FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Entscheidungsprozesse in Software-Großprojekten

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Informatikmanagement

eingereicht von

Andreas Musil, BSc.
Matrikelnummer 0127104

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, 19.07.2010 _____ _____

(Unterschrift Verfasser) (Unterschrift Betreuer)

Ehrenwörtliche Erklärung

Andreas Musil

Hauptstr. 26 / c2 / 6

2351 Wr. Neudorf

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wr. Neudorf, 19.07.2010 _____

Andreas Musil, BSc.

Für Simone, meinen wertvollsten Schatz.

Für meine Eltern, die mich immer nach besten Kräften unterstützen.

Kurzfassung

Das Ziel dieser Diplomarbeit ist die Beleuchtung des Spannungsfeldes, innerhalb dessen Entscheidungen getroffen werden, die die Abwicklung von großen Software-Projekten betreffen. Die Arbeit ist in zwei Teile geteilt, wobei sich der erste Teil mit den theoretischen Grundlagen dieser Thematik befasst, während der Praxisteil mit Hilfe von angeleiteten Interviews die in der Theorie gewonnenen Erkenntnisse kontrastriert.

Der Theorieteil behandelt Themen wie bspw. die Fragen, was ein Projekt zu einem Großprojekt macht, welche Designfragen bei großen Softwareprojekten besonders im Vordergrund stehen sollten, bzw. welch weites Feld Entscheidungsprozesse darstellen.

Im Praxisteil kommen Entscheidungsträger zu Wort, die bereits über Erfahrung mit Software-Großprojekten verfügen. Die Interviews werden in einer Weise zusammengefasst, die es ermöglicht, sie als Gegenüberstellung zu den theoretischen Erkenntnissen betrachtet werden können.

Abstract

The goal of this diploma thesis is the illumination of the field, in which descisions are made during the implementation of large-scale software projects. The work is divided into two parts, whereas the first part is concerned with the theoretical basices of this topic, while the field study contrasts the theory with the aid of guided interviews.

The theoretical part handles various topics such as what are the criterions of a large-scale project compared to a standard-sized project, what kind of design issues should be handled with priority in case of huge projects, as well as what kind of activities can be imagined under the terminus decision process.

In the practical part, experienced descision makers in the field of large-scale software projects, get a chance to speek in the form of interviews. The interviews are taken together in a way that enables the confrontation with the theoretical insights.

.

Inhaltsverzeichnis

1 Einleitung

„Software is a place where dreams are planted and nightmares harvested“
– Brad Cox

Die Einleitung ermöglicht dem Leser dieser Diplomarbeit einen Überblick darüber zu erhalten, worin die Zielsetzung dieser Arbeit besteht, wie der persönliche Zugang des Verfassers zu dem Thema ist, und schließlich wie sich der Aufbau dieser Arbeit gestaltet.

1.1 Zielsetzung bzw. Fragestellung

Das Ziel der Arbeit ist das Herausarbeiten von Aspekten, die für die Entscheidungsfindung in Softwaregroßprojekten eine Rolle spielen. Die Betonung liegt hierbei auf dem Wort „groß“, da es exakt darum geht, in wieweit sich Entscheidungsprozesse in Projekten normaler Größe von Projekten unterscheiden, die eine über das normale Maß hinausgehende Dimension erreichen.

Wobei selbstverständlich die Frage zu klären ist, ab wann ein Großprojekt ein Großprojekt ist.

Die leitende Fragestellung der Diplomarbeit lautet daher:

Wie funktionieren Entscheidungsprozesse in Software-Großprojekten, und wie sieht das Umfeld der dahinterstehenden Organisationen aus?

Die damit einhergehenden Forschungsfragen sind:

- Was sind die besonderen Merkmale, Anforderungen, bzw. Schwierigkeiten bei der Durchführung von Software-Großprojekten?
- Wie sehen die unterschiedlichen Projektorganisationen zur Durchführung von Software-Großprojekten aus, und welche Vor- bzw. Nachteile kann man ihnen zuschreiben?
- Was sagen die „Stimmen aus der Praxis“ zu dieser Thematik?

Die Beantwortung dieser Fragen erfolgt im Rahmen dieser Diplomarbeit einerseits über einen theoretischen Teil, der die existierende Fachliteratur zu diesem Thema aufarbeitet, sowie über einen praktischen Teil, der – wie bereits erwähnt – ausgewählte Stimmen aus der Praxis zu Wort kommen lässt.

1.2 Zugang zum Thema

Bevor ein inhaltlicher Überblick über die Arbeit gegeben wird, soll kurz der persönliche Zugang des Verfassers zum Thema dargelegt werden.

Als Software Engineer im Umfeld der österreichischen Sozialversicherung steht der Verfasser beruflich mit Projekten in Verbindung, die augenscheinlich über das normale Ausmaß eines herkömmlichen Software-Projekts hinausgehen. Am Beispiel „e-Card“ manifestiert sich diese Augenscheinlichkeit beispielsweise daran, dass für die Umsetzung dieses Projekts eine eigene Gesellschaft gegründet wurde, die mit der Umsetzung bzw. dem Betrieb sowie der Wartung und Weiterentwicklung des Systems beauftragt wurde. Neben den operativen Dimensionen – cirka 12.000 Vertragspartner führen pro Tag an die 500.000 Konsultationen online durch – sind in einem solchen System auch die folgenden Aspekte von großer Bedeutung:

- Datenschutz- bzw. Sicherheit
- Verfügbarkeit
- Erweiterbarkeit
- Wartbarkeit
- Benutzbarkeit

Auch sind die politischen Dimensionen bei Projekten größeren Umfangs zu berücksichtigen. Mit der Größe des Projekts nimmt offenbar auch die Anzahl der Interessensgruppen bzw. der involvierten Interaktionspartner zu, wie am Beispiel „e-Card“ zu beobachten ist. Hier kann es zu Interessenskonflikten kommen, die im Sinne eines funktionierenden Gesamtsystems gelöst werden müssen.

Fasst man nun das oben genannte zusammen, liegt die Annahme nahe, dass die Architektur eines solchen Systems keine triviale Angelegenheit ist. Die Designentscheidungen – sowohl das Gesamtsystem als auch spezifische Teile des Systems betreffend – haben weitreichende Folgen, und bilden daher das Fundament eines funktionierenden und zukunftsorientierten Systems.

Falsche Entscheidungen in der Design- und Implementierungsphase führen unweigerlich in ein Desaster, das es unter allen Umständen zu vermeiden gilt.

Der Umkehrschluss des vorigen Satzes kann als Beweggrund zur Wahl dieses Diplomarbeitsthemas angeführt werden:

- Wie muss das Umfeld einer Projektorganisation strukturiert sein, um allgemein betrachtet ein Projekt von überdimensionaler Größe erfolgreich abwickeln zu können?
- Was sind die Erfolgs- bzw. Misserfolgsfaktoren, die den Ausgang eines solchen Projekts beeinflussen?
- Was sind die größten Fehler, die gemacht werden können, bzw. welche Erfolgsrezepte gibt es?

All diese Fragen – und noch weitere – sollen im Rahmen dieser Arbeit beantwortet werden.

1.3 Struktur der Arbeit

Die Arbeit beinhaltet sowohl einen theoretischen als auch einen empirischen Teil, sowie ein Conclusio, in dem die Erkenntnisse aus den beiden Teilen zueinander in Bezug gebracht werden.

Im theoretischen Teil werden die zur Beantwortung der Forschungsfragen relevanten Aspekte anhand einschlägiger Fachliteratur herausgearbeitet, womit eine Art Fundament gelegt wird, auf das der empirische Teil der Arbeit aufsetzen wird.

Der empirische Teil wird – als Kontrast zu den zuvor gewonnenen theoretischen Erkenntnissen – mittels Interviews dokumentieren, wie Entscheidungsprozesse in Software-Großprojekten in der Praxis aussehen.

Im Conclusio werden schließlich Theorie und Praxis – quasi mit den entsprechenden, aus den Vorkapiteln gewonnenen Erkenntnissen ausgestattet – rückblickend betrachtet.

Nachfolgend wird der Aufbau der Arbeit im Detail beschrieben.

Kapitel 2 behandelt bereits ein zentrales Thema dieser Arbeit, nämlich das Softwaregroßprojekt. Dieses Kapitel dient dem Zweck herauszuarbeiten, was ein Projekt zu einem Großprojekt macht. Ausgehend von der Erkenntnis, was überhaupt ein Projekt ist – unabhängig von der Größe – werden die Merkmale erfasst, die ein Großprojekt als solches auszeichnen. Neben theoretischen Gesichtspunkten darf selbstverständlich der Bezug zur Praxis nicht fehlen, was in Form von Beschreibungen bekannter Großprojekte erreicht wird.

Überlegungen zum Thema Design werden in **Kapitel 3** angestellt. Angefangen dabei, was unter Design zu verstehen ist, was in Bezug auf Design zu beachten ist, bis hin zu, welchen Stellenwert Design in Verbindung mit Softwaregroßprojekten hat, wird das Thema von verschiedenen Gesichtspunkten aus betrachtet. Auch dieses Kapitel wird mit Anekdoten aus der Praxis abgerundet, um eingängig zu vermitteln, wie wichtig die Entscheidungen Design betreffend für ein Projekt sein können.

Kapitel 4 befasst sich mit der Frage, wie Entscheidungsprozesse funktionieren. Das Treffen von Entscheidungen ist ein integraler Bestandteil im Rahmen eines jeden Projekts. Falsch getroffene Entscheidungen können in einem Desaster enden, zumindest jedoch die Qualität eines jeden Systems erheblich verschlechtern. Daher ist es von immenser Wichtigkeit zu verstehen, wie Entscheidungsprozesse funktionieren, und welche Auswirkungen das Nichtfunktionieren dieser Prozesse haben kann. Es leuchtet ein, dass mit zunehmender Größe eines Projekts die Entscheidungsprozesse immer komplexer werden, womit die Auseinandersetzung mit dieser Thematik gerade in Softwaregroßprojekten eine zentrale Rolle einnehmen sollte.

Nachdem in den vorangegangenen Kapiteln die Theorie eingehend dargelegt wurde, widmet sich **Kapitel 5** mit dem praktischen Teil der Diplomarbeit – der Befragung. In einem Leitfadeninterview kommen Projektleiter zu Wort, die bereits über Erfahrungen mit Softwaregroßprojekten verfügen. Das Thema der Interviews ist auf den vorangegangenen Theorieteil entsprechend abgestimmt, sodass die Antworten kategorisiert und entsprechend ausgewertet werden können.

Last but not least werden im Zuge des Conclusios in **Kapitel 6** die theoretischen Erkenntnisse mit den praktischen Erfahrungen in Bezug gesetzt. Das Ziel ist einen Trend zu erkennen, in welchen Bereichen sich Theorie und Praxis decken oder zumindest annähern, und in welchen Punkten die Theorie eine gut gemeinte, in der Praxis jedoch kaum umsetzbare Methode ist.

2 Das Softwaregroßprojekt

„Plan to Throw One Away“
– Frederick P. Brooks, jr.

Dieses Kapitel widmet sich unter anderem der Frage, was ein Projekt zu einem Softwaregroßprojekt macht.

2.1 Was ist ein Projekt?

Um die Besonderheiten eines Projekts größeren Ausmaßes besser verstehen zu können, muss zunächst geklärt werden, was generell unter dem Begriff „Projekt“ zu verstehen ist.

In Zuser et. al. [01, p.27f.] wird ein Projekt wie folgt charakterisiert [vgl. Tjoa97 in ebd.]:

- Ein Projekt ist ein einmaliges Vorhaben.
- Ein Projekt ist zeitlich begrenzt.
- Ein Projekt hat klare Ziele.
- In Projekten werden neuartige und unbekannte Probleme gelöst.
- Ein Projekt verwendet unterschiedliche Methoden, um die Komplexität der Aufgabenstellung geeignet zu untersuchen, darzustellen und ein zufriedenstellendes Ergebnis zu erreichen.
- Aufgrund der hohen Komplexität von Projekten und fachlicher Wissensvermischungen ist die Zusammenarbeit von Personen aus unterschiedlichen Fachgebieten erforderlich, welche mit unterschiedlichen Kenntnissen zum Erfolg des Projekts beitragen sollen.
- Ein Projekt hat ein besonderes Risiko.
- Projekte haben ein eigenes Budget, welches nicht überschritten werden kann, ohne den Projekterfolg zu gefährden.

Die Deutsche Industrienorm (DIN) 69901 formuliert etwas knapper:

> *„Ein Vorhaben, bei dem innerhalb einer definierten Zeitspanne ein definiertes Ziel erreicht werden soll, und das sich dadurch auszeichnet, dass es im Wesentlichen ein einmaliges Vorhaben ist.“ [ebd.]*

Es gibt massenweise Definitionen zu dem Begriff „Projekt“ – im Kern sind sich alle sehr ähnlich. Abschließend veranschaulicht eine Graphik den Sachverhalt eines Projekts:

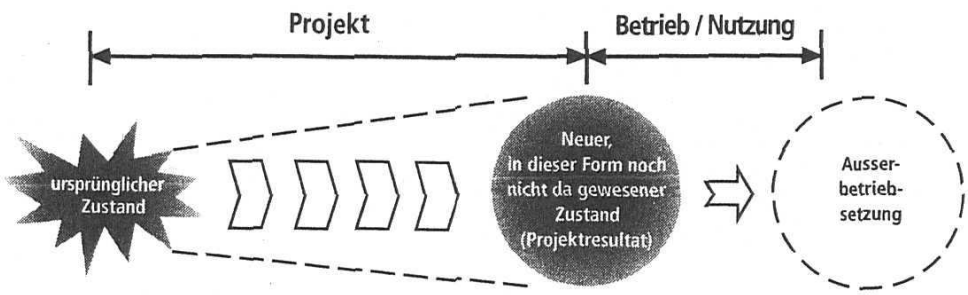


Abbildung 1: Transformationsprozess [Scheu02, p.18]

2.2 Verschiedene Typen von Projekten

Projekte lassen sich nach bestimmten Arten unterscheiden. Noe [09, p.22] nennt dabei drei Kriterien:

- Wesen des Auftraggebers
- Rolle des Auftraggebers
- Umfang des Projekts

Das **Wesen des Auftraggebers** wird dadurch bestimmt, welche Art von Projekten durchgeführt werden, also beispielsweise Non-Profit-Projekte, Wirtschaftsprojekte, Bauprojekte, Verwaltungsprojekte, Organisationsprojekte, Forschungsprojekte, Entwicklungsprojekte, Investitionsprojekte, strategische Projekte, oder auch operative Projekte.

Mit der **Rolle des Auftraggebers** beschreibt Noe, ob es sich um externe oder interne Projekte handelt. Bei externen Projekten wird ein auf die Abwicklung von Projekten (einer bestimmten Branche) spezialisiertes Unternehmen von einem anderen Unternehmen oder Behörde beauftragt, das Projekt durchzuführen. Bei

einem internen Projekt ist Auftraggeber und Durchführer ein und dasselbe Unternehmen, oft dienen interne Projekte zur Verbesserung interner Arbeitsabläufe.

Als Kriterien zur Beurteilung des **Umfangs eines Projekts** nennt Noe Projektgröße, Projektdauer, Projekttyp sowie Projektart. Diese Kriterien werden in einem späteren Kapitel näher betrachtet, wenn es um die Größe von Projekten geht.

2.3 Zum Begriff des Software Engineering

Nachdem im vorangegangenen Abschnitt das Projekt als solches allgemeinen beschrieben wurde, wird an dieser Stelle der Begriff „Software Engineering“ thematisiert.

Hellige [08, p.1] beschreibt die Entstehung des Begriffes „Software-Engineering“ wie folgt:

> *„Die in der zweiten Hälfte der 60er Jahre zutage getretene „Software-Krise“ war der Auslöser der Konferenz-Initiative, aus der die Disziplin „Software-Engineering“ hervorging.“[ebd.]*

Bei dieser Konferenz-Initiative handelte es sich konkret um eine von dem NATO Science Committee gesponserte Konferenz, die 1968 in Garmisch (D) abgehalten wurde, und die sich mit dem Themenkomplex Design, Produktion, Verteilung, und Servicing von Software befasste [vgl. SiPa03, p.337].

In dem aus dieser Konferenz hervorgegangenen Bericht wurde festgehalten, dass der Begriff „Software Engineering“ aus dem Grund angebracht scheint, da die Erstellung von Software auf Basis jener Prinzipien beruhen sollte, die sich aus den etablierten Ingenieurswissenschaften (Maschinenbau, Elektronik, etc.) ableiten lassen [vgl. NaRa68].

Doch bereits vor dieser von der NATO gesponserten Konferenz wurden in der Fachwelt Techniken diskutiert, die auf das methodische Abwickeln von Software-Projekten abzielen. So führt Hellige [08, p.3] an, dass bereits John v. Neumann Mitte der 1940er Jahre ein arbeitsteiliges 6-Phasenmodell für die Programmierung entwickelte. Wilkes, Wheeler und Gill wiederum übertrugen 1951 die aus der

Elektrotechnik gebräuchliche „modular design philosophy“ auf die Software-Konstruktion.

Rainwater [03, p.234] merkt leicht humoristisch zur Entstehung von Software Engineering an:

> *„Gerade für extrem große Software-Projekte, speziell denen, die durch die Regierung oder große Rüstungsfirmen initiiert wurden, führte dieser Ansatz (Software Engineering, Anm.) manchmal zum Erfolg und häufig zu spektakulären Fehlschlägen.“ [ebd.]*

2.4 Der Life-Cycle eines Datenverarbeitungs-Systems

Zunächst wird auf abstrakte und auch vereinfachte Weise veranschaulicht, welche Phasen üblicherweise der Life-Cycle eines Datenverarbeitungs-Systems umfasst.

An einem bestimmten Punkt wird zunächst die Idee geboren, ein System (das noch nicht existiert) zur Bewältigung einer konkreten Aufgabe einzusetzen. Nachdem die zu bewältigende Aufgabe in sämtlichen Aspekten analysiert, erfasst, und konkretisiert ist, wird mit der Planung begonnen, wie die Aufgabe gelöst werden kann. Hinter diesem „wie“ verbirgt sich bereits das grundlegende Design des zukünftigen Systems. Nachdem das Design in einem entsprechenden Detailierungsgrad vorliegt, wird mit der Implementierung begonnen. Die Implementierungsphase wird von stetigem Testen begleitet, bis an einem bestimmten Punkt das System ein Stadium erreicht hat, in dem es eingesetzt und somit in Betrieb genommen werden kann.

Die Betriebsphase wird im Regelfall von Wartungsarbeiten begleitet, und nicht selten wird das System im Laufe der Zeit mit zusätzlicher Funktionalität erweitert [vgl. Benn95, p.61]. Irgendwann wird das System nicht mehr seinen Zweck erfüllen (können), und entweder durch ein neues System ausgetauscht, oder auch ersatzlos abgeschaltet werden.

All die oben beschriebenen Phasen sind selbstverständlich nicht sequentiell chronologisch angeordnet, sondern überlappen sich, wiederholen sich, überholen sich, etc.

Maciszek [05, p.21] fasst den Life-Cycle in fünf Phasen prägnant zusammen:

- Business analysis
- System design
- Implementation
- Integration and deployment
- Operation and maintenance

2.5 Große Softwareprojekte

Die Dimension eines Projekts kann von unterschiedlichen Standpunkten aus betrachtet werden. Zuser et. al. [01, p.29f.] schlagen vor, die Größe eines Projekts primär an den Personenjahren für die Umsetzung der Anforderungen festzumachen.

Die Dauer des Projekts ergibt sich in weiterer Folge aus der Größe des Projektteams (Personenjahre dividiert durch Anzahl Projektmitglieder). Die folgende Übersicht soll die Größenordnungen veranschaulichen.

Größe	Beschreibung	Beispiel
Sehr klein	Eine Person für sich selbst	Rechenprobleme, einfache Spiele, Algorithmen
Klein	Eine Person für einen Kunden/Anwender	wie oben
Normal	Ein Team für einen Kunden und einige Anwender	Buchhaltung, Lagerverwaltung
Groß	Viele Personen für sehr viele sehr verschiedene Anwender	Compiler, Datenbank
Sehr groß	Sehr viele Personen programmieren etwas nie vorher Dagewesenes	Raumfahrt, Atomkraftwerk, elektronische Börse

Abbildung 2: Größenordnungen von Software-Projekten [ZuBi01, p.29]

Eckstein [04, p.3f.] schlägt vor, die Größe eines Projekts an Merkmalen festzumachen, die nachfolgend beschrieben werden. Bei Betrachtung dieser Merkmale ist zwischen primären und sekundären Merkmalen zu unterscheiden. Primäre Merkmale ergeben sich direkt aus den Anforderungen bzw. Rahmenbedingungen des Projekts – sekundäre Merkmale lassen sich aus den primären ableiten.

Der Umfang eines Projekts ist eine primäre Größe, die sich direkt aus der Funktionalität der Software ableiten lässt. Veränderungen am Umfang haben direkte Auswirkungen auf die nächsten beiden Merkmale – Dauer bzw. Anzahl der Mitarbeiter [vgl. ebd.].

Dauer ist eine klassische sekundäre Größe, da – um das Beispiel von Eckstein direkt aufzunehmen – kein Projekt *„mit dem Selbstzweck gestartet (wird), 20 Jahre zu dauern“* [ebd., p.3]. Die Dauer eines Projekts lässt sich direkt von der Größe anderer Dimensionen ableiten – beispielsweise vom Umfang der zu erstellenden Software. Eckstein führt weiters an, dass ein Projekt mitunter sogar unendlich lange dauern kann, weil, an einem gewissen Punkt angelangt, es niemand mehr wagt das Projekt abzubrechen [vgl. ebd.].

Als weiteres – sekundäres – Merkmal für die Größe eines Projekts können die Kosten herangezogen werden. Im Regelfall nehmen die Personalkosten den Löwenanteil an den Kosten für ein Projekt ein, sodass bei steigendem Umfang und / oder anwachsender Projektdauer automatisch auch der Faktor Kosten ansteigen wird [vgl. ebd.].

Die Anzahl der beteiligten Personen an einem Projekt ist ein Spezialfall. Zunächst erscheint es als logische Konsequenz, dass bei steigendem Umfang bzw. Risiko eines Projekts auch die Anzahl der eingesetzten Personen ansteigen wird – was auf eine sekundäre Größe hindeutet. Eckstein führt jedoch an, dass mitunter schon zu Beginn des Projekts die Anzahl der eingesetzten Personen unverhältnismäßig hoch ist, um die Wichtigkeit des Projekts zu unterstreichen [vgl. ebd.].

Schließlich zählt Eckstein noch die zu erwarteten Risiken eines Projekts zu den relevanten Merkmalen der Projektgröße. Das komplizierte an Risiken ist jedoch, dass sie sich auf sämtliche Aspekte eines Projekts beziehen können. So stellt beispielsweise das Einsetzen neuer Technologien ein Risiko dar, aber auch die schiere Größe des Projektteams kann bereits zu einem Risiko werden. Somit zählen die Risiken eindeutig zu sekundären Merkmalen der Projektgröße [vgl. ebd.].

2.6 Entstehung großer Softwareprojekte

Die Entstehung großer (Software-)Projekte wird oft durch das Lobbying bestimmter Interessensgruppen begünstigt. Speziell bei Projekten, die durch die öffentliche Hand beauftragt werden, gibt es in der Regel Organisationen, Unternehmen, Personengruppen, etc., die von der Umsetzung, bzw. dem Einsatz des Projekts profitieren, sodass oft mit großem Aufwand Lobbyarbeit für das Projekt geleistet wird.

Weltz et. al. [92, p.19] schreiben davon, dass die Rekonstruktion der Entstehungsgeschichte vieler Projekte oft ausserordentlich schwierig ist . Die Anfänge verlieren sich oft im Dunkeln, die ursprünglichen Anstöße bzw. Urheber sind nur schwer auszumachen.

Grimmer [in GoBa01, p.340] merkt an, dass die politische Einflussnahme auf die Verwaltungsmodernisierung mittels IT oft durch direkte Vorgaben (Entscheidungen) politischer Gremien geschieht. Das kann beispielsweise so aussehen, dass Verwaltungen bestimmte Ziele wie zum Beispiel Rationalisierungen etc. vorgegeben werden, die eine bestimmte technische Infrastruktur erfordern.

Wichtig ist festzuhalten, dass die Projektidee eine Vision darstellt, und noch keine konkrete Zielsetzung für einen durchzuführenden Projektauftrag darstellt [vgl. Karn02, p.42].

2.7 Ausschreibungen

> *„Seit 1993 sind in Österreich öffentliche Stellen – also insbesondere Bund, Länder und Gemeinden – dazu verpflichtet, Aufträge ab einem bestimmten Auftragswert nach einem formalisierten Verfahren (an den zu eruierenden Bestbieter) zu vergeben.“ [JaZe07, p.9]*

Jaburek et. al. [07, p.10] stellen fest, dass sich Entscheidungsträger im Zuge eines Vergabeverfahrens immer wieder mit den folgenden Fragen auseinandersetzen müssen:

- Wer unterliegt dem Vergaberecht?
- Unter welchen Bedingungen muss ich ausschreiben?

- Welche Verfahrensart kann/sollte ich wählen?
- Wie wird ein Vergabeverfahren durchgeführt und worauf muss ich besonders achten?
- Was ist als Bewerber/Bieter besonders zu berücksichtigen und welche Möglichkeiten der Rechtsdurchsetzung habe ich?

Die Art und Weise, wie Ausschreibungen gerade im öffentlichen Vergabewesen formuliert sind, kann zu Problemen führen. Krempl [04, p.219] bemerkt, dass in öffentlichen Ausschreibungen oft auf hunderten oder sogar tausenden Seiten zwar sehr viele Anforderungen gestellt werden, doch abgesehen von dem Ansatz, mit Hilfe der IT ein Reform- bzw. Innovationsprojekt umsetzen bzw. langfristig viel Geld sparen zu wollen, verbirgt sich dahinter oft wenig.

2.8 Anforderungen an Softwaregroßprojekte

Unterschiedlich dimensionierte Programme weisen unterschiedliche Anforderungen und Eigenschaften auf. An große Datenverarbeitungssysteme werden üblicherweise andere Anforderungen gestellt als beispielsweise an Applikationen für das iPhone.

Zuser et. al. [01, p.31] sprechen in diesem Zusammenhang von Domänen und bieten die folgende Übersicht an:

Domäne	Projekteigenschaften	besonders wichtige Produkte	Beispiele
Versicherungen/ Banken	Hohe Leistungsansprüche (Transaktionsraten)	Architektur, Entwurf	Transaktionsserver
Betriebssoftware	Sehr individuelle Wünsche des Kunden, teuer	Anforderungen, Anwenderschnittstelle	Zeiterfassung, Buchhaltung, Lagerverwaltung
Medizinische Anwendungen	Hohe Qualitätsansprüche	Qualitätsberichte, Leistungstests	Steuerungseinheiten für medizinische Geräte
Standardsoftware	Großes Leistungsspektrum	Anwenderdokumentation	Textverarbeitung, Grafikeditor

Abbildung 3: Anwendungsdomänen und deren Eigenschaften [ZuBi01, p.31]

2.9 Gesellschaftliche Aspekte von Softwaregroßprojekten

Frei nach dem Motto, dass alles, was technisch machbar ist, auch gemacht wird, wecken Computer alleine schon durch ihre technischen Möglichkeiten gewisse Begehrlichkeiten.

So weist Mölle [09, p.93] darauf hin, dass die schiere Möglichkeit der systematischen Erfassung von Daten bzw. deren rasante Verarbeitung oft dazu führt, dass die Begründung, warum eine bestimmte Software erstellt bzw. eingesetzt werden soll, darauf reduziert wird, dass es technisch möglich ist, bzw. dass bestimmte Daten zur Verfügung stehen.

Stehen gewisse Daten für Anwendung A erst einmal zur Verfügung, ist es nur eine Frage der Zeit bzw. der politischen Ausrichtung, diese Daten mit jenen für Anwendung B zu verknüpfen, um somit die Datenbasis für Anwendung C zu erhalten, von der ursprünglich (Anwendung A bzw. B) nie die Rede war. Ein Beispiel dafür sind Kommunikationsdaten aus der Mobiltelefonie, die mit den entsprechenden Standortdaten verknüpft werden.

In Zeiten der sogenannten terroristischen Bedrohung fällt es entsprechend motivierten Regierungen besonders leicht, auf unter Umständen bereits bestehendes Datenmaterial zwecks brisanter Auswertungen zurückzugreifen. Gerade in solch einem politischen Umfeld ist es besonders wichtig, dass es entsprechend starke Interessensgemeinschaften bzw. eine politische Opposition gibt, die in die unterschiedlichen politischen Entscheidungen in Form von Datenschutzgremien etc. eingebunden werden.

Während der österreichische Datenschutzrat in das Bundeskanzleramt eingegliedert ist, und daher als weder finanziell noch politisch unabhängig angesehen werden kann, ist die ARGE Daten ein gemeinnütziger Verein, der sich mit datenschutzrechtlichen Themen auseinandersetzt und zu aktuellen Themen Stellung bezieht.

Abschließend gibt Flyvbjerg [05, p.91] folgendes Dilemma bzgl. Großprojekten, die von staatlicher Seite durchgeführt werden, zu bedenken:

> *„The question has to be asked whether a government can act effectively as both promoter of a project, and the guardian of public interest issues such as protection of the environment, safety and of the taxpayer against unnecessary financial risks. The answer is negative.“ [ebd.]*

2.10 Das Scheitern großer Projekte

Krempl [04, p.218ff.] beschäftigt sich mit der Frage, warum so viele Großprojekte scheitern . Als Beispiele für de facto gescheiterte Projekte führt er die deutschen Projekte LKW-Maut, die Software für Hartz IV, die elektronische Gesundheitskarte, sowie den digital Polizeifunk an.

Die Komplexität der Vorhaben wird oftmals von den beteiligten Akteuren zunächst unterschätzt, und am Horizont auftauchende Probleme werden vorerst in ein späteres Stadium verdrängt. Die Projekte scheinen ausserdem aus allen Nähten zu Platzen, da es politischen Druck gibt, rasch Ergebnisse vorzuweisen, und eine perfekte Lösung zu präsentieren, die womöglich über den Stand der Technik hinausgeht [ebd.].

Das alles führt zu einer *„von Ehrgeiz getriebenen Überdimensionierung“* [ebd., p.219], gepaart mit durch Auftragnehmer illusorisch angesetzten Terminen bzw. blauäugig versprochenen Leistungen. Krempl merkt in diesem Zusammenhang an, dass fehlende Konventionalstrafen die Auftragnehmer geradezu dazu verleiten, unrealistische Planung einzureichen, um bei Ausschreibungen der Konkurrenz augenscheinlich einen Schritt voraus zu sein.

Obwohl IT-Architekturen durchaus mit vertrackten Bauvorhaben verglichen werden können, gibt es doch einen entscheidenden Unterschied: in der physischen Welt ist sofort erkennbar, ob beispielsweise ein Fundament schon fertig ist oder nicht; in der Welt der Software dagegen machen sich grobe Mängel erst spät – oft zu spät – bemerkbar. Ein Kurswechsel ist dann unter Umständen nicht mehr möglich, und das Projekt ist zum Scheitern verurteilt [ebd.].

Denker [05, p.25] führt Gründe an, warum „large-scale IT projects“ dazu neigen zu scheitern. Auszugsweise werden an dieser Stelle einige Erklärungen widergegeben:

- Auf Grund ihrer **kolossalen Komplexität** – Denker vergleicht Softwaregroßprojekte mit den Pyramiden der Ägypter – werden die Projekte verletzlich, frei nach dem Motto, dass die Kette nur so stark ist wie ihr schwächstes Glied.
- Denker schreibt Software eine gewisse Art von **Unsichtbarkeit** zu. Diese Eigenschaft kann zu einer Art Fehlwahrnehmung in dem Sinne führen, dass der IT zugetraut wird, alles nur Erdenkliche möglich zu machen – was in der Praxis dazu führt, dass der Aufwand der Umsetzung letztendlich stark unterschätzt wird.
- Die **Kluft zwischen Best Practice und Common Practice** macht sich laut Denker in der Art negativ bemerkbar, dass auf Grund der rasanten technologischen Weiterentwicklung kurzfristige Lösungen immer wieder der Etablierung von Best Practices entgegenwirken.

Um IT-Projekte vor dem Scheitern zu retten, schlägt Setzwein [09, p.76] vor, eine bestimmte Liste von Symptomen genauer im Auge zu behalten, um rechtzeitig reagieren zu können. Diese Symptome werden nun auszugsweise kurz beschrieben.

„Alle Arbeitspakete sind fast fertig – seit drei Wochen“ [ebd.]. Der Klassiker unter den Problemen beschreibt einen Mangel an Planung, vermutlich weil die Zeit und/oder der Aufwand falsch geschätzt wurden. In diesem Fall sind sofort realistische Fertigstellungstermine von den verantwortlichen Akteuren einzufordern.

„Mitarbeiter kennen Dokumente nicht oder wissen nicht, wo sie nach bestimmten Dokumenten suchen sollen“ [ebd.]. Es ist sicherzustellen, dass alle Beteiligten mit dem verwendeten Dokumentenmanagementsystem umgehen können. Im Zweifelsfall sind entsprechende Workshops anzubieten.

„Mitarbeiter [...] wagen es nicht, sich zu offen zu äussern“ [ebd.]. Die Projektmitarbeiter müssen die Eskalationswege kennen und rechtzeitig dazu ermuntert werden, diese auch zu nutzen. Wichtig ist das deshalb, weil die Mitarbeiter mit problematischen Situationen konfrontiert werden, von denen das Management erst erfährt, wenn der Mitarbeiter kommunikativ aktiv wird.

„Die Mitarbeitermotivation sinkt“ [ebd.]. Die möglichen Gründe wie bspw. Überlastung, schlechte Bezahlung, fehlende Entwicklungsmöglichkeiten, etc. sind umgehend durch den Vorgesetzen anzusprechen, und wenn möglich aus der Welt zu schaffen, da schlecht motivierte Mitarbeiter ein erhebliches Projektrisiko darstellen.

Abschließend ist der klassische Aufsatz von Johnson [94] zu erwähnen, der sich mit der Frage beschäftigt, warum so viele Software-Projekte scheitern. Johnson hat zu diesem Zweck umfangreiche Befragungen durchgeführt, die zu Erkenntnissen geführt haben, die nachfolgend vereinfacht vorgestellt werden.

Bei der Frage, was die Erfolgs-Faktoren für ein Projekt darstellen, wurde an erster Stelle die **Einbeziehung des Benutzers** genannt, gefolgt von der **Unterstützung durch das Top-Management**. An dritter Stelle wurden **klare Anforderungen** gereiht [vgl. ebd.].

Quasi als Gegenfrage wurde erhoben, welche Faktoren sich für ein Projekt als problematisch erweisen könnten. Hier wurden die **fehlende Einbindung der Benutzer** am öftesten genannt, an zweiter Stelle waren **unklare Anforderungen und Spezifikationen** zu finden. Als drittgrößtes Problem wurden sich **ändernde Anforderungen und Spezifikationen** genannt [vgl. ebd.].

Schließlich wurde die Frage gestellt, welche Faktoren ein Projekt letztendlich zum Scheitern bringen können. Hier schafften es die **lückenhaften Anforderungen** an die Spitze, gefolgt von **mangelhafter Benutzer-Einbindung** und **fehlenden Ressourcen** [vgl. ebd.].

Im Übrigen ist zu sagen, dass die Standish Group (www.standishgroup.com) laufend Statistiken veröffentlicht, die sich mit der Performance von Software-Projekten befassen.

2.11 Geschichtsträchtige Software-Großprojekte

Um ein Bild von großen Systemen zu vermitteln, werden zum Abschluss dieses Kapitels Software-Großprojekte vorgestellt, die es quasi in die Geschichtsbücher geschafft haben.

2.11.1 Semi-Automatic Ground Environment (SAGE)

Das Projekt „Semi-Automatic Ground Environment“ – kurz „SAGE“ – war ein militärisches Projekt der USA, und wurde zu dem Zweck ins Leben gerufen, den Luftraum der USA gegen feindliche Bomberangriffe zu verteidigen. Der Grundstein zu dem System wurde 1948 von Dr. George E. Valley gelegt, der die Idee hatte, sämtliche Radar-Informationen des US-amerikanischen Luftraums in Echtzeit den Operatoren zur Verfügung zu stellen, die dann zu entscheiden hatten, ob es sich um einen militärischen Angriff handeln könnte oder nicht. Nährboden dieser Idee war der beginnende kalte Krieg und die damit verbundene militärische Aufrüstung in Ost und West [vgl. URL01].

Im Laufe der 50er Jahre wurde das Projekt in Angriff genommen. Herzstück dabei war hardwareseitig eine „militär-gerechte“ Whirlwind-Version, softwareseitig mehr als 500.000 Zeilen Assembler-Code. Jeder Standort war über Telephonleitungen an mehrere Radarstationen angeschlossen, und komplett redundant ausgelegt. Die hohe Qualität der Komponenten sorgte trotz der relativ fehleranfälligen Vakuumröhren zu einer Verfügbarkeit von 99 Prozent [vgl. ebd.].

Operatoren konnten auf dem graphischen Terminal „Targets“ auswählen, und nähere Informationen darüber anfordern. Im Fall des Falles war es den Operatoren möglich, direkt einen Abfangbefehl an einen entsprechenden Luftstützpunkt abzusenden. Bei modernen Kampfflugzeugen war es möglich, den Autopiloten automatisch mit den Positionsdaten des „Targets“ zu füttern [vgl. ebd.].

1963 wurde das erste System in Betrieb genommen, 1983 wurde das letzte System abgeschaltet. Der Aufwand für dieses Projekt war enorm. Obwohl keine offiziellen Daten verfügbar sind, werden die Kosten auf acht bis zwölf Mrd. 1964-Dollars geschätzt. Es entbehrt nicht einer gewissen Komik, dass das System in Wirklichkeit komplett war, weil zu der Zeit, als es in Betrieb genommen wurde,

die russische Bomber-Gefahr durch die russische Raketen-Gefahr abgelöst wurde, gegen die das SAGE System komplett hilflos war [vgl. ebd.].

Nichtsdestotrotz brachte das Projekt bahnbrechende Erkenntnisse in den Bereichen online-Systeme, interaktives Computing, Echtzeitsysteme, bzw. die Kommunikation über weite Strecken per Modem [vgl. ebd.].

2.11.2 Semi-Automated Business Research Environment (SABRE)

Das Projekt „SABRE“ entstand aus der Anforderung, Buchungen für Flüge automationsunterstützt durchführen zu können [vgl. URL02].

Die Entstehung des Projekts war kurios: Die Chefs von American Airlines und IBM – C. R. Smith und Blair Smith – trafen sich zufällig auf einem gemeinsamen Flug, und kamen auf Grund ihrer Namensgleichheit ins Gespräch. American Airlines bekamen zusehends ein Kapazitätsproblem mit ihrer manuellen Flugreservierung. IBM hatten gerade mit SAGE ein gewaltiges online-System erfolgreich auf die Beine gestellt. Im Zuge des Gesprächs der beiden Manager kamen genau diese Themen zur Sprache, was dazu führte, dass tatsächlich 30 Tage später von IBM ein Forschungsprojekt vorgeschlagen wurde, dass die computergestützte Flugreservierung zum Ziel hatte [vgl. ebd.].

1957 wurde das Projekt gestartet, 1960 wurde das erste experimentelle System in Betrieb genommen – dazwischen lag ein Budget von unglaublichen 40 Millionen Dollar. Das System war in der Lage, täglich 83.0000 Telefonanrufe zu verarbeiten, und war damit ein voller Erfolg. 1964 wurden sämtliche Buchungen von American Airlines über das SABRE System abgewickelt [vgl. ebd.].

Ab 1976 wurde das System nicht nur von American Airlines direkt, sondern zusätzlich auch von Reisebüros genutzt. In den 1980ern wurden Reservierungen über das CompuServe Information Service angeboten, ab den 1990ern zusätzlich über America Online [vgl. ebd.].

Im März 2000 wurde SABRE von American Airlines ausgegliedert, und war als eigenständiges Unternehmen an der NYSE notiert. Zur Zeit wird das mittlerweile stark weiterentwickelte System von einer großen Anzahl an Akteuren genutzt, und zwar von (auszugsweise):

- 30.000 Reisebüros
- 400 Fluglinien
- 50 Autovermietungen
- 35.000 Hotels [vgl. ebd.]

Alles in allem kann das System als extrem erfolgreich eingestuft werden, da es möglich war, das System immer wieder an neue Gegebenheiten anzupassen, ohne den Anschluss an Technologie bzw. verändertes Kundenverhalten zu verpassen [vgl. ebd.].

2.11.3 Föderales integriertes standardisiertes computergestütztes Steuersystem (FISCUS)

Krempl [04, p.221] berichtet, dass es bei dem Projekt darum ging, einen Standard für die Datenverarbeitung in allen 650 deutschen Finanzämtern zu setzen. „Föderal“ bedeutete konkret, dass 17 (!) Auftraggeber an dem Projekt beteiligt waren.

> *„Nach 13 Jahren voller verzweifelter Versuche, das System zum Laufen zu bringen, beendete die Finanzministerkonferenz Mitte 2004 das Trauerspiel“ [ebd., p.221]*

Letztendlich wurden statt der veranschlagten 170 Millionen Euro cirka 900 Millionen Euro für unbrauchbare 1,6 Millionen Zeilen Code, 50.000 Seiten Dokumentation, etc. ausgegeben [vgl. ebd.].

2.11.4 Transfer and Automated Registration of Uncertificated Stock (TAURUS)

Das Ziel von TAURUS war, den Wertpapierhandel an der Londoner Börse papierlos zu machen. Es zeichnete sich im Vorfeld bereits ab, dass von dem Vorhaben etliche traditionelle Börsen-Jobs bedroht werden würden, was einige einflussreiche Parteien dazu veranlasste, gegen das Projekt zu opponieren [vgl. Bron00, p.38ff.].

Um möglichst viele Parteien „ins Boot zu holen“, wurde von der Bank of England ein Komitee gegründet, dass sämtliche Einwände berücksichtigen sollte um so

eine Kompromisslösung, die für alle Parteien annehmbar ist, zu entwickeln. Diese Entscheidung war der sogenannte „kiss of death“ für das Projekt – aus folgenden Gründen:

- Es fehlten deutliche, klar dokumentierte Ziele.
- Weder auf technischer, noch auf administrativer Ebene gab es klare Führungskompetenzen.
- Falsche Entscheidungen führten zu enorm hohen Kosten, die durch Customizing verursacht wurden.
- Die Kultur der Londoner Börse war nicht darauf ausgerichtet, sich einem zentralistischem technischem System anzupassen [vgl. ebd.].

Alle diese – und noch weitere – Gründe führten letztendlich dazu, dass das Projekt 1993 mit einem geschätzten Verlust von cirka 100 bis 400 Millionen Pfund offiziell zu Grabe getragen wurde. Schlimmer noch als der finanzielle Schaden war die Tatsache, dass der Ruf der Londoner Börse unter dem Debakel gelitten hatte [vgl. ebd., p.40].

2.12 Zusammenfassung

In diesem Kapitel ging es darum, ein Bild von Projekten im Allgemeinen, und von Software-Großprojekten im Besonderen zu zeichnen.

Es wurde darauf eingegangen, welche Umstände erfüllt sein müssen, damit ein Vorhaben vernünftigerweise als Projekt bezeichnet werden kann, und welche verschiedenen Typen von Projekten es gibt. Weiters wurde der Begriff „Software Engineering“ erklärt, und dessen Ursprung beleuchtet.

In der Vorstellung eines typischen Life-Cycles eines Datenverarbeitungs-Systems wurden die einzelnen Phasen besprochen, die ein Projekt „von der Wiege bis zur Bahre“ durchläuft.

In dem Abschnitt, der sich der Frage widmete was ein Projekt zu einem Großprojekt macht, wurden Merkmale erläutert, die darüber Auskunft geben können ab wann der Begriff „Software-Großprojekt“ gerechtfertigt ist.

Als nächstes wurde ein Licht auf die unterschiedlichen Entstehungsmöglichkeiten von Großprojekten geworfen, es wurde auf Ausschreibungen eingegangen, bzw. wurden Anforderungen beschrieben, die große Softwaresysteme erfüllen müssen.

Selbstverständlich durfte die Erwähnung gesellschaftlicher Aspekte bzgl. großer Datenverarbeitungssysteme nicht fehlen, und der Frage, warum gerade große Projekte dazu neigen zu Scheitern, wurde ebenfalls nachgegangen.

Zum Abschluss dieses Kapitels wurden geschichtsträchtige Projekte vorgestellt, die das Phänomen „Software-Großprojekt“ greifbar machen sollen.

Nachdem dieses Kapitel die Grundlagen zu Software-Großprojekten gelegt hat, wird im folgenden Kapitel das Thema Design von unterschiedlichen Perspektiven aus betrachtet.

3 Überlegungen zum Thema Design

„Design Is A Big Word“
– Alan Cooper

Design spielt gerade bei Software-Großprojekten eine entscheidende Rolle. Während bei konventionellen Projekten ein schlechtes Design unter Umständen noch irgendwie überarbeitet und ausgebessert werden kann, hat schlechtes Design bei Großprojekten meist fatale Folgen.

Warum das so ist, und was generell zum Thema Design in Bezug auf Großprojekte von Interesse ist, wird in diesem Kapitel behandelt.

Zunächst wird Design in seiner allgemeinen Form betrachtet, ohne vorerst auf die Besonderheiten in Verbindung mit Großprojekten einzugehen. Sobald ein klares Bild davon entstanden ist, worum es bei Design grundsätzlich geht, wird auf die speziellen Erfordernisse eingegangen, die sich aus der Tatsache ergeben, dass es sich bei dem in Frage kommenden Projekt um ein Softwareprojekt größeren Ausmaßes handelt.

3.1 Wofür steht Design?

Zu Beginn ein Zitat von Steve Jobs, das die Frage, worum es bei Design eigentlich geht, sehr prägnant trifft:

> *„Design is a funny word. Some people think design means how it looks. But, of course, if you dig deeper, it’s really how it works. To design something really well, you have to ‚get it’. You have to really grok [understand] what it’s all about.“ [Buxt07, p.309]*

Cooper [04, p.21f.] beschreibt den Begriff „Design“ als die Summe sämtlicher Entscheidungen, die im Zuge des Software-Erstellungsprozesses getroffen werden. Da es sich hierbei um einen „sea of design“ handelt, schlägt Cooper vor, eine Trennlinie zu ziehen, um Design in zwei Klassen einzuteilen. Die erste Klasse umfasst Design, welches direkt auf den Endbenutzer einwirkt – diese Klasse von Design bezeichnet Cooper als „interaction design“. Die zweite Klasse von Design umfasst alle anderen Aspekte der Software – diese Klasse bezeichnet Cooper als „program design“.

Cooper gibt jedoch zu bedenken, dass die erwähnte Trennlinie nicht immer komplett scharf gezogen werden kann. So würde beispielsweise die Wahl der Programmiersprache in die Klasse „program design“ fallen, da es prinzipiell für den Endbenutzer irrelevant ist, in welcher Sprache ein Programm geschrieben wurde. Andererseits hängt möglicherweise die Antwortzeit des Programmes von der Wahl der Sprache ab, in der das Programm geschrieben wurde – in diesem Fall würde die Wahl der Programmiersprache sehr wohl in die Klasse „interaction design“ fallen [vgl. ebd.].

Einen Konnex zwischen Design und Architektur stellen Brown et. al. [04, p.163] her:

> *„Die Software-Architektur ist ein Teil der gesamten Systemarchitektur, der alle Aspekte des Designs und der Implementierung einschließlich der Auswahl der Hardware und Technologie umfasst.“ [ebd.]*

Zu den wichtigsten Prinzipien zählen nach Meinung von Brown et. al., dass einerseits die Architektur einen Überblick über das gesamte System verschaffen soll – im Gegensatz zu Analyse- und Designmodellen, die sich lediglich mit Teil-Aspekten des Systems beschäftigen. Andererseits werden bei einem aussagekräftigen Modell des Gesamtsystems Perspektiven berücksichtigt, die jenen der verschiedenen Interessensgruppen bzw. Fachexperten entsprechen [vgl. ebd.].

3.2 Die Rolle des Benutzers

Gerade bei Großprojekten ist es wichtig, den Endbenutzer in den Mittelpunkt der Designarbeit zu stellen, da der Erfolg des Systems letztendlich dadurch bemessen wird, wie gut das fertige Produkt vom Benutzer angenommen wird.

> *„If you want an X, we can give it to you in ten months; if you turn out to want something other than an X, that’s your problem.“ [DeLi03, p.105]*

DeMarco et. al. bringen mit dieser Aussage zum Ausdruck, wie es NICHT sein darf, denn wenn das Ergebnis letztendlich nicht der Zielgruppe entspricht, ist es zu einem Problem aller beteiligten Parteien geworden.

Ein Programm, dass dem Benutzer den letzten Nerv raubt, wird – wenn es nicht unbedingt sein muss – an einem bestimmten Punkt schlicht und einfach nicht mehr verwendet werden.

Aus diesem Grund wurden Techniken entwickelt, welche die Designer dabei unterstützen sollen ergonomische Software zu entwickeln, um den Bedürfnissen der zukünftigen Anwender entgegen zu kommen.

Ein Ansatz ist jener von „Peronas“ und „Goals“. Die Idee zu Personas ist simpel:

> *„Develop a precise description of our user and what he wishes to accomplish.“ [Coop04, p.123]*

Es geht dabei nicht darum, sich einfach nur einen 08/15-Benutzer auszudenken und zu versuchen zu erraten, was die Wünsche dieses Benutzers sein könnten. Personas sind keine realen Menschen, aber sie begleiten als solche den kompletten Designprozess. Personas werden möglichst detailiert beschrieben, angefangen von deren Namen bis hin zu deren Hobbies, etc. [vgl. Coop04, p.123f.].

Goals wiederum sind nicht mit Tasks zu verwechseln:

> *„A goal is an end condition, whereas a task is an intermediate process needed to achieve the goal.“ [Coop04, p.150]*

Während es bei Tasks darum geht **wie** etwas zu geschehen hat, dreht sich bei Goals alles darum, **was** erreicht werden soll. Goals bleiben unabhängig von der Technologie relativ konstant, während sich Tasks entsprechend der technischen Möglichkeiten stark verändern können [vgl. Coop04, p.150f.].

Der Punkt ist, dass Programmierer dazu neigen Task-orientiert zu designen. Auf diese Weise kann zwar grundsätzlich das Ziel erreicht werden, jedoch entspricht die Art und Weise, wie das Ziel erreicht wird, nicht den Vorstellungen der späteren Benutzer.

Aus diesem Grund ist es erstrebenswert, Goal-orientiert zu designen – mit Hilfe von Personas. Wenn für die Ziele von Personas designed wird, können alternative Wege aufgezeigt werden, wie eine bestimmte Funktionalität erreicht werden kann. Üblicherweise entpuppen sich diese alternativen Wege bei weitem

benutzerfreundlicher als herkömmliche, durch task-orientiertes Design gefundene Lösungen [vgl. Coop04, p.152].

Löwgren streicht in diesem Zusammenhang die Bedeutung von Interaction Design hervor, das als Erweiterung von HCI (Human-Computer-Interaction) verstanden werden kann. Während es bei HCI hauptsächlich um Arbeits- bzw. Task-orientierte Situationen geht, die es unter Laborbedingungen zu optimieren gilt, geht Interaction Design einen Schritt weiter [vgl. URL04].

Interaction Design bezieht den Kontext, in der die Applikation genutzt wird, mit ein, und wird daher auch oft mit Experience Design in Verbindung gebracht. Gerade bei Computer-Systemen, die zunehmend in den 1990er Jahren an Bedeutung gewonnen haben – also beispielsweise mobile Anwendungen wie Mobiltelefone, Kameras, etc. – spielt Interaction Design eine weitaus größere Rolle, da nicht nur die Nutzung selbst, sondern auch das Umfeld, in dem die Nutzung stattfindet, Beachtung bzgl. des Designs finden muss [vgl. ebd.].

Das wiederum legt jedoch den Schluss nahe, dass gerade bei klassischen Software-Großsystemen wie beispielsweise Banken-Software, Interaction Design – verglichen mit traditionellem User Interface Design – weniger Beachtung finden wird.

3.3 Die Kunst des Requirements Engineering

> *„There's an old story about several blind men encountering an elephant for the first time. One grasps its leg and proclaims that the elephant is ‚very like a tree.' Another touches its side and states that it is ‚very like a wall.' Another grasps its trunk and declares it to be ‚very like a snake'." [Coop04, p.116]*

Diese Geschichte bringt sehr schön zum Ausdruck, was dabei herauskommt, wenn Zusammenhänge nicht vollständig erkannt werden. Wird nicht der Gesamtzusammenhang, sondern nur ein Ausschnitt vollständig verstanden, entsteht eine bestimmte Sicht auf einen Sachverhalt, die nicht den tatsächlichen Gegebenheiten entspricht.

Gerade in großen Softwareprojekten ist daher die Analyse der Anforderungen an das System nicht nur besonders wichtig, sondern auch besonders schwierig.

Die klassische Vorstellung, dass sich generalstabsmäßig die unterschiedlichen Phasen eines Projekts sequentiell aneinanderreihen (siehe Abbildung), ist mittlerweile längst überholt.

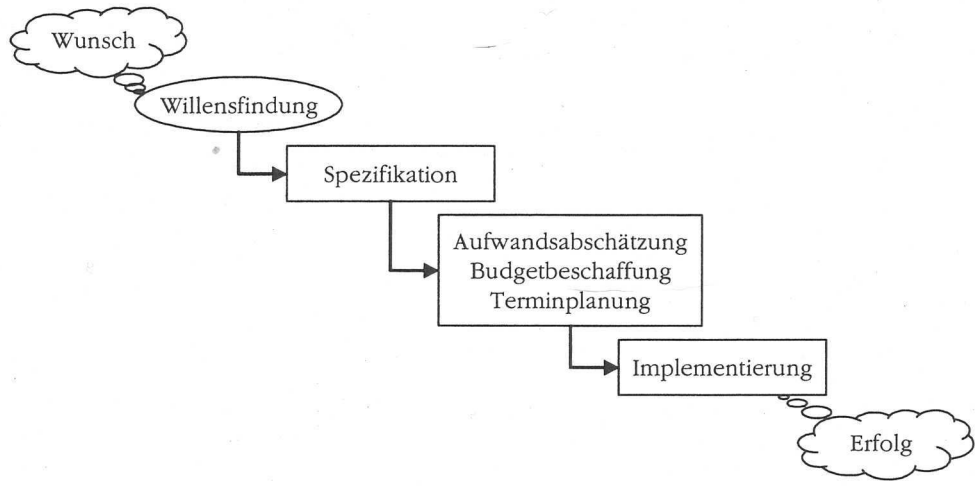


Abbildung 4: Prozessmodelle gehen davon aus, dass der Kunde weiß was er will. [Scha05, p.24]

Schabert [05, p.11] liefert auch gleich eine Reihe an Erklärungen, warum ein solcher Ansatz nicht funktionieren kann:

- Die Fachanwender sind der Meinung, dass ihre Probleme und Anforderungen von den Designern bzw. Architekten vollkommen verstanden wurden.
- Die Fachanwender erwarten, dass auch nicht kommunizierte Anforderungen Berücksichtigung finden.
- Die Marktsituation macht es erforderlich, dass rasch Ergebnisse geliefert werden.
- Die Kosten, die zu einem späteren Zeitpunkt durch schwammige Vorgaben entstehen, werden vom Management unterschätzt.

Da Anforderungen kritisch sind, weil sie als Ausgangspunkt für das Erzeugen von Ideen und möglichen Lösungen dienen, schlägt Berkun [09, p.104f.] einen Leitfaden vor, der für das erfolgreiche Beschreiben von Anforderungen herangezogen werden sollte:

- *„Erstellen Sie einen Plan zum Bestätigen und Anpassen von Anforderungen.*

- *Versuchen Sie, fehlerhafte Annahmen zu eliminieren.*
- *Finden Sie fehlende Informationen.*
- *Definieren Sie für jede Anforderung eine relative Priorität.*
- *Verbessern oder entfernen Sie ungewollt mehrdeutige Angaben.“ [ebd.]*

Wie wichtig eine saubere Spezifikation gerade in Software-Großprojekten ist, machen Bergsman et. al. mit folgender Graphik deutlich:

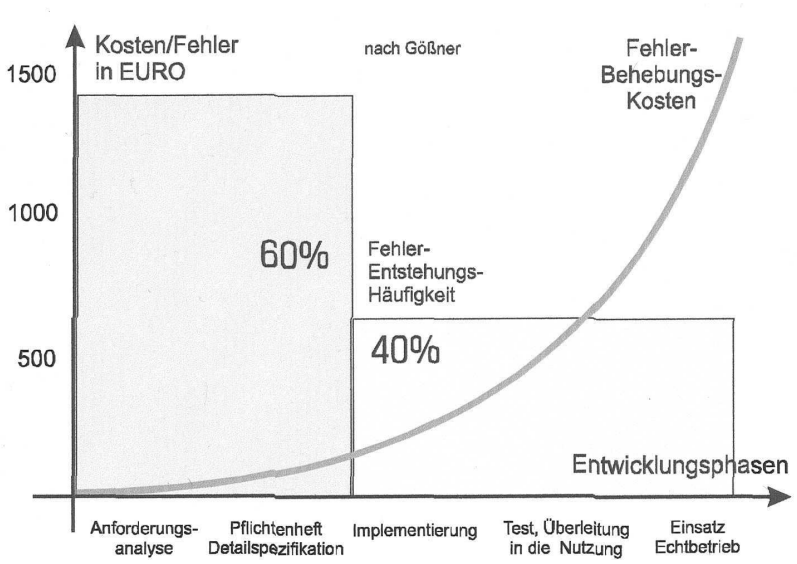


Abbildung 5: Fehlerkostenkurve und Fehlerentstehungshäufigkeit [BeAc04, p.7]

Statistiken belegen, dass die häufigsten Fehler ihre Ursache in der Spezifikationsphase haben. Das Problem dabei ist, dass der überwiegende Teil dieser Fehler erst in späteren Projektphasen entdeckt wird, und dass die Beseitigung dieser Fehler mit voranschreitender Zeit immer teurer wird [vgl. ebd.].

3.4 Requirement vs. Design

Mölle [09, p.93] weist darauf hin, dass es grundsätzlich eine idealisierte Vorstellung ist, dass Anforderungen lediglich vorgeben, WAS die Software leisten soll, ohne darauf einzugehen, WIE dies umzusetzen ist.

Dies macht sich anhand folgender Aussage bemerkbar:

> *„Angesichts dieser Trennung könnte man sogar zu der Annahme verleitet sein, Requirements wohnten noch keine Designentscheidungen inne. Dies ist jedoch aus mehreren Gründen falsch: Erstens können insbesondere nichtfunktionale Anforderungen bereits sehr enge Rahmenbedingungen für das Wie setzen, und zweitens kann schon das Was beliebig gravierende Verstöße gegen die Natur der jeweiligen Domäne enthalten.“ [ebd.]*

3.5 Einflüsse auf Design & Architektur

Schon 1968 formulierte Conway [68] die mittlerweile als „Conway’s Law“ bekannte Gesetzmäßigkeit:

> *„Organisationen, die Systeme entwerfen, unterliegen dem Zwang, Systeme zu bauen, die Kopien ihrer Kommunikationsstrukturen darstellen.“ [ebd.]*

Diese Aussage macht deutlich, wie wichtig funktionierende Kommunikationsstrukturen für den Projekterfolg sind – sie stellen einen direkten Quell für Design und Architektur dar.

Andere – teilweise unbekannte – Autoren haben Conway’s Law nach ihren eigenen Erfahrungen neu formuliert [URL03]:

- *„The architecture of a system is a copy of the architecture of the organization“ (Autor unbekannt)*
- *„The structure of a system is determined by the structure of the team“ (Autor unbekannt)*
- *„To understand the team, look at the software they’re producing. If it’s slow and bloated, the team is slow and bloated. If it’s lean and quick, the team is lean and quick“ (Jim McCarthy)*

Auf dieses Phänomen wird an späterer Stelle dieser Arbeit noch näher eingegangen.

3.6 Allgemeines zu Design in Softwaregroßprojekten

Eckstein [04, p.125] ist der Meinung, dass Architekturen nicht skalieren. Eine auf ein Projekt mittlerer Größe zurechtgelegte Architektur ist unter Umständen für ein großes System vollkommen ungeeignet. Genauso sind Erfahrungen, die mit kleinen Projekten gesammelt wurden, und die daraus resultierenden Empfehlungen mit Vorsicht zu genießen, da sie nicht direkt auf ein Großprojekt umgelegt werden können.

Eine äusserst komplexe und daher eher nur für Großprojekte geeignete Vorstellung von Design bietet Mayr [05, p.209ff.] an. Seiner Meinung nach legt Design mittels eines **Designmodells** fest, wie die Funktionalitätsanforderungen aus der Analysephase für die Implementierung aufbereitet werden. Dieses Designmodell umfasst einerseits das **Systemmodell**, welches die Systemarchitektur aus Entwicklersicht beschreibt, und andererseits das **Komponentenmodell**, welches die Komponenten beschreibt, die aus untereinander eng verwobenen und durch kompakte Schnittstellen gekapselte Klassen besteht.

Das Systemdesign wiederum umfasst die Modellierung von **Architektur**, **Schnittstellen**, und der **Datenhaltung**. Mayr spricht davon, dass die Entscheidungen, die in dieser Phase getroffen werden, später nur mit unverhältnismäßig hohem Aufwand geändert werden können. Die jeweiligen Modellierungen unterteilen sich wiederum in Sub-Modelle, auf die an dieser Stelle nicht weiter eingegangen wird. Erwähnt werden soll in diesem Zusammenhang lediglich die Modellierung der Schnittstellen, da Mayr an dieser Stelle das Benutzerschnittstellendesign sieht bzw. die Wichtigkeit dieser Arbeit hervorhebt [vgl. ebd., p.210].

Das Komponentendesign wiederum umfasst für jede Komponente die Modellierung der **Struktur** bzw. des **Verhaltens** einer Komponente. Auch auf diese beiden Modellierungen kann nicht näher eingegangen werden [vgl. ebd., p.213].

Insgesamt stellt sich die Frage, ob diese strikte Auftrennung der Designarbeit in der Praxis tatsächlich eine Rolle spielt.

Eckstein [04, p.116] vertritt die Meinung, dass bei großen Projekten die Architektur aus zwei Gründen möglichst einfach gehalten werden muss:

- Je einfacher die Architektur, desto größer das Verständnis aller in das Projekt involvierter Teammitglieder.
- Bei großen Projekten ist die Wahrscheinlichkeit hoch, dass sich die Anforderungen zu einem Zeitpunkt ändern, an dem sie sich nicht mehr ändern sollten – beispielsweise mitten in der Umsetzungsphase. Ein einfach gehaltenes Design lässt kurzfristige Kursänderungen eher zu als eine hochkomplexe Architektur, die komplett neu entworfen werden müsste.

3.7 Zur Bedeutung von Refactoring

Bei Refactoring handelt es sich um Veränderungen am Source-Code, ohne jedoch neue Features einzubauen. Es geht hier vielmehr um strukturelle Änderungen, die zur Verbesserung der architektonischen Qualität beitragen sollen. Refactorings sind umstritten, weil oftmals die Meinung vertreten wird, dass funktionierender Code möglichst nicht angegriffen werden soll, um keine unerwünschten Nebeneffekte zu provozieren.

Refactoring kann jedoch dazu beitragen überflüssige Teile über Board zu werfen, was dazu führt, dass ein System schlanker und somit weniger fehleranfällig wird. Eckstein [04, p.117] zitiert in diesem Zusammenhang Paul B. McCready, der zwar kein Beispiel aus der Software-Industrie, dafür aber aus der Luftfahrt anbieten kann :

> *„Als die Gossamer Condor zum ersten Mal in der Luft war, hörte die Entwicklung nicht einfach auf. Stattdessen wurde das Flugzeug immer einfacher und leichter gemacht, bis es nicht mehr fliegen konnte. Auf diese Weise fanden wir heraus, was unbedingt notwendig ist, damit ein Flugzeug in der Luft bleibt.“ [ebd.]*

Gerade bei großen Projekten kann es sich rächen, auf regelmäßiges Refactoring zu verzichten. Eckstein begründet dies damit, dass große Projekte oft eine sehr lange Laufzeit aufweisen – mitunter über Jahrzehnte. Das bedeutet, dass sämtlicher

Code mitgeschleppt und mitgewartet werden muss, obwohl unter Umständen die eine oder andere Komponente schon längst weggelassen hätte werden können [vgl. ebd.].

Auch die Einarbeitung neuer Teammitglieder in das System wird zu einem Problem, da sich die neuen Kollegen mit einem riesigen Moloch auseinandersetzen müssen, an dem jahrelang immer nur Code dazugepappt wurde ohne regelmäßig auszumisten [vgl. ebd.].

3.8 Das Problem des zweiten Systems

Brooks [95, p.55-58] erwähnt einen Effekt, den er als *„das Problem mit dem zweiten System“ [ebd.]* bezeichnet. Der Effekt hat folgende Gestalt:

Ein junger Systemarchitekt, der sein erstes großes System entwirft, wird zwar nach bestem Wissen und Gewissen Entscheidungen treffen, jedoch wird die Architektur auf Grund der geringen Erfahrung mit Schwachstellen behaftet sein. In der Retrospektive wird der Architekt all diese Mängel erkennen, daraus lernen, und für das nächste Projekt entsprechend gerüstet sein [vgl. ebd.].

Und hier beginnt die Problematik. Bei seinem zweiten System wird der Architekt versuchen, all die Punkte, die am ersten System nicht perfekt waren, jetzt perfekt zu machen. Das kann jedoch unter Umständen zu zwei verschiedenen Effekten führen. Das Projekt kann niemals zu einem Ende gebracht werden, weil in dem Vorhaben, alles perfekt umsetzen zu wollen, ein zu hoher Aufwand betrieben wird, der letztendlich das Projekt zum Scheitern bringt. Andererseits kann es passieren, dass das System zu einem „großen Haufen“ verkommt, das mit Funktionalität und Goodies vollgestopft ist, die kein Mensch braucht und die das Produkt nur unnötig belasten [vgl. ebd.].

Um diesem Dilemma zu entgehen, bietet Brooks zwei Lösungen an. Erstens rät er jungen Architekten, sich konsequent zurückzunehmen und sich auf die wesentlichen Dinge zu konzentrieren [vgl. ebd.].

Zusätzlich rät Brooks den verantwortlichen Projektleitern, bei großen, wichtigen Projekten ausschließlich auf erfahrene Systemarchitekten zu vertrauen, die bereits bei mehreren großen System mitgearbeitet haben [vgl. ebd.].

3.9 Modeerscheinungen

Zur Abrundung der Thematik ein paar Worte zum Thema Modeerscheinungen. Alle paar Jahre tauchen neue Methoden und Paradigmen auf, die nach ein paar Jahren gefeierten Einsatzes schon bald vom nächsten Hype abgelöst werden (sollen). Laut Mölle [09, p.95] ist der offen diskutierte Niedergang von SOA (Service-oriented Architecture) ein besonders lehrreiches Beispiel:

> *„Der Gedanke, Modularisierung so weit zu treiben, dass Komponenten unabhängig voneinander installiert und erst zur Laufzeit miteinander verknüpft werden können, ist nützlich, aber keine die Weltordnung auf den Kopf stellende Sensation (zumal sie der seit Jahrzehnten von den Unix-Utilities praktizierten Kultur nachkommt).“ [ebd.]*

Mölle bringt weiter zum Ausdruck, dass die Voraussetzung für gutes Design immer noch das grundsätzliche Verständnis und das Erkennen der zu lösenden Aufgabe ist. Die elegantesten Lösungen entstehen dann, wenn die zugrunde liegenden Strukturen der untersuchten Probleme präzise erfasst werden, und nicht, wenn ein saisonales Modethema als scheinbares Allheilmittel eingesetzt wird [vgl. ebd.].

Gerade beim Einsatz der speziell im Java-Umfeld beliebten Frameworks ist insofern Vorsicht geboten, da nicht immer garantiert ist, dass unterschiedliche Technologien, die zum Zeitpunkt x zueinander kompatibel waren, zum Zeitpunkt y immer noch reibungslos miteinander funktionieren. Langfristig kann das ein Problem darstellen, da im Extremfall eine Situation entstehen kann, dass das System niemals, oder nur sehr selten, auf neue Versionsstände der Laufzeitumgebungen etc. gehoben werden kann.

3.10 Designfehler

In seiner Fallstudie *„Apple, Design, and Business“ [Buxt07, p.41ff.]* beschreibt Buxton in acht Punkten seine Sichtweise auf die iMac Geschichte. In Punkt sechs – „The failures were keys to success“ – schreibt Buxton davon, dass auf lange Sicht betrachtet Sicherheit gefährlicher sein kann als Risiko.

Buxton meint damit, dass es oftmals notwendig ist, sich nicht nur auf der sicheren, bewährten Seite zu bewegen, sondern gelegentlich auch einmal das

bekannte Fahrwasser zu verlassen, um Neues auszuprobieren. Nur auf diese Weise bleibt der Lernprozess aufrecht, aber natürlich besteht auch die Gefahr des Scheiterns [vgl. ebd.].

Langfristig betrachtet ist es gewinnbringender, von Zeit zu Zeit Risiken einzugehen um innovativ zu bleiben. Für den Fall eines Fehlschlags kann immerhin der Lernfaktor als Teilerfolg verbucht werden – schließlich ist man zumindest um die Erfahrung reicher, wie Dinge nicht funktionieren [vgl. ebd.].

Bergsmann et. al. [04, p.3] merken jedoch an, dass die Situation im Bereich beispielsweise des e-Governments nicht so einfach betrachtet werden kann. Bei Systemen, die unter Umständen von zehntausenden oder gar millionen Nutzern bedient werden, verursacht ein Designfehler in der Ergonomie, der jeden Einzelnen nur 5 Minuten Zeit kostet, bei einer Million Anwendungen bereits einen volkswirtschaftlichen Schaden von über 10.000 Arbeitstagen.

3.11 Zur Bedeutung von Skizzen

Abschließend noch ein paar Worte zum Thema Skizzen, das sie ein unterschätztes Werkzeug für die Designarbeit darstellen. Oftmals werden Skizzen mit Prototypen verwechselt (und vice versa). Buxton [07, p.139f.] – ein Verfechter des exzessiven Skizzierens – veranschaulicht jedoch sehr eindrucksvoll, dass Skizzen und Prototypen zwei komplett unterschiedliche Dinge sind, die dementsprechend unterschiedliche Einsatzgebiete haben.

Prototypen sind schon recht konkrete Konstrukte, lassen kaum Interpretationsspielraum, und sind in der Herstellung kostenmäßig bereits auf einem Level, auf dem sie nicht ständig weggeschmissen und neu produziert werden können [vgl. ebd.].

Ganz anders bei Skizzen. Skizzen sind mitunter abstrakt, engen die Phantasie nicht ein sondern regen sie sogar an, und sind extrem schnell und günstig herzustellen [vg. ebd.].

Der Punkt ist, Skizzen und Prototypen richtig einzusetzen. Am Beginn des Design-Prozesses sollten möglichst viele Ideen zunächst in Form von Skizzen dargestellt werden, um vielen unterschiedlichen Gedanken „eine Chance zu

geben“. Wichtig dabei ist, sich auch mit auf den ersten Blick absurden Ideen Skizzen-technisch auseinanderzusetzen, da auf diese Weise wiederum neue Ideen entstehen können, die so sonst nicht erdacht worden wären [vgl. ebd.].

Erst in einem Stadium, in dem bereits vieles in eine bestimmte Richtung deutet, sollte mit der Erstellung von Prototypen begonnen werden.

Buxton illustriert diesen Sachverhalt sehr anschaulich mit der folgenden Abbildung:

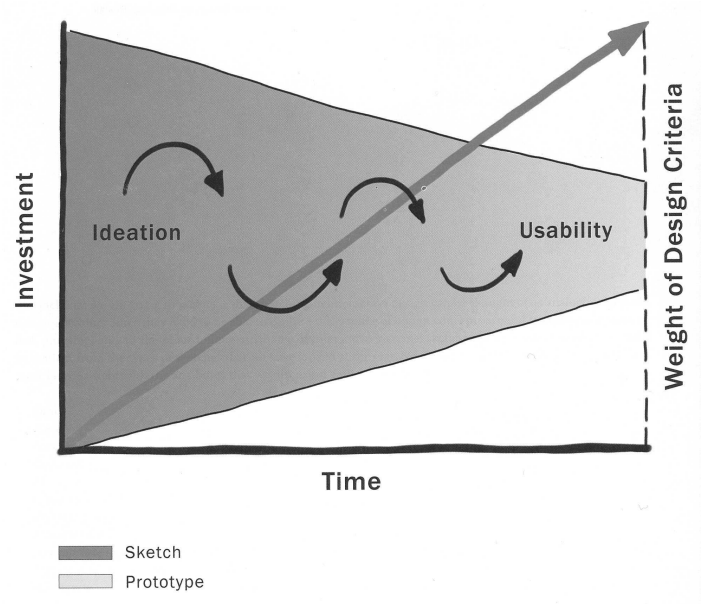


Abbildung 6: The Dynamics of the Design Funnel [Buxt07, p.138]

Zu Beginn der Designphase kann – dank Skizzen – mit relativ wenig Geld viel ausprobiert werden. Würde man statt Skizzen Prototypen anfertigen, wäre das Budget relativ rasch aufgebraucht. Je konkreter die Designvorstellungen werden, desto eher rentiert es sich, Prototypen anzufertigen.

Nachdem in diesem Abschnitt solide Erkenntnisse gewonnen wurden, was generell unter Design zu verstehen ist, wird im nächsten Abschnitt auf die Frage eingegangen, in wieweit sich diese Erkenntnisse auf Projekte größeren Ausmaßes übertragen lassen.

3.12 Zusammenfassung

Inhalt dieses Kapitels war es festzustellen, was erstens Design grundsätzlich bedeutet, und zweitens, welchen Stellenwert gutes Design gerade in Software-Großprojekten hat.

Zunächst wurden verschiedene Definitionen vorgestellt, die den Begriff „Design“ einerseits greifbar machen, andererseits zur Veranschaulichung dienen, aus welch unterschiedlichen Gesichtspunkten Design betrachtet werden kann.

Die Rolle des Benutzers, der ja tagtäglich mit dem Design des Produktes konfrontiert wird, wurde in weiterer Folge ausführlich behandelt.

Da Design und Anforderungen an das System grundsätzlich in Verbindung stehen, wurden zunächst Basics zum Requirements Engineering behandelt, und anschließend darauf eingegangen, wo die Unterschiede zwischen Requirements und Design liegen.

Der Einfluss der Organisationsform der Projektorganisation auf das Design wurde in einem eigenen Abschnitt behandelt, ebenso wie Allgemeines zu Designfragen in Softwaregroßprojekten.

Es wurden Tätigkeiten wie Refactoring analysiert, bzw. wurde das sogenannte Problem des zweiten Systems ebenso angesprochen wie Modeerscheinungen in Designfragen.

Abschließend wurde auch auf Designfehler eingegangen, und es wurde die Bedeutung von Skizzen als unterschätztes Design-Instrument vorgestellt.

Nachdem nun die Grundlagen für Projekt bzw. Design gelegt wurden, behandelt das nächste Kapitel das Herzstück dieser Arbeit, nämlich die Entscheidungsprozesse in Softwaregroßprojekten.

4 Wie funktionieren Entscheidungsprozesse

„Entscheiden Sie sich, entschieden zu entscheiden."

– Manfred Noe

Das Treffen von Entscheidungen ist ein zentrales Thema bei der Abwicklung von Projekten. Entscheidungen stehen laufend an, und daher ist die Art und Weise, wie der Entscheidungsprozess gelebt wird, für Erfolg bzw. Misserfolg eines Projekts von größter Relevanz.

Es ist daher kein Zufall, dass Rattay gerade die Komponente *„Entscheidungen treffen und einfordern"* im Mittelpunkt der neun Hauptaktivitäten einer Führungskraft steht:

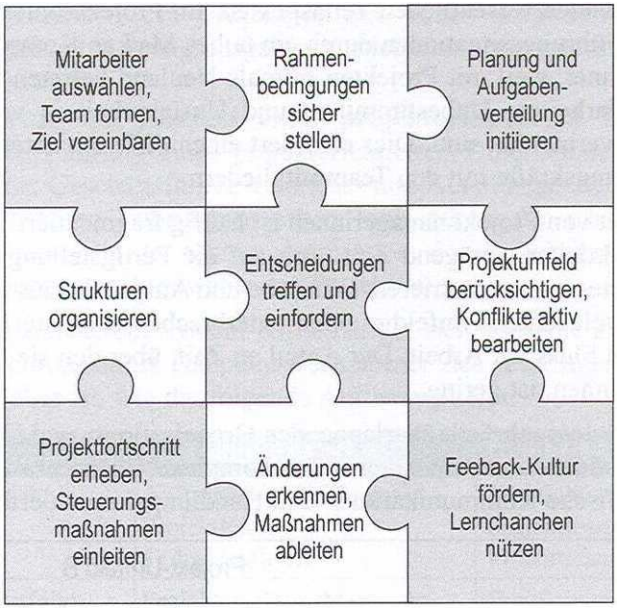


Abbildung 7: Die neun Hauptaktivitäten einer Führungskraft – im Projekt [Ratt07, p.15]

In diesem Kapitel wird daher ausgehend von unterschiedlichen Perspektiven analysiert, wie mit Entscheidungsprozessen in Projektorganisationen umgegangen wird.

4.1 Menschen & Teams

Ohne Menschen kann kein Projekt abgewickelt werden – so viel steht fest. DeMarco et. al. [91, p.5] zollen diesem Umstand große Bedeutung bei:

> *„Die größten Probleme bei unserer Arbeit sind keine technologischen Probleme, sondern soziologische Probleme.“ [ebd.]*

In den folgenden Unterkapiteln wird daher das Thema Menschen & Teams genauer untersucht.

4.1.1 Grundsätzliches

Je mehr Personen an einem Projekt beteiligt sind, desto eher wird schon alleine die Teamgröße zu einem Projektrisiko, wie Eckstein [04, p.38] bemerkt und weiter ausführt, dass die Qualität von Entscheidungen typischerweise in Relation zur Teamgröße steht: Bei zunehmender Teamgröße ist festzustellen, dass Entscheidungen zunehmend unklarer werden, da Teammitglieder bei steigender Anzahl der Kollegenschaft immer weniger bereit sind, Verantwortung zu übernehmen. Eckstein liefert als Erklärung für dieses Paradoxon, dass bei einer großen Anzahl an Mitarbeitern viele Personen davon ausgehen, dass sich wahrscheinlich andere Teammitglieder dafür finden werden, Verantworung zu übernehmen und Entscheidungen zu treffen.

Halbherzige oder gar nicht getroffene Entscheidungen führen jedoch im Regelfall dazu, dass sich die Dynamik des Teams entweder in Richtung Stillstand entwickelt, oder aber, dass zu viele individuelle Entscheidungen getroffen werden, die besser abgestimmt werden müssten um keine Probleme zu verursachen [vgl. ebd.].

Eckstein bringt menschliches Verhalten auf den Punkt:

> *„Es mag im ersten Moment ungewöhnlich erscheinen, aber es ist immer besser, eine klare, aber möglicherweise falsche Entscheidung zu treffen und diese ggf. später zu korrigieren. Eine falsche Entscheidung gibt die Möglichkeit, aus Fehlern zu lernen; keine Entscheidung ermöglicht dies nicht. Wenn die Entscheidung (ewig) hinausgezögert wird, lässt sich nie feststellen, was die richtige Lösung gewesen wäre. Wenn man zunächst eine falsche Entscheidung trifft, lernt man aus den Konsequenzen dieser Entscheidung und kann diese Entscheidung aufgrund der gewonnenen Erkenntnisse korrigieren.“ [ebd.]*

4.1.2 Akteure im Entscheidungsprozess

Ausgehend von der Größe des Projekts bzw. von der Tragweite einer Entscheidung, sind unterschiedliche Akteure an der Entscheidungsfindung beteiligt. An dieser Stelle seien der **Projektleiter**, der **Projektauftraggeber**, diverse **Führungsgremien**, aber auch **Arbeitsgemeinschaften**, **Konsortien**, oder **Benutzergruppen** erwähnt.

Spannend in diesem Zusammenhang ist die Frage, wie sich die involvierten Akteure in dem Spannungsfeld „Entscheidungskompetenz vs. Verantwortung vs. Zuständigkeit“ bewegen.

Die grundsätzlich höchste Instanz in einem Projekt, die mit der Kompetenz ausgestattet ist, Befugnisse zu erteilen bzw. zu entziehen, ist der Projektlenkungsausschuss (kurz PLA). Er kann als Verbindungs- bzw. Schlichtungsgremium verstanden werden, das sich aus Mitgliedern der Geschäftsleitung, dem Projektmanager, dem Projektleiter und dem Auftraggeber zusammensetzt [vergl. Kerb00, p.44].

Der PLA ernennt den Projektleiter und stattet ihn wiederum mit entsprechenden Kompetenzen aus. Dem Projektleiter selbst kommt eine bedeutende Funktion als Integrationsfigur zu, da er die technischen, wirtschaftlichen, und selbstverständlich politischen Blickwinkel des Projekts nicht aus den Augen verlieren darf, und all diese Bereich konsolidieren muss. Diese Aufgabe wird umso schwieriger, je größer das Projekt ist. Bei sehr großen Projekten wird es daher beispielsweise einen eigenen Projektleiter für technische bzw. wirtschaftliche Belange geben, usw.

Zusätzliche Komplexität erfährt das Projekt dadurch, dass häufig nicht ein (1) Unternehmen mit der Umsetzung beauftragt wird, sondern gleich mehrere.

Die Leitung solcher Arbeitsgemeinschaften bzw. Konsortien kann sich aus folgenden Gründen als äusserst schwierig erweisen:

- In den beteiligten Unternehmen herrscht eine vollkommen andere Kultur vor.
- Der Projektleiter wird von den Angestellten eines der beteiligten Unternehmen nicht als Führungsperson akzeptiert.

Aus diesen Gründen ist es sehr wichtig, dass der oder die Projektleiter vom PLA mit entsprechend starken Kompetenzen ausgestattet werden, bzw. dass sich alle beteiligte Entscheidungsträger auf einen Projektleiter einigen, der von sämtlichen Akteuren akzeptiert wird.

4.1.3 Teambildung im Großen

Je größer ein Projekt, desto größer das Team – so weit, so gut. Je größer das Team, desto schwieriger wird es zu managen. Das ist der Grund, warum große Teams in kleinere Sub-Teams unterteilt werden, um eine funktionierende Leitung zu ermöglichen. Eckstein [04, p.46ff.] schlägt als Faustregel vor, eine Sub-Teamgröße von zehn Personen nicht zu überschreiten, um uneingeschränkt handlungsfähig zu bleiben.

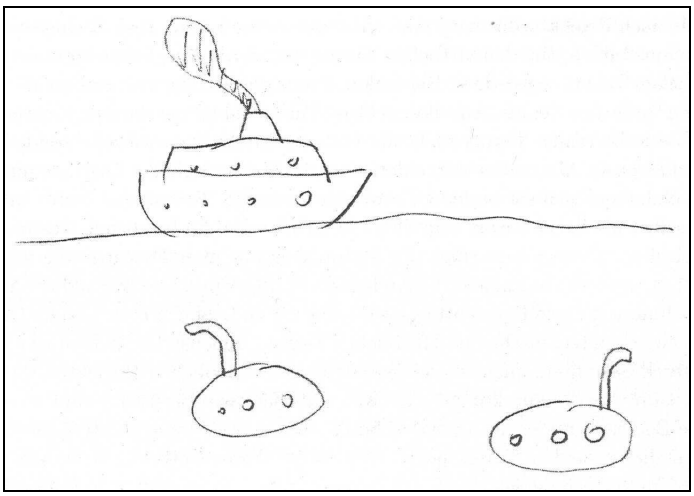


Abbildung 8: Subteams ... [Ecks04, p.47]

Eckstein führt weiters aus, dass die Konsolidierung großer Teams oftmals Taylors Theorie folgend anhand der Fähigkeiten einzelner Personen erfolgt. Das führt dazu, dass jeweils ein Team mit Analytikern, Designer, Testern, Entwicklern, etc. gebildet wird. Dies wird als **horizontale Teamaufteilung** bezeichnet [vgl. ebd.].

Obwohl sich der Taylorismus bei immer wiederkehrenden, gleichbleibenden Tätigkeiten bewährt hat (Stichwort „Fließbandproduktion“), werden Kreativität und ganzheitliches Denken durch eine tayloristische Teamkonstellation eher behindert. Aus diesem Grund sollten **vertikale Teams** geschaffen werden, wobei sich jedes Team um eine falchliche Komponente kümmert. Obwohl diese Form der Teams inspirierend auf die Mitglieder wirkt, weil – technisch gesehen – unterschiedliche Schwerpunkte aufeinandertreffen, kann trotz allem das Problem auftreten, dass nicht jedes Team mit gleich kompetenten Mitgliedern ausgestattet ist [vgl. ebd.].

Daher sollte vorerst ein kleines Team mit einer Kombination aus guten Fachtechnikern und Architekten auszustatten, und dieses Team zunächst die primäre Architektur des Systems ausarbeiten zu lassen. In weiterer Folge dient dieses Team sowie das grundsätzliche Design dann als Vorbild für die Bildung bzw. die Arbeitsweise der weiteren Teams [vgl. Ecks04, p.46].

4.1.4 Kleine feine Teams in Großprojekten?

Der Vorteil kleiner, leistungsfähiger Teams ist, dass sie mit erstklassigen Personen besetzt wahre Wunder vollbringen können. Das Problem an ihnen ist, dass sie zu klein sind, um wirklich große Projekte abzuwickeln.

Mills [71] schlägt daher vor, jeweils ein Segment einer Aufgabe einem Team zuzuordnen, das ähnlich einem Ärzteteam organisiert ist. Der „Chefarzt“ operiert, während die anderen Kollegen wichtige Zuarbeiten leisten.

Brooks [95, p.32ff.] beschreibt, wie ein solches Team aussehen könnte:

- Der **Chefprogrammierer** („Surgeon“) legt Spezifikationen fest, trifft sämtliche Entscheidungen, und programmiert selbst. Er besitzt beträchtliches Fachwissen und kann auf mindestens zehn Jahre Erfahrung zurückblicken.

- Der **Kopilot** („Copilot“) ist die rechte Hand des Chefprogrammierers, hat jedoch weniger Erfahrung und trifft selbst keine Entscheidungen. Auf Grund seiner Kompetenz ist er jedoch Ratgeber bzw. Backup für den Chefprogrammierer.
- Der **Verwalter** („Administrator“) ist der „Vollstrecker“ des Chefprogrammierers in Infrastruktur- und Personalfragen. Der Boss entscheidet, und der Verwalter sorgt für die wunschgemäße Umsetzung.
- Der **Redakteur** („Editor“) ist dafür zuständig, die Rohfassung der Dokumentation, die der Chefprogrammierer selbst erstellt, in eine vernünftige Form zu bringen. Er betreut somit die Dokumentation vom ersten bis zum letzten Tag.
- Dem Verwalter sowie dem Redakteur wird **jeweils ein Sekretär** („Secretary“) zur Seite gestellt. Die Sekretäre haben die Aufgabe, ihren Chefs sämtliche administrative Tätigkeiten wie Korrespondenz etc. abzunehmen.
- Der **Gehilfe** („Programming clerk“) ist für sämtliche technische Aufzeichnungen des Teams verantwortlich, und ist in verschiedensten Büroarbeiten ausgebildet. Er nimmt den Programmierern viel lästigen Papierkram ab.
- Der **Werkzeugmeister** („Toolsmith“) unterstützt das Team, in dem er Tools zur Vereinfachung der täglichen Arbeit erstellt. Das Schreiben von Funktions- und Makrobibliotheken zählt ebenfalls zu seinen Tätigkeiten.
- Der **Prüfer** („Tester“) ist dafür zuständig, Prüfroutinen zu erstellen die dafür verwendet werden, sämtlichen Code zu testen der im Team erstellt wird. Skuzessive entsteht auf diese Weise eine komplette Test-Suite für das Produkt.
- Das **Sprachgenie** („Language lawyer“) kennt jedes Detail der verwendeten Programmiersprache, und kann auf diese Weise die Programmierer dabei unterstüzten, Algorithmen elegant und effizient in die Sprache umzusetzen.

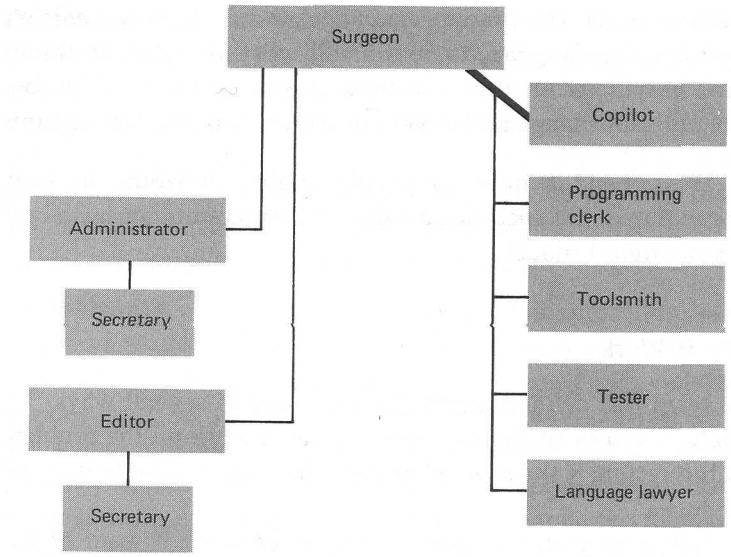


Abbildung 9: Communication patterns in 10-man programming teams [Broo95, p.36]

Die Challenge in einem Großprojekt ist nun, viele kleine „Ärzteteams“ zu etablieren, die auf hohem Niveau möglichst unabhängig voneinander arbeiten können [vgl. ebd.].

4.2 Grundsätzliches zur Kommunikation

Findet Kommunikation nur in sehr eingeschränkter Form, oder schlicht in schlechter Qualität statt, sind Projekte massiv gefährdet.

Ohne Kommunikation ist keinerlei Zusammenarbeit möglich, wie Eckstein [04, p.27] prägnant zusammenfasst:

> *„Kommunikation umfasst die Interaktion zwischen Teammitgliedern, verschiedenen Teams, zwischen den Teams und dem Management, zwischen Teams und den Kunden und so weiter.“ [ebd.]*

Weiters führt sie aus, dass keine Kommunikation so gute Ergebnisse erzielt wie die direkte Kommunikation von Angesicht zu Angesicht. Dieser Effekt wird durch die räumliche Beschaffenheit des Teams verstärkt – je weniger Raum sich zwischen den Mitgliedern befindet, desto besser wird die Kommunikation funktionieren [vgl. ebd.].

4.2.1 Mittel der Kommunikation

Cockburn [02] reiht die unterschiedlichen Kommunikationskanäle aufsteigend sortiert nach ihrem Nutzen wie folgt:

- Papier
- Tonaufzeichnung
- Zwei Personen via e-Mail
- Videoband
- Zwei Personen am Telefon
- Zwei Personen am Whiteboard

Eckstein [04, p.27] hält dazu fest, dass interessanterweise viele Projekte immer noch an der nach dieser Liste schlechtesten Kommunikationsform festhalten – dem Papier.

4.2.2 Teamgröße und Kommunikation

Es ist wahrscheinlich, dass mit zunehmender Größe des Projekts auch die Anzahl der am Projekt beteiligten Personen ansteigt. Unterschiedliche Teamgrößen bringen verschiedenartige Aspekte der Kommunikation mit sich, wie Eckstein [04, p.5] ausführt.

Wird ein Projekt von **einer Person** alleine durchgeführt, besteht lediglich Kommunikations- bzw. Abstimmungsbedarf mit dem Kunden. Es gibt keine anderen Teammitglieder, mit denen kommuniziert werden muss [vgl. ebd.].

Ab zwei Personen müssen die Teammitglieder bereits aus offensichtlichen Gründen damit beginnen, miteinander zu kommunizieren. Bis zu einer überschaubaren Teamgröße von cirka 10 Personen stellt die Kommunikation untereinander jedoch noch kein Problem dar – es ist prinzipiell möglich, dass bei sämtlichen Besprechungen immer alle Mitglieder anwesend sind, sodass einerseits jedes Teammitglied auf dem gleichen Wissensstand ist, und andererseits Abstimmungen untereinander rasch und unbürokratisch getroffen werden können [vgl. ebd.].

Ab zehn Personen ist es bereits notwendig, neben der technischen Abstimmung auch die Kommunikation selbst zu koordinieren. Kommunikationskanäle müssen etabliert werden, da nicht mehr jeder mit jedem informell Informationen austauschen kann [vgl. ebd.].

Ab 100 Personen ist oftmals bereits eine räumliche Trennung der Teammitglieder gegeben, sodass eine natürliche Kommunikation – also ohne technische Hilfsmittel – nicht mehr möglich ist. Ein funktionierender Informationsfluss ist nur noch möglich, indem Regeln aufgestellt werden an die sich alle Beteiligten halten müssen, um keine Missverständnisse etc. aufkommen zu lassen [vgl. ebd.].

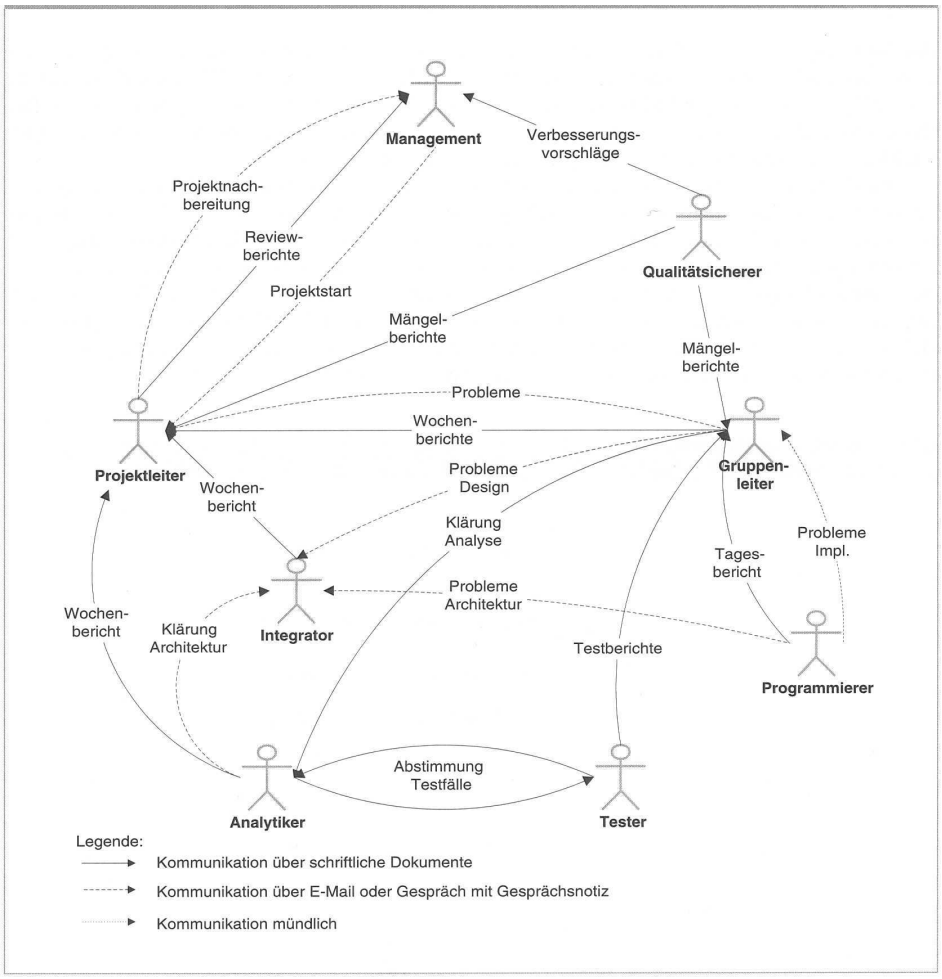


Abbildung 10: Kommunikationsbeziehungsdiagramm [ZuBi01, p.43]

Ab 1.000 Personen kommt zu der räumlichen Trennung oft auch eine „zeitliche“ Trennung hinzu. Das bedeutet, dass Teammitglieder unter Umständen auf unterschiedlichen Kontinenten in unterschiedlichen Zeitzonen zusammenarbeiten müssen. Das führt einerseits dazu, dass sich die Beteiligten gar nicht mehr

persönlich kennen, andererseits, dass bei dem Nachrichtenaustausch auf asynchrone Kommunikationsformen zurückgegriffen werden muss.

Zuser et. al. [01, p.43] schlagen daher vor, ab einer gewissen Teamgröße die Kommunikation untereinander nicht dem Zufall zu überlassen, sondern ein Kommunikationsbeziehungsdiagramm zu erstellen. Dieses Diagramm legt fest, wer mit wem in welcher Form über welches Thema zu kommunizieren hat, und kann wie in Abbildung 10 dargestellt aussehen.

4.2.3 Nachvollziehbarkeit von Kommunikation

Ein kritischer Punkt bei der Kommunikation innerhalb des Teams ist nach Meinung von Zuser et. al. [01, p.42] die Nachvollziehbarkeit von Absprachen zwischen Entwicklern und Kundenvertretern.

Dabei geht es darum, dass wichtige Informationen unter Umständen verloren gehen können, weil sie nach einem Gespräch (Telefonat, Besprechung, Arbeitsessen, etc.) nicht bleibend archiviert wurden. Das führt dann beispielsweise dazu, dass dieselben Themen mit dem Kunden öfters geklärt werden müssen, was mitunter zu einem Vertrauensverlust durch den Kunden führen kann, da der Kunde das Gefühl bekommt, dass ihm nicht zugehört wird [vgl. ebd.].

Werden im Zuge der Kommunikation auch Entscheidungen getroffen, ist es sogar von noch größer Bedeutung, die getroffenen Entscheidungen in irgendeiner Form festzuhalten. Wird auf die Dokumentaion von Entscheidungen verzichtet, führt dies mitunter zu unterschiedlichen Vorgehen in der Arbeitsgruppe, was letztendlich wiederum dazu führt, dass Ausbesserungsarbeiten geleistet werden müssen, die einen ungeplanten Zusatzaufwand für das Projekt bedeuten [vgl. ebd.].

4.2.4 Von der Wichtigkeit des „common vocabulary“

Cooper [04, p.185f.] merkt an, dass es für eine funktionierende Kommunikation extrem wichtig ist, dass unter den kommunizierenden Personen Einigkeit über das verwendete Vokabular und dessen Bedeutung herrscht.

Aus diesem Grund sollte von Beginn an geklärt werden, was unter bestimmten Schlagworten zu verstehen ist. Begriffe wie beispielsweise „Komponente“, „Modul“, „Paket“, etc. eigenen sich hervorragend für Missverständnisse, weil jeder Gesprächsteilnehmer etwas anderes darunter versteht.

Es gibt aus der Sicht des Projektleiters mehrere Möglichkeiten, um für ein „common vocabulary“ zu sorgen. Beispielsweise könnte es ein projektweites Glossar geben, in dem die gängigsten im Projekt vorkommenden Begriffe beschrieben werden. Jeder neue Mitarbeiter ist dazu angehalten, das Glossar durchzulesen und die Definitionen zu akzeptieren, unabhängig von der eigenen Vorstellung.

Im Zuge einer Besprechung bietet sich folgende Möglichkeit an: Bemerkt der Moderator, dass in einer Sache keine Einigung erzielt werden kann, und dabei immer wieder die gleichen Begriffe genannt werden, ist es angebracht, die Kontrahenten dazu aufzufordern, eine Definition der betroffenen Begriffe abzugeben. Oft stellt sich im Zuge dessen heraus, dass es sich um ein Missverständis gehandelt hat, und dass nach Klärung der Begrifflichkeiten einer Einigung nichts mehr im Wege steht.

4.2.5 Zu viel an Kommunikation?

Lientz et. al. [01, p.264f.] weisen darauf hin, dass es unter Umständen auch ein zu viel an Kommunikation geben kann. Das macht sich beispielsweise so bemerkbar, dass Teammitglieder beginnen, bei Besprechungen über persönliche Dinge zu plaudern. Dies ist so lange kein Problem, solange ein gewisses Maß nicht überschritten wird. Wird es überschritten, werden sich weniger soziale Teammitglieder ausgeschlossen fühlen. Davon abgesehen kann es passieren, dass schlicht und einfach zu viel Zeit für Plauderein und zu wenig Zeit für die Arbeit aufgewendet wird.

Einer solchen Entwicklung kann entgegengesteuert werden, indem das Gespräch vom Projektleiter konsequent auf geschäftliche Belange zurückgelenkt wird, ohne dabei jedoch den Personen das Gefühl zu geben, kein Interesse für ihre persönlichen Belange zu hegen [vgl. ebd.].

4.2.6 Interaktions- und Kommunikationsstrukturen

Im Zuge der Überlegungen, die zur Etablierung von funktionierenden Interaktions- und Kommunikationsstrukturen bei großen Projektteams angestellt werden müssen, gilt es eine Reihe von Punkten zu beachten die im Folgenden nach Eckstein [04, p.54ff.] beschrieben werden.

Die **direkte Kommunikation** – also beispielsweise das persönliche Gespräch – ist jene Form der Kommunikation, die den ungetrübtesten Informationsaustausch ermöglicht. Der erste Grund dafür ist, dass mit Hilfe von Mimiken und Gesten das gesprochene Wort zusätzlich betont werden kann. Der zweite Grund ist, dass per sofortigem Nachfragen Unklarheiten momentan beseitigt werden können [vgl. ebd.].

Das soeben Festgehaltene trifft uneingeschränkt auf die Kommunikation zwischen zwei Personen zu. Schwieriger kann die Situation bei Gruppengesprächen werden, da erstens aus Zeitgründen unter Umständen nicht alle Teilnehmer zu Wort kommen können, und zweitens eher introvertierte Kollegen selten bis gar nicht das Wort ergreifen werden [vgl. ebd.].

Ein oft vernachlässigter Aspekt bei der Überlegung zu Kommunikationsstrukturen ist die Tatsache, dass jede Person **unterschiedliche Kommunikationskanäle** verschieden gut nutzen kann. So können visuell geprägte Menschen Informationen besonders gut durch Beobachtung aufnehmen, auditiv orientierte Menschen durch Zuhören, und Kinesthetiker wiederum resorbieren Informationen besonder durch durch direkte Aktion [vgl. ebd.].

Beachtenswert ist in diesem Zusammenhang die **Überdosierung von Kommunikationsmedien**. Einfach ausgedrückt geht es hierbei darum, dass ein ursprünglich gut funktionierender Kommunikationskanal durch exzessiven oder falschen Gebrauch am Ende nicht mehr funktioniert.

Eckstein führt in diesem Zusammenhang das e-Mail als Paradebeispiel an [vgl. ebd.]:

> *„Wenn man zum Beispiel Informationen per E-Mail austauscht, wird man E-Mails so lange lesen und beantworten, bis die Anzahl von E-Mails, die man täglich bekommt, nicht mehr zu bewältigen ist. Dies führt dann dazu, dass man E-Mails ignoriert oder nur noch selektiv liest. In beiden Fällen besteht die Gefahr, dass der Sender einer E-Mail davon ausgeht, dass diese E-Mail gelesen wurde, obwohl dies gar nicht der Fall ist, und falsche Schlüsse aus der Tatsache zieht, dass diese E-Mail nicht beantwortet oder nicht entsprechend auf sie reagiert wurde.“ [ebd., p.55]*

Daraus folgt zusammenfassend, dass die Etablierung funktionierener Interaktions- und Kommunikationsstrukturen keinen starren Regeln folgen kann, sondern dass auf die an dem Projekt beteiligten Personen Rücksicht genommen werden muss, und dass es schlicht keine Patentrezepte gibt. Ein ausgewogener Mix ist die vielversprechendste Möglichkeit eine breite Basis der Teammitglieder anzusprechen, um auf diese Weise eine gesunde Kommunikationsbasis für das Projekt zu legen.

4.3 Der Projektleiter

Der Projektleiter spielt eine entscheidende Rolle in der Projektabwicklung. Aus diesem Grund wird in den folgenden Abschnitten ein genaueres Bild dieser Funktion gezeichnet.

4.3.1 Das Spannungsfeld des Projektleiters

Rattay [07, p.25] beschreibt eine Reihe von Einflussgrößen auf das Führungsverhalten von Projektleitern, welche einen Rahmen für das Handeln der Projektverantwortlichen darstellen und nachfolgend kurz beschrieben werden.

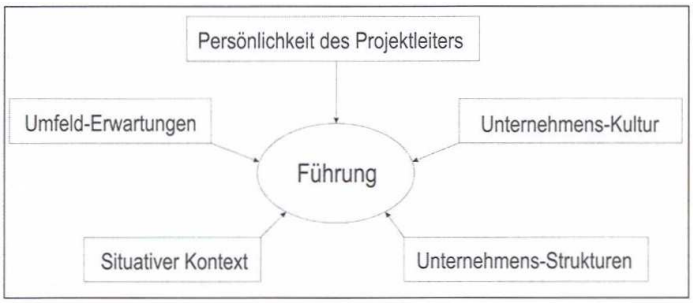


Abbildung 11: Einflusssfaktoren auf die Führung in Projekten [Ratt07, p.25]

Zu der Persönlichkeit des Projektleiters zählen die vererbten Anlagen einer Person, die die Führungsqualitäten ausmachen und bestimmte Einstellungen zu konkreten Situationen erkennen lassen [vgl. ebd.].

Die Unternehmenskultur im allgemeinen, bzw. die Projektkultur als Kulturvariante innerhalb der Unternehmenskultur im besonderen, hat entscheidenden Einfluss auf das Führungsverhalten, da solch zentrale Themen wie Kommunikations- und Umgangsformen, Rituale, sowie Entscheidungs- und Konfliktlösungsmechanismen davon betroffen sind [vgl. ebd.].

Ein weiterer wichtiger Faktor sind die Unternehmensstrukturen, da sie beispielsweise die Unterschriftenregelungen, Kompetenzen- und Verantwortungsverteilung, etc. festlegen, und somit eine Bandbreite für eigenständiges Handeln definieren. Gekoppelt mit der Unternehmenskultur ergibt sich ein System, das beschreibt, mit welchen Sanktionen beispielsweise eine Kompetenzüberschreitung geahndet wird [vgl. ebd.].

Die unterschiedlichen Standpunkte, Erwartungen und Befürchtungen des Projektumfelds (Behörden, Medien, Lieferanten, Kunden, Mitbewerberber – um nur einige zu nennen) haben einen nicht zu unterschätzenden Effekt auf das Handeln des Projektleiters. Je größer das Projekt und somit auch das Umfeld des Projekts, desto diplomatischer muss der Projektleiter agieren können [vgl. ebd.].

Die typischen Phasen eines Projekts – Start-, Arbeits-, Koordinations-, Abschlussphase – haben ebenfalls Auswirkungen auf das Führungsverhalten. Während in der Startphase der Stil durch Fragen und Verstehen gekennzeichnet ist, sollte in den Arbeitsphasen der Freiraum eines jeden Projektmitglieds im

Vordergrund stehen. In den Koordinationsphasen hingegen ist Moderations- bzw. Konfliktlösungsgeschick gefragt, und in der Abschlussphase schließlich steht Reflexionsfähigkeit bzw. die Fähigkeit, ein Projekt zum Abschluß zu bringen, im Vordergrund [vgl. ebd.].

4.3.2 Anforderungen an den Projektleiter

In diesem Kapitel werden grundsätzliche Anforderungen an den Projektleiter besprochen, die die Basis für funktionierende Entscheidungsprozesse darstellen.

Rattay [07, p.73ff.] geht davon aus, dass für das erfolgreiche Leiten von Projekten – und somit für das folgerichtige Treffen von Entscheidungen – eine Reihe von Anforderungen an den Projektleiter erfüllt sein müsssen:

- Erfahrung in der Projektarbeit
- Fachliche Kenntnisse zum Projektinhalt
- Kommunikationsfähigkeit
- Führungsfähigkeit
- Belastbarkeit und Anpassungsfähigkeit

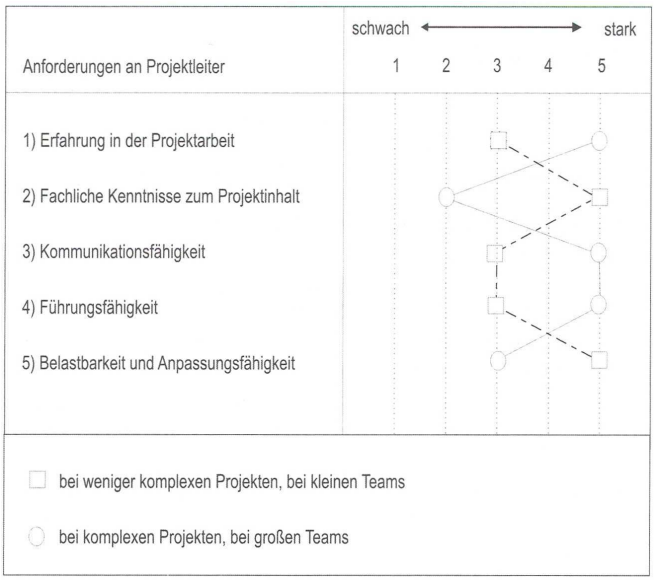


Abbildung 12: Ausprägung sozialer und fachlicher Eigenschaften eines Projektleiters im Vergleich zwischen Klein- und Großprojekten [Ratt07, p.76]

Abhängig von der Größe des Projekts, sollten die einzelnen Anforderungen unterschiedlich stark ausgeprägt sein, wie aus der Abbildung ersichtlich ist.

Zu hinterfragen ist in diesem Zusammenhang die Aussage, dass Projektleiter für Großprojekte weniger stark belastbar sein müssen als für Kleinprojekte. Möglicherweise ist damit gemeint, dass Projektleiter, die kleinere Projekte leiten, oftmals dafür mehrere Projekte gleichzeitig betreuen müssen, und daher das „Context Switching“ zwischen den einzelnen Projekten eine höhere Belastung darstellt, als wenn man sich auf ein einzelnes großes Projekt konzentrieren kann.

4.3.3 Das Urteilsvermögen

Eine der wichtigsten Eigenschaften eines Projektleiters ist das Urteilsvermögen. Noe [09, p.73f.] schlägt einen Katalog von 15 Fragen vor, die eine Bandbreite an Alltagsentscheidungen darstellen und vom Projektleiter jederzeit souverän beantwortet werden können müssen:

- Was ist der Umfang des Projekts?
- Was muss alles geplant werden?
- Welche Elemente des Projektmanagementprozesses gelten und welche sind zu ignorieren?
- Wann muss etwas eskaliert werden?
- Wann sollte man ins Detail gehen und wann überfliegen?
- Wann sollte man es selber tun und wann es delegieren?
- Wem kann man im Projektteam vertrauen?
- Was ist eine annehmbare Risikoebene?
- Was ist ein akzeptables Niveau paralleler Aktivitäten?
- Was ist eine akzeptable Anzahl von Änderungen?
- Wann sollten Sie den Änderungsmanagementprozess durchführen?
- Wann ist es sinnvoll eine Aktivität durchzuführen, die auf einer Annahme basiert?

- Wie viele Ebenen der Projektorganisation braucht man?
- Wann muss man den Schwerpunkt auf die Stakeholder legen?
- Wann ist das Projekt vollständig?

Aus Kapazitätsgründen kann an dieser Stelle nicht näher auf die Fragen eingegangen werden, jedoch finden sie im Praxisteil Beachtung.

4.3.4 Interkulturelle Kompetenz

Gerade bei Großprojekten ist es wahrscheinlich, dass auf Grund der Teamgröße bzw. der unter Umständen geographischen Verteilung des Teams, Menschen zusammenarbeiten, die einen unterschiedlichen kulturellen Hintergrund haben.

Kiesel [04, p.97] erwähnt in diesem Zusammenhang *„die Fähigkeit zum Umgang mit Menschen aus anderen Kulturen“ [ebd.]*. Hierbei geht es darum, dass an den Projektleiter die Anforderung gestellt wird, Vorstellungen, Motive, Probleme, etc. von Teammitgliedern aus fremden Kulturräumen nachvollziehen und entsprechend darauf eingehen zu können.

Kiesel ist der Meinung, dass ein Projektleiter, der schon im nationalen Kulturkreis mit Kollegen nicht zurechtkommt, erst recht bei internationalen (und damit großen) Projekten ein Problem bekommen wird [vgl. ebd.].

4.3.5 Einfluss der Unternehmenskultur

Die Entscheidungen, die tagtäglich im Rahmen einer Projektabwicklung getroffen werden, finden innerhalb eines kulturellen Rahmens statt, in dem die Teammitglieder erfolgreich und zufrieden arbeiten können.

Noe [09, p.113f.] umreißt diesen Rahmen mit einer Reihe von Fragen, die das kulturelle Klima einer Organisation widerspiegeln:

- Wie ist die Projektkultur im Unternehmen definiert und wie wird sie gelebt?
- Wie wurde das Projektteam aufgebaut, auf freiwilliger Basis oder wurden auch Personen gezwungen?

- Wie wird die Kultur empfunden und wie ist der Wohlfühlfaktor im Projektteam?
- Wurde der Umgang untereinander (Spielregeln) einvernehmlich festgelegt?
- Wie hat der Projektmanager das Projekt und die Ziele präsentiert und kommuniziert?
- Wie viel Freiraum, Vertrauen und Verantwortung wird den Projektmitgliedern für die Erledigung ihrer Aufgaben zugebilligt?
- Wie wird das Projektteam unterstützt und wie werden die persönlichen Ziele einzelner Teammitglieder berücksichtigt?
- Wie gestaltet der Projektmanager Motivation, Belobigungen und Incentivemaßnahmen, Weiterbildung, etc.?
- Wie geht der Projektmanager mit Konflikten im Team oder mit einzelnen Teammitgliedern um?

Alle diese Themen haben direkten Einfluss auf den Umgang mit den Teammitgliedern, was sich in weiterer Folge direkt auf die Arbeitsweise und somit auf die Art und Weise der Entscheidungsfindung auswirkt.

Im Praxisteil werden daher einige dieser Fragen thematisiert.

4.3.6 Führungsstile

Der Führungsstil des Projektmanagers ist die Basis für sämtliche Entscheidungsprozesse. In diesem Kapitel wird daher besprochen, welche Auswirkungen unterschiedliche Stile haben können.

Noe [09, p.101] ist der Meinung, dass es keinen idealen Führungsstil gibt, sondern dass je nach Situation der Manager auf seine Mitarbeiter einwirken muss.

Es gibt jedoch Stile, die zu vermeiden sind, da sie die Motivation der Projektmitglieder herabsetzen und daher die Qualität des Produkts negativ beeinflussen können. Nachfolgend werden diese Stile auszugsweise kurz beschrieben [vgl. ebd., p.102-108].

Der Diktator führt seine Mitarbeiter mit eiserner Hand, lässt keine andere Meinung oder Widerspruch zu. Dadurch, dass kein Vertrauen aufgebaut wird und ein Klima der Angst vorherrscht, können sich die Teammitglieder nicht entfalten und daher auch nicht ihr volles Potential abrufen [vgl. ebd.].

Der Macho steht für den Typus, immer alles zu wissen und nie für etwas schuld zu sein, wenn einmal etwas schiefgeht. Er ist immer aktiv und versucht stets, seine Grenzen auszuloten. Da dem Macho Eigennutz und Selbstbefriedigung wichtiger ist als der gemeinsame Nutzen, stellt er eine Gefahr für den Projekterfolg dar [vgl. ebd.].

Der Sachverwalter ist ein detailverliebter Bürokrat, der oftmals den allgemeinen Fortschritt des Projekts aus den Augen verliert, weil er wie besessen über seinen Plänen sitzt anstatt mit seinen Kollegen zu kommunizieren [vgl. ebd.]

Berkun [09, p.301] schlägt eine Reihe von Verhaltensweisen vor, um das Vertrauen der Teammitglieder zu erlangen, und auf diese Weise die verliehene Macht positiv einzusetzen:

- Erfolgreiche Zusagen bauen Vertrauen auf.
- Inkonsistentes Verhalten in wichtigen Angelegenheiten zerstört Vertrauen.
- Während sich verliehene Macht aus der organisatorischen Hierarchie ergibt, entsteht verdiente Macht aus den Reaktionen der Kollegen auf die gesetzen Handlungen. Obwohl beide Formen wichtig sind, ist verdiente Macht mehr wert.
- Die Delegation bestimmter Arbeiten schafft Vertrauen im Team.
- Auf Probleme sollte in einer Art und Weise reagiert werden, sodass die Mitarbeiter keine Angst haben müssen, Probleme rechtzeitig bekannt zu geben.
- Die Basis guten Führens ist ein gesundes Selbstvertrauen. Selbsterkenntnis ist ein probater Weg, Selbstvertrauen zu entwickeln.

Zu diesem Thema gäbe es noch etliches zu Schreiben, was aus Platzgründen jedoch leider nicht möglich ist. Stattdessen wird diese Thematik im Praxisteil behandelt.

4.4 Der kreative Problemlösungsprozess

Der kreative Problemlösungsprozess hat den Zweck, ausgehend von einem Problem mehrere Lösungsansätze zu definieren, und über eine Bewertung den geeignetsten Lösungsansatz zu bestimmen.

Rattay [07, p.169ff.] beschreibt den kreativen Lösungsprozesses wie folgt.

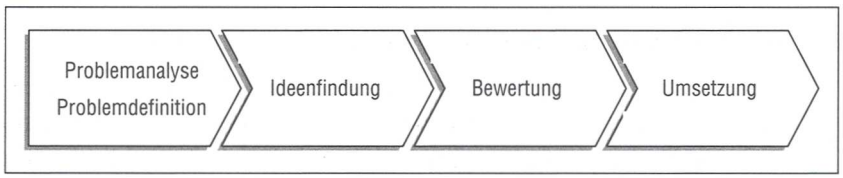


Abbildung 13: Der kreative Problemlösungsprozess [Ratt07, p.169]

4.4.1 Allgemeines

In **Stufe eins** wird das zu lösende Problem ausführlich analysiert, bzw. wird eine Definition des Problems vorgeschlagen. Das akribische Arbeiten in dieser Stufe ist von höchster Wichtigkeit, da die erlangten Erkenntnisse als Fundament für alle weiteren Stufen dienen [vgl. ebd.].

Stufe zwei – die Ideenfindung – dient dem Zweck, möglichst viele unterschiedliche Lösungsmöglichkeiten für die in Stufe eins analysierten Probleme zu generieren [vgl. ebd.].

In der **dritten Stufe** werden sämtliche Ideen aus Stufe zwei mit Hilfe unterschiedlicher Methoden bewertet, wobei zu beachten ist, dass diese Stufe sehr viel Zeit in Anspruch nehmen kann. Als Grundregel lautet:

> *„(...) es bewertet immer der, der mit der Lösung leben muss (...)“ [Ratt07, p.170]*

In **Stufe vier** werden schließlich die Lösungen, die in Stufe drei am besten bewertet wurden, umgesetzt [vgl. ebd.].

Anzumerken ist, dass die Stufen „Ideenfindung“ bzw. „Bewertung“ iterativ durchlaufen werden können, das bedeutet, dass diese Phasen öfters hintereinander in einem Zyklus durchgeführt werden können [vgl. ebd.].

4.4.2 Rollen im Prozess

Zur Umsetzung des kreativen Denkprozesses sind drei Funktionen / Rollen notwendig:

Der **Problemsteller** übernimmt die Problemdefinition bzw. -bewertung. Es bietet sich an, dass der Problemsteller erstens direkte Berührungspunkte zu der Problemsituation aufweist, und zweitens an der Umsetzung der Lösungsidee beteiligt ist [vgl. ebd.].

Die **Kreativgruppe** – üblicherweise vier bis zwölf Personen – übernimmt die Ideenfindung. Eine optimale Kreativgruppe setzt sich homogen bzgl. der sozialen Stellung, und heterogen hinsichtlich ihres Fachwissens zusammen [vgl. ebd.].

Das Leiten der Gruppe durch den Prozess obliegt dem **Moderator**. Er ist für die Zusammensetzung der Gruppe an sich verantwortlich, bzw. ist er es, der die Gespräche innerhalb der Gruppe moderiert [vgl. ebd.].

4.4.3 Techniken zur Entwicklung von Alternativen

Die Kreativgruppe bedient sich verschiedenster Techniken, um in der Phase der Ideenfindung ein möglichst breites Spektrum an Alternativen zu entwickeln [vgl. ebd.].

Analogien und Metaphern eignen sich hervorragend dafür, „über den Tellerrand“ zu blicken, da ein neuer Kontext eines ähnlichen Problems auch neue Perspektiven zulässt, die unter Umständen in den ursprünglichen Kontext wieder zurückübertragen werden können [vgl. ebd.].

Mittels **Assoziationen** können mentale Querverbindungen zwischen Ideen geschaffen werden [vgl. ebd.].

Beim **Mind Mapping** wird das Problem bildlich ins Zentrum gestellt, davon ausgehend werden in Beziehung stehende Ideen in Form von Ästen konstruiert [vgl. ebd.].

Das **Brainstorming** dient dazu, möglichst viele Ideen spontan zu generieren – auch auf den ersten Blick absurde Gedanken dürfen geäussert werden. Auf diese Weise können eingefahrene Denkmuster aufgebrochen werden und neue – brauchbare – Lösungen entwickelt werden [vgl. ebd.].

Bei der **Exkursionstechnik** werden die Teilnehmer dazu angehalten, eine gedankliche Reise durch imaginäre oder reale Welten vorzunehmen. Auf diese Weise können unter Umständen unkonventionelle Lösungsmöglichkeiten entstehen [vgl. ebd.].

Strukturierte Kleingruppen werden bei der **nominalen Gruppentechnik** dazu eingesetzt, Lösungen zu erarbeiten. Diese Technik eignet sich immer dann, wenn der Einfluss einer dominanten Persönlichkeit auf die gesamte Gruppe minimiert werden soll [vgl. ebd.].

Ähnlich dem Brainstorming werden beim **Storyboarding** die Gedanken der Teilnehmer zunächst erfasst, zusätzlich jedoch auch visualisiert. Im Zuge der Visualisierung werden erkannte Beziehungen zwischen den Ideen eingezeichnet, was dazu führen kann, dass gänzlich neue Ideen entstehen [vgl. ebd.].

4.4.4 Ergebnisse von Problemlösungsprozessen

In der Regel wird als Ergebnis eines Problemlösungsprozesses eine optimale Lösung erwartet. Seibert [98, p.81ff.] geht jedoch davon aus, dass die absolut optimale Lösung nicht immer erreichbar ist, und zwar aus folgenden Gründen:

- Es können nicht alle realisierbaren Lösungsalternativen entdeckt werden.
- Die Kriterien, die eine optimale Lösung beschreiben, sind oft unklar.
- Eine sorgfältige Suche und Bewertung nach Alternativen wurde auf Grund von Zeitdruck oder begrenzten finanziellen Mitteln verhindert.

Aus diesem Grund wird in der Praxis – da immer von Zeit- und finanziellem Druck auszugehen ist – gar nicht erst versucht, eine perfekte Lösung finden,

sondern man gibt sich mit einer Lösung zufrieden, die „gut genug“ ist [Seib98, p.82]. Aus einer optimalen Lösung wird eine optimale Enscheidung, was die folgende Graphik veranschaulicht:

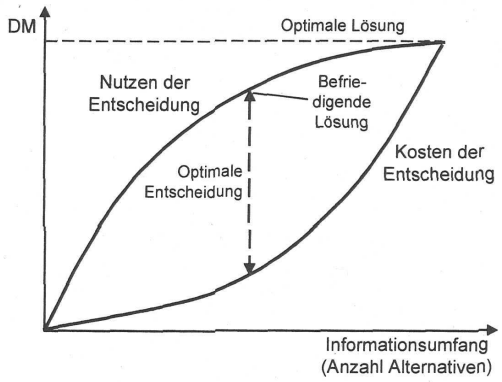


Abbildung 14: Kosten und Nutzen in Problemlösungsprozessen [Seib98, p.82]

Abschließend merkt Seibert noch an, dass es auf die Tragweite der Entscheidung ankommt, die letztendlich darüber entscheidet wie viel in das Finden einer optmalen Lösung investiert wird.

4.4.5 Dokumentation im Problemlösungsprozess

Haberfellner et. al. [92, p.55] gibt zu bedenken, dass die Dokumentation von Ergebnissen und Zwischenergebnissen im Zuge des Problemlösungsprozesses sehr wichtig ist. Der Grund dafür ist, dass einerseits die Glaubwürdigkeit erhöht wird, da Begründungen schriftlich festgehalten und daher jederzeit eingesehen werden können. Andererseits ist es im Rahmen von Retrospektiven möglich und durchaus nützlich, auf eine saubere Dokumentation eines Problemlösungsprozesses zurückgreifen zu können.

4.5 Meetings & Co.

Eine nicht unerhebliche Anzahl an Entscheidungen wird im Rahmen von Steuerungssitzungen getroffen – dementsprechend wichtig ist deren korrekte Durchführung. Blanckenburg et. al. [05, p.219] sind der Meinung, dass gut moderierte Sitzungen äusserst positive Effekte auf viele Bereiche der Projektarbeit haben können, beispielsweise:

- Effiziente Gestaltung von Besprechungen und Arbeitssitzungen

- Erhöhung des Engagements der Mitarbeiter
- Gewinnung von ausreichend Zeit für wichtige Probleme
- Systematische Bündelung der Problemlösungspotentiale einer Arbeitsgruppe
- Erhöhung von Motiviation, Kreativität und Arbeitszufriedenheit der Beteiligten

Doch um den vollen Nutzen einer erfolgreichen Sitzung zu erreichen, ist ebenso auf eine geeignete Vor- bzw. Nachbereitung zu achten.

Die folgenden drei Unterkapitel orientieren sich nach Rattay [07, p.152-167].

4.5.1 Sitzungsvorbereitung

Eine gute Sitzungsvorbereitung ist das Um und Auf für eine gelungene Sitzung.

Zunächst ist es wichtig, den Anlass der Sitzung zu definieren, und davon ausgehend die Sitzungsziele abzuleiten. Diese Ziele sind den teilnehmenden Personen im Vorfeld bekannt zu geben, da sie letztendlich die Themen vorgeben, die in der Sitzung behandelt werden, und es jedem Teilnehmer möglich sein soll, sich entsprechend vorzubereiten.

Sobald die Ziele definiert sind, kann daran gegangen werden die Liste der Sitzungsteilnehmer zusammen zu stellen. Es empfiehlt sich der altbekannte Leitsatz „So wenige wie möglich, so viele wie notwendig“. Neben Projektleiter und Kernteam können – falls die Notwendigkeit dazu besteht – Spezialisten oder Vertreter diverser Umfeldgruppen hinzugenommen werden. Im übrigen ist darauf zu achten, dass die Zusammensetzung der Sitzungsteilnehmer langfristig konstant bleibt, da sich eine personelle Kontinuität positiv auf die Entscheidungsfähigkeit auswirkt.

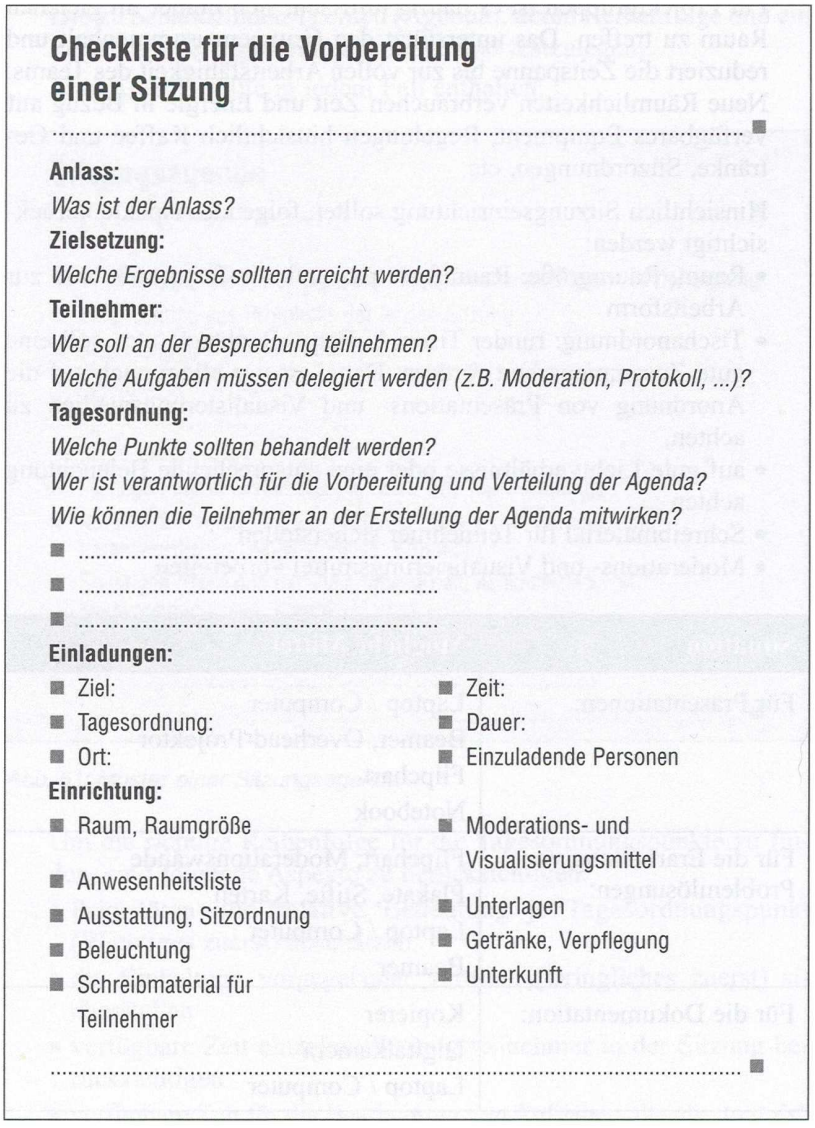
Als nächstes ist der Ablauf der Sitzung, also die zu behandelnden Themen festzulegen. Die Themen sind zu priorisieren, und es ist darauf zu achten, den einzelnen Punkten ein entsprechendes Zeit-Budget zuzusichern.

Der nächste Schritt ist das Verfassen bzw. Versenden der Sitzungseinladungen an die Teilnehmer. Die Einladung sollte neben Zeit und Ort auch das Ziel bzw. die

Tagesordnung enthalten. Zusätzlich sollte eine Liste der teilnehmenden Personen enthalten sein.

Als letzte Tätigkeit im Rahmen der Sitzungsvorbereitung ist die entsprechende Infrastruktur vorzubereiten. Dazu zählt das rechtzeitige Reservieren der Räumlichkeiten, sowie das bereitstellen von Hilfsmitteln wie beispielsweise Beamer, Flipchart, aber auch Verpflegung wie Getränke und kleine Imbisse.

Rattay empfiehlt, mittels einer Checkliste die Arbeiten der Sitzungsvorbereitungen zu kontrollieren. Eine solche Checkliste könnte wie folgt aussehen:



Checkliste für die Vorbereitung einer Sitzung

.....

Anlass:
Was ist der Anlass?
Zielsetzung:
Welche Ergebnisse sollten erreicht werden?
Teilnehmer:
Wer soll an der Besprechung teilnehmen?
Welche Aufgaben müssen delegiert werden (z.B. Moderation, Protokoll, ...)?
Tagesordnung:
Welche Punkte sollten behandelt werden?
Wer ist verantwortlich für die Vorbereitung und Verteilung der Agenda?
Wie können die Teilnehmer an der Erstellung der Agenda mitwirken?

- .....
- .....
- .....

Einladungen:

- Ziel:
- Tagesordnung:
- Ort:
- Zeit:
- Dauer:
- Einzuladende Personen

Einrichtung:

- Raum, Raumgröße
- Anwesenheitsliste
- Ausstattung, Sitzordnung
- Beleuchtung
- Schreibmaterial für Teilnehmer
- Moderations- und Visualisierungsmittel
- Unterlagen
- Getränke, Verpflegung
- Unterkunft

.....

Abbildung 15: Muster für eine Checkliste zur Sitzungsvorbereitung [Ratt07, p.156]

4.5.2 Sitzungsdurchführung

Eine erfolgreiche Sitzungsdurchführung lässt sich anhand einiger weniger strikt einzuhaltender Punkte realisieren.

Zu den Aufgaben des Projektleiters zählt zunächst der Start der Sitzung, der neben der Begrüßung der Teilnehmer einen kurzen Überblick über die Tagesordnung der Besprechung geben soll. Wenn es sich um die erste Sitzung dieser Art handelt, sollten Sitzungsregeln entweder gemeinsam entwickelt oder bekannt gegeben werden. Die Führungsaufgaben, denen der Projektleiter im Besprechungsverlauf nachkommen muss, umfassen einerseits die Durchsetzung allgemeiner Gesprächsmanieren (Aussprechen lassen, Störungen unterbinden, etc.), andererseits ist darauf zu achten, dass von den eigentlichen Themen nicht abgedriftet wird, bzw. dass sich Diskussionen in eine Richtung entwicklen, sodass eine konkrete Lösung als Resultat zu erwarten ist. Das Herbeiführen von Entscheidungen im Rahmen einer Diskussion wird als lösungsorientierte Steuerung bezeichnet, was vom Projektleiter glänzend beherrscht werden sollte. Der Abschluss der Sitzung umfasst das Vereinbaren von Aktivitätsplänen, sowie das Zusammenfassen von (Zwischen-)ergebnissen, beispielsweise in Form von Feedback-Runden.

Die Sitzordnung – ein immer wieder unterschätzter Faktor – kann Einfluss auf verbale bzw. nonverbale Kommunkation haben. Es ist darauf zu achten, dass die Sitzordnung nicht zu eng ist, und dass sich Gruppen mit unter Umständen gegensätzlichen Interessen nicht frontal gegenüber sitzen. Auch eine sich eingeschliffene rituelle Sitzordnung ist zu berücksichtigen. Für den Fall der Fälle kann mittels Platzkärtchen eine Sitzordnung vorgegeben werden.

Störungen jeglicher Art sollten von Beginn an thematisiert werden. Hierbei kann es sich um Telefonanrufe oder ähnliches handeln. Mitunter ist es hilfreich, länger dauernde Sitzungen durch Pausen zu unterbrechen, um störende Unterbrechungen im Vorfeld zu vermeiden.

Oft werden Besprechungen auf Grund des mangelnden Kommunikationsstils als zeitraubend und unnötig angesehen. Es liegt daher in der Verantwortung des Moderators, dafür zu sorgen, dass eine entsprechende Kommunikationskultur die

Grundlage dafür bietet, dass sämtliche Sitzungsteilnehmer das Gefühl haben, ausreichend zu Wort zu kommen um ihre Meinungen entsprechend zu äussern.

4.5.3 Sitzungsnachbereitung

Das während der Sitzung angefertigte Sitzungsprotokoll dient als Grundlage für die Sitzungsnachbereitung. Die Entscheidungen und Vereinbarungen, die während der Sitzung beschlossen wurden, stehen im Protokoll sämtlichen Teilnehmern zur Verfügung, womit das Protokll als Arbeitsgrundlage bzw. Checkliste benutzt werden kann.

Aus diesem Grund sollte das Protokoll möglichst bald nach der Sitzung allen Teilnehmern zur Verfügung stehen.

4.5.4 Meeting-Kultur

Abschließend noch ein paar Worte zur Meeting-Kultur. Hindel et. al. [09, p.134] merken an, dass eine praktizierte Meeting-Kultur zu den wichtigsten Spielregeln in einem Projekt gehören, und schlagen daher vor, die folgenden Punkte immer einzuhalten:

- *„Nie ohne Agende ein Meeting beginnen.*
- *Nie zu spät zu einem Meeting kommen.*
- *Nie ein Meeting ohne Rollendefinition durchführen.*
- *Nie ein Meeting ohne Protokoll veranstalten.*
- *Nie ein Meeting zeitlich überziehen“ [ebd.]*

4.6 Die Entscheidungsfindung

Rattay [07, p.180ff.] vertritt die Meinung, dass Entscheidungen, die in Gruppen getroffen werden, viele Vorteile gegenüber Einzelentscheidungen haben. Gerade bei komplexen Problemen bzw. emotional schwierigen Situationen können Gruppenentscheidungen unter Umständen eine höhere Qualität erzielen, als dies bei einer Entscheidungsfindung durch eine Einzelperson der Fall ist.

4.6.1 Treffen von Entscheidungen

Auch bei Entscheidungen in der Gruppe gibt es unterschiedliche Möglichkeiten, eine Entscheidung zu treffen – wie nachfolgend beschrieben wird [vgl. ebd.].

Bei der **Einzelentscheidung** wird zwar in der Gruppe über die zu treffende Entscheidung diskutiert, letztendlich entscheidet jedoch nur der Projektleiter [vgl. ebd.].

Bei der Entscheidung durch Abstimmung entscheidet entweder die **einfache Mehrheit**, oder eine **Zweidrittelmehrheit** über einen Sachverhalt [vgl. ebd.].

Bei der **Expertenentscheidung** wird den ausgewiesenen Experten eines Fachgebietes die Entscheidung für Themen überlassen, die in ihr Fachgebiet fallen [vgl. ebd.].

Bei der **Konsensentscheidung** wird innerhalb der Gruppe ein Kompromiss eingegangen, wobei auf diese Weise selten das aus Sicht einer Person optimale Ergebnis zustande kommt [vgl. ebd.].

4.6.2 Entscheidungsstile

Neben den oben erwähnten Entscheidungsstilen führt Rattay die folgenden unterschiedlichen Formen der Entscheidungsfindung an:

„Ich entscheide alleine“: Die Entscheidung erfolgt durch den Projektleiter ausschließlich auf Grund des eigenen Wissens, wobei die Mitarbeiter vor vollendete Tatsachen gestellt werden [vgl. ebd.].

„Beantworte mir meine Fragen“: Informationen werden von den Mitarbeitern eingeholt, ohne jedoch die befragten Mitarbeiter darüber zu informieren, dass das Einholen der Information zu einer Entscheidungsfindung beiträgt [vgl. ebd.].

„Gib mir deine Ratschläge“: Die Person, die die Entscheidung letztendlich alleine trifft, holt jedoch im Vorfeld Meinungen, Analysen, etc. von den Mitarbeitern ein [vgl. ebd.].

„Lasst uns diskutieren“: In der Gruppe werden offen Alternativen diskutiert die dem Entscheider helfen eine Entscheidung zu treffen [vgl. ebd.].

„Wir finden gemeinsam eine Lösung“: Der Projektleiter moderiert die Entscheidungsfindung durch Diskussion in der Runde, wobei letztendlich in der Gruppe durch Mehrheitsfindung entschieden wird [vgl. ebd.].

In unterschiedlichen Situationen können mitunter unterschiedliche Entscheidungsstile angebracht sein, wie die folgende Übersicht darstellt:

	Entscheidungsstil			
Situation	Autoritär durch den Projektleiter	Projektleiter nach Mitarbeiter-information	Projektleiter nach Team-besprechung	Entscheidung durch das Projektteam
Die nötigen Informationen sind im Team verteilt.	☹	😐	😐	☺
Die Akzeptanz der Entscheidung ist besonders wichtig.	☹	😐	😐	☺
Die Entscheidungs-schnelligkeit hat Vorrang.	☺	😐	☹	☹
☺ optimaler Entscheidungstyp ☹ nicht anzustrebender Entscheidungstyp 😐 möglicher, nicht idealer Entscheidungstyp				

Abbildung 16: Entscheidungsstile in Projekten [Ratt07, p.182]

4.6.3 Akzeptanz von Gruppenentscheidungen

> *„Die Art der Einbindung des einzelnen Gruppenmitglieds in den Prozess der Entscheidungsfindung macht den Grad der Akzeptanz, der Zustimmung und Verbindlichkeit der Gruppenentscheidung von Seiten des einzelnen Mitarbeiters aus.“ [Ratt07, p.182]*

Rattay [07, p.182f.] unterscheidet die folgenden Stufen zunehmender Zustimmung und Verbindlichkeit:

An unterster Stelle findet sich die **Abstinenz** – einzelne Gruppenmitglieder werden beim Entscheidungsprozess ausgeblendet, was die Akzeptanz durch eben jene Mitglieder auf ein Minimum absinken lässt [vgl. ebd.].

Bei der **Prozessakzeptanz** wird zwar das Vorgehen zur Entscheidungsfindung an sich goutiert, weil es sich unter Umständen um eine Mehrheitsentscheidung

handelt, das Resultat kann jedoch für einzelne Gruppenmitglieder nicht zufriedenstellend sein – für den Fall, dass sie überstimmt wurden [vgl. ebd.].

Das Anschließen an die Meinung von Experten wird als **inhaltliche Akzeptanz** bezeichnet. Hauptgrund dafür ist oftmals mangelndes Problemverständnis oder auch fehlendes Hintergrundwissen. Dadurch, dass die Entscheidung nicht mitvertreten wird, besteht auch keine inhaltliche Verbindlichkeit [vgl. ebd.].

Im Gegensatz dazu wird bei der **inhaltlichen Zustimmung** die eigene Position aufgegeben, um eine Gruppenlösung zu ermöglichen – ohne der Argumentation vollständig zu folgen, jedoch mit Verständnis für die gegenteilige Meinung [vgl. ebd.].

Die höchste Stufe der Akzeptanz stellt die **inhaltliche Identifikation** dar. Alle Beteiligten kommen im Zuge der Debatte zu einer einhelligen Meinung, und können die getroffene Entscheidung daher auch persönlich voll vertreten [vgl. ebd.].

4.6.4 Nachhaltigkeit von Entscheidungen

Eine Entscheidung kann dann als nachhaltig bezeichnet werden, wenn die Entscheidung nicht nur kommuniziert, sondern auch umgesetzt wurde. Schließlich ist es eine Sache, eine Entscheidung zu treffen; wenn sich nicht umgesetzt wird – aus welchen Gründen auch immer – trägt jedoch die beste Entscheidung nichts zum Projekterfolg bei.

Ein Erkennungsmerkmal für Entscheidungen, die zwar getroffen, aber nicht konsequent umgesetzt wurden, ist nach Eckstein [04, p.39] das Auftreten der immer wieder gleichen Fragen bzw. Diskussionen. Einer der Gründe, die einer Umsetzung entgegenwirken, ist oftmals ein Defizit an Verantwortung. Ein Mangel an Verantwortung begünstigt jedoch das Phänomen des **undefinierten Aufgabenbereichs**, der nach Eckstein wie folgt gekennzeichnet ist:

Bei der **mehrfachen Aufgabenverantwortung** fühlen sich mehrere Personen für die gleiche Aufgabe verantwortlich. Der Grund dafür kann möglicherweise sein, dass keine der Personen weiß, dass es noch andere Kollegen gibt die sich für ein und dasselbe Thema verantwortlich fühlen. Dass eine Aufgabe mehrfach

bearbeitet wird, ist noch das geringste Übel – problematisch wird es, wenn widersprüchliche Ergebnisse zu Konflikten führen, weil niemand von seiner Lösung abweichen möchte [vgl. ebd.].

Anders bei der **nichtexistierenden Aufgabenverantwortung**. Hier fühlt sich niemand für eine Aufgabe verantwortlich, weil jeder annimmt, dass dies das Thema von jemand anderem sei. Oftmals führt diese Situation nach deren Aufklärung zu gegenseitigen Beschuldigungen, nicht die Verantwortung übernommen zu haben – was zu Problemen bzgl. des Teamzusammenhalts führen kann [vgl. ebd.].

Brown et. al. [04, p.164] führen einen Aspekt an, der nicht zu vernachlässigen ist. Die architektonischen Entscheidungen, die üblicherweise von einer kleinen Personengruppe getroffen werden, werden in der Regel von einer sehr viel größeren Gruppe von Entwicklern implementiert. Es ist daher von immenser Wichtigkeit, dass die Architekten einen geschickten Umgang mit den Entwicklern an den Tag legen, da die Personen, die die Architektur letztendlich umsetzen, von dem Design „überzeugt“ werden müssen. Dadurch wird erreicht, dass sich das Projektteam mit der Architektur identifiziert, und es ist in solch einem Klima nicht zu erwarten, dass Entwickler passiven Widerstand leisten, weil sie mit den Grundzügen des Gesamtsystems nicht einverstanden sind.

4.6.5 Das Paradoxon der Übereinstimmung

Einstimmigkeit in Gruppen, bzw. die scheinbar vorherrschende Übereinstimmung zu einem Thema, kann nach Rattay [07, p.184f.] – ähnlich dem Auftreten von gegenteiligen Meinungen – unter gewissen Umständen problematisch sein. In diesem Zusammenhang sind zwei Phänomene anzuführen:

- Unter „**Group think**“ wird verstanden, dass einzelne Projektmitglieder ihre Meinung aufgeben, um dem sozialen Druck der Gruppe – die möglicherweise eine einheitliche Meinung vertritt – nicht ausgesetzt zu sein. Das Problem, dass dadurch das kreative Lösungspotential reduziert wird, kann von erfahrenen Projektleitern minimiert werden, indem duch geeignete Maßnahmen absichtlich Konflikte induziert werden. Dies kann

bspw. durch das Hinzuziehen fremder Personen erreicht werden, oder auch durch das Hervorheben unterschiedlicher Sichtweisen. Oft reicht es auch schon, das Problem direkt anzusprechen und die Mitglieder zu ermutigen, ihre eigene Meinung zu vertreten.

- Das „**Abilene-Paradoxon**“ tritt auf, wenn es im Zuge einer Teamentscheidung zu einer überwältigenden Übereinstimmung zu einem Thema kommt, obwohl in Wirklichkeit niemand diese Entscheidung voll inhaltlich vertritt. Zu dieser Situation kann ein zufällig eingebrachter Lösungsvorschlag eines Teammitglieds führen, der plötzlich von allen anderen Mitgliedern akzeptiert wird, in dem gegenseitigen Glauben, das komplette Team vertrete dieselbe Meinung. Oft tritt eine solche Situation auf, wenn die offene Kommunikation im Projektteam aus welchen Gründen auch immer nicht funktioniert.

4.7 Widerspruchsmanagement

Sobald ein Projekt jene Größe überschritten hat, sodass mehrere – wenn auch auf unterschiedlichen Ebenen agierende – entscheidungsberechtigte Personen aufeinandertreffen, sind Widersprüche in der Verfolgung von Zielen vorprogrammiert.

Rattay [07, p.236f.] schlägt in einer solchen Situation ein aktives und professionelles Widerspruchsmanagement vor, das die in der Abbildung ersichtlichen Phasen umfasst.

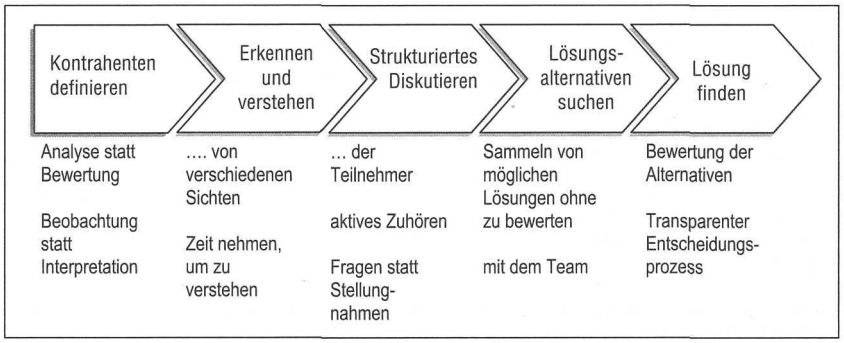


Abbildung 17: Die Phasen des aktiven Widerspruchsmanagements [Ratt07, p.236]

Nachfolgend werden die Schritte genauer beschrieben.

4.7.1 Definition der Kontrahenten

Die unterschiedlichen Meinungsträger werden vom Projektleiter identifiziert und letztendlich dazu gebracht, sich geordnet an einen Tisch zu setzen. Das ist der erste Schritt, der eine konstruktive Diskussion überhaupt erst möglich macht [vgl. ebd.].

4.7.2 Sammlung der unterschiedlichen Meinungen

Sämtliche Beteiligten bekommen – angeleitet durch den Projektleiter – die Möglichkeit, ihre unterschiedlichen Standpunkte vor der Gruppe darzulegen. Der Moderator muss in dieser Phase darauf achten, dass nicht Diskussion und Widersprüche, sondern aufmerksames Zuhören sowie Verständnisfragen im Vordergrund stehen [vgl. ebd.].

4.7.3 Diskussion

Sobald die unterschiedlichen Sichtweisen der Kontrahenten vorgebracht sowie gegenseitig verstanden wurden, kann auf Basis dessen eine Diskussion beginnen. Die Diskussion sollte stimmungstechnisch von solchen Fragen wie bspw. „Welche neuen Erkenntnisse gibt es nun?“, oder auch „Was bedeuten die gehörten Aussagen für die eigenen, bisher vertretenen Standpunkte?“ geleitet sein [vgl. ebd.].

Eine Zusammenfassung der Gesamtsituation sollte am Ende dieser Phase stehen. Wichtig dabei ist, dass es zu keiner Be- bzw. Entwertung einzelner Meinungen kommt [vgl. ebd.].

4.7.4 Suche nach Lösungsalternativen

Auf Basis der vorangegangenen zwei Schritte kann nun mit der Suche nach Lösungsalternativen begonnen werden. Es gibt eine Reihe von Techniken, die das Aufspüren von Ideen unterstützen (wie in Kapitel „Der kreative Problemlösungsprozess“ beschrieben ist) [vgl. ebd.].

4.7.5 Lösungsfindung

Die Bewertung der im vorherigen Schritt erarbeiteten Lösungsalternativen führt letztendlich zu einer Gegenüberstellung der Vor- bzw. Nachteile der einzelnen Lösungsansätze. Einen guten Projektleiter bzw. Moderator zeichnet aus, dass bereits in der Bewertungsphase für die Gruppe deutlich wird, welche Lösungen am geeignetsten erscheinen [vgl. ebd.].

Ist dies nicht der Fall, muss mittels eines geeigneten Auswahlverfahrens eine Entscheidung für eine Lösung getroffen werden (siehe Kapitel „Entscheidungsfindung in Steuerungssitzungen“) [vgl. ebd.].

4.8 Vorgehensmodelle

Vorgehensmodelle dienen dem Zweck, für bestimmte Phasen eines Projekts definierte Handlungs- bzw. Arbeitsanweisungen bereit zu stellen, um in einer Art „best practice“ immer wiederkehrende Abläufe zu normieren. Die Idee ist, die Effizienz zu erhöhen, in dem nicht jedesmal das Rad neu erfunden werden muss.

Hindel et. al. definieren den Begriff „Vorgehensmodell“ wie folgt:

> *„Ein Vorgehensmodell stellt Methoden und Elemente der Softwareentwicklung inklusive des Projektmanagements zu Prozessen und Projektphasen eines standardisierten Projektablaufs zusammen.“ [HiHö09, p.14]*

4.8.1 Sinnhaftigkeit von Vorgehensmodellen

In kleinen Unternehmen ist zu beobachten, dass es oftmals keine expliziten Vorgehensmodelle gibt, da die Definition bzw. Wartung der Modelle einen nicht unerheblichen Aufwand bedeutet, der in keiner Relation zum Geschäftsvolumen steht. Ganz anders bei großen Projektorganisationen. Da hier immer wieder gleich gelagerte Arbeitsabläufe stattfinden, können Vorgehensmodelle die Qualität der Arbeit erheblich verbessern, da volles Augenmerk auf die Tätigkeit selbst gerichtet werden kann, und keine unnötigen Ressourcen in das „Erfinden“ von funktionierenden Vorgehensweisen investiert werden muss.

Eckstein [04, p.146] gibt jedoch zu Bedenken, dass es unmöglich ist **einen** Prozess zu definieren, der für **alle** Arten von Projekten gleichermaßen anwendbar

ist. Eckstein zitiert in diesem Zusammenhang Abraham Maslow mit einer sehr passenden Metapher:

> *„Wenn man als Werkzeug nur einen Hammer hat, sieht jedes Problem plötzlich wie ein Nagel aus.“ [Maslow in ebd.]*

Um einen unternehmensweiten Prozess etablieren zu können, muss dieser äusserst abstrakt gehalten werden, um auf sämtliche Projekte anwendbar zu sein. Ein guter Projektleiter wird sich daher nicht sklavisch an einen strikten Plan halten, sondern immer in der jeweiligen Situation abwägen, in wieweit der Prozess entsprechend angepasst angewendet werden kann [vgl. ebd.].

4.8.2 Theorie und Praxis von Vorgehensmodellen

Gerade in größeren Unternehmen gibt es immer wieder eigene Abteilungen, deren einzige Aufgabe es ist, Vorgehensmodelle für Geschäftsprozesse des Unternehmens zu entwerfen. Oftmals werden diese Abteilungen auch gleich mit der Kompetenz ausgestattet zu überprüfen, ob die ausgearbeiteten Arbeitsanweisungen auch wirklich von den Fachabteilungen umgesetzt werden.

Yourdon [99] spricht in diesem Zusammenhang von einer „Methodenpolizei“, da bei Fachabteilungen mitunter das Gefühl entsteht, von der Methoden- bzw. Verfahrensabteilung kontrolliert, und bei Missachtung der vorgeschriebenen Vorgehensmodelle mit Sanktionen belegt zu werden. Senge bringt es auf den Punkt:

> *„Designing policies and strategies that no one can implement because they don’t understand or agree with the thinking behind them has little effect.“ [Seng94, p.342]*

Eine solche Situation kann zu absurden Auswüchsen führen, was sich beispielsweise darin äussert, dass plötzlich drei Vorgehensmodelle existieren, wie Eckstein [04, p.148] zu bedenken gibt:

Das **offizielle Vorgehensmodell** is jenes, das von der Verfahrensabteilung entwickelt wurde, und als Grundlage für die Arbeit in den Fachabteilungen dienen soll [vgl. ebd.].

Das **wahrgenommene Vorgehensmodell** ist jenes, das von den Projektmitgliedern anderen Kollegen gegenüber als das Modell vorgegeben wird,

nach dem sie selbst arbeiten. Tatsächlich kann jedoch der angegebene Ansatz dem tatsächlichen total Widersprechen [vgl. ebd.].

Das **tatsächliche Vorgehensmodell** ist jenes, das vom Team tatsächlich angewendet wird. Wird das Vorgehen von den Mitgliedern nicht als Modell wahrgenommen, ist das ein Zeichen dafür, dass es perfekt die Bedürfnisse der Projektabwicklung befriedigt [vgl. ebd.].

Eckstein streicht in ihren Bemerkungen abschließend hervor, dass in der Praxis lediglich das tätsächliche Vorgehensmodell eine Rolle spielt, da alle anderen Modelle lediglich in der Theorie existieren [vgl. ebd.].

4.8.3 Vorgehensmodelle für Softwaregroßprojekte

Gerade im öffentlichen bzw. militärischen Bereich spielen Vorgehensmodelle eine zentrale Rolle. Als Beispiel soll an dieser Stelle daher kurz das V-Modell XT – dem Vorgehensmodell für IT-Systeme des Bundes der Bundesrepublik Deutschland – erwähnt werden.

Hierbei handelt es sich um ein sehr umfangreiches Vorgehensmodell, dass ursprünglich für die Bundeswehr, später auch für sämtliche Bundesbehörden erstellt wurde [HiHö09, p.19]. Ohne hier konkret auf das Wesen dieses Vorgehensmodell einzugehen, sollen nur die Vor- bzw. Nachteile erwähnt werden [vgl. ebd., p.21f.]:

<u>Vorteile des V-Modells XT</u>

- Die Unterstützung von parallelen Aktivitäten und nicht nur von sequentiellen Phasen.
- Die Vorgabe von definierten Aktivitäten, Produkten, Methoden und Rollen sowie deren Zuordnung zueinander.
- Zurverfügungstellung von modernen Projektdurchführungsstrategien.
- Das V-Modell bietet ein generisches Vorgehensmodell mit definierten Möglichkeiten zum Tailoring des Prozesses.
- Es ermöglicht eine standardisierte Abwicklung von Projekten zur Systemerstellung.

Nachteile des V-Modells XT

- Hohe Komplexität des Modells, damit verbunden hohe Kosten für die Einführung des Modells.
- Bei kleineren und mittleren Software-Entwicklungsprojekten führt das V-Modell zu einer unnötigen Bürokratie bezüglich Dokumentation und Vorgehensweisen.
- Ohne geeignete CASE-Unterstützung ist das V-Modell nur schwer handhabbar

4.9 Die Notbremse

Zum Abschluss des Theorieteiles noch ein paar Worte zur Benutzung der Notbremse in Software-Großprojekten – quasi die brutalste aller denkbaren Entscheidungen.

Krempl [04, p.220] schreibt, dass der deutsche staatliche IT-Markt jährlich geschätzte 11 bis 13 Milliarden Euro schwer ist, und dass gerade Konzerne von diesem Kuchen mitnaschen wollen. Diese Tatsache führt mitunter zu dem Effekt, dass die Projektleiter der durchführenden Parteien oftmals nicht das Rückgrat haben, bei Problemen rechtzeitig die Wahrheit zu sagen. Stattdessen werden dem Auftraggeber immer weitere Versprechen gemacht, um im Boot zu bleiben und die öffentlichen Millionen zu kassieren.

Statt die Notbremse zu ziehen, vermuten die Auftraggeber das Ziel immer wieder *„hinter der nächste Biegung“ [ebd., p.220]*, und investieren munter weiter.

> *„Man hat eigentlich schon alles verspielt, macht aber noch einmal weiter, um doch noch den großen Gewinn einzufahren. Doch meist endet es mit der Bankrotterklärung.“ [ebd., p220]*

4.10 Zusammenfassung

In dem Kapitel „Entscheidungsprozesse“ ging es einerseits darum, die Wichtigkeit von funktionierenden Entscheidungsprozessen grundsätzlich aufzuzeigen, und andererseits Methoden vorzustellen, die funktionierende Entscheidungsprozesse überhaupt erst ermöglichen.

Der Mensch als Quelle jedweder Entscheidung wurde insofern betrachtet, als dass die beteiligten Akteure im Entscheidungsprozess beschrieben wurden, ebenso wie die Problematik, funktionierende Teams speziell in großen Softwareprojekten zu formen.

Eine der wichtigsten Komponenten in der Teamarbeit – die Kommunikation – stand als nächstes im Mittelpunkt. Ohne Kommunikation kann kein Projekt abgewickelt werden, daher wurde ausführlich Grundsätzliches zur Kommunikation, Mittel der Kommunikation, und selbstverständlich die besonders anspruchsvolle Kommunikation in großen Teams beschrieben.

Der nachfolgende Abschnitt war dem Projektleiter gewidmet, da er hauptverantwortlich für Entscheidungen ist, und seine Person daher einer genaueren Analyse bedarf. Diese Analyse umfasste das Spannungsfeld, in dem sich der Projektleiter bewegt, die grundsätzlichen Anforderungen an ihn, die Wichtigkeit seines Urteilsvermögens, sowie die interkulturelle Kompetenz – gerade in Großprojekten ein wesentlicher Punkt.

Abgerundet wurde das Thema Projektleiter mit einer Einschätzung des Einflusses der Unternehmenskultur auf sein Handeln, sowie einer kurzen Vorstellung der wichtigsten Führungsstile bzw. deren Einfluss auf den Entscheidungsstil.

Als wichtiges Mittel zur Entscheidungsfindung wurde der kreative Problemlösungsprozess genauer vorgestellt, bzw. wurden Besprechungen im Detail betrachtet, da sie häufig den Rahmen darstellen, in dem Entscheidungen letztendlich getroffen werden.

Die Entscheidungsfindung selbst wurde in weiterer Folge von diversen Blickwinkeln aus betrachtet, angefangen von Entscheidungsstilen, über die Akzeptanz von getroffenen Entscheidungen durch das Team, bis hin zu der Frage, wie die Nachhaltigkeit von Entscheidungen beurteilt werden kann. Abgerundet wurde das Thema mit einem Verweis auf das sogenannte Paradoxon der Übereinstimmung.

Ein eigener Abschnitt behandelte das Widerpsruchsmanagement, da nicht immer davon auszugehen ist, dass sämtliche Akteure mit den getroffenen Entscheidungen einverstanden sein werden, und daher das Aufarbeiten von Widersprüchen enorm wichtig ist.

Zum Abschluss des Kapitels wurden schließlich noch Vorgehensmodelle allgemein betrachtet und vorgestellt, wobei selbstverständlich der Scope des Software-Großprojektes Berücksichtigung fand.

Mit diesem Kapitel endet der Theorieteil dieser Arbeit, und es folgt der Praxisteil, in dem ausgewählte Aspekte der Theorie auf ihre Praxisrelevanz hin überprüft werden.

5 Die Befragung

„Fragen über Fragen ...“

– Unbekannt

Dieses Kapitel stellt den Praxisteil der Diplomarbeit dar, der die in den vorangegangenen Kapiteln gewonnen Erkenntnisse kontrastiert. Befragt wurden fünf Person, die über Erfahrung im Projektmanagement im Umfeld großer Softwareprojekte aufweisen können.

Zunächst werden die fünf Interviewpartner kurz vorgestellt, danach wird auf die Fragen eingegangen, die gestellt wurden.

Die anschließende Auswertung der Befragung wird Aufschluss darüber geben, in wieweit sich die Erkenntnisse aus dem Theorieteil tatsächlich in der gelebten Praxis widerspiegeln.

5.1 Vorstellung der Interviewpartner

Zunächst werden die interviewten Personen vorgestellt, um ein Bild zu vermitteln aus welchen Positionen heraus die Aussagen getroffen wurden.

Mag. Predrag Jovanovic ist seit acht Jahren nach anfänglichen Tätigkeiten im Developing als Consultant im SAP Bereich tätig. Seit vier Jahren ist er mit großen Projekten im öffentlichen Bereich beschäftigt. Als PMA-zertifizierter Projektmanager ist Predrag Jovanovic bei der Bundesrechenzentrum GmbH als Projektleiter und Senior Consultant tätig.

Mag. Christoph Mayerhofer ist seit 2001 im Projektmanagement tätig. Seine bisher eingenommenen Rollen waren die des technischen Projektleiters, Kundenprojektmanagers, sowie in größeren Projekten die des Projektmanagers in Form von Gesamtprojektmanagement. Christoph Mayerhofer ist als Projektleiter für eCommerce und Online Solutions bei der Telekom Austria AG beschäftigt.

Mag. Robert Thumfarth arbeitete nach seinem Studium als Teilprojektleiter bei der Wirtschaftskammer Österreich im SAP Umfeld. Später konnte er beim ORF immer wieder in Großprojekten wirken, beispielsweise bei der Ausschreibung für ein Broadcast-Archiv, dass letztendlich mit einer Beteiligung der ARD umgesetzt wurde. Robert Thumfarth ist bei der Flughafen Wien AG als Leiter ERP Applikationen in der Abteilung IE Anwendungsentwicklung beschäftigt, und ist dort für Projekte verantwortlich, wie beispielsweise der Einführung einer neuen Cargo-Software, die in ein Großsystem eingebettet ist und die bestehende Mainframe-Lösung ablösen wird.

Dipl. Ing. Georg Voigt konnte zunächst bei einer großen Website als Entwickler bzw. Projektmanager fungieren. In weiterer Folge war er als Entwickler an der Umsetzung eines Online-Roleplaying Games beteiligt, das über eine Entwicklungszeit von drei Jahren entwickelt wurde. Im Umfeld der Sozialversicherung war er am damals größten Projekt Österreichs (eCard) als Systemanalytiker bzw. technischer Projektleiter beteiligt. Georg Voigt ist bei der Sozialversicherungs-Chipkarten GmbH als Teamleiter für Analyse und Architektur, bzw. als stellvertretender Bereichsleiter Technik beschäftigt.

Mag. (FH) Stefan Marton begann seine berufliche Laufbahn als freiberuflicher Entwickler im Versicherungsumfeld. Nach mehrjähriger freiberuflicher Tätigeit als Entwickler und technischer Consultant, auch im benachbarten Ausland, wechselte er in ein Anstellungsverhältnis und ist seitdem im Bereich Projekt- und Servicemanagement für die ANECON GmbH, einem mittelständischen Entwicklungs- und Beratungsunternhemen, tätig. Zur Zeit leitet er bei einem Kunden aus der Telekommunikationsbranche die Bereiche Entwicklung und Betrieb für das Ordermanagement System für Geschäftskunden.

5.2 Der Interviewleitfaden

Aus dem Theorieteil wurden zehn Themenkomplexe ausgewählt, zu denen die Interviewpartner befragt wurden und die sich wie folgt darstellen:

1. Beschreibung der signifikanten Besonderheiten bzw. Anforderungen eines Groß-Projektes, verglichen mit einem Softwareprojekt normaler Größe.
2. Beschreibung der Akteure im Entscheidungsprozess, welche Entscheidungen im Allgemeinen zu treffen sind, und wie die Herangehensweisen in der Entscheidungsfindung sind.
3. Beschreibung der zentralen Eigenschaften, die ein Projektleiter mitbringen muss.
4. Beschreibung der in Großprojekten angetroffenen Entscheidunsstile, bzw. welche Maßnahmen ergriffen werden, damit die getroffenen Entscheidungen von den Teammitgliedern akzeptiert werden.
5. Beschreibung, welche Erfahrungen mit Meetings gemacht wurden, bzw. in welchem Rahmen und in welcher Form Problemlösungsprozesse kennengelernt wurden.
6. Beschreibung, was unter den Begriffen Vorgehens- bzw. Prozessmodell verstanden wird, bzw. welche konkreten Erfahrungen mit Vorgehensmodellen gemacht wurden.
7. Beschreibung der typischen oder üblichen Probleme großer Softwareprojekte.
8. Beschreibung, welche Erfahrungen mit gescheiterten Projekten gemacht wurden, was die Gründe für das Scheitern waren, und wie generell mit der Situation des Scheiterns umgegangen wurde.
9. Beschreibung der wichtigsten Aspekte bzgl. Design, bzw. die Wichtigkeit von Design generell.
10. Beschreibung, welche Erfahrungen zum Thema Architektur & Modeerscheinungen gemacht wurden.

Die Interviewpartner wurden in entspannter Atmosphäre mit diesen Themenkomplexen konfrontiert. Im folgenden Kapitel erfolgt die Auswertung der Antworten auf die gestellten Fragen.

5.3 Auswertung der Interviews

Auf Wunsch der Interviewpartner, bzw. um nicht in den Konflikt mit Datenschutzbestimmungen oder Geheimhaltungsrichtlinien der einzelnen Unternehmen zu geraten, wurden die Aussagen der Befragten anonymisiert.

Die Anonymisierung wurde realisiert, in dem die Themenkomplexe „horizontal“ ausgewertet wurden. Damit ist gemeint, dass die Antworten nicht getrennt nach Interviewpartner, sondern getrennt nach Themenkomplex betrachtet wurden, wie folgende Skizze verdeutlichen soll:

	Person 1	**Person 2**	**Person 3**
Thema 1	**...**	**...**	**...**
Thema 2	**...**	**...**	**...**
Thema 3	**...**	**...**	**...**

Aus diesem Grund werden in der Behandlung der einzelnen Themenkomplexe keine Namen genannt, sodass nicht zurückverfolgt werden kann, welche Aussage von welchem Interviewpartner stammt. Das ist auch der Grund dafür, warum in der Beschreibung der Themenkomplexe keine Namen der Befragten aufscheinen.

Nachfolgend wird beschrieben, welche Antworten zu den einzelnen Themenkomplexen gegeben wurden.

Themenkomplex 1 – Beschreibung der signifikanten Besonderheiten bzw. Anforderungen eines Groß-Projektes, verglichen mit einem Softwareprojekt normaler Größe.

Am häufigsten wurden die hohe Komplexität, und der damit verbunde Mehraufwand der Projektabwicklung genannt, der mit einem höheren Overhead in der Projektorganisation begegnet wird. In diesem Zusammenhang wurde auch erwähnt, dass ab einem bestimmten Projektvolumen mit einer gestiegenen

Aufmerksamkeit seitens höherer Managementebenen gerechnet werden kann, da im Regelfall bei großen Projekten signifikant mehr Geld im Spiel ist. Dies führt mitunter zu einer quantitativen Überdimensionierung der Entscheidungsträger, was sich beispielsweise in mehrstufigen Lenkungsausschüssen manifestiert.

Auf Grund der hohen Komplexität wurde der erhöhte Bedarf einer entsprechenden Strukturierung bzw. besseren Planung des Projektes erwähnt, was zu einer stärkeren Forderung des Projektmanagements führt.

Als Besonderheit von Großprojekten wurde insbesonders angegeben, dass es üblich ist das Projekt in Teilprojekte zu unterteilen. In solchen Fällen wird für jedes dieser Teilprojekte ein eigener Sub-Projektleiter benötigt.

Es wurde auch die Problematik erwähnt, dass die an dem Projekt beteiligten Personenkreise anwachsen, was dazu führen kann, dass nicht ganz klar ist welche Akteure Anforderungen definieren bzw. überhaupt definieren dürfen – das führt mitunter zu Missverständnissen.

Die Herausforderung, alles zu Überblicken, wurde in dem Zusammenhang genannt, dass für die Abwicklung großer Projekte viele verschiedene Skills benötigt werden. Neben den Management-Fähigkeiten, die der Projektleiter mitbringen muss, wurden auch das technische Wissen genannt, das innerhalb des Projektteams vorhanden sein muss, um ein entsprechendes Projekt abwickeln zu können. Dieses weit gefächerte Wissen ist unter Umständen nicht in einem einzigen Unternehmen anzutreffen, was dazu führt, dass mehrere, auf bestimmte Gebiete spezialisierte Unternehmen an der Projektabwicklung beteiligt sind.

Als besondere Herausforderung großer Softwareprojekte wurde die Kommunikation genannt. Hierbei wurde einerseits die Kommunikation innerhalb des Projektteams erwähnt, als auch der Abstimmungsaufwand mit dem Kunden. Der Kommunikationsaufwand innerhalb des Projektteams wird um so aufwändiger, je mehr Personen beteiligt sind, bzw. je höher der Parallelisierungsgrad des Projektes ist.

Insgesamt wurde die soziale Komplexität eines Software-Großprojektes – bezogen auf das Projektteam – als herausfordernder eingeschätzt als die technischen Ansprüche, die Systeme größerer Dimensionen mit sich bringen. Eine

funktionierende Kommunikation innerhalb der Projektmitglieder aufzubauen, wird daher als eine der anspruchsvollsten Künste innerhalb des Projektmanagements angesehen.

Themenkomplex 2 – Beschreibung der Akteure im Entscheidungsprozess, welche Entscheidungen im Allgemeinen zu treffen sind, und wie die Herangehensweisen in der Entscheidungsfindung sind.

Die Frage nach den Akteuren wurde recht unterschiedlich beantwortet. So wurde beispielsweise erwähnt, dass in manchen Projekten der Kunde den Projektleiter stellt, während das durchführende Unternehmen selbst einen Projektmanager nominiert. Während der Projektleiter technisch verantwortlich ist, leistet der Projektmanager hauptsächlich koordienierende Aufgaben. In einer solchen Konstellation werden Entscheidungen hauptsächlich zwischen Projektleiter und Projektmanager getroffen.

Bei internen Projekten wurden die unterschiedlichen Fachabteilungen als jene Akteure genannt, die Anforderungen stellen und daher in diesbezügliche Entscheidungsfindungen involviert sind. Dies führt mitunter auf Grund unterschiedlicher Sichtweisen auf die Thematik zu Konflikten zwischen den an dem Projekt beteiligten Fachabteilungen.

Inhaltlich erschien das Spektrum, weswegen Entscheidungen überhaupt getroffen werden, recht breit. Gerade die Anforderungsphase eines Projektes wurde als heiße Phase bzgl Entscheidungen genannt. In diesem Zusammenhang wurde beispielsweise die Architektur des Systems, das verfügbare Budget, der Zeitpunkt der Markteinführung, und ähnliche Themen genannt. Besonders hervorgehoben wurden gesetzliche Vorgaben – beispielsweise Datenschutzbestimmungen – die gerade bei Großprojekten in der Entscheidungsfindung eine signifikante Rolle spielen.

Dadurch, dass gerade Großprojekte relativ streng hierarchisch organisiert sind, wurde gesagt, dass oft nicht unbedingt technische Entscheidungen ein Problem darstellen, sondern dass gerade zu Beginn eines Projektes organisatorische Entscheidungen im Mittelpunkt stehen. Der Projektleiter ist hier gefordert

Sachverhalte klarzustellen, einen gemeinsamen Nenner zu finden, und zwischen den unterschiedlichen Parteien zu vermitteln.

Große Projekte werden üblicherweise zunächst in kleinere Teilprojekte zerlegt, da das Projekt als Ganzes sonst nicht managebar wäre. Die Teilprojektleiter sind für ihre Sub-Projekte verantwortlich und können innerhalb eines definierten Rahmens Entscheidungen selbst treffen. Bei Entscheidungen, die die Kompetenzen der Teilprojektleiter überschreiten würden, wird die Entscheidungsfindung an das jeweils übergeordnete Gremium eskaliert, bis hin zum Projektlenkungsausschuss (PLA).

In dem Zusammenhang mit Teilprojekten wurde erwähnt, dass gerade in einer solchen Situation, in der mehrere Sub-Projekte aufeinander abgestimmt werden müssen, es von größter Wichtigkeit ist, dass die gesetzten Meilensteine strikt eingehalten werden, um Verzögerungen zu vermeiden. Die Termintreue wurde auch in Verbindung mit gesetzlichen Vorgaben erwähnt, da beispielsweise zu einem bestimmten Zeitpunkt ein Gesetz in Kraft tritt, und daher das System ab diesem Zeitpunkt den Anforderungen entsprechen muss.

Gerade in Großprojekten wurden auf Grund der involvierten Personen Machtspiele erwähnt, die zu Problemen führen können, wenn unterschiedliche Interessen verfolgt werden. Die Entscheidungen bezüglich der Methodik bzw. der eingesetzten Technologie geraten daher oft in den Hintergrund, da zunächst politische Kämpfe ausgefochten werden. Es wurde auch darüber gesprochen, dass nicht selten Projekte für Monate stillstehen, weil auf organisatorischer Ebene keine Einigung erzielt werden kann.

Themenkomplex 3 – Beschreibung der zentralen Eigenschaften, die ein Projektleiter mitbringen muss.

Die genannten Eigenschaften eines Projektleiters (PL) werden der Übersicht wegen in Form einer Aufzählung widergegeben.

- Der PL muss entscheidungsfreudig sein.
- Der PL muss rechtzeitig erkennen, wenn er seine Kompetenz überschreitet.

- Der PL darf kein Einzelplayer sein, er muss sich auf sein Team verlassen können.
- Der PL muss das Gesamtsystem zumindest grob fachlich überblicken können.
- Der PL muss ein gutes Gespür für Diplomatie haben, er kann nicht immer „mit dem eisernen Schwert durchfahren“.
- Der PL muss sich mit den Problemen und Bedürfnissen der Stakeholder auseinandersetzen.
- Der PL muss strukturiert bzw. methodisch Vorgehen können.
- Der PL muss über die Teilprojekte immer gut informiert sein.
- Der PL muss über technisches Fachwissen verfügen, da ihm sonst „alles erzählt wird“.
- Der PL muss Menschen gut einschätzen können, um abschätzen zu können ob die ihm gelieferten Informationen plausibel sind.
- Der PL muss mit den Stakeholdern gut vernetzt sein.
- Der PL muss sich Autorität verschaffen können.
- Der PL muss über Softskills bzgl. Führungsqualitäten verfügen.
- Der PL muss mit einschlägigen Projektmanagement-Tools umgehen können.
- Der PL muss über soziale Fähigkeiten verfügen, da er üblicherweise mit vielen unterschiedlichen Menschen zusammenarbeitet.
- Der PL muss ein Gespür für Finanzen haben, da es bei Großprojekten um viel Geld geht.
- Der PL muss Begeisterungsfähigkeit unter seinen Projektmitgliedern verbreiten können.
- Der PL muss in der Lage sein, Arbeiten delegieren zu können.
- Der PL muss in der Lage sein, Entscheidungen an Leute abgeben zu können.

Themenkomplex 4 – Beschreibung der in Großprojekten angetroffenen Entscheidunsstile, bzw. welche Maßnahmen ergriffen werden, damit die getroffenen Entscheidungen von den Teammitgliedern akzeptiert werden.

Kleinere Entscheidungen werden oftmals auf recht unbürokratische Weise in Meetings getroffen. Für aufwändigere Entscheidungen werden Methoden wie beispielsweise die SWOT Analyse eingesetzt, wo mit Hilfe eines Gewichtungskriterienkatalogs ein Szenario als das vermeintlich Beste auserkoren wird.

Es wurde berichtet, dass Systementscheidungen in der Regel von Projektmitarbeitern über einen bestimmten Zeitraum (cirka vier bis sechs Monate) evaluiert werden, anschließend wird dem Auftraggeber eine Entscheidungsmatrix vorgelegt, aus der der Auftraggeber auf Grund der Kosten- und Techniksituation eine Variante auswählt, die letztendlich umgesetzt wird.

Die Art und Weise, wie Entscheidungen getroffen werden, hängt sehr stark vom Typ des Projektleiters ab – abhängig davon ob es sich um einen charismatischen, autoritären, oder sonstigen Führungsstil handelt. Es wurde jedoch auch berichtet, dass es situationsbedingt zu unterschiedlichen Entscheidungsstilen kommen kann. Ein Projektleiter, der ansonsten immer eine basisdemokratische Linie verfolgt, kann – beispielsweise unter Druck stehend – auch zu autoritären Führungsmethoden greifen.

Grundsätzlich wurde berichtet, dass autoritär gefällte Entscheidungen zu Problemen führen, weil kein Grunddiskurs im Team vorhanden ist. Auf der anderen Seite sah man schon demokratisch geführte Projekte im Chaos versinken, weil jeder das Gefühl hatte Chef zu sein.

Generell wurde erwähnt, dass es von Nachteil ist, wenn der Projektleiter Entscheidungen trifft, die nicht mit dem Team im Vorfeld abgestimmt wurden, da sonst keine Akzeptanz in der Gruppe für die getroffene Entscheidung vorhanden ist und sich dies negativ auf die Stimmung im Projekt auswirkt.

Abzuraten ist von Entscheidungen, die „aus dem Bauch heraus“ getroffen werden, da Entscheidungen immer fachlich und/oder technisch fundiert sein sollen. Ein

guter Projektleiter informiert sich vor Entscheidungen ausführlich bei seinem Team, und trifft dann die entsprechende Entscheidung.

Konsens herrschte darüber, dass jede Person einzeln behandelt werden muss, und dass die Mitsprache der Teammitglieder ermöglicht werden soll, um die Motivation hochzuhalten. Technisch sehr versierte Projektmitarbeiter, denen oft ein Autoritätsproblem nachgesagt wird, müssen unbedingt „mit ins Boot“ geholt werden, da ansonsten diese Teammitglieder ein Motivationsproblem bekommen und auf diese Weise die Qualität des Projekts negativ beeinträchtigt wird.

Bei Entscheidungen, die an Projektmitglieder delegiert wurden, und die sich als Endlosschleifen entpuppen, weil auf Grund unterschiedlicher Meinungen keine gemeinsame Linie gefunden werden kann, ist es wichtig, dass vom Projektleiter eingegriffen und eine Entscheidung getroffen wird – auch dann, wenn nicht alle Teammitglieder mit der eingeschlagenen Richtung einverstanden sind.

Ähnlich wurde von Entscheidungen berichtet, die ggf. von Vorständen ohne für die Teams ersichtliche Grundlagen getroffen werden. Diese Entscheidungen müssen dann von den Projektleitern entsprechend in das Team getragen werden, und das wurde als mitunter schwierig beschrieben, weil sich in solchen Fällen die Teammitglieder nicht mit den getroffenen Entscheidungen identifizieren können.

Schließlich wurde erwähnt, dass Entscheidungen, die von übergeordneten Stellen getroffen werden, aus zweierlei Sicht betrachtet werden können:

Einerseits hat dies den Vorteil, dass der Projektleiter nicht viel Zeit dafür aufwänden muss, um den Teammitgliedern die getroffene Entscheidung schmackhaft zu machen. Denn alle Beteiligten wissen sowieso, dass es nicht möglich ist die Entscheidung zumindest zu hinterfragen, oder sogar zu revidieren – die Entscheidung ist so zu akzeptieren.

Andererseits ist dieses Vorgehen sehr motivationshinderlich, was langfristig dazu führt, dass die Teammitglieder beginnen, ihr „Hirn auszuschalten“. Es ist klar, dass dadurch im Extremfall das Projekt zum Scheitern gebracht werden kann.

Themenkomplex 5 – Beschreibung, welche Erfahrungen mit Meetings gemacht wurden, bzw. in welchem Rahmen und in welcher Form Problemlösungsprozesse kennengelernt wurden.

Grundkonsens ist, dass Meetings nur dann Sinn machen, wenn Sie ein konkretes Ziel haben und moderiert werden, bzw. wenn mittels Protokoll festgehalten wird, was besprochen wurde. Es darf nicht sein, das „irgendetwas“ bei einem Meeting herauskommt, niemand protokolliert, und drei Wochen später weiß keiner mehr darüber Bescheid. Solche Szenarien führen dazu, dass die immergleichen Meetings zu den immergleichen Themen abgehalten werden – quasi eine Endlosschleife.

Generell sollten die teilnehmenden Personen vorbereitet in das Meeting gehen. Gerade bei Reviews welcher Art auch immer müssen sich die Leute fachlich vorher mit dem Thema auseinandersetzen – in der Praxis sieht es jedoch so aus, dass meistens in gelangweilte Gesichter geblickt wird, die zu allem „ja und Amen“ sagen und meinen, dass „eh alles super ist“. Oft stellt sich dann später heraus, dass doch nicht alles so super ist und genau die Personen, die während des Reviews sehr gelangweilt waren, plötzlich Einwände haben.

Es herrschte die Meinung vor, dass Meetings eher zur Kommunikation dienen als zur Problemlösungsentscheidung, weil Entscheidungen in bilateralen Gesprächen oder direkt im Projektlenkungsausschuss getroffen werden, und in Meetings nur noch formal abgesegnet werden.

In Meetings werden daher selten Probleme gelöst, sondern eher Probleme analysiert und unterschiedliche Sichtweisen gehört, um daraus eine Problemlösung abzuleiten. Meetings stellen daher eine gute Gelegenheit dar, um viele Personen an einen Tisch zu bekommen und deren unterschiedliche Meinungen zu einem Thema zu hören.

Der Beginn eines Problemlösungsprozesses mittels Brainstorming wird zwecks Ideenfindung eingesetzt, aber es wurde als sehr wichtig erachtet, dass es einen konkreten Output gibt, der in Folgeterminen aufgegriffen und weiterverarbeitet wird. Der eigentliche Problemlösungsprozess kann sich erst nach dem Meeting

entwickeln, bzw. wird noch ein extra Meeting benötigt, um die unterschiedlichen Problemlösungsmethoden durchzudiskutieren und entsprechend zu entscheiden.

Wenn in einem Meeting tatsächlich Entscheidungen getroffen werden, ist darauf zu achten, dies in kleinem Rahmen zu tun, da es in der Regel leichter ist zu Entscheidungen zu gelangen, wenn weniger Personen an der Entscheidungsfindung beteiligt sind. Auch die Kommunikation bereits getroffener Entscheidungen ist einfacher, wenn Sie zunächst nur den „wichtigsten“ Personen einer Projektorganisation mitgeteilt werden.

Obwohl Meetings oft länger dauern als sie dauern müssten, werden sie immer noch als die beste Quelle genannt, um an Informationen zu gelangen – im Gegensatz zu dem Austausch von Dokumenten, e-Mail, etc. Als besonders wichtig wurde in diesem Zusammenhang erwähnt, dass auf jeden Fall ein Protokoll über das Meeting anzufertigen ist. Auf diese Weise hat der Moderator die Möglichkeit, besprochene Punkte aus eigener Sicht darzustellen, und um Feedback auf das Protokoll durch die Teilnehmer innerhalb einer bestimmten Frist zu bitten. In der Regel gibt es jedoch kein Feedback.

Ganz schlecht wurde die Situation beurteilt, wenn Meetings ausarten und die „Meeting-Mania“ um sich greift, also wegen jeder Kleinigkeit eine Besprechung einberufen wird, da ab einem gewissen Punkt Meetings nicht mehr ernst genommen werden. Im Gegensatz dazu sollten Jour-Fixes, also regelmäßige Zusammenkünfte zwecks Statusmitteilungen, im Projekthandbuch festgelegt werden.

Schließlich wurden rückläufige Meetings als Indikator genannt, dass es im Projekt gut läuft.

Themenkomplex 6 – Beschreibung, was unter den Begriffen Vorgehens- bzw. Prozessmodell verstanden wird, bzw. welche konkreten Erfahrungen mit Vorgehensmodellen gemacht wurden.

Vorgehensmodelle für den Entwicklungsprozess wurden beispielsweise so definiert, dass – angefangen von Agil bis Wasserfall – es darum geht, wie Software entsteht, welche Zyklen durchlaufen werden, welche Feedbackschleifen es gibt, bzw. wie die entsprechende Dokumentation auszusehen hat. Oder, wie es

in einem Fall sehr simpel ausgedrückt wurde, wie die Entwicklung angegangen wird.

Am häufigsten wurde als statisches Modell das Wasserfall-Modell genannt, gefolgt von der agilen Methode Scrum. Es wurde darauf hingewiesen, dass bei fixen Budgets bzw. fixen Anforderungen schwer agile Methoden einsetzbar sind – in solchen Fällen muss dann üblicherweise auf das bewährte Wasserfall-Modell zurückgegriffen.

Es wurde erwähnt, dass bestehende Modelle in der Regel an den jeweiligen Kunden angepasst werden, abhängig davon wie viele Updates pro Jahr geliefert werden, ob es eine kontinuierliche Weiterentwicklung gibt, etc. Von diesen Faktoren hängt ab, ob sich ein Vorgehensmodell gut oder gar nicht eignet.

Das Prozessmodell wurde in einem Fall als Business Process zu einem Produkt beschrieben, der sich im technischen Prozess widerspiegelt. Diese Definition lässt wohl einigen Spielraum offen.

Im Zuge der Befragung wurden Projektmanagementmodelle genannt, bei denen es darum geht, wie Arbeitspakete aufgeteilt werden können, bzw. wie ein Projekt – allgemein betrachtet – durchgeführt werden kann. Es ist darauf zu achten, nicht streng nach Lehrbuch vorzugehen, sondern der jeweiligen Situation entsprechend Modifikationen an den Vorgehensweisen vorzunehmen.

Für den Betrieb eines Software-Produktes wurden als Modelle das Incident-Management bzw. ITIL genannt, wobei vorrangig erwähnt wurde, dass für den Fehlerfall entsprechende Arbeitsanweisungen vorhanden sein müssen, die im Vorfeld auch zu trainieren sind, um die Verfügbarkeit des Systems zu gewährleisten.

Jene Personen, die im SAP Umfeld anzutreffen sind, sprachen von einem strikten SAP Vorgehensmodell, das sich stark von den klassischen Modellen unterscheidet, die beispielsweise bei Eigenentwicklungen mittels Java eingesetzt werden. Dieses SAP Vorgehensmodell umfasst kurz und prägnant ausgedrückt Sollkonzept, Feinkonzept und schließlich Dokumentation. Am Ende der Anforderungsphase steht der sogenannte „Blue Print“. Der Blue Print beschreibt Prozesse sowie technische Wege für die Umsetzung, und ist eine Mischung aus

verbalen Anforderungen und grobem Konstrukt wie ein Projekt technisch umgesetzt werden soll. Wird der Blue Print vom Kunden abgesegnet, beginnt die Implementierungsphase.

Blue Prints werden durch Customizing-Leitfäden unterstützt, die im SAP Bereich standardisierte Vorgehensweisen für unterschiedliche SAP Module darstellen. Die Erfahrung mit Blue Prints bzw. Customizing-Leitfäden ist, dass es den Kunden leichter fällt mit immer wieder gleichen Modellen zu arbeiten.

Was die Erfahrung mit Modellen betrifft, wurde davon berichtet, dass Vorgehensmodelle zwar eingeführt, jedoch oft nicht wirklich gelebt wurden. Es handelte sich dabei oft um totes Papier, dass von kaum jemanden beachtet wurde.

In einem Fall wurde davon berichtet, versuchsweise in Zukunft Scrum als agile Methode einzusetzen, da in der Vergangenheit oft Probleme mit der Wasserfall-Methode auftraten. Der Grund dafür war, dass sich Kunden selten auf Anforderungen festlegen konnten, und daher das Projekt oftmals in einer Sackgasse endete. Mittels agiler Vorgehensweisen erhofft man sich, frühzeitig Probleme aufdecken zu können, jedoch ist zum momentanen Zeitpunkt noch schwierig, agile Methoden den höheren Managementebenen schmackhaft zu machen.

Agile Methoden betreffend, wurde erwähnt, dass oft der Irrglaube vorherrscht, nichts dokumentieren zu müssen. Entwickler, die häufig nicht so gerne dokumentieren, waren daher enttäuscht zu erfahren, dass sehr wohl auch bei agilen Vorgehensweisen dokumentiert werden muss. Weiters wurde davon berichtet, dass sich manche Kunden nicht dabei wohlfühlen, dass am Ende einer Anforderungsphase kein fix fertiges Pflichtenheft vorliegt. Je größer das Projekt, desto unwahrscheinlicher wurde der Erfolg eingeschätzt, mittels agiler Methoden entwickeln zu können.

Generell wurde zu Prozessmodellen gesagt, dass viele Unternehmen erst ihren Weg finden müssen, dass Personen mit unterschiedlicher unternehmerischer Herkunft oft mit bestimmten Vorgehensweisen „vorbelastet“ sind, und dass daher manchmal Konflikte vorprogrammiert sind, wenn Vorgehensmodelle wie Religionen gehandhabt werden, die Mitarbeiter sich jedoch einem Modell

unterordnen müssen – und zwar jenem, dass sie aus welchen Gründen auch immer ablehnen.

Letztendlich – darin waren sich alle Befragten einig – kommt es darauf an abzubilden, was der Kunde wünscht.

Themenkomplex 7 – Beschreibung der typischen oder üblichen Probleme großer Softwareprojekte.

Am Häufigsten wurde folgendes Problem genannt: Wenn sehr viele Parteien involviert sind, ist es schwierig alle standespolitischen Einflüsse unter einen Hut zu bringen, die sich zwangsläufig aus einer Vielzahl von Stakeholdern ergeben. Ist nur ein (1) Kunde involviert – was bei Großprojekten fast nie der Fall ist – ist dieses Problem auf ein Minimum reduziert.

In diesem Zusammenhang wurde auch Macht und Geld angesprochen. Da große Projekte einen gewaltigen Kapitaleinsatz mit sich bringen, stehen sie verstärkt im Fokus von Controllern und Vorständen. Dies kann zwar ein Vorteil sein, weil die Aufmerksamkeit in den höheren Managementebenen steigt und dadurch ein engagiertertes Vorgehen erwartet werden kann. Meistens jedoch führt diese Situation dazu, dass Machtkämpfe in das Projekt hineingetragen werden, die im Extremfall ein Projekt zum Scheitern bringen können, wenn Kämpfe auf Kosten des Projekts ausgefochten werden. Daher wurden im Zusammenhang mit großen Projekten Machtspiele, und nicht Prozesse oder Technik als zu nehmende Hürde genannt.

Weiters wurde sehr häufig erwähnt, dass eine hohe Fluktuation im Projekteam ein großes Problem darstellt, da die Einarbeitungszeit neuer Mitarbeiter in der Regel bei komplexen Systemen sehr hoch ist. Aus diesem Grund wurde immer wieder die Wichtigkeit eines kontinuierlichen Teams unterstrichen.

Als problematisch wurde genannt, dass bei der Verteilung des Projektteams auf mehrere Standorte die Kommunikationswege nicht mehr so gut funktionieren, als wenn das Team auf einen Standort konzentriert ist. Klare Kommunikationsstrukturen wurden generell als hoher Erfolgsfaktor in großen Softwareprojekten genannt. So wurde beispielsweise auch erwähnt, dass es immer

von Vorteil ist, wenn der Kunde zusammen mit dem Projekteam an ein und demselben Standort stationiert ist.

Es wurde ebenso von der Erfahrung berichtet, dass gerade bei großen Projekten, die manchmal auch über eine Zeitspanne von mehreren Jahren entwickelt werden, die Gefahr besteht, dass zwischendurch (mehrere) Technologiewechsel durchgeführt werden, oder dass sogar bereits abgeschlossene Arbeitspakete nochmals überarbeitet werden müssen, weil sich gewisse Umstände aus irgendwelchen Gründen geändert haben. Es wurde beobachtet, dass Projekte, deren Projektleiter sich „verrennen“, Gefahr laufen zu scheitern oder zumindest sich zu verzögern.

Im Zusammenhang mit der langen Projektlaufzeit wurde auch erwähnt, dass sich unter Umständen veränderte Anforderungen auf Grund von gesetzlichen Änderungen ergeben, die trotzdem noch berücksichtigt werden müssen, obwohl das Projekt kurz vor der Fertigstellung steht. Eine solche Situation strapaziert das Budget und verursacht im Regelfall einen zeitlichen Verzug.

In einem Interview wurde davon berichtet, dass während der Ablöse eines alten Systems durch ein neues, eine Firmenfusion stattgefunden hat. Auf Grund dieser Fusion und der Laufzeit der Ablöse – insgesamt ein Jahr – haben sich die Rahmenbedingungen so stark verändert, dass das Projekt in der geplanten Form nicht mehr durchgezogen werden konnte. Das Projekt wurde gestoppt und in dem veränderten Umfeld neu aufgerollt – die bisher aufgelaufenen Kosten waren als Sunken Costs zu betrachten.

Als weiteren Faktor hat sich herauskristallisiert, dass die hohe Komplexität großer Softwareprojekte zu einem Problem werden kann. Sobald die einzelnen Teilprojekte in ihrem Zusammenhang nicht mehr von einer Person (in Form des Projektleiters) erfasst werden können, kann davon ausgegangen werden, dass das System einen zu hohen Komplexitätsgrad aufweist und somit die Gefahr besteht, von niemanden mehr richtig verstanden zu werden.

In der Planung erweisen sich Projekte hoher Komplexität speziell dann als schwierig, wenn es um die Aufteilung des Gesamtsystems in Teilprojekte geht. Hier sind Projektleiter gefragt, die über die entsprechende Erfahrung verfügen, um

möglichst realistische Aussagen bzgl. der Ressourcen der einzelnen Arbeitspakete treffen zu können.

Als problematisch bei großen Softwareprojekten wurden auch Vergabeverfahren genannt. Gerade öffentliche Stellen sind ab einem bestimmten Volumen verpflichtet, neue Systeme zunächst national, und ab eines noch höheren Volumens sogar international auszuschreiben.

Das Problem an Vergabeverfahren ist, dass sie sehr langwierig sein können – insbesonders dann, wenn entsprechende Einspruchs- bzw. Stillhaltefristen eingehalten werden müssen. Gerade von IT-Projekten wird jedoch häufig erwartet, dass sie schnell umgesetzt werden können. Aus dieser Konstellation können sich in der Folge Konflikte ergeben, wenn die handelnden Akteure für die Problematik der Ausschreibungen nicht genügend sensibilisiert sind.

Themenkomplex 8 – Beschreibung, welche Erfahrungen mit gescheiterten Projekten gemacht wurden, was die Gründe für das Scheitern waren, und wie generell mit der Situation des Scheiterns umgegangen wurde.

Als Hauptgrund für das Scheitern von Projekten stellte sich im Zuge der Befragung heraus, dass die Ziele bzw. Anforderungen im Vorfeld des Projekts nicht eindeutig geklärt wurden.

Weiters wurde angeführt, dass Entscheidungsträger Aufgaben nicht delegieren konnten, weil sie nicht akzeptierten, wie Tasks durch das Team gelöst wurden – die Ergebnisse des Teams waren den Verantwortlichen schlicht und einfach nicht gut genug, was dazu führte, dass auf Grund des Nicht-Delegierens die Entscheidungsträger mit der Arbeitslast nicht mehr zurecht kamen.

Ebenfalls langfristig negativ hat sich der Umstand ausgewirkt, dass schon fertig gestellte Arbeitspakete immer wieder weggeschmissen und neu gemacht wurden. Dadurch wurde einerseits das Budget überbeansprucht, und andererseits litt die Motivation der Teammitglieder auf Grund dieser Vorgehensweise enorm, da man sich ständig zurück an den Start begab und so nie fertig wurde. Es wurde berichtet, dass „irgendwann der Geldhahn zugedreht“, und das Projekt niemals zu Ende gebracht wurde. Eine große Frustration der Teammitglieder war die Folge.

Weiters wurde von Projekten berichtet, deren Scheitern darin begründet lag, dass die Größenordnung schlicht unterschätzt wurde. Es gab keine klare Planung, wie die Arbeit einzuteilen ist, was dazu führte, dass viele Doppelgleisigkeiten entstanden, die untereinander inkompatibel waren.

Zwar nicht gescheitert im üblichen Sinn, aber trotzdem nicht produktiv gegangen sind beispielsweise Projekte, die auf Grund einer geänderten Gesetzeslage den Anforderungen nicht mehr entsprachen, und daher de facto trotzdem als gescheitert zu betrachten sind, da sämtliche eingesetzte Ressourcen in einem großen schwarzen Loch verschwanden. In solchen Fällen ist man um Schadensbegrenzung bemüht, indem der Lernfaktor mit beispielsweise neuen Technologien etc. hervorgehoben wird.

In einem Fall wurde davon berichtet, dass nach cirka 95 %iger Fertigstellung die Ablehnung der zukünftigen Benutzer so groß war, dass das Projekt tatsächlich eingestampft wurde. Dieser Sonderfall wurde jedoch als Ausnahme bezeichnet, da die Anwender in der konkreten Situation so viel Einfluss hatten, sich dem Produkt erfolgreich zu widersetzen.

Als Folge eines gescheiterten Projekts wurde festgestellt, dass eine Kultur der Schuldzuweisung entstand. Das führte dazu, dass im Folgeprojekt sehr viel Wert auf Protokollierung usw. gelegt wurde, weil niemand mehr niemandem traute, und auf diese Weise die Kommuniktion total gestört war. Jeder wollte für sich einen Persilschein, um nicht zur Verantwortung gezogen zu werden, falls auch dieses Projekt scheitern würde. Diese stark gestiegene Bürokratisierung hatte jedoch zur Folge, dass die Abwicklung der Folgeprojekte immer schlechter funktionierte, inkl. wieder gescheiterter Projekte.

Langfristig hat eine solche Situation zur Folge, dass die „guten“ Leute das Unternehmen verlassen und die „schlechten“ bleiben – was zu einer qualitativen Erosion der Projektorganisation führt.

Projekte werden generell dann als gescheitert betrachtet, wenn der angestrebte Fertigstellungstermin, das veranschlagte Budget, und / oder der vereinbarte Funktionsumfang nicht erreicht werden. Es wurde davon berichtet, dass in

bestimmten Fällen der Launch-Termin verschoben wurde, um das offizielle Scheitern eines Projektes zu verhindern bzw. zu verschleiern.

Schließlich wurde erwähnt, dass in politisch brisanten Projekten die Vorgehensweise gewählt wird, erreichte Teilprojekteziele als Erfolg zu verkaufen wenn das Gesamtprojekt zu scheitern droht, um zu bekräftigen, dass mit dem bisher aufgewendeten Budget doch etwas geschaffen wurde auf das man aufbauen kann, und daher das bisher eingesetzte Geld nicht komplett vernichtet wurde.

Themenkomplex 9 – Beschreibung der wichtigsten Aspekte bzgl. Design, bzw. die Wichtigkeit von Design generell.

Grundtenor der Befragten war, dass je besser die Anforderungen im Vorfeld des Projekts definiert sind, umso sauberer die Entwicklung abgewickelt werden kann. Je komplexer das zu implementierende System ist, desto stärker trifft diese Weisheit zu. Aus diesem Grund wird grundsätzlich empfohlen, dass viel Zeit sämtlicher Entscheidungsträger dafür investiert werden sollte, um ein gemeinsames Verständnis aller Beteiligten herzustellen.

Bei den Überlegungen bzgl. Design wurden neben den obligatorischen Requirements auch solche Faktoren genannt wie beispielsweise die geplante Laufzeit des fertigen Produktes, bzw. welche Technologie eingesetzt wird, um ggf. einen langfristigen Betrieb gewährleisten zu können. Die Skalierung spielt ebenfalls eine gewichtige Rolle, besonders dann, wenn zu erwarten ist, dass das System über die Laufzeit gerechnet mit steigenden Mengengerüsten konfrontiert wird.

Grundsätzlich wird versucht, immer mehr Open Source Produkte einzusetzen, weil der Betrieb bzw. die Wartung viel günstiger sind verglichen mit kommerzieller Software. Dies trifft besonders auf Betriebssysteme bzw. Web- und Applikationsserver zu. Ein weiterer Grund ist, unabhängiger gegenüber etwaigen Hardware-Herstellern zu sein. Die Ausnahme bildet lediglich das Datenbankmanagementsystem – hier werden nach wie vor kommerzielle Produkte eingesetzt.

In einem Fall wurde von einem Projekt erzählt, bei dem versucht wurde 30 Jahre alte Strukturen mit modernen Mitteln nachzubauen. Der Grund dafür war, dass es

zu wenige Gespräche mit dem Kunden gab, und daher zu wenig Zeit dafür aufgebracht wurde, eine von Grund auf neue Architektur zu entwerfen. Das Ergebnis war, dass das fertige Produkt vom Kunden nicht abgenommen wurde, weil die Performance nicht den Erwartungen entsprochen hat.

Im SAP Umfeld nehmen Design und Architektur keine Schlüsselpositionen ein, da das Basissystem fix vorgegeben ist. Hier stehen maximal Entscheidungsfragen im Customizing-Bereich an, bzw. geht es um die Auswahl diverser Tools von Drittanbietern wie bspw. Adobe Forms etc.

Themenkomplex 10 – Beschreibung, welche Erfahrungen zum Thema Architektur & Modeerscheinungen gemacht wurden.

In Verbindung mit Modeerscheinungen wurden beispielsweise Frameworks aus dem Java Umfeld genannt. Hier wurde es als schwierig bewertet, „auf das richtige Pferd zu setzen“, oder rechtzeitig von einem Framework auf das nächste umzusteigen. Diesbezüglich wird es als wichtig erachtet, regelmäßig an einschlägigen Konferenzen teilzunehmen bzw. den Kontakt zur entsprechenden Community zu suchen.

Nicht immer wurde auf die zukunftsträchtigste Technologie gesetzt – dies wirkt sich negativ auf die Wartungsfähigkeit etc. aus, speziell dann, wenn aus welchem Grund auch immer ein Framework gehoben werden muss, und dann nicht mehr kompatibel zu einem anderen Framework ist, das sich ebenfalls im Einsatz befindet und für das keine Upgrades mehr angeboten werden.

Gerade im Umgang mit Mainstream-Entwicklungssprachen wird empfohlen, regelmäßig Fachmagazine zu studieren und diese Magazine wenn möglich auch der gesamten Entwicklungsmannschaft zur Verfügung zu stellen, damit die Teammitglieder up to date bleiben und immer die neuesten Entwicklungen verfolgen können.

Es wurde von Projektoragnisationen berichtet, die in die sogenannte „Wartungsfalle“ gerieten, weil jahrelang jegliche neue Technologie abgelehnt wurde. Dies hatte langfristig zur Folge, dass man im Extremfall ehemalige Projektmitglieder aus der Pension zurückholen musste, weil das Know-How im

Team nicht mehr vorhanden war um Wartungs- oder Erweiterungsarbeiten durchzuführen.

Manche Projektorganisationen leisten sich kleine Teams, die mit nichts anderem beschäftigt sind, als neue Technologien zu evaluieren und Berichte darüber zu erstellen, die von anderer Stelle später als Entscheidungsgrundlage für Architektur-Entscheidungen herangezogen werden können.

Es wurde erwähnt, dass in gewissen Ausschreibungen bereits festgelegt wird, welche Technologien eingesetzt werden müssen, bzw. was auf keinen Fall eingesetzt werden darf. In solchen Fällen bleibt kaum Handlungsspielraum, da akzeptiert werden muss was in der Ausschreibung gefordert wird.

Grundsätzlich fiehl immer wieder der Begriff „Chefarchitekt“. Damit war gemeint, dass eine Person, die idealerweise mit dem entsprechenden Wissen und der Erfahrung ausgestattet ist, die Linie vorgibt, wie die Architektur für ein neu zu entwerfendes System auszusehen hat. Die restlichen Teammitglieder werden zwar in die Entscheidungsfindung mit eingebunden, müssen sich letztendlich aber der Entscheidung des Chefarchitekten anschließen. Ansonsten besteht die Gefahr, dass sich jeder Entwickler selbst verwirklichen möchte, und die Architektur des Systems einem „Fleckerlteppich“ gleicht.

Als Stichwort wurde häufig das Schichtenmodell genannt. Offensichtlich hat sich dieses Modell durchgesetzt, wenn es um die Trennung einzelner Komponenten geht. Als Argument wurden Unabhängigkeit zwischen den Systemen, sowie Überlegungen im Hinblick auf Security etc. genannt.

Manche der Befragten berichteten, dass mitunter aus dem Management Vorgaben für spezielle Technologien gemacht werden. Bei genauerer Betrachtung bzw. Nachfrage stellte sich fallweise heraus, dass die Marketingaktivitäten großer Konzerne ganze Arbeit geleistet hatten, und dem Management selbst nicht ganz klar war, welche Vorteile die vermeintlich neue Technologie bringen sollte.

5.4 Zusammenfassung

Es zeigte sich, dass es in gewissen Bereichen eindeutige Übereinstimmungen zwischen den Befragten gab, während andere Aspekte unterschiedlich bewertet und eingeschätzt wurden.

Überraschend waren vor allem die Aussagen zum Thema Requirements, weswegen an dieser Stelle nochmals explizit darauf eingegangen wird. Dafür, dass das Requirements Engineering durchgehend als sehr wichtig bzw. sogar mitunter als überlebenswichtig für den Erfolg eines Projektes angesehen wurde, kam es in den unterschiedlichen Themenkomplexen so gut wie gar nicht zur Sprache.

Diese Ambivalenz macht deutlich, was immer wieder in der Praxis beobachtbar ist: Grundsätzlich wissen alle Beteiligten um die Wichtigkeit von Requirements Engineering und Design Bescheid, aber irgendwie – überspitzt ausgedrückt – fühlt sich niemand explizit dafür zuständig, diese Belange in einer Art und Weise zu forcieren, wie es eigentlich notwendig wäre.

Buxton greift exakt diesen Umstand auf, in dem er schreibt:

> *„So let's get to the point by asking a few questions: Who makes design decisions in your company? Do they know that they are making design decisions? If you are the CEO of the company: Is design leadership an executive level position? Do you have a Chief Design Officer reporting to the president? If the answer to the last two questions is no, remember that your actions speak loader than your words." [Buxt07, p.17]*

Große Softwareprojekte stellen natürlich ganz spezielle Anforderungen an die Vorgehensweise der Projektverantwortlichen, und diese Diplomarbeit soll dazu beitragen, ein wenig Licht in das oft vernachlässigte Gebiet der Entscheidungsfindung zu bringen.

Nicht nur, dass die sogenannten Entscheidungsträger nicht nur über entsprechendes technisches Fachwissen verfügen müssen, sondern auch ausgeprägte Kompetenzen in den viel beschriebenen Soft Skills gefragt sind. Doch selbst wenn diese Voraussetzungen erfüllt sind, kann keine noch so ausgeprägte Begabung bzw. noch so tiefes Fachwissen die Erfahrung ersetzen, die man sich nur über Jahre der Arbeit in großen Softwareprojekten aneignen kann.

Natürlich sind ideal konstruierte Entscheidungsträger nur die halbe Miete. Die besten Entscheidungen können nur dann umgesetzt werden, wenn die gesamte Entwicklungsmannschaft, die hinter dem Projekt steht, die getroffenen Entscheidungen akzeptiert und ohne größeren Widerstand umsetzt.

Um die Teammitglieder mit ins Boot zu holen, ist neben einer gut strukturierten Projektorganisation – Stichwort räumliche und kulturelle Belange – eine Integrationsfigur in Person des Projektleiters gefragt, der die Mitarbeiter Motivieren und auch Inspirieren kann.

6 Conclusio

„Software is big business“
– Tarek Abdel-Hamid

Am Ende der Diplomarbeit angelangt, ist es nun an der Zeit zurückzublicken und Resümee zu ziehen.

Stellt man die Aussagen aus dem Praxisteil den aus dem Theorieteil gewonnenen Erkenntnissen gegenüber, ergeben sich zwei Schlussfolgerungen. Es gibt

- Aussagen, die sich mit den theoretischen Überlegungen decken, und
- theoretische Erkenntnisse, die von den Interviewpartnern nicht erwähnt wurden.

Im Großen und Ganzen wurden die theoretischen Erkenntnisse durch die Befragungen bestätigt, jedoch gab es den einen oder anderen Aspekt, der von den Interviewpartnern nicht thematisiert wurde.

Zu diesen Aspekten zählten beispielsweise die gesellschaftlichen Einflüsse großer Software-Systeme (Stichwort „Speicherung sensibler Daten“), sowie die Wichtigkeit ergonomischer Benutzer-Schnittstellen.

Die Forschungsfrage wurde aus theoretischer Sicht beantwortet, und durch die Erkenntnisse aus den Interviews hinreichend untermauert.

Die folgenden abschließenden Gedanken stellen den Schlusspunkt der Arbeit dar.

Auf Grund der Omnipräsenz von Computern – oder verwandten Geräten wie beispielsweise Smartphones – sind wir tagtäglich in vielen Situationen unseres Lebens mit Softwareprodukten konfrontiert, die ihren Ursprung in Software-Projekten hatten.

Viele dieser Produkte werden von uns mit einer Selbstverständlichkeit benutzt, sodass uns oft gar nicht bewusst ist, dass sich im Hintergrund mächtige Computersysteme verbergen – beispielsweise wenn wir am Bankomaten Geld beheben, oder mit Hilfe einer Suchmaschine im World Wide Web nach einer Information suchen.

Und genau in diesen Situationen, in denen wir keinen Gedanken an die Arbeitsweise dieser Systeme verschwenden, wird deutlich, dass die Menschen, die an der Entstehung dieser Produkte beteiligt waren, oft die richtigen Entscheidungen getroffen haben, damit wir uns nicht mit schlecht durchdachten Produkten herumschlagen müssen.

Denn die Qualität des Produktes ist direkt davon abhängig, welche Entscheidungen im Zuge des Entwicklungsprozesses getroffen wurden. Wer hat sich nicht schon über ergonomisch fragwürdige Software geärgert, und sich in dem Moment des Ärgerns gedacht: „Was haben sich die Leute eigentlich dabei überlegt?“.

Was im kleinen Rahmen vielleicht maximal ein Ärgernis ist, kann im großen Rahmen zu einer Katastrophe führen, wenn man beispielsweise an die Steuerungssysteme eines Atomkraftwerkes denkt. Aus diesen – und noch vielen anderen – Gründen ist es daher von höchster Wichtigkeit, bei den Entscheidungen, die tagtäglich in der Softwareentwicklung großer Systeme anstehen, mit Bedacht vorzugehen.

Auch wenn es nicht gleich um Menschenleben gehen muss – als Projektverantwortlicher ist es notwendig, sich dafür einzusetzen das bestmögliche Produkt im Hinblick auf Datensicherheit, Datenschutz, Funktionstüchtigkeit, Verlässlichkeit, etc. zu entwerfen. Hierbei geht es nicht nur um den wirtschaftlichen Erfolg des umsetzenden bzw. betreibenden Unternehmens, sondern – und insbesonders – auch um die Verpflichtung gegenüber einer möglicherweise breiten Benutzerbasis.

Literaturverzeichnis

[AbMa91] Abdel-Hamid, Madnick: **Software Projekt Dynamics – An Integrated Approach**
Prentice-Hall 1991

[BeAc04] Bergsmann, Achtert: **Die Bedeutung der Anforderungsspezifikation bei der Beschaffung und Ausschreibung im Bereich e-Government**
Vortrag gehalten im Rahmen der e-Gov-Days in Wien (15. - 16. März 2004) 2004

[Benn95] Bennatan, E. M.: **Software Projekt Management – A Practitioner's Approach**
McGraw-Hill International 1995

[Berk09] Berkun, Scott: **Die Kunst des IT-Projektmanagements**
O'Reilly 2009

[BlBö05] Blanckenburg, Böhm, Dienel, Legewie: **Leitfaden für interdisziplinäre Forschergruppen: Projekte initiieren – Zusammenarbeit gestalten**
Franz Steiner Verlag 2005

[Bron00] Bronzite, Michael: **System Development – A Strategic Framework**
Springer 2000

[Broo95] Brooks, Frederick P. jun.: **The Mythical Man-Month**
Addison-Wesley 1995

[BrMa04] Brown, Malveau, McCormick, Mowbray: **Anti Patterns**
mitp 2004

[Buxt07] Buxton, Bill: **Sketching User Experiences**
Morgan Kaufmann 2007

[Cock02] Cockburn, Alistair: **Agile Software Development**
Addison-Wesley 2002

[Conw68] Conway, Melvin E.: **How do committees invent?**
Datamation, 14, 4 (April 1968)

[Coop04] Cooper, Alan: **The inmates are running the asylum**
Sams 2004

[CoHa05] Coplien, James O., Harrison, Neil B.: **Organizational Patterns of Agile Software Development**
Pearson 2005

[DeLi91] DeMarco, Lister: **Wien wartet auf Dich! Der Faktor Mensch im DV-Management**
Hanser 1991

[DeLi03] DeMarco, Lister: **Waltzing with Bears**
Dorset House 2003

[Denk05] Denker, Ahmet: **The Challenge of Large-Scale IT Projects**
in: World Academy of Science, Engineering and Technology
9 2005

[Ecks04] Eckstein, Jutta: **Agile Software Entwicklung im Großen**
dpunkt.verlag 2004

[Flyv05] Flyvbjerg, Bent: **Megaprojects and risk: an anatomy of ambition**
Cambridge University Press 2005

[GoBa01] Gora, Bauer (Hrsg.): **Virtuelle Organisationen im Zeitalter von E-Business und E-Government**
Grimmer, Klaus: **Polititsche Rahmenbedingungen für die Verwaltungsmodernisierung mit IT**
Springer 2001

[HaNa92] Haberfellner, Nagel, Becker, Büchel, von Massow: **Systems Engineering – Methodik und Praxis**
Verlag Industrielle Organisation Zürich 1992

[Hell08] Hellige, Hans Dieter: **Software-Manufaktur – Software-Architektur – Software-Engineering Leitbildwandel der Software-Konstruktion in den 50er / 60er Jahren**
2008

[HiHö09] Hindel, Hörmann, Müller, Schmied: **Basiswissen Software-Projektmanagement**
dpunkt.verlag 2009

[JaZe07] Jaburek, Zehentner: **IT-Ausschreibungen nach dem BVergG 2006**
Verlag Medien und Recht 2007

[John94] Johnson, Jim: **The CHAOS Report**
The Standish Group 1994

[Karn02] Karnovsky, Hans: **Grundlagen des Projektmanagements**
Paul Bernecker Verlag 2002

[Kerb00] Kerber-Kunow, Annette: **Projektmanagement und Coaching**
Hüthig GmbH 2000

[Kies04] Kiesel, Manfred: **Internationales Projektmanagement**
Fortis im Bildungsverlag Eins GmbH 2004

[Krem04] Krempl, Stefan: **Das Casino-Prinzip – Warum so viele IT-Großprojekte scheitern**
in c't 23/2004, Heise 2004

[LiRe01] Lientz, Bennet; Rea, Kathryn: **Breakthrough Technology Project Management**
Academic Press 2001

[Maci05] Maciaszek, Leszek A.: **Requirements Analysis and System Design**
Pearson 2001

[Mayr05] Mayr, Herwig: **Projekt Engineering – Ingenieurmäßige Softwareentwicklung in Projektgruppen**
Carl Hanser Verlag 2005

[Mill71] Mills, H.: **Chief programmer teams, principles, and procedures**
IBM Federal Systems Division Report FSC 71 1971

[Möll09] Mölle Daniel: **Desgin by Demut**
in iX 9/2009, Heise 2009

[NaRa68] Naur, P., Randell, B.: **Software Engineering**
Proc. Conf. sponsored by NATO Science Committee 1968

[Noe09] Noe, Manfred: **Der effektive Projektmanager**
Publicic Publishing 2009

[Ratt07] Rattay, Günter: **Führung von Projektorganisationen**
Linde Verlag Wien 2007

[Rain03] Rainwater, Hank: **Katzen hüten**
mitp 2003

[Scha05] Schabert, Karl: **Requirements Analysis realisieren**
Vieweg & Sohn 2005

[Sche02] Scheuring, Heinz: **Der www-Schlüssel zum Projektmanagement**
Orell Füssli Verlag AG 2002

[Seib98] Seibert, Siegfried: **Technisches Management**
B. G. Teubner Stuttgart 1998

[Setz09] Setzwein, Christian: **Die Kurve kriegen – IT-Projekte vor dem Scheitern retten**
in iX kompakt 3/2009, Heise 2009

[Seng94] Senge, Peter: **The Fifth Discipline – The Art & Practice of The Learning Organization**
Currency Doubleday 1994

[SiPa03] Simmons, Parmee, Coward: **35 years on: to what extent has software engineering design achieved its goals?**
in: IEE Proc.-Softw., Vol 150, No. 6 2003

[WeOr92] Weltz, Ortmann: **Das Softwareprojekt**
Campus 1992

[Your99] Yourdon, Edward: **Death March**
Prentice Hall 1999

[ZuBi01] Zuser, Biffl, Grechenig, Köhle: **Software Engineering**
Pearson 2001

Online-Ressourcen

[URL01]
http://en.wikipedia.org/wiki/Semi_Automatic_Ground_Environment
abgerufen am 23.05.2010

[URL02]
http://en.wikipedia.org/wiki/Sabre_%28computer_system%29
abgerufen am 23.05.2010

[URL03]
http://c2.com/cgi/wiki?ConwaysLaw
abgerufen am 23.05.2010

[URL04]
http://www.interaction-design.org/encyclopedia/interaction_design.html
abgerufen am 23.05.2010

Abbildungs- und Tabellenverzeichnis