

# Requirements-Engineering und Aufwandsabschätzung

## Ein integrierter Ansatz für Webapplikationen

Martin Zimmermann  
Hochschule Offenburg

Mirjam Drescher  
Swiss Life Zürich

m.zimmermann@fh-offenburg.de

mirjam.drescher@swisslife.ch

**Abstract:** Wir stellen eine Use Case zentrierte Methodik vor, die darauf basiert aus den Use Case Diagrammen und Use Case Beschreibungen sowie weiteren Artefakten der Verhaltensmodellierung, verlässliche Aufwandsabschätzungen mittels Function Points abzuleiten. Dazu werden die Artefakte analysiert und mittels Komplexitätsregeln durch Function Point bewertet. Wir stellen im Detail Komplexitätsregeln vor, um zielgerichtet Aufwandsabschätzungen vornehmen zu können.

## 1 Vorgehensweise

Bei der Planung eines neuen Softwaresystems tauchen oft die Fragen auf: Wie lange dauert die Entwicklung und mit welchem Entwicklungsaufwand ist zu rechnen? Welchen Aufwand haben die einzelnen zu realisierenden Funktionen? Welche Funktionen sind besonders kritisch bei der Aufwandsbewertung?

Wir stellen eine Use Case zentrierte Methodik vor, die darauf basiert aus den Use Case Beschreibungen und weiteren Artefakten der Verhaltensmodellierung eine Aufwandsabschätzung mittels Function Points abzuleiten.

Use Cases betonen Anwenderziele und -perspektiven und gelten daher als Schlüsselmethode zur Identifikation und Beschreibung der funktionalen Anforderungen eines Systems [Coc01]. Sie beeinflussen viele Aspekte eines Projektes, insbesondere die zahlreichen Details der Verhaltensmodellierung, wie Sequenz- und Zustandsdiagramme in der Designphase.

Wir zeigen, wie ausgehend von einem Domain-Modell systematisch verschiedene aufeinander aufbauende Artefakte zur Verhaltensmodellierung konsistent entwickelt werden können (Abbildung 1). Ausgangspunkt sind die Use Case Diagramme. Zu jedem Use Case wird eine Beschreibung erstellt, die dann als Grundlage zur Spezifikation sog. System-Sequenzdiagramme und Systemoperationen dienen.

Ausgehend von den Artefakten des Requirements-Engineering ermöglicht unsere Methodik den Aufwand zur Implementation von Softwaremodulen systematisch zu bestimmen (Abbildung 1).

In Kapitel 2 führen wir zunächst als optionale Erweiterung von Use Case Diagrammen eine auf Pfadausdrücken basierte Sprache zur Spezifikation des zeitlichen Ablaufs bei der Initiierung von Use Cases durch die Akteure ein. Von zentraler Bedeutung sind hierbei die sog. System-Sequenzdiagramme und Systemoperationen, einschließlich Vor- und Nachbedingungen, da sie eine Blackbox-Sicht auf ein System ermöglichen.

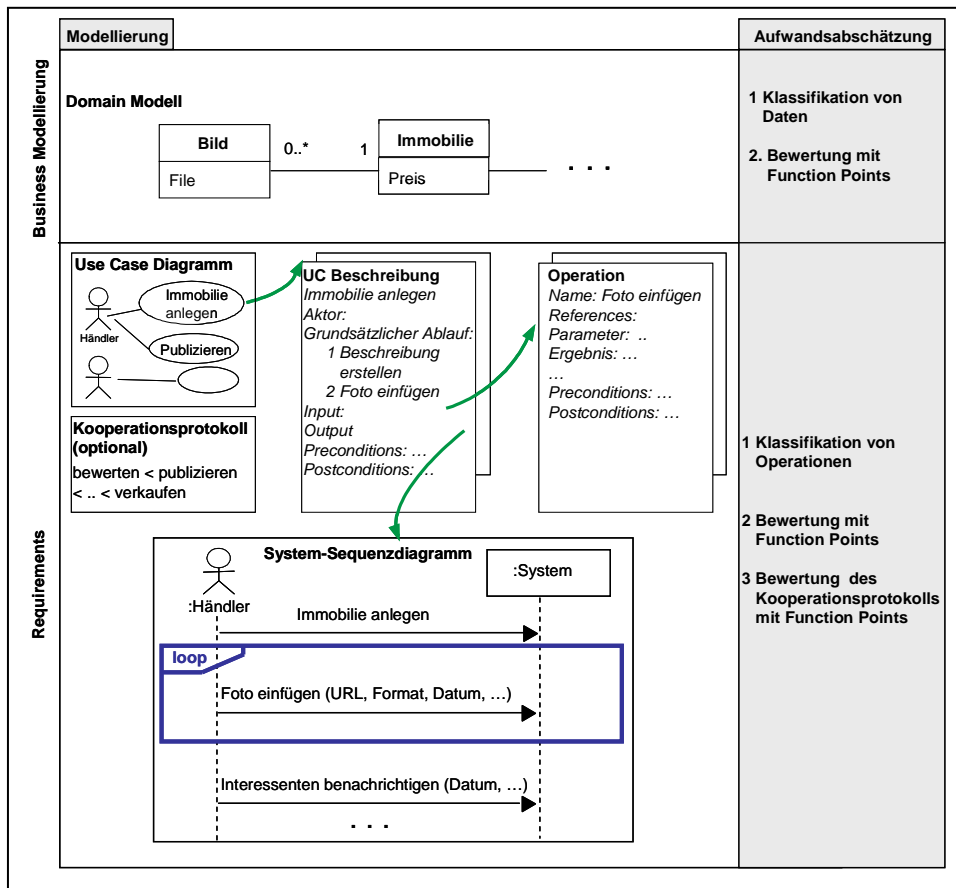


Abb. 1. Requirements-Engineering und Aufwandsabschätzung

Darauf aufbauend zeigen wir in Kapitel 3, wie sich aus dem Domain-Modell und den Artefakten der Verhaltensmodellierung schrittweise verlässliche Aufwandsabschätzungen basierend auf der Function Point (FP) Analyse herleiten lassen. Dazu werden die Artefakte, z.B. Klassen und Beziehungen des Domain-Modells klassifiziert und mittels Komplexitätsregeln durch Function Point bewertet (Abbildung 1). Wir stellen im Detail Regeln vor, um zielgerichtet Function Points zu bestimmen.

## 2 Requirements-Engineering

Ausgangspunkt für die Modelle zur Verhaltensbeschreibung, wie sie im Rahmen des Requirements-Engineering entwickelt werden, ist ein Domain-Modell [Lar05].

Ein Domain-Modell ist das wichtigste und klassische Modell der OO-Analyse. Es betont die problemrelevanten Konzepte in einer Domäne.

Domain-Modelle und die Artefakte zur Verhaltensmodellierung beeinflussen sich gegenseitig:

- Aus den Use Case Beschreibungen lassen sich oft wesentliche Domänenklassen, Begriffe, Attribute und Assoziationen ableiten.
- Umgekehrt beschreiben Nachbedingungen von Operationen, wie sich der Zustand von Domänenobjekten, Attributen und Assoziationen nach Ausführung eines Use Cases ändert.

Abbildung 2 zeigt den Ausschnitt eines solchen Domain-Modells für eine Immobilien-Applikation, die wir als durchgängiges Beispiel benutzen werden.

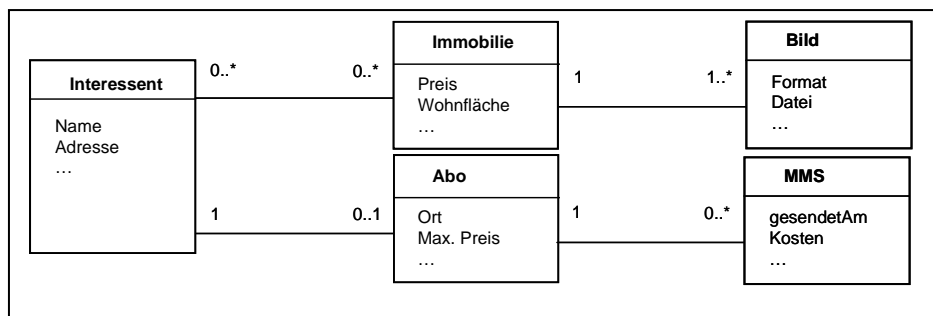


Abb. 2. Ausschnitt eines Domain-Modells

Zu einer Immobilie gehört mindestens ein Bild. Ein Interessent kann darüber hinaus im Rahmen eines Abonnements neue Angebote abonnieren. D.h. bei Vorliegen einer neuen Immobilie, die den Kriterien, wie im Abonnement angegeben (z.B. bezüglich dem maximalen Preis, Ort, etc.) genügt, wird der Interessent benachrichtigt, z.B. über email oder MMS.

### 2.1 Use Cases

Ausgangspunkt einer Aufwandsabschätzung ist ein Use Case Diagramm und die erweiterte Beschreibung der einzelnen Use Cases.

Das Use Case Diagramm in Abbildung 3 zeigt einen Teil der Use Cases für das Immobilienmarketing.

Wir schlagen für die UC Diagramme folgende optionale Erweiterung vor: Mittels eines Kooperationsprotokolls kann das Verhalten der Akteure beim Anstoßen der Use Cases präzisiert werden. Ein solches Protokoll ist ein optionaler Bestandteil eines Use Case Diagramms und definiert in welcher Weise die Use Cases eines Systems benutzt werden müssen. Es werden dazu Einschränkungen bei der Initiierung von Use Cases spezifiziert, d.h. *wer, wann, welche* Interaktionen auslösen darf.

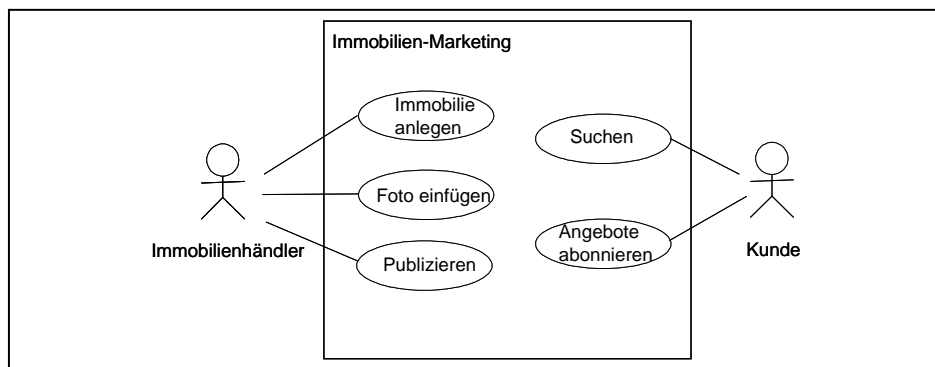


Abb. 3. Beispiel eines Use Case Diagramms (Ausschnitt)

Die Sprache zur Spezifikation eines Protokolls wurde mit dem Ziel entworfen, auf einen Blick erfassbare Formulierungen für Reihenfolgebedingungen und Synchronisationsbedingungen zu ermöglichen.

Pfadausdrücke erlauben die Spezifikation von Reihenfolgebedingungen für die Use Cases. Seinen  $p$ ,  $p1$  und  $p2$  Pfadausdrücke, so lassen sich die folgenden Operatoren anwenden:

- $p1 < p2$   
Zunächst sind die Use Cases des Ausdrucks  $p1$  und dann die des Ausdrucks  $p2$  durchführen
- $p 1..*$   
Die Use Cases des Ausdrucks  $p$  werden ein bis  $n$  mal ausgeführt
- $p1 \parallel p2$   
Die Use Cases von  $p1$  und  $p2$  können parallel ausgeführt werden
- $p1; p2$   
Es werden die Use Cases von  $p1$  oder von  $p2$  ausgeführt
- $T[p]$   
Die Durchführung der Use Cases von  $p$  erfolgt im Rahmen einer Transaktion

Zur besseren Lesbarkeit von komplexen Pfadausdrücken können diese modular aus hierarchisch angeordneten Teilpfadausdrücken aufgebaut werden.

Im Falle unserer Beispiel-Applikation sind folgende **Abhängigkeiten** zwischen den Operationen bekannt:

Am Anfang muss die Immobilie angelegt werden (UC „Immobilie anlegen“). Dies beinhaltet eine Bewertung der Immobilie (UC Bewerten) und das Einfügen von mindestens einem Foto bzw. Video.

Anschließend wird die Immobilie publiziert (UC „Im Internet Publizieren“). Nach dem Publizieren kann die Immobilie verkauft werden. Abschließend muss ein Verkaufsreport (UC „Verkaufsreport Erstellen“) erstellt werden. Solange die Immobilie noch nicht verkauft ist, können jederzeit neue Fotos eingestellt werden.

Das folgende Beispiel in Abbildung 4 illustriert zwei Pfadausdrücke und damit das Protokoll für das Auslösen von Use Cases im Falle der Immobilien-Applikation. Hierbei werden auch hierarchisch aufgebaute Pfadausdrücke verwendet der Use Case „Immobilie anlegen“ ist selbst ein Pfadausdruck).

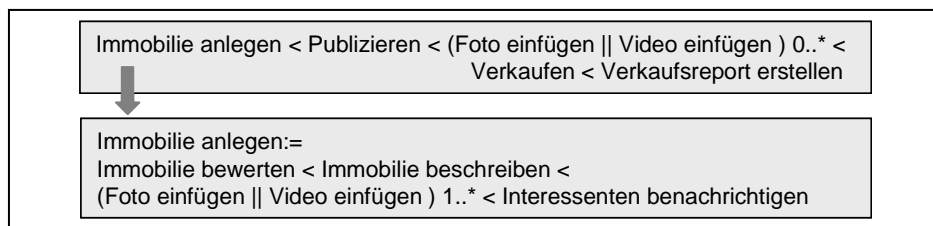


Abb. 4. Beispiel eines Kooperationsprotokolls

UML liefert mit OCL (Object Constraint Language) eine Sprache zur Beschreibung von Bedingungen für UML Diagramme, speziell Klassendiagramme, insbesondere für die Spezifikation von Invarianten (d.h. gültige Objektzustände), Vor- und Nachbedingungen von Operationen und Methoden. OCL eignet sich allerdings weniger für die Beschreibung eines Kooperationsprotokolls, d.h. das Verhalten der Akteure bei der Initiierung von Use Cases.

## 2.2 Spezifikation von Operationen mit Vor- und Nachbedingungen

Aus den Use Case Beschreibungen lassen sich die sog. Systemoperationen und deren Aufruf im Rahmen von „System“-Sequenzdiagrammen ableiten. UML definiert keinen speziellen Diagrammtyp für ein „System“-Sequenzdiagramm. Wir verwenden in Anlehnung an [Lar05] die Qualifikation „System“, um die Anwendung dieses Diagrammtyps auf Systeme zu betonen, die als Blackboxes behandelt werden. Wir konzentrieren uns im Folgenden auf die Formulierung von Systemoperationen mit Nachbedingungen (Contracts) als zentrales Element der Verhaltensbeschreibung.

Contracts gehen ursprünglich auf Tony Hoar zurück. In den frühen 90er Jahren beschrieb Grady Booch in seiner Booch Methode die Anwendung von Contracts auf Objektoperationen. Außerdem griffen Derek Coleman und seine Kollegen bei HP Labs die Idee der Operation Contracts auf [Col94]. Die UML definiert OCL [WK99] in der Bedingungen von UML Operationen ausgedrückt werden können. In der Praxis werden diese aber bisher kaum verwendet.

Nach unserer Erfahrung sind Nachbedingungen nicht in jedem Fall erforderlich. In vielen Fällen können die Konsequenzen verhältnismäßig leicht aus einem Use Case (bzw. einer Operation) hergeleitet werden. Allerdings gibt es Fälle, in denen eine präzisere Darstellung der Nachbedingungen in Form von sog. Contracts Vorteile bringen. Nachbedingungen beschreiben die erforderlichen Ergebnisse präzise als Zustandsänderungen bezogen auf die Objekte und Beziehungen des Domainmodells. Hierbei steht die Spezifikation der folgenden Aspekte im Vordergrund:

a) Erzeugte bzw. gelöschte Objekte (instance creation/deletion)

Beispiel: „Bildobjekt wurde erzeugt“

b) Geänderte Attributwerte (attribute modification)

Beispiel: „insgesamt benutzter Speicher wurde erhöht (um die Größe des neuen Bildes)“

c) Neue/gelöschte Assoziationen bzw. Kompositionen zwischen Objekte (associations formed/broken)

Beispiel: „ein Objekte der Klasse Bild wurde mit der Klasse Immobilie verbunden“

Nachbedingungen sollten in der Vergangenheitsform ausgedrückt werden, da sie Beobachtungen von Zustandsänderungen betonen, die als Folge einer Operation eingetreten sind, und nicht Aktionen, die (noch) stattfinden sollten.

Abbildung 5 zeigt ein Beispiel eines Contracts.

<b>CONTRACT CO_NR: Interessenten benachrichtigen</b>	
<b>Querverweise:</b>	Use Cases: Immobilie anlegen
<b>Parameter:</b>	Datum, Uhrzeit
<b>Vorbedingung:</b>	keine
<b>Nachbedingung:</b>	für alle <b>a[k]: Abo</b> gilt: falls Immobilie <b>i</b> den Anforderungen des Abos <b>a[k]</b> genügt (Preis, Lage): dann wurde MMS Objekt <b>m</b> erzeugt <b>m</b> wurde mit <b>a[k]</b> verbunden (Assoziation)

Abb. 5. Beispiel eines Contracts

Folgende Kernfragen sollten daher bei der Spezifikation eines Contracts formuliert werden (diese helfen, die wichtigsten Bedingungen systematisch zu identifizieren):

- Welche neuen Objekte sollten erzeugt worden sein, nachdem die Operation erfolgreich abgelaufen ist?
- Welche Attribute von vorhandenen oder neu erzeugten Objekten sollten geändert worden sein?
- Welche Assoziationen zwischen vorhanden oder neuen Objekten sollten aufgebaut bzw. aufgehoben worden sein, nachdem die Operation abgelaufen ist?

### 2.3 Artefakte des Designs

Wir konzentrieren uns auf die Aspekte der Verhaltensmodellierung. Abbildung 6 illustriert, wie die Ergebnisse des Requirements-Engineering die Artefakte des Designs zur Verhaltenmodellierung beeinflussen am Beispiel eines Sequenzdiagramms.

Die Nachbedingungen aus den Contracts der Operationen müssen durch entsprechende Operationsaufrufe bzw. durch das Erzeugen / Löschen von Objekten (bezogen auf das Domain-Modell) gewährleistet sein. Die folgende Abbildung 6 zeigt dies an einem Beispiel.

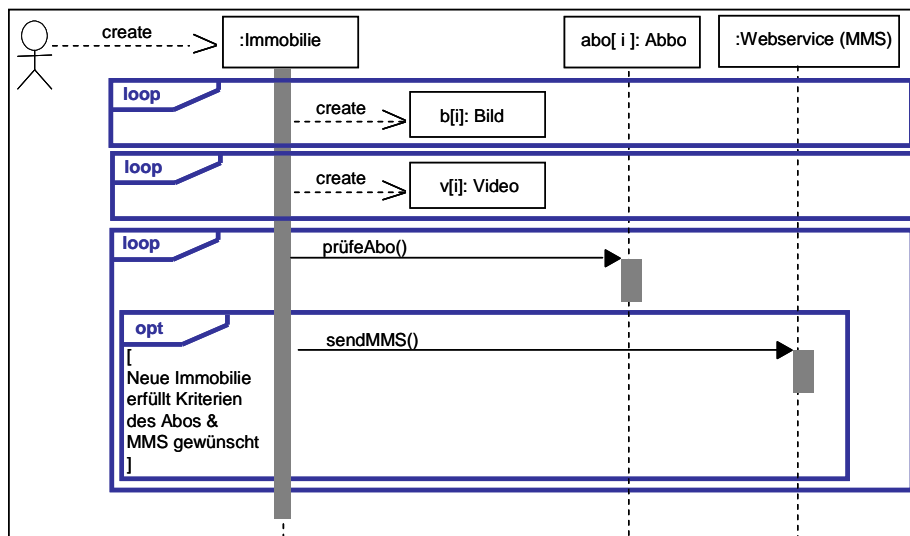


Abb. 6. Umsetzung von Contracts durch Sequenzdiagramme

Im Sequenzdiagramm werden die Designaspekte illustriert: als Folge des Use Cases “Immoblie anlegen” müssen zunächst entsprechende Bild- und / oder Videoobjekte erzeugt werden.

Zum Verschicken der MMS Nachrichten wird ein entsprechender Webservice, wie er von einigen Telekom-Providern angeboten wird, aufgerufen.

Die Nachbedingungen für den Use Case, wie sie in Abbildung 5 dargestellt sind, werden wie folgt im Sequenzdiagramm umgesetzt:

- die ersten beiden Schleifen gewährleisten, dass mindestens ein Objekt der Klasse Bild existiert bzw. zusätzlich optional mehrere Videoobjekte erzeugt wurden.
- Die dritte Schleife mit dem zusätzlichen Konstrukt “opt” stellt sicher, dass an alle betroffenen Interessenten (die im Rahmen eines Abonnements ihre Kriterien an Immobilienangebote definiert haben) eine MMS verschickt wird.

Insgesamt müssen die Nachbedingungen, wie Sie durch die Contracts zu den Operationen formuliert wurden, erfüllt werden.

### **3 Aufwandsabschätzung**

Der Begriff Function Point Analysis [FPA] wurde von Alan Albrecht, einem Mitarbeiter der IBM, Ende der 70er Jahre erstmals gebraucht [Hür05]. Dabei betrachtete A. Albrecht ein Softwaresystem nicht mehr aus Sicht des Programmierers (der vor allem die Programmieranweisungen und somit die Lines of Codes sieht), sondern aus Sicht des Anwenders (der die Software nutzt und so auch ganz unmittelbar ihre Leistung erfährt). Damit wird seine Messtechnik von der technischen Realisierung der gestellten Aufgabe unabhängig. 1985 publizierte IBM dann in Deutschland die als „IBM-Kurve“ bekannte Grafik. Diese stellte erstmals den Zusammenhang von Aufwand und Function Points [FPs] anhand von 54 Projekten dar.

Die FPA zählt zu den funktionalen Metriken, da sie bei der Messung die Funktionen eines Produkts zugrunde legt. Funktionale Metriken eignen sich zu einem frühen Zeitpunkt im Entwicklungsprozess besser als Grundlage zur Aufwandschätzung als konstruktive Metriken. Da man zu einem frühen Zeitpunkt zumindest einen Teil der fachlichen Anforderungen kennt, kann man mit der FPA den fachlichen Umfang abschätzen. Mit den gleichen Informationen fällt es jedoch schwer, die Anzahl der Lines of Codes [LoCs] zu bestimmen.

Es gibt verschiedene Richtungen, in die die Function Point Analyse in den letzten Jahren weiterentwickelt wurde. Wir verwenden hier ausschließlich die Variante der IFPUG (International Function Point User Group) [IF00]. Bei der Function Point Analyse steht nicht die Sicht des Entwicklers, sondern die Sicht des Anwenders im Vordergrund. Aus diesem Grunde kann die Zählung bereits nach der Erfassung der Use Cases angewendet werden.

### 3.1 Klassifikation von Datenbeständen

Die FPA zählt die durch das System verwalteten Datenbestände zum Funktionsumfang einer Anwendung dazu. Entscheidend ist hier die Anwendersicht und nicht die technische Implementierung. Es zählen die in der Anwendung verwalteten Informationen, die einerseits fachlich und andererseits für den Anwender sichtbar sind.

Die folgende Abbildung 7 zeigt einen Überblick der Tätigkeiten, die im Rahmen einer Aufwandsanalyse bezogen auf die Daten einer Anwendung zu erledigen sind.

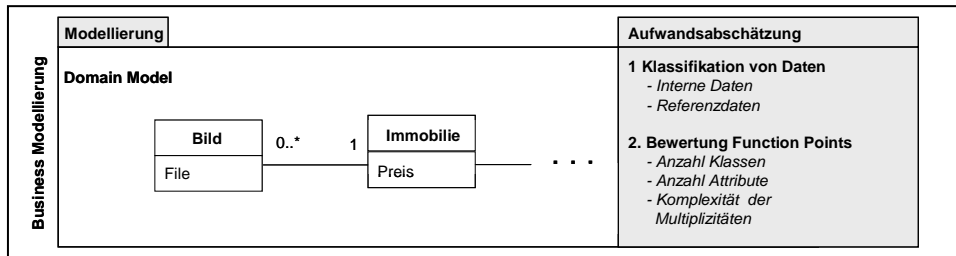


Abb. 7. Aufwandsabschätzung resultierend aus dem Domain-Modell

Einen Datenbestand, der innerhalb der Anwendung selbst gepflegt wird, bezeichnet man als internen Datenbestand oder internal logical file [ILF] während ein Datenbestand, der nur lesend verwendet wird, Referenzdatenbestand oder external interface file [EIF] genannt wird. Abbildung 8 zeigt exemplarisch die Definition eines Referenzdatenbestandes (weitere Definitionen sind in [Dre07]).

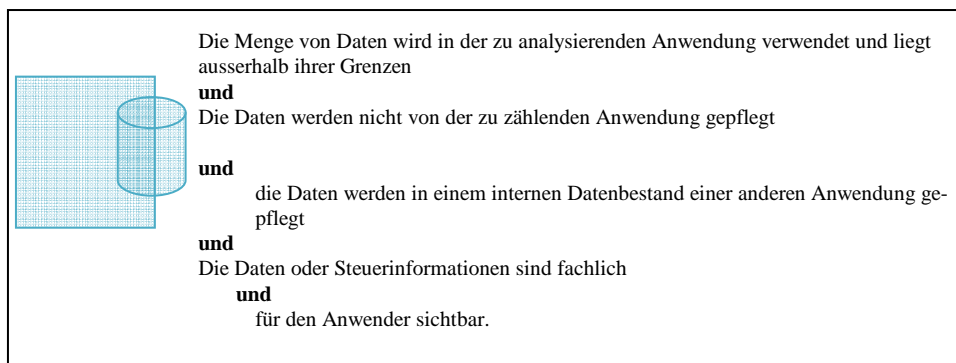


Abb. 8. Definition einer Referenzdatenbestands

### 3.2 Klassifikation von Operationen (Elementarprozesse)

Das Prinzip des Elementarprozesses (*elementary process*) wie es in der FP-Analyse verwendet wird, ist letztlich die wesentliche Voraussetzung, überhaupt den Funktionsumfang quantifizieren zu können. Ein Elementarprozess ist die kleinste für den Anwender sinnvolle Aktivität und er muss in sich abgeschlossen sein sowie die Anwendung in einem konsistenten Zustand belassen.

Die folgende Abbildung zeigt einen Überblick der Tätigkeiten, die im Rahmen einer Aufwandsanalyse bezogen auf die Operationen einer Anwendung zu erledigen sind.

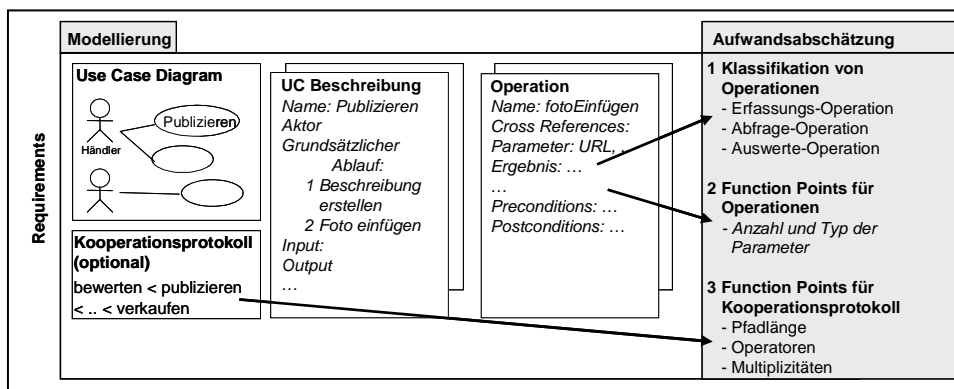


Abb. 9. Definition einer Referenzdatenbestands

Im ersten Schritt werden Operationen klassifiziert. Darauf aufbauend erfolgt die Bestimmung der Function Points unter Berücksichtigung der Signatur einer Operation. Details hierzu sind in Kapitel 3.4 aufgeführt. Ist ein Kooperationsprotokoll vorhanden, erfolgt abschließend die Bewertung dieses Kooperationsprotokolls mit Function Points.

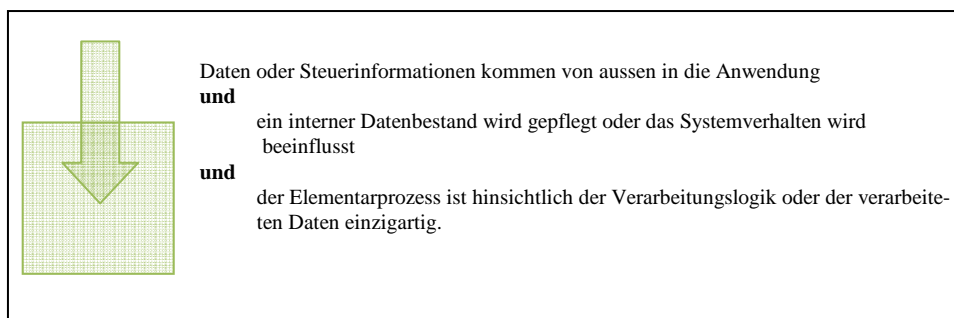


Abb. 10. Definition einer Referenzdatenbestands

Es werden drei Arten von Funktionen unterschieden: **Erfassungs-, Abfrage-, und Auswertungsfunktionen**, die jedoch auch oft als Eingabe-, Abfrage- und Ausgabefunktion

bekannt sind. Wir verwenden in Anlehnung an [Poe05] für die Kennzeichnung dieser drei Funktionen folgende Symbole und Begriffe (Abbildung 10):

**Auswertung:** Als Auswertung werden all jene Elementarprozesse definiert, bei denen Daten die Anwendung verlassen, sei dies als Anzeige, als Ausdruck auf Papier oder als Ausgabe auf einen anderen Datenträger (Magnetband, CD, usw.) oder in eine andere Datei. Abbildung 11 zeigt die Definition für die Auswertungsfunktion [PB05].

Abfragefunktionen sind im Grunde vereinfachte Auswertungsfunktionen. Im Gegensatz zur Auswertungsfunktion, beinhaltet eine Abfragefunktion keine berechneten oder abgeleiteten Daten.

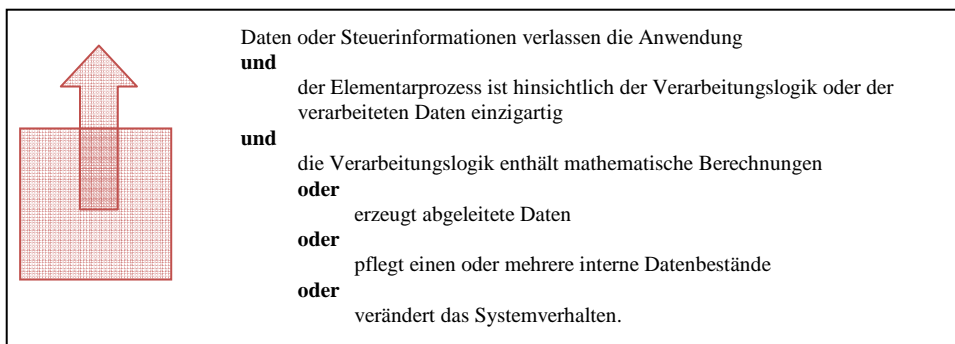


Abb. 11. Definition einer Referenzdatenbestands

In der FPA wird die Komplexität eines Elementarprozesses über die Betrachtung der Anzahl der Datenfelder und der referenzierten Datenbestände, bestimmt. Analog werden für Datenbestände die Anzahl der Datenfelder (DET) und der „Feldgruppen“ untersucht.

- niedrige Komplexität (*low complexity*)
- mittlere Komplexität (*average complexity*)
- hohe Komplexität (*high complexity*)

Erfassung		Anz. Datenelement (DETs)		
		1-4	5-15	>=16
Anz. Datenbestände (FTRs)	0-1	3	3	4
	2	3	4	6
	>=3	4	6	6

Interner Datenbestand		Anz. Datenelemente (DETs)		
		1-19	20-50	>=51
Anz. Datenbestände (RETs)	1	7	7	10
	2-5	7	10	15
	>=6	10	15	15

Abb. 12. Komplexitätsregeln zur Bewertung von Daten und Operationen

Diese Größen werden als Maß genommen, um die Elementarprozesse sowie die Datenbestände in drei Gewichtungsstufen einordnen zu können (siehe Abbildung 12):

### 3.3 Regeln zur Bewertung von Datenbeständen

Nachfolgend sind exemplarisch einige Regeln zur Ermittlung und Bewertung der Datenbestände aufgelistet (für eine vollständige Liste, siehe [Dre07]).

- Datenbestände werden nur einmal gezählt, unabhängig davon wie häufig sie in den Geschäftsfunktionen angesprochen werden. Begründung: Gezählt wird, was für den Anwender als Information sichtbar ist. Die Nutzung der sichtbaren Informationen wird bei den Funktionen gezählt. [Hür05]
- Alle Beziehungen (bzw. Fremdschlüsseln bei relationalen Datenbanken) von anderen Datenbeständen werden bei der Ermittlung der Anzahl Datenelementtypen berücksichtigt und gehen in die Zählung ein. (Begründung: Beziehungen sind für den Anwender ersichtlich).
- Bei der Bewertung von Multiplizitäten wird grundsätzlich unterschieden, ob die Beziehung zwischen zwei Klassen optional ist und welche Abhängigkeit zwischen diesen zwei Klassen genau besteht. Die genaue Bedeutung der Abhängigkeit wird in der untenstehenden Tabelle an der jeweiligen Stelle genauer erläutert. Die Abkürzung LF steht dabei für logical file und bezieht sich entweder auf einen internen Datenbestand oder einen Referenzdatenbestand.

Beziehungstyp A und B	von	Zählung von A und B	Bemerkungen
0..1 : 0..*		2 LFs	
1 : 1..*		1 LF, 2 RETs, Summe der DETs	
1 : 0..*		1 LF, 2 RETs, Summe der DETs	Falls B von A abhängig ist. D.h. das Löschen von A ist erlaubt und alle verlinkten Bs werden automatisch gelöscht.
		2 LFs	Falls B unabhängig von A ist. D.h. das Löschen von A ist nicht erlaubt, solange noch Bs dazu verlinkt sind.

Abb. 13. Bewertung von Multiplizitäten

Im ersten Schritt ist die Anzahl der Datenbestände, die in der Function Point Analyse gezählt werden, zu ermitteln. Anhand dieser Datenbestände oder auch *logical files* (LFs) genannt, können meist sehr einfach auch die RETs, d.h. die Datengruppen, bestimmt

werden. Danach ist in einem dritten Schritt festzulegen, um welche Art von Datenbestand es sich handelt. Dies hat einen Einfluss auf die FP-Bewertung. Ein interner Datenbestand wird mit mehr Function Points bewertet als ein Referenzdatenbestand. Die Begründung ist, dass dieser in der Anwendung gepflegt werden muss. Der vierte Schritt umfasst die Zählung der Attribute und somit auch die Zuweisung der Anzahl Datenelementen oder DETs. Im letzten Schritt kann nun gemäß der Anzahl DETs und Anzahl RETs in der jeweiligen für den Datenbestand bestimmten Komplexitätstabelle, wie in Abbildung 12 dargestellt, der Function Point Wert abgelesen werden.

Beispiel: Die Assoziation zwischen Bild und Immobilie ist eine 1..\* Beziehung.

Nun ist zu analysieren, ob es sich dabei um abhängige oder unabhängige Klassen handelt. D.h. die Frage ist, ob im Beispiel das Bild ( entspricht Klasse B) abhängig ist von der Immobilie (entspricht Klasse A) oder nicht. Falls das Bild abhängig wäre von der Immobilie, würde dieses beim Löschen der Immobilie automatisch auch gelöscht. Dies ist tatsächlich der Fall und deshalb handelt es sich um den ersten Fall, bei dem B abhängig ist von A. Somit werden die zwei Klassen als ein Datenbestand gezählt mit zwei "Feldgruppen".

### 3.4 Regeln zur Bewertung von Operationen

Nachfolgend sind einige exemplarische Regeln zur Ermittlung und Bewertung der Funktionen aufgelistet (für eine vollständige Liste, siehe [Dre07]).

- Bei Abfragen werden sowohl die DETs für die Erfassung als auch die DETs der Auswertung gezählt, mit der Einschränkung, dass dabei übereinstimmende Felder nur einmal gezählt werden dürfen. Beispiel: Suchfelder, die in der Ausgabe nochmals vorkommen, werden nur einmal gezählt. Begründung: Abfragen sind sozusagen eine einfache Kombination von Erfassung und Auswertung.
- Die Anmeldung an ein System und dessen Aufruf wird nicht als Erfassungsfunktion betrachtet. Begründung: Sie dienen nicht der betriebswirtschaftlichen Aufgabe, sondern sie steuern den Gesamtablauf der Anwendungen
- Bei der Bewertung einer tabellarischen Eingabe oder Ausgabe werden Zeilen mit gleichem Aufbau nur einmal gezählt. Begründung: Die Funktionalität betrachtet nur Datenstrukturen und nicht deren einzelne Ausprägungen.

Bei der Bewertung einer Operation muss daher zuerst die Operation einer der drei Typen, Erfassung, Auswertung oder Abfrage, zugeordnet werden. Als nächstes können die Anzahl der Parameter bestimmt werden. Außerdem muss die Anzahl von referenzierten Datenbeständen (FTR, File Type Referenced) ermittelt werden.

Für eine übersichtlichere Darstellung wird folgende Schreibweise gewählt:

Operation *Name*(in *Parameter*): (out *Parameter*)

*Funktions*typ → [*Anzahl* DETs] [*Anzahl* FTRs] → [*Anzahl* FPs]

Beispiel:

Bild erfassen (in Version, Benutzer, Datum, Kommentar, Stichwörter, Dateiname, Format): (out)

Die Tabelle in Abbildung 12 bestimmt für diesen Fall:

Erfassung → [7 DETs] [1 FTR] → [4 FPs]

### 3.5 Regeln zur Bewertung von Kooperationsprotokollen

Mittels Kooperationsprotokollen bzw. Pfadausdrücken lässt sich das zeitliche Verhalten der verschiedenen Akteure spezifizieren, z.B. Reihenfolgebedingungen für Use Cases oder Wiederholungen einzelner Pfadausdrücke.

Solche Kooperationsprotokolle müssen entsprechend implementiert werden, z.B. als Zustandsautomaten, wodurch zusätzlicher Aufwand verursacht wird. Die Komplexität und damit der Aufwand hängen von folgenden Faktoren ab:

- Pfadlänge
- Angaben zur Multiplizität bezogen auf die verschiedenen Pfadausdrücken
- Anzahl verschiedener Operatoren (z.B. <, ||), Verwendung von Transaktionen

Zur Entwicklung einer entsprechenden Tabelle mit Komplexitätsregeln, ähnlich wie sie in Abbildung 4 für Operationen und Daten dargestellt ist, müssen noch weitere Erfahrungen gesammelt werden. Die Kooperationsprotokolle unserer bisher betrachteten Projekte sind relativ einfach, so dass hier noch keine allgemeinen Aussagen für komplexere Protokolle möglich sind.

### 3.6 Gesamtbewertung eines Systems

Die Gesamtbewertung zeigt die Bewertungen der Use Cases bzw. Systemoperationen und die Bewertung der Datenbestände. Die folgende Abbildung zeigt einen solchen Ausschnitt für Operationen, geordnet nach Name, Typ, DETs und FPs.

Funktionen	Akteur	Typ	DETs	FTRs	Komplexität	FPs
<b>Gesamt- und Detailsicht der Bilder anzeigen</b>						
Übersicht der Bilder anzeigen	user	Auswertung	4	1	low	4
Details eines Bildes anzeigen	user	Auswertung	3	1	low	4
<b>Bilder verwalten</b>						
Übersicht der Bildinformationen anzeigen	publisher	Auswertung	10	1	low	4
Bild erfassen	publisher	Erfassung	16	1	average	4
Bild ändern	publisher	Erfassung	17	1	average	4
Bild löschen	publisher	Erfassung	2	1	low	3

Abb. 14. Gesamtbewertung (Ausschnitt)

### 3.7 Herleiten einer Aufwandschätzfunktion

Beim Herleiten einer Aufwandschätzfunktion geht es darum, durch die vier ermittelten Function Point Werte eine Trendfunktion zu legen. Dabei werden die drei Trendtypen linear, potentiell und exponentiell berücksichtigt. Um eine geeignete mathematische Funktion zu ermitteln, wird ein fünfstufiges Vorgehen angewendet, das auf dem Prinzip der Minimierung der Abstandsquadrate basiert. Der am besten geeignete Trendtyp wird mittels des Regressionskoeffizienten bestimmt [Dre07]

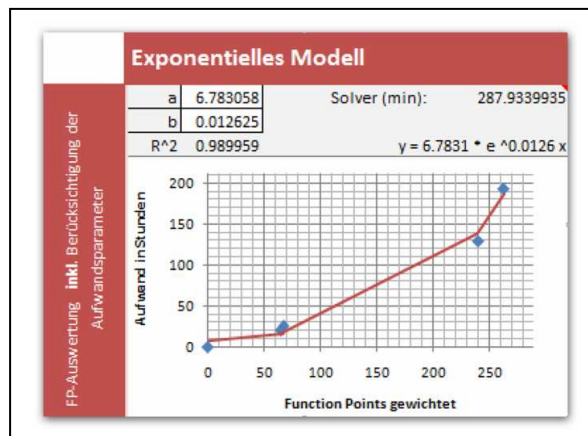


Abb. 15. Herleiten einer geeigneten Aufwandschätzfunktion

In Abbildung 15 ist die aufgrund der vorhandenen Daten von vier Projekten am besten geeignete Aufwandschätzfunktion ersichtlich. Es handelt sich dabei um eine exponentielle Funktion, welche die Aufwandstreiber oder -parameter berücksichtigt.

## 4 Zusammenfassung und Ausblick

Der beschriebene Ansatz wurde in mehreren Projekten eines IT-Dienstleistungsunternehmens, das Web-Applikationen für KMUs entwickelt, erfolgreich eingesetzt. Unsere Methodik, d.h. ausgehend vom Domain-Modell und den Artefakten des Requirements-Engineering schrittweise den Aufwand zur Implementation der Softwaremodule zu bestimmen, hat sich als sehr tragfähig erwiesen.

Es wurden hierbei sehr unterschiedliche Kategorien von Applikationen betrachtet: Präsentation (CMS), Verkauf (Online-Shop), Prozesse (z.B. Projekt-Zeiterfassung), Service (Download-Bereiche, Formulare etc.) und Analyse (z.B. Umfragen).

Web-Applikationen weisen im Vergleich zu klassischen Dialoganwendungen für eine Analyse drei Besonderheiten auf: Vermischung von „Präsentation“ und Navigation mit fachlicher Funktionalität, Multiple Möglichkeiten, die gleiche fachliche Funktion in verschiedenen Kontexten zu verwenden, „Nahtlose“ Übergänge zwischen verschiedenen Anwendungen, die für den Anwender kaum noch erkennbar sind. Wichtig ist daher die konsequente und saubere Unterscheidung zwischen der Präsentation statischer Informationen und den Funktionen als Elementarprozesse im Sinne der FPA. Da Web-Applikationen sich durch eine starke Benutzerorientierung auszeichnen, muss zudem genau darauf geachtet werden, was als Elementarprozess identifiziert werden kann und was als Navigation zu werten ist.

Ein dritter Punkt ist die Eigenheit, dass über Links oder Aufrufe schnell in eine andere Anwendung gewechselt werden kann, teilweise schon so, dass der Benutzer den Wechsel gar nicht mehr erkennen kann. Daher ist im Vorfeld klar herauszuarbeiten und zu definieren, was fachlich zur Anwendung gehört und was Funktionalitäten anderer Anwendungen sind. Ein Augenmerk muss zudem darauf gelegt werden, dass ein bestimmter Inhalt oder eine bestimmte Funktionalität an unterschiedlichen Stellen, in unterschiedlicher Granularität und eventuell in unterschiedlichen Zusammenhängen auftauchen kann. Eine solche Funktion darf nur einmal in ihrer „Maximalausprägung“ bewertet werden.

## Danksagungen

Wesentliche Ergebnisse wurden im Rahmen von Diplomarbeiten an der FHS St. Gallen in Zusammenarbeit mit einem Internet-Dienstleister für KMUs entwickelt.

## Literaturverzeichnis

- [Hür05] Hürten, R. „Function-Point Analysis Theorie und Praxis. Die Grundlage für das moderne Softwaremanagement (2. Aufl.). Renningen: expert verlag (2005).
- [IF00] IFPUG „Function Point Counting Practices Manual“ (CPM). Release 4.1.1. (2000).
- [WK99] Warmer, J und Kleppe, A. “The Object Constraint Language: Precise Modeling with UML, Reading, MA.: Addison Wesley (1999)
- [Col94] Coleman, D. et. al. “Object-oriented Development: The Fusion Method, Prentice Hall (1994)
- [Coc01] Cockburn, A. “Writing Effective Use Cases”, Addison Wesley (2001)
- [Dre07] Drescher, M. “Aufwandsabschätzung für komplexe Web-Applikationen mittels Function Points, Diplomarbeit, FHS St. Gallen (2007)
- [Lar05] Larman, C. “Applying UML and Patterns”, Prentice Hall (2005)
- [Poe05] Poensgen, B. „Function-Point-Analyse“, White Paper. <http://www.quantimetrics.de/WhitePaperFPA.pdf> (2005).