

Die Analyse

1.0 Motivation und Überblick

Die Anforderungsermittlung abstrahiert die Problemwelt und bildet diese in der Anforderungsspezifikation in möglichst natürlicher Weise ab. Dabei wurde hauptsächlich nach dem „Was“ gefragt.

Insbesondere bei großen Software-Projekten hat sich gezeigt, dass eine Anforderungsspezifikation als Voraussetzung für die Realisierung nicht hinreichend ist. Hier empfiehlt es sich einen *Zwischenschritt*, die Analyse, einzuschieben. In der Analyse wird die Anforderungsspezifikation im Hinblick auf Realisierungserfordernisse aufgearbeitet und präzisiert. Die Anforderungsermittlung ohne die Tätigkeiten der Analyse enthält eine gewisse Unschärfe bezüglich der Funktions- und Verhaltensmodellierung; im Gegensatz dazu kann die strukturelle Modellierung der Anforderungsermittlung i.d.R. unverändert übernommen werden.

Die Analyse bildet nun den Brückenschlag zwischen Anforderungsspezifikation und Realisierung. Es wird also die Anforderungsspezifikation überarbeitet, weiter strukturiert und präzisiert und in die **Analysespezifikation** überführt, die selbst wiederum aus

- dem Analyse-Klassenmodell,
- Interaktionsdiagrammen,
- Zustandsdiagrammen und
- Dokumenten besteht, in denen die Durchführung und die Ergebnisse der Verifikationsaktivitäten festgehalten sind.

Die im Rahmen der Analyse durchgeführten **Tätigkeiten** werden im Folgenden grob skizziert. Wichtig ist, dass die Tätigkeiten nicht streng sequentiell, sondern teilweise iterativ und parallel erfolgen:

- Die in der Anforderungsspezifikation festgehaltenen **Anwendungsfälle werden in Klassen überführt**, d.h. die funktionalen Anforderungen werden als Verantwortlichkeiten von Klassen formuliert. In der Anforderungsermittlung wurden das Funktionale, das Zustandsorientierte und das Strukturelle voneinander getrennt untersucht - jetzt wird alles zusammengeführt.
- Die in der Anforderungsspezifikation festgehaltenen Anwendungsfälle liefern die Basis, um das Domänen-Klassenmodell mit **Operationen** auszukleiden, um dadurch zusammen mit den neuen Klassen das **Analyse-Klassenmodell** zu entwickeln. Hand in Hand damit wird das ablauforientierte Verhalten komplexer Anwendungsfälle mit Mechanismen und Interaktionsdiagrammen, die auf dem Analyse-Klassenmodell basieren, spezifiziert.
- Falls nötig werden existierende Zustandsdiagramme verfeinert und/oder neu erstellt.
- Abschließend findet eine **Verifikation** statt, bei der die einzelnen Modelle in sich auf Konsistenz und Vollständigkeit überprüft, sowie gegeneinander abgeglichen werden.

Alles zusammen mündet dann in der *Analysespezifikation*, welches die Anforderungsspezifikation ergänzt und weiterführt. Da die Analysespezifikation zusammen mit dem Anwendungsfallmodell der Anforderungsspezifikation die zentrale Ausgangsbasis der Realisierung darstellt, muss die Analysespezifikation ausführlich verifiziert werden.

Die Analyse hat zwei grundsätzliche Aufgaben. Die Analyse...

- entlastet die vorangegangene Anforderungsspezifikation von zu realisierungsbezogenen und feingranularen Aspekten.
- liefert ergänzende, präzisere und verifizierende Dokumente für die nachfolgende Realisierung.

Bei kleineren Projekten mit weniger komplexen Anforderungen sowie bei einzelnen Iterationen in Vorgehensmodellen wie dem RUP kann es sinnvoll sein, auf die Analyse als Aktivität zu verzichten. Man mache sich aber klar, dass keine systematische Softwareentwicklung auf die hier zur Analyse zusammengefassten Tätigkeiten völlig verzichten kann. Sie werden lediglich auf die Anforderungsermittlung und die Realisierungsaktivitäten aufgeteilt und nicht als eigenständige Aktivität aufgefasst.

Diese Zusammenfassung basiert auf dem Skript von **Herrn Prof. Dr. H.W. Six**, Lehrgebiet Software Engineering, der [FernUniversität in Hagen](#).

2.0 Heuristiken für die Klassenmodellierung

Unter den Modellen der Analyse besitzt das Klassenmodell die größte Komplexität und Bedeutung. Dieses Kapitel enthält einige Hinweise (Heuristiken) zur Erstellung sauber strukturierter Analyse-Klassenmodelle.

Erscheinen im Laufe der Modellierung mehrere Alternativen als gleichwertig, so entscheide man sich vorläufig für irgendeine der Alternativen. Dabei sollten die Vor- und Nachteile der einzelnen Alternativen dokumentiert werden, damit bei späteren Problemen daraufhin reagiert werden kann.

1-1 Assoziationen sollten daraufhin überprüft werden, ob die beiden in Verbindung stehenden Klassen nicht zu einer einzigen zusammengefassten Klasse umgeschrieben werden können.

Zwei verschiedene Klassen sind die bessere Wahl, wenn die 1-1 Assoziation zwischen diesen beiden Klassen in einer oder beiden Richtungen optional, d.h. mit der Multiplizität 0..1, spezifiziert ist.

Zwei verschiedene Klassen sind die bessere Wahl, wenn eine Instanz eine Verbindung zu Instanzen noch andere Klassen eingehen kann.

D.h. dieses Objekt wird noch anderweitig gebraucht und ist deshalb besser in einer eigenständigen Klasse aufgehoben.

Wie ist vorzugehen, wenn die Klasse, die in zwei Klassen mit einer 1:1 Assoziation aufgeteilt werden soll, bereits an Assoziationen beteiligt ist? In diesem Fall muss entschieden werden, wie die ursprüngliche Assoziation am sinnvollsten mit den beiden anderen Klassen zu verknüpfen sind.

Wird eine Klasse in die ursprüngliche Assoziation und eine neue Klasse mit einer neuen 1-1 Assoziation zwischen den beiden aufgeteilt, bleibt eine ursprüngliche Assoziation weiterhin auf die ursprüngliche Klasse bezogen, wenn Objektverbindungen bez. dieser Assoziation nicht geändert werden müssen, sobald sich eine Objektverbindung bez. der neuen 1-1 Assoziation ändert.

Unterscheiden sich Objekte in nur wenigen Eigenschaften, hängt die Entscheidung, ob eine Generalisierung sinnvoll ist, wesentlich davon ab, worin und wie sehr sich die Objekte unterscheiden.

*Lassen sich Objekte mit denselben Attributen und demselben Verhalten disjunkt klassifizieren (d.h. in bestimmte Arten aufteilen) so werden sie in einer Klasse zusammengefasst und mit **wenigen** zusätzlichen Attributen unterschieden. Insbesondere wenn sich die unterscheidenden Eigenschaften dynamisch ändern können, wird die Modellierung mit Attributen vorgezogen.*

Das dynamischen Eigenschaften besser als Attribut denn als Assoziation zu einer entsprechenden Klasse modelliert werden soll, liegt daran, dass sich Attribute schnell und bequem ändern lassen, dagegen lässt sich die Zugehörigkeit zu einer Klasse nicht so einfach ändern. Bei kleineren Unterschieden ist die Attribut-Methode vorzuziehen.

Wenn sich die Objekte bezüglich Attribute und Verhalten unterscheiden, wird u.U. die Generalisierung benutzt. Sie wird auch für Spezialfälle verwendet, wenn sich die Objekte nur bezüglich ihrer Attribute unterscheiden und nicht bezüglich ihres Verhaltens und umgekehrt.

Im Zweifelsfall werden die Objekte zunächst anhand der Attributwerte unterschieden. Sollte sich im Laufe der Modellierung herausstellen, dass sich in Abhängigkeit von den Attributwerten unterschiedliche Verhaltensweisen oder Verbindungen ergeben, kann man immer noch zur Unterklassenbildung greifen.

3.0 Klassenmodellierung

Trotz einer gewissen Realisierungsnähe stellen Analyseklassen hinsichtlich folgender Aspekte immer noch Abstraktionen von Teilsystemen, Modulen oder Klassen des Entwurfs oder der Implementation dar.

- Fokus auf die funktionalen Anforderungen, d.h. hier wird die Auskleidung des Klassenmodells mit den verschiedenen Methoden vorgenommen.
- Attribute werden aus dem Problembereich entnommen und das Domänenklassenmodell evtl. daraufhin ergänzt. Keinesfalls werden Typen auf programmiersprachlicher Ebene angegeben.
- Beziehungen zwischen den Analyseklassen ausgehend von der Problemwelt abstrahieren, dabei werden keine Navigierbarkeiten bei Assoziationen angegeben.
- Generalisierungen werden nur zur konzeptuellen Strukturierung und nicht zur reinen Redundanzvermeidung eingesetzt.
- Verhalten von Analyseklassen wird soweit wie möglich in Verantwortlichkeiten in textueller Form skizziert. Jede Verantwortlichkeit umreißt einen inhaltlich zusammenhängenden Teil des Verhaltens der Klasse.
- Bei der Angabe von Nicht-Standard-Operationen besteht die Gefahr, zu stark in Realisierungskategorien zu denken. Oft reichen zur Beschreibung einer Analyseklasse die Standardoperationen aus. Darüber hinausgehende (Nicht-Standard-) Operationen werden nur aufgenommen, wenn sie zum Verständnis der Verantwortlichkeiten der zugehörigen Klassen notwendig sind.
- Präzise Schnittstellenbeschreibungen mit Signaturen werden nur für komplexe (Nicht-Standard-) Operationen angegeben.

Jede der Klassen im Analyse-Klassenmodell lässt sich in eindeutiger Weise einer der folgenden Klassenarten zuordnen:

- Schnittstellenklassen (boundary classes)

- Kontrollklassen (control classes)
- Entitätsklassen (entity classes)

Diese haben folgende UML-Notation (graphische Stereotype):



Entitätsklassen repräsentieren (persistente) langlebige Informationen im Anwendungssystem. Sie umfassen damit alle Domänenklassen der Anforderungsspezifikation, d.h. das Domänenklassenmodell kann vergleichsweise schematisch übernommen werden. Zusätzlich können noch weitere, eher systembezogene Klassen als Entitätsklassen aufgenommen werden. Diese ergeben sich z.B. aus Zugangsbeschränkungen oder zu berücksichtigenden technischen Sachverhalten.

Die in der Anforderungsermittlung gewonnenen Anwendungsfälle werden in der Analyse in Schnittstellenklassen bzw. Kontrollklassen transformiert.

Schnittstellenklassen bündeln die Interaktionen zwischen Akteuren und dem Anwendungssystem. Sie dienen zur

- Auswahl von Anwendungsfällen und
- Unterstützung der externen Interaktion
- Repräsentation und Akteur-gesteuerten Modifikation von Daten und
- Umsetzung von Kommunikationsprotokollen im Falle technischer Akteure.

Hauptaufgabe der Schnittstellenklassen ist also, die Interaktionen von Akteuren in anwendungsinterne Ereignisse zu übersetzen und umgekehrt Anwendungsdaten, an denen die jeweiligen Akteure interessiert sind, in eine für die Akteure verwertbare Form zu überführen.

Für jeden Akteur definieren wir eine **Akteur-Schnittstellenklasse** (kurz: **AS-Klasse**), welche dem Akteur erlaubt, die ihm zur Verfügung stehenden Anwendungsfälle zu aktivieren.

Im Folgenden konzentrieren wir uns auf menschliche Akteure.

Für jeden Anwendungsfall und jeden am Anwendungsfall beteiligten Akteur wird eine **Anwendungsfall-Akteur-Schnittstellenklasse** (kurz: **AAS-Klasse**) modelliert, über welche die Interaktion des Akteurs mit dem AWF abgewickelt wird.

Die in den Anwendungsfällen zugrunde liegende Logik wird in *jeweils* einer **Kontrollklasse** pro Anwendungsfall verkapselt.

Zu den Verantwortlichkeiten der Instanz einer Anwendungsfall-Kontrollklasse gehört hauptsächlich die Kontrolle der an dem Anwendungsfall beteiligten (Instanzen von) Entitätsklassen gemäß der fachlichen Logik.

Anwendungsfall-Kontrollklassen spielen damit eine Art *Vermittlerrolle* zwischen Schnittstellen- und Entitätsklassen.

4.0 Operationen

Um die Anforderungsspezifikation frei von Realisierungsaspekten zu halten, wurde bei der Anforderungsermittlung bewusst auf die Modellierung von Operationen im Domänenklassenmodell verzichtet. Dies wird jetzt in der Analyse nachgeholt.

Zur **Auskleidung des Analyse-Klassenmodells mit Operationen** liefern die in der Anforderungsspezifikation festgehaltenen Anwendungsfälle zusammen mit den als Interaktionsdiagrammen notierten externen Interaktionen die Basis.

Jede in den Szenarien des Anwendungsfalls auftretende externe Interaktion *zerfällt bei genauerer Betrachtung in einzelne, fachlich separierbare Aktionen*. Jeder dieser Aktionen sollte eine Operation entsprechen, welche die Aktion im Analyse-Klassenmodell umsetzt. Nun kann es sich dabei **entweder** um Standard- **oder** Nicht-Standard-Operationen handeln. Ist dies eine Nicht-Standard-Operation, die bisher noch nicht in Erscheinung getreten ist, so wird sie in das Analyse-Klassendiagramm aufgenommen und ihre Verantwortlichkeit textuell festgehalten.

Die präzise Modellierung des Ablaufverhaltens von so entstandenen fachlichen Operationen ist Gegenstand des Entwurfs.

Mit der Bestimmung der Operationen geht gleichzeitig die Präzisierung des Verhaltens der Anwendungsfälle einher.

Die Auskleidung des Analyse-Klassenmodells mit Operationen stellt nicht nur einen Schritt in Richtung Realisierung dar, sondern ist Voraussetzung für den in der Verifikation durchzuführenden wichtigen Abgleich der verschiedenen Modelle gegeneinander.

5.0 Verhaltensmodellierung

In der Anforderungsermittlung hat sich die Modellierung des ablauforientierten Verhaltens von komplexen Anwendungsfällen auf externe Interaktionen (Operationsaufrufe) beschränkt.

Die Verhaltensmodellierung in der Analyse verfeinert die in der Anforderungsermittlung erstellten Interaktionsdiagramme in Hinblick auf das Analyse-Klassenmodell.

Die Verfeinerung der Interaktionsdiagramme geht Hand in Hand mit der Auskleidung des Analyse-Klassenmodells mit Operationen. Wir erinnern uns, Operationen wurden in der Aktivität Anforderungsermittlung bewusst nur rudimentär behandelt.

Insgesamt werden wir in zwei Schritten vorgehen:

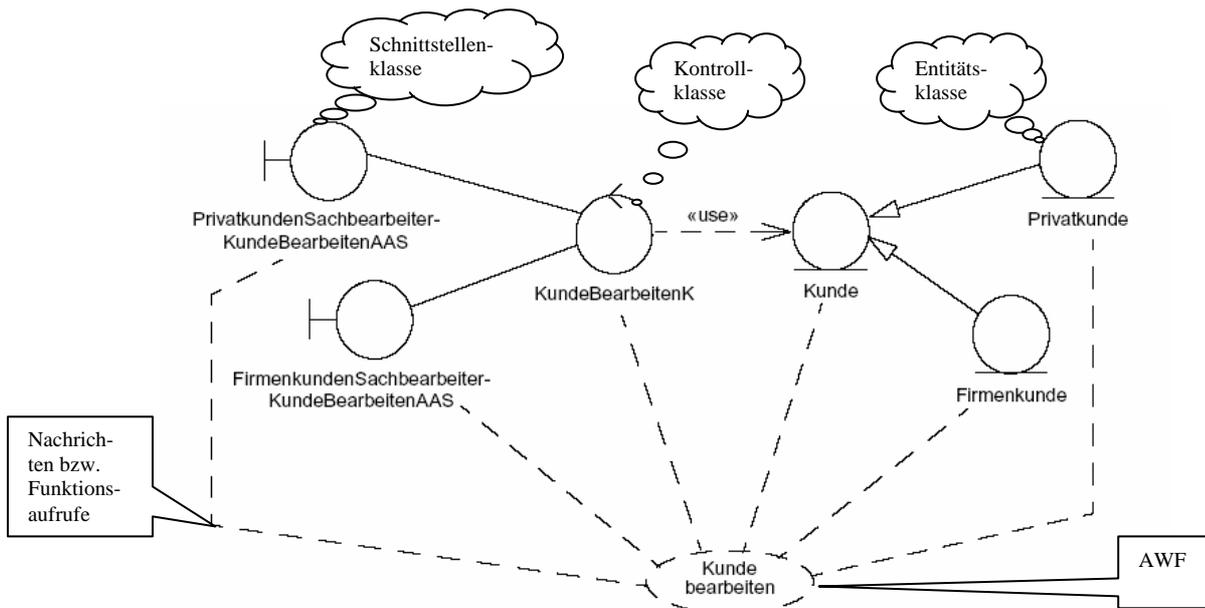
- (1) Zunächst werden die den Anwendungsfall „realisierenden“ Klassen mit einem Mechanismus kenntlich gemacht.
- (2) Anschließend wird das Ablaufverhalten des Anwendungsfalls modelliert, indem für zugehörige Szenarien Kollaborationsdiagramme mit Instanzen von Klassen aus dem Mechanismus angefertigt werden. Stößt man dabei auf Nicht-Standard-Operationen, die noch nicht aufgeführt sind, werden diese ins Klassenmodell aufgenommen.

In den kommenden Zeilen wird nun nicht mehr explizit zwischen Klasse und Objekt unterschieden. Im Einzelfall ist jedoch offensichtlich, ob es sich um eine Klasse oder ein Objekt einer Klasse handeln soll.

Zu (1):

Die Menge aller Objekte die notwendig an der Erfüllung eines Anwendungsfalls innerhalb eines Objektdiagramms vorkommen müssen und die Verbindungen bzw. Abhängigkeiten zwischen diesen wird als sog. **Mechanismus** dargestellt. Ein Mechanismus ist ein partielles Klassendiagramm, in dem der Anwendungsfall selbst gestrichelt angedeutet und durch gestrichelte Linien mit allen an ihm beteiligten Klassen verbunden wird.

Während die an einem Anwendungsfall beteiligten Schnittstellen- und Kontrollklassen unmittelbar zum Mechanismus gehören, müssen die zugehörigen Entitätsklassen noch ermittelt werden. Hierzu empfiehlt sich das Studium der textuellen Beschreibungen des zu realisierenden Anwendungsfalls und insbesondere die darin aufgeführten Vor- und Nachbedingungen.



Hinweise:

- Akteur-Schnittstellen (AS-Klassen) erlauben dem Akteur die ihm zur Verfügung stehenden AWF zu aktivieren
- Nach Auswahl eines AWF wird dieser selbsttätig gesteuert
- Für jeden AWF und jeden am AWF beteiligten Akteur wird eine AWF-Akteur-Schnittstelle (AAS-Klasse) modelliert

Erläuterung zur Skizze:

In der Skizze ist nur ein Teil der Ganzen Entwicklung zu sehen, denn das System erzeugt eine Instanz der Hauptkontrollklasse. Diese identifiziert den Akteur anhand seiner Anmeldung (Login); nach Identifikation wird eine Instanz der entsprechenden Akteur-Schnittstellen-Klasse (AS-Klasse) erzeugt.

Im Anschluss daran hat der Akteur die Wahl zwischen die ihm zur Verfügung stehenden Anwendungsfälle. Nach Auswahl eines Anwendungsfalls wird daraufhin automatisch eine Instanz der Anwendungsfall-Akteur-Schnittstellen-Klasse (AAS-Klasse) erzeugt.

Ab jetzt, d.h. nach Erzeugung der AAS-Klasse, sind die Abläufe in visualisierter Form in obiger Skizze zu sehen.

Wie wir wissen ist eine Instanz der Kontrollklasse nötig, welche zwischen der AAS-Klasse und den beteiligten Entitäten vermittelt. Nur Kontrollklassen dürfen lesend und schreibend auf eine Entität zugreifen.

Die «use»-Beziehung symbolisiert insbesondere die Standardoperationen (z.B. get(), set(), read() oder write()). Es ist darauf zu achten, dass ausgehend von der Schnittstellenklasse keine manipulierende «use»-Beziehung hin zu einer Entität verläuft: dem Ak-

teur soll lediglich lesenden Zugriff auf gespeicherte Daten (sprich Entitätsklassen) gewährt werden.

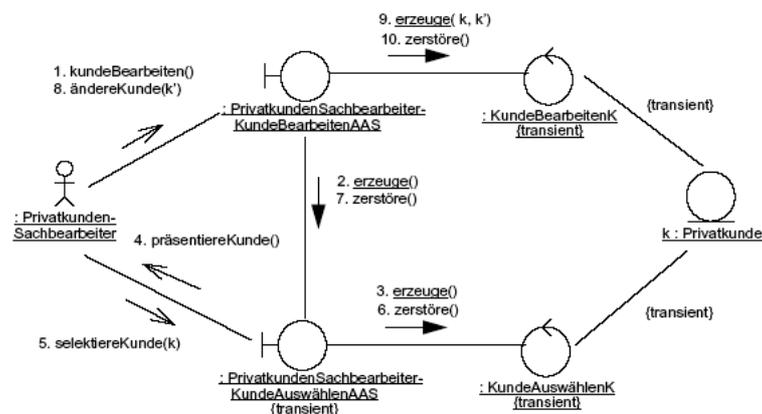
Ist jedoch nur lesender Zugriff vorgesehen, z.B. bei einer gib()-Methode, so spricht grundsätzlich nichts dagegen eine «use»-Beziehung zwischen einer Schnittstellen- und einer Entitätsklasse zu setzen.

Zu (2):

Ausgehend vom Anwendungsfall-Modell der Anforderungsspezifikation werden im zweiten Schritt die Modellierung des **Ablaufverhaltens des Anwendungsfalls vervollständigt** und dabei gleichzeitig das **Analyse-Klassenmodell verfeinert**.

Um dies zu erreichen werden zunächst die Szenarien-Kollaborationsdiagramme angefertigt, die auf dem Mechanismus für den Anwendungsfall aufbauen. Bei komplexen Vorgängen können auch Sequenzdiagramme verwendet werden.

Die Verhaltensmodellierung der Analyse richtet sich nun nicht mehr an die oberflächlichen Operationsaufrufe, sondern an die neu eingeführten AAS-Objekte. Es wird also das Systemobjekt, welche bisher alle Operationsaufrufe in der Anforderungsermittlung „aufgenommen hat“ verfeinert und weiter untergliedert.



Die Erstellung des Kollaborationsdiagramms identifiziert Nicht-Standard-Operationen, welche dann im Analyse-Klassendiagramm nachgetragen werden können.

6.0 Verifikation

Die Verifikation in der Analyse umfasst die Intra-Modell Verifikation und die Inter-Modell Verifikation. Insgesamt geht es um die Frage „Entwerfen wir das richtige Produkt?“.

Bei einer **Intra-Modell Verifikation** werden die einzelnen Modelle hinsichtlich ihrer

- Vollständigkeit
- Eindeutigkeit und
- Konsistenz (strenger gedanklicher Zusammenhang)

überprüft.

Ein Modell ist **formal vollständig**, wenn die Beschreibung seiner Modellelemente die *Spezifikationsrichtlinien* befolgt.

Ein Modell ist **eindeutig**, wenn es nur auf eine Art und Weise *interpretiert* werden kann.

Ein Modell ist **konsistent**, wenn alle Querbezüge auf existierende und „richtige“ Modellelemente verweisen und *keine* redundanten oder gar widersprüchlichen Spezifikationen vorkommen.

Bei der Verifikation werden folgende Modelle bzw. Diagramme untersucht:

- Anwendungsfallmodell
- Klassenmodell
- Interaktionsdiagramme
 - ~ Kollaborationsdiagramme
 - ~ Zustandsdiagramme

Im *Anwendungsfallmodell* werden insbesondere die textuellen Beschreibungen dahingehend überprüft, ob die von include-Beziehungen referenzierten Anwendungsfälle und die in extend-Beziehungen angegebenen Erweiterungspunkte existieren und die textuell angegebenen (und keine anderen) Beziehungen in den Anwendungsfalldiagrammen eingezeichnet sind.

Im *Klassenmodell* ist z.B. zu prüfen, ob alle Assoziationsenden mit Multiplizitäten versehen sind, keine ableitbaren Attribute existieren und in Generalisierungsbeziehungen keine redundanten Features in den Unterklassen enthalten sind.

Bei Interaktionsdiagrammen wird verifiziert, ob z.B. in Sequenzdiagrammen jeder Pfeil mit einem Operationsnamen beschriftet ist und ob jedes nicht ganz oben im Diagramm angeordnete Objekt auch explizit erzeugt wird.

Zum Schluss untersucht man die Zustandsdiagramme. Da diese in den hier betrachteten Anwendungsdomänen relativ einfach sind und nur selten zusammengesetzte Zustände oder komplizierte Bedingungen enthalten, ist hier eine tiefere Untersuchung der Diagramme unnötig.

Bei der **Inter-Modell Verifikation** werden die verschiedenen Modelle gegeneinander abgeglichen:

- Analyse-Klassenmodell und Anwendungsfallmodell
- Analyse-Klassenmodell und Zustandsdiagramme sowie
- Interaktionsdiagramme und Analyse-Klassenmodell.

Die Konsistenz von Analyse-Klassenmodellen und Anwendungsfall-Modellen ist dann gewährleistet, wenn folgende drei Eigenschaften erfüllt sind:

1. Die Interaktionsdiagramme (zu den Szenarien eines Anwendungsfalls) stützen sich vollständig auf Operationen im Klassenmodell ab. In anderen Worten: für jede Aktion in einem Szenario muss es eine Operation geben, die diese Aktion im Klassenmodell umsetzt.
2. Für jede Operation im Klassenmodell gibt es eine Aktion in einem Szenario, die von dieser Operation im Klassenmodell umgesetzt wird. In anderen Worten: jede Operation wird in mind. einem Szenario „benutzt“.
3. Die Spezifikation der Anwendungsfälle und der Operationen im Klassenmodell müssen konsistent sein. Dies ist gegeben, wenn für jedes Szenario eines jeden

Anwendungsfall gilt: die Vorbedingung des Anwendungsfalls stellt die Vorbedingung der Operation im Klassenmodell sicher, welche die erste Aktion des Szenarios umsetzt, und nach der Ausführung aller während des Szenarios aufgerufenen Operationen ist die Nachbedingung des Anwendungsfalls erfüllt.

Abschließend möchten wir betonen, dass sich die Vorgehensweise in der Analyse (und auch im noch folgenden Entwurf) an den Anwendungsfällen orientiert, d.h. nach der vergleichsweise schematischen Übertragung des Domänen-Klassenmodells werden die Anwendungsfälle nach und nach in der Analysespezifikation verankert. Die Orientierung an den Anwendungsfällen wird lediglich von der Modellbereinigung und der Paketbildung durchbrochen, die sich beide auf das gesamte Analyse-Klassenmodell beziehen.