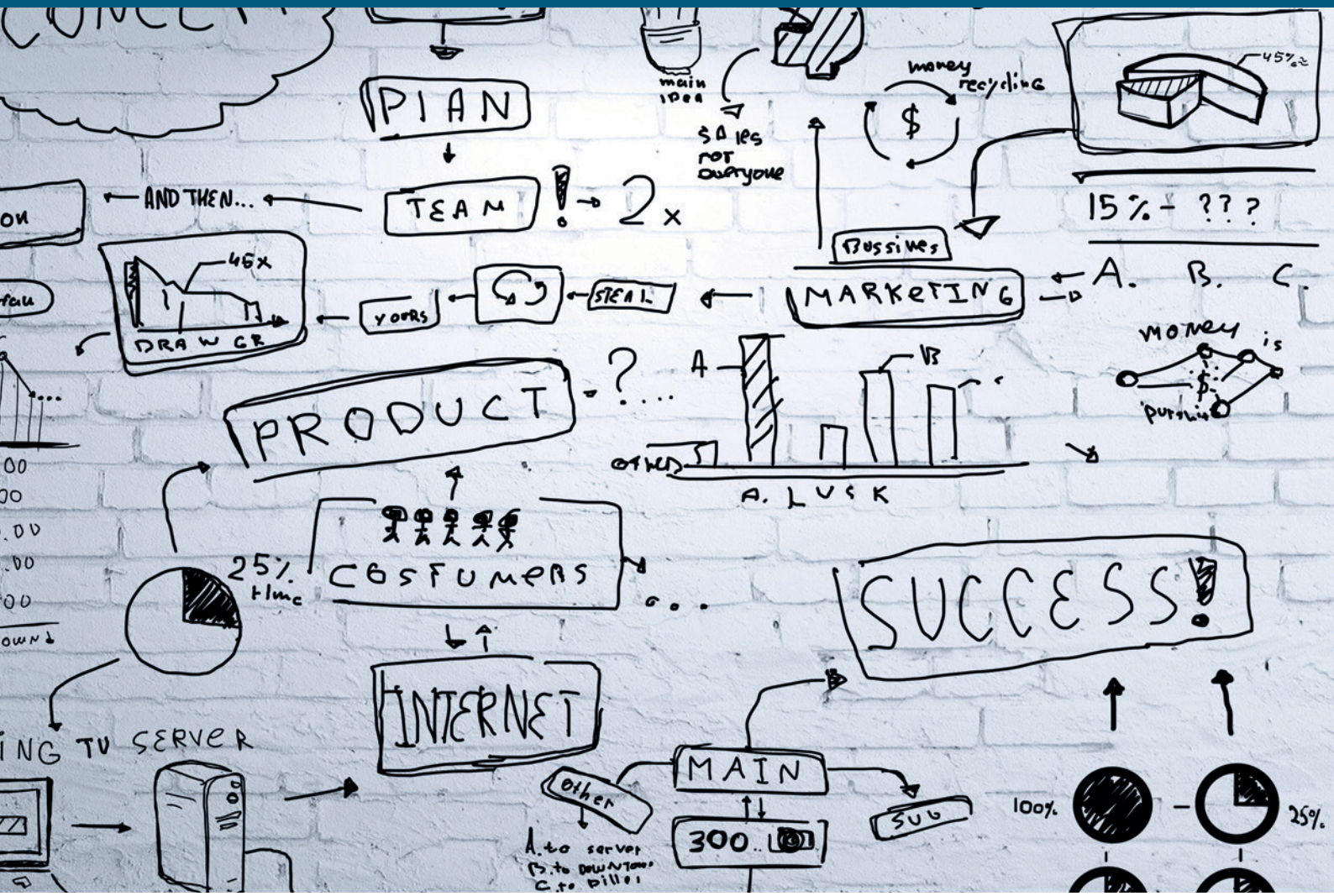
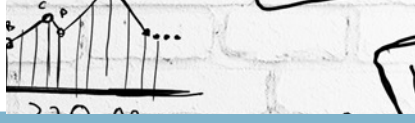


Agiles Software Engineering Made in Germany

Leitfaden

BITKOM



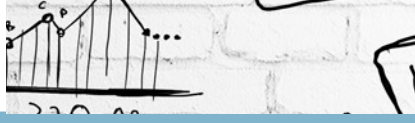
Impressum

Herausgeber:	BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. Albrechtstraße 10 A 10117 Berlin-Mitte Tel.: 030.27576-0 Fax: 030.27576-400 bitkom@bitkom.org www.bitkom.org
Ansprechpartner:	Manuel Fischer Tel.: 030.27576-233 m.fischer@bitkom.org
Copyright:	BITKOM 2013
Redaktion:	Manuel Fischer (BITKOM)
Grafik/Layout:	Design Bureau kokliko / Astrid Scheibe (BITKOM)
Titelbild:	© peshkova – Fotolia.com

Agiles Software Engineering Made in Germany

Leitfaden

BITKOM



Inhaltsverzeichnis

1 Motivation und Übersicht

Der Siegeszug der IT in allen Lebensbereichen hält an: Jeder ist überall und jederzeit erreichbar und kann von überall seine E-Mails prüfen. Immer mehr Geräte können via IT angesprochen, angesteuert und abgefragt werden. Aus immer mehr kleineren IT-Applikationen und -Diensten werden durch geschickte Interoperabilität und Vernetzung immer größere Systeme und die Verortung einzelner Systembausteine spielt scheinbar eine immer unwichtigere Rolle. Die Entwicklung großer IT-Systeme verändert sich entsprechend rasant und stellt insbesondere die deutsche IT-Branche vor eine Vielzahl neuer Herausforderungen. Diese moderne Softwareentwicklung verändert sich aktuell entlang unterschiedlicher Dimensionen:

- Organisatorisch wird immer häufiger das Konzept des »Global Delivery« gefordert, d.h. die sinnvolle Einbeziehung kostengünstiger Ressourcen für dedizierte Aufgabenpakete.

- Ökonomisch sieht sich jedes IT-Projekt unter einem fortwährenden Kosten- und Rechtfertigungsdruck; jeder einzelne Kostenblock muss im Vorfeld bekannt und begründet werden.

- Zudem führt der stetige Kostendruck zu einer zunehmenden Zentralisierung und Service Orientierung (Stichwort Cloud) mit den entsprechenden Risiken und Problemen (Sicherheit, »One Solution for all«).

- Prozessoral muss ein IT-Projekt heute leichtgewichtig und höchst dynamisch sein. Lange Planungs- und Entwurfsphasen weichen zunehmend iterativen und inkrementellen Prozessen mit regelmäßigen Lieferungen.

- Technisch sehen sich IT-Projekte heute einer Vielzahl vorhandener, teilweise stark unterschiedlicher Techniken gegenüber; sowohl hinsichtlich der technologischen Ausgangsbasis (z. B. Programmiersprachen) als auch in Bezug auf die Zielplattformen (z. B. mobile Endgeräte).

- Vertraglich muss ein IT-Projekt heute hohe Verbindlichkeiten eingehen: Klassische Kostenabrechnungen nach Zeit und Aufwand weichen zunehmend einer Festpreis oder zumindest einer transaktionsbasierten Kostenstellung.

- Menschlich fordert ein IT-Projekt heute zunehmende Qualifikation hinsichtlich, z. B. offener Kommunikation, Bewertung neuer Trends, internationale/interkulturelle Teams, zunehmende Konkurrenz. Der Allrounder spielt im Gegensatz zu früher eine wesentlich wichtigere Rolle als der Spezialist.

Jede Dimension liefert hierzu eine Vielzahl von Schlagwörtern, die es heute zu berücksichtigen gilt: Beispiele sind agile Softwareentwicklung, Lean Organisation, Industrialisierung, Engineering und Projektmanagement. Es vergeht wohl kaum ein Monat, in dem nicht ein weiteres Schlagwort hinzukommt, das für jeden IT-Verantwortlichen in Deutschland dahingehend bewertet werden muss, ob es relevant und erfolgsversprechend oder nur ein kurzer »Hype« ist.

Dieser Leitfaden möchte ein Rahmenwerk aufzeigen, das hilft, die unterschiedlichen Bereiche, aus denen diese Schlagwörter entstammen, zu strukturieren. Er unterstützt konkrete Projekte dabei, sich aus dem Baukasten der vielen verfügbaren Konzepte, Verfahren und Techniken die am besten geeignetsten auszuwählen (ohne dabei eine Übersicht über die verfügbaren Alternativen zu geben). Insbesondere hilft dieser Leitfaden aber zu verhindern, dass die Bereiche im Unbestimmten verbleiben und so z. B. die Einführung agiler Methoden ohne eine entsprechende Qualifikation des Personals oder ohne eine passende Projektmanagement-Methode erfolgt. Ebenso hilft er, überbestimmte Selektionen zu verhindern, in dem aus einem Bereich mehrere, evtl. sogar konkurrierende Lösungen gewählt werden.

Zusätzlich zum Rahmenwerk schlägt dieser Leitfaden eine für Deutschland gewinnträchtige Vorbesetzung pro Bereich vor. Diese nutzt Deutschlands Stärken und kompensiert etwaige Nachteile. Die Vorbesetzung ist gleichzeitig der Grund für den Namen des Leitfadens: Agiles Software Engineering Made in Germany.

Grundlage der diesem Leitfaden zugrundeliegenden ganzheitlich betrachteten Softwareentwicklung ist die Berücksichtigung einer Vielzahl relevanter Einflussfaktoren. Kern dieses Leitfadens ist nicht die detaillierte Auflistung all dieser Faktoren sondern das Aufzeigen eines Rahmenwerks, das diese Faktoren strukturieren hilft. Es zeigt damit einen Raum auf, innerhalb dessen sich jedes IT-Projekt verorten können sollte. Ist eine Dimension nicht bestimmt, ist dies ein guter Indikator für ein unvollständig definiertes Projekt. Existieren für ein Projekt mehrere Punkte, ist dies hingegen ein guter Indikator für ein inkonsistent definiertes Projekt.

Das in diesem Leitfaden vorgestellte Rahmenwerk schlägt konkret die folgenden drei Ebenen mit den jeweiligen Vorbesetzungen vor, die aufeinander aufbauen (s. Abbildung 1):

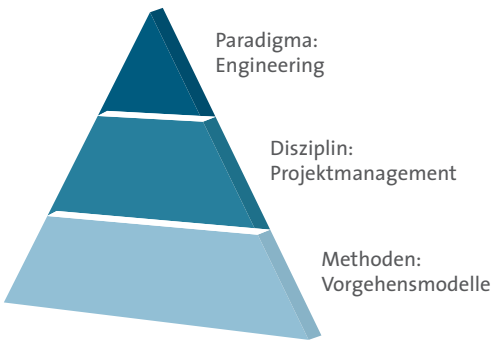


Abbildung 1: Die drei essenziellen Ebenen der Softwareentwicklung

Die einzelnen Ebenen fokussieren unterschiedliche Abstraktionsebenen der Softwareentwicklung:

- Auf der Ebene der Paradigmen wird die grundsätzliche Erwartungshaltung an ein Projekt formuliert:
  - Gibt es überhaupt Ziele?
  - Ist ein Endtermin definiert?
  - Sind die Ressourcen beschränkt?
  - Ist Transparenz während eines Projektes gewünscht?

- Auf der Ebene der Disziplin wird definiert, wie das konkret gewählte Paradigma umgesetzt werden kann:
  - Wie wird ein Projekt organisiert?
  - Wie wird es strukturiert?
  - Wie erfolgt ein etwaiges Reporting?
  - Wie erfolgt die rechtzeitige Feststellung von Problemen und wie wird damit umgegangen?

- Auf der Ebene der Methode wird definiert, wie das Projekt konkret bearbeitet wird, welche Methoden und Werkzeuge kommen zum Einsatz. Dabei ist die Methode fest in die überliegende Disziplin integriert.
  - Wie wird codiert?
  - Wie wird getestet?
  - Welche Techniken kommen zum Einsatz?
  - Woher kommt Feedback?

Jede dieser drei Ebenen wird im Folgenden in jeweils einem separaten Kapitel behandelt. Neben der generellen Beschreibung der Zielsetzung jeder Ebene wird eine auf Deutschland zugeschnittene Ausgestaltung vorgestellt: Die empfohlenen Ausgestaltungen sind dabei

- Engineering als Paradigma
- Projektmanagement als Disziplin
- Agile Vorgehensmodelle als Methode

Zum Abschluss des Dokumentes erfolgt eine Empfehlung, wie Softwareentwicklung in Deutschland im Idealfall aufgesetzt sein sollte, um die Stärken zu nutzen und die Schwächen zu kompensieren.

2 Paradigma Engineering

2.1 Definition

Unter einem Paradigma wird im Allgemeinen ein vorherrschendes Denkmuster verstanden. Paradigmen spiegeln einen gewissen allgemein anerkannten Konsens über Annahmen und Vorstellungen wider, die es ermöglichen, für eine Vielzahl von Fragestellungen Lösungen anzubieten. Eine präzisere Definition ist im Folgenden angegeben:

»Zu Paradigmen zählen sowohl methodologische Konzepte als auch intuitive Grundeinstellungen zu Phänomenen. Ein Paradigma regelt, was als untersuchenswerter Gegenstand wissenschaftlicher Betrachtung zu gelten hat, die Art und Weise, wie dieser Gegenstand zu beobachten ist und was als befriedigende Lösung eines wissenschaftlichen Problems anzusehen ist.« ([1]).

Bezogen auf das Feld der Softwareentwicklung beleuchtet ein Paradigma also die grundsätzliche Sichtweise auf IT-Projekte und beantwortet Fragen, wie z. B.:

- Müssen Projekte geplant werden?
- Müssen sie gesteuert werden?
- Muss ein Projekt Ziele haben?
- Welche Facetten eines Projektes sind überhaupt relevant?

Im weiteren Verlauf dieses Kapitels soll das Paradigma »Engineering« näher beschrieben werden, da es eine der Stärken Deutschlands darstellt. Nichtsdestotrotz werden zum Abschluss dieses Kapitels überblicksartig alternative Paradigmen vorgestellt. Sie werden allerdings im Rahmen dieses Leitfadens als nicht empfehlenswert für die professionelle Softwareentwicklung Made in Germany empfohlen.

2.2 Software Engineering

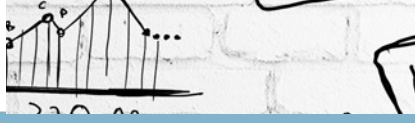
Im deutschsprachigen Raum wird Software Engineering meist mit dem Begriff Softwaretechnik gleichgesetzt. Diese Umschreibung deutet zwar weniger als das englische Pendant auf ein Handeln hin, trifft aber ansonsten ebenfalls den Sachverhalt gut.

Helmut Balzert beschreibt Software Engineering knapp und mit Fokus auf Komplexität mit den folgenden Worten:

»Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen« ([2], S. 36).

Das grundsätzliche Verständnis des Software Engineerings ist im umfassenden SWEBOK, dem Software Engineering Body of Knowledge mittlerweile etabliert (vgl. [3]). Demzufolge umfasst Software Engineering die folgenden 10 Teildisziplinen:

- Software-Anforderungen
- Software-Design
- Software-Konstruktion
- Software-Testen
- Software-Wartung
- Software-Konfigurationsmanagement
- Software-Management
- Softwareentwicklung-Prozess
- Softwareentwicklungs-Werkzeuge
- Software-Qualität



Entlang der Definition von Paradigma (s.o.) sind diese 10 Disziplinen folglich die grundsätzlich betrachteten Gegenstände wissenschaftlicher Arbeit.

Neben den Teildisziplinen des Engineerings ist in der Definition explizit die Zielorientierung genannt: Engineering ist ohne Ziele nicht denkbar, jegliches ingenieurmäßige Vorgehen benötigt Ziele, die es zu erfüllen gilt. Daher werden die Ziele im Folgenden näher beleuchtet.

2.3 Ziele

Ein heute häufig anzutreffender Katalog von Zielen präzisiert die grundlegende Anforderung, die angebotene/vereinbarte Projektleistung in der geforderten Qualität, zum vereinbarten Termin und im geplanten Budgetrahmen erbringen. Dazu gehören häufig auch die vereinbarten Methoden, Tools und Hilfsmittel sowie das Abrechnungsprozedere.

Bevor diese eher allgemeinen Ziele weiter verfeinert werden, soll dieses starr wirkende Verständnis von Zielen entlang des Engineerings mit eigenen Bordmitteln relativiert werden:

Die Zielerreichung wird in heutigen Projekten vor allen Dingen dadurch erschwert, dass sich die Ziele während der Projekte verändern: Es ist heute nicht mehr zeitgemäß, zu glauben, dass am Anfang eines Projektes alle Ziele feststehen und genau so spezifiziert werden können, dass der Kunde am Ende mit der Lösung vollumfänglich zufrieden ist. Vielmehr ist es heute so, dass zu Beginn nur eine ungefähre Vorstellung bezüglich einer Lösung vorhanden ist. Eine Konkretisierung oder Alternativen kristallisieren sich erst im Projektverlauf heraus oder verschieben sich aufgrund der globalen Dynamik und sich ändernder Rahmenbedingungen während des Projektverlaufs mitunter dramatisch. Diese Möglichkeit ist allerdings keineswegs »un-ingenieurmäßig« sondern durch die explizite Nennung des Software-Konfigurationsmanagements als Disziplin ermöglicht: Jedes Artefakt, das im Kontext eines Software-Projektes entsteht (auch Anforderungen) hat seinen eigenen Lebenszyklus, d.h. es kann verändert, erweitert oder gelöscht werden. Das häufig im Kontext agiler Softwareentwicklung (s. weiter unten) als wesentlich genannte Ziel, möglichst nah am Kunden zu agieren, frühes Feedback einzusammeln, Lösungen gemeinsam zu erarbeiten und Änderungen gegenüber offen zu sein, ist damit durch das Paradigma des Software Engineerings explizit abgedeckt und aufgrund des Charakters von Software sogar gefördert: Das Schöne und zugleich Gefährliche an Software ist ja, dass man jederzeit alles ändern kann. Beim Hausbau hingegen gestaltet es sich schwierig, am Tage des Richtfestes den Keller auszutauschen oder einen Swimming Pool auf den Schornstein zu setzen.

Engineering umfasst all diese Herausforderungen, stellt sie aber konsequent in einen systematischen Kontext, d.h. auch Änderungen an Anforderungen werden wiederum grundsätzlich zielorientiert durchgeführt.

Eine weitere Möglichkeit, Projekte nicht zu starr auffassen zu müssen ist der Lösungsansatz des Tailorings der Projekte. Statt eines großen Projektes, das ingenieurmäßig bearbeitet werden muss, wird das Grundproblem, gemäß des »Teile und Herrsche«-Ansatzes, in mehrere kleine Teilprobleme zerlegt. Ein mögliches Kriterium für diese Zerlegung ist beispielsweise eine abzusehende erhöhte Änderungsnotwendigkeit: Bestimmte Bereiche eines Software-Projektes sind heute etwa durch die Gesetzgebung, bereits beginnend in Brüssel, schon soweit reglementiert, dass die Ziele und die gewünschten bzw. verpflichtend einzuhaltenden Anforderungen durchaus beschrieben und bekannt sind (Steuerrecht, Bilanzvorschriften, Datenschutz und diverse jeweils fachliche Gesetzgebungen). Diese lassen sich in einem separaten Paket ingenieurmäßig bearbeiten. Software-Konfigurationsmanagement zum Abdecken anstehender Änderungen wird hier vermutlich nur partiell benötigt.

Im Gegensatz dazu gibt es natürlich auch Funktionalität, die bei Beginn eines Projektes nur unscharf und hypothetisch formuliert ist. Diese Projektanteile werden ein sehr viel intensiveres Software-Konfigurationsmanagement benötigen, sich dabei aber immer noch hervorragend ingenieurmäßig bearbeiten lassen.

In Folgenden werden die grundlegenden Ziele eines ingenieurmäßigen Vorgehens weiter erläutert. Die Umsetzung des Paradigmas Engineering erfordert zwangsläufig, dass diese Ziele a priori bekannt sind und das Erreichen dieser Ziele während des Projektverlaufes gesteuert wird (siehe hierzu Disziplin Projektmanagement).

2.3.1 Qualität

Software-Qualität ist einer der wesentlichen Bereiche des Software Engineerings. Qualität ist nicht nur ein abstraktes Ziel des Software Engineerings, das einmal definiert wird, sondern fortwährende Bestrebung sämtlicher Software-Engineering-Aktivitäten. Die Norm EN ISO 9000:2005 definiert Qualität als

»Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt«. (ISO-Definition in [4])

Aus dieser Definition lassen sich drei starke Einflussfaktoren auf die Qualität ableiten:

- Eine präzise Festlegung und Beschreibung von Zielen und Anforderungen ist notwendig; ohne eine solche ist eine Qualitätsbestimmung gar nicht erst möglich. Dies passt hervorragend zum Software Engineering, das genau diese Ziele und Anforderungen thematisiert und gleichzeitig die Verbindung zum Konfigurationsmanagement aufzeigt, da sich Anforderungen naturgemäß ändern können.

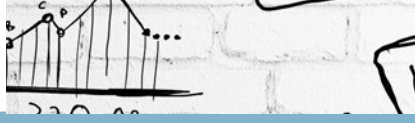
- Es bedarf geeigneter Rahmenbedingungen, die die Erfüllung der Anforderungen begünstigen. Gemeint sind beispielsweise die konstruktive und die organisatorische Qualitätssicherung (z. B. Unabhängigkeit der testenden oder das Vieraugenprinzip), aber auch alle anderen Software-Engineering-Teildisziplinen).

- Die Messbarmachung, also die Befähigung zur Messung der »inhärenten Merkmale« ist wesentlich; diese werden meist im Rahmen von Methoden der analytischen Qualitätssicherung und geeigneter Metriken realisiert. Dies ermöglicht überhaupt erst eine Bewertung, ob die Qualität als Ziel gut, schlecht oder wenigstens ausreichend erfüllt ist. Auch wenn der Begriff der Messung im Software Engineering nicht explizit gefordert wird (wohl aber im klassischen Engineering!) so führt an der Messung als Ist-Stand-Anzeiger kein Weg vorbei. Eng verbunden mit der Messung ist meist auch eine Gewichtung, da nicht alle Merkmale gleich schwergewichtig zur Qualität beitragen oder diese behindern.

Das Ziel einer möglichst hohen Qualität weist also auf eine Kerndisziplin des Software Engineerings hin: Das Anforderungsmanagement. Es ist dem Engineering zu Eigen, grundsätzliche Ziele in Form von Anforderungen zu formulieren. Darüber hinaus strebt das Engineering aber auch an, diese Ziele bestmöglich, d.h. in hoher Qualität zu erreichen. Es genügt also für ein ingenieurmäßiges Vorgehen nicht, einmalig ein Zieldokument zu erstellen, das anschließend nicht mehr benötigt wird. Zusätzlich zur Zielvorgabe ist es dem ingenieursmäßigen Vorgehen zu Eigen, auch sicherzustellen, dass alle Anforderungen dieses Zieldokumentes, ggfs. nach intensiver Änderung und Anpassung (s.o.), auch effektiv erfüllt sind. Die so erreichte hohe Qualität ist ebenfalls Ziel des Engineerings.

2.3.2 Zeit

Die »zielorientierte Bereitstellung« umfasst nicht nur die präzise Beschreibung von Anforderungen, die im Kontext von Qualität zu erfüllen sind. Da Anforderungen und Lösungen grundsätzlich nur in einem bestimmten zeitlichen Kontext Gültigkeit haben, ist die zeitliche Dimension ebenfalls zielrelevant: Es gibt keine Anforderung, deren Erfüllung zeitlich invariant ist. Im Sinne des Software Engineerings hat eine derartig motivierte Software-Erstellung immer zeitliche Vorgaben, die es zu erfüllen gilt. Werden sie nicht eingehalten, droht ein Wertverlust der überfälligen Systeme.



Die Zeit wird im Kontext des Software Engineerings von wenigstens den folgenden Faktoren beeinflusst:

- Umfang und Komplexität der Anforderungen sowie das Maß, in dem spezielle Qualitäts-Anforderungen erfüllt sein müssen. Bekannte Beispiele für Zeit-Treiber sind außergewöhnlich hohe Lastanforderungen in Bezug auf die Anzahl gleichzeitiger Benutzer, das Transaktionsvolumen usw., eine überdurchschnittlich hohe Ausfallsicherheit oder auch Anforderungen zur Barrierefreiheit. Für das Engineering ist wichtig, dass dieser Faktor nicht linear die Zeit beeinflusst: Ein System der Größe und Komplexität 2x benötigt demzufolge nicht nur doppelt so viel Zeit wie das System 1x. Je größer ein System desto höher ist beispielsweise der Aufwand für Kommunikation und Management und desto komplexer werden gruppendynamische Effekte, etwa bei steigender Teamgröße oder Teamverteilung.

- Anforderungen und Ziele sollen möglichst sofort und so früh wie möglich schlüssig und fehlerfrei beschrieben werden, da Zeit und Aufwand zur Fehlerbehebung mit fortschreitendem Entwicklungsprozess größer werden. Diesem Konzept der »frühen Fehlererkennung« liegt die Erkenntnis zugrunde, dass jedes menschliche Arbeitsergebnis potenziell Fehler enthält und diese in jedes darauf basierende Arbeitsergebnis Folgefehler induzieren können. Je später ein Fehler in dieser Kette identifiziert wird, desto mehr Zeit wird für das Gesamtentwicklungsprojekt benötigt und desto mehr Aufwände fließen in zusätzliche Aufwände für das Bugfixing und begleitende Änderungen, z. B. an der Architektur oder an der Dokumentation. Es ist daher speziell für Softwaresysteme wichtig, dass Entwickler ihre Komponenten besser gleich funktionsfähig, robust (und z. B. eben auch konform zu Codierungsrichtlinien) erstellen, als diese Fehler bei den Maßnahmen zur analytischen Qualitätssicherung ex post festzustellen und als Befund an die Entwickler zurückzugeben, die dann mit zusätzlicher Zeit die Fehler und mögliche Folgefehler anpassen müssen.

- Grad der Wiederverwendbarkeit von Ergebnissen: In früheren Projekten entwickelte und qualitätsgesicherte Komponenten oder auch Services können wiederverwendet werden, beispielsweise im Rahmen einer Serviceorientierten Architektur (SOA). Diese Facette ist gerade im IT-Bereich relevant, da trotz aller Vielfalt immer wieder gleiche oder gleichartige Komponenten erkennbar sind. Dazu gehören z. B. Datenbanken, GUI-Frameworks oder auch Persistenz-Bibliotheken: Je mehr ein neues System derartige Systeme wiederverwenden kann (sowohl in Form des aktiven Ausschau-Haltens nach wiederverwendbaren Komponenten als auch dem aktiven Bereitstellen entsprechender Software-Teile) desto weniger Zeit wird für die Entwicklung eines Systems bei wenigstens gleichzeitiger Qualität und wenigstens gleichem Budget benötigt.

- Grad der Automatisierung: Automatisierbar sind beispielsweise Teile von Entwicklungsprozessen (durch modellgetriebene Softwareentwicklung, Codegenerierung, Build-Automation, usw.), aber auch Testmethoden (durch automatisierte Codeanalysen, Unit-Tests, GUI-basierte Tests, usw.). Automatisierung stellt bei wiederkehrenden Aufgaben ein hohes Potenzial zur Zeitreduktion dar: Je mehr wiederkehrende Aufgaben automatisiert bearbeitet werden können, desto mehr Zeit kann eingespart werden.

Neben dem begünstigenden Einfluss der Wiederverwendbarkeit und Automatisierung üben nicht-automatisierbare Tätigkeiten einen eher negativen Einfluss auf die Zeit aus. Wichtige Faktoren sind dabei die Verfahrenssicherheit der Teammitglieder, die Verfügbarkeit von Wissen und Erfahrungen, die Komplexität des Produkts und die damit verbundenen Einschränkungen bei der Parallelisierung von Aufgaben. Wichtig ist für das Engineering nicht zwangsläufig eine bestimmte Gewichtung zwischen diesen beiden Extremen, sondern eine im voraus bekannte Planung: Jegliche zielorientierte Bereitstellung fordert die Definition des Ziels ex ante. Und dieses Ziel kann beispielsweise die Automatisierung völlig ausschließen; diese Entscheidung fußt dann aber auf analysierten Fakten, die dann z. B. kein Potenzial für Automatisierung aufgezeigt haben.

2.3.3 Budget

Eng verbunden mit Qualität und Zeit ist das Budget: Kaum Qualität liefern zu müssen wird i.d.R. deutlich weniger Zeit und Budget kosten, als viel Qualität liefern zu müssen. Wichtig für das Engineering ist das Verständnis, dass Qualität und der dafür notwendige Budgetrahmen nicht zum Selbstzweck erreicht werden sollte: Das Ziel eines IT-Systems umfasst meist einen monetären Rahmen, der ein bestimmtes Qualität- und Zeitverständnis mit sich bringt. Engineering optimiert den Rahmen »nur« noch, d.h. es versucht, das gesteckte Ziel auch bzgl. der monetären Ausrichtung zu erreichen und somit ein möglichst ausgewogenes Verhältnis zwischen Zeit, Qualität und Kosten zu erreichen.

Im Software Engineering dominieren meist die Personalkosten. Zusammen mit dem typischen Charakteristikum von Software, immateriell zu sein, zeigt sich das Software Engineering daher als die Vorzeigedisziplin, innerhalb derer Offshore-Ressourcen Anwendung finden: Menschen mit Hochschulstudium, teilweise auch mit entsprechend international anerkannten Zertifikaten, die zu einem Bruchteil des deutschen Lohnniveaus Software-Engineering-Tätigkeiten annehmen. Das Engineering mit seiner klaren Systematik unterstützt diese Möglichkeit, in dem die entsprechenden Rollen daraufhin untersucht werden können, ob Offshore-Ressourcen einsetzbar sind. Entgegen dem »jeder tut alles«-Ansatz, in dem Offshoring keinen Platz hat, können die einzelnen Disziplinen (z. B. das Qualitätsmanagement) daraufhin untersucht werden, ob es Aufgaben gibt, die so präzise beschrieben werden können, dass sie auch ohne einen entsprechenden Kontext auswärts erledigt werden können. Die operative Testausführung ist zum Beispiel ein solcher Bereich: Werden die durchzuführenden Tests im Vorfeld präzise mit dem Auftraggeber erarbeitet, wofür aufgrund der dafür notwendigen interaktiven Diskussion nur bedingt Offshore-Kräfte eingesetzt werden können, können die Testskripte, die dann durchzuführen sind, sehr gut durch Offshore-Mitarbeiter bearbeitet werden. Dies führt zu einer Senkung des Projektbudgets bei gleichbleibender Qualität und Zeit.

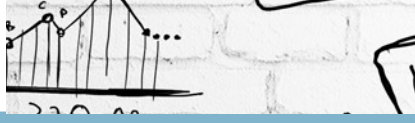
Neben den Personalkosten beanspruchen aber auch Lizenzkosten das Budget. Einen bedeutenden Einfluss auf die Lizenzkosten hat der Nutzungsgrad von Open Source Software, die für nahezu alle Bereiche verfügbar ist (Office, Wikis, Frameworks, Versionierung, Entwicklungsumgebungen, Build-Automation, Analysetools, Werkzeuge zur Testautomation, Bug-Tracking, usw.). Auch hier hilft die durch das Engineering aufgespannte Systematik Lizenzkosten strukturiert zu optimieren. Da allerdings die Hauptkosten in Softwareprojekten in einem Hochlohnland wie Deutschland nach wie vor Personalkosten sind, zahlt sich eine höhere Effektivität des Personals durch Softwarewerkzeuge mit seinen Lizenzkosten häufig jedoch deutlich aus. Optimierungsziel müssen also folglich die reduzierten Gesamtkosten sein.

2.4 Vorstudie zur Sicherstellung der Vorbedingungen

Auf der Ebene der Paradigmen wird die grundsätzliche Erwartungshaltung an ein Projekt formuliert. Doch was, wenn Ziele zwar explizit gewünscht sind, also Engineering das klar präferierte Paradigma ist, sich die Ziele aber trotzdem noch nicht klar festlegen lassen? Gibt es überhaupt (schon) Ziele? Ist ein Endtermin (bereits) definiert oder wenigstens grundsätzlich definierbar? Sind die Ressourcen beschränkt? Ist Transparenz während eines Projektes tatsächlich gewünscht?

Es zeigt sich in vielen Situationen, dass Engineering zwar gewünscht ist, aber trotzdem noch nicht alle Vorbedingungen erfüllt sind. Oft gibt es im Umfeld von Kundenprojekten viele unterschiedliche Interessenslagen, die positiv, oder behindernd wirken, oder das Projekt sogar zum Scheitern bringen können, bevor es überhaupt gestartet ist.

In der Praxis findet sich daher häufig die Situation, dass Engineering als Paradigma explizit vom Management gewünscht ist, die Vorbedingungen aber schlichtweg noch fehlen. Es ist in solchen Fällen hilfreich, in Form einer Vorstudie die wesentlichen Voraussetzungen für ein erfolgreiches Engineering erst noch zu schaffen. Es



ist müßig darüber zu diskutieren, ob diese Vorarbeiten bereits zum Engineering gehören oder nicht. Fakt ist aber, dass ohne diese Vorbedingungen klassische Engineering-Aspekte schlichtweg noch nicht direkt anwendbar sind. Vielmehr ist es in dieser Vorstudie des Engineerings relevant, unter Berücksichtigung der o.g. Einschränkungen, mit minimalsten Mitteln eine möglichst effiziente Vorarbeit zu leisten um beispielsweise bestehende Widerstände aufzulösen. Dann kann das Paradigma Engineering effektiv angewendet werden.

Eine Vorstudie besteht vor allem darin, so viel wie zwingend nötig aber auf keinen Fall mehr als nötig an Aufwand in die planerischen Elemente des Engineerings, also z. B. Vorgaben hinsichtlich Qualität, Zeit und Budget zu investieren. Erst wenn die Voraussetzungen vollständig geschaffen sind, wird die klassische Engineering Kunst in vollem Umfange eingesetzt.

2.5 Distinktionsmerkmal für Deutschland

Engineering ist eine stark mit Deutschland assoziierte Disziplin. Allen voran der deutsche Maschinenbau und die deutsche Automobilbranche sind globale Vorzeigeindustrien, die allesamt via der Aussage »Made in Germany« beworben werden.

»Entstanden ist der Ursprungsbegriff »Made in Germany« in der zweiten Hälfte des 19. Jahrhunderts, als sich britische und deutsche Exporteure einen heftigen Verteilungskampf um die Weltmärkte lieferten. Die Anweisung der britischen Regierung an ihre Überseekolonien und Territorien, deutsche Erzeugnisse nur noch mit einer entsprechenden Ursprungsmarkierung hereinzulassen, erwies sich bald als kontraproduktiv. Die Hochwertigkeit deutscher Produkte sprach sich schnell herum und es dauerte nicht lange, da galt das »Made in Germany«-Siegel als besonderes Qualitätsmerkmal.« [5]

Im globalen Vergleich werden von deutschen Produkten (und Software kann hier als Produkt aufgefasst werden) die beiden folgenden zwei Distinktionsmerkmale erwartet:

- Hohe Qualität: »Noch heute gelten deutsche Produkte international als besonders wertbeständig, technisch anspruchsvoll und langlebig.« [5]

- Verlässlichkeit als Vertragspartner: »Hinzu kommt, dass deutsche Exporteure als zuverlässige Geschäftspartner gesehen werden, auf deren Vertragstreue man sich im Ausland verlassen kann.« [5]

Nach einem Urteil des BGH (vgl. [5])bedeutet das Siegel »Made in Germany« dabei keineswegs, dass alle Herstellungsprozesse in Deutschland stattfinden. Die Verwendung von Offshore-Ressourcen stellt von daher kein Widerspruch zu den deutschen Distinktionsmerkmalen dar. Wichtig ist lediglich, dass wesentliche geistige Leistungen, das Design sowie der technische Innovationsstand aus Deutschland stammen: »So hat das OLG Stuttgart 1995 entschieden, dass auch dann, wenn ganze Baugruppen im Ausland zugekauft werden, die Bezeichnung »Made in Germany« noch erlaubt ist, sofern die (Produkt-) Eigenschaften, die für die Endware nach Einschätzung der Verbraucherkreise im Vordergrund stehen, in Deutschland erbracht wurden [...]« [5].

Qualität und Verlässlichkeit zielen im Kern bereits auf das Engineering: Qualität ist dort explizit durch eigenständige Teildisziplinen angesprochen und die Verlässlichkeit basiert im Wesentlichen auf der Zielgerichtetheit des Vorgehens und der Entwicklung.

Den Transfer des Engineerings auf die Software-Branche sieht z. B. auch die Unternehmensberatung Roland Berger als eine hervorragende Ausgangsposition für die deutsche bzw. die europäische Softwarebranche. Speziell für das Anbieten Software-basierter Lösungen und Dienstleistungen über die Cloud heißt es dort »Anbieter aus Europa werden nach unserer Erfahrung mit Werten wie Vertrauen, Zuverlässigkeit, Sicherheit und Datenschutz in Verbindung gebracht. Dies ist wichtig, da in Europa [...]

ein besonnener Umgang mit Daten erwartet und auch gepflegt wird. Dieser Vertrauensvorsprung könnte zum entscheidenden Wettbewerbsvorteil für europäische Cloud-Anbieter werden.« [6]

Es ist hervorzuheben, dass neben der allgemeinen Zielsetzung des Engineerings als Distinktionsmerkmal auch bereits eine bestimmte Gewichtung der drei Zieldimensionen Qualität, Zeit und Budget implizit mitschwingt: Qualität ist hierbei das dominierende Element, dem Zeit und Budget nachgelagert folgen. Hat ein Produkt einen anderen Schwerpunkt (z. B. Budget), kann dies durchaus noch dem Paradigma des Engineerings entsprechen, ist aber nicht mehr zwangsläufig an Deutschland geknüpft. Eine Verfeinerung des Paradigmas für den Standort Deutschland in »Qualitativ hochwertiges Software Engineering« ist folglich korrekter.

2.6 Alternative Paradigmen

Natürlich ist Engineering nicht das einzig mögliche Paradigma. Auch wenn gerade für Deutschland qualitativ hochwertiges Software Engineering prägend ist, sollen für eine bessere Kontrastierung wenigstens zwei alternative Paradigmen kurz vorgestellt werden.

2.6.1 Ad-hoc

Der Präfix »Ad-hoc« wird heute in vielen Kontexten verwendet (z. B. ad-hoc Analysen, ad-hoc Testen, ad-hoc Maßnahmen, etc.). Entlang der Merriam Webster Enzyklopädie beschreiben ad-hoc-Tätigkeiten vor allen Dingen solche Aktivitäten, die »for the particular end or case at hand without consideration of wider application« gedacht sind. Völlig anders als Engineering umfassen ad-hoc-Tätigkeiten also keine ganzheitliche Sicht (vgl. z. B. die 10 Unterdisziplinen des Engineerings) und sind bewusst auf Einmal-Aktivitäten beschränkt, die erst einmal keine Systematik für eine spätere Wiederverwendung oder für Feedback-Schleifen benötigen.

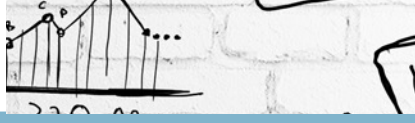
Ad-hoc kennt aufgrund der eingeschränkten Sicht keine ausgewiesenen Ziele, keine Steuerung, die ohnehin nur mit expliziten Zielen möglich wäre, und keine Nachhaltigkeit. Letzterer Aspekt birgt ein hohes Risiko, da kurzfristige Erfolge andere Seiteneffekte dominieren. Qualität und Verlässlichkeit, beides deutsche Distinktionsmerkmale, lassen sich aber immer nur langfristig erarbeiten.

Auch im Software-Bereich kommt das Ad-hoc-Paradigma zum Einsatz. Dies vor allem in zwei typischen Anwendungsfällen:

- Incident-Management: Treten Software-Unfälle auf und kann das systematische Engineering keine Hilfe mehr anbieten, werden häufig ad-hoc-Maßnahmen als letzte Möglichkeit in Betracht gezogen. Zu diesem Zeitpunkt wird ein etwaiges Engineering-Paradigma ausgesetzt und der Ausnahmezustand ausgerufen. Typische Anwendungsfelder im Software-Bereich sind akute Virenbefälle oder auch disruptive Änderungswünsche für bestehende Projekte, die wesentliche Vorergebnisse obsolet werden lassen.

- Prototyping: Da das klassische Engineering aufgrund seiner vielschichtigen Sichtweise eher schwerfällig in Bezug auf Zeit und Budget erscheint, wird für schnelle Durchstiche und Piloten häufig das ad-hoc-Paradigma verwendet. Auch hier gilt häufig der Ausnahmezustand und typische Aspekte wie Risikoanalyse, Qualitätssicherung, Planung und Dokumentation werden kurzerhand zugunsten des einen einzigen Ziels, irgendetwas Lauffähiges möglichst früh demonstrieren zu können, (vorübergehend) ausgeschaltet.

Beide Anwendungsfelder kommen sowohl reinrassig in Unternehmen vor, die sich nur dadurch behaupten können, wie auch eingewoben in klassische Vorhaben, welche solche Methoden etwa im Rahmen von Innovationsprojekten gezielt einsetzen. Die wenigen Szenarien, in denen sie darüber hinaus in klassischen Softwareprojekten zum Einsatz kommen, sind meist ungern tolerierte Ausnahmen, die es zu minimieren gilt. Ein gutes Engineering kann dieselben Aktivitäten motivieren wie ein ad-hoc-Paradigma, nur existieren im ersteren Fall wohldefinierte Rahmenparameter und ganzheitliche Abschätzungen.



2.6.2 Explorativ

Das Engineering benötigt per definitionem eine klare Zielvorgabe, die zu erfüllen ist. Gerade in einem wissenschaftlichem Kontext ist diese Zielsetzung allerdings wiederum per definitionem nicht bekannt: Wäre sie bekannt, würde das wissenschaftlich Neue in Frage gestellt sein. Explorative Vorgehen zeichnen sich also dadurch aus, dass der Weg das Ziel ist in der Hoffnung, wertvolle Ergebnisse beliebiger Art vorzufinden.

Exploratives Vorgehen ist eng verknüpft mit heuristischem Vorgehen und fokussiert ein Erkunden und Suchen (anstelle eines Erreichens). Das explorative Vorgehen sieht sich in einem Entdeckungszusammenhang, in dem alles erlaubt ist. Der Gegensatz dazu ist der Begründungszusammenhang, in dem »die Hypothesenprüfung nach strengen Kriterien im Mittelpunkt steht« [7]. Exploratives bzw. heuristisches Vorgehen basiert demnach auf Methoden des alltäglichen Lernens und »begnügt sich meist mit einer spielerischen und ungeplanten bzw. mäßig geplanten Vorgehensweise« [7]. Das explorative Vorgehen wechselt im Gegensatz zum Engineering während der Durchführung den Forschungsgegenstand – frei nach dem Motto: »wenn ich etwas suche, finde ich vielleicht etwas anderes«.

Die Anwendungsfelder des explorativen Paradigmas sind entsprechend konträr dem Engineering und ad-hoc Paradigma:

- Es gibt Vermutungen oder wenigstens Intuitionen, die ein Ergebnis vermuten lassen, allerdings ist die Zielsetzung noch zu unscharf, um Engineering-mäßig bearbeitet zu werden. Das explorative Testen hat beispielsweise genau dieses Ziel: Es dient als Voraktivität zu systematischen, in das Engineering eingebetteten Tests, welche das Ziel hat, den Testgegenstand erst einmal soweit kennenzulernen, dass eine Planung und ein Design möglich ist.

- Ein Problem lässt sich anders nicht mehr lösen: Exploratives Vorgehen benötigt nicht einmal eine Idee: Allein die Hoffnung, beim Beschreiten eines Weges irgendetwas mit Mehrwert zu finden, genügt oft als Motivation.

Das explorative Vorgehen wird heute zumeist in universitären und forschungsbezogenen Bereichen angewendet. Es lässt sich als Vorstufe eines Engineering-mäßigen Vorgehens verstehen, in dem alle für die Zieldefinition des Engineerings notwendigen Parameter erkundet werden. Für den Testbereich hat sich hier z. B. die Disziplin des explorativen Tests herauskristallisiert (vgl. [4])

3 Disziplin Projektmanagement

3.1 Definition

Wenn wie oben formuliert das Paradigma die grundsätzliche Sichtweise auf Probleme und Lösungen definiert, so genügt das für das operative Software-Geschäft noch nicht. Gefragt sind Disziplinen, also Teilbereiche des Wissenskanons, die helfen, das Paradigma zu unterstützen. Im Folgenden wird das konkrete Paradigma Engineering als Ausgangsbasis herangezogen und nun mit entsprechenden, das Engineering unterstützenden Disziplinen unterfüttert. Allen voran und sicherlich zu einem nicht unwesentlichen Teil auch wieder einer spezifisch deutschen Sicht geschuldet geht es daher um die Disziplin der Projekte und des Projektmanagements:

Ein Projekt ist dabei wie folgt definiert:

»Ein Projekt ist ein zielgerichtetes, einmaliges Vorhaben, das aus einem Satz von abgestimmten, gelenkten Tätigkeiten mit Anfangs- und Endtermin besteht und durchgeführt wird, um unter Berücksichtigung von Zwängen bezüglich Zeit, Ressourcen (zum Beispiel Geld bzw. Kosten, Produktions- und Arbeitsbedingungen, Personal) und Qualität ein Ziel zu erreichen.« [8], Abschnitt 3.4.3

Bereits diese Definition zeigt hervorragend die Eignung von Projekten, die mit dem Engineering verbundenen Ziele zu erfüllen. Das gesamte Initiieren, Aufsetzen, Planen, Steuern, Durchführen und Abschließend von Projekten obliegt dabei dem sogenannten Projektmanagement.

3.2 Software-Projektmanagement

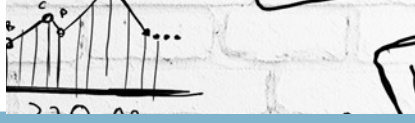
Das Projektmanagement wird seine erstaunliche Entwicklung der letzten drei Jahrzehnte auch in den nächsten Jahren rasant fortsetzen. Laut der Studie »Expedition Deutschland« der Deutschen Bank werden in 2020 schon 15 % der Wertschöpfung in Deutschland durch die Projektwirtschaft generiert. Im Jahr 2007 waren es noch 2 %. Projektarbeit wird immer mehr zum Erfolgsfaktor. In den Unternehmen wird es neue Herausforderungen an die Arbeitsweisen und die Organisation geben. Die Modelle von Morgen sind projektorientierte Strukturen und reine Projektorganisationen. Der Projektmanagementkompetenz als Voraussetzung für erfolgreiche Projekte kommt immer mehr Bedeutung zu, um die Erfolgsquoten der Projekte nachhaltig zu steigern. Nach der Studie der Standish Group waren im Zeitraum 1998 bis 2006 nur ca. ein Drittel der Projekte erfolgreich im Sinne der Erreichung der Projektziele (Aufgaben, Qualität, Zeit und Budget).

3.3 Ziele

Ein wichtiges Ziel des Projektmanagements ist es, Einflüsse auf Qualität, Zeit und Budget zu messen, zu steuern und zu kontrollieren und damit die Ziele des Engineerings (s.o.) zu erreichen. Im Folgenden werden einige wesentlichen Bereiche von Aktivitäten aufgezeigt, die durch ein effektives Projektmanagement sichergestellt werden müssen. Diese Bereiche sind es dann auch, die die steuernde Kraft des Projektmanagements ganz maßgeblich bestimmen.

3.3.1 Anforderungsmanagement

Das Projektmanagement soll beschreiben, wie Ziele des Entwicklungsvorhabens identifiziert werden. Dies können Business-Ziele, technologische Ziele usw. oder auch Nicht-Ziele sein. Nach Möglichkeit sollte es sich um quantitative Ziele handeln, damit die Zielerreichung mit Hilfe entsprechender Metriken verifiziert werden kann. Aus den Zielen der Produktentwicklung werden Anforderungen abgeleitet. Diese wiederum sind die Grundlage für Konzepte. Konzepte werden implementiert. Eine wichtige Methode zur Kontrolle ist hier die Rückverfolgung der Ziele und Anforderungen in den Konzepten, der Implementierung usw.



3.3.2 Qualitätssicherung

Prozesse geben Verfahrenssicherheit. Auch agil durchgeführte Projekte folgen Prozessen und haben eine Ablauf- und Terminplanung, die sich meist nur durch die Länge der Lieferzyklen (Iteration, Sprints) von anderen Prozessmodellen unterscheidet. Auch die kollektive Verantwortung für die Qualität der Ergebnisse ist kein Alleinstellungsmerkmal agiler Projekte. Methoden wie Pair-Programming oder Pair-Testing haben sich längst auch in prozessorientierten Vorgehensmodellen bewährt und gehören zum Instrumentarium des Projektmanagements. Ähnlich verhält es sich mit dem in einer Projektorganisation etablierten Qualitätsverständnis. Viele aus der industriellen Produktion stammende Modelle verwenden seit den 60er Jahren die Begriffe »Null-Fehler-Produktion« oder »Null-Fehler-Toleranz«. Obwohl völlig klar ist, dass es weder fehlerfreie Software gibt noch deren Nachweis erbracht werden kann, hat sich die »Null-Fehler-Toleranz« als ideelles Ziel bewährt, um deutlich zu machen, dass Fehler nie als Normalfall betrachtet werden dürfen und eine Fehlerquote ungleich Null nicht akzeptabel ist.

Das Projektmanagement sollten Metriken bereitstellen, die eine quantitative organisatorische Qualitätssicherung ermöglichen. Möglich sind hier Projekt-, Prozess- und Produktmetriken. Eine einfache Projekt-Metrik zur Einhaltung von Ablauf- und Terminplänen ist z. B. die Messung der Zeitdifferenzen zwischen den geplanten und erreichten Terminen für Meilensteine. Darauf basierend können mit der Methoden z. B. der Meilensteintrendanalyse terminliche Risiken kontrolliert werden.

Weitere Metriken, beispielsweise als Bestandteil von Methoden wie Kanban, sind die Anzahl der Aufgaben, die gleichzeitig in einem Projektschritt bearbeitet werden können (»Work in Progress«), der Durchsatz, d.h. die Anzahl der Aufgaben, die in einem definierten Zeitabschnitt erledigt werden, und die Fehlerrate, d.h. die Anzahl der Fehler in einem bestimmten Zeitabschnitt oder einer Teststufe.

Eine weitere Prozessmetrik ist die Produktivität. Sie wird meist definiert als das Verhältnis von Umfang der produzierten Software (ermittelt mit Hilfe einer Umfangsmetrik, beispielsweise Function Point oder andere, meist unternehmensspezifische Methoden) zu Aufwand, der für einen definierten Entwicklungsprozess oder -abschnitt benötigt wird (beispielsweise in Mann-Tagen).

Ein weiteres Ziel des Projektmanagements ist die Bereitstellung von Testmethoden und Produktmetriken, mit denen die Erfüllung der Anforderungen verifiziert werden kann. Funktionale Anforderungen werden meist durch funktionale Tests überprüft. Für nicht-funktionale Anforderungen gibt es beispielsweise Lasttests, Usability-Tests, Sicherheitstests usw. Quantifizierbare Anforderungen erfordern Metriken. Beispiel: Der Code soll eine Kommentardichte von 30 % haben – gemessen durch den Quotienten aus der Anzahl Kommentarzeilen (ohne leere Kommentarzeilen und ohne auskommentierten Code) und der Gesamtzeilenzahl.

Quality Gates stellen eine vorhersagbare Ergebnisqualität am Ende bestimmter Projektphasen oder für die Verwendung bestimmter Ergebnisse sicher. Auch agile Projekte sind darauf angewiesen, dass durch die Beschreibung von Anforderungen alle vorher definierten Projektziele berücksichtigt werden, der Code vor einem Build-/Integrationsprozess syntaktisch korrekt ist und ggf. dem vereinbarten Style-Guide entspricht, ein auszulieferndes Release vollständig und robust ist usw. Quality-Gates werden durch eine Menge von Prüfungen (Testmethoden oder auch Schwellenwerte für Metriken) und Erfolgskriterien definiert, die das jeweilige Testobjekt erfüllen muss, damit das »Gate« durchschritten werden darf, d. h. das Testobjekt an den nachfolgenden Entwicklungsschritt übergeben werden darf.

Das Dilemma, dass intensives Testen zwar das Risiko von Fehlern in einem Produkt senkt, sich jedoch nachteilig auf Zeit und Budget auswirkt, kann heute durch die Automatisierung vieler Testmethoden gelöst werden. Gerade agile Projekte mit ihren meist kurzen Iterationen sind auf eine »permanente« analytische Qualitätssicherung angewiesen. Dies kann durch eine kontinuierliche Integration

erreicht werden, bei der beispielsweise jede Nacht der gesamte Code per Codeanalyse auf Korrektheit und Konformität zum Style-Guide geprüft, durch einen Build-Prozess zu einer lauffähigen Anwendung integriert wird, die Anwendung anschließend funktional durch Unit-Tests der Entwickler oder durch Skripte getestet wird. Es gibt Tools, mit denen komplette End-to-End-Tests per Capture & Play oder Skriptprogrammierung erstellt und im Anschluss an einen automatischen Build-Prozess ausgeführt werden können. Das Entwicklungsteam hat so zu Beginn jedes Arbeitstags ein Protokoll vorliegen, welches dem Team entweder Fehler aus den Tests der letzten Nacht aufzeigt, beispielsweise aufgrund von Seiteneffekten ihrer letzten Änderungen, oder eine adäquate Qualität und Reife bestätigt.

3.3.3 Wissensmanagement, kontinuierliche Verbesserung

Weitere Ziele des Projektmanagements sind das Management von Wissen sowie eine kontinuierliche Verbesserung der eingesetzten Ressourcen und der angewendeten Prozesse (hier also vor allen Dingen Projektmanagement im Gefolge des Engineerings). Ein Erfolgsfaktor agiler Projekte ist, dass sich Teammitglieder durch die intensive Zusammenarbeit gut bezüglich ihres impliziten Wissens austauschen und ergänzen. Fällt implizites Wissen aufgrund von Änderungen in der Teamzusammensetzung weg, treten jedoch oft Probleme auf. Das Projektmanagement soll sicherstellen, dass Wissen explizit gemacht wird. Dies kann durch Dokumentation geschehen – ob in Dokumenten oder Wikis – durch Strukturierung und Kommentierung des Codes usw.

Neben dem Management von Wissen soll das Projektmanagement auch das Lernen, nicht nur einzelner Mitarbeiter sondern auch der ganzen Organisation, unterstützen. Als Basis einer kontinuierlichen Verbesserung der Prozesse, Methoden, Werkzeuge usw., in der Hauptsache also Maßnahmen der konstruktiven Qualitätssicherung, dienen hauptsächlich die Ergebnisse aus der analytischen Qualitätssicherung (z. B. Fehler-Ursachen-Analysen), Retrospektiven, Projekt-Reviews usw.

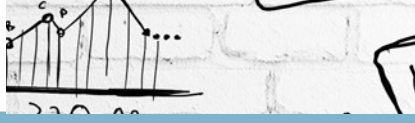
3.3.4 Risikomanagement

Risikomanagement identifiziert als Teil des Projektmanagements schädliche Einflüsse auf Qualität, Zeit und Budget und betrachtet deren Wahrscheinlichkeit und die möglichen Auswirkungen. Gerade in agilen Projekten mit kurzen Lieferzyklen ist es wichtig, eine Bedrohung der Ziele frühzeitig zu erkennen und ad-hoc wirksame Gegenmaßnahmen ergreifen zu können. Das Risikomanagement hat neben der Risikoreduktion allerdings immer als Gegenpol die Kosten- und Zeitparameter: Die Reduktion des Risikos führt meist zu einer deutlichen Kostensteigerung und Auslieferungsverzögerung, was seinerseits trotz der dann hohen Qualität zu reduzierten Marktchancen führen kann. Risikomanagement hat hier also mit einem fortwährenden Abwägen (Tradeoff) zwischen Risiko und Chance zu tun.

3.4 IT-Projektmanagement

Im Folgenden findet sich eine zusammengefasste aber detaillierte Darstellung der einzelnen Geschäftsvorfälle, die im Rahmen des Projektmanagements von IT-Projekten zu bearbeiten sind. Die Darstellung ist bewusst ohne prozessuale Verknüpfungen gewählt. Ziel ist es, dem verantwortlichen Projektleiter und seinen Mitarbeitern alle planerischen, kaufmännischen, steuernden und managementtypischen Aufgaben aufzuzeigen. Da diese Aufgaben eines ganzheitlichen Projektmanagements sich nicht zwingend in einen allgemeingültigen Ablauf einordnen lassen, sondern teilweise parallel oder bei aktuellen Bedarf einzeln durchgeführt werden, sind die zeitliche Abfolge im zu erstellenden konkreten Projektablaufplan den jeweiligen Zielterminen anzupassen und inhaltliche Abstimmungen durchzuführen.

Ein für dieses individuelle Zuschneiden sinnvoller Katalog von Einzelprozessschritten ist in der folgenden Abbildung dargestellt:



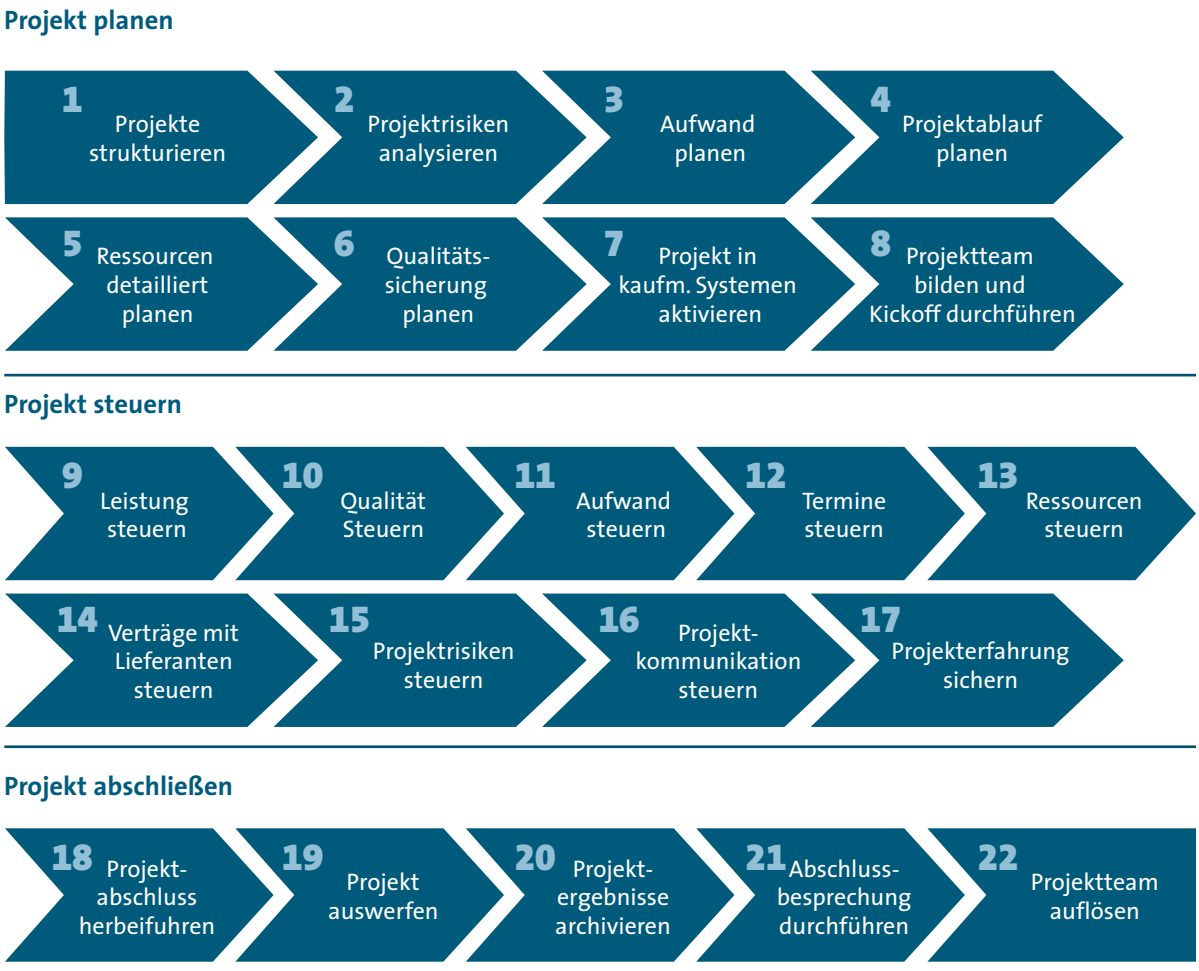


Abbildung 2: Vereinfachtes Prozessmodell

Zu berücksichtigen ist dabei, dass Vorgehensmodelle sich in der Regel auf die eigentliche Softwareentwicklung konzentrieren und im Rahmen des Projektmanagements bei der Leistungserbringung des Projektes eine wesentliche Rolle spielen. Dabei sollte der Charakter der jeweiligen Arbeitsaufgabe, das zu wählende Vorgehensmodell bestimmen (s. Kapitel 4).

3.5 Erfolgsfaktoren für Projekte

Es ist kein Geheimnis, dass die Anzahl und Komplexität der Projekte in Unternehmen stetig zunehmen: Wo heute 3% der gesamten Wertschöpfung in Projektarbeit abgewickelt wird, wird sich diese Zahl in 10 Jahren verfünffacht haben. Je mehr die Anzahl der Projekte steigt, desto stärker hängt auch der gesamte Unternehmenserfolg von der Leistung dieser einzelnen Projekte ab. Uns steht demnach ein Strukturwandel innerhalb der Unternehmen, der viele Chancen mit sich bringt, bevor. Die Unternehmen werden aber auch mit großen Risiken konfrontiert – gerade wenn man bedenkt, dass immer noch rund 2/3 aller gestarteten Projekte nicht planmäßig verlaufen oder sogar komplett scheitern.

Die Gründe für dieses Scheitern, liegen dabei hauptsächlich in der Initiierungsphase, was ein professionelles Aufsetzen und Steuern von Projekten demnach unabdingbar macht und sind überwiegend durch fehlendes Wissen über passende Projektmanagementmethoden und deren richtiger Anwendung begründet. Weitere Gründe sind außerhalb des Projektes bzw. des direkten Einflusses des Projektmanagements angesiedelt (vgl. Abbildung 3).

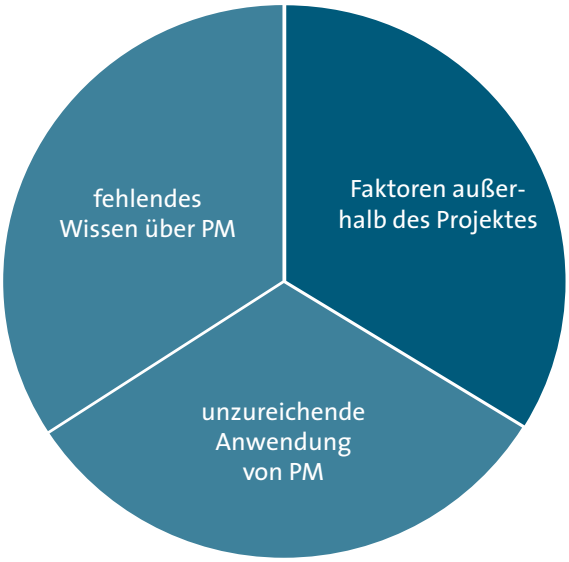


Abbildung 3: Gründe für das Projektscheitern

Was können Unternehmen tun, um Projekte im Sinne Ihrer eigenen Ziele erfolgreicher zu machen? Die wesentlichen Hebel sind hier der Einsatz von Projektmanagementspezialisten sowie der Ausbau der eigenen Projektmanagementkompetenz im Unternehmen. Eine höhere Integration von Projektmanagementprozessen und -werkzeugen sowie die Adaption von Projektmanagementstandards können Unternehmen auf dem Weg zu erfolgreichen Projekten unterstützen. Aber auch Vorbedingungen hinsichtlich der Unternehmenskultur sind relevant, wie etwa eine Kultur des eigenverantwortlichen Handelns, das zwar Vorgaben erhält (z. B. über das Paradigma und die konkrete Disziplin), das Wie der Umsetzung aber nicht vorgibt.

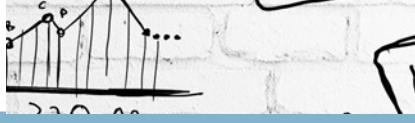
3.5.1 Externe Projektmanagement-spezialisten

Mit der zunehmenden Größe und Komplexität der Projekte, Programme und Projektorganisationen nimmt die Spezialisierung der Projektleitungsaufgaben immer mehr zu. Auf der einen Seite steht der traditionelle Projektleiter, der überwiegend die fachlichen und inhaltlichen Themen des Projektes betreut. Er ist verantwortlich für die Erreichung des Projektziels (Produkt oder Dienstleistung) und die Kommunikation mit den Fachleuten innerhalb und außerhalb des Projektes. Die organisatorischen und methodischen Aufgaben bei der Umsetzung der Projekte nimmt auf der anderen Seite das Projektmanagement war. Der Fokus des Projektmanagements liegt in der Einhaltung der vorgegebenen Termine, der Budgets und der Qualität des Projektes. In diesem Bereich sind Projekterfahrung und Methodenkompetenz von wichtiger Bedeutung.

Die bisherigen Organisationsformen, dass das Projektmanagement der Projektleitung unterstellt ist, wird sich immer mehr verändern. Die beiden Aufgabenbereiche werden zunehmend gleichberechtigt agieren, da die jeweiligen Projektziele mitunter konträr zueinander stehen. Es gibt heute schon Projektorganisationen, in denen der Projektleiter (im Sinne eines technischen Projektleiters) dem Projektmanager (im Sinne eines kaufmännischen Projektleiters) unterstellt ist.

3.5.2 Interne Projektmanagementkompetenz

Die zeitliche Beschränkung von Projekten bedeutet auch, dass nach Ablauf eines Projektes die Erfahrungen bei der Umsetzung des Projektes, hinsichtlich der Anwendung der Methoden und die entstandenen Tools und Techniken dem Unternehmen nicht mehr oder nur sehr eingeschränkt zur Verfügung stehen. Die weitere Anwendung dieser Erkenntnisse und Methoden auf neue Projekte hängt dann nicht von der Organisation, sondern von den jeweiligen Projektmitarbeitern ab. Um diesem Erfahrungsverlust entgegenzuwirken, wird die Projektmanagementkompetenz durch die Einrichtung sogenannter



PMOs (Project Management Office) in der Organisation verankert und die gemachten Erfahrungen im Kontext des Wissensmanagements (vgl. Kapitel 3.3.3) für eine kontinuierliche Verbesserung aufgearbeitet

Diese PMOs sind entweder taktisch (lokal) oder strategisch (global) aufgestellt. Erfahrene Projektmanager prägen die Methoden, Richtlinien und Standards (zum Beispiel durch ein Projektmanagementhandbuch) aus, unterstützen die Projektdurchführung durch Audits und Reviews (vgl. hierzu auch Kapitel 3.3.2), führen ein einheitliches Berichtswesen ein und etablieren so die unternehmensspezifische Projektmanagementkultur. Oft werden auch Aufgaben im Portfolio- und Programmmanagement sowie in den Lenkungskreisen übernommen.

3.5.3 Integration von Projektmanagementprozessen und -werkzeugen

In der Abwicklung von Projekten haben sich im Laufe der Zeit bestimmte Tools und Techniken sowie Prozessabläufe ausgeprägt, die oft in der gesamten Prozess- und Toollandschaft als Insellösungen agieren. Bei der künftigen Abwicklung größerer und komplexerer Projekte gilt es, diese zu integrieren. Das betrifft sowohl die bestehenden System- und Prozesslandschaften (Enterprise-Systeme, Collaboration-Tools, unterschiedliche Frontends für zentrale Services, etc.), als auch die Insellösungen und -prozesse der Projektabwicklung (Planungstools, Portfoliomanagement, Templates etc.).

Zwischenzeitlich haben auch die Lösungsanbieter die Voraussetzungen für ein »Enterprise Project Management« geschaffen, um den Schritt von den »Insellösungen« hin zu den »verbundenen Prozessen« und den »integrierten Werkzeugen« umzusetzen.

3.5.4 Projektmanagementstandards

Im Projektmanagement können unterschiedliche Standards zur Anwendung kommen. Die Auswahl der Standards orientiert sich an den eigenen Standards, an vorhandenen Erfahrungen der Mitarbeiter und den Unternehmensbedürfnissen. Die bekanntesten Standards sind:

- IPMA/GPM: IPMA steht für die International Project Management Association (IPMA). Die nationale Ausprägung in Nürnberg heißt Gesellschaft für Projektmanagement (GPM). Der Schwerpunkt dieses Projektmanagementstandards liegt auf der Projektmanagementkompetenz. Dieser Standard ist in Deutschland und Europa stark vertreten.

- PMI®: PMI steht für das Project Management Institute (USA). Auch hier gibt es nationale Dependancen, sogenannte Chapter. In Deutschland sind die z. B. in Frankfurt, Köln, München, und Berlin. Der Schwerpunkt von PMI liegt auf den Projektmanagementprozessen; aufgrund seiner amerikanischen Wurzeln ist PMI vor allen Dingen in den USA vertreten.

- PRINCE2: PRINCE2 steht für Projects in Controlled Environments Verison 2. In Deutschland wird PRINCE2 vertreten durch den PRINCE2 Deutschland e. V. in Dreieich. Der Schwerpunkt von PRINCE2 liegt auf standardisierten Projekten und der Sammlung unterschiedlichster Best Practices. PRINCE2 ist vor allen Dingen vertreten im vereinigten Königreich und in den Niederlanden.

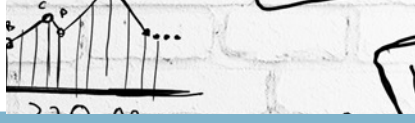
- DIN 69901. Die DIN als Deutsche Industrie-Norm beschreibt Grundlagen, Prozesse, Prozessmodelle, Methoden, Daten, Datenmodelle und Begriffe im Projektmanagement. Inhaltlich ist die DIN 69901 stark durch die GPM geprägt. Im Einsatz ist die DIN 69901 vor allen Dingen in Deutschland.

- ISO 21500. Die Internationale Standard Organisation hat mit der ISO 21500 einen »Guide to Project Management« veröffentlicht. Die Inhalte orientieren sich dabei eng an PMI. Dieser Standard ist noch sehr neu (veröffentlicht Ende 2012).

3.6 Distinktionsmerkmal für Deutschland

Das Projektmanagement, also das strukturierte und disziplinierte Vorgehen, scheint stark in der deutschen Kultur verankert zu sein. Es gibt eine Vielzahl von Untersuchungen, was die kulturellen Spezifika von Deutschland sind. Einige hochrangige Referenzen sind mit jeweils typischen Textbausteinen:

- Patrick Schmidt, American-German Cross-Cultural Consulting, in einem Vortrag vor dem Goethe Institut [9]
  - »American casualness clashing with German formality«
  - »The simplistic can-do approach turned out to be incredibly successful [...] All in all, the American Experience quickly revealed a multitude of reasons for rejecting the rigors of European behavioral codes.«
  - »Driven by the value -time is money-, American's are quick to change their way of thinking, quick to try something new, always ready to make a deal; Contrast this with German austerity, discretion, objectivity.«
  - »Germans at work communicate basically on the objective level; social and personal factors, although nice, are considered secondary and not necessary.«
  - »There is a strong emphasis [für Deutschland, Anm.d.A.] on being objective, which tends to make their conversation factoriented and somewhat formal You can experience this every evening at 8 o'clock when ARD presents the news. The news announcer is the quintessence of German objectivity, speaking in steady monotone voice, appearing to display absolutely no emotion. No matter what may be happening in the world, objectivity remains.«
  - » Typically German, i.e. perfection, objectivity and need for order«
  - »Perfectionist behavior, absolute correctness, colorless objectivity...from 1945 on, these became a kind of Leitmotiv to feelings of worthlessness in an environment of total chaos.«
- Patrick Schmidt, American-German Cross-Cultural Consulting, in einem Artikel im »Trade«-Magazin, wiederveröffentlicht im DIA (Delta Intercultural Academy), 2004 [10])
  - »German business conversation emphasizes content and downplays personal relationships. The unconscious desire is to appear credible and objective, making discussions fact-oriented and often academic. The inherent goal is to get at the truth (Wahrheitssuche). Germans aren't afraid to explore all sides of an issue, even if it means being unpleasant, confrontational and spending an excessive amount of time analyzing a problem.«
  - »[...]they're [Deutschen, Anm. d.A.] generally very direct when it comes to stating facts, offering criticism and giving orders.«
  - »Adapting to change and coping with uncertainty is the second major area where Americans and Germans differ. The latter show a high degree of uncertainty-avoidance and behavioral rules, both written and unwritten, are rigid. Knowledge is respected and «experts" seldom questioned. Projects are thoroughly researched and risks are kept to a minimum. The more structure there is, the better.«
  - »Schroll-Machl [eine Psychologin, Anm. d. A.] noticed that, at the outset of a project, Germans showed a greater need for detailed information and discussion. They tended to see the process from an engineering point of view, considering all of the difficulties that might arise, planning hypothetical solutions. The goal was to make sure everything would be done correctly, every element



possible kept «under control". »The action-oriented Americans found these discussions trying, often outright boring«.

- »The Americans felt obsession with plans, and sticking to them, meant being locked into a rigid pattern, with no flexibility during the implementation-phase. Once a plan was established, German team members were able to work relatively independently.«

Das Denken in Projekten mit klaren Zielen, Anfang und Ende scheint also eine typisch deutsche Denkweise zu sein (für historische Begründungen siehe wieder [9]). Projektmanagement, also das professionelle und verlässliche Aufsetzen, Initiieren, Planen, Durchführen, Steuern und Abschließen von Projekttätigkeiten ist demzufolge ein deutsches Charakteristikum, das uns unter der Vorgabe des Paradigmas des Engineerings als klarer Wettbewerbsvorteil für den Standort Deutschland dienen kann.

3.7 Alternative Disziplinen

Die Disziplinen geben jeweils einzelne Wissensbereiche vor, die entlang des in diesem Leitfaden aufgespannten Dreischritts das Paradigma ermöglichen sollen. Das Projektmanagement definiert also diejenigen Bereiche, die für die Umsetzung von Engineering relevant sind. Diese Wissensbereiche (in Form der Disziplinen) können natürlich auch sehr viel leichtgewichtiger sein, wenn dies durch die Vorgabe in Form der Paradigmen möglich ist. Ein ad-hoc-Paradigma könnte mit der einfachen Disziplin des Tuns unterfüttert werden. Es bedarf aufgrund der eingeschränkten Betrachtung möglicher Einflussfaktoren keiner aufwändigen Planung, keiner Steuerung und keines Abschlusses. Das oben beschriebene action-oriented Vorgehen liefert Nuancen einer solchen Disziplin, der zufolge das Tun höher gewichtet wird als Planung, Design und Abschluss.

4 Methode Agilität

4.1 Definition

Die Methoden auf der untersten Ebene sind die konkreten Vorgehen, Werkzeuge und Best-Practices, die man, integriert in die jeweilige Disziplin, anwendet, um ein IT-System zu entwickeln, zu erweitern oder anzupassen.

Methode (griech. methodos = Weg, Zugang) bezeichnet im allgemeinsten Sinne eine Vorgangsweise, die sich im Hinblick auf ein bestimmtes Ziel als wählbarer Weg (Möglichkeit unter Möglichkeiten) aufzeigt. Im Kontext des BITKOM-Arbeitskreises »Softwareentwicklungsprozesse und Werkzeuge«, der diesen Leitfaden erarbeitet hat, fokussieren die folgenden Abschnitte vor allen Dingen die agilen Methoden der Softwareentwicklungsprozesse; grundsätzlich lassen sich aber auch andere Methoden im Dreiklang Paradigma, Disziplin, Methode anführen, z. B. in Form spezieller Werkzeuge, spezieller Programmiersprachen oder auch spezieller Architekturen.

Dieser Leitfaden versucht dabei, keine einzelne agile Methode hervorzuheben, sondern die agilen Methoden als Ganzes einzuführen und in den Kontext Disziplin und Paradigma zu stellen: »Nach unserer Erfahrung beantwortet nämlich keine agile Methode alle Fragen, die sich bei der Softwareentwicklung methodisch stellen.« [11].

Die folgende Definition, die lediglich aus zwei Einzeldefinitionen aus [11] kombiniert ist, erläutert das diesem Leitfaden zugrundeliegende Verständnis einer agilen Methode:

Eine agile Methode ist eine konkrete benannte Zusammenstellung agiler Praktiken, also von etablierten Herangehensweisen, in einem ausgewählten Ausschnitt oder Aspekt der Softwareentwicklung agil vorzugehen, also die agilen Werte zu berücksichtigen. (nach [11])

Die agilen Werte, entnommen dem agilen Manifest, taugen allerdings nur schwer für eine Definition von Agilität (als übergeordnete Integrationsklammer agiler Methoden). Die Nicht-Definierbarkeit von Agilität wurde von Fowler selbst bestätigt und erst von Westphal nachgeliefert:

Demnach kann eine Methode der Softwareentwicklung als agil bezeichnet werden, wenn die folgenden drei essentiellen Charakteristika erfüllt sind (vgl. Ralf Westphal hat in seinem Blog »Agil persönlich definiert« folgende Indikatoren benannt ([12]):

1. Inkrementell: Das Problem wurde in kleine logische funktionale Projekteinheiten zerlegt, die dem Kunden Mehrwert bringen. Durch die Zerlegung des Problems können funktionale Einheiten priorisiert werden, sodass zu jedem Zeitpunkt des Projektes die für den Kunden wichtigsten Funktionen zuerst ausgeliefert werden. Diese Einheiten (»User Stories«) werden als technischer Durchstich des Gesamtsystems realisiert und an den Kunden als sogenanntes Produktinkrement ausgeliefert. Durch das inkrementelle Vorgehen bekommt der Kunde bereits sehr früh im Projektzeitraum die Chance sich ein Bild von der Lösung zu machen und das Team kann mit dem Kunden eventuelle Änderungen abstimmen, bevor diese zu teuer werden.

2. Lernend: Einer der Grundpfeiler von Scrum & Co. ist die Retrospektive am Ende jeder Iteration. Die drei Fragen: Was war gut, was war schlecht und was wollen wir besser machen? zielen darauf ab, aus der vergangenen Iteration zu lernen und sich konstant weiterzuentwickeln.

   Nach Westphal bewegt sich das agile Team hinsichtlich des Produktes in einer Lernschleife:

   - Herausfinden, was gewünscht ist
   - Inkrement herstellen
   - Überprüfen, ob das Inkrement für den Kunden akzeptabel ist

3. Unmittelbar: Nach Westphal ist die Betonung der unmittelbaren Kommunikation ein Ausdruck von agilen Projekten. Im Sinne der Unmittelbarkeit ist die Zusammenführung von funktionsübergreifenden Teams ein wichtiger Schritt.

Ein Vorgehensmodell zur Entwicklung von Software, das inkrementell, lernend und unmittelbar ist, wird im Folgenden als Agil definiert.

4.2 Die Entstehung der Agilität

Es ist schon viele Jahre her, in den 1990ern war es, da kamen Ideen auf, die Entwicklung von Software effizienter zu gestalten. Im Jahre 2001 wurde dann das agile Manifest geschrieben, in dem die wesentlichen Optimierungsgedanken zusammengefasst und veröffentlicht wurden. Hier liegen die Ursprünge von dem, was heute als agil bezeichnet wird.

4.3 Das universelle agile Vorgehensmodell

Allen Methoden zur Erschaffung neuer IT-Systeme entlang der Disziplin Projektmanagement gemeinsam ist ein Vorlauf, in dem die Anforderungen an das Projekt genauer spezifiziert werden (vgl. Kapitel 3.3.1). So entsteht vor jedem Projekt zunächst eine Vision. Sie ist eine möglichst konkrete Vorstellung von dem Ergebnis, das aus dem Projekt hervorgehen soll.

Nach der Projektinitiierung bedienen sich agile Methoden hingegen feingranularer sogenannter Projekteinheiten. Eine Projekteinheit ist prinzipiell eine vollständige Zusammenfassung der klassischen Projektphasen. Das bedeutet, dass eine Projekteinheit jeweils

- eine Planungsphase beinhaltet, in der die Menge der Arbeitspakete für die Projekteinheit festgelegt wird;
- eine Ausführungs- und Controlling-Phase durchläuft, in der die Arbeiten ausgeführt und auf Korrektheit geprüft werden;
- in einer Abschlussphase endet, die jede Projekteinheit formal abschließt.

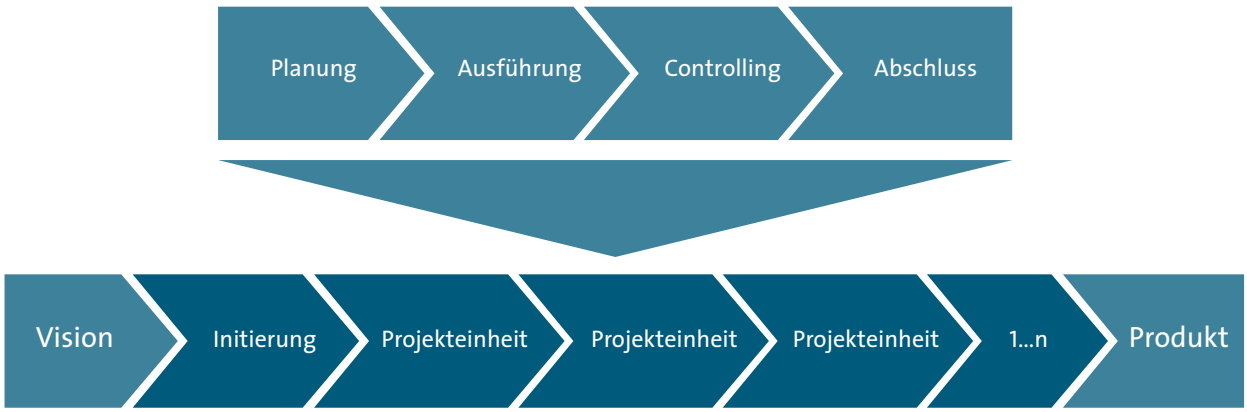


Abbildung 4: Allgemeiner Agiler Projektverlauf (nach [13])

Die Anzahl der Projekteinheiten variiert dabei je nach Art und Umfang der Arbeitspakete, die auf Basis der Vision festgelegt wurden. Somit stellt jede Projekteinheit nach dem Verständnis des klassischen Projektmanagements grundsätzlich ein eigenes Teilprojekt dar. Agile Methoden integrieren sich nahtlos in die Disziplin Projektmanagement.

In agilen Methoden wird ein abgeschlossenes Teilprojekt als Iteration bezeichnet. Eine Iteration ist eine Zusammenfassung von Arbeitstagen, an denen ein Teil der Aufgaben, die für ein Release vorgegeben wurden, erstellt werden. Diese zusätzliche Aufteilung ist zum einen der höheren Flexibilität geschuldet, andererseits wird nach jeder Iteration, jedem Release oder Projekt eine Retrospektive, ähnlich den Lessons-Learned, durchgeführt.

4.4 Erfolgsfaktoren für Agilität im Projekten

4.4.1 Inkrementelles Vorgehen

Ein wesentlicher Vorteil agiler Methoden ist das inkrementelle Vorgehen mit sehr kleinen Inkrementen. Unter der Vorgabe des Paradigmas Engineering benötigt die Disziplin Projektmanagement möglichst viele Steuerungspunkte, um den Status Quo der Entwicklung zu erfassen und ggf. gegensteuern zu können. Natürlich versucht das Projektmanagement fortwährend dieser Aufgabe gerecht zu werden, allerdings bietet die natürliche Zäsur zwischen zwei Inkrementen eine mächtige Steuerungsschraube. Unter der Prämisse, dass die sequentiellen Inkremente gegen das fertige Produkt konvergieren, gilt es explizit an diesen Zäsuren in den Dimensionen des Projektmanagements nachzusteuern:

- Sind Risiken aufgetreten?
- Entspricht die Teillieferungsqualität den Erwartungen des Kunden?
- Wie steht es um die Termin- und Budgettreue?

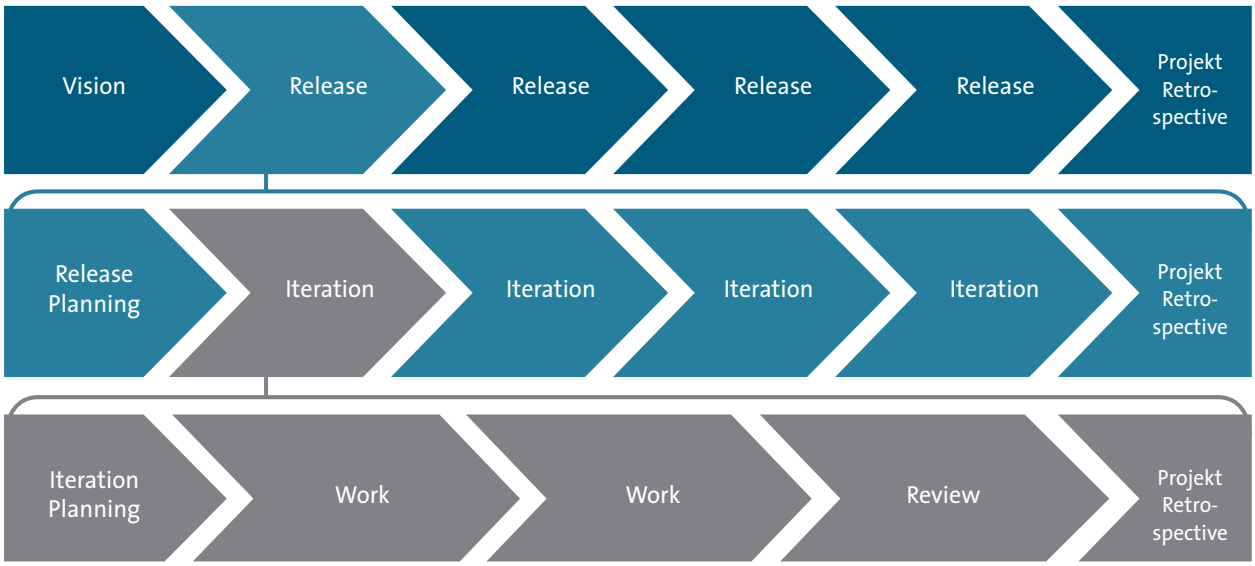
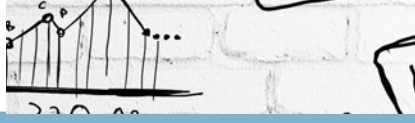


Abbildung 5: Agiler Projektlebenszyklus (nach [13])



Agilen Methoden wird häufig nachgesagt, dass die autonom agierenden Teams während der Abarbeitung einer Iteration nur bedingt steuerbar sind. Bis zu einem bestimmten Grad kann dies auch durchaus in dieser Form umgesetzt werden, um dann zwischen den Iterationen eine umso stärkere Steuerung zu erfahren. Da jede Iteration als autonomes Projekt geplant und durchgeführt wird sind auch die Abhängigkeiten zwischen den Iterationen beherrschbar: Entspricht ein Iterationsergebnis nicht der Erwartung kann das Ergebnis entweder bis auf die vorherige Version verworfen werden oder einfach das nächste Folgeprojekt angegangen werden.

4.4.2 Lernendes Vorgehen

Nur die Forderung nach einem inkrementellen Vorgehen ist noch nicht agil. Im Extremfall könnte dies ebenfalls durch die nahtlose Aneinanderkettung von Iterationsarbeiten erfolgen. Agile Methoden gehen aber mit der Zielvorgabe des lernenden Vorgehens einen Schritt weiter und laden Projektsteuerungsinstrumente (hier also die entsprechende Disziplin) explizit ein, Rückschau zu halten und steuernd nach vorne zu schauen.

Agile Methoden zeigen auch hier wieder ihre hervorragende Integrierbarkeit in das Projektmanagement: Allen Projektmanagementstandards gemeinsam ist eine Projektabschlussphase, in der Probleme reflektiert, Best-Practices extrahiert, und Lessons-Learned für nachfolgende Projekte identifiziert werden. Agile Methoden ihrerseits liefern mit ihrem lernenden Charakteristikum auf operativer Ebene genau die operative Umsetzung dieser Steuerungsanforderung.

Die typische Lernschleife in Form von Herausfinden, was gewünscht ist, ein Inkrement herstellen und abschließend prüfen, ob das Ergebnis den Anforderungen genügt, ist selbst wiederum hochgradig systematisch und steuerbar (z. B. mit Hilfe von Kennzahlen).

Ein wichtiger Aspekt des Lernens fokussiert das initiale Lernen, also solche Situationen, in denen agile Methoden das erste Mal zum Einsatz kommen. Typische Risikobereiche beim Einsatz agiler Methoden sind:

- Nicht alle Stakeholder tragen das Projekt mit.
- Die Stakeholder sind sich über die zu erreichenden Ziele, den Projektumfang und die dafür aufzubringenden Ressourcen noch nicht einig.
- Es gibt keine ausreichende Transparenz über den Gesamtumfang des Projektes.
- Es gibt keine ausreichende Transparenz über die zu erwartenden Auswirkungen des Projektes.
- Es gibt Parallelprojekte und Alternativlösungen, die nicht fallengelassen werden.
- Es gibt Abhängigkeiten zu anderen Projekten, die nicht angemessen berücksichtigt werden.
  Da der Einsatz von Ressourcen für die Schaffung der relevanten Projektvoraussetzungen (in Form der Abschaffung dieser Risiken) ggf. vergeblich aufgewendet sein könnte, besteht die Maxime darin mit minimalem Aufwand (agil) die oben genannten oder sonstige Widerstände aus dem Wege zu räumen. Der lernende Aspekt agiler Methoden unterstützt dies vor allem darin, so viel wie zwingend nötig, aber auf keinen Fall mehr als nötig an Aufwand in die planerischen Elemente des Projektes, also z. B. Anforderungsspezifikation, Zielsetzungen, Umsetzungspläne, Kalkulationen, Kommunikationsmaßnahmen zu investieren. Erst wenn die Voraussetzungen vollständig geschaffen sind (über mehrere Lern-Zyklen), werden agile Methoden wirklich effektiv im Rahmen des Projektmanagements das Engineering in vollem Umfange unterstützen.

4.4.3 Unmittelbare Kommunikation

Ein sehr offener Umgang mit jeglicher Form von Feedback ist ebenfalls essentiell für agile Methoden. Dies ermöglichte maximale, objektive Transparenz und stellt so eine Vorbedingung jeglicher Steuerung auf der Ebene der Disziplinen erst sicher: Wenn unklar ist, wo man steht, da verschleiert, geschönt, gelogen oder schwergewichtige Verträge vorgeschoben werden, können nur schwer notwendige Gegenmaßnahmen abgeleitet und ihr Erfolg wiederum bewertet werden. Die grundsätzlich angedachte Zielstrebigkeit von Engineering als Paradigma findet hier ihr operatives Pendant: Wenn etwas nicht gefällt so gehört dies so früh wie möglich kommuniziert. Etwaige Gegenmaßnahmen können entweder auf der Ebene der agilen Methoden oder aber auch auf der Ebene der Disziplin Projektmanagement stattfinden.

Da die Forderung nach unmittelbarer Kommunikation insbesondere auch den Kunden einschließt ist auch sichergestellt, dass kein häufig beobachtbares Zwei-Fronten-System entsteht: Auftraggeber und Auftragnehmer haben ein gemeinsames Ziel: Ein erfolgreiches IT-System zu erstellen, wobei »Erfolg« für den Auftraggeber meist bedeutet, dass deren Business-Prozesse durch das neue System effizient und effektiv unterstützt werden, während es für den Auftragnehmer zusätzlich bedeutet, entsprechend profitabel arbeiten zu können. Der offene Umgang aufgrund der unmittelbaren Kommunikation ermöglicht auf der Ebene des Projektmanagements ein gewinnbringendes Projektmanagement für beide Seiten.

Wichtig ist an dieser Stelle der Hinweis, dass unmittelbare Kommunikation einige Vorbedingungen haben muss. Neben einer entsprechenden Kultur (insbesondere für Kommunikation zwischen den Instanzen, d.h. z. B. unmittelbare Kommunikation zwischen dem klassischen Projektmanagement und der Softwareentwicklung) bedarf die unmittelbare Kommunikation z. B. der grundsätzlichen Möglichkeit der Kommunikation. Hierzu gehören Sprache, Zeitpunkt und geografische Verteiltung. Wichtig ist auch die vertragliche Ermöglichung von Kommunikation: Manches Festpreisprojekt hat explizit das Ziel KEINER Kommunikation, da Kommunikation a) schwer bzgl. des Aufwandes planbar ist und b) in der Regel nur Änderungen hervorruft, die ggfs. wiederum zu schwergewichtigen Änderungen des gesamten Vertragswerkes bewirken. Diese Vorbedingungen unmittelbarer Kommunikation sind i.d.R. umso schwerer zu erfüllen, je größer ein Projekt oder die beteiligten Organisationen sind.

4.5 Distinktionsmerkmal für Deutschland

Werden agile Methoden als das verstanden, als das sie in den vorherigen Abschnitten beschrieben werden, nämlich inkrementelle, lernende und unmittelbare Vorgehen, so unterstützen Sie hervorragend den Standort Deutschland als IT-Entwicklungsland:

- Der Binnenmarkt Deutschland/Europa und die Forderung nach unmittelbarer Kommunikation erfordern zwangsläufig eine hohe lokale Nähe zum Kunden. Da agile Methoden innerhalb einer Iteration häufig sehr autonom arbeiten ist zudem ein hoher Wissensstand bzgl. der Branche, für die ein IT-System erstellt wird, nötig, d.h. es wird von den Software-Erstellern nicht nur ein kulturell ähnlicher Hintergrund, sondern auch detaillierte Kenntnisse über Geschäftsprozesse, Regulatorien und Branchen-Nomenklaturen erwartet. Agile Methoden erfordern also für Binnenmarkt-Projekte (Deutschland/Europa) deutsche/europäische Software-Ersteller. Agile Projekte für den hiesigen Binnenmarkt können dagegen nur schwer vollständig aus Offshore-Ländern erbracht werden.

- Das für ein Lernen notwendige Kommunizieren wird durch die deutsche/europäische Kultur unterstützt. Der typisch deutsche direkte Umgang (Kapitel 3.6) hilft, ohne Umwege, Probleme zu erkennen und gemeinsam sehr effizient an Lösungsstrategien zu arbeiten.

- Als weiteres typisches deutsches Charakteristikum ist das Wissen um Deutschland als Bildungsland relevant: Ohne nennenswerte Bodenschätze, ohne geographische vorteilsbehaftete Besonderheiten oder sonstige gesellschaftliche Vorteile (im Gegenteil sogar in Form abnehmender Population) ist Bildung Deutschlands

wichtigstes Gut. Lernen gehört damit für einen Großteil der Bevölkerung bis ins hohe Alter zur Kultur. Auch wenn der Bildungsetat im Vergleich zu vielen anderen, vor allen Dingen aber europäischen Ländern, nur im Durchschnitt liegt [13], sind die Bildungsaktivitäten und die daraus resultierende Offenheit gegenüber lebenslangem Lernen ein Vorteil gegenüber vielen anderen, vor allen Dingen im typischen Offshore-Bereich liegenden Ländern.

- Neben der Durchführung der einzelnen Iterationen hängt der Erfolg agiler Methoden vor allen Dingen von einem systematischen und geschickten Entwurf der Iterationen ab. Diese Architektur-Aufgabe, die mittels klassischer Dekomposition etwas Großes in Teile zerlegt, einzeln behandelt, und anschließend wieder zum Ganzen zusammenfügt, weist viele Parallelen zu anderen klassischen Ingenieursdisziplinen auf, wie etwa dem Maschinenbau oder dem Automobilbau.

- Agile Methoden können also logische Folgerung eines möglichst effizienten und effektiven Projektmanagements zur Unterstützung der Zielvorgabe nach einem echten Software Engineering verstanden werden. Die jeweils auf der Ebene des Paradigmas und der Disziplin vorgebrachten Distinktionsmerkmal lassen sich daher ohne weiteres auf agile Methoden in Form inkrementeller, lernender und unmittelbarer Entwicklungsprozesse projizieren.

4.6 Alternative Methoden

Selbstverständlich existiert eine große Anzahl alternativer Methoden auf der Ebene der Vorgehensmodelle. Ein guter Querschnitt möglicher alternativer Methoden, verbunden mit der Angabe, wie häufig die jeweilige Methode heute in Deutschland Anwendung findet, ist im Kontext einer im Kontext des BITKOM-Arbeitskreises »Softwareentwicklungsprozesse und Werkzeuge« in Zusammenarbeit mit der Technischen Universität München, der Software Quality Systems AG und der Fachgruppe Vorgehensmodelle der Gesellschaft für Informatik (GI e.V.) durchgeführten Online-Befragung entstanden:

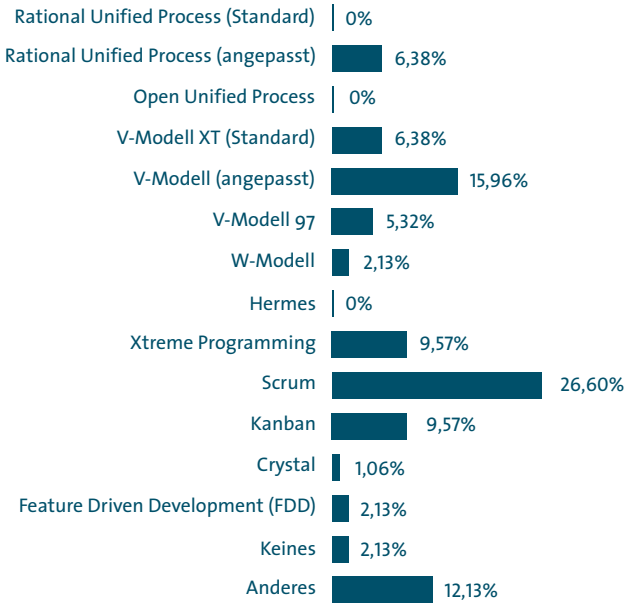


Abbildung 6: Alternative Vorgehensmethoden und ihre Verbreitung (n=39 für die Frage: »Welche(r) der folgenden Entwicklungsprozesse findet/n bei Ihnen Anwendung?«)

Auch wenn die agilen Methoden (z. B. Scrum und Xtreme Programming) bereits eine große Verbreitung haben dominieren immer noch die klassischen Methoden wie das V-Modell oder der RUP. Auch sie können selbstverständlich ins Projektmanagement für ein echtes Software Engineering integriert werden, allerdings scheinen die in Kapitel 4.5 aufgeführten Vorteile in vielen Fällen zu dominieren. Nichtsdestotrotz gibt es insbesondere hoch regulatorische Bereiche oder auch Bereiche, in denen insbesondere die unmittelbare Kommunikation nur bedingt ihre Vorteile ausspielen können (wenn z. B. die Spezifikation formal präzise und vollständig existiert), in denen auch die anderen Methoden ihre Daseinsberechtigung haben, jeweils aber erfolgreich nur im Zusammenspiel mit einem guten Projektmanagement.

Neben einigen Regulatorien gibt es auch technische Vorbedingungen von Agilität, die bei Nicht-Erfüllung alternative Methoden empfehlen: Monolithische Systeme lassen sich z. B. kaum mit kleinen agilen Teams im Wochenrhythmus verändern: Hier bedarf es der bewussten Integration ausgewiesener Experten, die nicht agil über Grob- und Feinentwurf eine anstehende Änderung »schwergewichtig« planen, entwerfen und durchführen.

5 Roadmap: Agiles Software Engineering in Deutschland

Es scheint sinnvoll, jegliche Softwareentwicklung zumindest entlang der drei folgenden Ebenen zu klassifizieren:

- Das Paradigma legt die Grundeinstellung und die fundamentale Sichtweise auf die Softwareentwicklung fest.
- Die Disziplin definiert und ordnet die Wissensbereiche und liefert somit etablierte Wissenskataloge, die entlang des Paradigmas relevant sind.
- Die Methoden schließlich beschreiben die konkreten Vorgehensweisen, die im Rahmen der jeweiligen Disziplin die Software entstehen lassen.

Es ist ein wichtiges Ergebnis dieses Leitfadens, dass jede Ebene »besetzt« sein muss, um Softwareentwicklung erfolgreich gestalten zu können: Methoden ohne eine entsprechende Disziplin sind demnach ebenso gefährlich einzusetzen wie Disziplinen ohne ein Paradigma, da unklar ist, welche Motivation die Disziplinen besitzen.

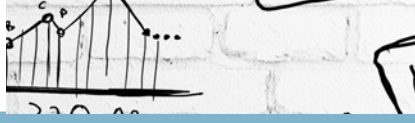
Neben der puren Besetzung der drei Ebenen müssen die einzelnen Ebenen allerdings auch kompatibel zueinander sein. Wenn ein Software-Projekt, wie dies z. B. im wissenschaftlichen Kontext durchaus üblich ist, eher explorativ durchgeführt wird, macht ein restriktives Projektmanagement nur bedingt Sinn. Ebenso ist ein strikt sequentielles Wasserfallmodell nur bedingt dafür geeignet, in ein echtes Projektmanagement integriert zu werden, da die Steuerpunkte und Feedback-Schleifen auf operativer Ebene fehlen.

Speziell für den Standort Deutschland scheint es pro Ebene eine ideale Vorbesetzung zu geben: Auf der Ebene des Paradigmas scheint diese historisch begründet (»Deutsches Ingenieurstum«), auf der Ebene der Disziplinen kulturell verankert (»Deutsche planen und steuern innerhalb wohldefinierter Leitplanken«) und auf der Ebene der Methode kausal und lokal begründet (»Logische Folgerung, die zudem lokale Besonderheiten perfekt abdeckt«). Dieser Leitfaden empfiehlt konkret die folgenden drei Vorbesetzungen:

- Paradigma: Softwareerstellung sollte als Ingenieurdisziplin aufgefasst werden, d.h. sie ist zielorientiert, systematisch und arbeitsteilig mit den übergeordneten Zielen gute Qualität, geringe Kosten und geringe Zeit.
- Disziplin: Das Denken in und das Durchführen von Projekten obliegt dem Projektmanagement, das mit seinen unterschiedlichen Wissensfacetten (z. B. die 10 Wissensbereiche des PMBoK), das die organisatorischen und prozessoralen Vorgaben für Softwareentwicklung á la Software Engineering liefert.
- Methode: Das Verwenden agiler Methoden, also inkrementeller, lernender und unmittelbarer Vorgehen, passt sich hervorragend in das Projektmanagement ein, da es feingranulare Steuerungspunkte, feedbackorientiertes Lernen und objektive, faktenorientierte Kommunikation ermöglicht.

Damit scheint agile Softwareentwicklung in einem neuen Licht: Agilität nicht als anarchisches, unsteuerbares, unplanbares und basisdemokratisches Vorgehen, sondern als perfekte Integration in ein professionelles Projektmanagement, um gemeinsam echtes Software Engineering zu ermöglichen. Agiles Software Engineering Made in Germany ist damit eine zukunftsfähige Kombination für Softwareentwicklung Made in Germany.

Der Vollständigkeit sei an dieser Stelle allerdings darauf hingewiesen, dass auch andere Kombinationen zielführend sind. Im hochregulativen Bereich mit fest vorgegebenen Spezifikationen zeigt sich z. B. durchaus auch das Paradigma schon anders, da hier viele Werkzeuge und Prozesse schlichtweg vorgegeben sind und damit nicht



mehr bzgl. ihrer Ziel-Adäquatheit evaluiert werden müssen. Auch die Steuerung auf der Ebene der Disziplin muss in solchen Projekten weniger das frühe Feedback der Nutzer einholen, sondern eher strikt nach langfristigem Plan abarbeiten. Solche Teilbereiche lassen sich in großen Projekten immer wieder identifizieren und sind damit gleichzeitig Möglichkeiten des Global Delivery, d.h. der bewussten Nutzung kostengünstigerer Ressourcen.

6 Literaturverzeichnis

[1] G. V. (Herausgeber), »Gabler Wirtschaftslexikon,« 2013. [Online]. Available: http://wirtschaftslexikon.gabler.de/Archiv/8033/paradigma-v9.html

[2] H. Balzert, Lehrbuch der Software-Technik, Heidelberg: Spektrum Akademischer Verlag, 2001.

[3] A. Abran, P. B. James W. Moore, R. Dupuis und L. L. Tripp, Guide to the Software Engineering Body of Knowledge, IEEE, 2004

[4] ISTQB/GTB, »Standardglossar der Testbegriffe, Deutsch/Englisch,« 30 September 2010. [Online]. Available: http://www.german-testing-board.info/downloads/pdf/CT_Glossar_DE_EN_V21.pdf

[5] F.-J. Drees, »Made in Germany,« Export- und Zollpraxis kompakt, pp.
verfügbar unter: http://www.zoll-export.de/wmhtml/wa4740/downloads/made_in_germany.pdf, 2012

[6] C. Rossbach und B. W. (. Berger), »Survival of the fittest: Wie Europa in der Cloud eine führende Rolle übernehmen kann,« Roland Berger Strategy Consultants, verfügbar unter: http://www.rolandberger.de/media/pdf/Roland_Berger_Cloud_Ecosystem_D_20111130.pdf, 2011

[7] G. M. Falb, »Die Heuristik - eine Findestrategie,« Methodenwerkstatt von Mag. Dr. Andrea Payrhuber, Universität Wien, 2006. [Online].
Available: http://homepage.univie.ac.at/andrea.payrhuber/methodenwerkstatt/handout_falb.pdf

[8] E. I. 9000:2005, Qualitätsmanagementsysteme – Grundlagen und Begriffe, Berlin: Beuth Verlag, 2005

[9] A.-G. C.-C. C. Patrick Schmidt, American openness... German objectivity,
verfügbar unter: http://www.agcc.de/media/Chicago%20speech2.pdf, 2003.

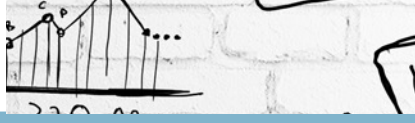
[10] P. Schmidt, »Bridging the Intercultural Gap: Non-conventional Truths about American-German Business,« Trade Magazine der German American Chamber of Commerce Inc.,
verfügbar unter: http://www.agcc.de/media/Non-Conventional%20Truths.pdf, 2004.

[11] W.-G. Bleeck und H. Wolf, Agile Software Entwicklung, Heidelberg: dPunkt-Verlag, 2008.

[12] R. Westphal, »One Man Think Tank Gedanken,« Agil persönlich definiert,
verfügbar unter: http://blog.ralfw.de/2012/08/agil-personlich-definiert.html, 2012.

[13] N. Pröpper, Agile Techniken für klassisches Projektmanagement – Qualifizierung zum PMI-ACP, mitp-Verlag, 2012.

[14] M. G. Schmidt, »Ausgaben für Bildung im internationalen Vergleich,« Beilage zur Wochenzeitung DAS PARLAMENT: Aus Politik und Zeitgeschichte, verfügbar unter: http://www.bpb.de/system/files/pdf/J5B0W9.pdf, 2003



7 Danksagung

Besonderer Dank gilt dem BITKOM-Arbeitskreis Software Engineering, insbesondere den Autoren des Leitfadens:

- Wolfgang Dorst, Kompetenzbereichsleiter | BITKOM e.V.
  Motivation, Organisation
- Dr. Frank Simon, Arbeitskreisleiter | Bluecarat AG Idee, Synchronisation der Fachbeiträge, roter Faden, Struktur, Roadmap
- Dr. Marco D'Onorio De Meo | Kienbaum Management Consultants GmbH
  Paradigma Erweiterungen/Vorbedingungen, Projektrisiken auf Methodenebene
- Stefan Luckhaus | Pass Consulting Group
  Qualitätsmanagement, Engineering-Ziele, Software-Metrie
- Uwe Tewes | Gemalto GmbH
  Software Qualität, Agile Methoden, Trends, Alternative Vorgehen
- Dr. Christian Schneider | Yatta Solutions GmbH
  Software Engineering, Werkzeugaspekte
- Matthias Gärtner | CDI Concepts Development Integration AG
  Projektmanagement, agile Methoden
- Dr. Marco Kuhrmann | Technische Universität München
  Gesamtes Review, Konsistenzprüfung, Hervorhebung des roten Fadens.


PRODUCT

Der Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. vertritt mehr als 2.000 Unternehmen, davon über 1.200 Direktmitglieder mit etwa 140 Milliarden Euro Umsatz und 700.000 Beschäftigten. Hierzu gehören fast alle Global Player sowie 800 leistungsstarke Mittelständler und zahlreiche gründergeführte, kreative Unternehmen. Mitglieder sind Anbieter von Software und IT-Services, Telekommunikations- und Internetdiensten, Hersteller von Hardware und Consumer Electronics sowie Unternehmen der digitalen Medien und der Netzwirtschaft. Der BITKOM setzt sich insbesondere für eine Modernisierung des Bildungssystems, eine innovative Wirtschaftspolitik und eine zukunftsorientierte Netzpolitik ein.

BITKOM

Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.

Albrechtstraße 10 A
10117 Berlin-Mitte
Tel.: 030.27576-0
Fax: 030.27576-400
bitkom@bitkom.org
www.bitkom.org