**Sylvia Bilek**

# Geschäftsprozessanalyse und Konzeption zur Prozessverbesserung unter Verwendung agiler Methoden und Praktiken bei einer mittelständischen Internetagentur

DIPLOMARBEIT

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

Mathematik/Physik/Informatik

Mittweida, 2009

**Sylvia Bilek**

# Geschäftsprozessanalyse und Konzeption zur Prozessverbesserung unter Verwendung agiler Methoden und Praktiken bei einer mittelständischen Internetagentur

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

Mathematik/Physik/Informatik

Mittweida, 2009

Erstprüfer: Prof. Dr.-Ing. Wilfried Schubert
Zweitprüfer: Prof. Dr.-Ing. Mario Geißler

Vorgelegte Arbeit wurde verteidigt am: 18.12.09

Bibliographische Beschreibung:

Sylvia Bilek:
Geschäftsprozessanalyse und Konzeption zur Prozessverbesserung unter Verwendung agiler Methoden und Praktiken bei einer mittelständischen Internetagentur – 2009. – 87 Seiten
Mittweida, Hochschule Mittweida – University of Applied Sciences,
Fachbereich Mathematik/Physik/Informatik, Diplomarbeit, 2009

Referat:

Ziel dieser Diplomarbeit ist es, zunächst die Ist-Prozesse in der Internetagentur der tro:net GmbH zu analysieren, zu dokumentieren sowie Probleme zu identifizieren. Im nächsten Schritt werden die Prozesse unter Verwendung agiler Methoden und Praktiken verbessert. Dazu sind Anforderungen an eine agile Methode zu formulieren und vorhandene Methoden hinsichtlich dieser Anforderungen zu bewerten. Nach Identifizierung einer geeigneten Methode muss diese noch an die Gegebenheiten der Internetagentur angepasst werden.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

ARIS – Architektur integrierter Informationssysteme

BPMN – Business Process Modeling Notation

BPMS – Business Process Management System

EPK – Ereignisgesteuerte Prozesskette

eEPK – erweiterte Ereignisgesteuerte Prozesskette

FDD – Feature Driven Development

# Abbildungsverzeichnis

# 1 Einleitung

Im Laufe des letzten Jahrzehnts gab es in der Softwareentwicklung einen zunehmenden Wandel weg von den traditionellen monumentalen hin zu den leichtgewichtigen agilen Methoden. Bereits 56% der Unternehmen nutzen agile Methoden in allen ihren Projekten bzw. in einem Teil davon, haben einige agile Praktiken übernommen oder starten erste Pilotprojekte. (vgl. [12])

Die Vorteile der agilen gegenüber den monumentalen Methoden liegen laut ihrer Begründer auf der Hand: eine wesentlich geringere Bürokratie durch das Verfassen nur weniger Dokumente, dafür bereits eine frühe Entwicklung funktionsfähiger Software und daraus resultierend eine größere Flexibilität bei Änderungswünschen, da in Iterationen geplant wird. Ferner wird eine größere Zufriedenheit des Kunden erreicht, der mehr Einfluss auf den Entwicklungsprozess nehmen kann. Die Risiken, dass ein Projekt verspätet beendet, das Budget überschritten oder zu wenig Funktionalität ausgeliefert wird, sinken somit.

Laut dem Chaos Report 2009 der Standish Group, der jährlich Erfolgsfaktoren in IT-Projekten untersucht, werden nur 32% aller Projekte ohne Budget- und Zeitüberschreitung und mit der vereinbarten Funktionalität fertig gestellt. 44% der Projekte werden mit Budget-, Zeitüberschreitung und/oder weniger Funktionalität als gefordert, beendet. Die restlichen 24% werden entweder vor Fertigstellung storniert oder ausgeliefert, aber nie vom Kunden genutzt. [13]

Insgesamt gibt es in der IT-Branche vier Trends (vgl. [2] S. 684f):

- Abmagerung traditioneller monumentaler Modelle
- Erweiterung agiler Modelle
- Schaffung hybrider Modelle aus monumentalen und agilen Modellen (z.B. V-Modell XT mit Extreme Programming)
- Kombination monumentaler und agiler Modelle mit Rahmenmodellen (z.B. Scrum und CMMI)

Neben Nutzern von monumentalen Modellen gibt es auch Unternehmen, die ein gewachsenes Modell nutzen, d.h. es wurde nie offiziell ein Modell eingeführt, aber aufbauend auf den Erfahrungen der Mitarbeiter und den Erfahrungen aus abgeschlossenen Projekten hat sich eine bestimmte Arbeitsweise entwickelt. Abgeschlossene Projekte bilden die Grundlage für weitere Projekte, wo Dokumente und Arbeitsweise dann entsprechend den neuen Erfordernissen angepasst werden. Eine Dokumentation vorhandener Prozesse gibt es meistens nicht.

## 1.1 Unternehmen

Die *tro:net Internet Systemhaus GmbH* mit Sitz in Troisdorf ist ein solches Unternehmen. Ihr Geschäftsbereich efruits Internetagentur ist seit 15 Jahren in der Online-Medienproduktion und im begleitenden Online-Marketing für Unternehmen tätig. Zu ihren Produkten zählen Internetauftritte, Online-Shops, Kundenportale, Mitarbeiter-Intranets und E-Mail-Newsletter. Das Leistungsspektrum umfasst dabei Kreativleistungen wie Medienkonzeption, Text und Gestaltung sowie die technische Umsetzung von der HTML-Codierung, über Softwareintegration und individuelle Anwendungsentwicklung bis hin zum Server-Hosting und Management. Für die technische Umsetzung werden größtenteils die Standardsoftware Typo3 Web CMS und OXID eShop genutzt.
Zu den Kunden der tro:net gehören vorwiegend mittelständische Unternehmen, Vereine, Verbände und behördliche Institutionen. Pro Jahr werden 30-50 Projekte zu je 5-50 Manntagen durchgeführt.

Da die efruits Internetagentur ausschließlich im Online-Geschäft tätig ist und in dieser schnelllebigen Welt das Time-to-Market möglichst kurz sein muss, bieten sich agile Methoden für die Prozessverbesserung an.

## 1.2 Ziele

Nahezu alle Vorgehensmodelle der agilen Softwareentwicklung sind auf die reine Anwendungsentwicklung zugeschnitten. In der Webentwicklung sind jedoch noch andere Faktoren wesentlich, z.B. spielt das Layout eine wichtigere Rolle als das GUI-Design bei reinen Anwendungssystemen, denn der Internetauftritt soll Informationen bereitstellen und/oder als Verkaufsplattform dienen. Informationen oder Berichte, wie agile Methoden konkret für die Webentwicklung eingesetzt werden können, finden sich in der Fachliteratur und im Internet allerdings nur wenige. Daher ist ein Ziel dieser Arbeit, herauszufinden, wie agile Methoden oder Praktiken an die Webentwicklung angepasst werden können.

Bezogen auf das Unternehmen tro:net sind die bestehenden Geschäftsprozesse in der Internetagentur zu analysieren und Systematiken und Lücken der Prozesse zu dokumentieren. Auf Basis dieser Analyse sollen Optimierungsmöglichkeiten der Prozesse mit Hilfe moderner agiler Softwareentwicklungsmethoden aufgezeigt werden. Darüber hinaus soll ein Werkzeug empfohlen werden, mit dem die Prozesse abgebildet, dokumentiert und verfolgt werden können.

Der Schwerpunkt der Betrachtung liegt dabei auf der technischen Entwicklung. Dazu gehören die Einrichtung von Standardsoftware, Technische User Interface Entwicklung (HTML-Template-

Codierung), Entwicklung von Erweiterungs- und Schnittstellenmodulen zu Standardsoftware sowie die individuelle Anwendungsentwicklung.
Die Darstellung der Prozesse soll den Zeitraum zwischen dem Erstkontakt mit dem Kunden und der Auslieferung umfassen. Folgeleistungen für die tro:net wie z.B. Serverhosting/-management und Pflege sind kein Bestandteil dieser Diplomarbeit. Schnittstellen zu Kreativleistungen (Kommunikationskonzeption, Gestaltung, Text) finden Berücksichtigung in den Prozessbeschreibungen, sind aber ansonsten ebenfalls kein Bestandteil.

## 1.3 Vorgehensweise

Da die Analyse der aktuellen Geschäftsprozesse die Grundlage für Prozessverbesserungen darstellt, werden im **zweiten Kapitel** zunächst drei Möglichkeiten zur Modellierung von Geschäftsprozessen vorgestellt. Eingegangen wird dabei auf *Ereignisgesteuerte Prozessketten (EPK)/erweiterte Ereignisgesteuerte Prozessketten (eEPK), Aktivitätsdiagramme, Business Process Modeling Notation (BPMN)* sowie deren Vor- und Nachteile.

Für die Verbesserung der Prozesse in der Internetagentur sollen agile Methoden und Praktiken genutzt werden. Daher werden im **dritten Kapitel** die Geschichte und die Ziele der agilen Softwareentwicklung und der Unterschied zu monumentalen Methoden erklärt sowie anschließend drei der bekanntesten agilen Methoden vorgestellt: *Extreme Programming (XP)*, *Scrum* und *Feature Driven Development (FDD)*.

Im **vierten Kapitel** werden ausgehend von einer Mitarbeiterbefragung die Prozesse und Strukturen in der Internetagentur analysiert und die Prozesse mit Hilfe der im zweiten Kapitel vorgestellten Modellierungsmethoden für Geschäftsprozesse dargestellt. Die in dieser Analyse identifizierten Probleme sind im Weiteren bei der Auswahl einer agilen Methode zu berücksichtigen.

Auf Basis der gewonnenen Informationen werden im **fünften Kapitel** Anforderungen an eine agile Methode definiert und mit den im dritten Kapitel vorgestellten Methoden verglichen. Anschließend wird eine geeignete agile Methode identifiziert und gegebenenfalls angepasst bzw. eine Methode aus bewährten agilen Praktiken zusammengesetzt.

Im **sechsten Kapitel** wird anhand eines kleinen Typo3-Projekts die praktische Anwendung der im fünften Kapitel erarbeiteten Anforderungsanalyse und des Feature-Aspekts gezeigt.

Zum Abschluss wird im **siebten Kapitel** das Ergebnis der Arbeit bewertet und es werden Hinweise für eine stufenweise Einführung der Lösung sowie ein Ausblick auf mögliche Anschlussprojekte und ggf. noch offene Probleme gegeben.

# 2 Geschäftsprozessmodellierung

Um die Ist-Prozesse der Internetagentur grafisch darstellen zu können, werden in diesem Kapitel Möglichkeiten zu ihrer Modellierung vorgestellt sowie deren Vor- und Nachteile betrachtet.

Die Geschäftsprozessmodellierung umfasst die beiden Aspekte Prozessanalyse und Prozessgestaltung. In der Prozessanalyse werden die Ist-Prozesse dargestellt. In der Prozessgestaltung wird dagegen versucht, die gewünschten Soll-Prozesse zu modellieren. „Ein Geschäftsprozess umfasst eine bestimmte Menge von Tätigkeiten, die zur Erfüllung einer betrieblichen Aufgabe notwendig sind und in einer vorgegebenen Ablauffolge zu erledigen sind“ ([5] S. 310). Sie können mit verschiedenen grafischen Notationen abgebildet werden.

Nachfolgend werden die drei verbreitetesten Möglichkeiten zur Darstellung von Geschäftsprozessen vorgestellt: *Ereignisgesteuerte Prozesskette/erweiterte Ereignisgesteuerte Prozesskette* und *Business Process Modeling Notation* sowie der UML2-Standard zur Darstellung von *Aktivitätsdiagrammen*.

## 2.1 Ereignisgesteuerte Prozesskette/erweiterte Ereignisgesteuerte Prozesskette

### 2.1.1 Ereignisgesteuerte Prozesskette

Die *Ereignisgesteuerte Prozesskette (EPK)* wurde unter Prof. Scheer an der Universität des Saarlandes speziell für die Darstellung von Geschäftsprozessen im Rahmen des ARIS-Konzepts entwickelt und 1992 in einer Schrift [15] veröffentlicht.

Die EPK enthält die drei Elemente *Ereignis*, *Funktion* und *Informationsobjekt* sowie die logischen Verknüpfungen *And*, *Or* und *Xor*. Ein *Ereignis* beschreibt einen Zustand. Eine *Funktion* stellt einen Vorgang dar, mit dem ein Zustand in einen anderen überführt wird. Mit einem *Informationsobjekt* wird ein Gegenstand der realen Welt abgebildet.

Für die Darstellung eines Geschäftsprozesses mit EPK gelten folgende Regeln:

- Der Geschäftsprozess beginnt und endet mit Ereignissen.
- Ereignisse und Funktionen müssen sich abwechseln.

Da diese wenigen Elemente nicht ausreichen, um komplexere Prozesse darzustellen, wurde die EPK um neue Elemente erweitert.

### 2.1.2 Erweiterte Ereignisgesteuerte Prozesskette

Diese *erweiterten Ereignisgesteuerten Prozessketten (eEPK)* enthalten zusätzlich zu den Elementen der EPK noch Elemente, um Stellen, Organisationseinheiten und Teilprozesse. darzustellen. Zum Teil wird die eEPK inzwischen auch einfach nur noch als EPK bezeichnet (siehe z.B. [10]).

Die Darstellung der einzelnen Elemente ist nicht einheitlich. In unterschiedlichen Programmen werden teilweise unterschiedliche Darstellungen für ein und dasselbe Element verwendet. Auch ist die Menge an Elementen nicht immer gleich. In manchen Programmen können Klassen, Datenbanken etc. mit eingefügt werden, andere beschränken sich auf die klassischen Elemente.

Folgendes Beispiel zeigt eine eEPK für den Prozess der Kreditbearbeitung.

**Legende**

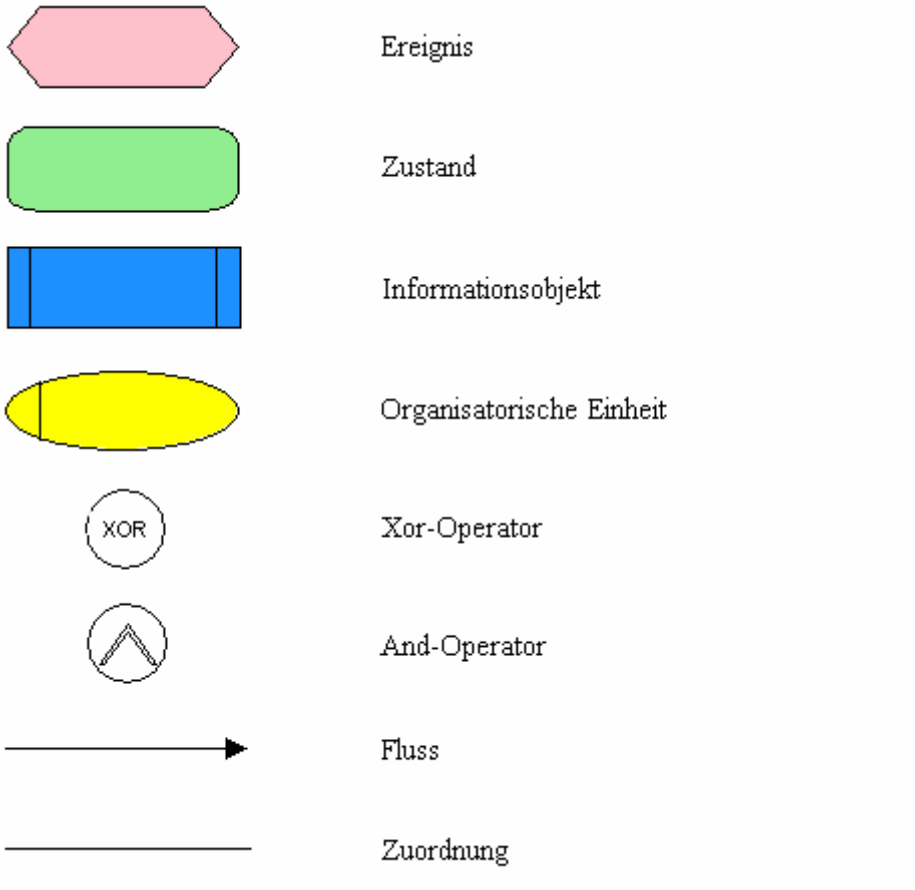


Abbildung 2.1: Elemente der eEPK

**Beispiel**

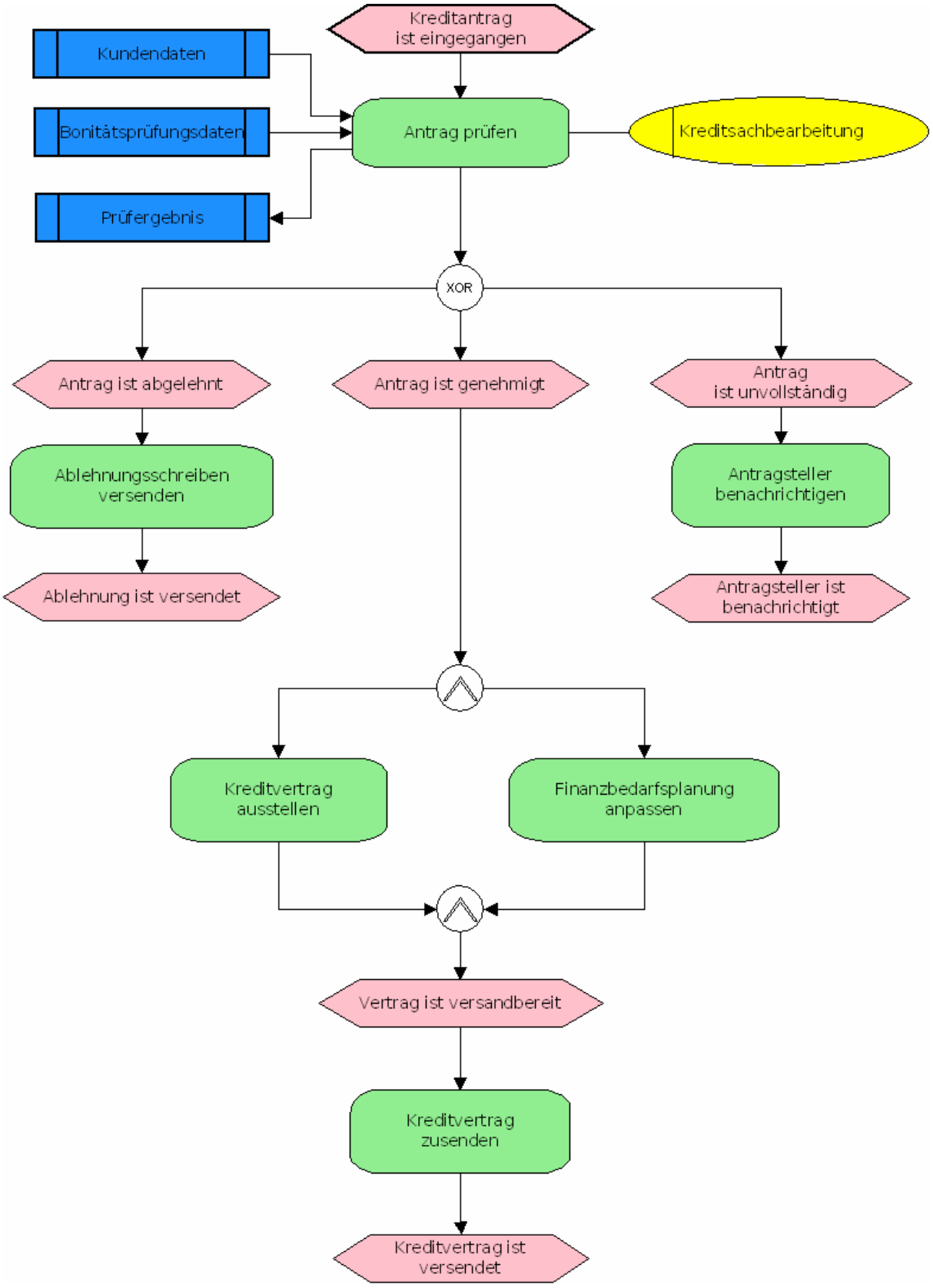


Abbildung 2.2: Kreditbearbeitung als eEPK nach [5] S. 337, erstellt mit Dia

In dem in Abbildung 2.2 dargestellten Beispiel geht es um die Bearbeitung eines Kredits. Ein Kreditantrag geht bei der Kreditsachbearbeitung ein. Für die Prüfung des Antrags werden Kundendaten und Bonitätsprüfungsdaten benötigt und anschließend ein Prüfergebnis generiert. Als Ergebnis dieser Prüfung tritt genau eins von drei Ereignissen ein:

- Wird der Antrag abgelehnt, wird ein Ablehnungsschreiben versendet und der Prozess endet.
- Ist der Antrag unvollständig, wird der Antragsteller benachrichtigt und der Prozess endet.
- Bei einer Genehmigung des Antrags, müssen die beiden Aufgaben „Kreditvertrag ausstellen“ und „Finanzbedarfsplanung anpassen“ erledigt werden. Anschließend wird der Kreditvertrag zugesendet und der Prozess endet.

## 2.2 Aktivitätsdiagramm

Das Aktivitätsdiagramm ist eines von insgesamt 13 Diagrammen der UML. Ursprünglich ist diese Diagrammart für die Modellierung von Programmflüssen eingeführt wurden. Aktivitätsdiagramme sind also in erster Linie für den Entwurf von Programmen gedacht. Allerdings lassen sie sich auch für die Modellierung von einfachen Geschäftsprozessen verwenden, da mit ihnen der Ablauf von Aufgaben, Verzweigungen sowie verschiedene Organisationseinheiten dargestellt werden kann.

Ausgehend von einem oder mehreren *Startknoten* wird eine Reihe von Aktivitäten bis zu einem *Endknoten* durchlaufen. Zu den wichtigsten Elementen gehören *Aktionsknoten* und *Kontrollknoten*. Bei Aktionsknoten wird eine bestimmte Aktion/Aufgabe durchgeführt. Kontrollknoten lassen sich in *And-* und *Or-Knoten* unterteilen. Daneben gibt es noch Symbole für technisch ausgerichtete Diagramme, beispielsweise zum Anzeigen beteiligter Klassen, für Ein- und Ausgabeparameter, Objekte u.a. Eine Übersicht zu sämtlichen Elementen gibt das Dokument [16]. Um verschiedene Organisationseinheiten darzustellen, kann das Diagramm außerdem vertikal in mehrere *Bahnen* (auch *Swimlanes* genannt) aufgeteilt werden (vgl. [7] S. 70ff).

Abbildung 2.4 zeigt die Kreditbearbeitung als Aktivitätsdiagramm.

**Legende**

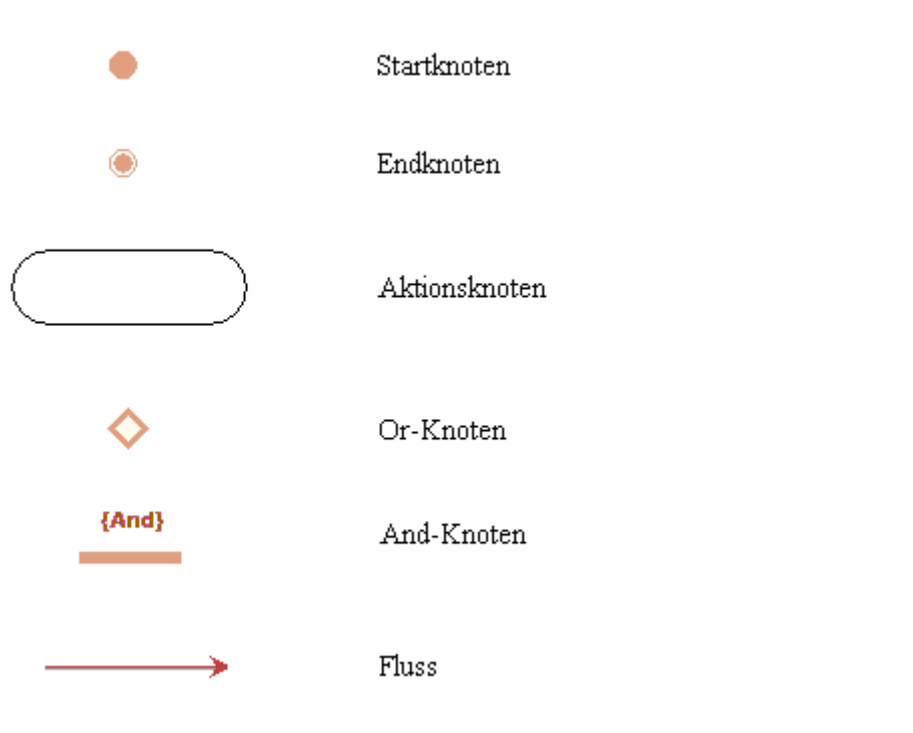


Abbildung 2.3: Elemente des Aktivitätsdiagramms

**Beispiel**

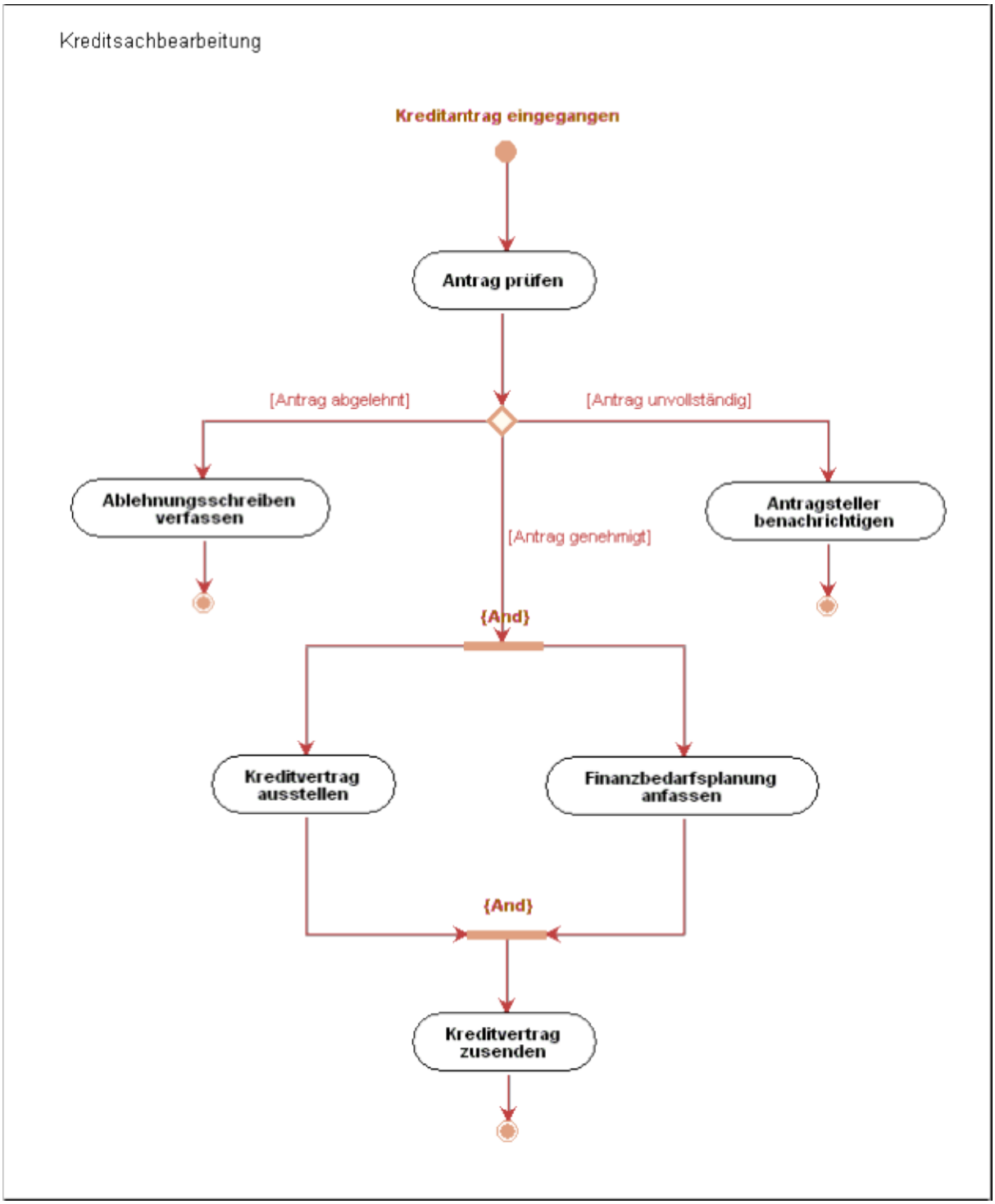


Abbildung 2.4: Kreditbearbeitung als Aktivitätsdiagramm, erstellt mit objectiF 7.0

Abbildung 2.4 zeigt das bekannte Beispiel der Kreditbearbeitung. Die Organisation wird hier durch die Swimlane „Kreditsachbearbeitung“ dargestellt. In Aktivitätsdiagrammen gibt es keine Ereignisse, sondern nur Aktionen/Aufgaben. Bei Entscheidungsknoten können die ausgehenden Pfade beschriftet werden und erfüllen damit die gleiche Funktion wie die Ereignisse in den EPK/eEPK. Denkbar wäre hier noch eine zweite Swimlane „System“, welche die Verarbeitung der Kundendaten und Bonitätsprüfungsdaten sowie die Generierung des Prüfergebnisses durch ein Computersystem darstellt.

## 2.3 Business Process Modeling Notation

Die *Business Process Modeling Notation (BPMN)* dient zur grafischen Darstellung von Prozessen und ist dabei sowohl für fachliche als auch technische Modelle geeignet. Mit BPMN lassen sich Modelle nicht nur darstellen, sondern auch mit einer Process-Engine eines Workflow- oder Business Process Management Systems (BPMS) ausführen.

Die erste Version der BPMN-Spezifikation wurde 2004 von Stephen A. White (IBM) veröffentlicht. Aktuell liegt sie in der Version 1.2 vor (Stand Januar 2009, vgl. auch [11]).

Der Prozessablauf findet in einem *Pool* statt, dieser kann beispielsweise ein Unternehmen repräsentieren oder auch ein Computersystem. Ein Pool lässt sich außerdem in Swimlanes unterteilen, die z.B. Abteilungen oder andere Organisationseinheiten darstellen. In einem BPMN Diagramm können auch mehrere Pools verwendet werden, die miteinander kommunizieren.
BPMN enthält, verglichen mit anderen Notationen für Geschäftsprozesse, viele Elemente. Die Kernelemente sind in Abbildung 2.5 dargestellt. Von einigen dieser Elemente existieren verschiedene Ausführungen. Für nähere Informationen zu sämtlichen BPMN-Elementen, siehe [11] unter „BPMN Notation Graphics“ oder [17].

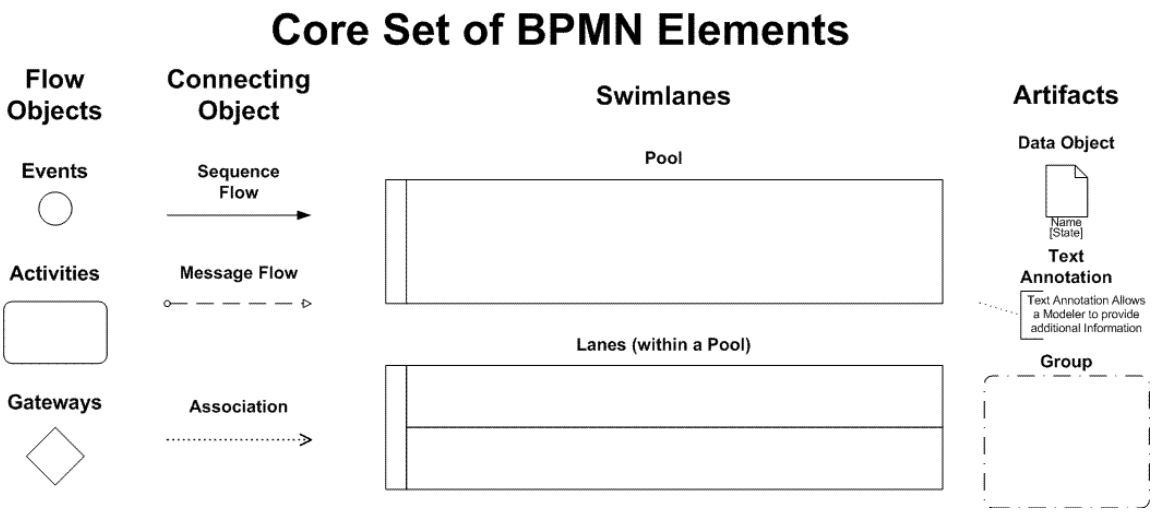


Abbildung 2.5: Kernelemente von BPMN [20]

Folgendes Beispiel stellt die Kreditbearbeitung als BPMN-Diagramm dar.

**Beispiel**

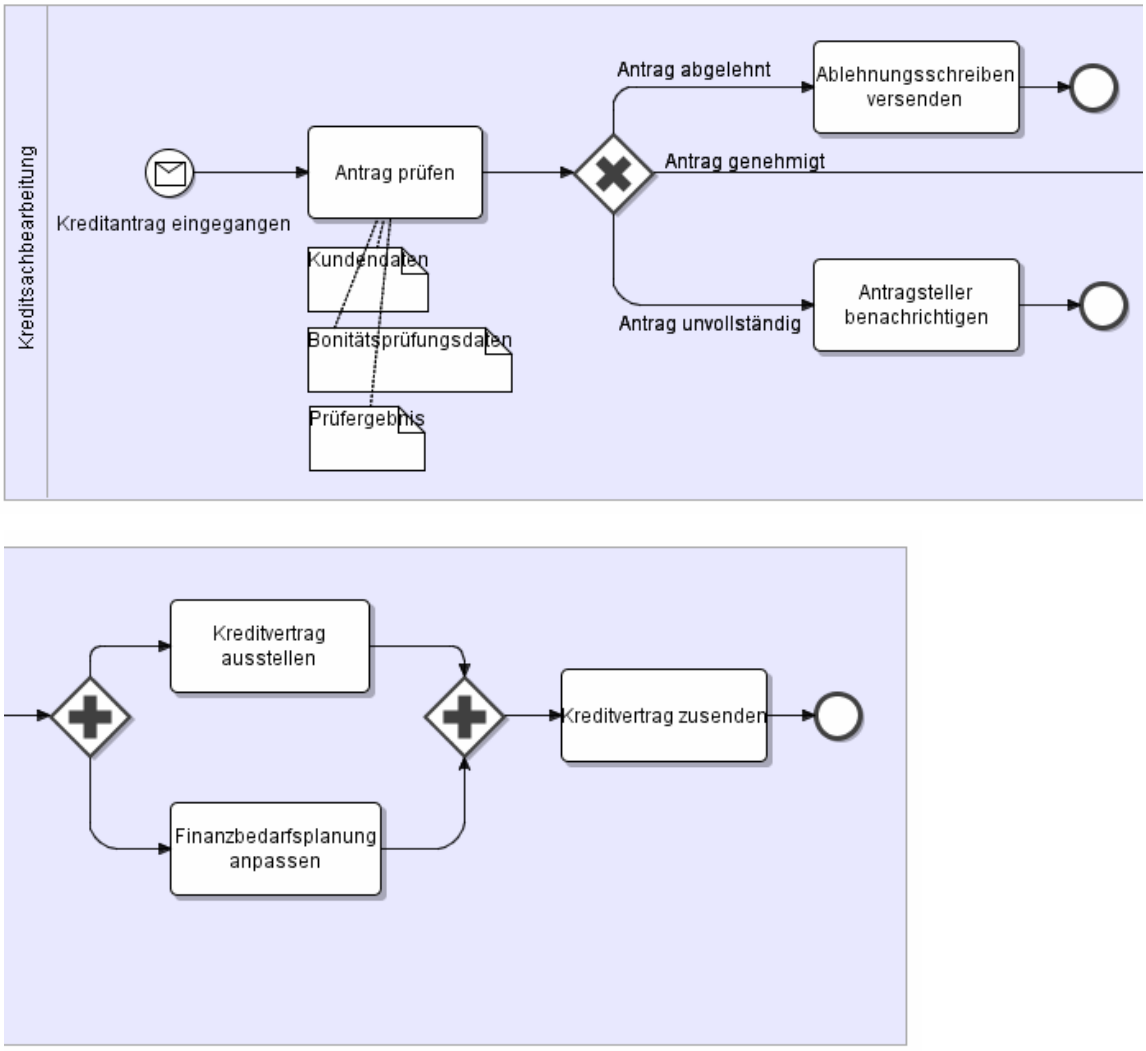


Abbildung 2.6: Kreditbearbeitung als BPMN, erstellt mit Intalio Designer

Abbildung 2.6 zeigt erneut die Kreditbearbeitung. Wie im Aktivitätsdiagramm gibt es auch in BPMN Swimlanes, nur sind sie im Allgemeinen horizontal statt vertikal ausgerichtet. Es stehen eine Reihe von Startereignissen zur Auswahl, im Beispiel zeigt das Symbol mit dem Brief eine eingehende Mitteilung, den Kreditantrag. Benötigte oder generierte Daten werden als Datenobjekte dargestellt, die mit der jeweiligen Aufgabe („Antrag prüfen“) verbunden werden. Verzweigungen werden über Rauten abgebildet, deren Symbol angibt, um was für eine Art es sich handelt. Das X steht für eine Xor-Verzweigung, das Pluszeichen für eine And-Verzweigung.

## 2.4 Vorteile und Nachteile der Darstellungsformen

EPK und BPMN sind beide gut für die Darstellung von Geschäftsprozessen geeignet.
Ein Vorteil der EPK ist, dass diese Darstellungsform wenige Symbole enthält und leicht verständlich ist. Von Nachteil ist allerdings, gerade bei großen Prozessen und der Darstellung von Arbeitsabläufen, der Zwang zum ständigen Wechsel von Ereignissen und Funktionen. Eine Abfolge von Arbeitsschritten wird so unnötig aufgebläht.
BPMN umfasst dagegen sehr viele Symbole, der anfängliche Lernaufwand zum Verstehen und Erstellen der Diagramme ist daher höher. Es lassen sich allerdings auch mehr Details darstellen. Außerdem steht mit der Community-Edition von Intalio eine kostenlose Software zur Verfügung, um BPMN-Diagramme zu erstellen und später auch als Workflow ausführen zu können.
Aktivitätsdiagramme sind auf Grund ihrer geringeren Menge an Symbolen eher zur Darstellung von einfachen Ablaufdiagrammen oder technischen Diagrammen geeignet.

Die in diesem Kapitel vorgestellten Modellierungsformen werden im vierten Kapitel zur Analyse der Geschäftsprozesse in der Internetagentur der tro:net benötigt. Im nächsten Kapitel sollen zunächst die Grundlagen der agilen Softwareentwicklung erläutert und konkrete Methoden vorgestellt werden, die später zur Prozessverbesserung eingesetzt werden.

# 3 Agile Softwareentwicklung

Bis heute ist nicht klar definiert, was agile Softwareentwicklung ist bzw. wann eine Methode agil genannt werden kann ([4] S. 114). Der Begriff „agil“, der soviel wie beweglich bedeutet, wurde jedoch nicht zufällig gewählt. Agile Methoden zeichnen sich vor allem durch Flexibilität aus, da sie davon ausgehen, dass sich Anforderungen während der Entwicklungszeit ändern können.
Dieses Kapitel beschäftigt sich mit den Zielen und Merkmalen agiler Softwareentwicklung und zeigt verschiedene Ausprägungen in Form konkreter agiler Methoden auf.

## 3.1 Geschichte und Zielsetzung

Die agile Softwareentwicklung stellt eine Gegenbewegung zu den oft schwerfälligen monumentalen Methoden dar. Erste Ideen gehen zurück bis in die 1990er Jahre, die von der Überlegung ausgehen, dass zu Beginn eines Softwareprojekts oft nicht klar ist, was der Kunde genau möchte. Monumentale Methoden sind zu starr, um auf Änderungen der Anforderungen reagieren zu können, die sich auf Grund eines besseren Verständnisses bei Kunden und Entwicklern im Laufe der Entwicklung ergeben. Daher liefern sie am Ende oft nicht das, was der Kunde braucht. Als Lösung für dieses Problem wurden „leichtgewichtige“ Methoden entwickelt, die den gesamten Softwareentwicklungsprozess flexibler gestalten.
Populär wurden agile Methoden erstmalig mit dem von Kent Beck 1999 veröffentlichten Buch „eXtreme Programming eXplained“.

Im Februar 2001 versammelten sich auf einer Konferenz in Utah 17 Begründer verschiedener agiler Methoden. Auf dieser Konferenz wurde auch der Begriff „agil“ geprägt. Vorher wurde als Gegensatz zu den traditionellen schwergewichtigen (engl. heavyweight) Methoden von leichtgewichtigen (engl. lightweight) Methoden gesprochen. Ihre Ziele fassten sie im sog. Agilen Manifest zusammen:

**Manifesto for Agile Software Development**

„We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

**Individuals and interactions** over processes and tools
**Working software** over comprehensive documentation
**Customer collaboration** over contract negotiation
**Responding to change** over following a plan

That is, while there is value in the items on
the right, we value the items on the left more." [9]

So unterschiedlich die Prozessabläufe und eingesetzten Praktiken agiler Methoden auch teilweise sind, gibt es Merkmale, die allen gemein sind:

- Iteratives Vorgehen
- Keine große Planungsphase zu Beginn
- Sich abwechselnde kurze Planungs- und Entwicklungszeiten
- Früher Einsatz funktionierender Software
- Häufiges Prüfen gegen die Anforderungen

## 3.2 Agile Methoden und Praktiken

Im Laufe der Zeit haben sich einige agile Methoden und Praktiken entwickelt. Für ein einheitliches Verständnis der beiden Begriffe „Agile Praktik" und „Agile Methode" sollen folgende Definitionen nach [3] dienen:

**Agile Praktik**
„Unter einer agilen Praktik verstehen wir eine etablierte Handlungsweise, in einem ausgewählten Ausschnitt oder Aspekt der Softwareentwicklung agil vorzugehen, also die agilen Werte zu berücksichtigen." ([3] S.18)

**Agile Methode**

„Unter einer agilen Methode verstehen wir eine konkrete benannte Zusammenstellung von agilen Praktiken.“ ([3] S.18)

Zu den drei bekanntesten und verbreitetesten Methoden zählen Extreme Programming (XP), Scrum und Feature Driven Development (FDD) (nach [2] S. 652). Daneben gibt es noch eine Vielzahl an anderen Methoden. Als Beispiele seien hier Crystal, Adaptive Software Development, Agile Unified Process, Dynamic Systems Development Method und Open Unified Process genannt. Da die drei erstgenannten Methoden ein weites Spektrum von sehr groben bis hin zu sehr detaillierten Prozessvorgaben und wenig bis hohe Eigendisziplin der Teams abdecken, werde ich mich im Folgenden auf diese beschränken.

## 3.2.1 Einordnung agiler Methoden

Abbildung 3.1 zeigt die Einordnung verschiedener Vorgehensmodelle nach den Kriterien Vorgehensflexibilität und Phasen-/Iterationsprinzip. Die schwergewichtigen Methoden sind eher funktionsorientiert mit langen Phasen und im Vorgehen sehr starr, d.h. sie können auf Änderungen während des Projekts nur schwer bis gar nicht reagieren. Dagegen sind die leichtgewichtigen agilen Modelle sehr flexibel und nutzen Iterationen statt Phasen, in denen die Software schrittweise entwickelt und verbessert wird.

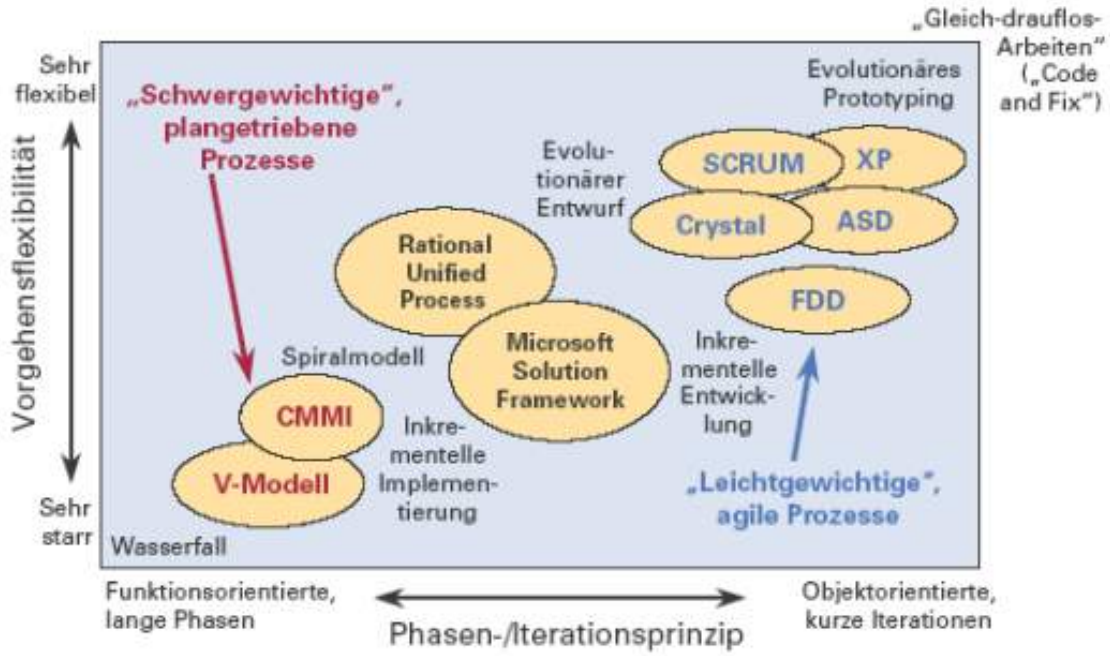


Abbildung 3.1: Einordnung von Vorgehensmodellen [21]

### 3.2.2 Extreme Programming

Der folgende Abschnitt enthält eine Zusammenfassung aus [2], [3] und [4].

Extreme Programming (XP) wurde 1999 von Kent Beck in seinem Buch „eXtreme Programming eXplained“ vorgestellt. Zu den Mitbegründern zählen außerdem Ward Cunningham und Ron Jeffries. 2004 hat Kent Beck nach Kritik an seinem ersten Entwurf noch einmal einige Änderungen an XP vorgenommen. Im Weiteren beziehe ich mich auf diese verbesserte Version.
XP setzt sehr stark auf sich selbst organisierende kleine Teams und eine hohe Kundenbeteiligung. Der Fokus liegt auf der Programmierung, daher ist diese Methode gut geeignet, wenn ein Team/Unternehmen Vorgehensweisen für die Implementierung und begleitende Tests benötigt. Innerhalb eines XP-Projekts existieren die im folgenden Unterkapitel erläuterten Rollen.

#### 3.2.2.1 Rollen

**Entwickler**
Der Entwickler ist verantwortlich für Entwurf, Programmierung, Aufwandsschätzung, Builds, Tests und die Kommunikation mit dem Kunden.

**Kunde**
Der Kunde hat die Aufgabe, Anforderungen passend aufzuschreiben und zu priorisieren. Er gibt jederzeit Auskunft zu Details der Anforderungen und Rückkopplung zu existierenden Systemversionen.

**Tracker**
Der Tracker ist zuständig für die Planung von Ressourcen an Mitarbeitern, Rechnern, Coaches etc. und berichtet dem Management.

#### 3.2.2.2 Ablauf

Der Ablauf eines XP-Projekts lässt sich grob in 2 Phasen unterteilen. Zu Beginn der Planungsphase schreibt der Kunde zunächst sämtliche Anforderungen an das System auf sogenannte Story Cards. Jede Story (auch Geschichte genannt) beschreibt dabei eine Anforderung in ca. 3 Sätzen in der Terminologie des Kunden.

Auf Basis der Stories erstellen die Entwickler eine prototypische Architektur in Form eines Klassendiagramms. Für jede Story werden in Zusammenarbeit mit dem Kunden Akzeptanztests geschrieben, die später prüfen sollen, ob die Anforderung korrekt umgesetzt wurde. Danach wird wiederum zusammen mit dem Kunden ein Release Plan erstellt. Die Stories werden von den Entwicklern auf ihre Realisierbarkeit hin überprüft, ihr Zeitaufwand abgeschätzt und vom Kunden priorisiert. Mit Hilfe der gewonnenen Informationen wird schließlich ein Projektplan erstellt. Der Kunde wählt die umzusetzenden Stories für das erste Release und die nächste Iteration.

Die zweite Phase läuft iterativ ab. Die Entwickler schreiben zunächst für die vom Kunden für diese Iteration ausgewählten Anforderungen Unit Tests und erst danach den erforderlichen Code. Es findet ein kontinuierliches Refactoring des Codes statt, da die Architektur so einfach wie möglich gehalten werden soll, d.h. es wird nur soviel Code geschrieben, wie für die Umsetzung der momentanen Anforderungen nötig ist. Eine Anforderung gilt erst dann als erledigt, wenn sie alle vorher geschriebenen Unit Tests besteht. Am Ende der Iteration werden Akzeptanztests durchgeführt und die Ergebnisse dem Kunden präsentiert. Auf Grundlage der Erfahrungen und Rückkopplungen des Kunden wird die Planung für die nächste Iteration bzw. das nächste Release vorgenommen. Hat der Kunde neue Anforderungen, werden diese wiederum geschätzt und priorisiert.

#### 3.2.2.3 Werte, Prinzipien und Praktiken

XP ist ein Vorgehensmodell, das sehr stark auf Werten, Prinzipien und Praktiken aufbaut. Die fünf Werte und zu beachtenden 13 Prinzipien werden im Folgenden aufgezählt.

**Werte**

- Einfachheit
- Kommunikation
- Rückkopplung
- Mut
- Respekt

**Prinzipien**

- Menschlichkeit:
  Soziale Bedürfnisse der Mitarbeiter müssen beachtet werden.
- Wirtschaftlichkeit:
  Projekte müssen wirtschaftlich sein.

- Selbstähnlichkeit:
  Es wird geprüft, ob erfolgreiche Lösungen auch anderweitig einsetzbar sind.
- Verbesserung:
  Es soll eine kontinuierliche Verbesserung stattfinden.
- Mannigfaltigkeit:
  Vielfalt wird als Chance begriffen.
- Reflexion:
  Die Teammitglieder denken ständig darüber nach, wie sie ihre Arbeit verbessern können.
- Fluss:
  Die Software verbessert sich im Projektverlauf kontinuierlich.
- Gelegenheit:
  Probleme werden als Chance begriffen.
- Redundanz:
  Im Entwicklungsprozess soll es Redundanzen geben, um Ausfälle auszugleichen.
- Fehlschlag:
  Fehlschläge werden akzeptiert.
- Qualität:
  Es muss Qualitätsarbeit abgeliefert werden.
- Babyschritte:
  Veränderungen werden in kleinen Schritten durchgeführt.
- Akzeptierte Verantwortung:
  Wer eine Story übernimmt, ist für Entwurf, Programmierung und Tests verantwortlich.

Neben Werten und Prinzipien werden Praktiken angewandt. Die 13 Primärpraktiken bilden die Basis.

**Primärpraktiken**

- Räumlich zusammen sitzen:
  Alle Beteiligten sitzen – idealerweise in einem Raum – nahe beieinander.
- Komplettes Team:
  Das Team besitzt die erforderliche fachliche Kompetenz und die notwendige Technik, um alle Anforderungen eigenständig umsetzen zu können.
- Informative Arbeitsumgebung:
  Informationen über den Fortschritt des Projekts sollen in der Arbeitsumgebung des Teams sichtbar sein, wie z.B. in Form von Ausdrucken an der Wand.
- Energiegeladene Arbeit:
  Alle Entwickler arbeiten engagiert.

- Programmieren in Paaren:
  Jeweils zwei Entwickler sitzen gemeinsam an einem PC, einer an der Tastatur, der andere daneben. Zwischen beiden Entwicklern findet dabei ein ständiger Dialog zu Umsetzung der Anforderung, Architektur, Bugfixing etc. statt. Die Plätze und die Programmierpartner werden regelmäßig getauscht.
- Stories:
  Eine Story ist eine vom Kunden aufgeschriebene Anforderung an das System.
- Wochenzyklus:
  Eine Iteration dauert eine Woche.
- Quartalszyklus:
  Ein Release dauert drei Monate.
- Freiraum:
  Die Mitarbeiter erhalten Zeit, um sich weiterzubilden und zu experimentieren.
- 10 Min Build:
  Das Erstellen einer Systemversion soll nicht länger als 10 Minuten dauern.
- Kontinuierliche Integration:
  Der Code wird regelmäßig in ein Versionskontrollsystem eingecheckt.
- Testgetriebene Entwicklung:
  Vor der Programmierung werden stets Tests geschrieben.
- Inkrementeller Entwurf:
  Es erfolgt ein schrittweiser Entwurf, basierend auf den jeweils aktuellen Anforderungen.

Sind alle Primärpraktiken erfolgreich im Team umgesetzt, schließen sich elf Folgepraktiken an, die auf den Primärpraktiken aufbauen.

**Folgepraktiken**

- Echte Kundenbeteiligung:
  Ein Kunde arbeitet mit dem Team vor Ort zusammen.
- Inkrementelle Auslieferung:
  Erstellte Releases sollen beim Kunden bereits einsetzbar sein.
- Team-Kontinuität:
  Alle Mitarbeiter sollen nur in genau einem Projekt arbeiten.
- Schrumpfende Teams:
  Mitarbeiter werden zu 100% freigesetzt, damit sie in anderen Teams mitarbeiten können.
- Ursprungsursachen-Analyse:
  Es findet eine systematische Problemanalyse statt, um eine Ursache zu finden.

- Gemeinsamer Code:
  Jeder Entwickler hat Zugriff auf den gesamten Code und kann diesen auch verändern.
- Code und Tests:
  Code und Tests stellen die einzige Dokumentation dar.
- Eine Codebasis:
  Es gibt nur eine einzige Codebasis. Verzweigungen sollen vermieden werden.
- Tägliche Auslieferung:
  Jeden Tag soll die aktuelle Systemversion an die Anwender ausgeliefert werden.
- Vertrag mit aushandelbarem Umfang:
  Es wird ein Vertrag abgeschlossen, der ein festes Budget aber einen verhandelbaren Funktionsumfang beinhaltet.
- Bezahlung pro Benutzung:
  Funktionen werden pro Benutzung bezahlt.

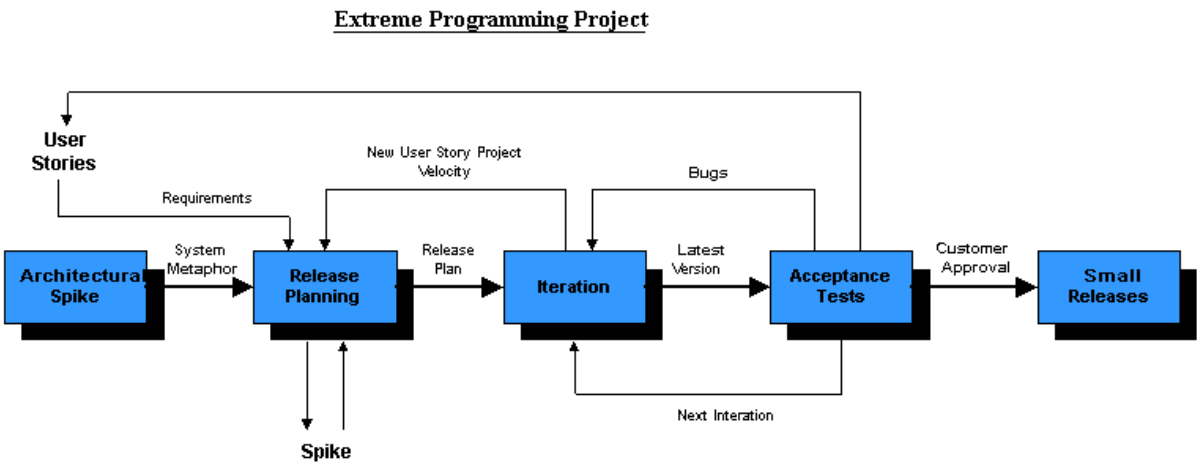


Abbildung 3.2: Der XP-Prozess [22]

### 3.2.3 Scrum

Der folgende Abschnitt enthält eine Zusammenfassung aus [2], [3], [4] und [8].

Scrum ist eine agile Methode, die nicht nur in der Softwareentwicklung eingesetzt wird. Die Anfänge von Scrum liegen in dem 1986 von Takeuchi und Nonaka veröffentlichten Artikel „The New Product Development Game“, in dem sie ausführten, „…dass Projekte mit kleinen, hochvernetzten Teams häufig die besten Ergebnisse erzielen und dass diese Teams mit der sogenannten Scrum Formation im Rugby vergleichbar seien“ (nach [4] S.125). In der Softwareentwicklung wurde Scrum in den 1990er Jahren durch Ken Schwaber, Jeff Sutherland und Mike Beedle bekannt gemacht.

Anders als XP oder FDD ist Scrum eher eine Methode für das Projektmanagement als für die Programmierung. Daher müssen bei einer Anwendung von Scrum noch Praktiken für Entwurf, Implementierung und Tests hinzugefügt werden. In der Praxis wird Scrum deshalb gern mit XP kombiniert, das diese Praktiken bereits enthält. Scrum setzt noch stärker als XP auf sich selbst organisierende Teams.

### 3.2.3.1 Rollen

**Product Owner**

Der Product Owner vertritt die Kundenseite. Er formuliert und priorisiert Anforderungen und stellt zusammen mit dem Team für jeden Sprint den Sprint Backlog zusammen. Außerdem steht er für Rückfragen des Teams zur Verfügung

**Team**

Das Team besteht aus 5-10 Entwicklern, die eigenverantwortlich für die Umsetzung der Anforderungen zuständig sind. Es handelt mit dem Product Owner aus, welche Anforderungen in welcher Zeit erledigt werden sollen.

**Scrum Master**

Er unterstützt Product Owner und Team, achtet auf die Einhaltung der Prozesse und hilft bei Problemen. Der Scrum Master sollte nicht Mitglied des Teams sein.

### 3.2.3.2 Ablauf

Zuerst werden die Anforderungen an das System ermittelt und in einer Auftragsbestandsliste – dem Product Backlog – gesammelt. Jede Anforderung wird vom Team geschätzt und vom Product Owner priorisiert. In das Product Backlog können zu jeder Zeit von jedem Projektbeteiligten neue Anforderungen hinzugefügt werden. In Zusammenarbeit mit dem Product Owner bestimmt das Team im Sprint Planning Meeting für die nächste Iteration Ziel und Zweck und überführt entsprechende Anforderungen aus dem Product Backlog in eine Aufgabenliste, das Sprint Backlog. Die Anforderungen werden in einzelne Arbeitsschritte herunter gebrochen und unter den Teammitgliedern verteilt.

In der Sprint Phase finden die Entwicklung und die Tests statt. Scrum macht keine Vorgaben wie diese genau aussehen sollen, das Team entscheidet über sein Vorgehen selbst. Ein Sprint dauert 30 Tage.

Während dieser Phase müssen die Anforderungen im Sprint Backlog unverändert bleiben, der Product Owner kann also keine Änderungen einbringen. Im Gegenzug darf das Team nur Anforderungen aus dem Sprint Backlog abarbeiten. Es gibt ein tägliches Meeting, das im Stehen abgehalten wird (Daily Scrum) und auf 15 Minuten begrenzt ist. Das Meeting ist rein informativer Natur. Die Teammitglieder berichten, was sie seit dem letzten Meeting getan haben, was sie bis zum nächsten tun werden und wo es evtl. Probleme gibt. Fachliche Angelegenheiten und Lösungen von Problemen werden nach dem Meeting nur von den betreffenden Personen diskutiert.

Im Sprint Review Meeting wird dem Product Owner das Ergebnis des Sprints vorgeführt. Außerdem findet eine Retrospektive statt, in der die Teammitglieder besprechen, was gut lief und was evtl. noch verbessert werden kann.

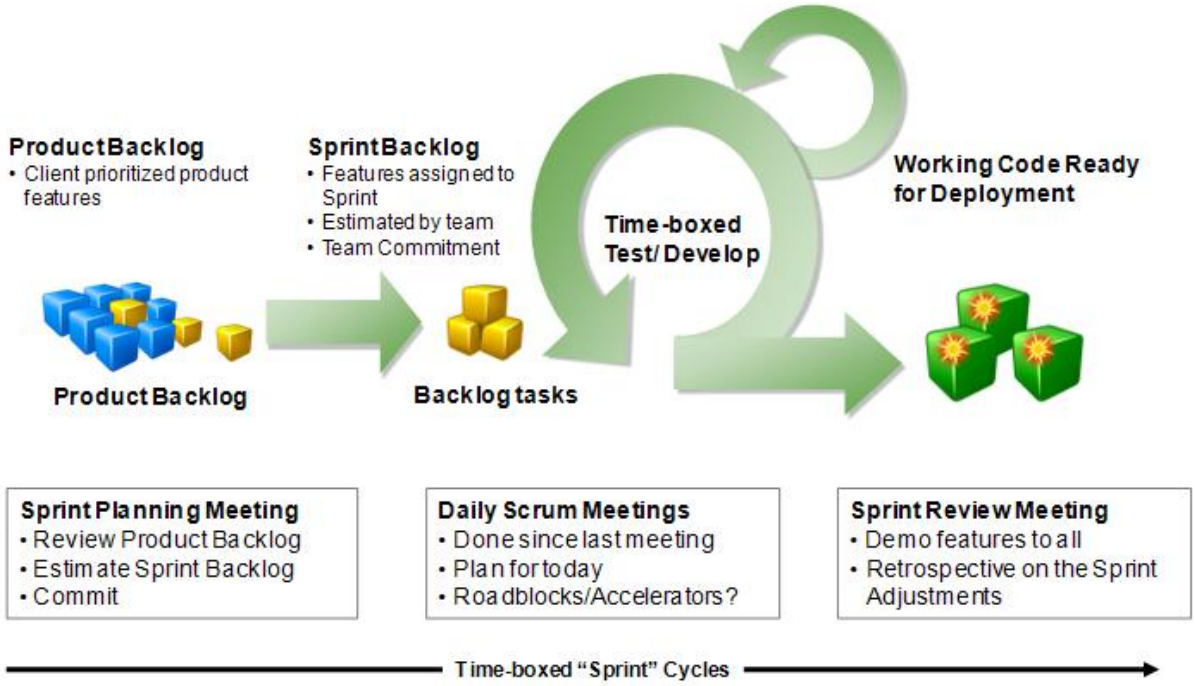


Abbildung 3.3: Der Scrum-Prozess [23]

### 3.2.4 Feature Driven Development

Der folgende Abschnitt enthält eine Zusammenfassung aus [1].

Entwickelt wurde Feature Driven Development (FDD) 1997 während eines Großprojekts in Singapur von Jeff De Luca, Peter Coad und Stephen R. Palmer. Erwähnung in der Literatur fand FDD erstmals 1999 in dem Buch „Java Modelling in Color with UML“ von Peter Coad, bis es dann 2002 eine

detailierte Beschreibung in „A Practical Guide to Feature-Driven Development“ von Stephen R. Palmer und John M. Felsing gab.

Feature Driven Development ist eine speziell für Festpreisprojekte entwickelte agile Methode. Gegenüber Methoden wie Extreme Programming oder auch Scrum besitzt sie einen recht hohen Grad an Prozessbeschreibungen. Insgesamt gliedert sich FDD in fünf Einzelprozesse, von denen die ersten drei sequenziell und Prozess 4 und 5 für jedes Feature einzeln ausgeführt werden.

Der Grundgedanke von FDD ist, dass die gesamten Anforderungen des Kunden in Features zerlegt werden, dabei ist ein Feature „… a small, client valued function expressed in the form: <action> <result> <object> (e.g. ‚calculate the total of a sale‘).“ ([1] S. 57).
Im Entwicklungszyklus einer Software umfasst FDD die Bereiche Entwurf und Programmierung. Ausgelassen werden u.a. die erste Anforderungsanalyse, der Change Request Prozess und die abschließenden System Tests. Hier sind bei einer Anwendung von FDD selbst geeignete Teilprozesse zu erstellen. FDD ist sehr gut für die objektorientierte Programmierung im Rahmen von Festpreisverträgen geeignet. Dabei muss bei dieser agilen Methode – anders als etwa bei XP oder Scrum – nicht auf einen hierarchischen Aufbau des Teams verzichtet werden, was diese Methode gerade für Unternehmen, die bisher mit traditionellen Modellen gearbeitet haben, interessant macht.

#### 3.2.4.1 Rollen

Da FDD bis hin zu sehr großen Projekten skaliert, gibt es auch eine große Zahl von vordefinierten Rollen. Diese gliedern sich in Schlüsselrollen, unterstützende Rollen und sonstige Rollen. Es müssen nicht zwangsläufig alle Rollen besetzt werden. Je nach Projektgröße können Rollen weggelassen bzw. auch mehrere auf eine Person zusammengelegt werden.

**Schlüsselrollen**

**Projektmanager (Project Manager)**
Dieser ist verantwortlich für Fortschrittsberichte, die Budgetverwaltung, den Projektumfang, den Zeitplan sowie für das Personal.

**Chefarchitekt (Chief Architect)**
Der Chefarchitekt ist verantwortlich für den Gesamtentwurf des Systems und ist auch Ansprechpartner für alle hierzu aufkommenden Fragen.

**Entwicklungsleiter (Development Manager)**

Er leitet die alltägliche Arbeit in der Entwicklung und löst etwaige Ressourcenkonflikte zwischen den Chefprogrammierern.

**Chefprogrammierer (Chief Programmer)**

Der Chefprogrammierer nimmt an der High-Level Anforderungsanalyse und am Entwurf des Gesamtsystems teil. Desweiteren leitet er kleine Teams aus 3-6 Entwicklern, die sich um die Low-Level Anforderungsanalyse, den Entwurf und die Entwicklung der neuen Features kümmern und arbeitet mit anderen Chefprogrammierern bei alltäglichen Technik- und Ressourcenproblemen zusammen.

**Klassenbesitzer (Class Owner)**

Der Klassenbesitzer ist ein Entwickler, der in einem kleinen Entwicklungsteam unter der Leitung eines Chefprogrammierers (der Chefprogrammierer ist selbst auch ein Klassenbesitzer) mitarbeitet. Zu seinen Aufgaben zählen Programmierung, Entwurf, Tests und Dokumentation der einzelnen Features.

**Fachexperte (Domain Expert)**

Der Fachexperte wird von der Kundenseite gestellt, üblicherweise ist er ein späterer Anwender, ein Business Analyst oder der Auftraggeber. Er erklärt den Entwicklern die Anforderungen an das zukünftige System.

**Unterstützende Rollen**

- Domain Manager: Leitet ein Team aus Fachexperten
- Release Manager: Stellt sicher, dass die Chefprogrammierer regelmäßig über den Fortschritt berichten
- Language Guru: Spezialist für eine bestimmte Programmiersprache bzw. Technologie
- Build Engineer: Ist verantwortlich für den Build-Prozess
- Toolsmith: Programmiert kleinere Entwicklungswerkzeuge
- System Administrator: Ist verantwortlich für die technische Infrastruktur

**Weitere Rollen**

- Tester: Ist verantwortlich für die Durchführung von Tests
- Deployer: Konvertiert bestehende Daten in das Format des neuen Systems
- Technical Writer: Schreibt die Dokumentation für das System

### 3.2.4.2 Ablauf

Nachdem die Anforderungen an das zu entwickelnde System bekannt sind, stellen in Prozess 1 die Fachexperten zunächst den Fachbereich dar, d.h. sie erklären den Entwicklern die ablaufenden Prozesse und stellen alle sonstigen notwendigen Informationen bereit. Danach wird das Gesamtmodell des Systems in Kleingruppen erstellt. Die Modelle werden im gesamten Team besprochen und es wird eine Variante ausgewählt, bzw. aus den Modellen ein neues erstellt. Das Gesamtmodell wird durch die Fachexperten überprüft.

In Prozess 2 wird aus den in Prozess 1 gewonnenen Informationen eine Featureliste erstellt. Unterteilt wird diese Liste in Fachgebiete (auch Major Feature Sets genannt), die eine Reihe von Geschäftsaktivitäten (Feature Sets) enthalten, die sich wiederum aus einzelnen Schritten zusammensetzen, den Features.

In Prozess 3 wird die Entwicklungsreihenfolge der Geschäftsaktivitäten festgelegt. Dabei sind Abhängigkeiten zwischen den Features zu beachten, außerdem sollte die Arbeitsbelastung der Klassenbesitzer möglichst gleichmäßig sein. Die Geschäftsaktivitäten werden den Chefprogrammierern zugewiesen sowie Klassen an einzelne Programmierer.

Danach werden in Prozess 4 die Feature Teams zusammengestellt. Dazu identifiziert der Chefprogrammierer die Klassen, die er für die Umsetzung seines Feature Sets braucht. Der Chefprogrammierer und die Klassenbesitzer bilden das Feature Team. Ein Klassenbesitzer kann Mitglied in mehreren Feature Teams sein, genauso wie ein Chefprogrammierer auch Mitglied eines Feature Teams unter der Leitung eines anderen Chefprogrammierers sein kann.
Für die umzusetzenden Features werden Sequenzdiagramme erstellt und mit den gewonnenen Informationen das Gesamtmodell verfeinert. Danach werden die Klassen- und Methodenrümpfe erstellt und der gesamte Entwurf einer abschließenden Inspektion unterzogen.

In Prozess 5 erfolgt die eigentliche Implementierung des Features, gefolgt von einer Codeinspektion und Unit Tests. Ist das Feature fehlerfrei umgesetzt, wird es dem Gesamtsystem hinzugefügt.
Prozess 4 und 5 werden solange wiederholt, bis alle Features umgesetzt worden sind. Die Prozesse 1-3 werden normalerweise nur einmal durchlaufen, bei Bedarf kann aber auch nochmal zu diesen zurückgesprungen werden.

### 3.2.4.3 Kernpraktiken

In FDD werden insgesamt acht Kernpraktiken eingesetzt:

- Inspektionen
- Objektorientierte Modellierung des Anwendungsgebiets
- Entwicklung pro Feature
- Individueller Codebesitz
- Feature-Teams
- Regelmäßige System-Builds
- Configuration Management
- Sichtbarkeit des Fortschritts

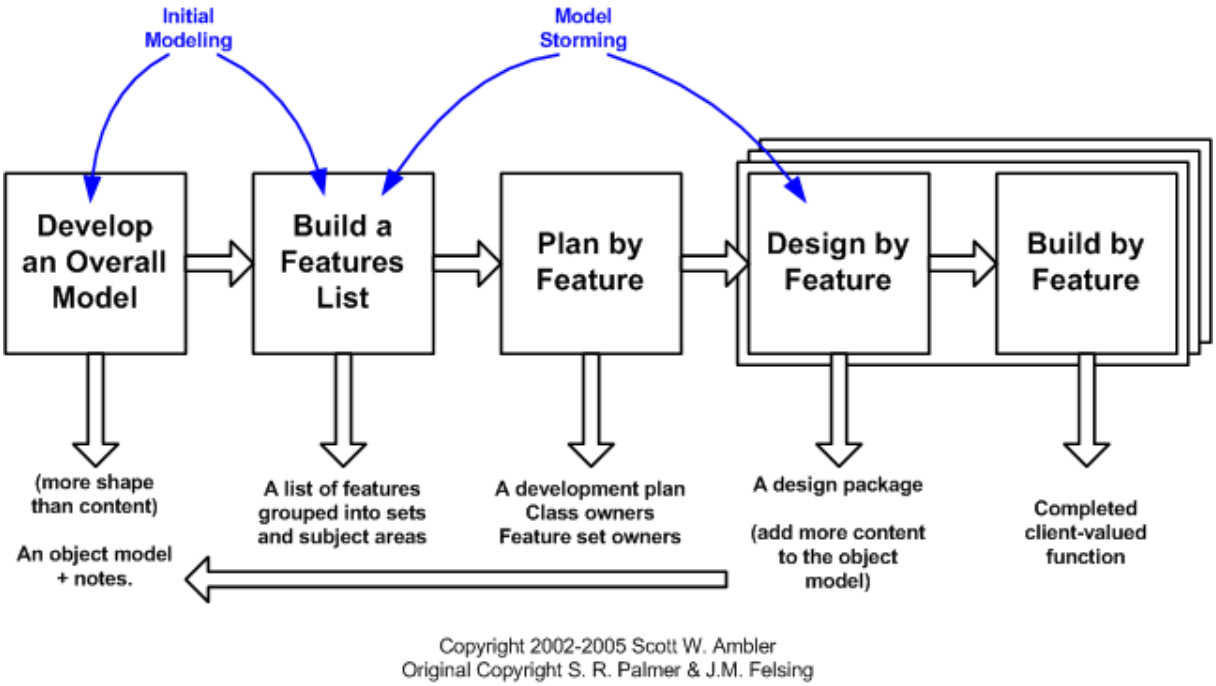


Abbildung 3.4: Der FDD-Prozess [24]

Nachdem im zweiten und dritten Kapitel die Grundlagen für die Prozessanalyse und anschließende Verbesserung mittels agiler Methoden gelegt wurden, werden im nun folgenden Kapitel die Ist-Prozesse und Strukturen der Internetagentur analysiert und dargestellt.

# 4 Analyse des Unternehmensumfeldes

Um sinnvolle Vorschläge für eine Verbesserung des Arbeitsablaufes machen zu können, bzw. eine neue Software-Entwicklungsmethode in der Internetagentur der tro:net einzuführen, muss zunächst das Umfeld analysiert werden. Daher werden in diesem Kapitel die Prozesse der Internetagentur dargestellt sowie Strukturen und Arbeitsweisen beschrieben.
Zu den Details wurden die Geschäftsführung und die jeweils zuständigen Mitarbeiter im Rahmen eines unstrukturierten Interviews befragt. Die Themen waren hierbei:

**Geschäftsführung**

- Organisatorischer Aufbau
- Gesamtprozess (Prozessrahmen)
- Behandlung von Änderungswünschen des Kunden im laufenden Projekt (Change Request)
- Prozesse im Projektmanagement, insbesondere die Bearbeitung von Tickets mit der Software Mantis

**Projektmanagement**

- Arbeitsabläufe
- Change Request
- Projektfortschrittsmessung
- Kundeneinbindung
- Störfaktoren in Arbeitsabläufen

**Vertrieb**

- Arbeitsabläufe
- Kenntnisse über eigene Produkte
- Störfaktoren in Arbeitsabläufen

**Anwendungsentwicklung**

- Arbeitsabläufe
- Anforderungsanalyse
- Entwurf
- Verwendete Technik (Programmiersprache, Entwicklungsumgebung, Objektorientierung etc.)

- Organisation untereinander (Arbeit an Projekten, Code Standards etc.)
- Kundeneinbindung

**Mediengestaltung**

- Arbeitsabläufe
- Kundeneinbindung
- Verwendete Programme
- Störfaktoren in Arbeitsabläufen

Die gewonnenen Ergebnisse aus der Mitarbeiterbefragung sind in die folgenden Prozessdarstellungen, Beschreibungen und Texte eingeflossen.

## 4.1 Organisatorischer Aufbau

Die tro:net gliedert sich in vier Abteilungen:

- Verwaltung
- Vertrieb/Marketing
- Produktion E-Business/Internetagentur (efruits)
- Produktion Networking & Providing

Für die nachfolgende Betrachtung ist nur die Abteilung Produktion E-Business/Agentur relevant. Sofern für das Gesamtverständnis nötig, werden Schnittstellen zu den anderen Abteilungen aufgezeigt.

Die Abteilung Produktion E-Business/Agentur bestand zu Beginn der Diplomarbeit aus insgesamt fünf Mitarbeitern: einem Abteilungsleiter, der gleichzeitig Projektmanager ist, zwei Anwendungsentwicklern und zwei Mediengestaltern.
Im Verlauf der Arbeit veränderte sich die Struktur in dieser Abteilung, so dass diese nunmehr aus einem Abteilungsleiter und einem Teamleiter besteht, die sich die Projektmanagementaufgaben teilen, sowie einem Entwicklerteam, dem ein Anwendungsentwickler und zwei Mediengestalter angehören. Im Folgenden schließt der Begriff „Entwickler“ sowohl Anwendungsentwickler als auch Mediengestalter ein.

## 4.2 Projektarten

Innerhalb der Internetagentur der tro:net werden vier Projektarten unterschieden:

**Webseitenentwicklung mit dem Contentmanagementsystem Typo3**
Die Arbeiten umfassen hierbei das Aufsetzen eines Typo3 Basissystems, die Codierung der Templates – Vorlagen für die Webseiten – nach Kundenwünschen, die Integration von Standarderweiterungen für Typo3, sowie ggf. die Entwicklung individueller Erweiterungen für Typo3.

**Entwicklung eines Internetshops mit der Shopsoftware Oxid**
Die Arbeiten bei diesem Projekttyp umfassen die Installation des Basissystems, die Integration von Standardmodulen, ggf. die Entwicklung individueller Module sowie optische Anpassungen nach Kundenwunsch.

**Individuelle Softwareentwicklung**
Hierzu gehören alle Projekte, die sich nicht mit Standardsoftware umsetzen lassen, z.B. die Umsetzung eines Bestellvorgangs mit entsprechenden (Eingabe)Formularen.

**Interne Projekte/Eigenentwicklungen**
Die tro:net besitzt momentan drei Eigenentwicklungen, dabei handelt es sich um Schnittstellen und Erweiterungen für die Standardsoftware Typo3 und Oxid. Diese sind normalerweise keine eigenen Projekte – es sei denn, es wird ein komplett neues Produkt entwickelt oder es ist eine generelle Anpassung an eine neue Produktversion nötig – sondern werden je nach Bedarf an das zu entwickelnde Produkt für den Kunden angepasst.
Interne Projekte können aber auch die Entwicklung von Werkzeugen für die alltägliche Arbeit sein. Diese Art von Projekten unterliegt einer internen Versionierung.

Bei Kundenwünschen, die sowohl eine Lösung auf Basis einer Standardsoftware als auch einen Anteil an individueller Softwareentwicklung beinhalten (z.B. eine Webseite mit einem einzigartigen Bestellvorgang), wird der Auftrag in mehrere Projekte gesplittet.

## 4.3 Projektmanagement mit Mantis

Für die Projektplanung und -durchführung wird der Bugtracker *Mantis* verwendet, der von der tro:net dahingehend erweitert wurde, dass er nicht nur für das Erfassen von Fehlern, sondern auch für das gesamte Projektmanagement verwendet werden kann.

In Mantis können Arbeitspakete als Tickets eingetragen, eingeplant und verfolgt werden. Außerdem können Abhängigkeiten zwischen den einzelnen Tickets definiert und Kommentare verfasst werden. Wenn ein Kunde möchte, kann er sich einen Account in Mantis anlegen lassen und die Entwicklung an seinem Projekt verfolgen, selbst Tickets anlegen (z.B. bei Änderungswünschen) und Kommentare verfassen. Kunden die bereits in Mantis registriert sind, können auch nach Abschluss des Projekts noch Tickets erstellen, beispielsweise wenn sie ein Angebot für eine bestimmte Dienstleistung wünschen.

### 4.3.1 Ticket-System

Nach Auftragseingang wird für jedes Arbeitspaket ein Ticket in Mantis angelegt. Das Ticket gehört einem der im Folgenden aufgeführten elf Anliegen an.

**Projektanforderung**
Eine Projektanforderung ist eine Anforderung wie im Angebot beschrieben, z.B. Interface/ Screendesign.

**Serviceanforderung**
Bei einer Serviceanforderung handelt es sich um eine Anforderung für eine Dienstleistung, die nicht im Angebot vereinbart wurde. Kontingente können in einem Pflegevertrag vereinbart werden. Ein Beispiel hierfür ist die Befüllung einer Seite mit Texten und Bildern.

**Angebotsanforderung**
Eine Angebotsanforderung ist eine Anfrage des Kunden nach einer Angebotserstellung oder einem neuen Feature in einem laufenden Projekt.

**Oberanliegen Fehler**
Diese acht Anliegen werden alle als Fehler behandelt:

- Änderungs-/Optimierungswunsch
- Frage/Support
- Fehler im Inhalt
- Fehler im Design
- Kleinerer Funktionsfehler
- Schwerer Funktionsfehler
- Fehlende Funktion
- Blocker (d.h. das vorliegende Problem blockiert die Weiterarbeit am gesamten Projekt)

Jedes Ticket besitzt einen von acht Stati:

| **Statusbezeichnung** | **Bedeutung** |
|---|---|
| Neu | Der Standardstatus, wenn ein Ticket neu angelegt wird. |
| Planung | Das Ticket muss vom Projektmanagement eingeplant werden. |
| Analyse | Von der Fachabteilung ist eine Aussage zu Aufwand/Machbarkeit erforderlich. |
| Umsetzung | Das Ticket wurde einem Bearbeiter zugewiesen und kann umgesetzt werden. |
| Erfordert Rückmeldung | Es ist eine Rückmeldung vom Kunden nötig. |
| Rückmeldung | Die Rückmeldung durch den Kunden ist erfolgt. |
| Erledigt | Das Ticket wurde bearbeitet und ist zum Test freigegeben. |
| Geschlossen | Das Ticket ist abgearbeitet und wurde erfolgreich getestet. |

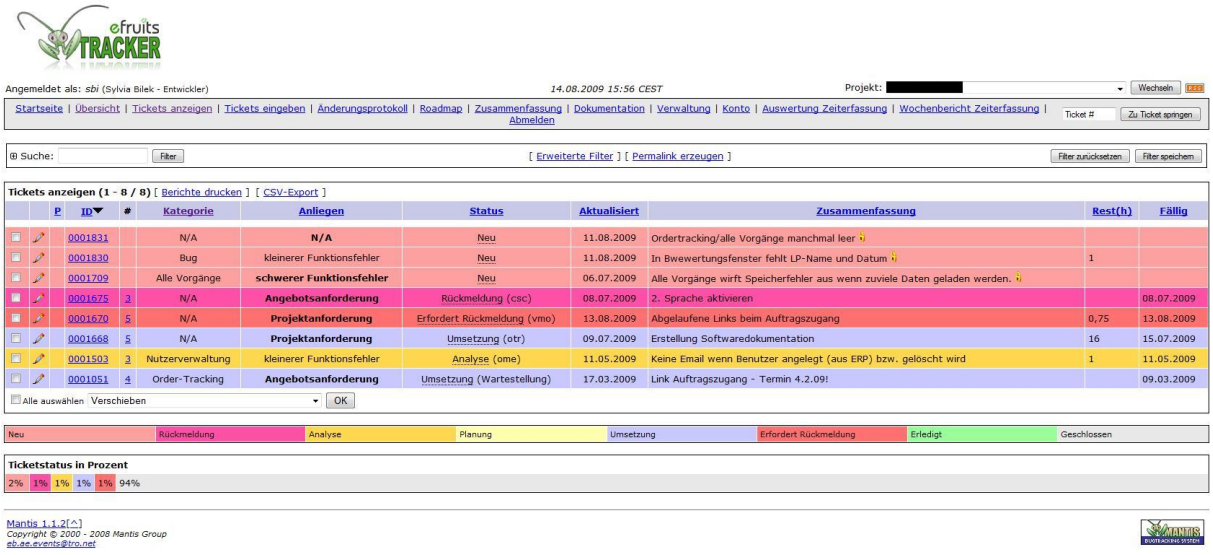


Abbildung 4.1: Mantis

### 4.3.2 Probleme bei der Arbeit mit Mantis

Es gibt für den Lebenszyklus eines Tickets zwar Richtlinien, die eingehalten werden sollten, aber bisher keine genaue Prozessdarstellung bzw. Vorgaben, dies führt zu folgenden Problemen:

- Der Entwickler hat ein Ticket erledigt, belässt den Status aber auf „Umsetzung“, da er meint, der Kunde könnte sich noch einmal melden.
- Der Entwickler setzt ein Ticket auf „Erledigt“, obwohl er einen kleinen Unterpunkt noch nicht abgearbeitet hat.
- Der Entwickler schafft die Erledigung eines Tickets nicht bis zum angegebenen Fälligkeitstermin, belässt es aber auf „Umsetzung“, statt es wieder auf „Planung“ zurückzustufen.
- Der Kunde stuft einen Fehler als schwerer ein, als er eigentlich ist.

## 4.4 Entwicklung

Die Entwicklung gliedert sich in die zwei Bereiche Anwendungsentwicklung und Mediengestaltung. Zugeteilt wird die Arbeit über Mantis, hier kann jeder Entwickler sehen, welche Tickets er im Laufe des Tages bearbeiten soll. Normalerweise arbeitet jeder Entwickler an einem anderen Teil eines Projekts und an mehreren Projekten gleichzeitig. Der Arbeitsbereich der Entwickler umfasst hierbei die Abschätzung und Implementierung/Codierung der Anforderungen, falls nötig bzw. vom Kunden gewünscht, außerdem Entwurf, Dokumentierung und Tests.
In der Anwendungsentwicklung wird fast ausschließlich PHP als Programmiersprache genutzt, Ausnahmen davon gibt es selten. Da Standardsoftware wie z.B. Oxid Shop bereits ein Framework bereitstellt, wird diese häufig nur um neue Methoden ergänzt, seltener werden eigene Klassen geschrieben. Generell wird nach dem MVC-Prinzip programmiert.
Die Entwicklungsumgebung kann sich jeder Anwendungsentwickler selbst auswählen, diese sind also nicht einheitlich. Ein Versionsverwaltungssystem wird bisher nicht genutzt, ist aber bereits evaluiert (Subversion) und soll in näherer Zukunft eingesetzt werden. Offizielle Richtlinien für die Programmierung gibt es nicht, die Anwendungsentwickler haben sich jedoch untereinander Coding-Standards gesetzt, um den Code anderer besser und schneller verstehen zu können.

Die Mediengestalter erstellen ihre Design-Entwürfe vornehmlich mit Photoshop. Zur anschließenden Codierung wird die Software YAML verwendet. YAML ermöglicht eine einheitliche Darstellung der Seite in den gängigen Browsern. Ansonsten besteht die Arbeit der Mediengestalter hauptsächlich in der Konfiguration des Content-Management-Systems Typo3 und der Anpassung von Templates.

Wenn vom Kunden gewünscht, führen sie nach Projektende außerdem eine Schulung in Typo3 für den Kunden durch.

## 4.5 Prozesse

Auf Grund ihrer Größe befinden sich die Diagramme zum Gesamtprozess und zum Change Request auf der dieser Diplomarbeit beiliegenden CD. Die Diagramme zu den Mantis-Prozessen befinden sich im Anhang und zusätzlich auf CD.

### 4.5.1 Gesamtprozess

Das Diagramm zum Gesamtprozess stellt den kompletten Ablauf eines Projekts von der Akquise bis zur Auslieferung des Systems dar. Schnittstellen zum Finanzwesen sind im Diagramm der Vollständigkeit halber mit angegeben, werden hier aber nicht extra erklärt.

Der Prozess beginnt mit der Anfrage eines Kunden, ihm ein Angebot zu erstellen bzw. dem Versuch des Vertriebs über Marketing und Akquise einen Kunden zu gewinnen. Hat der Kunde ein entsprechendes Interesse, übermittelt er seine Anforderungen bzw. ein Lastenheft und wird vom Consulting bezüglich Umsetzung, Machbarkeit etc. beraten. Als Ergebnis dieses Gesprächs wird ein grobes Anforderungsprofil erstellt.

Enthält das Projekt individuelle Anforderungen, z.B. eine Funktion für den Oxid-Shop, für die es kein Standardmodul gibt, oder handelt es sich um ein komplettes Individual-Software-Projekt, wird zunächst für jede nicht abschätzbare Anforderung von den Mediengestaltern bzw. Anwendungsentwicklern eine Machbarkeitsanalyse erstellt. Bei nichtumsetzbaren Anforderungen wird der Kunde informiert und es werden ihm ggf. Alternativvorschläge unterbreitet. Ansonsten werden im Anschluss noch eine Aufwandsschätzung und Grobkalkulation erstellt.

Wenn alle Anforderungen abgeschätzt sind, erstellt der zuständige Consultant eine Gesamtkalkulation. Bei mehreren für das Projekt zu entwerfenden Software-Architekturen wird dieses in Teilprojekte aufgesplittet. Danach verfasst der Consultant das Angebot für den Kunden, welches aus dem eigentlichen Angebotsteil sowie einer Leistungsbeschreibung besteht. Je nachdem, über wen der Kundenkontakt hergestellt worden ist, kümmern sich um Angebotsversand und Nachverfolgung entweder der Vertrieb oder der Consultant.

Nach Erhalt prüft der Kunde das Angebot. Bei Änderungswünschen wendet er sich zunächst an den Vertrieb, dieser hat bei Nachverhandlungen einen Spielraum von bis zu 10% des Gesamtvolumens für Rabatte/Preisnachlässe. Bei Überschreitung dieses Prozentsatzes übernimmt der Consultant die Verhandlungen mit dem Kunden. Der Consultant kürzt ggf. die Leistungen und es wird ein neues Angebot erstellt.

Erfolgt seitens des Kunden eine Beauftragung, nimmt der Vertrieb eine formelle Prüfung vor und das Sekretariat erfasst den Auftrag. Im Anschluss nimmt das Projektmanagement telefonischen Kontakt mit dem Kunden auf oder veranstaltet bei Bedarf ein Kick Off Meeting, um den Zeitplan abzustimmen. Im Anschluss erfolgt die Struktur- und Zeitplanung, der Eintrag aller Leistungspakete in Mantis sowie eine Freigabe an das Finanzwesen zwecks Erstellung einer Rechnung über die vereinbarte Anzahlung.

Danach analysiert der Konzepter die Anforderungen bzw. das Lastenheft und wenn vorhanden, den bisherigen Internetauftritt des Kunden. Anschließend führt er einen Anforderungs- und Beratungs-workshop mit dem Kunden durch. Wird Individual-Software entwickelt, ermitteln die Medien-gestalter/Anwendungsentwickler eine Basis für die Umsetzung, z.B. bereits auf dem Markt existierende Standardsoftware. Daraufhin erstellt der Konzepter ein Website Konzept bzw. bei Individual-Software ein User Interface Konzept und versendet es an den Kunden. Hat der Kunde noch Änderungswünsche, werden diese durch den Change Request Prozess behandelt. Im nächsten Schritt erstellen die Mediengestalter einen oder mehrere Design-Entwürfe und verfassen, wenn vom Kunden beauftragt, einen Style-Guide. In der Regel erfolgen danach noch mehrere Änderungen, bis der Kunde das Design abnimmt.

Nun beginnt die eigentliche Entwicklung. Dabei wird nach den Projektarten Typo3, Oxid-Shop und Individual-Software unterschieden.
Bei der Projektart Typo3 bestehen die ersten Arbeitsschritte aus Basiscodierung, Codierung der Templates, Installation und Konfiguration des Basissystems und der Integration der Templates. Danach werden Standarderweiterungen integriert (z.B. eine Bildergalerie) und optisch angepasst, sowie bei Bedarf individuelle Erweiterungen programmiert bzw. unternehmenseigene Produkte angepasst. Wenn gewünscht, werden Inhalte (Texte, Bilder etc.) in Typo3 eingepflegt.
Für die Projektart Oxid-Shop werden zuerst das Basissystem installiert und danach die gewünschten Standardmodule integriert. Bei Bedarf werden außerdem individuelle Module programmiert bzw. unternehmenseigene Produkte angepasst. Zuletzt erfolgt die Anpassung der Templates und Stylesheet-Dateien.
Bei der Projektart Individual-Software wird zunächst von den Anwendungsentwicklern/Medien-gestaltern geprüft, ob bereits vorhandene Standardsoftware als Basis für die Umsetzung verwendet

werden kann. Auf Grundlage dieser Prüfung werden die nächsten Arbeitsschritte individuell festgelegt.
Parallel zum Entwicklungsprozess erfolgen die Kontrolle der abgearbeiteten Tickets durch das Projektmanagement sowie etwaige Bugfixes, Teilrechnungen werden freigegeben und dem Kunden ein Link zum Entwicklungsserver zugesendet, auf dem er die Entwicklung verfolgen kann. Tests werden nur durchgeführt, wenn der Kunde diese kostenpflichtig beauftragt hat. Ein Testszenario besteht normalerweise aus Testkonzeption, Durchführung, Bugfixing und Nachtest.

Nach Abschluss der Entwicklungsarbeiten wird dem Kunden über das Projektmanagement die Fertigstellung gemeldet. Es folgt die Abnahme durch den Kunden. Treten hierbei noch Fehler auf, muss nachgebessert werden und die Abnahme wird erneut durchgeführt. Am Ende des Projekts wird die Endabrechnung freigegeben und die Software auf das Live-System des Kunden übertragen.

Der erste Pool im Diagramm stellt das Unternehmen des Kunden dar, der zweite die tro:net mit den für diesen Prozess relevanten Organisationseinheiten, der dritte das Finanzwesen. Mitarbeiter können im Laufe des Projekts Aufgaben in mehreren Organisationseinheiten übernehmen.
Das Diagramm ist auf der CD unter „Gesamtprozess“ zu finden.

### 4.5.2 Change Request

Auch wenn in den Verträgen ein fester Funktionsumfang zu einem festen Preis vereinbart wurde, kommt es regelmäßig vor, dass ein Kunde mitten im Projekt feststellt, dass er eine bestimmte Funktion bzw. ein Designelement anders als vereinbart umgesetzt oder zusätzlich haben möchte.
Zunächst wird geprüft, ob die Anforderung des Kunden im Konzept enthalten ist, es sich also um ein Versäumnis des Unternehmens handelt. Ist das der Fall, wird die Anforderung kostenlos umgesetzt. Ist sie nicht im Konzept enthalten und der Aufwand so niedrig, dass die Umsetzung kaum Kosten verursacht, kann sie auf Kulanzbasis kostenlos umgesetzt/geändert werden.
Wenn der Aufwand einer Anforderung nicht bekannt ist, wird zunächst eine Machbarkeitsanalyse durchgeführt und bei positivem Ergebnis eine Aufwandsschätzung und Kalkulation. Verursacht die Anforderung so hohe Kosten, dass sie nicht mehr auf Kulanzbasis umgesetzt werden kann, wird diese in Absprache mit dem Kunden entweder auf ein Folgeprojekt verlegt oder ihm zusätzlich in Rechnung gestellt.
Werden für den Kunden zusätzliche Anforderungen umgesetzt, die nicht im Vertrag enthalten sind, ist das vereinbarte Fertigstellungsdatum für die tro:net nicht mehr bindend.

Der erste Pool im Diagramm stellt das Unternehmen des Kunden dar, der zweite die tro:net mit den für diesen Prozess relevanten Organisationseinheiten Projektmanagement und Mediengestaltung/Anwendungsentwicklung.
Das Diagramm ist auf der CD unter „Change Request“ zu finden.

### 4.5.3 Mantis-Prozesse

Für die drei Anliegen Projektanforderung, Serviceanforderung, Angebotsanforderung und das Oberanliegen Fehler gibt es jeweils einen Abarbeitungsprozess, der befolgt werden sollte. Zu den meisten Aufgaben gibt es zudem noch einzelne Arbeitsschritte.

Da in BPMN-Diagrammen, die mit Intalio erstellt werden, Rückflüsse nicht möglich sind, wurden zur Darstellung der Mantis-Prozesse Aktivitätsdiagramme verwendet. Im Gegensatz zu den BPMN-Diagrammen sind die Bahnen zu Projektmanagement, Mediengestaltung/Anwendungsentwicklung und Kunde hier senkrecht angeordnet.

Die sechs Diagramme sind im Anhang und zusätzlich auf der CD unter „Mantis-Prozesse“ zu finden. Vier Diagramme zeigen jeweils die Abfolge der Aufgaben für die Anliegen Projektanforderung, Serviceanforderung, Angebotsanforderung und das Oberanliegen Fehler. Die beiden anderen zeigen die Unterprozesse Analyse und Bearbeitung.

## 4.6 Qualitätsmanagement

Standardisierte Tests werden bisher nicht durchgeführt. Ist ein Ticket erledigt, wird die Fehlerfreiheit der Umsetzung durch den Entwickler selbst und zusätzlich noch durch den Ticketeröffner geprüft (Vier-Augen Prinzip). Ticketeröffner können hierbei der Projektmanager oder der Kunde sein. Sind noch Fehler vorhanden, werden diese beseitigt und anschließend ein weiteres Mal nach dem Vier-Augen Prinzip getestet.
Umfangreichere Tests mit Testkonzeption, Durchführung, Bugfixing und Nachtests werden nur dann durchgeführt, wenn der Kunde dies ausdrücklich wünscht und auch entsprechend bezahlt.

## 4.7 Probleme

Nach Auswertung der Antworten der Mitarbeiter und Analyse der Prozesse ergeben sich folgende Probleme:

**Anforderungsanalyse**
Die Fachabteilung ist häufig nicht bei Erstellung der Anforderungen/Konzepte beteiligt, sondern erst ab dem Zeitpunkt der Zeitabschätzung.

**Entwurf**
Bisher ist standardmäßig kein Entwurf vorgesehen bzw. nur rudimentär als „Bleistiftskizze“. Erst beim Bearbeiten eines Tickets plant der jeweilige Entwickler kurz die Implementierung der Anforderung.

**Implementierung/Codierung**
Das Konzept bzw. die Anforderungen sind oftmals zu ungenau, es gehen Informationen in der Kommunikationskette *Kunde -> Consultant -> Entwickler* verloren, die dann bei der Umsetzung fehlen. Dadurch sind wiederum Nachbesserungen nötig, da nicht das umgesetzt wird, was der Kunde ursprünglich wollte.
Bei Einsatz einer externen Mediengestalterin sind die Entwürfe teilweise schwer umsetzbar, weil gezeichnete Elemente nicht immer auch so codiert werden können.
Ein Versionsverwaltungssystem fehlt bisher, soll aber baldmöglichst eingeführt werden.

**Qualitätsmanagement**
Wenn der Kunde keine Tests beauftragt, werden auch keine durchgeführt bzw. Tickets nur nach dem Vier-Augen Prinzip überprüft. Das Problem hierbei ist, dass der Kunde trotzdem funktionierende und seinen Anforderungen entsprechende Software erwartet, auch wenn er keine separaten Tests bezahlen will. Das Vier-Augen Prinzip ist allerdings unzuverlässig, da jedes Ticket nur einzeln überprüft wird. Wenn durch die Umsetzung eines späteren Tickets ein Fehler in bereits bestehender Funktionalität entsteht, wird das u.U. gar nicht bemerkt: ein Ticket kann fehlerfrei umgesetzt worden sein und das folgende ebenfalls, trotzdem steckt ein Fehler im Gesamtsystem, der mit dieser Methode nicht gefunden wird oder erst sehr spät. Am Ende eines Projekts sind daher oft noch einige Nachbesserungen nötig.

**Sonstiges**
Bisher gibt es keinen Vorgang für die Prozessverbesserung, z.B. ein Meeting aller Mitarbeiter der Agentur, um Lösungen für vorhandene Probleme/Verbesserungsmöglichkeiten zu finden.

Bei häufigen Projektwechseln der Entwickler ist jeweils eine Einarbeitung/Umdenken in das neue Projekt nötig.
Für den Vertrieb werden keine/kaum Schulungen durchgeführt, dadurch verfügen die Vertriebsmitarbeiter nicht über ausreichende Produktkenntnisse, um einen Kunden optimal beraten zu können.
Die Kommunikation mit dem Kunden funktioniert oft nur schlecht. Lieferzeiten für Bilder, Texte etc. oder auch Rückmeldungstermine werden von einigen Kunden nicht eingehalten.

Nachdem in diesem Kapitel die Ist-Prozesse analysiert und Probleme identifiziert wurden, sollen die Prozesse nun im folgenden Kapitel mittels agiler Methoden und Praktiken verbessert und Optimierungsmöglichkeiten aufgezeigt werden.

# 5 Problemlösung mittels agiler Methoden

In diesem Kapitel sollen die im vierten Kapitel analysierten Probleme mittels agiler Methoden und Praktiken gelöst werden. Dazu sind zunächst die Besonderheiten des Webdesigns zu betrachten. Anschließend werden die Anforderungen an eine agile Methode aufgestellt und mit den im zweiten Kapitel vorgestellten Methoden abgeglichen. Die gewählte Methode muss dann an die Gegebenheiten der Internetagentur der tro:net angepasst werden.

## 5.1 Besonderheiten des Webdesigns

Die Software-Projekte der efruits Internetagentur umfassen sowohl Webdesign als auch Programmierung. Anders als bei der Entwicklung von reinen Desktop-Anwendungen spielt bei der Entwicklung von Anwendungen für das Internet (Webseiten, Shops etc.) das Design eine zentrale Rolle. Der Auftraggeber will über seinen Internetauftritt einerseits neue Kunden gewinnen, auf diese muss das Design ansprechend und überzeugend wirken, andererseits sollen Bestandskunden mit News, Informationen und Downloads versorgt werden. Die Seitenstruktur muss also ein schnelles Auffinden gewährleisten.

Der Auftraggeber hat oftmals noch keine genaue Vorstellung, wie sein Internetauftritt aussehen soll. Er weiß, welche Botschaft er dem Kunden vermitteln möchte, welche Gefühle beim Betrachten des Internetauftritts beim Kunden ausgelöst werden sollen, welches Image und welche Werte sein Unternehmen darstellen soll.

Beispiele aus einem realen Angebot:

- Wer unsere Sachen anschaut, der weiß, dass der Preist stimmt.
- Was schön ist, funktioniert auch.
- Nicht billig, aber preiswert.
- Qualität, Professionalität, Perfektionismus

Die Vorstellungen des Auftraggebers in ein Design umzusetzen, ist dann Aufgabe des Mediengestalters. Erreicht werden kann dies z.B. durch entsprechende Bilder, die Farbwahl und die Anordnung der Elemente. Beachtet werden muss hierbei auch die Branche, die Webseite eines Anwalts sollte beispielsweise nicht in Pastelltönen gestaltet sein. Der Kunde kann dies natürlich

letztendlich so verlangen, der Mediengestalter muss ihm aber die möglichen Auswirkungen seines Wunsches aufzeigen.
Ein weiteres wichtiges Kriterium ist die Usability. In einem Shop soll z.B. der Warenkorb durch den Benutzer schnell erreichbar und der Bestellvorgang intuitiv bearbeitbar sein. Ein Künstler will dagegen evtl. nur zeigen, dass er kreativ ist. Dieser braucht für seine Webseite eine andere Struktur als ein Unternehmen.

## 5.2 Anforderungen an eine agile Methode

Aus der im vierten Kapitel durchgeführten Analyse und den Ausführungen in Abschnitt 5.1 ergeben sich folgende Anforderungen an eine agile Methode:

- Sie muss für Festpreisprojekte (fester Funktionsumfang und fester Preis) geeignet sein.
- Sie berücksichtigt die Anforderungen an das Webdesign bzw. ist dahingehend erweiterbar.
- Eine Hierarchie ist im Entwicklungsteam der Internetagentur vorhanden, diese sollte auch von der Methode unterstützt werden.
- Sie soll eine Struktur in die bisher gewachsene Herangehensweise bringen (Planung, Entwurf, Implementierung, Tests).
- Der Kunde sollte nicht zu stark eingebunden werden (z.B. Kunde vor Ort), da nicht jeder Kunde willens ist, sich (erheblich) am Entwicklungsprozess zu beteiligen.
- Qualitätssicherungsmaßnahmen müssen eingebunden werden.

## 5.3 Vorteile und Nachteile der einzelnen Methoden

Die im Abschnitt 5.2 definierten Anforderungen müssen nun mit agilen Methoden abgeglichen werden, um eine passende auswählen zu können.

### 5.3.1 Extreme Programming

XP ist eine agile Methode, die sich auf Implementierung und Tests konzentriert. Sie geht von einer kontinuierlichen Änderung und Anpassung der Anforderungen aus. Dieser Umstand äußert sich z.B. in einer sehr kurzen Planungsphase. XP ist daher nicht für Festpreisprojekte mit außerdem festem Funktionsumfang geeignet.

Da XP zu den testgetriebenen Methoden gehört, werden die Tests konsequent vor der Implementierung geschrieben und soweit möglich automatisiert. Diese Art des Testens fordert eine sehr hohe Disziplin und entsprechende Programmierkenntnisse der Entwickler. Das gleiche gilt für das nahezu ständig ablaufende Refactoring. Entwickler mit nur wenig Erfahrung auf diesen Gebieten hätten hier große Schwierigkeiten. Fraglich ist außerdem, ob sich diese Vorgehensweise auf Webseitenkomponenten anwenden lässt, die nicht in einer objektorientierten Programmiersprache geschrieben sind.

XP fordert eine starke Beteiligung des Kunden, nach Möglichkeit direkt vor Ort beim Team. Der Kunde ist verantwortlich für das Schreiben von User Stories und Akzeptanztests. Natürlich wäre es positiv für ein Projekt, wenn der Kunde sich derart einbringt, in der Realität ist dies allerdings bei Kunden der tro:net nicht der Fall.

Es werden zahlreiche Praktiken definiert, die sich laut Kent Beck gegenseitig verstärken. Daher ist es zweifelhaft, ob sich der erwünschte Erfolg einstellt, wenn man einige für XP wichtige Praktiken weglässt, weil sie nicht auf die Internetagentur übertragbar sind (z.B. Paarprogrammierung, ständiges Refactoring etc.).

### 5.3.2 Scrum

Scrum ist ein Prozessrahmen, der Planung und Projektmanagementaktivitäten abdeckt, jedoch nicht definiert, wie Implementierung und Tests auszusehen haben. In der Praxis wird Scrum häufig mit anderen agilen Methoden kombiniert, die diesen Teil abdecken. Die weitverbreiteteste Kombination ist hierbei Scrum und XP, da XP auf die reine Programmierung von Software ausgerichtet ist und sich somit gut mit Scrum ergänzt.

Scrum ist genauso wie XP nicht für Festpreisprojekte ausgelegt. Ein weiterer Nachteil ist die sehr starke Einbeziehung des Kunden. Die allerwenigsten Kunden der tro:net wären bereit, soviel Zeit in das Projekt zu investieren. Desweiteren gibt es im Entwicklerteam von Scrum keine Hierarchie, das Team organisiert sich vielmehr selbst.

Zwei interessante Aspekte von Scrum sind jedoch das Daily Scrum und die Retrospektive. Ein kurzes Meeting im Stehen, dass jeden Tag durchgeführt wird, würde die Kommunikation zwischen den Entwicklern verbessern. Jeder erfährt, an was die anderen gerade arbeiten und wo evtl. Hilfe benötigt wird. Die rein informative Natur und Zeitbegrenzung des Daily Scrum sorgen außerdem dafür, dass durch dieses Meeting keine Zeit verschwendet wird.

Die Retrospektive ist ein Mittel um Prozesse kontinuierlich zu verbessern. Jeweils zu Projektende durchgeführt, kann sie helfen, Probleme rechtzeitig zu erkennen und in den folgenden Projekten zu vermeiden.

### 5.3.3 Feature Driven Development

FDD ist die Methode mit der am stärksten ausgearbeiteten Prozessstruktur, die Planung, Implementierung und Tests abdeckt. Weiterhin unterstützt diese Methode anders als XP und Scrum ein hierarchisches Teammodell, das sich gut auf die Internetagentur der tro:net übertragen lässt.

Ein Kunde vor Ort ist in FDD nicht notwendig, er wird jedoch bei der Modellierung und Verifizierung mit einbezogen.

Ein weiterer Vorteil von FDD ist, dass es speziell für Festpreisprojekte entwickelt wurde und deshalb davon ausgeht, dass nahezu alle Anforderungen bereits zu Beginn bekannt sind. Sich ergebende Änderungen können jedoch durch das iterative Vorgehen trotzdem berücksichtigt werden.

An ein paar wenigen Stellen ist FDD zu überdimensioniert, beispielsweise bei den Feature Teams und der objektorientierten Modellierung. Gleichzeitig sind jedoch zu wenig Tests vorhanden, nur Unit Tests reichen in der Webentwicklung nicht aus.

### 5.3.4 Fazit

Keine der drei agilen Methoden kann unverändert auf die Internetagentur übertragen werden. FDD bildet jedoch die beste Basis, um mit einigen Anpassungen eine geeignete Methode selbst zu entwerfen.

**Ausbau der Tests**
Neben den Unit Tests werden noch Tests für z.B. Layout und Usability benötigt.

**Anpassung an die Webentwicklung**
Die Vorgehensweise für das Anfertigen einer Feature-Liste ist auf die reine Programmierung zugeschnitten, für die Webentwicklung muss diese angepasst werden.

**Anpassungen beim Entwurf und der Modellierung**

Eine objektorientierte Modellierung ist nicht in allen Fällen nötig, z.B. bei reinen Website-Projekten mit Typo3. Zudem sind die Anforderungen meistens nicht derart komplex, dass Sequenzdiagramme erstellt werden müssen.

**Feature Teams**

Die Internetagentur der tro:net ist zu klein, um diese Praktik wie in der Prozessspezifikation von FDD zu verwenden. Hier ist eine Anpassung auf ein kleines Team notwendig.

Da nun FDD als erfolgsversprechendste Basis für Verbesserungen im Entwicklungsprozess identifiziert wurde, wird in den folgenden Abschnitten FDD auf die Anforderungen der Internetagentur angepasst und evtl. fehlende Elemente werden hinzugefügt.

## 5.4 Anforderungsanalyse

Ein in FDD nicht berücksichtigter, aber dennoch für die Softwareentwicklung sehr wichtiger Vorgang ist die Anforderungsanalyse. Daher habe ich zusammen mit dem Teamleiter ein Verfahren für die Anforderungsanalyse entwickelt. Abschnitt 6.1 zeigt später die Umsetzung an einem Beispiel.
In einem Gespräch müssen mit dem Kunden zunächst die Anforderungen an seine neue Webseite geklärt werden:

- Welche Zielgruppe möchte er ansprechen?
- Welche Botschaft möchte er vermitteln?
- Wie möchte er sich selbst/sein Unternehmen darstellen?
- Welche Reaktionen/Gefühle möchte er beim Kunden erzeugen?

Gegebenenfalls hat der Kunde bereits genaue bzw. teilweise Vorstellungen, anderenfalls müssen diese zusammen mit einem Consultant und Entwickler erarbeitet werden. Im Anschluss werden nichtfunktionale Anforderungen, aufgeteilt in Design, Usability und ggf. Sonstiges, definiert. Sowohl die Wirkung des Designs als auch die Usability sind testbar (siehe Abschnitt 5.6).
Im nächsten Schritt werden diese nichtfunktionalen Anforderungen in eine oder mehrere Layout-Entwürfe umgesetzt und dem Kunden vorgelegt. Evtl. müssen noch Änderungen vorgenommen werden, bis der Kunde den Entwurf abnimmt.

## 5.5 Praktiken in FDD

FDD definiert bereits acht Kernpratiken (siehe Abschnitt 3.2.4.3). Diese werden im Folgenden auf ihre Anwendbarkeit hin untersucht und näher beschrieben sowie wenn nötig angepasst.

### 5.5.1 Inspektionen

FDD unterscheidet zwei Arten von Inspektionen: die Entwurfsinspektion und die Codeinspektion. Erstere dient dazu, Fehler im Entwurf bzw. der Architektur des Systems zu erkennen. Mit der Codeinspektion sollen Bugs und ineffizienter Code erkannt werden.
Inspektionen sind als Ergänzung zu Tests (Unit Tests etc.) zu sehen, nicht als Ersatz oder Alternativmöglichkeit. Die Erfolgsquote beim Auffinden von Programmfehlern liegt für Inspektionen bei ca. 55-80%, die von Unit Tests wird dagegen nur mit ca. 24% angegeben ([1] S.49f).

Neben dem Aufdecken von Fehlern eignen sich Inspektionen auch für den Wissenstransfer innerhalb eines Teams. Indem erfahrene Entwickler ihre Techniken erklären, eignen sich weniger erfahrene neues Wissen und bessere Techniken an. Weiterhin helfen Inspektionen einen im Team aufgestellten Coding-Standard einzuhalten, denn Entwickler, die dies nicht tun, müssen damit rechnen, dass ihr Code bei einer Inspektion durchfällt und überarbeitet werden muss. Wenn die Ergebnisse von Inspektionen gespeichert und analysiert werden, lassen sich zudem häufige Fehlerquellen aufdecken und mit einer entsprechenden Änderung des Entwicklungsprozesses beseitigen.

Wichtig für eine erfolgreiche Durchführung von Inspektionen ist die Kultur. Entwickler sollten sich nicht angegriffen fühlen, wenn sie auf Fehler hingewiesen werden. Stattdessen sollten sie die Inspektion als ein Werkzeug zum Finden von Fehler und als Lernmöglichkeit auffassen.
Teilnehmen sollten nur die Entwickler, deren Code inspiziert wird bzw. die evtl. später daran arbeiten müssen. Der zu inspizierende Code wird ausgedruckt und von jedem Teilnehmer vor dem Meeting durchgeschaut, im Meeting selbst werden entsprechende Stellen dann durchgesprochen. Einer der Anwesenden führt dabei Protokoll.

### 5.5.2 Objektorientierte Modellierung des Anwendungsgebiets

FDD sieht zwei Arten der objektorientierten Modellierung vor. Zu Beginn wird ein grobes Klassendiagramm erstellt, das später weiter verfeinert wird. Zudem werden für jedes Feature Sequenzdiagramme erstellt.

Es ist vorgesehen, dass Fachexperten bei der Modellierung des Klassendiagramms anwesend sind und dieses auch mit gestalten können. Dafür ist jedoch eine Schulung der Fachexperten in UML notwendig, sofern sie keinen technischen Hintergrund haben. In der Agentur wäre ein solcher Aufwand gegenüber dem Nutzen nicht gerechtfertigt.
An die Stelle der reinen objektorientierten Modellierung sollte hier eine generelle Entwurfsphase treten, in der Klassendiagramme, DB-Schema, das Seitenlayout und eine Sitemap erstellt werden. Allerdings sind nicht bei jedem Projekt alle genannten Entwürfe nötig.
Erklärungsbedürftig sind aber evtl. Wirkungen von Farben und Elementen etc., wenn die Vorstellungen des Kunden nicht zu den vom ihm eigentlich gewollten Auswirkungen passen.

### 5.5.3 Entwicklung pro Feature

Da Webseiten und Shops mit Standardsoftware umgesetzt und nicht von Grund auf programmiert werden, kann das Feature-Template von FDD nur bedingt eingesetzt werden.

In Typo3 wird das Design einer Webseite durch Templates erstellt. Diese enthalten Marker, die Platzhalter für verschiedene einzufügende Seitenelemente darstellen. Die Definition der Marker wird über das Typo3 eigene TypoScript vorgenommen. Soll eine Webseite in mehreren Sprachen darstellbar sein, wird dies ebenfalls mit Markern realisiert.
In Typo3 können verschiedene Erweiterungen integriert werden (z.B. Newssystem, Bildergalerie etc.). Diese müssen nur einmal installiert werden und können danach beliebig oft in einzelne Seiten integriert werden.

In Oxid Shop wird das Design ebenfalls mit Templates basierend auf Smarty – einer Template Engine – realisiert. Diese werden nach Kundenwunsch angepasst. Über Erweiterungen können zusätzliche Funktionen bereitgestellt werden. Auch hier gibt es wie bei Typo3 bereits vorgefertigte Erweiterungen, individuelle Kundenwünsche müssen programmiert werden. Dabei werden entweder vorhandene Klassen und/oder Methoden überschrieben bzw. neue Klassen und/oder Methoden erstellt. Die dadurch bereitgestellte Funktionalität kann dann über die Templates in einzelne Seiten integriert werden.

Zusammenfassend lassen sich somit die folgenden vier Kategorien von Features für die Entwicklung mit Typo3 und Oxid Shop klassifizieren.

**Basiscodierung**

Die Basiscodierung umfasst die Erstellung von HTML-Seiten, die später als Vorlagen dienen. Normalerweise besteht diese aus einer Hauptseite und einer oder mehreren Vorlagen für die Folgeseiten.

**Standarderweiterungen**

Standarderweiterungen sind Erweiterungen für Typo3 oder Oxid, die bereits fertig erhältlich sind. Die Umsetzung besteht hier aus Installation, Konfiguration und Anpassung. Danach kann die Erweiterung so oft wie nötig in einzelne Seiten integriert werden.

**Individuelle Erweiterungen**

Diese werden nach Kundenwunsch programmiert. Hier kann das Feature-Template von FDD angewendet werden. Je nach Art der Erweiterung kann diese aus nur einem Feature bestehen oder aber mehrere enthalten. Im ersten Fall reicht die Aufzählung des Features aus, im zweiten Fall sollte ein Feature-Set <Name der Erweiterung> mit den zugehörigen Features erstellt werden.

**Sonstiges**

In diese Kategorie fallen alle Features, die sich nicht in die drei erstgenannten einordnen lassen, also beispielsweise eine Animation.

Abbildung 5.1 zeigt, wie nichtfunktionale Anforderungen über den Entwurf in die funktionalen Anforderungen (Features) einfließen können. Beispielsweise wird die Usability-Anforderung „Logo und Menü sind das erste, was der Kunde sehen soll“ durch einen entsprechend gestalteten Layout-Entwurf umgesetzt. Dieser Entwurf wird dann wiederum mit der Basiscodierung in HTML o.ä. umgesetzt.

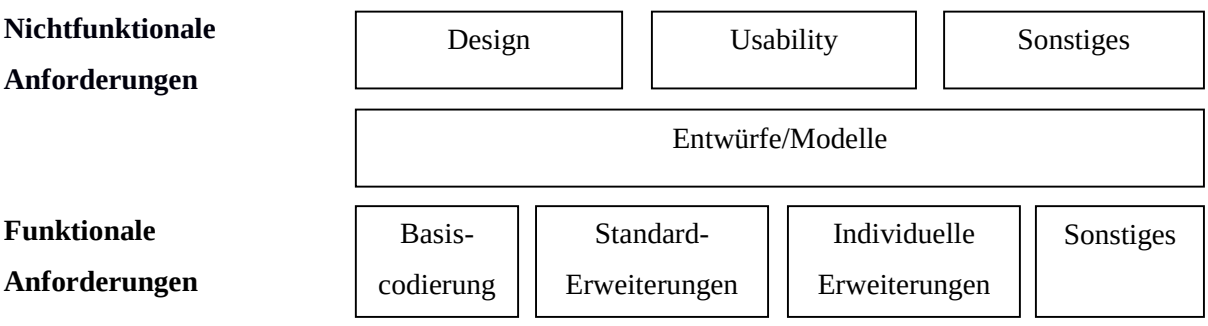


Abbildung 5.1: Einfließen nichtfunktionaler in funktionale Anforderungen

### 5.5.4 Individueller Codebesitz

Ein individueller Codebesitz ist nicht notwendig, da die Arbeit größtenteils mit Frameworks erfolgt und neue Methoden/Klassen recht schnell geschrieben bzw. geändert sind. Eine der Ideen hinter dem Codebesitz kann auch auf Features angewendet werden: derjenige, dem das Feature zugewiesen wird, kümmert sich auch um alle Tests, die dafür anstehen.

### 5.5.5 Feature-Teams

Normalerweise wird jeder Entwickler Besitzer einer oder mehrerer Klassen. Für ein Arbeitspaket stellt jeder Chefprogrammierer dann ein Team derjenigen Entwickler zusammen, die die notwendigen Klassen besitzen. Da in der Agentur jeder Entwickler zumeist an einem eigenen Teil des Projekts arbeitet, ist auch ein Ein-Mann-Feature-Team (vgl. [18]) möglich, d.h. jeder bekommt sein eigenes Arbeitspaket. Der Chefprogrammierer fungiert dann als Mentor, plant die Aufgaben und führt mit den Entwicklern die Inspektionen durch.

### 5.5.6 Regelmäßige System-Builds

In regelmäßigen Zeitabständen wird das Gesamtsystem zusammengebaut und kompiliert um festzustellen, ob bei der Integration Fehler auftreten und um dem Kunden ein lauffähiges System präsentieren zu können.
Da in der Agentur Webauftritte mit HTML, PHP etc. erstellt werden, die nicht kompiliert werden müssen, ist diese Praktik am ehesten mit dem Aufsetzen bzw. Aktualisieren des bisher entwickelten Systems auf einem Entwicklungsserver vergleichbar.

### 5.5.7 Configuration Management

Unter Configuration Management wird die Versionsverwaltung von allen Projektdokumenten verstanden. Dazu gehört nicht nur der Code, sondern auch Entwürfe, Testfälle, Verträge etc. Außer Subversion als Versionsverwaltung ist zudem ein Wiki sinnvoll, von dem nötige Dokumente aufgerufen werden können.

### 5.5.8 Sichtbarkeit des Fortschritts

FDD setzt auf eine grafische Übersicht, die sog. „Parking Lot Chart“, um den Fortschritt gegenüber Kunden oder Vorgesetzten zu präsentieren. Der Name kommt daher, dass die Anordnung der einzelnen Felder zum Fortschritt der Feature Sets einem Parkplatz ähnelt.

Ein Feld symbolisiert ein Feature Set. Einzelne Felder können wiederum zu einem Major Feature Set zusammengefasst werden (Umrahmung).
Jedes Feld besteht aus drei Teilen. Im oberen Teil steht die Bezeichnung des Feature Sets sowie in Klammern die zugehörige Anzahl der Features. Darunter wird die Fertigstellung des Feature Sets in Prozent angezeigt. Je nach Status des Feature Sets ist dieser Teil mit einer von vier Farben eingefärbt:

- Weiß: Die Arbeit am Feature Set wurde noch nicht begonnen.
- Blau: Das Feature Set ist in Arbeit.
- Rot: Der Fälligkeitstermin des Feature Sets ist bereits überschritten.
- Grün: Das Feature Set ist fertig gestellt.

Den mittleren Teil bildet ein Fortschrittsbalken, der in grüner Farbe anzeigt, wie viel des Feature Sets bereits erledigt wurde.
Im unteren Teil befindet sich das Fälligkeitsdatum des Feature Sets auf weißem Grund. Bei Fertigstellung wird dieser Teil grün eingefärbt. Ein fertig gestelltes Feature ist somit sofort an einem komplett grünen Feld erkennbar.

Abbildung 5.2 zeigt ein Beispiel einer Parking Lot Chart. Eine vergrößerte Darstellung eines Feature Sets ist in Abbildung 5.3 zu sehen.

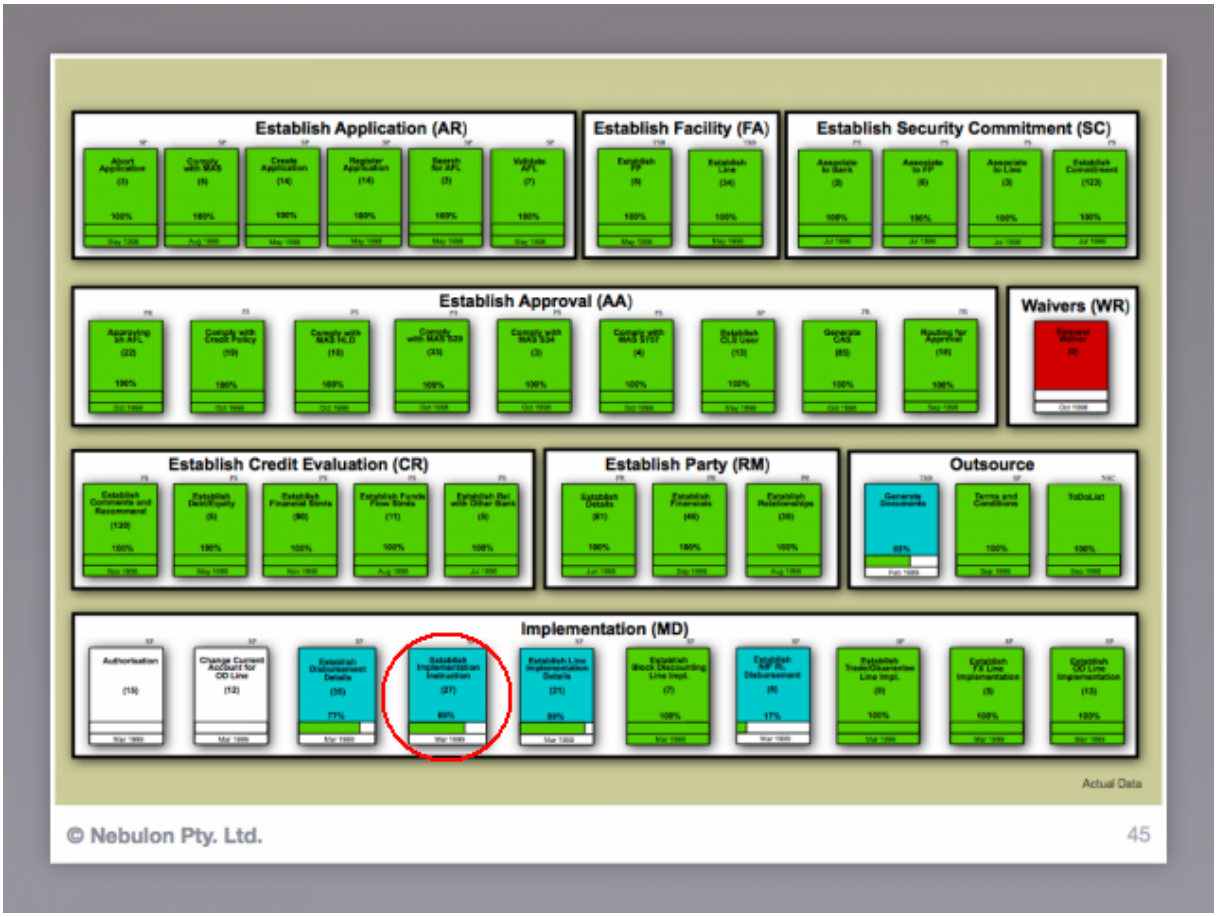


Abbildung 5.2: Parking Lot Chart [25]

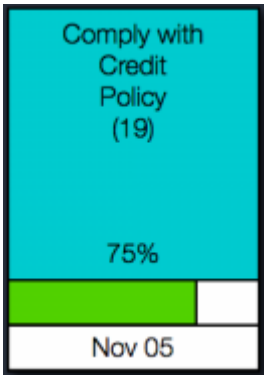


Abbildung 5.3: Darstellung eines Feature Sets, Ausschnitt aus [26]

Bei automatischer Generierung bietet diese grafische Art des Fortschrittsberichts einen schnellen Überblick über das gesamte Projekt.

Normalerweise unterteilt FDD ein Feature noch in die einzelnen Arbeitsschritte der Phasen 4 (Entwirf pro Feature) und 5 (Konstruiere je Feature). Jedem Arbeitsschritt wird ein Prozentwert zugeordnet, der den Anteil an der Gesamtentwicklungsdauer des Features widerspiegelt. Daraus lässt sich natürlich ein sehr genauer Bericht erstellen, zu wie viel Prozent ein Projekt tatsächlich fertig gestellt ist. Sinnvoll ist das aber nur bei großen Projekten und wenn sich die Bearbeitungszeit eines Features in Tagen und

nicht in Stunden misst, wie das zumeist in der Internetagentur der Fall ist. Daher sollte in nahezu allen Projekten eine Fortschrittsmessung auf Basis einer Parking Lot Chart ausreichend sein.

## 5.6 Qualitätsmanagement

Neben den in Abschnitt 5.5.1 beschriebenen Inspektionen setzt FDD auf Unit Tests zur Qualitätssicherung. Allerdings reichen Unit Tests allein nicht aus, deshalb sollten insgesamt die im Folgenden erläuterten Tests durchgeführt werden.

**Selenium Test**
Selenium ist ein Framework zum automatischen GUI-Testen von Webanwendungen. Es wurde von dem Unternehmen ThoughtWorks entwickelt und ist als freie Software erhältlich. Mit diesem Framework ist es möglich, Testskripte zu erstellen, die eine Webseite überprüfen. Es können Elemente lokalisiert und gewünschtes gegen tatsächliches Verhalten getestet werden. [14]

**Unit Test**
Der Unit Test prüft einzelne Komponenten z.B. Klassen oder Methoden auf ihre Fehlerfreiheit. Diese Art von Tests kann auch automatisiert durchgeführt werden. Außerdem gibt es spezielle Software, die das Schreiben, Verwalten und Durchführen von Unit Tests erleichtert, z.B. PHPUnit.

**Markt- und Meinungsforschung**
Mit diesem Test wird geprüft, welche Wirkung ein Layout oder bestimmte Farben, Elemente etc. auf potentielle Kunden haben. Es können Testpersonen beispielsweise mehrere Designvorschläge vorgelegt werden oder auch ohne Angabe eines Unternehmensnamen eine Webseite, die die Testperson mit einer Branche assoziieren soll. Ebenso kann getestet werden, welche Reaktionen ein Layout hervorruft und welche Unternehmenswerte damit transportiert werden. Da jeder Mensch subjektiv wahrnimmt, kann das Ergebnis nie perfekt sein, es kann jedoch sichergestellt werden, dass der Internetauftritt zumindest zu einem bestimmten Prozentsatz die beabsichtigte Wirkung auf die gewünschten Zielgruppen hat.

**Usabilitytest**
Mit einem Usabilitytest wird geprüft, ob Vorgänge intuitiv und benutzerfreundlich durchgeführt werden können, z.B. ob ein Benutzer in einen Online-Shop sofort den Warenkorb findet und den Bestell- und Bezahlvorgang ohne Probleme vornehmen kann.

Bisher werden die Tests in einem separaten Punkt in der Leistungsbeschreibung abgerechnet. Das kann dazu führen, dass der Kunde diesen Posten ablehnt, da er für ihn zusätzliche Kosten verursacht. Problem ist hierbei aber, dass der Kunde trotzdem erwartet, dass seine Software fehlerlos ist. Besser wäre es daher, wenn die Kosten für Tests auf die einzelnen Anforderungen aufgeschlagen würden, um dieses Kundenverhalten zu vermeiden.

## 5.7 Kommunikations- und Prozessverbesserung

Neben den in FDD vorgesehenen Praktiken eignen sich insbesondere auch zwei aus Scrum entlehnte Praktiken für die Internetagentur. Diese verbessern die Kommunikation im Team und schaffen außerdem einen festen Termin, an dem über Prozessverbesserung im Team gesprochen wird.

### 5.7.1 Standup Meeting

Das Standup Meeting ist aus Scrum entnommen und sorgt für eine bessere Kommunikation und Abstimmung der Teammitglieder. Es soll max. 15 Minuten dauern und wird im Stehen und möglichst jeden Tag zur gleichen Zeit abgehalten. Dabei beantwortet jedes Teammitglied folgende Fragen:

- Bist du mit dem fertig geworden, was du dir bis zu diesem Meeting vorgenommen hast?
- Was wirst du bis zum nächsten Meeting tun?
- Gibt es ein Problem, das deine Arbeit behindert?

Dadurch, dass jeder Antwort auf diese Fragen vor allen Teammitgliedern geben muss, wird es unwahrscheinlicher, dass jemand seine Arbeit nur unzureichend erledigt, denn niemand möchte vor seinen Arbeitskollegen schlecht da stehen. Gleichzeitig kann aber jedes Teammitglied um Hilfe bitten und sollte diese dann auch gewährt bekommen. Im Meeting selbst werden nur diese drei Fragen beantwortet. Kann jemand bei einem Problem helfen oder sind zu einem Thema Diskussionen nötig, dann setzen sich die betreffenden Teammitglieder nach dem Meeting zusammen.

### 5.7.2 Retrospektive

Die Retrospektive wird sinnvollerweise nach Abschluss eines Projektes bzw. einer Projektphase durchgeführt und dient der Prozessverbesserung bei zukünftigen Projekten. In einem Rückblick auf

das letzte Projekt/die letzte Projektphase wird analysiert, was gut funktioniert hat und was besser gestaltet werden könnte.
Im Vorfeld der Retrospektive bereitet sich jedes Teammitglied vor und es werden Daten und Informationen über das zurückliegende Projekt gesammelt. Im Meeting selbst sollte darauf geachtet werden, dass jeder zu Wort kommt und seine Ansichten und Verbesserungsvorschläge vorbringen kann. Ein Moderator ist hierbei von Vorteil, damit Schuldzuweisungen und persönliche Angriffe verhindert werden. Für die Erarbeitung von Verbesserungsmöglichkeiten eignen sich Kreativtechniken wie Brainstorming, Mindmaps u.a. Als Abschluss der Retrospektive werden gemeinsam konkrete Maßnahmen beschlossen. Ggf. muss die zuständige Führungsperson hinzugezogen werden, damit diese Maßnahmen dann auch umgesetzt werden können. [6] S. 35ff

## 5.8 Sonstige Optimierungsmöglichkeiten

**Externer Mediengestalter**

Bei Einsatz eines externen Mediengestalters sollten diesem zuvor die verwendbaren Elemente sowie deren gestalterische Möglichkeiten und Grenzen erläutert werden. Die Entwürfe werden dadurch realistischer und leichter umsetzbar.

**Vertrieb**

Für den Vertrieb sollten regelmäßig Schulungen über die Möglichkeiten der verwendeten Standardsoftware und Eigenentwicklungen der tro:net durchgeführt werden. Vorteil dieser Schulungen ist, dass sich der Vertrieb besser mit den eigenen Produkten auskennt und dadurch den Kunden besser beraten kann.

**Kommunikation mit dem Kunden**

Um das bestmögliche Projektergebnis zu erzielen, sollte dem Kunden die Vorgehensweise bei der Entwicklung erklärt und ihm außerdem bewusst gemacht werden, dass er, um das für ihn bestmögliche Ergebnis zu erhalten, selbst aktiv mitarbeiten muss.

## 5.9 Rollen

Von den in FDD vorgesehenen drei Rollengruppen (siehe Abschnitt 3.2.4.1) ist nur die der Schlüsselrollen relevant, da das Team der Agentur sehr klein ist. Die sechs Schlüsselrollen

- Projektmanager
- Chefarchitekt
- Entwicklungsleiter
- Chefprogrammierer
- Klassenbesitzer
- Fachexperte

lassen sich wie in der folgenden Tabelle dargestellt auf das Team der Agentur bzw. den Kunden verteilen.

| Rolle in der Agentur | Rolle in FDD |
|---|---|
| Teamleiter | Chefarchitekt, Chefprogrammierer, Entwicklungsleiter, Projektmanager für die Detailplanung |
| Projektmanager | Projektmanager für die Meilensteinplanung |
| Entwickler<br>(= Anwendungsentwickler/Mediengestalter) | Klassenbesitzer |
| Kunde | Fachexperte |

## 5.10 Ablauf

In Anlehnung an [19] ist im Folgenden der komplette Ablauf in fünf einzelnen Prozessen beschrieben. Da in FDD die Anforderungsanalyse nicht enthalten ist, stellt sie hier den ersten Prozess dar. Prozess 4 und 5 des ursprünglichen FDD-Prozessablaufs sind zu einem zusammengefasst.

**Prozess 1 - Anforderungsanalyse**

| | | |
|---|---|---|
| Anforderungsanalyse | Projektmanager, Teamleiter, Entwickler | Notwendig |
| Zusammen mit dem Kunden werden die Anforderungen ermittelt. Dazu gehören insbesondere Anforderungen an das Design und die Usability, die Zielgruppe, die zu vermittelnde Botschaft und die gewünschten Funktionen. Unbekannte Anforderungen müssen von den Entwicklern abgeschätzt werden, damit diese entsprechend bei der Meilensteinplanung und im Vertrag berücksichtigt werden können. | | |
| Meilensteinplanung | Projektmanager, Kunde | Notwendig |
| Beginn und Ende, sowie wichtige Meilensteine des Projekts werden vereinbart. | | |

**Prozess 2 – Gesamtmodell**

| | | |
|---|---|---|
| Dokumentenstudium, Analyse des bestehenden Internetauf-tritts des Kunden | Projektmanager, Teamleiter, Entwickler | Optional |
| Vom Kunden zur Verfügung gestellte Dokumente und ein evtl. bereits bestehender (und zu ersetzender) Internetauftritt des Kunden werden analysiert. | | |
| Entwurf | Teamleiter, Entwickler | Notwendig |
| Es werden Klassendiagramm, DB-Schema, Sitemap, Seitenlayout und etwaige sonstige Modelle/ Entwürfe erstellt. Die Art des Projekts bestimmt, welche Modelle nötig sind.<br>Insbesondere Seitenlayout und Sitemap werden mit dem Kunden abgestimmt und evtl. in mehreren Iterationen schrittweise verbessert, bis der Kunde diese abnimmt. | | |

**Prozess 3 - Erstellen der Featureliste**

| Erstellen der Featureliste | Teamleiter | Notwendig |
|---|---|---|
| Der Teamleiter erstellt aus den Anforderungen und Modellen aus Prozess 1 und 2 eine Featureliste. Zusammengehörige Features (z.B. bei einem selbst zu entwickelnden Modul) werden zu einem Feature Set zusammengefasst.<br><br>Zu programmierende Features werden in der Form <Aktion> <Ergebnis> <Objekt> beschrieben. Beispiel: Berechne Summe eines Verkaufs | | |

**Prozess 4 - Planung je Feature**

| Entwicklungsreihenfolge festlegen | Teamleiter | Notwendig |
|---|---|---|
| Die Entwicklungsreihenfolge der Features aus Prozess 3 wird festgelegt und jedem Feature eine Fälligkeit zugewiesen. Komplexe oder risikobehaftete Features sollten, soweit es die Abhängigkeiten zwischen den Features erlauben, vorgezogen werden. | | |
| Zuweisung der Features | Teamleiter | Notwendig |
| Der Teamleiter weist sich selbst und jedem Entwickler Features zur Bearbeitung zu. Die Arbeitsbelastung der Entwickler sollte über alle Projekte möglichst gleichmäßig sein. Außerdem sollten zu häufige Kontextwechsel der Entwickler vermieden werden, um (Wieder-)Einarbeitungszeiten zu verringern. | | |

**Prozess 5 - Entwurf und Entwicklung pro Feature**

| Rücksprache mit dem Kunden | Entwickler oder Projektmanager, Kunde | Optional |
|---|---|---|
| Sind weitere Details zu einem Feature oder Änderungen nötig, wird Rücksprache mit dem Kunden gehalten. | | |

| | | |
|---|---|---|
| Modelle verfeinern | Teamleiter, Entwickler | Optional |
| Das Modell wird auf Basis gewonnener zusätzlicher Informationen verfeinert. | | |
| Implementierung | Teamleiter, Entwickler | Notwendig |
| Das Feature wird implementiert/codiert. | | |
| Tests | Teamleiter, Entwickler | Notwendig |
| Es werden alle notwendigen Tests durchgeführt um zu prüfen, ob das Feature sämtliche Anforderungen ohne Fehler erfüllt. In regelmäßigen Abständen und am Ende des Projekts werden Systemtests durchgeführt. | | |
| Inspektion | Teamleiter, Entwickler | Notwendig |
| Die Inspektion prüft zusätzlich zu den Tests die Fehlerfreiheit des Systems. Sie kann auch vor den Tests durchgeführt werden. | | |

Nachdem in diesem Kapitel eine agile Methode ausgewählt und an die Gegebenheiten der Internetagentur angepasst wurde, soll im folgenden Kapitel an einem kleinen Beispiel die Umsetzung der Anforderungsanalyse und des Feature-Aspekts gezeigt werden.

# 6 Beispiel einer Umsetzung

Anhand eines kleinen Typo3-Projekts soll hier die Umsetzung der im fünften Kapitel vorgestellten Anforderungsanalyse und des Feature-Aspekts im Webdesign gezeigt werden. Der Kunde ist ein Unternehmen aus der Maschinenbaubranche und stellt Maschinen für die industrielle Fertigung her. Da der Kunde anonym bleiben soll, ist dessen Name durch *„Kundenunternehmen“* ersetzt und das Logo in den Screenshots geschwärzt worden.

## 6.1 Anforderungsanalyse

Der Kunde möchte mit seiner Webseite eine Botschaft an eine bestimmte Zielgruppe vermitteln, diese ist vom Kunden wie folgt definiert worden:

Zitat aus dem Angebot

Zielgruppe:
Einkäufer oder technische Angestellte auf der Suche nach einem Partner für eine individuelle Fertigungslösung.

Botschaft:
*Kundenunternehmen* ist ein modernes Unternehmen, was mit bewährten Techniken nachhaltige Lösungen schafft. Mit Liebe zum Produkt und Detail sowie mit einem Fokus auf Energieeffizienz. Wer *Kundenunternehmen*-Produkte kennt, weiß dass der Preis stimmt.

Folgende Ideensammlung wurde von einem Berater des Kundenunternehmens zur Verfügung gestellt:

Wortlaut Kunde

Identitätsmerkmale:

1. Produktpreis:
nicht billig, aber preiswert

hochwertig => PreisWERT
Spruch: wer unsere Sachen anguckt, der weiß dass der Preis stimmt
Preis sollte auch Qualität/Nachhaltigkeit reflektieren.
Konkurrenz nicht nachhaltig/Problem zur Preiserkennung -> Projekte sind nicht transparent (NACHFOLGEKOSTEN)

2. Qualität
Qualität gekennzeichnet durch
Taktzahlen
Produktionsdauer
Idee: Puls als Zeichen für Dauerhaftigkeit
Retooling

3. Design
Spruch: Was schön ist, funktioniert auch
Spruch: Boah, die ist aber schön (Reaktion des Kunden)
Schöne Maschinen/abgerundete Kanten inklusive

4. Gebrauch
Energiesparpotential
Geballte Energie
Kunden einbeziehen, gemeinsames Entwickeln, Benutzer soll Maschine lieben, maßgeschneidert, individuell
Nachhaltigkeit => Spruch: Ich möchte nicht, dass meine Technik irgendwann in der Tonne liegt
Idee: Rätsel Makro von mechanischen Bauteilen (Nippel) zeigt Detailliebe, ist attraktiv

Besondere Merkmale für Produkte und Unternehmen (USP)
FASZINATION!!!
Ausgelöst durch:
Perfektionismus (positive) = mechanisch perfekt (Vorteil Mechanik: sicherer, elektrisch: unsicher)
Die Idee dahinter: Ein Schlosser mit einer Werkzeugkiste kann die Maschinen selbst reparieren
Faszination/USP Klein aber fein, groß können alle
Idee: Makro Schmiernippel

KNOW HOW
Patente

Sprüche aufgefangen:
Uns möchten Sie gerne zuhören.
Wir suchen uns unsere Kunden selber aus.
Wer unsere Sachen anguckt, der weiß dass der Preis stimmt.
Was schön ist, funktioniert auch.

Besondere Merkmale für Produkte und Unternehmen (USP)
Unternehmenspersönlichkeit
= Website muss ergeben
Faszination (wird über feinmechanisches Detail dargestellt)
-> Bewegung -> Perpetuanz -> Querdenken -> zurück zu Werten

Wichtige Schlüsselworte:
Faszination Technik, nachhaltig, schön, funktional, Werte

### 6.1.1 Anforderungen an das Design

Aus dieser Ideensammlung lassen sich Anforderungen an das Design ableiten.

**Werte/Attribute die das Unternehmen ausstrahlen soll:**

- Qualität
- Nachhaltigkeit
- Schönheit des Designs
- Funktionalität
- Faszination
- Perfektionismus
- Know How
- Dauerhaftigkeit
- Individualität
- Kundenorientierung
- Liebe zu Produkten
- energiesparend
- preiswert
- hochwertig

**Ausgelöste Reaktionen/Gefühle beim Kunden:**

- Der Preis stimmt
- Die Maschine ist aber schön
- Was schön ist, funktioniert auch
- Kleines aber feines Unternehmen

### 6.1.2 Anforderungen an die Usability

Hinzu kommen Anforderungen an die Usability, die abhängig von der Branche, Art des Unternehmens, sowie der Botschaft und den vermittelten Werten sind:

- Startseite dient der Neukundengewinnung, Eyecatcher soll neugierig auf das Unternehmen machen
- Folgeseiten dienen der Informationsvermittlung und sind insbesondere auf die Bestandskunden ausgerichtet
- Klar strukturierte Darstellung von Informationen
- Klare Menüführung
- Das Menü befindet sich auf jeder Seite an der gleichen Position
- Texte müssen gut lesbar sein (schwarze Schrift auf weißem Hintergrund)
- Logische Informationsstruktur (Hauptpunkte, wenige Unterpunkte)
- Logo und Menü sind das erste, was der Kunde sehen soll
- Die Schriftgröße gibt die Priorität der Informationen an

Als sonstige Anforderungen wurden seitens des Kunden Bilder und eine Sitemap zur Verfügung gestellt, die in die Webseite einzuarbeiten sind. Außerdem soll das Corporate Design beibehalten werden.

## 6.2 Layout-Entwurf

Der Mediengestalter entwirft nun nach Maßgabe dieser Anforderungen ein Layout für die Webseite. Die Startseite soll als Eyecatcher dienen und bekommt daher ein gesondertes Layout, während sämtliche Folgeseiten zwar jeweils andere Informationen enthalten werden, aber vom Grundaufbau her gleich sind. Es entstanden folgende Entwürfe für die Startseite und die Folgeseiten der Webseite.


Abbildung 6.1: Die Startseite

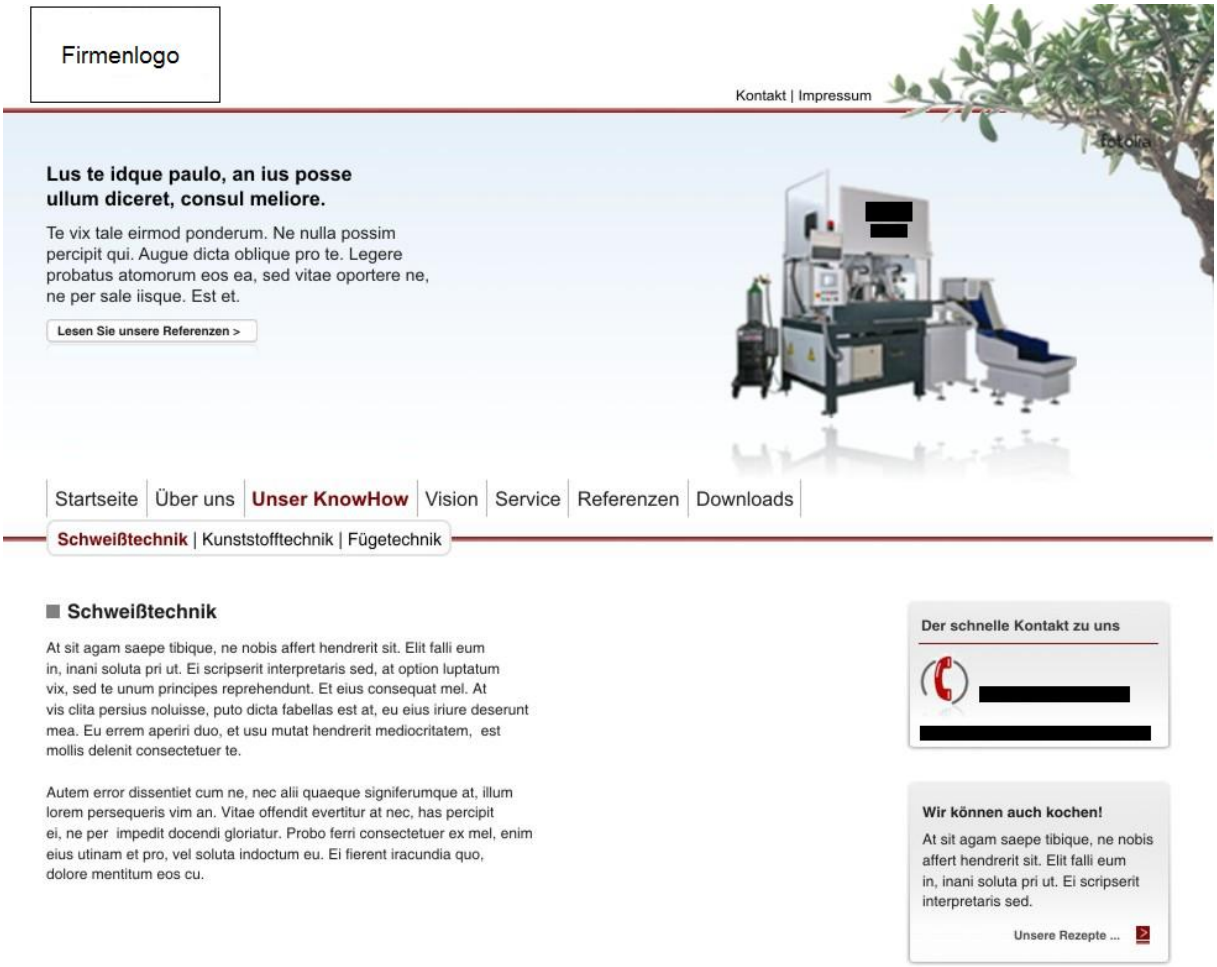


Abbildung 6.2: Die Folgeseiten

## 6.3 Features

Da das Projekt vom Umfang her ein recht kleines ist, gibt es bei diesem auch nur wenig Features.

**Basiscodierung**

Es sind insgesamt zwei Vorlagen zu erstellen:

- Startseite
- Folgeseiten

**Erweiterungen in Typo3**

- Bildergalerie
- Google Maps

**Sonstiges**

- Animation für die Webseite, die den Ablauf von Bestellung bis Auslieferung zeigt

# 7 Fazit

## 7.1 Ergebnis

Jedes Unternehmen ist individuell zu betrachten: es gelten andere Voraussetzungen, es werden unterschiedliche Projekte umgesetzt, es wird mit verschiedenen Techniken gearbeitet etc. Daher muss auch für jedes Unternehmen ein individuell maßgeschneidertes agiles Vorgehensmodell entwickelt werden. Es ist hier empfehlenswert, zunächst ein existierendes Modell als Basis zu nutzen, das am besten zum Unternehmen und den mit der Prozessverbesserung verbundenen Zielen passt und dieses in einem zweiten Schritt anzupassen.

Zunächst habe ich eine vollständige Ist-Analyse der bestehenden Geschäftsprozesse der Internetagentur vorgenommen und eine umfassende Dokumentation angefertigt. Auf Basis dieser Arbeit war von mir ein auf die Agentur zugeschnittenes agiles Modell zu entwickeln. Hierbei hat sich gezeigt, dass kein agiles Vorgehensmodell ohne besondere Anpassung auf die Agentur der tro:net übertragen werden konnte. Hauptprobleme hierbei waren die veränderten Anforderungen des Webdesigns gegenüber der reinen Anwendungsentwicklung und die Notwendigkeit der Berücksichtigung von Festpreisverträgen, die bei agilen Vorgehensmodellen nicht die Regel sind.

Ich habe dafür die in dieser Arbeit vorgestellten agilen Methoden im ersten Schritt hinsichtlich ihrer Einführbarkeit in der Agentur bewertet sowie die Eignung zur Lösung der vorhandenen strukturellen Probleme – kein Entwurf, mangelhaftes Qualitätsmanagement, kaum Einbeziehung der Entwickler bei der Analyse – geprüft. Nach Rücksprache mit der Geschäftsführung und dem Teamleiter habe ich mich für FDD als Basis entschieden. In einem zweiten Schritt habe ich dieses Modell bezüglich des generellen Ablaufs, der Aufteilung in Features und der Testarten an die Gegebenheiten der Agentur angepasst. Zusätzlich waren von mir noch eine Anforderungsanalyse zu entwickeln, die in FDD nicht enthalten ist, sowie Maßnahmen zur Verbesserung der Kommunikation im Team aufzuzeigen.
Ein Problem, das von mir nicht gelöst werden konnte, ist die Kommunikation mit schwierigen Kunden. FDD gibt einige Praktiken vor, wie Kunden in den Entwicklungsprozess eingebunden werden können und sich damit das Projektergebnis verbessern lässt. Dies setzt allerdings voraus, dass der Kunde auch willens ist, entsprechend mitzuarbeiten, was nicht bei allen der Fall ist.

Agile Vorgehensmodelle sind insbesondere für kleine Unternehmen interessant, die einen geringen Overhead an Dokumenten wünschen und für die daher traditionelle Modelle nicht geeignet sind. Agile Modelle können hier einen sehr guten Kompromiss zwischen gar keinem und zuviel Prozess bieten,

mit dem zusätzlichen Vorteil, besser auf Änderungen von Anforderungen reagieren zu können. Dies gilt allerdings immer unter der Voraussetzung, dass ein solches Modell angepasst und auch in Zukunft im Zuge von Prozessverbesserungsmaßnahmen regelmäßig gegen aktuelle Anforderungen geprüft und wenn nötig, geändert wird.

## 7.2 Ausblick

### 7.2.1 Einführung des Modells

Die Einführung des Modells sollte stufenweise erfolgen:

- Standup Meetings und Retrospektive
- Entwicklung pro Feature
- Tests und Inspektionen
- Vollständiges Modell

Die Kommunikationsmaßnahmen stehen hier bewusst ganz am Anfang, denn im Zuge der Einführung eines neuen Modells sollten Fortschritte und Probleme offen diskutiert werden.

### 7.2.2 Mögliche Anschlussprojekte und offene Probleme

**Entwicklung eines Projektmanagementtools**

Die vorhandenen Tools für FDD sind nicht ausreichend, um den kompletten Entwicklungsprozess der Internetagentur der tro:net abzudecken. Vor allem fehlt hier die Interaktion mit dem Kunden, dem es wie beim momentan eingesetzten Mantis-System möglich sein soll, selbst Anfragen zu verfassen, Fehler zu melden und so auf den Entwicklungsprozess Einfluss zu nehmen.

Herkömmliche Projektmanagementtools können dagegen die Entwicklung pro Feature nur unzureichend abdecken und bieten nicht die Möglichkeit, Fortschrittsberichte wie in FDD automatisch zu generieren.

Die Lösung ist hier ein maßgeschneidertes Tool, das die Vorteile beider Systemvarianten vereint: die Darstellung der Entwicklung pro Feature, die automatische Erstellung von Fortschrittsberichten und die Interaktion mit dem Kunden.

**Erstellung eines Workflows**

Mit Intalio ist es möglich, Geschäftsprozesse grafisch abzubilden und diese dann in einem Workflow ablaufen zu lassen. Zudem lassen sich Services aufrufen und Formulare erstellen und einbinden. Die Community Edition von Intalio ist frei verfügbar.

**Die Mitwirkung des Kunden**

Wie bereits im Abschnitt 7.1 erwähnt, ist ein wichtiges aber noch ungelöstes Problem, wie Kunden zur aktiven Mitarbeit motiviert werden können. Einerseits gibt es durchaus experimentierfreudige Kunden, die gerne mitarbeiten und auch wissen, dass dies für einen größtmöglichen Projekterfolg nötig ist. Andererseits gibt es aber auch Kunden, die nur einen Auftrag erteilen und mit der Projektdurchführung nichts zu tun haben wollen. Die tatsächlichen Bedürfnisse des Kunden sind für den Auftragnehmer dann weitestgehend unbekannt und nicht selten ist der Kunde am Ende unzufrieden mit dem Ergebnis, lastet dies aber dem Auftragnehmer an, ohne sich bewusst zu sein, dass auch er eine Mitwirkungspflicht hat.

Verfechter agiler Methoden tendieren dazu, Verträge mit schwierigen Kunden bzw. potenziell schwierigen Kunden gar nicht erst abzuschließen. Aus finanziellen Gründen können sich aber gerade kleinere Unternehmen das oftmals nicht leisten. Agile Methoden geben zwar viele Praktiken vor, die die Zusammenarbeit und Kommunikation zwischen Kunde und Auftragnehmer verbessern, zeigen aber nicht auf, wie der Kunde dazu gebracht werden kann, ebenfalls agil vorzugehen. Die Lösung dieser Problematik wäre also ein wichtiger Schritt hin zu einer höheren Quote an erfolgreichen IT-Projekten.

# A Anhang

Im Folgenden sind die Anliegen Projektanforderung, Serviceanforderung, Angebotsanforderung und das Oberanliegen Fehler aus Mantis (vgl. Abschnitt 4.3.1) sowie die beiden Teilprozesse Analyse und Bearbeitung in Aktivitätsdiagrammen (vgl. Abschnitt 2.2) dargestellt. Die dazugehörigen Tabellen listen zu den einzelnen Aufgaben die dazugehörigen Arbeitsschritte (wenn vorhanden) und Akteure auf.

## A.1 Projektanforderung

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Ticket erstellen | • Kategorie festlegen<br>• Beschreibung und Zusammenfassung eintragen<br>• Evtl. Abhängigkeit eintragen | Projektmanagement |
| Einplanung Analyse | • Plan-/Restzeit und Fälligkeit eintragen<br>• private Notiz anlegen<br>• Bearbeiter zuweisen<br>• Status auf „Analyse“ setzen | Projektmanagement |
| Teilprozess Analyse | | Anwendungsentwicklung/ Mediengestaltung |
| Einplanung | • Plan-/Restzeit und Fälligkeit eintragen<br>• Bearbeiter zuweisen<br>• Status auf „Umsetzung“ setzen | Projektmanagement |
| Teilprozess Bearbeitung | | Anwendungsentwicklung/ Mediengestaltung |
| Überprüfung | • Notiz anlegen | Projektmanagement |
| Überprüfung (Kunde) | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Ticket abschließen | • Status auf „Geschlossen“ setzen | Projektmanagement |

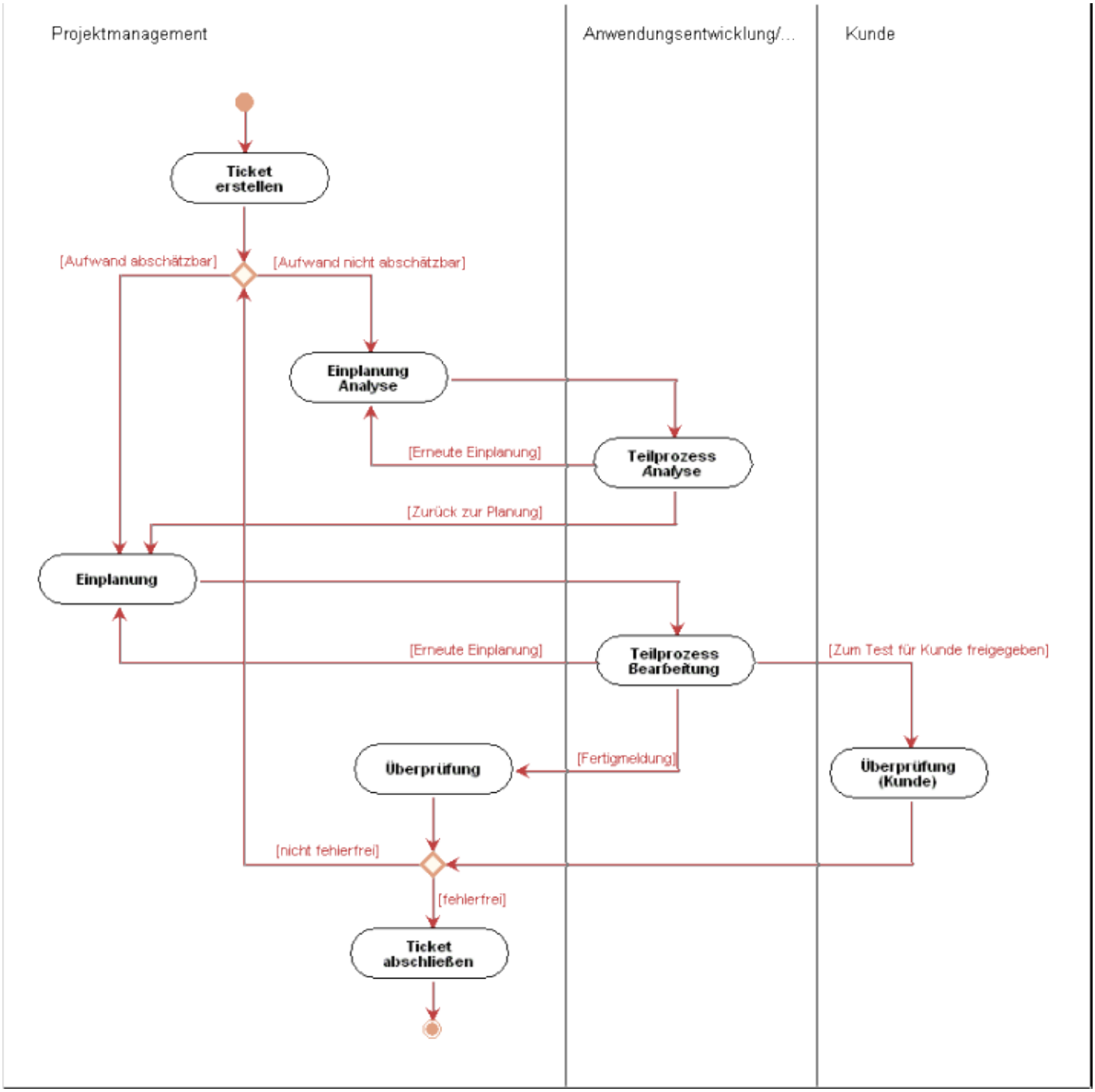


Abbildung A.1: Projektanforderung

## A.2 Serviceanforderung

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Ticket erstellen | • Kategorie festlegen<br>• Beschreibung, Zusammenfassung eintragen | Projektmanagement, Kunde |
| Überprüfung des Tickets | • Evtl. Notiz anlegen und Status auf „Erfordert Rückmeldung“ setzen | Projektmanagement |
| Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Einplanung Analyse | • Plan-/Restzeit, Fälligkeit und besondere Abrechnungsinfo eintragen<br>• Private Notiz anlegen<br>• Bearbeiter zuweisen<br>• Status auf „Analyse“ setzen | Projektmanagement |
| Teilprozess Analyse | | Anwendungsentwicklung/ Mediengestaltung |
| Einplanung | • Plan-/Restzeit, Fälligkeit und besondere Abrechnungsinfo eintragen<br>• Bearbeiter zuweisen<br>• Status auf „Umsetzung“ setzen | Projektmanagement |
| Teilprozess Bearbeitung | | Anwendungsentwicklung/ Mediengestaltung |
| Überprüfung | • Notiz anlegen | Projektmanagement |
| Überprüfung (Kunde) | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Ticket abschließen | • Status auf „Geschlossen“ setzen | Projektmanagement |

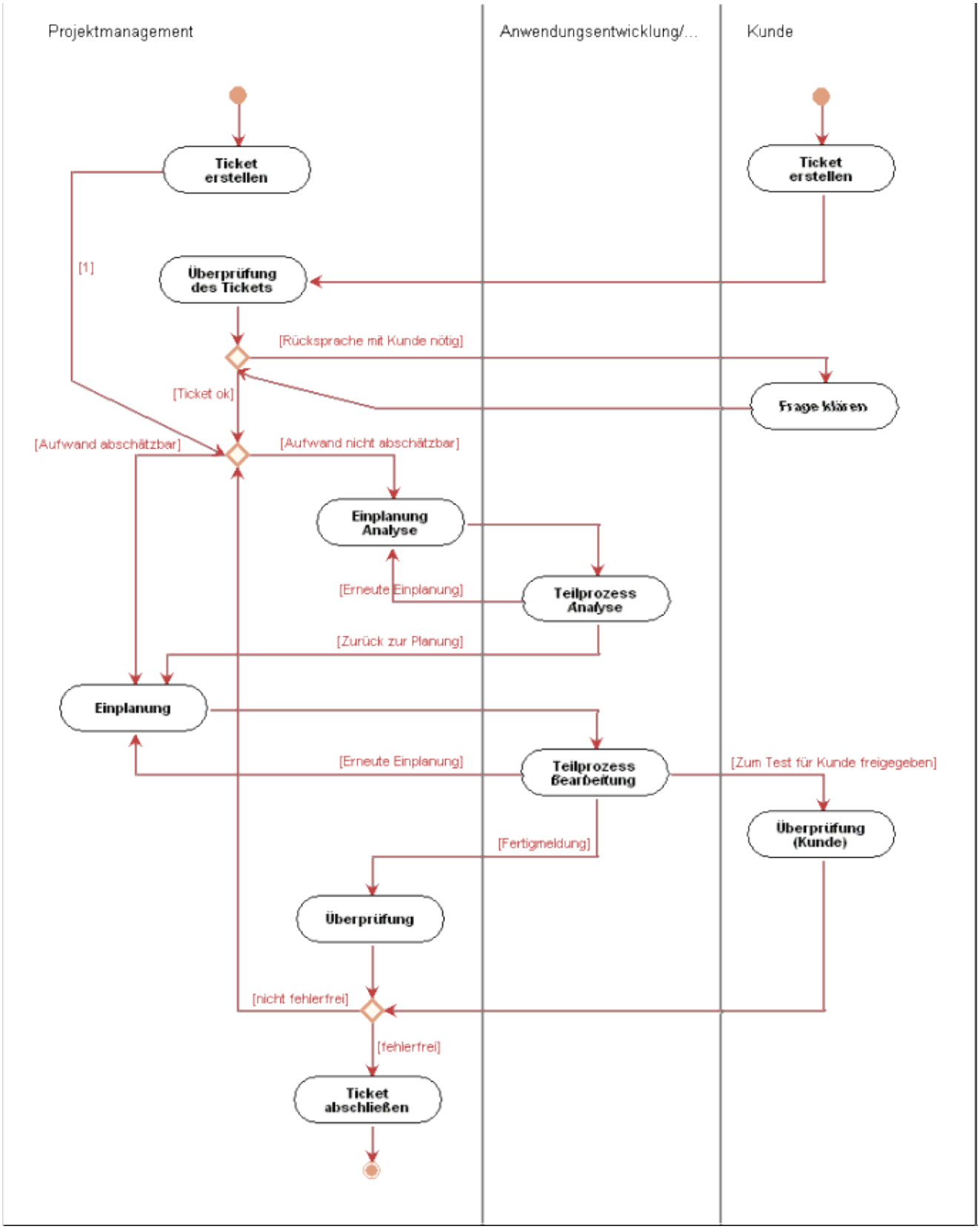


Abbildung A.2: Serviceanforderung

## A.3 Angebotsanforderung

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Ticket erstellen | • Kategorie festlegen<br>• Beschreibung, Zusammenfassung eintragen | Projektmanagement, Kunde |
| Überprüfung des Tickets | • Evtl. Notiz anlegen und Status auf „Erfordert Rückmeldung“ setzen | Projektmanagement |
| Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Einplanung Analyse | • Plan-/Restzeit und Fälligkeit eintragen<br>• Private Notiz anlegen<br>• Bearbeiter zuweisen<br>• Status auf „Analyse“ setzen | Projektmanagement |
| Teilprozess Analyse | | Anwendungsentwicklung/ Mediengestaltung |
| Einplanung | • Plan-/Restzeit und Fälligkeit eintragen<br>• Bearbeiter zuweisen<br>• Status auf „Umsetzung“ setzen | Projektmanagement |
| Angebot erstellen | • Angebot als Notiz hinzufügen oder in Papierform erstellen | Projektmanagement |
| Restzeit anpassen | | Projektmanagement |
| Überprüfung (Kunde) | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Ticket abschließen | • Status auf „Geschlossen“ setzen | Projektmanagement |

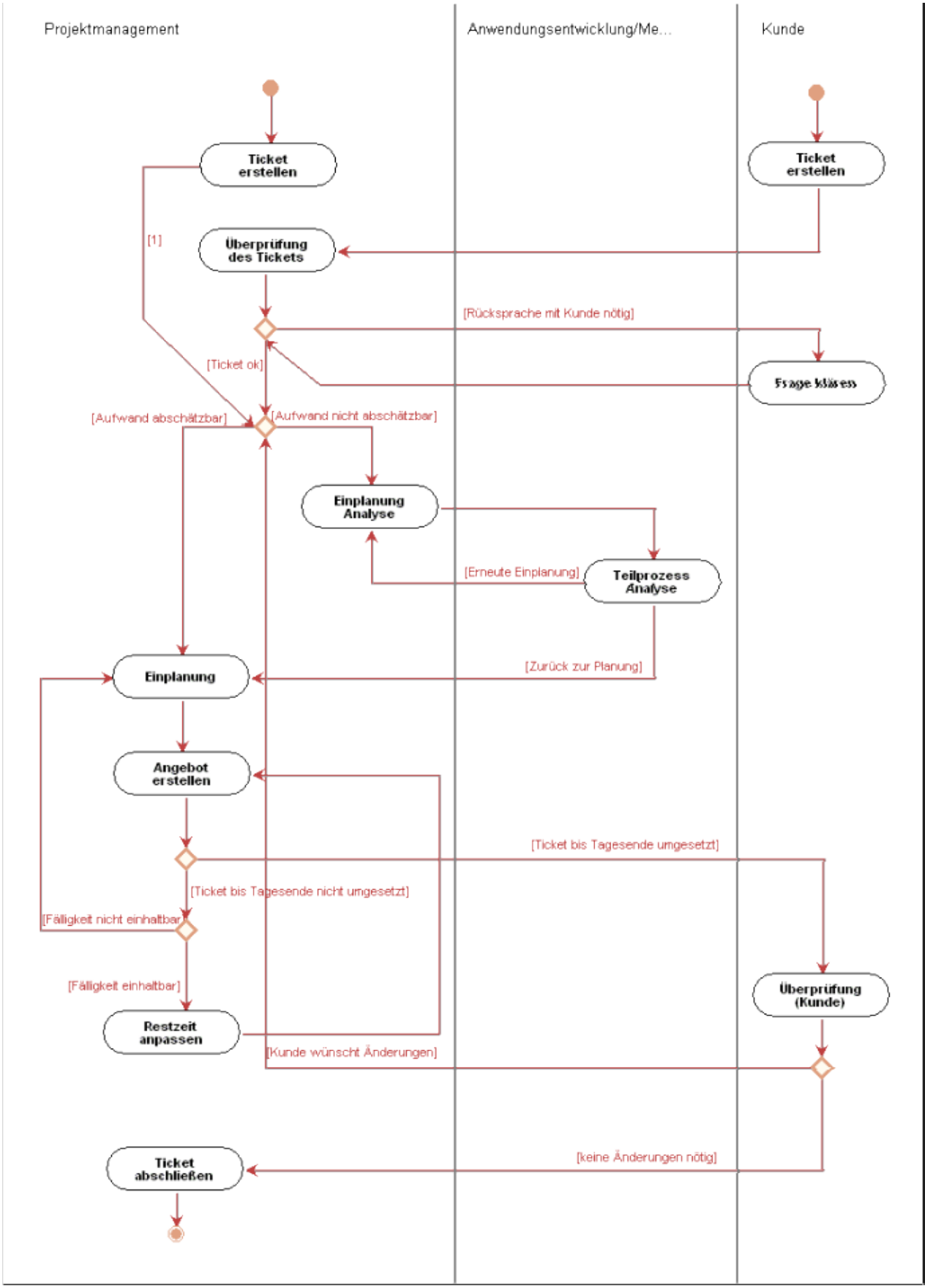


Abbildung A.3: Angebotsanforderung

## A.4 Fehler

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Ticket erstellen | • Kategorie festlegen<br>• Beschreibung, Zusammenfassung eintragen | Projektmanagement, Kunde |
| Überprüfung des Tickets | • Evtl. Notiz anlegen und Status auf „Erfordert Rückmeldung“ setzen | Projektmanagement |
| Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Kategorie anpassen | | Projektmanagement |
| Evtl. Fehlergrad anpassen | | Projektmanagement |
| Einplanung Analyse | • Plan-/Restzeit und Fälligkeit eintragen<br>• Private Notiz anlegen<br>• Bearbeiter zuweisen<br>• Status auf „Analyse“ setzen | Projektmanagement |
| Teilprozess Analyse | | Anwendungsentwicklung/ Mediengestaltung |
| Einplanung | • Plan-/Restzeit und Fälligkeit eintragen<br>• Bearbeiter zuweisen<br>• Status auf „Umsetzung“ setzen | Projektmanagement |
| Teilprozess Bearbeitung | | Anwendungsentwicklung/ Mediengestaltung |
| Überprüfung | • Notiz anlegen | Projektmanagement |
| Überprüfung (Kunde) | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Ticket abschließen | • Status auf „Geschlossen“ setzen | Projektmanagement |

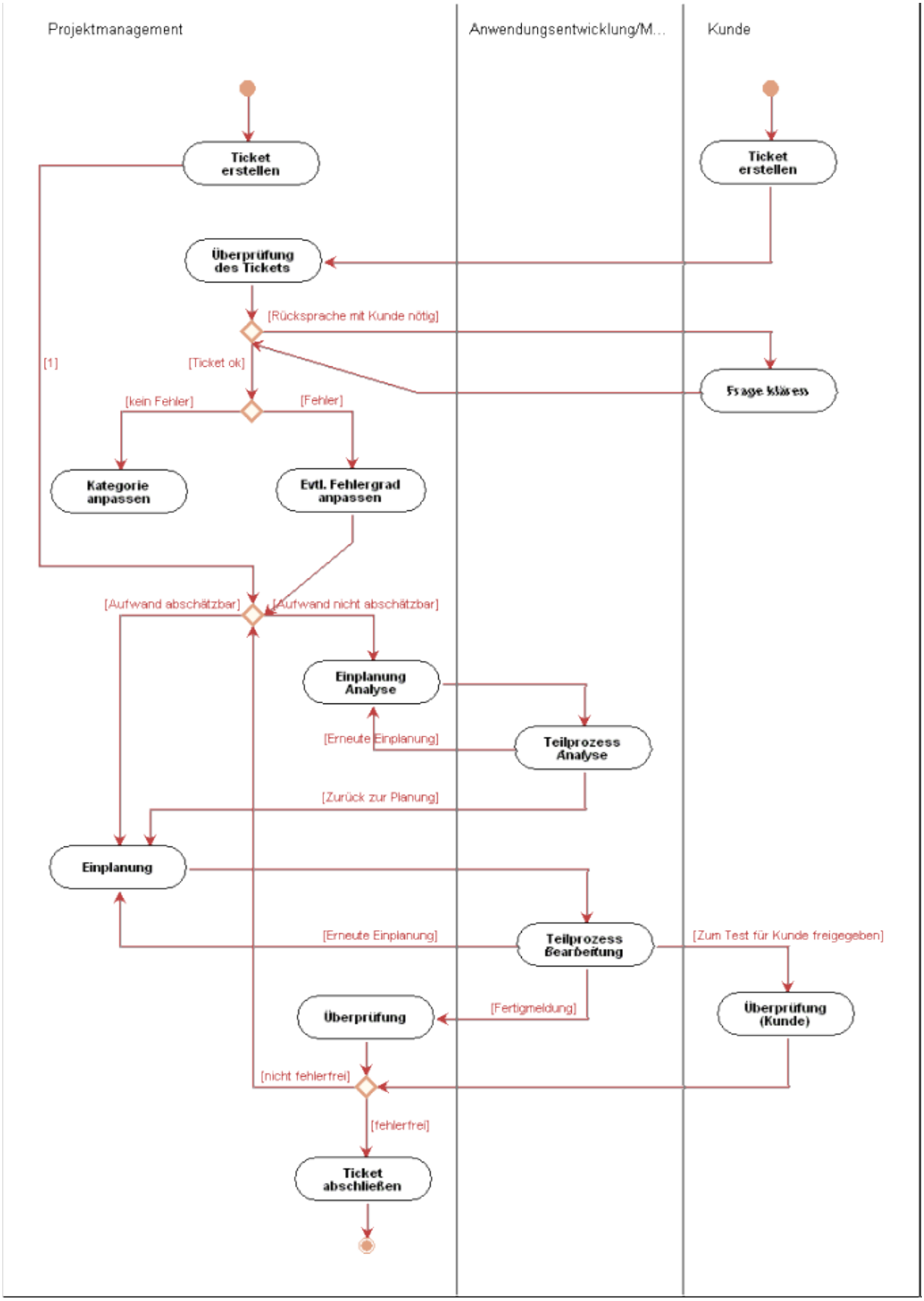


Abbildung A.4: Fehler

## A.5 Teilprozess Analyse

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Analyse | | Anwendungsentwicklung/ Mediengestaltung |
| Rücksprache mit Kunde | • Notiz anlegen<br>• Status auf „Erfordert Rückmeldung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Mit Kunde in Verbindung setzen und Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Projektmanagement, Anwendungsentwicklung/ Mediengestaltung |
| Zurück zur Planung | • Restzeit auf 0 setzen<br>• Private Notiz anlegen<br>• Status auf „Planung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Erneute Einplanung | • Private Notiz anlegen<br>• Rest-Aufwand angeben<br>• Fälligkeitsdatum löschen<br>• Status auf „Planung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Restzeit anpassen | | Anwendungsentwicklung/ Mediengestaltung |

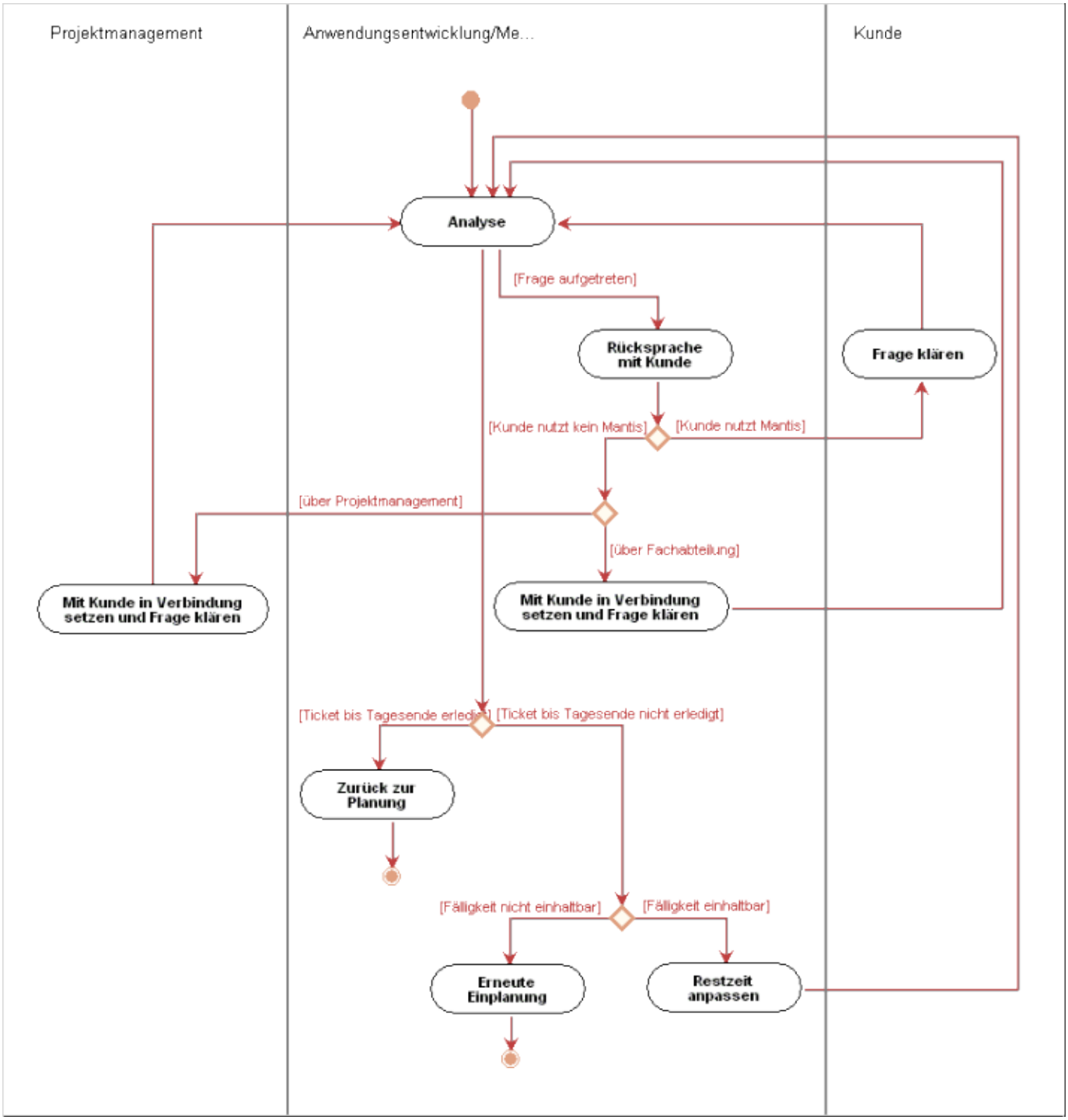


Abbildung A.5: Teilprozess Analyse

## A.6 Teilprozess Bearbeitung

| **Aufgabe** | **Arbeitsschritte** | **Akteur** |
|---|---|---|
| Bearbeitung | | Anwendungsentwicklung/ Mediengestaltung |
| Rücksprache mit Kunde | • Notiz anlegen<br>• Status auf „Erfordert Rückmeldung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Kunde |
| Mit Kunde in Verbindung setzen und Frage klären | • Notiz anlegen<br>• Status auf „Rückmeldung“ setzen | Projektmanagement, Anwendungsentwicklung/ Mediengestaltung |
| Fertigmeldung | Vorbedingung: Jeder einzelne Punkt der Anforderung wurde vollständig abgearbeitet.<br>• Notiz anlegen<br>• Restzeit auf 0 setzen<br>• Status auf „Erledigt“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Zum Test für Kunde freigeben | Vorbedingung: Jeder einzelne Punkt der Anforderung wurde vollständig abgearbeitet.<br>• Notiz anlegen<br>• Restzeit auf 0 setzen<br>• Status auf „Erfordert Rückmeldung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Erneute Einplanung | • Private Notiz anlegen<br>• Rest-Aufwand angeben<br>• Fälligkeitsdatum löschen<br>• Status auf „Planung“ setzen | Anwendungsentwicklung/ Mediengestaltung |
| Restzeit anpassen | | Anwendungsentwicklung/ Mediengestaltung |

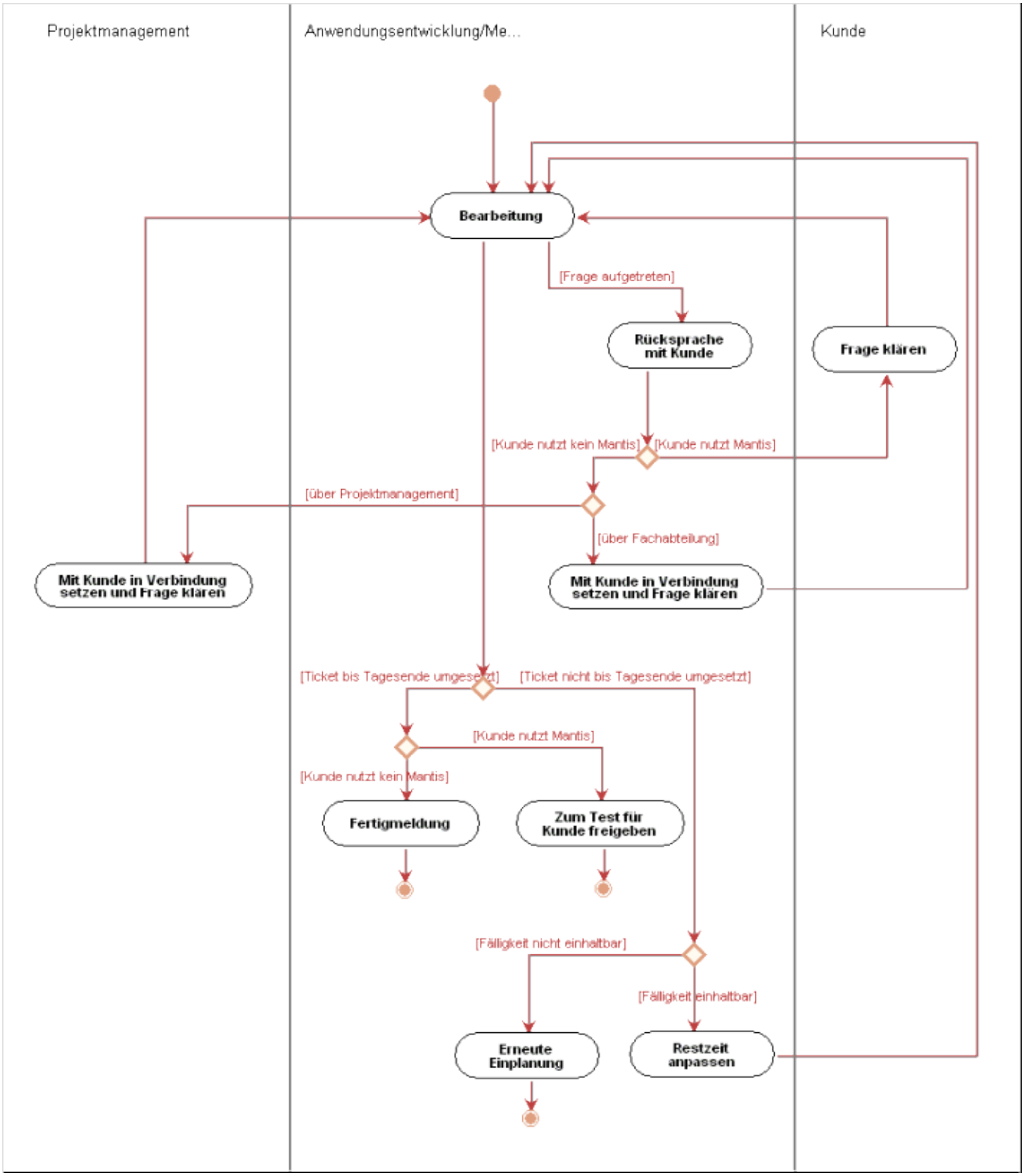


Abbildung A.6: Teilprozess Bearbeitung

# B Literaturverzeichnis

## Bücher


[1] Palmer, Stephen R.; Felsing, John M.: A Practical Guide to Feature-Driven Development. – 1. Aufl. – Upper Saddle River: Prentice Hall International, 2002

[2] Balzert, Helmut: Lehrbuch der Softwaretechnik: Softwaremanagement. – 2. Aufl. – Heidelberg: Spektrum Akademischer Verlag, 2008

[3] Bleek, Wolf-Gideon; Wolf, Henning: Agile Softwareentwicklung: Werte, Konzepte und Methoden. – 1. Aufl. – Heidelberg: Dpunkt Verlag, 2008

[4] Bunse, Christian; von Knethen, Antje: Vorgehensmodelle kompakt. – 2. Aufl. – Heidelberg: Spektrum Akademischer Verlag, 2008

[5] Brugger, Ralph: IT-Projekte strukturiert realisieren. – 2. Aufl. – Wiesbaden: Vieweg+Teubner, 2005

[6] Hruschka, Peter; Rupp, Chri; Starke, Gernot: Agility Kompakt – 2. Aufl. – Heidelberg: Spektrum Akademischer Verlag, 2009

[7] Balzert, Heide: Lehrbuch der Objektmodellierung: Analyse und Entwurf mit der UML 2. – 2. Aufl. – Heidelberg: Spektrum Akademischer Verlag, 2005


## Internetquellen


[8] Kriegisch, Alexander: Scrum-Einführung.
URL: http://scrum-master.de/content/blogsection/4/31/, verfügbar am 18.08.2009

[9] Ward Cunningham: Manifesto for Agile Software Development.
URL: http://agilemanifesto.org/, verfügbar am 18.08.2009

[10] IDS Scheer AG: Ereignisgesteuerte Prozesskette (EPK).
URL: http://www.ids-scheer.de/de/ARIS/ARIS_Modellierungsstandards_EPK/79890.html?mod_srch[result_link]=1, verfügbar am 27.08.2009

[11] Object Management Group: BPMN Information Home.
URL: http://www.bpmn.org/, verfügbar am 10.07.2009

[12] Martinig & Associates: Software Development Poll: Programming, Testing, Open Source, UML, Agile, Tools.
URL: http://www.methodsandtools.com/dynpoll/oldpoll.php?Agile2, verfügbar am 18.08.2009

[13] The Standish Group International: Standish Newsroom - CHAOS 2009.
URL: http://www.standishgroup.com/newsroom/chaos_2009.php, verfügbar am 18.08.2009

[14] SeleniumHQ: Introducing Selenium - Selenium Documentation.
URL: http://seleniumhq.org/docs/01_introducing_selenium.html#id1, verfügbar am 20.08.2009

[15] Scheer, A.-W.; Keller, G.; Nüttgens, M.: Semantische Prozessmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“.
URL: http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf, verfügbar am 18.08.2009

[16] oose Innovative Informatik GmbH: UML 2.0 Notation.
URL: http://www.oose.de/fileadmin/Dateien/uml-2-Notationsuebersicht-oose.de.pdf, verfügbar am 25.08.2009

[17] KAISHA-Tec Co. Ltd: BPMN - Business Process Modeling Notation 1.2.
URL: http://www.activemodeler.com/download/Documents/Poster_BPMN_de.pdf, verfügbar am 25.08.2009

[18] Nebulon Pty. Ltd: FDD in the small.
URL: http://www.featuredrivendevelopment.com/node/648, verfügbar am 18.08.2009

[19] Nebulon Pty. Ltd: FDD processes.
URL: http://www.featuredrivendevelopment.com/files/fddprocessesA4.pdf, verfügbar am 18.08.2009

## Bildquellen


[20] http://www.bpmn.org/Samples/Elements/BPMN_E1.gif, verfügbar am 02.12.2009

[21] http://www.pmaktuell.org/uploads/PMAktuell-200701/036-Wissen.jpg, verfügbar am 02.12.2009

[22] http://www.rzwire.com/images2/extreme.gif, verfügbar am 02.12.2009

[23] http://www.1dot0.com/images/agile-scrum-process.jpg, verfügbar am 02.12.2009

[24] http://www.agilemodeling.com/images/lifecycleFDD.gif, verfügbar am 02.12.2009

[25] http://www.featuredrivendevelopment.com/files/images/ParkingLot.045.preview.png, verfügbar am 02.12.2009

[26] http://www.featuredrivendevelopment.com/files/images/ParkingLot.044.preview.png, verfügbar am 02.12.2009

# C Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, den 03.12.2009

_____

Sylvia Bilek