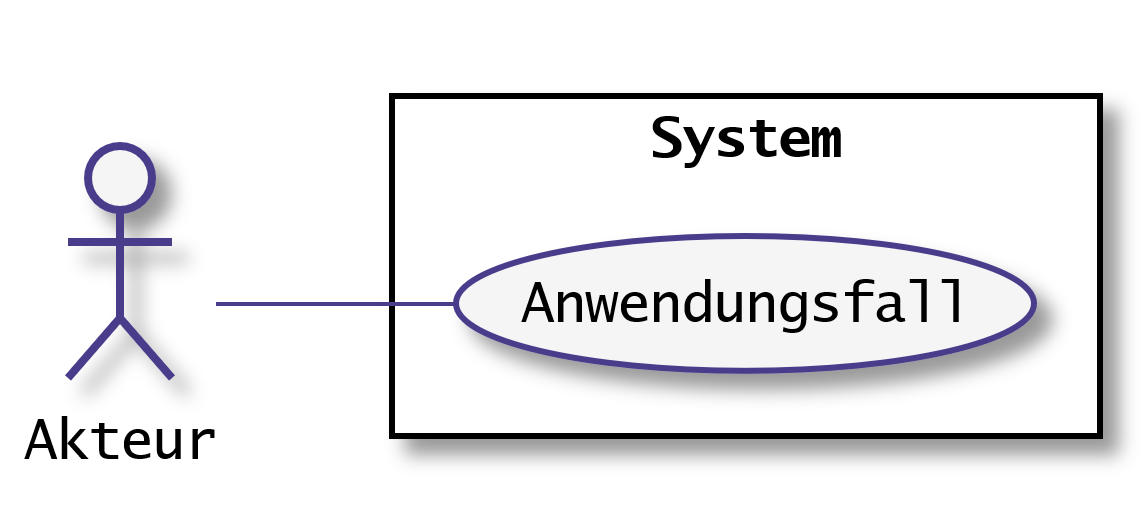
Use-Case-Diagramm / Anwendungsfalldiagramm (UML)

Anwendungsfall-(engl.: Use-Case)-Diagramme beschreiben wie Software mit Anwendern interagiert. Sie werden eingesetzt, um Anforderungen an eine zu entwickelnde Software zu beschreiben oder Anforderungen dialogisch mit Anwendern zu entwickeln.

Dies geschieht auf einem hohen Abstraktionsniveau aus Blackbox-Sicht des Anwenders, daher werden nur wenige Notationselemente verwendet:

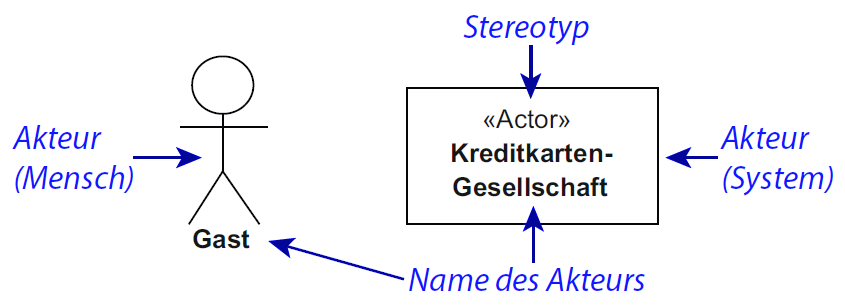
* Anwendungsfälle (Use-Cases)
* Akteure
* Systemgrenzen

Anwendungsfall-Diagramme modellieren *welche* Anwendungsfälle ein System anbietet, nicht *wie* sie realisiert werden. Es werden keine Reihenfolgen oder Abläufe modelliert, lediglich eine Übersicht der vom System angebotenen Anwendungsfälle.

Mit Hilfe von Anwendungsfalldiagrammen kann die Vollständigkeit und Korrektheit des Systemverständnisses von Auftraggeber und Auftragnehmern oder innerhalb von Projektteams abgestimmt werden.

Konkrete Abläufe der einzelnen Anwendungsfälle können später im Top-Down-Verfahren über tabellarische Darstellung oder über andere Verhaltensdiagramme (wie UML-Aktivitätsdiagramme oder UML-Zustandsautomaten) detaillierter beschrieben werden.

Ein **Akteur** modelliert einen Typ oder eine Rolle, die ein externer Benutzer (Strichmännchen, *stickman*) oder ein externes System (Rechteck) während der Interaktion mit einem System wahrnimmt. Sofern es der Erläuterung der jeweiligen Rolle dient können in dem Rechteck auch Symbole zur Veranschaulichung gewählt werden.



Ein **Anwendungsfall** (*use case*) spezifiziert eine abgeschlossene Menge von Aktionen, die von einem System bereitgestellt werden und einen erkennbaren Nutzen für einen oder mehrere Akteure erbringen. Die einzelnen Anwendungsfälle werden mit einem Kurznamen in Ellipsen notiert. Alle Anwendungsfälle, die ein bestimmtes System erfüllen soll, liegen innerhalb dessen Systemgrenzen. Als Alternative wenn auch weniger verbreitete Notation kann auch ein Rechteck mit dem Stereotyp <<Use Case>> sowie die Bezeichnung des Use-Case unter der Ellipse notiert werden:

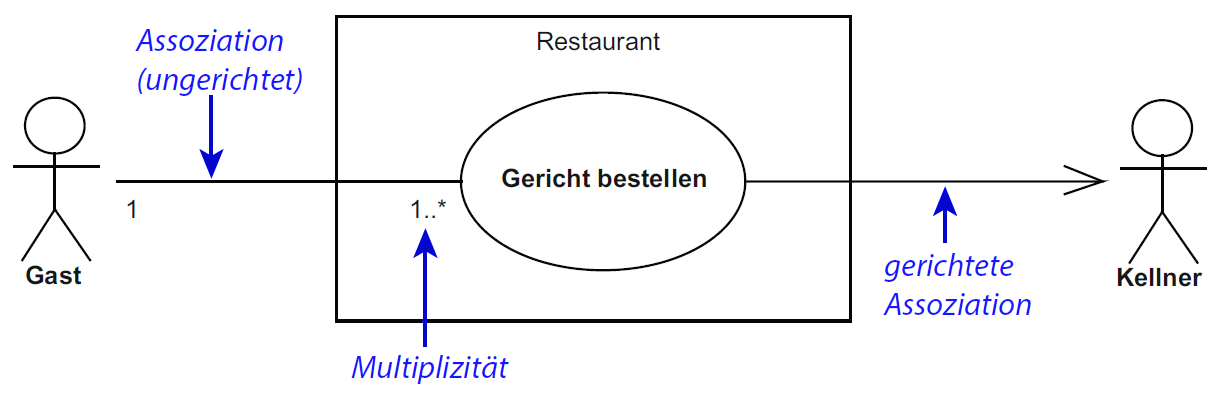


Die ***Systemgrenze*** (*system boundary*) umfasst ein System, das die benötigten Anwendungsfälle bereitstellt und mit dem die Anwender interagieren. Die Systemgrenze legt in der Praxis häufig fest, welche Anwendungsfälle innerhalb des zu realisierenden Projekts umgesetzt werden müssen, und welche nicht Bestandteil des Projekts sind.

Systemgrenzen müssen nicht zwingend angegeben werden, es ist jedoch sehr empfehlenswert, um das Softwaresystem von seinen Akteueren und externen Systemen abzugrenzen.

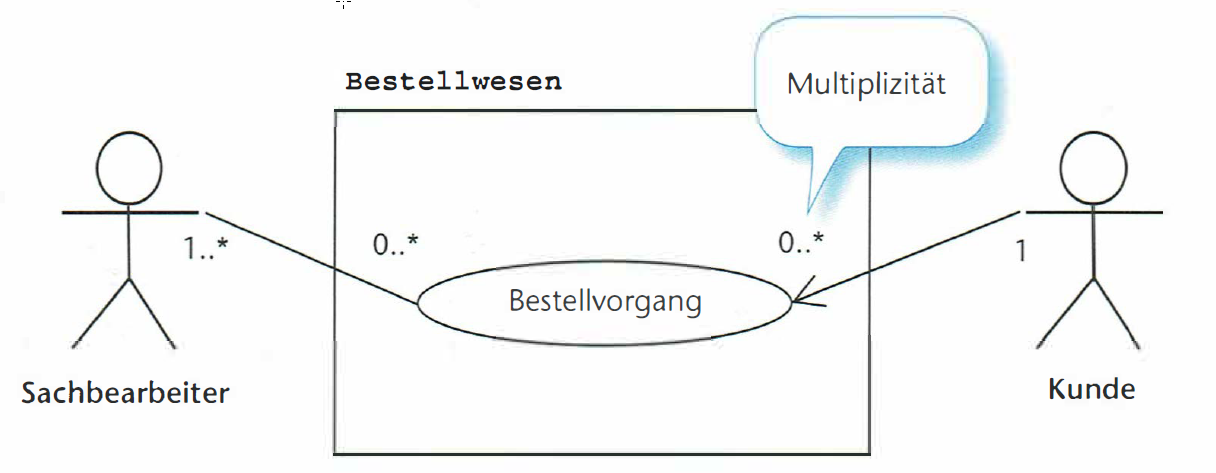
Assoziationen und Multiplizitäten im Anwendungsfalldiagramm

Eine Assoziation (*association*) modelliert in Anwendungsfalldiagrammen eine Beziehung zwischen Akteuren und Anwendungsfällen und legen somit fest, welcher Akteur welche Anwendungsfälle wahrnehmen darf.



Mit gerichtete Assoziationen können erlaubte Kommunikationsrichtungen zwischen Akteur und Anwendungsfall spezifiziert werden: im obigen Beispiel kann der Kellner den Anwendungsfall zwar nicht selbst initiieren, es muss jedoch am Anwendungsfall beteiligt sein und erhält Informationen vom/über den Anwendungsfall (Kommunikationsrichtung des Assoziationspfeils).

Die einfache Assoziation sagt nichts darüber aus, ob ein Akteur mit einem oder mehreren gleichen Anwendungsfällen in Beziehung steht. Ebenso ist es unklar, ob ein Anwendungsfall mit genau einem Akteur oder verschiedenen, gleichartigen Akteuren kommuniziert. Aus diesem Grund ist es möglich, die Beziehung durch sogenannte Multiplizitäten weiter zu qualifizieren. Die Multiplizitäten geben an, wie viele Akteure mit wie vielen Anwendungsfällen in Beziehung stehen. Das folgende Beispiel zeigt die Beziehungen zwischen Sachbearbeitern, Bestellungen und Kunden.



**Die Multiplizitäten sind so zu lesen:**

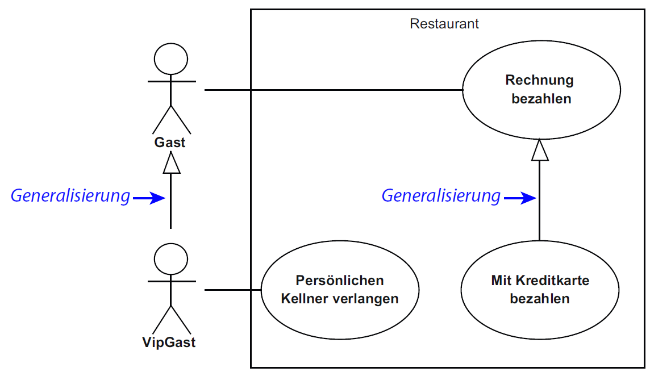
• Ein Sachbearbeiter kann keine bis beliebig viele Bestellvorgänge bearbeiten: (0 .. \*)

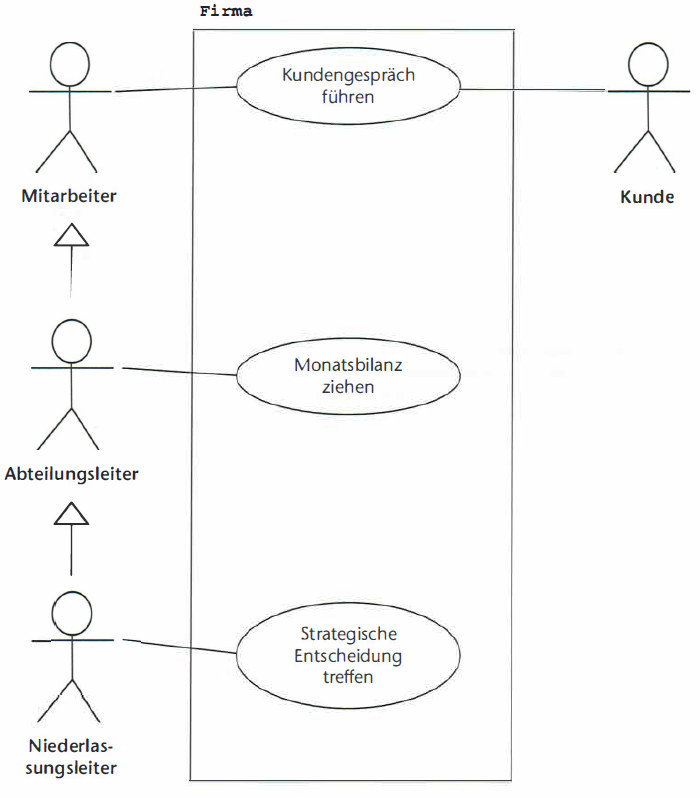
• Ein Bestellvorgang wird von einem oder mehreren Sachbearbeiten bearbeitet: (1 .. \*)

• Ein Kunde kann keine bis beliebig viele Bestellvorgänge initiieren: (0 .. \*)

• Ein Bestellvorgang ist genau einem Kunden zugeordnet: (1)

Vererbung im Use-Case-Diagramm

Das Grundkonzept der Vererbung aus der Objektorientierung findet auch in Anwendungsfalldiagrammen Verwendung. Mit Hilfe von Generalisierungen („erbt“) können Unterfälle zu Anwendungsfällen gebildet werden oder Akteure spezialisiert werden.

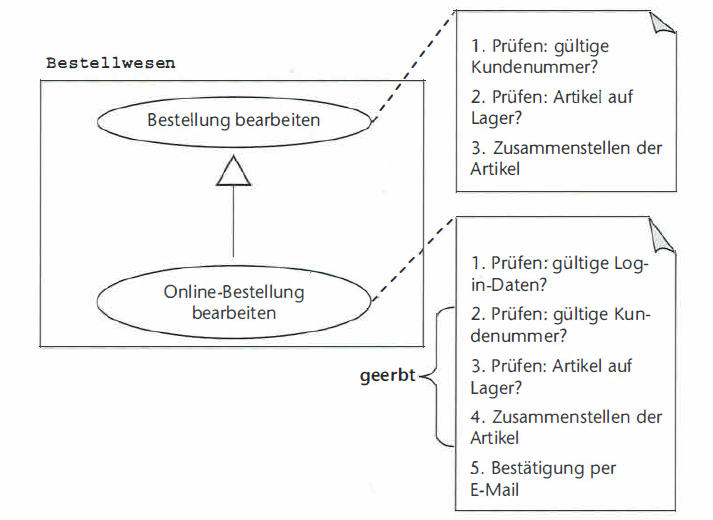


**Vererbung von Akteuren:**

Beim Abteilungsleiter handelt es sich um eine Spezialisierung des Mitarbeiters. Den Anwendungsfall „Kundengespräch führen“ erbt dieser vom Mitarbeiter. In umgekehrter Leserichtung spricht man beim Mitarbeiter von einer Generalisierung eines Abteilungsleiters.

Der Niederlassungsleiter verfügt in nebenstehendem Beispiel auch über den Anwendungsfalls „Kundengespräch führen“.

Die Vererbung wird häufig eingesetzt, um die unterschiedlichen Rollen von Akteuren zu modellieren (z.B. Anwender und Administrator).

**Vererbung von Use-Cases:**

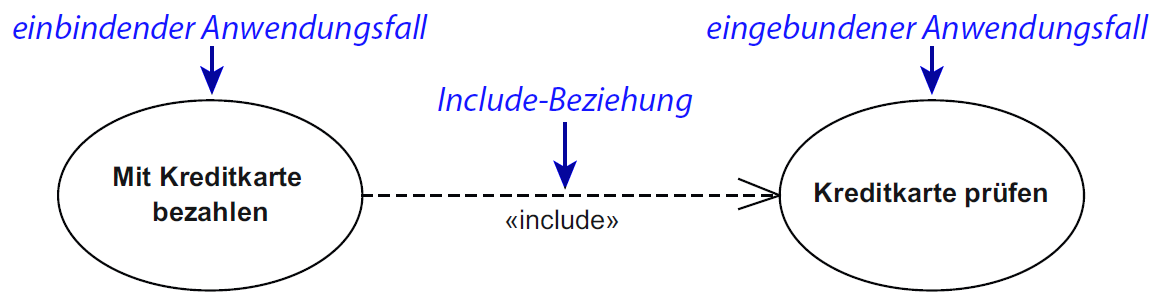
In der Praxis seltener anzutreffen sind Vererbungsstrukturen bei Use-Cases:

der Use-Case „Bestellung bearbeiten“ alle Szenarioschritte und spezialisiert diese.

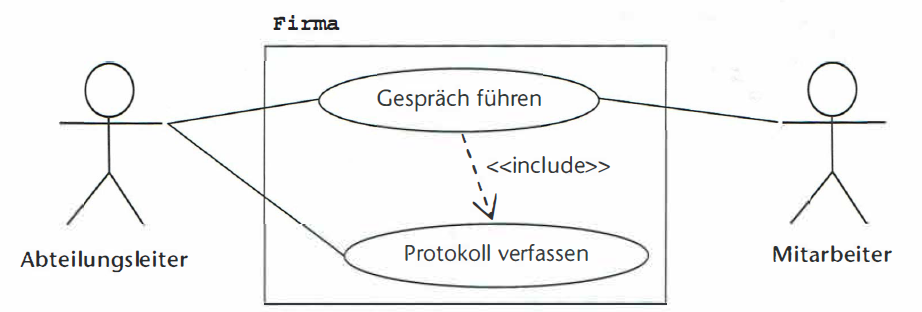
Hinweis: die Darstellung der einzelnen Schritte ist nicht Bestandteil eines Use-Case-Diagramms!

Unbedingte Einbindung von Anwendungsfällen über *<<include>>*

Eine Include-Beziehung (*include relationship*) modelliert die unbedingte Einbindung der Funktionalität eines Anwendungsfalls in einen anderen Anwendungsfall. Dies wird durch den gestrichelten Pfeil und den Stereotyp *<<include>>*.dargestellt.

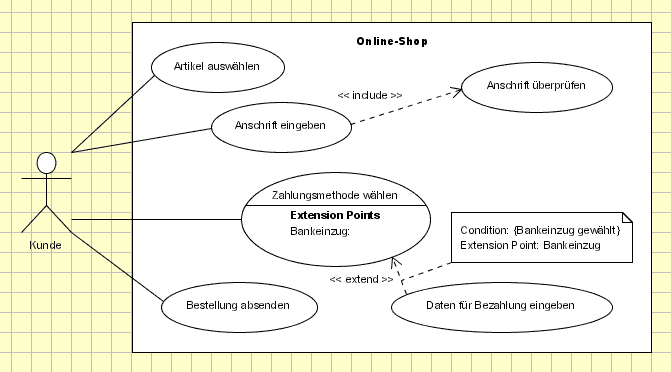


Teilprozesse, die in mehreren Geschäftsfällen auftreten und eigenständige Anwendungsfälle darstellen, können somit für verschiedene Anwendungsfälle wiederverwendet werden, ohne wiederholt beschrieben werden zu müssen.



Der Anwendungsfall „Gespräch führen“ hat zur unbedingten Folge, dass ein Protokoll verfasst wird. Der Anwendungsfall „Protokoll verfassen“ ist in den Anwendungsfall „Gespräch führen“ eingebunden (inkludiert).

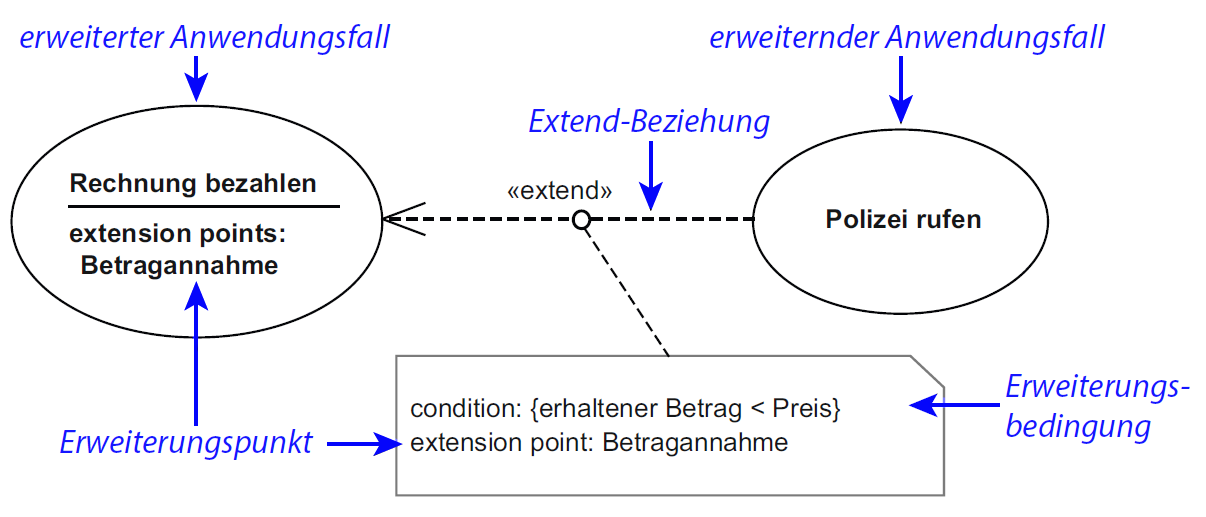
Diese Notation der Wiederverwendung von Anwendungsfällen dient der Übersichtlichkeit und der Lesbarkeit von Use-Case-Diagrammen. Es sollte daher vermieden werden, analog zu Unterfunktionsaufrufen im Programmablaufplan auch solche Funktionen einzubinden, die keinen eigeneständigen Anwendungsfall darstellen.

Ein weiteres Beispiel: Der Anwendungsfall Anschrift eingeben beinhaltet den Anwendungsfall Anschrift überprüfen in der obigen Abbildung.

Quelle / Bildnachweis: <http://www.highscore.de/uml/usecasediagramm.html>

Bedingte Erweiterung von Anwendungsfällen mit *<<extend>>*

Eine extend-Beziehung (*extend relationship*) modelliert die bedingte Einbindung der Funktionalität eines Anwendungsfalls in einen weiteren Anwendungsfall.

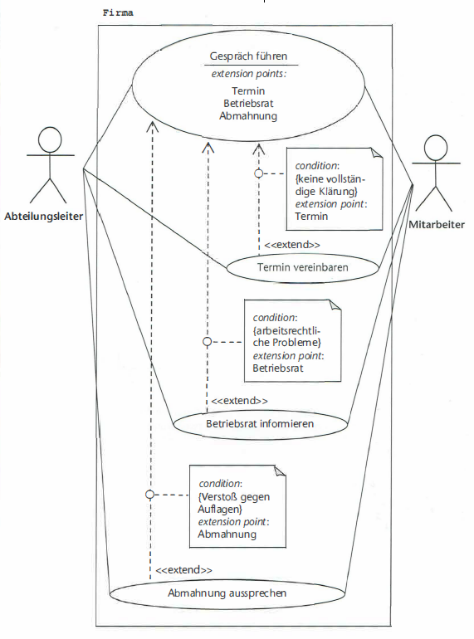


Anwendungsfälle können auch bei zutreffen definierter Bedingungen durch andere Anwendungsfälle erweitert werden (*<<extend>>*).

Im Unterschied zur Einbindung per <<include>> ist jedoch der erweiterte Use-Case vom erweiternden unabhängig und kann auch ohne diesen ausgeführt werden.

Use-Cases können beliebig oft erweitert werden (auch kaskadenhaft), nur zyklische Abhängigkeiten dürfen nicht entstehen.

Beim erweiterten Anwendungsfall wird unterhalb des Trennstrichs eine Liste aller Erweiterungspunkte (*extension points*) angegeben.

Über eine Notiz kann optional wird der erweiterungspunkt benannt sowie die Bedingung (condition) der Erweiterung benannt.

Der Anwendungsfall Zahlungsmethode wählen wird durch den Erweiterungspunkt Bankeinzug um den Anwendungsfall Daten für die Bezahlung eingeben erweitert, sofern die Bedingung Bankeinzug gewählt zutrifft.

Es handelt sich um optionale Verzweigungen im Ablauf eines Geschäftsprozesses.

Auch wenn die erweiterung eine Art if-then-else-Struktur aufweist sollte man nicht der Versuchung erliegen, im Use-Case-Diagramm eine Art Programmablaufplan zu modellieren.

Weitere Stereotypen im Anwendungsfall-Diagramm

weitere Notationen:

* «provides parameters»: ein Anwendungsfall stellt einem weiteren Anwendungsfall Parameter bereit
* «trace»
* «refine»
* «invoke» stammt aus dem OML-Standard (und ist nicht Bestandteil der UML2) und wird genutzt um zu kennzeichnen, dass ein Use-Case einen anderen aufruft.
* «precedes» stammt aus dem OML-Standard (und ist nicht Bestandteil der UML2) beschreibt, dass ein Use-Case vor einem anderen Use-Case ausgeführt werden muss.

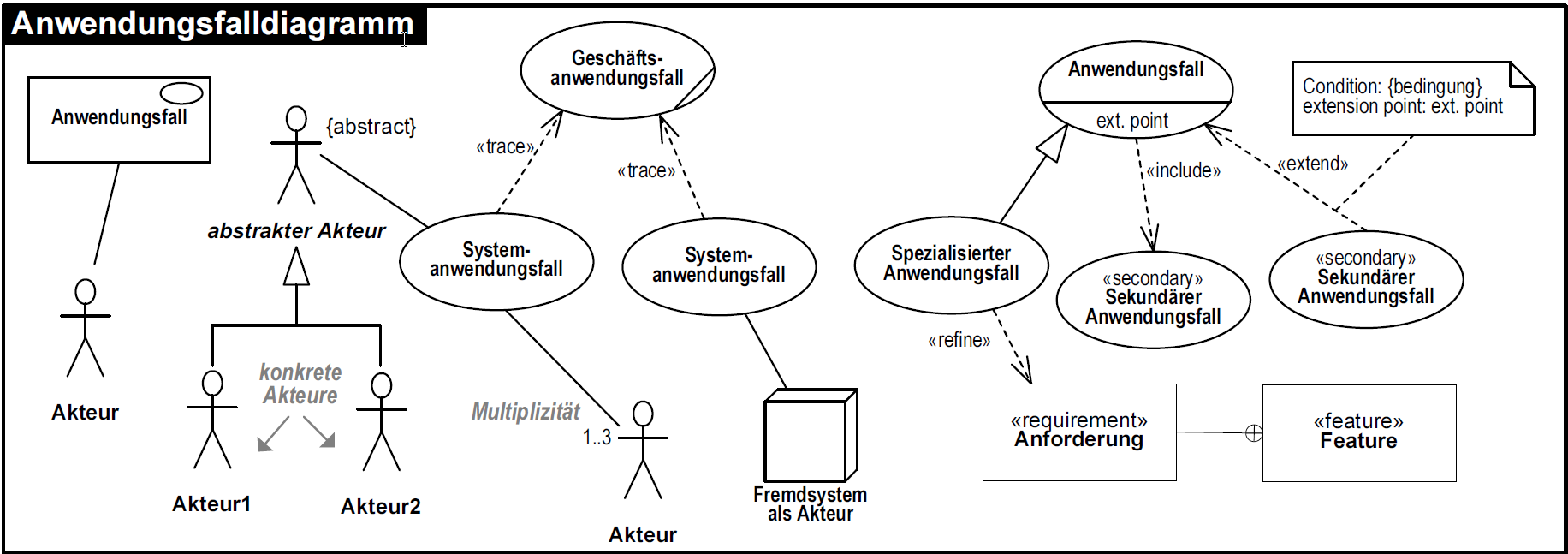
Häufige Fehler

1. Systemgrenze wird häufig vergessen
2. Akteur werden fälschlicherweise innerhalb der Systemgrenze notiert
3. «include» und «extend» werden mit falscher Linie (z.B. durchgezogen und Dreieckspfeil) dargestellt oder die Pfeilrichtung ist verkehrt notiert
4. Anwendungsfälle sind nicht erreichbar (nur gerichtete Assoziationen, «include»-Pfeil verkehrt usw.)
5. Name des Akteurs fehlt
6. Vererbung nicht von Akteur zu Akteur oder von Use-Case zu Use-Case, sondern gemischt.
7. Es werden Abläufe modelliert, keine Anwendungsfälle (z.B. per <<include>> werden Anwendungsfälle verknüpft, die zwar nacheinander folgen, die aber nicht im jeweils anderen enthalten sind).
8. ODER-Beziehungen von Akteuren werden durch zwei Kommunikationsbeziehungen (UND) dargestellt (besser: über abstrakte Struktur mit zwei erbenden Akteuren und einer Kommunikationsbeziehung).

Weiterführende Literatur / Quellen

Die folgenden Texte und Grafiken sind zum großen Teil entnommen:

* Christoph Kecher, Alaxander Salvanos UML 2.5 – Das umfassende Handbuch, Rheinwerk Verlag 2015
* Chris Rupp, Stefan Queins & die SOPHISTen: UML2 glasklar, Hanser Verlag
* UML für IT-Berufe, Europa-Verlag



Quelle: UML 2.5 Notationsübersicht © 2013 by oose Innovative Informatik GmbH Aktuelle Fassung, Info und Download: <http://www.oose.de/uml>