

# Anforderungs- ermittlung

## 1. Motivation und Überblick

*Die Anforderungsermittlung ist ein Verfahren, eine vollständige und widerspruchsfreie Menge von Anforderungen zu sammeln. Das Ergebnis der Anforderungsermittlung ist die so genannte Anforderungsspezifikation – ein Dokument bestehend aus dem Anwendungsfallmodell, dem Domänenklassenmodell und den Verhaltensmodellen.*

Eine Anforderung ist eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.

Der Erfolg eines Softwareprojektes hängt stark davon ab, in welchem Umfang die Erwartungen der Kunden erfüllt werden können. Anforderungsspezifikation und Anforderungsmanagement sind notwendige Voraussetzungen, um ein Projekt mit Erfolg abschließen zu können. Die Qualität der Anforderungsermittlung bestimmt in hohem Maß den Erfolg des Gesamtprojektes.

Die Anforderungsermittlung und die darauf folgende Analyse bilden das Fundament der Software-Entwicklung.

Diese Zusammenfassung basiert auf dem Skript von **Herrn Prof. Dr. H.W. Six**, Lehrgebiet Software Engineering, der [FernUniversität in Hagen](http://www.fernuni-hagen.de).

## 2. Grundsätzliches zur Anforderungsermittlung

Ziel der Anforderungsermittlung ist es, die Wünsche und die Bedürfnisse der Anwender an das zu entwickelnde System zu identifizieren und in einer Form zu dokumentieren, die die Kommunikation zwischen Auftraggeber, zukünftigen Anwendern und dem Projektteam ermöglicht.

Der IEEE Standard 830, „IEEE Recommended Practice for Software Requirements Specifications“ [IEEE93] unterscheidet funktionale Anforderungen (functional requirements) von nichtfunktionalen Anforderungen (non-functional requirements):

*Funktionale Anforderungen:*

- Funktionen oder Eigenschaften
- Externe Schnittstellen  
(Vorgaben für die Benutzungsschnittstelle, Formate für Ein- bzw. Ausgabedaten, Schnittstellendefinitionen externer Systeme, ...)

*Nicht-Funktionale Anforderungen:*

- Technische Anforderungen  
(Effizienz, Lastverhalten, ..., Entwurfsbeschränkungen aufgrund der Zielplattform, Software-Entwicklungsumgebung)
- Qualitätsanforderungen  
(Benutzbarkeit, Zuverlässigkeit, Sicherheit, Änderbarkeit, ...)

Wir werden uns im Folgenden auf die funktionalen Anforderungen konzentrieren.

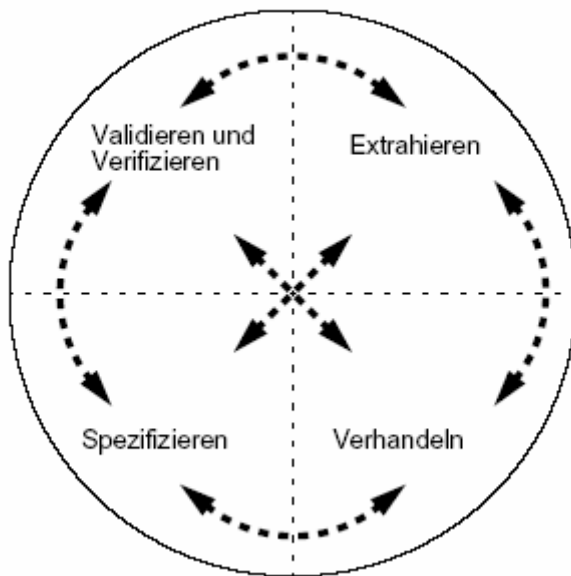
Bei der Anforderungsermittlung stehen die Fragen nach dem

- „Was“ (Funktionen) dem
- „Warum“ (Ziele) und dem
- „Womit“ (z.B. Objekte und Beziehungen der Problemwelt)

im Vordergrund.

Die Frage nach dem „Wie“ spielt grundsätzlich erst in darauffolgenden Entwicklungsaktivitäten eine Rolle. Die Trennung von Verantwortlichkeiten dient natürlich der Komplexitätszerlegung. Dennoch ist es nicht immer möglich alle Realisierungsfragen gänzlich außer Acht zu lassen. So sind z.B. oft Schnittstellen zu vorhandenen Systemen und dabei bestimmten Datenformaten und Protokollen zu berücksichtigen.

Die Anforderungsermittlung gliedert sich grundsätzlich in die folgenden Teilaufgaben:



Die Anforderungsermittlung startet mit der **Extraktion** der Anforderungen, den Wünschen und Einschränkungen an das zu erstellende Software-Produkt.

In **Verhandlungen** über Umfang und Art der Anforderungen werden die einzelnen Anforderungen zunächst nach Kriterien gewichtet.

Beim **Validieren** (gültig machen) der Anforderungen ist sicherzustellen, dass die Anforderungsspezifikation vollständig ist und die Erwartungen des Auftraggebers und der Benutzer trifft.

Beim **Verifizieren** geht es um die Überprüfung, ob die verschiedenen Einzelspezifikationen korrekt und

untereinander konsistent sind. Beim Verifizieren geht es um die Überprüfung, ob die verschiedenen Einzelspezifikationen korrekt und untereinander konsistent sind.

Die Validierung bzw. Verifikation der Anforderungsermittlung ist von großer Bedeutung, da durch eine frühzeitige Aufdeckung von Fehlern Kosten eingespart werden können.

Das Ergebnisdokument der Anforderungsermittlung ist die so genannte **Anforderungsspezifikation** - ein Dokument bestehend aus dem Anwendungsfallmodell, dem Domänenklassenmodell und den Verhaltensmodellen.

Insgesamt ist festzuhalten, dass diese erste Aktivität der Software-Entwicklung von äußerster Wichtigkeit ist. Diese Aktivität legt (neben der Analyse) die Basis alles Weiteren, d.h. hier kann vermieden werden, dass später an den eigentlichen Anforderungen „vorbeientwickelt“ wird.

Von einer objektorientierten Anforderungsermittlung sprechen wir, wenn das Ergebnis eine Anforderungsspezifikation ist, die in jedem Fall ein Anwendungsfallmodell und ein Klassenmodell sowie oft auch Verhaltensmodelle enthält.

Das Motto in der Anforderungsermittlung muss lauten:

***Nicht zu viel auf einmal und alles zu seiner Zeit.***

Jede Modellierung innerhalb der Anforderungsspezifikation sollte sich in natürlicher Weise aus der Problemwelt ergeben, so dass die Sichtweise der Anwender wiedergegeben wird. Schließlich kann nur der Anwender die Anforderungen bzw. die Modellierung validieren. Diese Eigenschaft einer Modellierung nennen wir **Problemadäquatheit**. Alle unnatürlich erscheinenden Modellierungen sollten hinterfragt und geprüft werden.

Eine zu frühe zu detaillierte Spezifikation kann einen hohen Änderungsaufwand in späteren Aktivitäten nach sich ziehen. Auch hier sollte man obiges Motto beherzigen.

Ein typisches Vorgehensmuster bei einer Anforderungsermittlung ist folgendes:

1. Erstellung des Lastenheftes
2. Analytiker schaffen sich ein Grundverständnis und einen Überblick über die Domäne (Teil der Problemwelt), welche durch das Lastenheft spezifiziert wurde.
3. Analytiker besprechen konkrete Szenarien der Problemwelt mit den Anwendern.
4. Analytiker destillieren Anwender bzw. Akteure, Szenarien bzw. Anwendungsfälle und Objekte bzw. Klassen aus den Gesprächen, dem Lastenheft und sonstigen gewonnenen Erkenntnissen.
5. Vorläufige Abstraktionen werden zusammen mit dem Anwender diskutiert.

Dabei können die Punkte 2 – 5 mehrmals oder aber auch in differenter Reihenfolge durchlaufen werden.

### 3. Anwendungsfallmodellierung

Wie wir bereits festgestellt haben ergeben sich die **Anwendungsfälle** aufgrund des Lastenheftes insbesondere aus den Szenarien also aus konkreten Abläufen. Zunächst werden die Szenarien, die zu einer im Lastenheft genannten Funktion gehören, zu einigen wenigen Anwendungsfällen zusammengefasst, die jeweils wesentliche Teilfunktionen abdecken.

Ist ein so entstandener Anwendungsfall zu umfangreich, versucht man, ihn in kleinere Anwendungsfälle zu zerlegen, wobei man sich an den jeweils beteiligten Akteuren und dem zeitlichen und räumlichen Zusammenhang der innerhalb des Anwendungsfalls vorkommenden Interaktionen orientiert. Dieser Prozess wird fortgesetzt, bis jeder Anwendungsfall von den Akteuren unter Beachtung weniger, einfacher Geschäftsregeln in einer zusammenhängenden Interaktion mit dem Anwendungssystem bearbeitet werden kann.

*Jeder Anwendungsfall behandelt eine klar abgegrenzte Aufgabe und liefert ein relevantes Ergebnis.*

Trifft man bei der Zerlegung auf Anwendungsfälle, denen kein Akteur zugeordnet werden kann, so ist die Funktionalität wahrscheinlich zu fein aufgegliedert. Auf jeden Fall muss man sich fragen, für wen bzw. für was denn das Ergebnis eines Anwendungsfalls ohne Akteur relevant ist.

*Anwendungsfälle ohne Akteur sind suspekt.*

Wichtig ist, dass Anwendungsfälle die Systembenutzung aus der Sicht der Benutzer beschreiben und nicht etwa lediglich eine Auflistung der vom System angebotenen Funktionen darstellen.

*Anwendungsfälle beschreiben die Systembenutzung nicht das System.*

Unter ständiger Rücksprache mit Anwendern, welche die Rollen der entsprechenden Akteure spielen können, erstellen Analytiker die textuellen Beschreibungen der Anwendungsfälle. Keinesfalls sollten die Anwendungsfälle von den Anwendern selbst erstellt werden. Dies würde den Transfer des Anwendungswissens zu den Analytikern behindern und evtl. zu impliziten Annahmen oder Anforderungen führen.

*Anwendungsfälle werden von Analytikern, nicht von Anwendern geschrieben.*

Andererseits müssen die von den Analytikern verfassten Anwendungsfälle auch für die Anwender verständlich sein, damit diese die fachliche Vollständigkeit und Korrektheit validieren können.

*Einfachheit und Problemadäquatheit der Anwendungsfälle gehen vor der Eleganz des Anwendungsfallmodells.*

Jeder Anwendungsfall wird textuell spezifiziert, wobei auch eine Vorbedingung und Nachbedingung notiert werden können. Die Vorbedingung gibt die minimalen Voraussetzungen an, mit denen der Anwendungsfall begonnen werden kann. Die Nachbedingungen präzisieren das „normale“ Ergebnis des Anwendungsfalls und mögliche Ausnahmen.

#### 4. Glossar, Verbindungen, Akteure und Pakete

Um eine reibungslose Kommunikation zu ermöglichen, wird ein **Glossar** erstellt, das eine zentrale Sammlung der Definitionen aller wichtigen Gegenstände und Sachverhalte der Domäne beinhaltet. Ein Glossar hat zur Hauptaufgabe alle am Projekt beteiligten Entwickler sozusagen „gleichzuschalten“.

Die Begrifflichkeiten die im Glossar festgehalten werden, müssen klar und eindeutig formuliert sein, so dass Fehlinterpretationen vermieden werden und damit unangenehme Überraschungen bei der Einführung des Systems nach Möglichkeit ausgeschlossen werden.

Nicht nur unklare und widersprüchliche Anforderungen auch fehlende oder *unnötige Anforderungen* führen zu erheblichem Mehraufwand in Design und Entwicklung. Nicht berücksichtigte, nicht schriftlich formulierte oder falsch umgesetzte Anforderungen können sogar zur Änderung der gesamten Softwarearchitektur führen. Dies ist besonders teuer, wenn dieser Umstand erst bei Einsatz der Software entdeckt wird.

In der Anwendungsdomäne werden die Benutzer des zu entwickelnden Systems identifiziert und als **Akteur** modelliert. Ein Benutzer kann dabei ein unabhängiges System, eine Maschine aber auch ein Menschen sein, d.h. ein Anwender. Folgende Grundregel sollte dabei eingehalten werden:

*Für jeden menschlichen Akteur muss mindestens eine Person existieren, welche die Rolle des Akteurs spielen kann.*

Als Ergebnis der Betrachtungen bezüglich des Akteurs erhält man zum einen die Akteure selbst und zum anderen einen Überblick über die von ihnen benutzten Funktionen des Anwendungssystems.

Teilszenarien, die in Szenarien unterschiedlicher Anwendungsfälle vorkommen, können zu selbständigen Anwendungsfällen zusammengefasst und mit einer include- oder extend-Beziehung wiederverwendet werden. Optionale oder nicht-optionale Abhängigkeiten nennen wir **Verbindungen**. Auch hier ist auf Problemadäquatheit zu achten, d.h. die Beziehungen müssen sich natürlich aus der Problemwelt ergeben und nicht künstlich herbeigeführt werden.

*Beziehungen (Verbindungen) zwischen Anwendungsfällen sind möglichst spät und sparsam einzusetzen.*

Wichtig ist, dass Abläufe nur innerhalb eines Anwendungsfalls angegeben werden. Die Reihenfolgen, in der verschiedene Anwendungsfälle ausgeführt werden können, sind nicht vorrangiges Ziel der Anwendungsfallmodellierung und werden nur indirekt von den include- und extend-Beziehungen widergespiegelt.

*Beziehungen zwischen Anwendungsfällen modellieren keine Abläufe, sondern (statische) funktionale Zerlegungen.*

In großen Anforderungsspezifikationen werden die Anwendungsfälle zu handhabbaren Paketen zusammengefasst. Auch hier orientiert man sich grundsätzlich an den fachlichen Gegebenheiten der Problemwelt (Problemadäquatheit).

*Pakete von Anwendungsfällen werden nach ihrer fachlichen Zusammengehörigkeit gebildet.*

Weiterhin sollte man darauf achten, dass zwischen Anwendungsfällen aus verschiedenen Paketen nur minimale Abhängigkeiten bestehen.

## 5. Domänen-Klassenmodell

Bereits bei der Ermittlung und Beschreibung von Anwendungsfällen stößt man laufend auf Gegenstände und Sachverhalte, welche für die strukturelle Modellierung wesentlich sind. Man notiert diese potentiellen Objekte, Klassen, Attribute und Assoziationen, aber auch Arbeitsabläufe, Ziele bestimmter Abläufe, mögliche Skizzen zu Benutzungsoberflächen oder zu berücksichtigende existierende Systeme und Schnittstellen.

Die beim Studium der Szenarien und Anwendungsfälle identifizierten Objekten und Klassen, aber auch die im Lastenheft und Glossar aufgeführten Gegenstände und Sachverhalte der Domäne, stellen den Ausgangspunkt für die Domänen-Klassenmodellierung dar. Ferner unterstützen die während der Anwendungsfallmodellierung erstellten, rudimentären Klassendiagrammskizzen den eigentlichen Entwurf.

Auch Attribute und Eigenschaften der Klassen bzw. Objekte werden bereits parallel zu den Anwendungsfällen ermittelt.

Eine wichtige Eigenschaft des Domänen-Klassenmodells besteht im **Fehlen jeglicher Operationen**, da über Operationen grundsätzlich bereits Realisierungsaspekte einfließen. Das Aukleiden des Domänen-Klassenmodells mit Operationen erfolgt schrittweise in den nachfolgenden Entwicklungsaktivitäten.

Vor allem sollte die Modellierung des Domänen-Klassenmodells *problemadäquat* sein.

So sind insbesondere Generalisierungen als problematisch zu betrachten, sie können zwar zur Redundanzvermeidung und somit zu kompakteren, übersichtlicheren Modellen beitragen, andererseits gibt es aber oft in der Problemwelt keinen Gegenstand oder Sachverhalt, der genau die herausfaktorierten Eigenschaften besitzt. So kann bspw. der Fall eintreten, dass in der Problemwelt keine Rolle bzw. Funktion existiert, die einem allgemeineren Akteur bzw. Anwendungsfall entspricht.

Zusammenfassend ist festzuhalten:

*Jede Modellierungsaktivität in der Anforderungsermittlung orientiert sich ausschließlich an den Gegebenheiten und Erfordernissen der Problemwelt (Problemadäquatheit) und darf nicht Selbstzweck werden und sollte möglichst keine Ge-*

*sichtspunkte der Realisierung vorwegnehmen. Insbesondere sind Modellelemente zu hinterfragen, die kein Pendant in der Problemwelt besitzen.*

Das Domänenklassenmodell bildet also Strukturen aus der Problemwelt ab. Abhängig vom Vorgehensmodell kann die gesamte Problemwelt auf einmal (z.B. Wasserfallmodell) oder jeweils nur einen Ausschnitt (z.B. RUP) Gegenstand der Modellierung sein. Wie für jedes Modell der Anforderungsspezifikation bildet auch hier die Problemadäquatheit das zentrale Kriterium.

*Jedes Element des Domänen-Klassenmodells sollte ein Pendant besitzen, das ein relevanter Gegenstand oder Sachverhalt der Problemwelt ist. Ausnahmen von dieser Regel müssen hinterfragt werden und sind zu begründen.*

Außerdem ist in den Gesprächen aber auch in Dokumenten auf häufig verwendete Begrifflichkeiten zu achten, um eventuelle Klassen zu identifizieren. Auf diese Weise erreicht man eine Vielzahl an möglichen Klassen, die dann auf das richtige Maß reduziert werden muss.

*Auf jede Klasse des Domänen-Klassenmodells muss in mindestens einem Anwendungsfall oder Szenario Bezug genommen werden.*

Mit fortschreitender Modellierung werden die gefundenen Klassen mit Attributen und Assoziationen versehen. Zusätzlich notiert man die Verantwortlichkeiten jeder Klasse, d.h. ihren Beitrag bzw. ihre Aufgaben in der Problemwelt.

## 6. Modellbereinigung

Bevor man zu den meist aufwändigen textuellen Spezifikationen und zur eventuellen Zusammenfassung von Klassen zu Paketen kommt, wird das Domänen-Klassenmodell nach irrelevanten Elementen durchsucht. Im Wesentlichen wendet man dazu die folgenden Merkgeregeln möglichst in der angegebenen Reihenfolge an.

Klassen mit nur einem Attribut sind in jedem Fall genau zu inspizieren. Gegebenenfalls kann das Attribut einer anderen Klasse zugeteilt und die jetzt überflüssige Klasse aus dem Modell entfernt werden.

*Jede Domänenklasse sollte mehr als ein Attribut besitzen.*

Attribute und Assoziationen, die aus anderen Sachverhalten abgeleitet bzw. berechnet werden können, enthalten redundante Informationen und sollten daher als abgeleitet gekennzeichnet werden.

*Ableitbare Attribute und Assoziationen einer Klasse sollten als abgeleitet markiert werden.*

Auch Klassen mit nur einem Objekt (Multiplizität) bedürfen einer kritischen Überprüfung. Sie können sinnvoll sein, wenn sie die Problemwelt reflektieren. Oft sind sie aber ein Hinweis auf eine zu feine Klassenzerlegung.

*Zu jeder Domänenklasse sollte im Normalfall mehr als ein Objekt in der Problemwelt existieren.*

Wie bei großen Anwendungsfalldiagrammen bündelt man bei umfangreichen Klassendiagrammen zusammengehörige Klassen zu Paketen, um dem Betrachter den Überblick und das Verständnis zu erleichtern. Eine Zusammenfassung von Klassen nach ihrer fachlichen Zusammengehörigkeit (hohe Kohäsion), die sich soweit wie möglich auf der Zu-

sammenfassung der Anwendungsfälle abstützt, wird diesem Gesichtspunkt am ehesten gerecht.

*Package von Domänenklassen werden nach der fachlichen Zusammengehörigkeit gebildet.*

## 7. Verhaltensmodellierung

Während der Anforderungsermittlung können für Anwendungsfälle mit komplexeren externen Interaktionen Interaktionsdiagramme entwickelt werden, um ihre Abläufe (genauer: ihre Szenarien) zu präzisieren.

Aussagekräftige Verhaltensmodelle, die präziser als die (in der Anforderungsspezifikation) üblichen textuellen Beschreibungen sind, fallen normalerweise feingranularer als die anderen Modelle der Anforderungsspezifikation aus.

Die aussagekräftige Modellierung der innerhalb eines Szenarios auftretenden Veränderungen von Domänen-Objekten greift im Allgemeinen detailliert auf Objekte sowie auf Attribute und Assoziationen von Klassen des Domänen-Klassenmodells zurück. Voraussetzung hierfür ist eine entsprechende Detaillierung des Domänen-Klassenmodells, die oft nicht gegeben ist.

Will man also ein aussagekräftiges Verhaltensmodell, so erfordert dies einen gewissen Grad an Detailliertheit. Mögliche spätere Änderungen würden den Entwicklungsaufwand ungemein erhöhen, so dass die Erstellung von Verhaltensmodellen in jedem Fall *zurückgestellt werden sollte*, bis die Anforderungsspezifikation eine gewisse Stabilität erreicht hat.

Dies ist einer der Hauptgründe, warum das ablauforientierte Verhalten von Anwendungsfällen vergleichsweise *selten* (schon) in der Anforderungsspezifikation modelliert wird.

In der Anforderungsermittlung werden einerseits

- die *externen Interaktionen* beim Ablauf von Anwendungsfällen und andererseits
- das *zustandsorientierte Verhalten* komplexer Objekte modelliert.

Die Modellierung des ablauforientierten Verhaltens von Operationen und Anwendungsfällen mit Hilfe von Interaktionsdiagrammen reicht aber nicht aus, wenn Objekte mit einem komplexen Zusammenspiel von Zuständen und Operationen auftreten.

Für Domänenklassen, deren Instanzen ein kompliziertes zustandsorientiertes Verhalten zeigen, kann die Erstellung von Zustandsdiagrammen sinnvoll sein.

Dieses so genannte zustandsorientierte Verhalten von Objekten modelliert man mit Hilfe von Zustandsdiagrammen, welche aus Zuständen und Ereignissen bestehen.

Der **Zustand** eines Objektes zu einem bestimmten Zeitpunkt ist durch die vorliegende Kombination seiner konkreten Attributwerte und Verbindungen zu anderen Objekten determiniert.

Grundlage der Modellierung des zustandsorientierten Verhaltens in der Anforderungsspezifikation sind Geschäftsregeln, welche die Zustände der bearbeiteten Objekte berücksichtigen.

Ebenso wie bei der Modellierung des ablauforientierten Verhaltens ist auch der Grad der Detaillierung beim zustandsorientiertem Verhalten für ein aussagekräftiges Modell recht hoch. Entsprechend können alle Anmerkungen zum ablauforientierten Verhalten analog übernommen werden. Deshalb werden derartige Modellierungen oftmals erst in der Analyse durchgeführt. In technischen Anwendungsdomänen stellt sich jedoch die Situation

aber oft anders dar. Hier existieren Objekte, die technische Geräte modellieren, welche von Hause aus ein wohldefiniertes zustandsorientiertes Verhalten aufweisen oder aufweisen sollen.

## 8. Validierung und Verifikation

Bei der Validierung geht es um die Frage, ob die in der Anforderungsspezifikation festgehaltenen Anforderungen den Vorstellungen und Wünschen der Anwender entsprechen. Da Anwender im Wesentlichen „funktional“ denken und keine formalen Spezifikationen verstehen, beschränkt sich die Validierung auf das Anwendungsfallmodell. Bei der Validierung geht es also um die Frage „Are we building the right product?“.

Die Validierung bildet einen eminent wichtigen Bestandteil der Softwareentwicklung. Ein schlichtes Durchgehen durch die textuelle Spezifikationen der Anwendungsfälle in Form von so genannten Reviews hat sich nicht nur als vergleichsweise wenig ergiebig, sondern auch als für die Anwender überaus ermüdend erwiesen.

Deshalb geht man in zwei Schritten vor:

1. Die Anwender prüfen in Reviews die textuellen Beschreibungen der Anwendungsfälle. Zur Unterstützung bieten sich Checklisten an.
2. Im zweiten Schritt werden für jeden Anwendungsfall ausgewählte Szenarien und die betroffenen Domänenobjekte in Walk-Throughs mit den Anwendern durchgespielt und diskutiert. Eine visuelle Aufbereitung der Szenarien und der betroffenen Domänenobjekte verbessert die Anschaulichkeit der Anwendungsfälle, die damit von den Anwendern besser beurteilt werden können.

Empfehlenswert ist auch die Zuhilfenahme der Skizzen und Mock-ups der Benutzungsoberflächen. Hierdurch wird nicht nur die Validierung der Anwendungsfälle erleichtert, sondern zugleich die grundsätzliche Gestaltung der Benutzungsoberfläche selbst validiert. Natürlich können Prototypen ebenso zur Validierung beitragen.

Unter der Verifikation eines Modells versteht man den Abgleich hinsichtlich ihrer Vollständigkeit und Konsistenz. Es geht also um die Frage „Are we building the product right?“. Allerdings sollte man in dieser frühen Phase der Softwareentwicklung von der Verifikation nicht allzu viel erwarten, da bspw. das Domänen-Klassenmodell noch recht unvollständig ist.