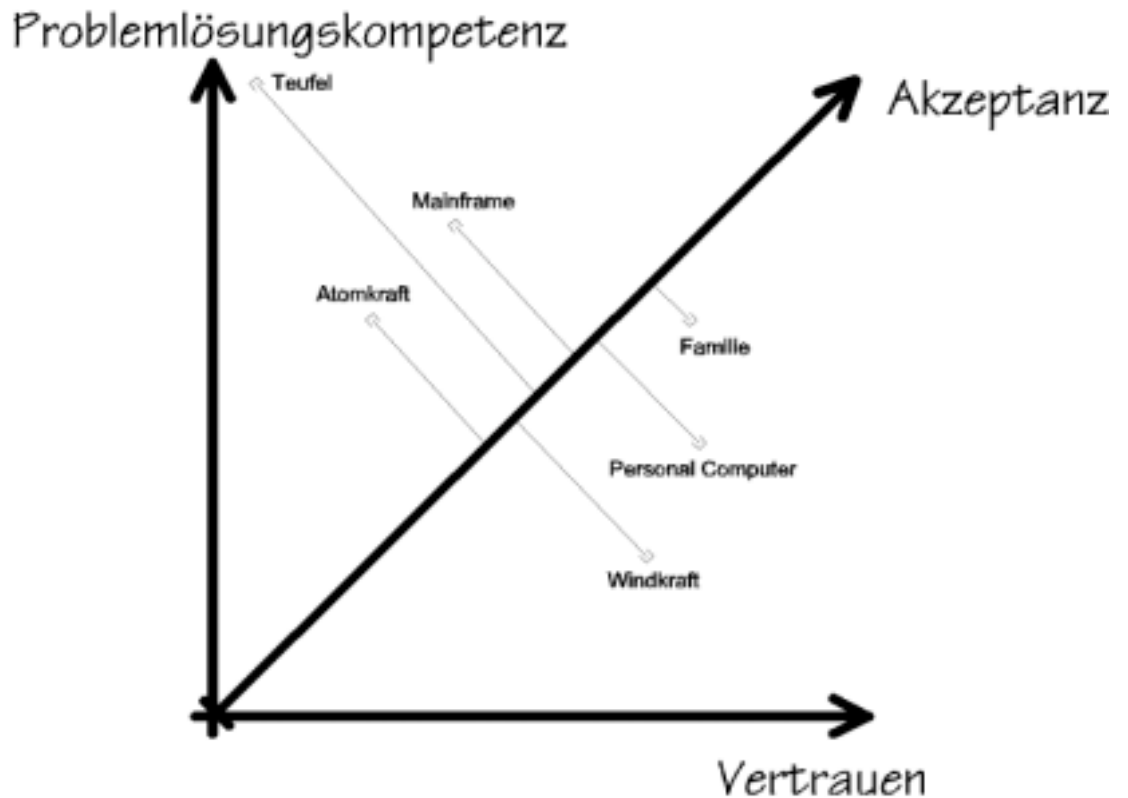


Abbildung 7.1: Akzeptanz als Kombination von Vertrauen und Problemlösungskompetenz

Eines Ihrer Projektziele ist es, möglichst hohe Akzeptanz zu erreichen. Dies können Sie zum einen durch die Erhöhung Ihrer Problemlösungskompetenz erreichen. Dazu dienen

- Zielorientierung
- Wissensmanagement
- Planung und Steuerung
- Das Fällen von Entscheidungen
- Risikomanagement

Zum anderen können Sie das Vertrauen in Ihr Projekt stärken. Dazu gehören

- Umgang im Team
- Kommunikation
- Führung und Förderung

Die richtige Kombination führt zur höchsten Akzeptanz. In den Abschnitten dieses Kapitels finden Sie Hinweise darauf, wie Sie Ihre Akzeptanzposition verbessern können.

7.1 Zielorientierung

In Abschnitt 1.5 „Gemeinsames Erfolgsverständnis“ haben wir darauf hingewiesen wie wichtig es ist, die Ziele des Projektes und den Weg dorthin zu klären. Die Aufgabe des Projektmanagements ist es nun den Kurs auf das Ziel zu halten oder wenn nötig neue Ziele festzulegen und eine Kursänderung vorzunehmen.

Zerlegung der Projektziele

Setzen Sie sich Meilensteine, die Sie auf dem Weg zu ihrem eigentlichen Projektziel erreichen wollen. Die Festlegung der Meilensteine hängt von der Art Ihres Vorgehens ab:

- Wenn Sie nach dem *Wasserfall-Prinzip* vorgehen, dann liefern Ihnen die Ergebnisse der ersten Phasen des Projektes die Grundlage für die nächste Phase des Projektes. Potentielle Meilensteine sind der Beginn einer neuen Phase (d.h. die Modellierung beginnt nach dem genügend Anforderungen vorliegen) und das Ende einer Phase (alle Anforderungen sind aufgenommen).
- Wenn Sie iterativ vorgehen, dann lernen Sie aus den Fehlern erster Schritte. Sie gehen Ihre Projektziele an, indem Sie sich einen immer größeren Umfang zur Realisierung vornehmen. Jeder Iterationszyklus bringt sie dem Ziel näher und Sie lernen aus den Problemen der Anfangsschritte für die folgenden Schritte.
- Wenn Sie inkrementell vorgehen, dann teilen Sie das Projekt in Teilgebiete („Inkremente“) auf, die parallel zueinander bearbeitet werden und mit eigenen zeitlichen und inhaltlichen Zielen versehen werden. Das Gesamtprojektziel wird durch Integration aller inkrementell erstellten Ergebnispakete erreicht.

Die größten Vorteile erreichen Sie unserer Erfahrung nach durch iteratives oder inkrementelles Vorgehen. Dies hat die Umfrage bestätigt:



Erfolgsfaktor:

Schrittweise iterative oder inkrementelle Näherung an das Projektziel.



Wenn Sie diesen Erfolgsfaktor beherzigen, dann wird:

- Das Gesamtrisiko kleiner, da Sie je Meilenstein ein Ergebnis bekommen, auf das Sie reagieren können.
- Das Tempo von Meilenstein zu Meilenstein höher, da Sie die Erkenntnisse der vorherigen Meilensteine zur Optimierung nutzen können.

- Das Reagieren auf Änderungen leichter, da jeder Schritt die Möglichkeit bietet, den Kurs zu ändern.

Achten Sie bei der Gestaltung der Iterationen oder Inkremente auf die richtige Größe. Bei einer Projektlaufzeit von zwei Jahren sollten die Projektabschnitte zum Beispiel maximal etwa 3 Monate gross sein – sonst treten in den Abschnitten bereits die Probleme „ganzer“ Projekte auf. Bei kürzeren Projekten sollten Sie dennoch keine Abschnitte bilden, die kürzer als 3 Wochen sind, denn der Aufwand zum Fertigstellen, Integrieren, Präsentieren und Diskutieren der Zwischenergebnisse ist sonst im Verhältnis zu dieser Zeitspanne zu hoch.

Erwartungsmanagement und Verbindlichkeit

Eines der wichtigsten Ziele des Softwareprojektes ist es die Erwartungen der Beteiligten zu erfüllen. Alles was Sie dafür tun müssen:

- Dafür sorgen, dass sie überhaupt in der Lage sind Erwartungen zu erfüllen.
- Wissen welche Erwartungen sie mit den gegebenen Mitteln erfüllen können.
- Dafür sorgen, dass Sie nur so hohe Erwartungen wecken, wie Ihren Erfüllungsmöglichkeiten entspricht.



Erfolgsfaktor:

Seien Sie verbindlich - machen Sie nur Versprechungen, die sie auch halten können. Halten Sie sich an Ihre Versprechen!

Bei diesem Erfolgsfaktor gilt - leichter gesagt als getan. Es ist niemals möglich alle Versprechungen zu 100% einzuhalten - alleine schon deswegen, weil es Unterschiede im Verständnis gibt. Es ist nur natürlich, dass jemand, dem Sie ein Verprechen geben, dieses zu seinen Gunsten auslegt. Auf der anderen Seite ist die Versuchung gross, mehr zu versprechen, als man halten kann, wenn damit Vorteile verbunden sind.

Unserer Erfahrungen nach ist es wesentlich einfacher, Erwartungen im Rahmen zu halten, als zu versuchen, überzogene Erwartungen zu erfüllen. Zudem wirkt die Vorgeschichte Ihrer Beziehungen sich auf die Wahrnehmung der Beteiligten aus. Das Zuverlässigkeit in der Vergangenheit wird auf die Zukunft projiziert. Ein gleichmässiges Verhalten ist leichter nachvollziehbar als eine wechselhafte Erfüllung von Versprechungen.

7.2 Entscheidungen fällen

Wie lassen sich Entscheidungen in Softwareprojekten vorantreiben? Wie oft haben Sie in den Zwischenergebnissen Ihrer Projekte schon den Hinweis „muss noch geklärt werden“ oder „to be defined“ vorgefunden?

Das Treffen von Entscheidungen ist eine der wichtigsten Aktivitäten in Softwareprojekten überhaupt. Erstaunlicherweise kommt die Aktivität „Entscheidungen treffen“ in so manchem Vorgehensmodell überhaupt nicht vor! Unserer Erfahrung nach lässt sich an der Art- und Weise wie in einem Softwareprojekt Entscheidungen getroffen werden am schnellsten die Erfolgchance eines Projektes ablesen:

- Die Erfolgchance eines Projektes, indem keine Entscheidungen getroffen werden ist null.
- Werden Entscheidungen zu leichtfertig und ohne Abstimmung getroffen, dann kommt das Projekt scheinbar schnell voran, aber die Risiken steigen.
- Eine Entscheidungsfreudige Konsenskultur im Team hat die besten Erfolgchancen.

Der Aufsatz „Decisions, Decision“ aus [Con01] verweist darauf, wie wichtig die Gruppenkultur für die Entscheidungsfindung und die Qualität der Entscheidungen ist. Die wichtigste Kernaussage ist, dass sich der Leiter einer Gruppe aus der Entscheidungsfindung so weit es geht heraushalten muss, damit es zu einer möglichst guten Entscheidung kommt.

Klärung und Entscheidung trennen

In unseren Projekten haben wir oft erlebt, dass die Klärung eines Themas sehr schleppend vorangeht, wenn die Beteiligten zur gleichen Zeit versuchen ein Thema durchzusetzen, also eine Entscheidung suchen.

Hr. Angelet, Hr. Entrop und Fr. Teschner diskutieren darüber, wie die Projektverwaltung Internet-fähig gemacht werden soll. Hr. Angelet ist dafür Java Applets zu verwenden und will gerade erklären, wie das geht da unterbricht Hr. Entrop. Er argumentiert Java Applets seien veraltetet und man müsse Java Server Pages verwenden. Das geht eine Weile so hin- und her bis Frau Teschner unterbricht und gerne erstmal verstehen möchte, was denn überhaupt der Unterschied zwischen Java Applets und Java Server Pages ist und was für das eine und was für das andere spricht.

Die Trennung des Klärungsprozesses vom Einigungsprozess hilft sowohl bessere Entscheidungen zu bekommen als auch schneller zu einer Entscheidung zu kommen. Je genauer und präziser festgelegt ist, welches Problem gelöst, welche Ursachen abzustellen, welche Ziele zu erreichen sind, desto präziser, sicherer und klarer kann die Entscheidung getroffen werden. Wenn die Klä-

rung nicht von der Entscheidung getrennt wird, dann entsteht schnell ein Klima des Mißtrauens und der Aggression. Statt einer Entscheidung kommt es zu:

- unendlichen ergebnislosen Diskussionen
- einem Kampf um persönliche Standpunkte
- konfliktloser Unterwerfung

Das Ziel ist jedoch, durch Aushandeln, das gemeinsame entwickeln von Kompromissen bzw. die gemeinsame Auswahl der „richtigen“ Entscheidung zu einer sachlich begründeten guten Entscheidung zu kommen.

In einem Projekt haben wir monatelang um die richtigen Entscheidungen für die Softwarearchitektur gerungen und kamen nur sehr langsam voran. Die Beteiligten waren schon ziemlich entnervt von den langen sich wiederholenden Ritualen der Sitzungen. Der Durchbruch gelang, als folgendes Muster für die Sitzungen angewendet wurde:

- Welche Entscheidungen liegen für heute zur Diskussion vor?
- In welcher Reihenfolge wollen wir diese Entscheidungen behandeln?
- Entscheidung über die einzelnen Punkte

Zu jeder Entscheidung wurde folgender Verlauf gewählt:

Klärung:

- Was steht im Detail zur Entscheidung an?
- Welche Ziele werden verfolgt?
- Welche Vorschläge/Möglichkeiten gibt es, diese Ziele zu erreichen?

Entscheidung:

- Bewertung der Vorschläge bezüglich des Grades der Zielerreichung
- Vorläufige Entscheidung
- Analyse der möglichen Folgeprobleme
- Endgültige Entscheidung

Entscheidend war, die sachliche Klärung bezüglich des Problems vollkommen zu trennen von den der eigentlichen Auswahl der Lösung. Bei der Protokollierung der Sitzungen wurden die unterschiedlichen Meinungen explizit festgehalten. Bei der eigentlichen Entscheidung wurde im Konsens entschieden - jeder Beteiligte musste mindestens sagen können „Mit dieser Entscheidung kann ich leben“.

Risiko und Entscheidung

!!!





Erfolgsfaktor:

Treffen Sie ihre Entscheidungen entsprechend dem Risiko das damit zusammenhängt und trennen Sie die Klärung und die eigentliche Entscheidung.

Gründe, Entscheidungen schnell zu fällen:

- Die Voraussetzung für weitere Entscheidungen in abhängigen Bereichen wird geschaffen.
- Das Risiko einer Fehlentscheidung ist gering, der Zeitverlust durch Abwarten käme auf jeden Fall teurer.

Gründe, Entscheidungen aufzuschieben:

- Angst vor der Verantwortung, eine falsche Entscheidung getroffen zu haben.
- Hohes Risiko einer Fehlentscheidung - es ist günstiger, die Entscheidung aufzuschieben, da die Wahrscheinlichkeit und Schwere der Folgen einer Fehlentscheidung hoch ist.
- Hoffnung, dass durch mehr Information später eine bessere Entscheidung möglich ist.

7.3 Vom Umgang mit Eisbergen

Nach der Eisberg-Theorie stellen die formalen und beobachtbaren Teile der Zusammenarbeit nur etwa 10% aller Aspekte dar, die Zusammenarbeit ausmachen. Aufgabenbeschreibungen, Vorgehensmodelle, Pläne, Richtlinien, Vereinbarungen, Protokolle und andere greifbare Ergebnisse stellen demnach nur die Spitze des Eisbergs dar. Die 90% des Eisbergs, die sich als informaler nicht greifbarer Anteil unter dieser Oberfläche verbergen haben die größere Wirkung auf die Zusammenarbeit. Das liegt daran, dass Menschen viel stärker durch Ihr Unterbewußtsein gesteuert werden als durch den Verstand. Alle Entscheidungen werden grundsätzlich „aus dem Bauch heraus“ getroffen. Über den Verstand kann der Bauch beeinflusst werden aber nur indirekt. Persönliche Beziehungen, Rollenverhalten, Machtverteilung, Projekt- und Unternehmenskultur sind deshalb wichtiger als es oberflächlich betrachtet erscheint. Für die Gestaltung eines Softwareprojektes ist die Kenntnis von beiden Ebenen des Beziehungsgefüges von Bedeutung (vgl. Abbildung 7.2).

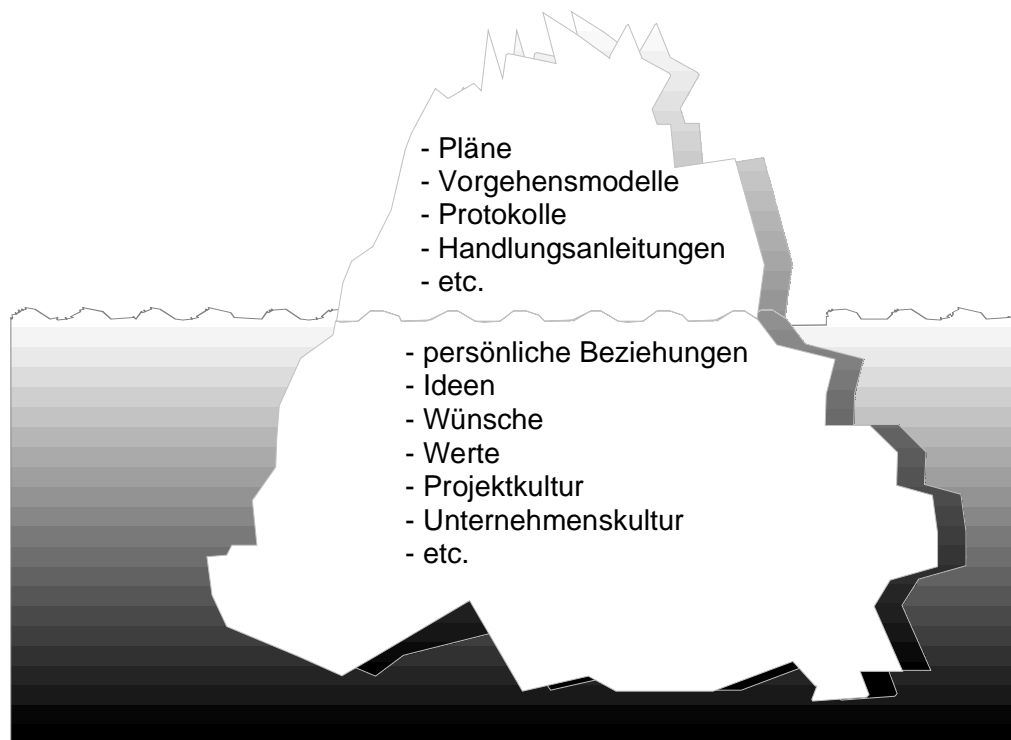


Abbildung 7.2: Die Eisberg-Theorie der Kommunikation

Die Macht des Eisberg-Teils, der unter der Oberfläche liegt ist gross. Ob sich dieser Teil zum Nutzen oder zum Schaden ihres Projektes bewegt hat erheblichen Einfluss auf den Erfolg des Projektes. Jahrzehntlang haben Verfasser von Methodenempfehlungen für die Softwareentwicklung sich darauf konzentriert, die Eisberg-Theorie zu ignorieren und lediglich immer neue Konzepte für die formelle Organisation von Softwareprojekten angeboten. In den letzten Jahren hat sich das Bild deutlich gewandelt. Leichtgewichtige und agile Ansätze legen den Schwerpunkt darauf, dass die Zusammenarbeit auf den informellen Kanälen gefördert wird. implizites Wissen (tacit knowledge) spielt für das Handeln eine größere Rolle als der durch schriftliches Niederlegen erreichbare Teil der Kenntnisse des Teams (siehe dazu auch Abschnitt 7.8 „Erfahrungen teilen“).



Maximen für
agiles Handeln



Erfolgsfaktor:

Gestalten Sie aktiv alle Ebenen der Kommunikation entsprechend der Eisberg-Theorie.

Die maximen agilen Handelns lassen sich direkt aus der Eisberg-Theorie ableiten:

- eher offen für Änderungen als starres Festhalten an Plänen
- eher Menschen und Kommunikation als Prozesse und Tools
- eher ergebnisorientiert als prozessorientiert
- eher „darüber miteinander reden“ als „gegeneinander schreiben“

- eher Vertrauen als Kontrolle
- eher „Best Practices“ aus Erfahrung als verordnete Vorgaben
- eher Angemessenheit als Extremismus

Alle diese Maximen zielen darauf ab, dass der unsichtbare Teil des Eisbergs das größere Gewicht hat.

Fr. Probst wird seit einigen Tagen das Gefühl nicht los, dass zwischen Hr. Teschner und Fr. Entrop „etwas nicht stimmt“. Die beiden sprechen kaum miteinander und gehen sich aus dem Weg. Fr. Probst beschliesst, Augen und Ohren offen zu halten, um heraus zu bekommen, was die Ursache ist. Als Fr. Probst Hr. Angelet nachträglich zum Geburtstag gratuliert erfährt sie, dass dieser dieses Jahr kein Geburtstagsgeschenk bekommen hat. Fr. Entrop war dies schrecklich peinlich. Fr. Entrop hatte sich darauf verlassen, dass sich Hr. Teschner kümmern würde. Hr. Angelet fand es überhaupt nicht schlimm und hat sich schon alleine darüber gefreut, dass alle an ihn gedacht haben. Mit diesem Anhaltspunkt weiss Fr. Probst nun mit der „dicken Luft“ besser etwas anzufangen. Nach einiger Zeit kommt Hr. Teschner zu ihr und bittet um Vermittlung. Es stellt sich heraus, dass Fr. Entrop Hr. Teschner gar nicht absichtlich aus dem Weg gegangen ist, sondern dass es sich nur zufällig so ergeben hat. Hr. Teschner glaubte jedoch, Fr. Entrop würde ihn wg. der Geschenk-Sache absichtlich schneiden ...

In unseren Projekten haben wir die Erfahrung gemacht, dass besonderes in größeren Organisationen der Widerstand gegen die Erkenntnisse der Eisberg-Theorie gross ist. Dies gilt wohlgerne nur für die nach innen gerichteten Abteilungen wie die Softwareentwicklung. In den gleichen Organisationen wird der Vertrieb oftmals in Seminaren geschult, um mit Hilfe der Eisberg-Theorie besser zu verstehen, wie Kunden gewonnen werden können.

Gerade wenn Sie kein „Naturtalent“ im Umgang mit den Eisbergen sind, helfen folgende Punkte:

- Kümmern Sie sich aktiv um beide Ebenen der Kommunikation - sowohl den direkt spürbaren als auch den nur zu erahnenden Teil.
- Behandeln Sie Ihre Projektmitarbeiter wie Ihre Kunden.
- Unterstützen Sie anderen Ihre persönlichen Ziele zu erreichen, dann werden Sie auch Ihre Projektziele erreichen.
- Halten Sie Augen und Ohren auf – nutzen Sie Ihre „Antennen“ für das unter der Oberfläche ablaufende Geschehen.

7.4 Das Team

Wer gehört alles zum Projektteam? Wie können die Projektziele gemeinsam erreicht werden?

Wer gehört dazu?

Zum Projektteam gehören mindestens Vertreter des Auftraggebers und die Mitarbeiter des Auftragnehmers, die an der Erstellung der Softwarelösung mitwirken. Es gibt aber noch eine ganze Reihe von anderen Betroffenen. Dies können z.B. sein:

- Anwender
- Lieferanten
- Partner
- Kunden

!!!



Erfolgsfaktor:

Identifizieren und beteiligen Sie alle Betroffenen, insbesondere die Anwender.

Ausrichtung des Teams auf die Ziele¹⁶

In einem gut funktionierenden Team ziehen die Menschen „an einem Strang“ in in etwa die gleiche Richtung. Tatsächlich wird es nie den Idealfall geben, dass alle Beteiligten das Projekt exakt in die gleiche Richtung vorantreiben. Persönliche Ziele, Erfahrung, Trägheit und Durchsetzungsvermögen spielen eine grosse Rolle für die Richtung und Stärke des Beitrags des einzelnen Teammitglieds (vgl. Abbildung 7.3).

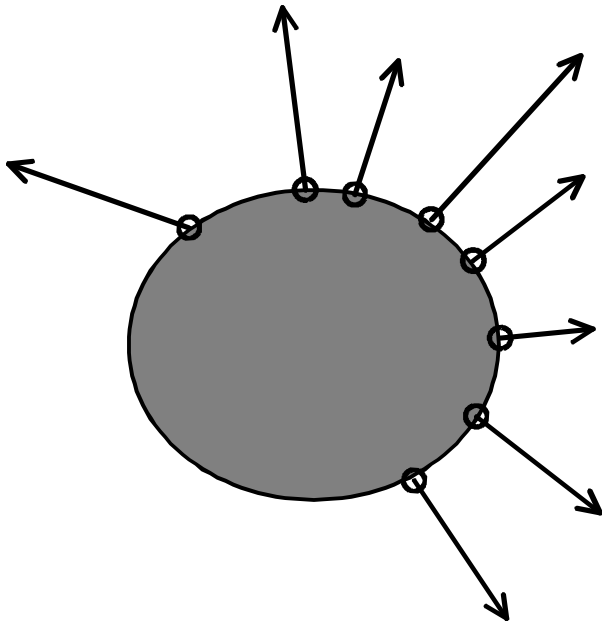


Abbildung 7.3: Durchschnittliches Team, das versucht ein rechts liegendes Ziel zu erreichen.

Es wird nie gelingen, die einzelnen Mitglieder in ihrem Verhalten radikal zu beeinflussen, aber schon eine Reihe von kleinen Änderungen können ein erheblich besseres Ergebnis bringen (vgl. Abbildung 7.4).

¹⁶ vgl. [Coc01] - Teams as Communities

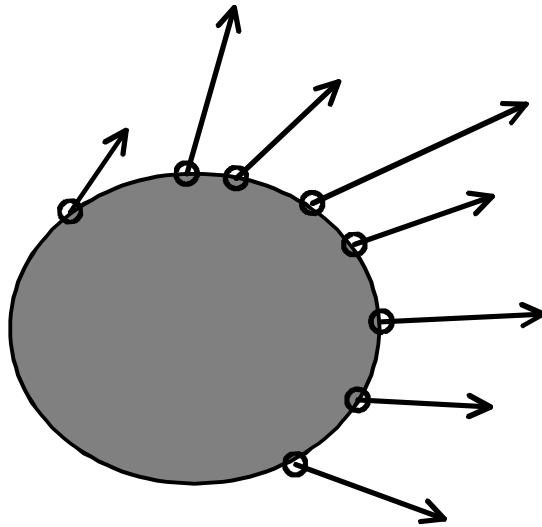


Abbildung 7.4: Ein etwas besser auf das Ziel ausgerichtetes Team.

Die Menschen bringen einen etwas besser auf das Ziel ausgerichteten Beitrag, wenn:

- sie mehr und genaueres darüber erfahren, wo das Ziel liegt
- sie die Folgen ihres eigenen Handelns spüren, und daran feststellen können in welche Richtung ihre eigenen Aktivitäten ziehen
- sie mehr und bessere Gründe haben, in die Richtung auf das gemeinsame Ziel zu arbeiten

Alle kleinen Richtungsänderungen bringen insgesamt einen erheblichen Effekt.

Schwimmen in Richtung auf das Ziel

Beim Schwimmen entsteht der Vortrieb durch den richtigen Umgang mit dem Wasserwiderstand. Damit die Schwimmzüge die grösste Wirkung entfalten, muss der Schwimmer bei jedem Zug

- möglichst viel Widerstand gegenüber dem Wasser suchen. Er „greift Wasser“ und zieht sich am Wasser nach vorne.
- möglichst viel Widerstand gegenüber dem Wasser vermeiden. Der Körper wird in einer möglichst Stromlinienförmigen Haltung belassen und der Schwimmer achtet darauf möglichst wenig Widerstandsflächen zu bieten.

In Softwareprojekten ist es nicht anders als beim Schwimmen. Es geht darum, an den richtigen Stellen Widerstand zu suchen und Widerstand zu vermeiden. Wenn Ihr Team einem Hindernis begegnet und Sie Aussagen hören wie:

- das geht nicht
- das verstehen wir nicht

- das ist nicht sinnvoll

dann können Sie darauf reagieren, in dem Sie Widerstand suchen und das Hindernis versuchen zu bezwingen:

- wie könnte es doch funktionieren?
- was ist zu tun, damit wir es verstehen?
- wie könnte es sinnvoll werden oder was fehlt uns noch, um den Sinn zu erkenne?

oder Widerstand vermeiden indem Sie nach Alternativen suchen:

- was für eine andere funktionierende Lösung gibt es?
- was für eine einfachere und verständlichere Lösung gibt es?
- was für sinnvolle Alternativen gibt es?

Entscheiden Sie sich jeweils bewusst für den in der jeweiligen Situation angemessenen Weg.

Das richtige Projektklima schaffen

Begeisterung hilft enorme Kräfte der Projektbeteiligten freizusetzen. Begeisterung ist ansteckend. Wie eine Flamme oder eine Krankheit überträgt sie sich von einem zum anderen und wird genährt oder erstickt.



Erfolgsfaktor:

Begeisterung ist ansteckend. Sorgen Sie für die Verbreitung indem Sie das richtige Klima dafür schaffen.



Schaffen und Erhalten Sie das Klima für Begeisterung. Negative Bemerkungen, Unpünktlichkeit, Entscheidungen ohne Absprache fällen – all diese Punkte gelten schnell als Kleinigkeiten. Jeder dieser und ähnlicher Punkte ist jedoch zur Eintrübung des Klimas für Begeisterung geeignet.

Erfolgserlebnisse sind das wichtigste Mittel zur Förderung eines begeisternden Klimas. Dabei ist die Regelmäßigkeit der Erfolgslebnisse wichtiger als die Größe. Gut sind viele kleine stetige Erfolgslebnisse, die nur selten von negativen Erfahrungen getrübt werden. Selbst große Erfolgslebnisse nützen nichts, wenn sie so selten sind, dass sie von vielen kleinen negativen Erfahrungen getilgt werden.

Im Team von Fr. Probst hat sich eingebürgert, dass jedes zehnte laufende Prüfbeispiel gefeiert wird. Am Anfang führte das zu einem Treffen bei Kaffee und Plätzchen pro Woche. Inzwischen gibt es jeden zweiten Tag ein Zehnerpaket zu feiern - die Belohnung variiert ständig und besonderes Fr. Entrop und Hr. Teschner entwickeln viel Phantasie dafür eine Überraschungstüte zu füllen, die derjenige, der den Testfall implementiert hat für das Team öffnen darf.

Einem Team aus der Schweiz haben wir beim Ausfüllen unseres Fragebogens zuschauen können, einer schrieb und die anderen Teilnehmer standen im Halbkreis um den Schreiber herum. Als es darum ging, den Punkt „Begeisterung“ auszufüllen kam sofort Stimmung in die Mannschaft - Anekdoten wurden über das Projekt erzählt und gemeinsam „Projektlatein“ gesponnen. Die Begeisterung des Teams war zu spüren.

Begeisterung kann man nicht verordnen. Wenn sie fehlt, dann wird das Projekt dadurch behindert und geht nur schleppend voran. Ihre eigene positive innere Einstellung ist die wichtigste Voraussetzung dafür, dass Sie andere damit anstecken können.

Offenheit der Beteiligten gegenüber neuen Vorgehensweisen

Teamgröße, Harmonie und Konfliktbereitschaft

7.5 Führung und Förderung

Kenntnisse und Fähigkeiten



*Erfolgsfaktor:
Fähige Mitarbeiter.*

Aufgabenverteilung

Die Rolle der Verantwortlichen



Erfolgsfaktor:
Förderung durch die Verantwortlichen.



Die Rolle des Projektleiters



Erfolgsfaktor:
Erfahrener Projektleiter.



Abschätzung von Aufwänden, Terminen und Kosten

Warum sind die Schätzungen von Aufwänden, Terminen und Kosten in Softwareprojekten so schlecht?

- Niemand wird für gute Schätzungen belohnt. Lob gibt es nur für unreal günstigste Wunschträume.
- Die Frage nach einer Schätzung ist oft nicht ernst gemeint - in Wirklichkeit wird vom Fragesteller die Bestätigung seiner Vorstellung erwartet
- Was eine gute Schätzung ausmacht bleibt unklar
- Es gibt keine Erfahrungswerte aus der Vergangenheit

Dennoch gilt:



Erfolgsfaktor:
Je zuverlässiger Ihre Schätzungen sind, desto besser.



!!!

Berichte über gelöste und noch vorhandene Probleme

Vertretung des Projektes nach Aussen

Projektmarketing



Erfolgsfaktor:

Vertreten Sie Ihr Projekt nach aussen mit gutem Marketing und politischem Geschick.

Ausgleich zwischen Unternehmens- und Projektinteressen

7.6 Projektplanung und Steuerung

Warum lassen sich Softwareprojekte schlechter planen und steuern als Projekte eines Handwerkers?

Der Softwareentwicklung fehlt die Sinnfälligkeit, die im Handwerk auf natürliche Weise gegeben ist. Wenn ein Handwerker seine Arbeit plant und den Fortschritt seines Werks verfolgt, dann ist dies mit dem blossen Auge möglich. Ob es sich um einen Maler handelt, der Ihre Wohnung streicht, einen Zimmermann, der einen Dachstuhl baut oder einen Mauer beim Hausbau - immer ist es auch für den Laien möglich, den Fortschritt der Arbeit mit den eigenen Sinnen zu sehen, hören, schmecken, riechen und zu fühlen. In der Softwareentwicklung sind die meisten produzierten Ergebnisse nicht direkt mit den menschlichen Sinnen erfassbar. Selbst wenn sich tausende von Dateien auf der Festplatte eines Projektes angesammelt haben, kann dieser Fortschritt nur mit Hilfe eines Computers und durch aktives Eingreifen sichtbar gemacht werden.

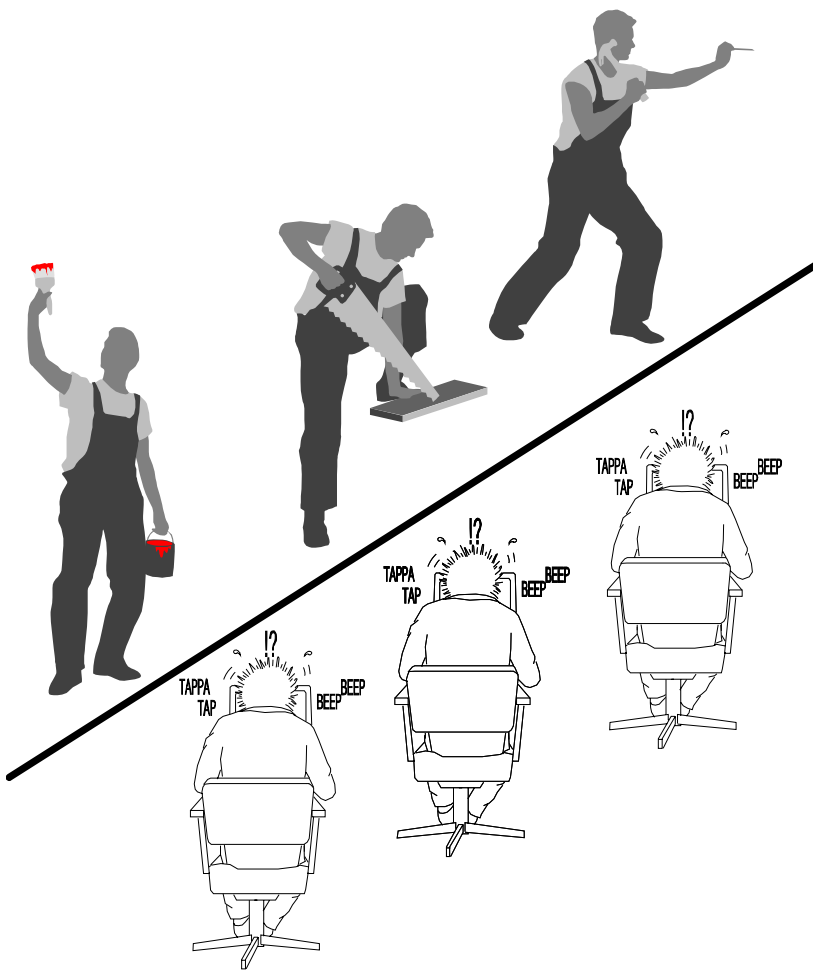


Abbildung 7.5: Sinnfälligkeit der Softwareentwicklung im Vergleich zum Handwerk

Die Planung und Verfolgung eines Softwareprojektes kann sich nur an sichtbar gemachten Ergebnissen orientieren. Schon das sichtbar machen ist allerdings mit Aufwand und Fehlerquellen behaftet. Welche Ergebnisse eignen sich am besten zur Planung?

- leicht sichtbar zu machende Ergebnisse wie Geschäftsprozesse, Anwendungsfälle, Bildschirm und Druckformulare, Anforderungen, Prüfbeispiele, Modellelemente, Programmcode, Testfälle
- leicht bewertbare Ergebnisse wie Testfälle (geht/geht nicht), realisierte Funktionen (erstellt und besteht alle Tests/noch nicht erstellt)

Planen geht nur mit Erfahrung. Stellen Sie sich vor, Sie nehmen sich ihren ersten Marathon vor und bevor Sie noch mit dem Training angefangen haben fragt Sie jemand: „Wie willst Du Dir das Rennen einteilen und welche Endzeit hast Du Dir vorgenommen?“. Sie werden entweder keine Antwort wissen oder versuchen aus Ihrer Erfahrung mit sehr viel kürzeren Strecken auf die Marathonstrecke zu schliessen. Nach einigem Training, der Beschäftigung mit dem Rennen und dem Gedankenaustausch mit erfahrenen Marathonläufern wird Ihre Antwort schon sicherer ausfallen.

Sie brauchen also zumindestens ein wenig Training mit bewerteten und sichtbar gemachten Ergebnissen, die Sie für Ihre Planung heranziehen wollen. Besser noch ist es, wenn Ihnen Vergleiche der geplanten und tatsächlichen Zahlen aus vorherigen Aufgaben vorliegen.

Wer soll planen? Natürlich diejenigen, die das Ergebnis liefern sollen. Planzahlen haben den schrecklichen Nebeneffekt, das sie mit der Übernahme von Verantwortung einhergehen. Es hat keinen Zweck anderen Planzahlen und die damit einhergehende Verantwortung „überzustülpen“. Planung ist eine hochpolitische Angelegenheit, weil sowohl die Auftraggeber als auch die Auftraggeberseite bemüht sein wird, die Planung zu ihren Gunsten zu beeinflussen.

In [Bec00] beschreibt Kent Beck als Lösungsstrategie dieses natürlichen Konflikts zwischen den Parteien das „Planungsspiel“. Es gliedert sich in die drei Phasen:

- Erforschungsphase
- Verpflichtungsphase
Die Erforschungsphase und Verpflichtungsphase entspricht dem von uns vorgestellten Bilden eines Anforderungsprofils in mit „Storycards“ vereinfachter Form.
- Steuerungsphase
In der Steuerungsphase wird iterativ gearbeitet und regelmässig der Fortschritt des Projektes bzgl. der erfüllten Anforderungen geprüft. Änderungen an den Anforderungen und die ständige Einschätzung der Entwicklungsgeschwindigkeit führen zur Anpassung des Plans in jedem Zyklus.



Erfolgsfaktor:
Planen.

Regelmässige direkte Abstimmung

7.7 Umgang mit dem Zufall

Wie gut die Planung Ihres Softwareprojektes später mit dem wirklichen Verlauf übereinstimmt ist von Zufällen abhängig es gilt „Planung ist der Ersatz des Zufalls durch den Irrtum“. Das Risiko besteht aus folgenden Punkten:

- Welche zufälligen Ereignisse können Ihr Projekt - insbesondere negativ beeinflussen

- Wie wahrscheinlich ist es, dass diese zufälligen Ereignisse eintreten?
- Wie groß ist das Ausmass der Folgen dieser Ereignisse?

Für jedes Ereignis können Sie das Risiko bewerten, in dem Sie die Auftretenswahrscheinlichkeit mit Schwere des Ereignisses multiplizieren. Wenn Sie zum Beispiel damit rechnen, dass jeweils etwa ein zwanzigstel der Mitarbeiter Ihres Projektes krank ist, dann ist die Auftretenswahrscheinlichkeit $1/20$, die Schwere des Ereignisses ist ein verlorener Arbeitstag pro betroffenem Mitarbeiter. Das Risiko beträgt also täglich $0.05 \cdot \text{Anzahl der Mitarbeiter}$, gemessen in Personentagen.

Die meisten Risiken in einem Projekt lassen sich leider nicht so schön berechnen. In diesen Fällen hilft die Gruppierung in einen der Quadranten entsprechend Abbildung 7.6 „Qualitative Risikoeinschätzung“.

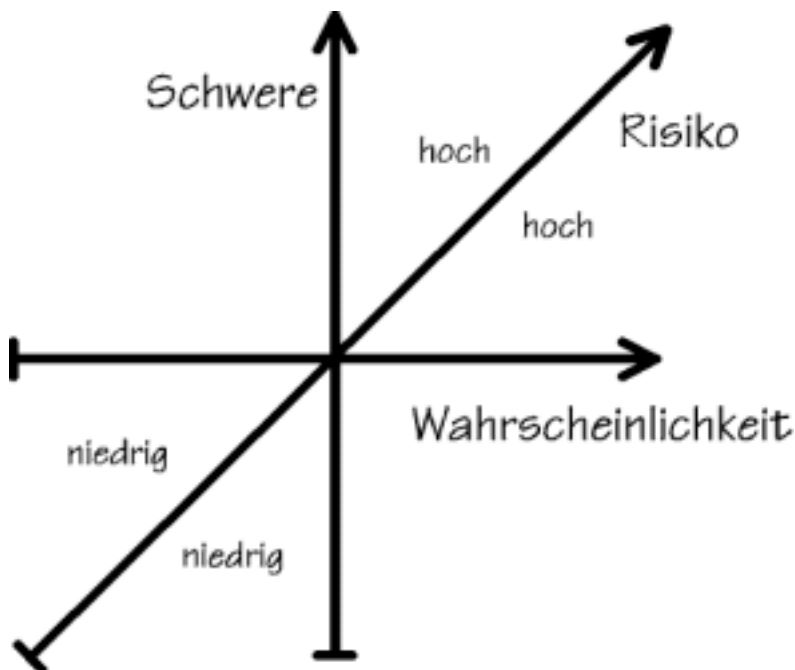


Abbildung 7.6: Qualitative Risikoeinschätzung

Wenn sie bei der Beurteilung eines Risikos im unteren linken Quadranten liegen, es also unwahrscheinlich ist, dass Sie Schaden erleiden werden oder der Schaden gering sein wird, wenn er Eintritt, dann sollten Sie sich von einem solchen Risiko nicht von Entscheidungen abhalten lassen (siehe 7.2 „Entscheidungen fällen“).



Erfolgsfaktor:

Schätzen Sie Ihre Risiken ein und verfolgen Sie diese aktiv.



Wieviel Risiko sie sich in Ihrem Projekt erlauben können hängt von Ihrem Projekttyp ab. Risikovermeidung bringt zusätzlichen Aufwand mit sich. Mit Risikovermeidung tauschen Sie also vielleicht eintretende Probleme gegen mit

Sicherheit auftretende Folgen ein. Die folgende Kurve stellt dar wie sich der Aufwand im Verhältnis zu zunehmend gewünschter Sicherheit verhält:

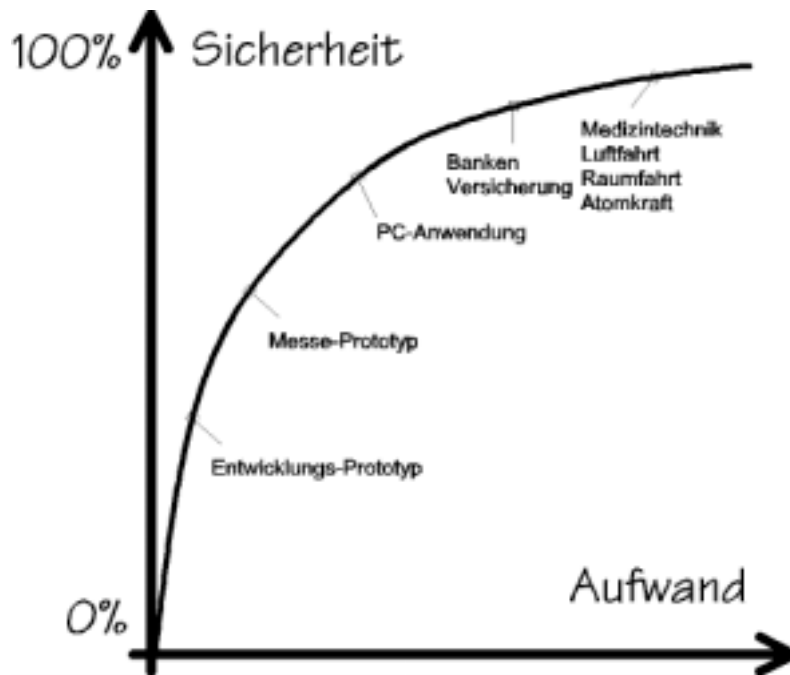


Abbildung 7.7: Aufwand im Verhältnis zur Sicherheit

Das Bedürfnis nach Sicherheit und die Bereitschaft dafür zu investieren hängt demnach stark vom Ziel Ihres Projektes und der Branche ab, in der sie agieren. Vollständige Sicherheit im Sinne von 100% gibt es nicht. Es ist gut möglich, 99,99... % zu erreichen. Aber jede „9“ zusätzlich hinter dem Komma erhöht den Aufwand um ein Vielfaches. In [Gla97] ist beschrieben, wie ein 8 Milliarden US\$ teures Projekt vor allem daran scheiterte, dass eine Ausfallrate von nur wenigen Sekunden pro Jahr gefordert wurde, d.h. eine Ausfallsicherheit von 99,99999%. Ein Mitarbeiter des Projektes dazu „die ersten drei neunen hinter dem Komma waren kein Problem, die nächsten zwei waren nicht zu schaffen ...“. Eine Ausfallrate von 99,999% bedeutet 5 Minuten pro Jahr. In fünf Minuten kann man einen Computer bequem neu starten. Die geforderte Rate von 99,99999% bedeutet einen Neustart in weniger als 5 Sekunden - das wäre nur mit unvermässig hohem Aufwand möglich gewesen. Die Verfolgung dieses Ziels wurde aufgegeben.

Was geht um Sie herum vor?

Irgendwann stellen Sie vielleicht überrascht fest, dass etwas mit Ihrem Projekt nicht in Ordnung ist. Das seltsame daran ist, dass alles mit Ihrem Projekt in Ordnung zu sein scheint - Sie liegen im Plan, im Projekt sind alle zufrieden - es geht voran. Das Besorgniserregende ist, dass sich um das Projekt herum in-

zwischen soviel verändert hat, dass Ihr Projekt dadurch gefährdet ist. Beispiele dafür können sein:

- Ihr Unternehmen fusioniert mit einem anderen Unternehmen, dort gibt es bereits die Lösung an der Ihr Projekt gerade arbeitet.
- Als das Projekt gestartet wurde, hat keiner so recht an den Erfolg geglaubt, jetzt ist abzusehen, dass der Bedarf erheblich grösser ist, als angenommen.
- Der Projekttitel lautet „ProjectWare 2003“, als aktuelles Schlagwort für Ihre Thema hat sich aber „TimeTracing“ durchgesetzt.
- Der Auftraggeber für Ihr Projekt hat sich entschieden, die Zuständigkeiten neu zu organisieren. In Zukunft werden Ihre Ansprechpartner in drei verschiedenen Bereichen arbeiten, ein Bereich wird dabei als selbständiges Unternehmen ausgegliedert.
- Die Technologie, die als Plattform für Ihre Lösung dient, wird vom Hersteller vom Markt genommen.

Diese Beispiele zeigen, dass es rund um das Projekt viele Veränderungen geben kann, die Ihr Projekt zwar nur indirekt, dafür aber umso massiver, betreffen können. Verkriechen Sie sich also nicht in einen „Elefenbeinturm“ sondern gehen Sie mit den Veränderungen aktiv um:



Erfolgsfaktor:

Beachten Sie das Umfeld Ihres Projektes und reagieren Sie rechtzeitig und angemessen darauf.



Spencer Johnson hat eine schöne Parabel über den Umgang mit Veränderungen geschrieben [Joh00]. Er beschreibt wie Mäuse und Menschen unterschiedlich reagieren, wenn man ihnen Ihren Käse wegnimmt. Während die Mäuse schnell loslaufen und sich neuen Käse suchen verbringen die Menschen noch viel Zeit mit Jammern und Lamentieren. Machen Sie es wie die Mäuse - riechen Sie täglich an Ihrem Käse, ob er noch frisch ist und reagieren Sie auf Veränderungen in Ihrem Projektumfeld rasch und ohne zu klagen.

Beim Zwischenreview nach zwei Monaten Projektlaufzeit verkündet Hr. Aufenberg freudig, das die Gründer AG ihren Firmensitz nach Frankfurt verlegen wird. Damit soll größere Nähe zu den Hauptkunden in der Bankbranche erreicht werden. Hr. Exner ist das Entsetzen ins Gesicht geschrieben - offensichtlich befürchtet er, dass er vor seinem Ruhestand noch einmal umziehen muss. Bei den Mitarbeitern von PV-Soft löst die Nachricht ebenfalls Unruhe aus. Das Projekt selbst scheint noch nicht betroffen, da der Umzug erst in 6 Monaten erfolgen soll. Was wird aber in der Wartungsphase geschehen? Wird es dann häufiger nötig sein nach Frankfurt zu fliegen? Insbesondere Fr. Teschner ist bei dem Gedanken mulmig zu, da sie ungern fliegt.

In den nächsten Tagen zeigt sich die positive Seite dieser Nachricht: die geforderte Eigenschaft „Internetfähigkeit“ des Projektverwaltungssystem war bisher von Hr. Exner und den anderen Gründer AG Mitarbeitern sehr stiefmütterlich behandelt worden. Meistenteils wurde dies als „neumodischer Schnickschack“ oder als völliger Unsinn angesehen, da ja alle Gründer - Mitarbeiter bisher über das hauseigene Computernetz direkt miteinander verbunden waren. Mit dem

bevorstehenden Verteilen auf mehrere Standorte und der persönlichen Betroffenheit der Mitarbeiter wandelt sich die Einstellung zu dem Thema radikal - nun steht dieser Punkt in vielen Gesprächen im Fokus. Fr. Probst beschliesst beim nächsten Projektreview darauf besonders einzugehen.

Die rechtzeitige Reaktion auf Änderungen im Projektumfeld kann für Ihr Projekt überlebenswichtig sein.

7.8 Erfahrungen teilen

Wie stellen Sie sicher, dass die Informationen, Erkenntnisse und Erfahrungen über die Sie im Projekt und in Ihrer Organisation verfügen nutzbringend eingesetzt werden?

Wissen ist die wichtigste Voraussetzung für Ihre Problemlösungskompetenz. Sie finden es in den folgenden Formen vor:

- **Explizites Wissen**
Schriftlich festgehaltenes Wissen, das in Form von Büchern, Zeitschriften, Video, CDs, Datenbanken, Internet und anderen Medien mehr oder weniger schnell zugreifbar ist. Dieses Wissen ist bereits kommuniziert und zugänglich gemacht.
- **Implizites Wissen**
Stillschweigendes Wissen der Beteiligten, das sich aus (gemeinsamen) Erfahrungen ergibt. Dieses Wissen lässt sich nur schwer und mit großem Aufwand erfassen und in explizites Wissen überführen.
- **Privates Wissen**
Das Wissen jedes einzelnen, das sein spezifisches Weltbild prägt.
- **Kollektives Wissen**
Ausgetauschte oder gemeinsam gemachte Erfahrungen, die innerhalb der gesamten Gruppe bekannt und verfügbar sind.
- **Strukturiertes Wissen**
Systematisch auffindbares Wissen, das leicht zugreifbar und durch seine Struktur gegliedert ist.
- **Unstrukturiertes Wissen**
Dokumente, die unstrukturiert abgelegt sind, wie Berichte und Notizen sowie nicht dokumentiertes implizites Wissen.



Erfolgsfaktor:

Betreiben Sie Wissensmanagement: sorgen Sie dafür, dass Erfahrungen gemacht, ausgetauscht und genutzt werden.

In unseren Projekten haben wir die besten Erfahrungen mit einem unkomplizierten Austausch von Informationen gemacht. Kernpunkte dafür sind:

- gemeinsame elektronische Ablage von Dokumenten
- ein leicht zugängliches einheitliches elektronisches Diskussionforum (Projekttagbuch) mit Querverweisen zu den elektronischen Dokumenten
- Zentraler und freier Zugang zu allen anderen Medien (Bücher, Akten, Kundendokumente)

Der Schlüssel zum Erfolg liegt dabei nicht in der angewendeten Technik - entscheidend ist die Bereitschaft der Beteiligten mitzumachen. Dafür muss eine Anfangshürde überwunden werden: jede gemeinsame Ablage von Wissen ist zunächst leer und damit uninteressant. Keiner mag den ersten Schritt tun und die Ablage füllen. Es schaut auch keiner rein, weil ja noch nichts da ist. Die ersten Wochen Projektarbeit sind in dieser Hinsicht die Wichtigsten. Wenn es Ihnen gelingt in dieser Zeit genug interessantes Material zusammenzutragen. Wenn dabei möglichst von allen Beteiligten Beiträge kommen, die fair verteilt sind, dann wird der weitere Austausch von explizitem Wissen ein Selbstläufer.

Für den Austausch von impliziten Wissen ist räumliche Nähe und die Schaffung von Gelegenheiten zum ungezwungenen Gespräch wichtig. Wolfgang Fahl hat bei seinem Praktikum bei Hewlett-Packard erlebt, wie einfach eine gute Atmosphäre dafür geschaffen werden kann: jeden Morgen gab es in einer Ecke des Grossraumbüros Kaffee und Brötchen umsonst. Niemand schaute darauf, ob die Pausenzeiten eingehalten wurden - denn normalerweise entwickelten sich rund um die Kaffeepause abteilungsübergreifende Gespräche zu den Themen der Projekte. Über die Frühstücksecke war es auch wesentlich leichter einen Gespräch mit dem Entwicklungsleiter zu suchen als über seine Sekretärin einen Termin mit ihm zu ergattern ...

Der Unterschied zwischen Unternehmenskulturen ist am Umgang mit Wissen sehr leicht zu spüren. In Beratungsunternehmen finden Sie zum Beispiel oft eine Kultur vor, in der jeder Einzelne sein Wissen sorgsam hütet. Ein Austausch von Wissen findet nur äußerst ungern statt. Im Gegensatz dazu nutzen OpenSource-Projekte das Internet zum regen Austausch von Erfahrungen. Viele Details werden dort explizit und größtenteils unstrukturiert der Öffentlichkeit und damit potentiellen Beteiligten präsentiert.

7.9 Literatur

- [Bec00] K. Beck: **Extreme Programming**, Addison-Wesley, 2000
Das Manifest zum Thema extreme Programming - "XP".
- [Bro75] F. P. Brooks: **The Mythical Man-month - Essays on Software Engineering**, Addison-Wesley, 1975
Klassiker indem die Aussage "Adding more people to a late project makes it later" geprägt wurde.
- [Coc01] A. Cockburn: **Agile Software Development**, Addison-Wesley, 2001
Credo für „Agilität“ - Menschen in Softwareprojekten arbeiten demnach am Besten, wenn Sie nicht in starre Raster gepresst werden sondern entlang eigener Vorstellungen angemessen agieren können. Softwareentwicklung wird als kooperatives, zielorientiertes Spiel betrachtet.
- [Con01] L. L. Constantine: **The Peopleware Papers**, Prentice-Hall, 2001
siehe Abschnitt 1.9.
- [Gla97] L. R. Glass: **Software Runaways Monumental Software Disaster**, Pearson Professional Education, 1997
Sammlung der größten Softwarefehlschläge, die es je gegeben hat, z.B. Flughafen Denver und amerikanisches Flugüberwachungssystem. Sehr lehrreich für alle Arten von politischen Problemen in Projekten.
- [Joh00] S. Johnson: **Die Mäuse-Strategie für Manager - Veränderungen erfolgreich begegnen**, Hugendubel, 2000
Wer hat mir meinen Käse weggenommen? Eine Parabel rund um die Reaktion von Menschen und Mäusen zeigt klar auf, worauf es beim Umgang mit Veränderungen ankommt.
- [Kell01] H. Kellner: **Die Kunst IT-Projekte zum Erfolg zu führen**, Carl Hanser Verlag, 2001
Projektverantwortliche finden hier reichlich Tipps aus der Praxis, die helfen die Klippen und Untiefen eines Projektes sicher zu umfahren.
- [You97] E. Yourdon: **Death March**, Prentice-Hall, 1997
Beantwortet die Frage „Wie kann man als Softwareentwickler Projekte mit unmöglichen Zielen überleben?“

„Die meisten Menschen haben einen heiligen Respekt vor Worten, die sie nicht begreifen können und betrachten es als Zeichen der Oberflächlichkeit des Autors, wenn sie ihn begreifen können.“
Albert Einstein

8

Computerfachchinesischlexikon

Computer-Fachbegriffe, -Anglizismen und -Abkürzungen mit ihrer Übersetzung in möglichst einfache deutsche Sprache











Begriff	Erklärung
Algorithmus	Vorschriften zur Lösung meist mathematischer oder technischer Verfahren. Eine Art Kochrezept für die Lösung eines Teilproblems, das einem Computer gestellt werden soll. Der Algorithmus beschreibt detailliert und in einzelnen Schritten, wie aus den Vorgaben eines Problems die erwarteten Ergebnisse erzeugt werden. Euklid (ca. 300 v. Chr.) gilt als der „Erfinder“ des algorithmischen Ansatzes – der Begriff nimmt auf den arabischen Mathematiker Al-Kowarizmi (ca. 800 n. Chr.) Bezug.
API	Application Program Interface - Zugriffsmöglichkeit für Anwendungsprogramme auf Funktionen einer Komponente über vordefinierte Schnittstellen.
Assembler	Ein Assembler (to assemble := zusammensetzen) ist ein Programmierwerkzeug mit dem sehr maschinennah programmiert werden kann. Im Gegensatz zu Programmiersprachen wie COBOL, C, Basic oder Pascal entspricht ein Ausdruck in Assembler mehr oder minder einer Maschinen-Anweisung, nur eben in von Menschen lesbarer Form.
Bug	Fehler
Bug Report	Fehlermeldung
Business Object	Geschäftsobjekt
CASE	Computer Aided Software Engineering - Computer unterstützte systematische Softwareentwicklung
Change Request	Änderungswunsch
Divide & Conquer	Teile und Herrsche - ein aus der Politik übernommener Begriff. Durch zerlegen eines Problems in kleinere Teile verringern sich die Widerstände zur Problemlösung. Die kleineren Probleme sind leichter beherrschbar als das ganze Problem. So lässt sich schrittweise ein großes zunächst möglicherweise unlösbar erscheinendes Problem bewältigen.
Feature	Eigenschaft
Framework	Ein Framework (abgeleitet aus dem englischen Begriff für Drahtgittermodell) bildet nur das Grundgerüst einer Softwarestruktur, ohne Inhalte oder Oberflächen zu beinhalten. Es besteht aus anpassbaren oder erweiterbaren Klassen. Als Modellform sind Frameworks sehr breit verwendbar, es ist jedoch die Anpassung an das spezielle Problem erforderlich.















Begriff	Erklärung
IDE	Integrated Development Environment - integrierte Entwicklungsumgebung
J2EE	Java 2 Enterprise Edition Eine Erweiterung der Java 2 Standard Edition (J2SE) zur Entwicklung und für den Betrieb von unternehmensweiten Applikationen.
Knowledge Management	Wissensmanagement Systematischer Ansatz, mit dem Informationen, Erkenntnisse und Erfahrungen (Wissen) geschaffen, gebündelt, verbreitet und ausgetauscht werden. Ziel ist, das Wissen möglichst gewinnbringend zu verwerten und zum Nutzen der Organisation einzusetzen..
MDA	Model Driven Architecture - Modell getriebener Architekturansatz
Patterns	Entwurfsmuster. Beschreibung allgemein anerkannter und weit verbreiteter Muster für den Entwurf von Software in standardisierter Form (meist UML).
Open source	Software deren Quellcode offen gelegt ist und anderen Entwicklung sich jeder Beteiligten kann, der möchte und die Regeln einhält, die von dem Rechteinhaber festgelegt worden sind. Die Software unterliegt einem entsprechend angepassten Urheberrecht.
PDM	Platform dependend Model - von der Zielumgebung abhängiges Modell
PIM	Platform independent model - von der Zielumgebung unabhängiges Modell
Release planning	Planung der nächsten Auslieferung über ein vorgegebenes Anforderungsprofil
Software Configuration Management	Die Verwaltung und Steuerung von Softwareversionen zur Prüfung und Freigabe.
TBD	to be defined - noch nicht geklärt Sachverhalt - eine offene Frage
Tool	Werkzeug
UML	Unified Modelling Language
Use Case	Anwendungsfall - Ein Anwendungsfall beschreibt einen abgegrenzten Bereich von Aktionen eines Systems aus der Sicht seiner Anwender, die für die Anwender zu einem wahrnehmbaren Ergebnis führen. Ein Anwendungsfall wird stets durch einen Anwender initiiert. Anwendungsfälle werden vollständig beschrieben.
WYSIWYG	What you see is what you get - Die Anzeige während der Arbeit entspricht dem später zu erwartenden Ergebnis
XP	extreme Programming - ein leichtgewichtiger Ansatz zur Softwareentwicklung der auf Vermeidung von Überflüssigem Ballast und enge Kooperation der Beteiligten setzt

Anhänge

9.1 Liste der Erfolgsfaktoren




Erfolgsfaktoren der Umfrage

Seite	Kurzbezeichnung	Beschreibung	Relevanz laut Umfrage		
			Wert	Rang	Symbol
84	Frameworks einsetzen	Der Einsatz von Frameworks, Komponenten und Klassenbibliotheken verkürzt ihre Entwicklungszeit und macht Ihre Lösung zukunftssicherer.	1,5	1	
33	Anforderungen aufnehmen	Nur wer rechtzeitig sagt, was er will, kann sicher gehen, dass er es auch bekommt.	1,4	2	
41	Beispiele dokumentieren	Nur das, was ich prüfen kann, kann ich auch beurteilen.	1,3	3	
42	Anforderungen priorisieren	Die richtig priorisierte Auswahl der Anforderungen legt das zu realisierende Anforderungsprofil fest.	1,3	4	
70	Komponenten einsetzen	Der Einsatz von Frameworks, Komponenten und Klassenbibliotheken verkürzt ihre Entwicklungszeit und macht Ihre Lösung zukunftssicherer.	1,3	4	
72	Modularisierung (Teile und Herrsche)	Zerlegen Sie Problem und Lösung schrittweise in kleinere von einander möglichst unabhängige Teile.	1,2	6	
103	Entscheidungen fällen	Treffen Sie ihre Entscheidungen entsprechend dem Risiko das damit zusammenhängt und trennen Sie die Klärung und die eigentliche Entscheidung.	1,2	6	
104	Kommunikation	Gestalten Sie aktiv alle Ebenen der Kommunikation entsprechend der Eisberg-Theorie.	1,2	6	
99	Iteratives Vorgehen	Schrittweise iterative oder inkrementelle Näherung an das Projektziel.	1,2	6	
Fehler! Text mark e nicht	Team	Fehler! Verweisquelle konnte nicht gefunden werden.	1,2	6	

Seite	Kurzbezeichnung	Beschreibung	Relevanz laut Umfrage		
			Wert	Rang	Symbol
definiert.					
24	Sprache des Auftraggebers sprechen	Sprechen Sie als Auftragnehmer die Sprache Ihres Auftraggebers!	1,2	6	
70	Klassenbibliotheken verwenden	Der Einsatz von Frameworks, Komponenten und Klassenbibliotheken verkürzt ihre Entwicklungszeit und macht Ihre Lösung zukunftssicherer.	1,2	6	
109	Begeisterung	Begeisterung ist ansteckend. Sorgen Sie für die Verbreitung indem Sie das richtige Klima dafür schaffen.	1,2	6	
52	Modellieren	Erstellen Sie ein Modell der Realität.	1,1	14	
20	Aufgabe, Ziel und Vorgehen klären	Klären Sie zu Beginn des Projektes und bei Bedarf im weiteren Verlauf Aufgabenverständnis, Ziele und Vorgehen.	1,1	14	
75	Patterns verwenden	Vereinfachen und standardisieren Sie Ihre Entwürfe durch die Verwendung von Entwurfsmustern.	1,0	16	
117	Projektumfeld beachten	Beachten Sie das Umfeld Ihres Projektes und reagieren Sie rechtzeitig und angemessen darauf.	0,9	17	
	Planen		0,9	17	
22	Klare Zieldefinition	Definieren Sie die Projektziele klar anhand von messbaren Kriterien und festgelegten Werten.	0,9	17	
118	Wissensmanagement betreiben	Betreiben Sie Wissensmanagement: sorgen Sie dafür, dass Erfahrungen gemacht, ausgetauscht und genutzt werden.	0,9	17	
54	Objektorientierung	Verwenden Sie durchgängig den objektorientierten Ansatz.	0,8	21	
95	Versionsverwaltung	Setzen Sie eine Ihren Bedürfnissen entsprechende Softwareversions- und Konfigurationsverwaltung ein.	0,8	21	
87	Werkzeuge	Wählen Sie Werkzeuge systematisch nach denen von Ihnen festgelegten Kriterien aus.	0,8	21	
115	Risikomanagement	Schätzen Sie Ihre Risiken ein und verfolgen Sie diese aktiv.	0,8	21	

Seite	Kurzbezeichnung	Beschreibung	Relevanz laut Umfrage		
			Wert	Rang	Symbol
73	Problem und Lösung trennen	Lösen Sie technische und fachliche Probleme getrennt voneinander.	0,8	21	
	Information Hiding (Kapselung)		0,8	21	
	Abstrahieren		0,8	21	
	Redundanzen vermeiden		0,7	28	
47	Änderungen berücksichtigen	Stellen Sie sich aktiv darauf ein, dass es zu Änderungen der Anforderungen kommen wird und berücksichtigen Sie diese so früh wie möglich!	0,7	28	
80	Änderungen berücksichtigen	Verfolgen Sie die Fehlermeldungen, Änderungswünsche und Bitten um Hilfe Ihrer Anwender systematisch.	0,7	28	
	Durch Messen verfolgen können		0,7	28	
18	Vorgehensmodell	Einigen Sie sich auf ein für Ihr Projekt geeignetes Vorgehensmodell und wenden Sie dieses aus Überzeugung an.	0,6	31	
100	Verbindlichkeit	Seien Sie verbindlich - machen Sie nur Versprechungen, die sie auch halten können. Halten Sie sich an Ihre Versprechen!	0,5	32	
	Computerrealität berücksichtigen		0,5	32	
84	Infrastruktur	Kümmern Sie sich rechtzeitig um eine adäquate und standardisierte Infrastruktur/Arbeitsumgebung für Ihr Softwareprojekt.	0,4	33	

Praxistipps der Autoren

Seite	Kurzbezeichnung	Beschreibung	Relevanz
			Symbol
12	Problem vom Menschen und vom Computer her betrachten	Betrachten Sie das zu lösende Probleme aus den zwei Blickwinkeln: 1. Vom Menschen aus 2. Vom Computer aus	
112	Projektmarketing	Vertreten Sie Ihr Projekt nach aussen mit gutem Marketing und politischem Geschick.	
62	Verständlichkeit	Achten Sie auf die Verständlichkeit Ihrer Spezifikationen und Modelle.	

Erfolgsfaktoren der CHAOS-Studie

Seite	Kurzbezeichnung	Beschreibung	Relevanz laut Studie		
			Prozentwert	Rang	Symbol
47	Förderung	Förderung durch die Verantwortlichen.	18%	1	
106	Beteiligung der Anwender	Identifizieren und beteiligen Sie alle Betroffenen, insbesondere die Anwender.	16%	2	
111	Erfahrener Projektleiter	Erfahrener Projektleiter.	14%	3	
22	Klare Zieldefinition	Definieren Sie die Projektziele klar anhand von messbaren Kriterien und festgelegten Werten.	12%	4	
42	Anforderungen priorisieren (zum Minimalumfang)	Die richtig priorisierte Auswahl der Anforderungen legt das zu realisierende Anforderungsprofil fest.	10%	5	
84	(Standard) Software infrastruktur	Kümmern Sie sich rechtzeitig um eine adäquate und standardisierte Infrastruktur/Arbeitsumgebung für Ihr Softwareprojekt.	8%	6	
48	Stabile Rahmenanforderungen	Achten Sie darauf, dass die Rahmenanforderungen für Ihr Projekt stabil bleiben.	6%	7	
18	Verbindliches Vorgehen	Einigen Sie sich auf ein für Ihr Projekt geeignetes Vorgehensmodell und wenden Sie dieses aus Überzeugung an.	6%	8	
47	Zuverlässige Schätzungen	Je zuverlässiger Ihre Schätzungen sind, desto besser.	5%	9	
110	Fähige Mitarbeiter	Fähige Mitarbeiter.	5%	10	

9.2 Fragebogen

Hier finden Sie den im Vorwort erwähnten Fragebogen, wie ihn Frau Probst ausgefüllt hat:

In Softwareprojekten erfolgreich zu sein, ist oft ein schwieriges Unterfangen. Von den Erfahrungen anderer zu lernen, ist deshalb umso wichtiger. Unter www.b-agile.de (!!!) sammeln wir Erfolgsrezepte, die sich in Ihren Projekten als entscheidend herausgestellt haben. Über Ihren Beitrag würden wir uns freuen und die Ergebnisse stellen wir im Web öffentlich vor.

Bitte verwenden Sie fünf Minuten Ihrer wertvollen Zeit um einige repräsentative Projekte, an denen Sie beteiligt waren, nach deren Erfolg zu bewerten.

Bitte füllen Sie diesen Fragebogen für bis zu drei Ihrer Projekte aus.

Angaben zu den Projekten

	Projekt 1	Projekt 2	Projekt 3
Größe (in Personenjahren)	3		
Branche	IT		
Auftrag oder Produkt?	Auftrag		
Intern oder extern beauftragt?	extern		
Zielplattform	Internet		
Besonderheiten	Ablösung COBOL- Altsystem		

Weitere Angaben zur Beschreibung meines Projekts

1.			
2.			
3.			

Bitte bewerten Sie im *nach Schulnoten* (trifft 1: vollständig zu – 6: gar nicht zu).

Mein Projekt ist / war

Erfolgreich	2		
-------------	---	--	--

Weil ...

Termin gehalten	2		
Kosten eingehalten	3		
Anforderungen erfüllt	1		

Weitere zur Messung des Projekterfolgs herangezogene Kriterien:

1. Kundenzufriedenheit	1		
2. Mitarbeiterzufriedenheit	2		

Bitte verwenden Sie die folgende Bewertung für den Einfluss auf das Projekt:
 ++ stark gefördert, + gefördert, 0 nicht beeinflusst, - behindert, -- stark behindert

Folgende Erfolgsrezepte haben das Projekt beeinflusst

	Projekt 1		Projekt 2	
	Angewendet (✓/-)	Einfluss	Angewendet (✓/-)	Einfluss
Anforderungen aufnehmen	✓	++		
Beispiele dokumentieren	✓	++		
Anforderungen priorisieren	✓	++		
Änderungen berücksichtigen	✓	ö		
Problem und Lösung trennen	✓	+		
Aufgabe, Ziel und Vorgehen klären	✓	++		
Modellieren	✓	++		
Komponenten einsetzen	✓	+		
Knowledge Management betreiben	-			
Objektorientierung	✓	+		
Information Hiding (Kapselung)	✓	++		
Klare Zieldefinition	✓	++		
Iteratives Vorgehen	✓	+		
Abstrahieren	✓	0		
Divide & Conquer (Modularisierung)	✓	++		
Redundanzen vermeiden	✓	0		
Durch Messen verfolgen können	✓	+		
Infrastruktur	✓	0		
Versionsmanagement	✓	+		
Werkzeuge	✓	0		
Risikomanagement	✓	0		
Frameworks einsetzen	-			
Patterns verwenden	✓	+		
Klassenbibliotheken verwenden	✓	++		
Planen	✓	+		
Kommunikation	✓	++		
Entscheidungen fällen	✓	++		
Verbindlichkeit	✓	0		
Begeisterung	✓	++		
Team	✓	++		
Vorgehensmodell	✓	+		
Projektumfeld beachten	✓	+		
Computerrealität berücksichtigen	-			
Sprache des Auftraggebers sprechen	✓	++		

9.3 extreme Programming

Best Practices

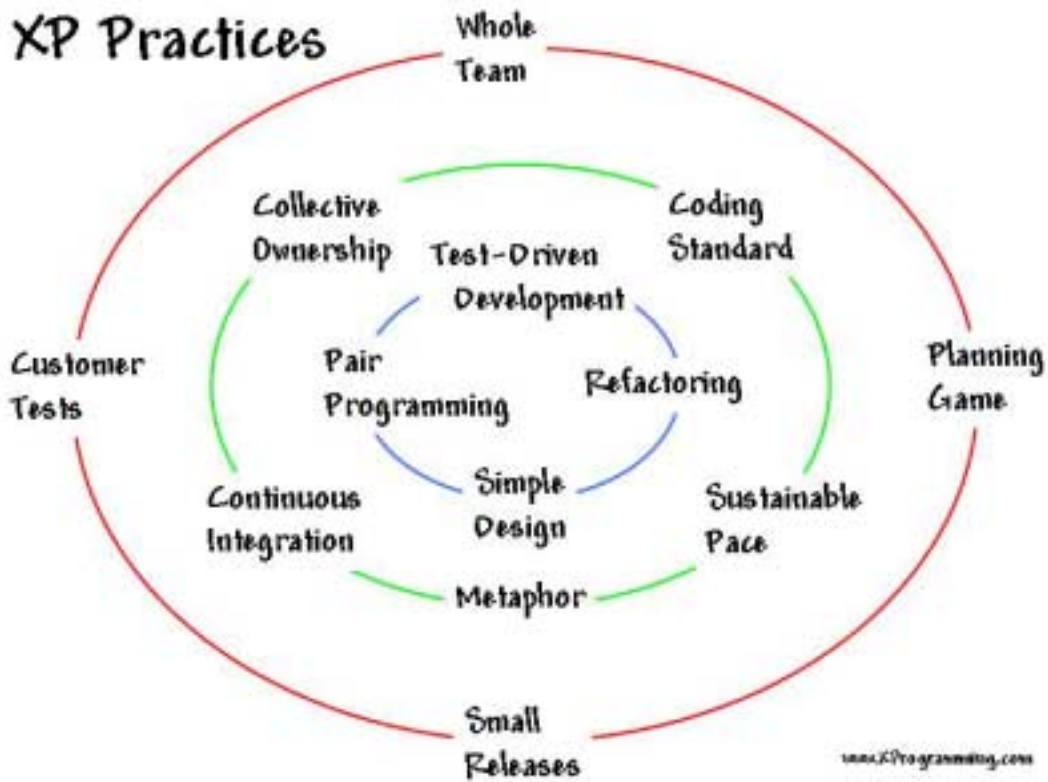


Abbildung 9.1 Die Regeln für extreme Programming

9.4 Internet Referenzen

- [Url5] http://www.standishgroup.com/chaos_chronicles/index.php
Liste der Erfolgsfaktoren, die von der Standish Group durch Untersuchung von über 35.000 Projekten seit 1994 zusammengetragen wurden.
- [Url6] http://www.sdm.de/dt/tec/eve/2001/ppts/f_3_parnas.pdf
Kurzweilige Beschreibung von David Parnas selbst, wie er auf das Information Hiding gekommen ist und wie dies noch heute Bedeutung behält.
- [Url7] <http://beat.doebe.li/bibliothek/>
Interaktive Literaturliste, Begriffslexikon und Fragenkatalog
- [Url8] <http://webquality.ethz.ch/verstaendlichkeit/>
Zur Buchreferenz siehe [Lan99] in Abschnitt 3.6, Online-Kurs und Austauschplattform zum Thema „Verständlichkeit“
- [Url9] <http://webquality.ethz.ch/verstaendlichkeit/>
Buchreferenz, Online-Kurs und Austauschplattform zum Thema „Verständlichkeit“
- [Url10] <http://www.esi.es/VASIE/>
“Best Practices“ Sammlung des Europäischen Softwareinstituts ESI.
- [Url11] <http://www.xprogramming.com>
Alles über extreme Programming (XP).
- [Url12] <http://www.well.com/user/smalin/miller.html>
Der Artikel [Mil56] von George Miller über die 7 ± 2 Regel mit freundlicher Genehmigung des Autors im Internet veröffentlicht.

9.5 Literaturverzeichnis

- [Con01] L. L. Constantine: **The Peopleware Papers**, Prentice-Hall, 2001
- [Jol193] R. L. Jolles: **How to run Seminars and Workshops**, John Wiley & Sons, 1993
- [May01] H. Mayr: **Projekt Engineering**, Fachbuchverlag Leipzig, 2001
- [Mil56] G. A. Miller: **The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information**, The Psychological Review, 1956, vol. 63, pp. 81-97
- [Par91] D. L. Parnas, G. Asmis, and J. Madey: **Assessment of Safety-Critical Software in Nuclear Power Plants**. Nuclear Safety 32(2), 1991
- [Weiz78] J. Weizenbaum: **Die Macht der Computer und die Ohnmacht der Vernunft**, suhrkamp, 1978

9.6 Index

- abstrahieren 57
- Aktion* 41, 57
- Algorithmus 58, 121
- Änderungswunsch 80, 121, 126
- Anforderung 32
- Anforderungsaufnahme 14
- Anforderungsprofil 34, 42
- Anwendungsfall 17, 66, 122
- Assembler 64
- Attribut 59
- Aufgabe 19
- Auftrag 14
- Auftraggeber 12
- Auftragnehmer 12
- Begeisterung 109
- Begriffe → Glossar
- Beispielprojekt 8, 10, 13, 16, 18, 20, 21, 23, 24, 54
- Beziehung 59
- Bindung 72
- Brücke 2, 13, 16, 25
- Brückenmetapher → Brücke
- Bug → Fehler
- Bug Report → Fehlermeldung
- Business Object → Geschäftsobjekt
- CASE 83
- Change Request → Änderungswunsch
- CHAOS-Studie 5, 128
- Divide & Conquer → Teile und Herrsche
- Einfachheit 62
- Eisberg-Theorie 103
- Entscheidung 39
- Entwurfsmuster 75, 122
- Erfolgsfaktor** 18
- Erstinterview 16
- Erwartetes Ergebnis* 41, 58
- Erwartungen 14, 100
- Erwartungsmanagement 27, 100
- Feature → Eigenschaft
- Fehler 80, 126
- Fehlermeldung 80, 126
- Frameworks 70, 76, 124, 125
- Geheimnisprinzip 72
- Geschäftsobjekt* 17, 31
- Geschäftsprozess 31
- Geschäftsprozessanalyse 14, 31
- Gliederung 62, 63
- Glossar 24
- IDE 83, 122
- implizites Wissen 104
- Implizites Wissen 118
- Information Hiding → Geheimnisprinzip
- Infrastruktur* 16, 83
- ISO 9000 → Qualität
- Iteratives Vorgehen 19
- Klarheit 62
- Klasse 59
- Klassenbibliothek 76
- Klassenbibliotheken 70, 124, 125
- Klassendiagramm 56
- Knowledge Management → Wissensmanagement
- Kollaborationsdiagramm 56
- Kompliziertheit → Einfachheit
- Komponenten 32, 70, 124, 125
- konzipieren 58
- Kopplung 72
- MDA *Siehe* Model Driven Architecture
- Meilensteine 99
- Model Driven Architecture 95
- Modell 52, 125
- Modellierung 59
- Modularisierung 72, 124
- Objekt 59
- objektorientierte Analyse 59
- Offene Frage 39
- Operation 59
- Ordnung 62, 63
- Patterns → Entwurfsmuster
- PDM *Siehe* platform dependent model
- PIM *Siehe* platform independent model
- Planungsspiel 114

- Prägnanz 62
- Projektauswahl →
 - Geschäftsprozessanalyse
- Projektleiter 111
- Projektmarketing 112, 127
- Projektverwaltungs-Projekt →
 - Beispielprojekt
- Prüfbeispiele 16, 38, 39, 42, 51
- Qualität 21, 25
- Rahmenanforderungen 31, 47
- Release planning 122 →
 - Anforderungsprofil
- Risiko 114
- Risikomanagement 98, 125
- Schätzung 111
- Situation* 41
- Softwarequalität → Qualität
- Softwareversion* 92
- strukturieren 57
- tacit knowledge → implizites Wissen
- TBD → offene Frage
- teile und herrsche 72, 95
- Tool → *Werkzeug*
- Umfrage 5
- UML 55, 65, 66, 91, 122
- Unified Modelling Language → UML
- Use Case *Siehe* Anwendungsfall
- Verantwortung 57
- Verbindlichkeit 14, 100
- Versionsverwaltung 94
- Verständlichkeit 41, 62, 127
- Vorgehen 19
- Vorgehensmodell 18
- Weitschweifigkeit → *Prägnanz*
- Werkzeug 83
- Werkzeugauswahl 87
- Wissensmanagement 98, 122, 125
- Wunsch 14
- WYSIWYG 122
- Ziele 19
- Zusammenhanglosigkeit
 - *Gliederung*

Die Autoren

Wolfgang Fahl



Wolfgang Fahl hat in Aachen Informatik studiert. Bei den Unternehmen Hewlett-Packard und Dräger lernte er Software mit Fokus auf Qualität zu entwickeln. Als Verantwortlicher für Methoden und Verfahren betreute er über 30 Softwareprojekte für lebenserhaltende Systeme. Seit 1996 ist er als Berater für Unternehmen in den Branchen Versicherung, Banken, Telekommunikation und Versorgung tätig. In seiner Freizeit macht er bei Volkstriathlonveranstaltungen unter der Devise „Ankommen und nicht Letzter sein“ mit. Mit Frau Maria und Kindern Sandra und Martin trifft man ihn am

Niederrhein auch schon mal auf einer Radtour an.

Sie erreichen Ihn unter wf@bitplan.com.

Frank Hoffmann



Frank Hoffmann studierte Informatik an der Technischen Universität in Berlin, wo er auch seine ersten 28 Lebensjahre verbrachte. Seit 1990 beschäftigt er sich mit objektorientierten Methoden. Seine berufliche Karriere begann er deshalb auch bei microTOOL, einem der führenden Toolanbieter. Mit der stärkeren Fokussierung auf das Beratungs- und Trainingsgeschäft verschlug es ihn nach Nordrhein-Westfalen zunächst zu Rösch Consulting. 1999 gründete er zusammen mit Wolfgang Fahl und Martin Gummersbach mit der BITPlan GmbH ein eigenes Beratungsunternehmen. Seitdem hat er in zahlreichen

Softwareprojekten Objektorientierung eingeführt und umgesetzt. Zu seiner Passion gehören Trainings, in denen er sein Wissen an mehrere Hundertschaften wissensdurstiger Teilnehmer weitergegeben hat. In seiner Freizeit erwischen Sie ihn entweder vor der Stereoanlage beim Hören guter Musik oder - bei entsprechendem Wetter - auf einem der vielen Seen, auf denen er mit seiner Frau Chris segelt.

Sie erreichen Ihn unter fh@bitplan.com.

Erfolgsfaktoren der Softwareentwicklung

Eine Umfrage unter Software-Projektleitern untermauert es: Es sind immer wieder die gleichen Faktoren, die zum Erfolg von Software-Projekten beitragen und deren Fehlen das Scheitern von Projekten verursacht. Basierend auf dieser Umfrage und auf ihrer jahrelangen Erfahrung haben die Autoren in diesem Buch die Erfolgsfaktoren zusammengestellt, die in jedem Softwareentwicklungsprojekt beachtet werden sollten.

Sie stellen Best Practices vor sowohl für den gesamten Entwicklungsprozess von der Analyse bis zur Implementierung als auch für die begleitenden Managementaufgaben. „Weiche“ Faktoren, die sich auf Menschen, Management und Politik beziehen, sind dabei genauso wichtig wie z.B. die Auswahl des richtigen und angemessenen Vorgehens.

Die Autoren analysieren die Erfolgsfaktoren, zeigen, wie man sie umsetzen kann, und bieten ein durchgängiges, anschauliches Beispiel. So ist ein Leitfaden entstanden, der zahlreiche Praxistipps und Best Practices enthält und Softwareentwicklern und Projektmanagern den Weg zum Erfolg zeigt.