In diesem Kapitel:
- *Iterativer Software Prozess und Use Cases*
- *Risikoanalyse*
- *Systemgrenzen*
- *Aktoren*
- *Use Cases*
- *Szenarien*
- *Ereignsfluss*
- *komplexe Use Cases*

10

UML bei der Erfassung der Nutzeranforderungen im ROP (requirement capture)

10.1. Ausgangslage

Use Cases beschreiben das System aus einer externen Sicht. Use Cases helfen aber auch bei der Erarbeitung von Testfällen und der Benutzeranleitung. Sie helfen Designs zu erstellen und zu validieren. Die Vollständigkeit der Anforderungen (aus Benutzersicht) kann am Besten an Hand der Use Cases erfolgen.

10.1.1. Iterativer Software Prozess und Use Cases

Use Cases sind in unterschiedlichen Prozessmodellen einsetzbar, insbesondere im OO Umfeld und zur iterativen Software Entwicklung.

Use Cases im ROP werden in verschiedenen Phasen entwickelt, bezw. erweitert, vervollständigt und ergänzt.

Die Grundanforderungen werden in der **Startphase** (inception phase) gestartet. In dieser Phase wird auch der Projektumfang bestimmt und typische Geschäftsfälle aufgezeichnet. Am Ende der Phase (Startphase / inception phase) muss der Entscheid gefällt werden, ob weiter gefahren wird, oder ob das Projekt abgebrochen wird.

In der **Entwurfsphase** (elaboration phase) werden die Anforderungen (requirements) vertieft untersucht und eine vertiefte Risikoanalyse durchgeführt. Zudem wird die Implementationsphase im Detail geplant.

In der **Konstruktionsphase** (construction phase) werden in mehreren Iterationen die Komponenten, Teilsysteme und schliesslich das Gesamtsystem gebaut. In jeder Iteration wird eine Analyse, Design und Implementation inklusive Testen durchlaufen. Als Ergebnis erhalten wir neben den Dokumentationen (Benutzermanual, Systemdokumentation, Trainingsunterlagen, ...) auch einen detaillierten Plan betreffend dem Rollout, der Einführung beim Benutzer.

Use Cases treten in verschiedenen Phasen unterschiedlich auf:

- In der Startphase werden generelle Use Cases erfasst. Diese müssen insbesondere die Frage beantworten : was wird im Projekt berücksichtigt, was wird weg gelassen (unter Berücksichtigung von Budget und Zeitplanung)
- In der Entwurfsphase werden die Use Cases verfeinert. Insbesondere erfolgt eine detaillierte Risikoanalyse und ein Erarbeiten der Architektur der Lösung. Use Cases bilden hier auch die Basis für die Planung der Konstruktionsphase
- In der Konstruktionsphase dienen die Use Cases die Basis für den Entwurf und die Entwicklung der Testpläne. Im Rahmen der Iterationen können die Use Cases weiter verfeinert werden und liefern die Basis für die Planung der nächsten Iteration.
- In der Übergangsphase werden mit Hilfe der Use Cases Benutzerdokumentationen und Trainingsunterlagen erstellt.

10.1.2. Ein Projektbeispiel

Wir möchten ein Bestellwesen (mail order) für eine Firma erstellen, die neu ein Bestellwesen im Web einrichten möchte, da eine Kundengruppe typischerweise technologisch geschulte Personen sind und die zweite Kundengruppe eine intensivere Beratung in einem Laden benötigt.

Zwei Mitarbeiter der Firma treffen sich beim Kaffee und überlegen sich, dass ein solches System eigentlich für die Firma einen neuen Vertriebsweg eröffnen würde und sicher äusserst interessant wäre.

Der eine Mitarbeiter, er studiert Informatik, erzählt von Objekten, Klassen und verteilten Systemen. Für den andern ist dies alles unverständlich.

Beide beschliessen, als erstes Material zu sammeln, um die Projektidee konkretisieren zu können. Als Ergebnis dieser Recherchen resultiert:

- Eine (kurze) Projektbeschreibung
- Risikofaktoren
- Annahmen, die wir treffen müssen
- Marktuntersuchungen (ist so ein System sinnvoll, wer macht etwas ähnliches, wie und mit welchem Erfolg, kann man eventuell mit andern zusammen arbeiten, kann man etwas wesentlich besser machen, was machen unsere Kollegen im Ausland?)

10.1.2.1. Projektbeschreibung

Die Projektbeschreibung sollte im Bereich "ein Paragraph" für ein einfaches Projekt, bis zu mehreren Seiten für ein sehr komplexes Projekt reichen.

In der Praxis hat sich gezeigt, dass eine Projektbeschreibung oft am Besten von wenigen Personen entworfen und anschliessend breiter diskutiert wird zum Beispiel im Rahmen eines Workshops.

Eine Projektbeschreibung MUSS auf jeden Fall geschrieben werden, selbst wenn das Projekt "sonnenklar" ist und jeder die Ansicht vertritt, der Aufwand für eine Projektbeschreibung lohne sich nicht.

Fangen wir an mit der Beschreibung:

Problembeschreibung

Wir entwickeln eine Auftragsbearbeitungssoftware für eine mail order Gesellschaft, die wir der Einfachheit halber "TechnoMail" nennen. Die Firma verkauft verschiedene Produkte, welche sie von verschiedenen Herstellern kauft.

Zweimal pro Jahr wird ein Katalog veröffentlicht, der alle gängigen Produkte umfasst und an alle Kunden und Interessenten versendet wird.

Der Firmeninhaber stellt sich schon beim Durchlesen die Frage:

- Reicht ein zweimaliger Versand pro Jahr?

Der Informatiker antwortet:

- Wir starten erst mal klein, und sehen dann wie sich das Geschäft entwickelt

Zusätzliche Anforderungen

Kunden kaufen Produkte, indem sie eine Liste mit den gewünschten Produkten , zusammen mit der (voraus) Bezahlung an TechnoMail senden. TechnoMail setzt die Bestellung zu einer Lieferung um und sendet die Produkte zum Kunden.

Die Auftragsabwicklungs-Software verfolgt den Auftrag ab dem Zeitpunkt, ab dem die Firma den Auftrag erhält, bis zum Zeitpunkt zu dem die Ware abgesandt wird.

Die Firma TechnoMail setzt auf Geschwindigkeit: jeder Auftrag soll mit der schnellst möglichen Versandart zum Kunden geschickt werden.

10.1.2.2. Selbsttestaufgabe

Beurteilen Sie die obige Beschreibung.

- Ist daraus ersichtlich, was die Firma macht beziehungsweise machen will?
- Ist ersichtlich, was die Firma nicht macht?
- Ist ersichtlich, welche besonderen Rahmenbedingungen im Projekt stecken?

10.1.2.3. Vorläufige Risikoanalyse

Nachdem wir wissen, was die Firma erreichen möchte, müssen wir das Projektrisiko abschätzen!

Einige Punkte, die für unser Projekt wichtig sind:

- Wer ist unsere Konkurrenz und was macht sie wie?
- Welche Technologie setzen wir sinnvollerweise ein?
  Web
  Objekttechnologien
  PCs
  LANs
- Welche Markttrends zeichnen sich ab? Entwickeln wir ein System für eine Marktsituation, die nicht mehr lange relevant ist?
- Wie viele Benutzer wird unser System etwa haben?
- Wie ändert sich das Einkaufsverhalten unserer Kunden:
  mehr Home-Shopping?
  mehr Einkauf in Quartierladen?
  andere, konsumentenfreundlichere Öfnnungszeiten?
- Wie viele Transaktionen erwarten wir per Zeiteinheit?
- Wie lange benötigen wir die (speziellen) Funktionalitäten?
- Welche Schnittstellen müssen wir berücksichtigen?

Unsere TechnoMail Firma wird von einem dynamischen Team geleitet. Aus Sicht des Teams sind folgende Trends erkennbar:

Mail Order Marktfaktoren

In den meisten Haushalten arbeiten alle Erwachsenen, zumindest Teilzeit.

Sie haben somit weniger Zeit zum Einkaufen und sind in der Regel bereit etwas mehr für einen Zusatzservice zu bezahlen. Es ist also durchaus denkbar, dass diese Käuferschicht Produkte elektronisch bestellt und dann die Produkte nach Hause geliefert bekommt.

Web Shopping und Home Shopping sind recht beliebt. Verschiedene Konkurrenten sind bereits in diesem Vetriebskanal tätig, lokal und global.

Andere MailOrder Firmen bieten 24-Stunden Auslieferungsservice dh. die bestellt Ware ist innerhalb 24 Stunden beim Kunden.

Andere Konkurrenten bieten 24-Stunden Bestellannahme; aber die Auslieferung erfolgt innerhalb einer, zweier Wochen.

Zusätzlich werden spezielle "Frequent Shopper" Programme angeboten.

Unsere Firma versucht daraus ein Gefahren- und Bedrohungs-Portfolio zu erstellen:
Welche Risiken kommen auf uns zu?
Was kann alles schief gehen?

Eine mögliche Risikoliste:

- Fehlende Benutzer-Akzeptanz
- Zu starke Technologieabhängigkeit; die Technologie ändert zu schnell
- Entwicklungszeit dauert zu lange. Wir kommen damit zu spät auf den Markt!
- Es gibt zu viele Benutzer
- Das Team hat zu wenig Erfahrung
- Die Zeit, die für das Projekt zur Verfügung steht, ist zu knapp.
- Die Firma wächst zu schnell
- Wir sind zu früh damit im Markt
- Unsere Lieferanten können uns die Produkte nicht schnell genug zuliefern

Für unsere Firma TechnoMail definiert das Team folgende Rsikien:

TechnoMail Risikoliste
• Einige der Projektmitarbeiter sind sehr unerfahren
• Wie vermeiden wir Auftragsverluste im System?
• Ist das System auch für technische Laien nutzbar?
• Sind wir auch erfolgreich, wenn wir kein Web Interface haben?
• Werden wir bei Systemstart von einer Bestellflut erschlagen?
• Wie können wir unsere Benutzer am Besten schulen?
• Was passiert, falls die Datenbank abstürzt.

Die obige Liste müssten wir sinnvollerweise noch ergänzen durch Annahmen, die wir getroffen haben: wann, unter welchen Bedingungen sehen wir die Risiken?

10.1.2.4. Zusammenfassung

Problembeschreibung

- Wir entwickeln eine Auftragsbearbeitungssoftware für eine mail order Gesellschaft, die wir der Einfachheit halber "TechnoMail" nennen. Die Firma verkauft verschiedene Produkte, welche sie von verschiedenen Herstellern kauft.
- Zweimal pro Jahr wird ein Katalog veröffentlicht, der alle gängigen Produkte umfasst und an alle Kunden und Interessenten versendet wird.
- Kunden kaufen Produkte, indem sie eine Liste mit den gewünschten Produkten , zusammen mit der (voraus) Bezahlung an TechnoMail senden. TechnoMail setzt die Bestellung zu einer Lieferung um und sendet die Produkte zum Kunden.
- Die Auftragsabwicklungs-Software verfolgt den Auftrag ab dem Zeitpunkt, ab dem die Firma den Auftrag erhält, bis zum Zeitpunkt zu dem die Ware abgesandt wird.
- Die Firma TechnoMail setzt auf Geschwindigkeit: jeder Auftrag soll mit der schnellst möglichen Versandart zum Kunden geschickt werden.
- Kunden haben das Recht, Ware zurück zu senden, zum Teil jedoch nur bei gleichzeitiger Bezahlung einer Bearbeitungsgebühr

Annahmen

- Eine elektronische Schnittstelle, zum Beispiel ein Web Interface, wäre aus Kundensicht sicher empfehlenswert
- Wir werden mit mehreren Logistik-Unternehmen zusammen arbeiten und die Produkte unterschiedlich versichern

Risiko Faktoren

- Einige der Projektmitarbeiter sind sehr unerfahren
- Wie vermeiden wir Auftragsverluste im System?
- Ist das System auch für technische Laien nutzbar?
- Sind wir auch erfolgreich, wenn wir kein Web Interface haben?
- Werden wir bei Systemstart von einer Bestellflut erschlagen?
- Wie können wir unsere Benutzer am Besten schulen?
- Was passiert, falls die Datenbank abstürzt.

Mail Order Marktfaktoren

- In den meisten Haushalten arbeiten alle Erwachsenen, zumindest Teilzeit.
- Sie haben somit weniger Zeit zum Einkaufen und sind in der Regel bereit etwas mehr für einen Zusatzservice zu bezahlen. Es ist also durchaus denkbar, dass diese Käuferschicht Produkte elektronisch bestellt und dann die Produkte nach Hause geliefert bekommt.
- Web Shopping und Home Shopping sind recht beliebt. Verschiedene Konkurrenten sind bereits in diesem Vertriebskanal tätig, lokal und global.
- Andere MailOrder Firmen bieten 24-Stunden Auslieferungsservice dh. die bestellt Ware ist innerhalb 24 Stunden beim Kunden.
- Andere Konkurrenten bieten 24-Stunden Bestellannahme; aber die Auslieferung erfolgt innerhalb einer, zweier Wochen.
- Zusätzlich werden spezielle "Frequent Shopper" Programme angeboten.

10.1.2.5. Weitere Aktivitäten in der Phase "Inception" aus Use Case Sicht

Vollständig	Ergebnisse
✓	Projektbeschreibung
✓	Risiko Analyse
	Use Case Diagramme
	Beschreibung der Aktoren und Use Cases
	Projekt Proposal

10.1.2.6. Festlegen der Systemgrenzen

Wir haben bis jetzt eine sehr abstrakte Beschreibung des zu entwickelnden Systems, aus Benutzersicht.

Als nächstes müssen wir uns klar werden, WAS zu unserem System genau dazu gehört, und was wir besser weglassen!

Beispiel für Systemgrenzen

Was sind Systemgrenzen?

- Unsere TechnoMail Firma muss Aufträge / Bestellungen ausliefern können. Dazu gehört das physische Liefern, aber auch das Verpacken, das Beschriften und die Optimierung der Versandkosten, die optimale Versandart, die Versicherung, Gewichtbestimmung, Zielort, usw.
- Sollen wir in unserem System ein Labeldrucksystem vorsehen oder geschieht dies extern; soll unser System die Logistikkosten optimieren und uns den optimalen Versandweg vorschlagen?

Wir möchten die Systemgrenzen unseres Systems mit Hilfe der Aktoren festlegen

10.1.2.7. Identifikation der Aktoren

Aktoren sind all das, was mit unserem System wechselwirkt, also eine Schnittstelle (im weitesten Sinne) hat.

Jeder Aktor definiert eine spezielle "Rolle". Jede Entität, jedes Objekt ausserhalb unseres Systems können wir als Aktor beschreiben.

Falls Herr Schmutz gleichzeitig Kunde und Mitarbeiter bei TechnoMail ist, dann tritt er als zwei Aktoren auf!

Aktoren sind immer EXTERN!

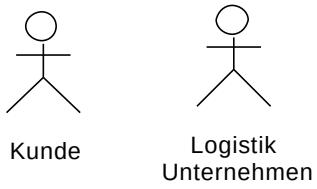
Aktoren lassen sich identifizieren, indem wir uns folgende Fragen stellen:

- Wer nutzt unser System?
- Wer installiert unser System?
- Wer startet unser System?
- Wer wartet unser System?
- Wer stoppt unser System?
- Welche andern Systeme nutzen unser System?
- Wer erhält Informationen von unserem System?
- Wer liefert Informationen an unser System?
- Geschieht irgend etwas automatisch zu einer vordefinierten Zeit?

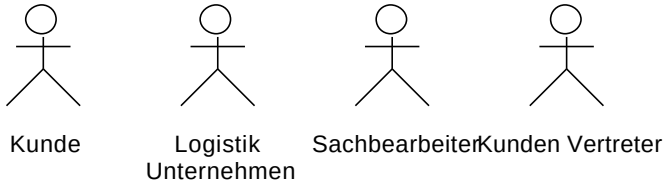
10.1.2.7.1. Selbsttestaufgabe:

Überlegen Sie sich die obigen Fragen für unsere TechnoMail Systemlösung!

Beispiel für Aktoren:



Beispiel für Aktoren einer Auftragsbearbeitung:



10.1.2.8. Identifikation der Use Cases

Als nächstes müssen wir uns fragen, was jeder der Aktoren konkret mit unser Applikation tun möchte / soll / darf.

Ein Use Case wird IMMER von einem Aktor gestartet (zumindest in UML).

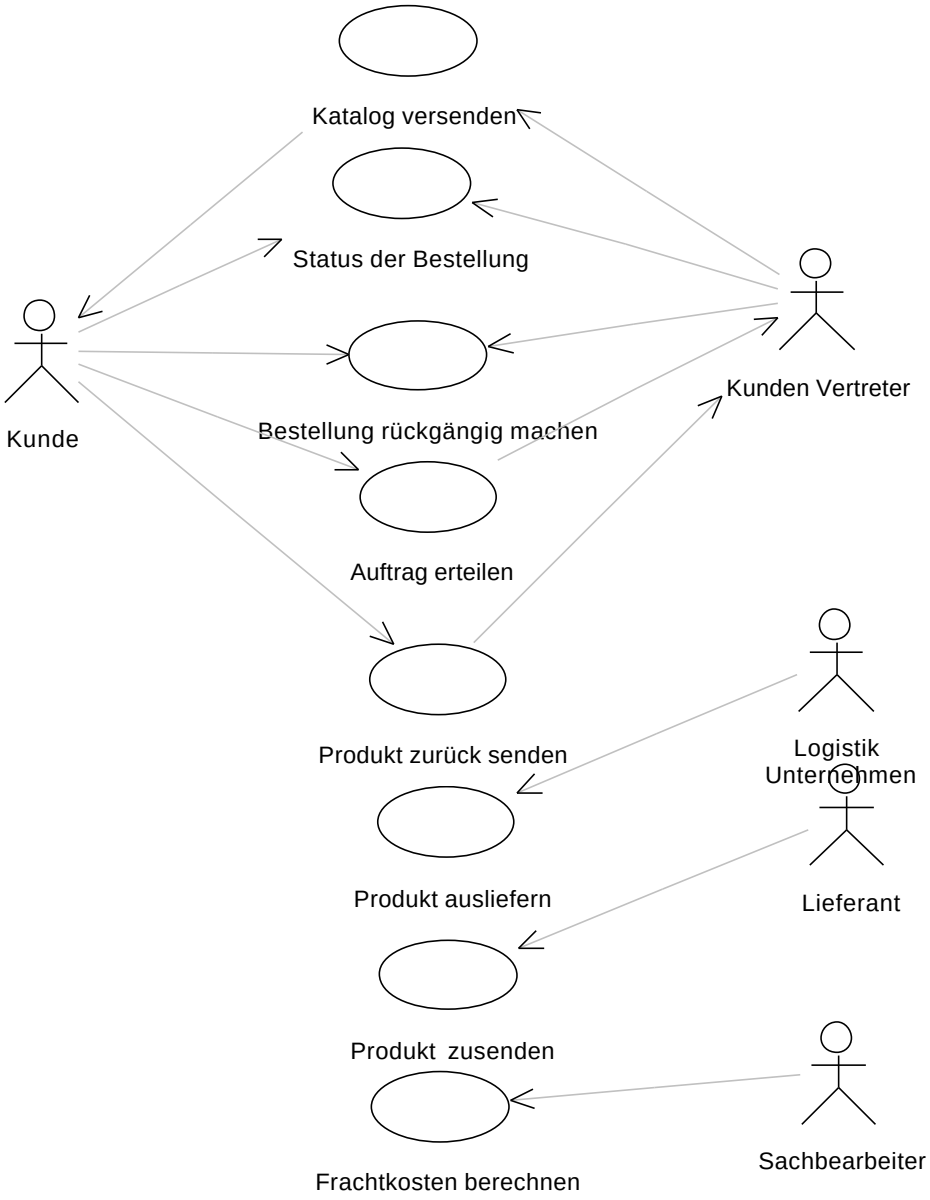
Fragen, die helfen, die Aktoren zu finden:

- Welche Funktionen wird der Aktor wahrnehmen?
- Speichert das System Informationen? Welcher Aktor kreiert, liest, verändert, oder löscht diese Informationen?
- Muss das System einen Aktor über eine Zustandsänderung informieren?
- Gibt es externe Ereignisse, die das System kennen muss? Welche Aktoren informieren das System über solche Ereignisse?

10.1.2.8.1. Selbsttestaufgabe:

Geben Sie die Antworten auf die obigen Fragen für unser TechnoMail Beispiel.

10.1.2.9. Übersicht über das MailOrder System



10.1.2.10. Beschreibung der Aktoren und der Use Cases

Für jeden Aktor brauchen wir eine sinnvolle Kurzbeschreibung.

Auftragsbearbeitung Aktor	*Beschreibung*
Kunde	eine Person, die Produkte bei TechnoMail bestellt
Kundenvertreter	ein Mitarbeiter der TechnoMail, der Bestellungen annimmt
Logistikunternehmen	UPS, FedEX oder eine ähnliche Firma
Sachberabeiter	Mitarbeiter der TechnoMail, der die Ware einpackt, beschriftet, versendet
Lagerverwaltungssystem	Software, mit der das Warenlager der Firma verwaltet wird
Buchhaltungssystem	Buchhaltungssoftware der Firma TechnoMail

Genau so benötigen wir für jeden Use Case eine Kurzbeschreibung

Auftragsbearbeitung Aktor	*Beschreibung*
Vom Kunden	Bestellung, senden Sie bitte einen Katalog, Bestellstatus, Produkt zurück senden, Bestellung annulieren, Beschwerden
Vom Kundenvertreter	Bestellung plazieren, Katalog senden, Status einer Bestellung abfragen, Bestellrückgabe bearbeiten, Bestellung annulieren, Beschwerde aufnehmen
Zum Logistikunternehmen	verpackte Sendung
Vom Sachberabeiter	Mailing Labels drucken, Versandkosten berechnen
ZumLagerverwaltungssystem	Produktinfos übermitteln, Lager nach führen
Vom Lagerverwaltungssystem	nachbestellte Ware wurde geliefert
Zum Buchhaltungssystem	Konto nach führen, Kreditorenbuchhaltung

10.1.2.11. Zeit im Use Case

Die Zeit kann in einem Use Case unterschiedlich behandelt werden:

- Falls eine Aktivität an ein bestimmtes zeitliches Ereignis gebunden ist, dann kann man dieses Ereignis als Aktor definieren
- Die Zeit kann auch explizit im Use Case mit genommen werden dh. der zeitliche Ablauf wird im Use Case modelliert

10.1.2.12. Mögliche Systemgrenzprobleme

Im Laufe der Modellierung des Systems kann die Erkenntnis wachsen, dass ein Teil des Systems doch nicht oder doch noch zum System gehören sollte.

In diesem Falle müssen wir die Systemgrenzen neu überdenken und allfällige Anpassungen vornehmen:

Wer kümmert sich um diese Anforderungen?
Die TechnoMail entscheidet, dass alle Versandpakete versichert werden müssen. Wer kümmert sich um diese Versicherung? Wie muss dies in unserem System berücksichtigt werden? Welche Funktionalität benötigen wir konkret? Benötigen wir diese Funktionalität tatsächlich intern oder reicht es extern?

10.1.2.13. Projektumfang

Wollen wir diese Anforderung berücksichtigen?
Die Firma TechnoMail hat sich im Markt umgesehen und festgestellt, dass eCommerce populär ist bezw. wird. Also braucht TechnoMail eine Web Seite, einen Online Katalog und eine elektronische Web-basierte Auftragsabwicklung!

10.1.2.14. Review

Fragen nach dieser Teilphase die wir uns stellen müssen:

- Sind alle Anforderungen in Use Cases berücksichtigt?
- Haben wir alle Actors beschrieben? Müssen wir sie noch detaillierter beschreiben?
- Sind die Systemgrenzen und die Projektgrenzen klar definiert?
- Sind alle Unsicherheiten festgehalten?

abgeschlossen	Ergebnisse
ok	Projektbeschreibung
ok	Risiko Analyse
ok	Use Case Diagramme
ok	Beschreibung der Aktoren und Use Cases
ok	Projekt Vorschlag

10.2. Szenarien

Nach der Start / Inception Phase, in der wir das Projekt klarer umschrieben haben, wenden wir uns in der Entwurfsphase (elaboration phase) der eigentlichen Projektarbeit zu!

Als erstes müssen wir unsere grundlegenden Anforderungen weiter detaillieren und technische Details erarbeiten und unserer Projektdokumentation hinzu fügen.

10.2.1.1. Der vollständige Use Case

Betrachten wir zuerst ein Beispiel:

Auftrag plazieren
Vorbedingung: der Benutzer hat korrekt in unser System eingelogged
Ereignisfluss:

10.2.1.1.1. Prinzipieller Ablauf

1. Der Use Case startet sobald der Benutzer die Menü Option "bestellen" anwählt
2. Der Kunde gibt Namen und Adresse ein
3. Falls der Kunde nur die Postleitzahl eingibt, dann wird das System die Stadt beistellen
4. Der Kunde gibt den Produktcode ein
5. Das System liefert Produktbeschreibung und Preisinformationen
6. Das System legt sich eine Liste an mit allen Produkten, die der Kunde bestellen möchte
7. Der Kunde gibt die Kreditinformationen ein
8. Der Kunde schickt die Bestellung weg
9. Das System prüft alle Informationen und speichert sie in einem Transaktionsfile. Die Bestellmenge und die Produkte werden an den Innendienst übermittelt, die Finanzinformationen gelangen zur Finanzabteilung
10. Falls die Kreditinformationen bestätigt werden, dann wird die Bestellung als "confirmed" gekennzeichnet und wird bearbeitet
    Damit ist der Use Case abgeschlossen

10.2.1.1.2. Alternativer Ablauf

Falls die Informationen in Schritt 9 nicht korrekt sind, wird der Benutzer um Korrektur gebeten

10.2.1.1.3. Postcondition:

Die Bestellung wird im System gespeichert und als korrekt vermerkt.

10.2.1.1.4. Bemerkungen

Die obige Form der Beschreibung funktioniert nur gut bei relativ einfachen Use Cases. Aber wir können unsere Beschreibung ja weiter verfeinern!

10.2.1.1.5. Pre- und Post-Konditionen

Pre-und Post-Konditionen geben Hinweise auf das was vor oder nach dem Use Case kommt.

Pre-Konditionen müssen erfüllt sein, damit der Use Case überhaupt gestartet werden kann.
Post-Konditionen müssen nach dem Use Case erfüllt sein.

Beispiel für Pre-und Post-Konditionen

Mehrwertsteuer Use Case

Vorbedingung: quartalsweise Abrechnung der Mehrwertsteuer durch TechnoMail
Ereignisfluss:

1. Das System bestimmt, wieviel Mehrwertsteuer fällig ist
2. Das System sendet elektronisch den entsprechenden Betrag an die "..."Verwaltung

Nachbedingung: die Verwaltung verbucht unsere Zahlung und führt unser Konto nach

Vorbedingung:
Unabhängig davon, ob unsere Überweisung funktioniert oder nicht: wir schulden dem Staat die Mehrwertsteuer

10.2.1.1.6. Ereignisfluss

Unter Ereignisfluss verstehen wir den Idealfall der Abwicklung. Alle Sonderfälle beschreiben wir unter Alternativen.
Wichtig ist es, dass wir als erstes eine (positive) Gesamtsicht unseres System bekommen!

Betrachten wir ein weiteres Beispiel mit Verzweigungen:

Finde Bestellung

Ereignisfluss:

1. Der Benutzer gibt Bestellnummer, Kundennummer oder den Kundennamen ein
2. Der Benutzer drückt die "Suche" Taste
3. **Falls der Benutzer eine Bestellnummer eingab**
   a) das System zeigt die Bestellung an
4. **Falls der Benutzer einen Kundennamen oder eine Kundennummer eingibt**
   a) das System liefert eine Liste aller Bestellungen des Kunden
   b) der Benutzer wählt eine der Bestellungen aus
   c) das System zeigt die Bestellung an

Analog zu den Programmiersprachen kann man spezielle "Kontrollstrukturen" einführen:

Beispiel mit Repeat

Bestellen Use Case

Ereignisfluss:

1. Der Use Case startet sobald der Kunde seine Bestellung eingibt
2. Der Kunde tippt seinen Namen und seine Adresse ein
3. Der Kunde gibt den Produktcode / die Artikelnummer ein, vom Produkt das er bestellen möchte
4. **REPEAT:**
   Für jedes eingegebene Produkt
   a) das System stellt die Produktbeschreibung zur Verfügung
   b) das System zeigt den Preis des Artikels und das Einkaufstotal
   END
5. Der Kunde gibt die Kreditinformationen ein
6. Der Kunde schickt die Bestellung weg
7. Das System verifiziert die Informationen und sichert die Bestellung als "in Bearbeitung" und schickt die Kreditinfo an die Buchhaltung
8. Falls die Kreditinformationen korrekt sind, dann wird die Bestellung als "bestätigt" gekennzeichnet, erhält eine Bestellnummer und der Use Case endet

Beispiel mit While

Bestellen Use Case

Ereignisfluss:

1. Der Use Case startet sobald der Kunde seine Bestellung eingibt
2. Der Kunde tippt seinen Namen und seine Adresse ein
3. Der Kunde gibt den Produktcode / die Artikelnummer ein, vom Produkt das er bestellen möchte
4. **WHILE**
   Für jedes eingegebene Produkt
   a) das System stellt die Produktbeschreibung zur Verfügung
   b) das System zeigt den Preis des Artikels und das Einkaufstotal
   END
5. Der Kunde gibt die Kreditinformationen ein
6. Der Kunde schickt die Bestellung weg
7. Das System verifiziert die Informationen und sichert die Bestellung als "in Bearbeitung" und schickt die Kreditinfo an die Buchhaltung
8. Falls die Kreditinformationen korrekt sind, dann wird die Bestellung als "bestätigt" gekennzeichnet, erhält eine Bestellnummer und der Use Case endet

Wir wollen unseren Use Case so ergänzen, dass der Benutzer die Bestellung vor dem Wegschicken noch einmal als Ganzes prüfen kann und erst dann wegschickt:

Beispiel mit Repeat und alternativem Ablauf

Bestellen Use Case

Ereignisfluss:

1. Der Use Case startet sobald der Kunde seine Bestellung eingibt
2. Der Kunde tippt seinen Namen und seine Adresse ein
3. Der Kunde gibt den Produktcode / die Artikelnummer ein, vom Produkt das er bestellen möchte
4. **REPEAT**
   Für jedes eingegebene Produkt
   a) das System stellt die Produktbeschreibung zur Verfügung
   b) das System zeigt den Preis des Artikels und das Einkaufstotal
   END
5. Der Kunde gibt die Kreditinformationen ein
6. Der Kunde schickt die Bestellung weg
7. Das System verifiziert die Informationen und sichert die Bestellung als "in Bearbeitung" und schickt die Kreditinfo an die Buchhaltung

Alternativer Ausführungspfad

vor dem Wegschicken hat der Kunde die Möglichkeit die Bestellung abzubrechen. Die Bestellung wird nicht gesichert und der Use Case endet!

Im Schritt 7: falls irgend eine Information unkorrekt ist, dann wird das System den Benutzer auffordern die Informationen zu korrigieren

10.2.1.2. Komplexe Use Cases

Im Fall komplexer Use Cases stellt man den Use Case oft zweistufig dar.
Man führt dann ein "primäres" und ein "sekundäres" Szenario ein.

10.2.1.2.1. Szenarios

Das Durchlaufen eines Use Cases entlang eines Pfades wird als Szenario bezeichnet.

Beispieszenarios:

- Eine Bestellung trifft ein, die Bezahlung ist korrekt und das Material liegt vor
- Eine Bestellung trifft ein, aber die Finanzinformationen sind nicht korrekt
- Eine Bestellung trifft ein, aber die Lieferadresse fehlt

10.2.1.2.2. Das primäre Szenario

Im primären Szenario beschreiben wir die Abläufe unter der Voraussetzung, dass alles korrekt abläuft!

Im primären Szenarien gibt es also keine Ausnahmezustände, alles läuft perfekt.

Alle Szenarios sind aus Sicht des Benutzers geschrieben, also aus der Sicht der Aktoren. In andern Worten: Szenarios beschreiben eigentlich Use Cases.

Ein Szenario beschreibt:

- Wer fängt einen Use Case an
- Was passiert
- Wie endet er

Oft findet man beim Durchlaufen eines Szenarios neue Teile der Anwendungsarchitektur, neue Aktoren oder neue Use Cases und neue Risiken. Alle neu gefundenen Erkenntnisse müssen sofort in die Dokumentation einfliessen!

Bestelleingangs-Szenario

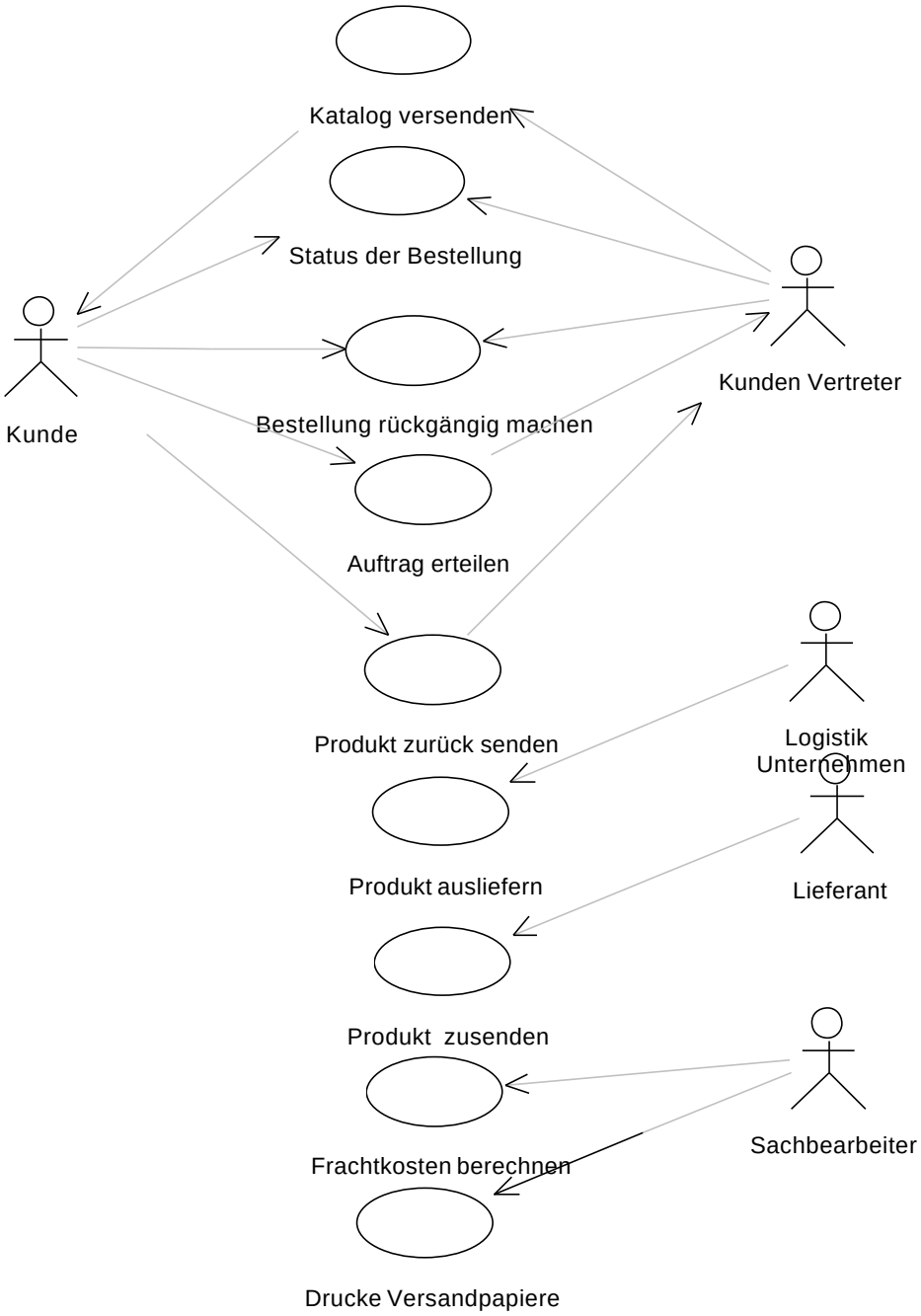
Ereignisfluss

Primär-Szenario

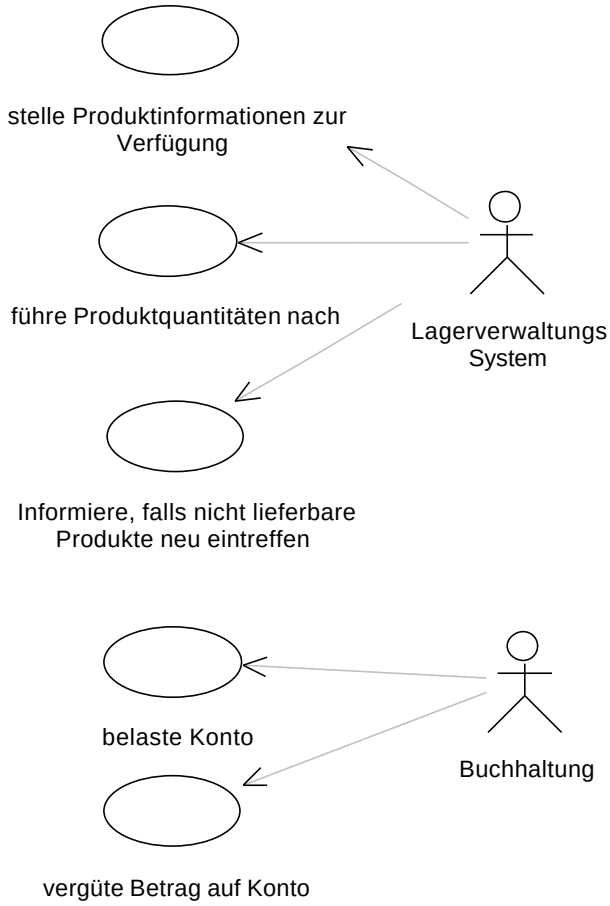
1. Der Use Case startet sobald der Kunde die Option Bestellen auswählt
2. Der Kunde gibt seinen Namen und seine Adresse ein
3. Der Kunde gibt die Bestellnummer für seinen gewünschten Artikel ein
4. Das System schaut die Produktbeschreibung nach und den Preis und zeigt diese an
5. Das System merkt sich den Artikel, berechnet das Gesamttotal und trägt diese Informationen in die "Shopping Card" ein
6. Der Kunde gibt seine Kreditkarteninformationen ein
7. Der Kunde schickt die Bestellung weg
8. Das System verifiziert alle Informationen, sichert die Bestellung als "pendent" und schickt die Kreditkarteninformation an die Buchhaltung
9. Sobald die Kreditinformationen bestätigt worden sind, wird die Bestellung als "bestätigt" markiert, eine Bestellnummer vergeben und an den Kunden zurück gemeldet.

   Damit endet der Use Case "Bestellen / Bestelleingang"

10.2.1.3. Auftragsbearbeitungs-Use Case (modifiziert)



10.2.1.4. Bestellung Use Case (vorläufiges Szenario)



10.2.1.5. Richtlinien für die Korrektheit und Vollständigkeit

Jeder Schritt des Szenarios sollte aus einer kurzen Beschreibung bestehen; kurz vor allem deshalb, damit man nicht zu sehr ins Detail geht und auch weil dadurch die Beschreibung vermutlich korrekt ist.

Falls die Reihenfolge der einzelnen Schritte variabel ist (typischerweise ist dies in Windows Programmen der Fall, da diese ereignisgesteuert sind), dann kann dies einfach am Anfang des Szenarios festgehalten werden.

Detail möchten wir schrittweise hinzu fügen! Deswegen muss die Beschreibung am Anfang auch nicht vollständig sein; sie wird es im Laufe der Zeit!

Allerdings muss die Beschreibung so genau sein, dass alle Schritte, die im Use Case durchlaufen werden, auch nach vollziehbar festgehalten sind.

Viele Use Cases beginnen und enden mit einem Actor, natürlicherweise. Einige Use Cases beginnen mit einem Actor und enden intern.

Gemäss UML darf KEIN Use Case intern beginnen. Der Grund ist einfach: der Use Case soll ja die EXTERNE Sicht repräsentieren!

Szenarios werden aus der Sicht des Actors beschrieben, per Definition! Deswegen sind alle Szenarios von extern nachvollziehbar. Der Benutzer muss also auch in der Lage sein, die Szenarios zu verstehen! Alle Szenarios, die der relevante Benutzer nicht versteht, müssen neu geschrieben werden.

Szenarios dienen der Kommunikation mit den zukünftigen Benutzer!

Eine weitere Kontrolle, die wir durch führen können, besteht darin, dass wir jedes Szenario durch gehen und uns dabei die Frage stellen:
" Was geschieht hier wohl am wahrscheinlichsten?".
Genau das was am wahrscheinlichsten eintritt müssen wir in unsere Beschreibung aufnehmen.

Szenarios brauchen auch nicht von Anfang an gleich perfekt zu sein! Da wir ja iterativ vorgehen, können wir uns zu einem späteren Zeitpunkt immer noch eine Verfeinerung des Szenarios ausarbeiten und alles festhalten, was wir in er Zwischenzeit gelernt haben! Szenarios werden immer realistischer je mehr wir von unserem zu entwickelnden System im Detail kennen und verstehen.

10.2.1.6. Weitere Anforderungen

Beim Aufschreiben der primären Szenarios entdecken wir in der Regel auch Use Cases oder Teile davon, die ein Standardbenutzer nie sehen und EXPLIZIT brauchen wird. Zudem kann es sein, dass wir auf Anforderungen betreffend der Performance oder des Datenvolumens treffen, die wir unbedingt festhalten müssen!

Alle diese zusätzlichen Anforderungen müssen wir festhalten. Falls nötig fügen wir einen Abschnitt "Spezielle Anforderungen" unserer Systembeschreibung hinzu.

Betrachten wir zur Illustration ein Beispiel!

Spezielle Anforderungen

Bestellung eingeben

PreCondition: ein gültiger und berechtigter Benutzer hat ins System eingelogged

Ereignisablauf:
Primäres Szenario
1. Der Use Case startet sobald der Kunde die Option Bestellen auswählt
2. Der Kunde gibt seinen Namen und seine Adresse ein
3. Der Kunde gibt die Bestellnummer für seinen gewünschten Artikel ein
4. Das System schaut die Produktbeschreibung nach und den Preis und zeigt diese an
5. Das System merkt sich den Artikel, berechnet das Gesamttotal und trägt diese Informationen in die "Shopping Card" ein
6. Der Kunde gibt seine Kreditkarteninformationen ein
7. Der Kunde schickt die Bestellung weg
8. Das System verifiziert alle Informationen, sichert die Bestellung als "pendent" und schickt die Kreditkarteninformation an die Buchhaltung
9. Sobald die Kreditinformationen bestätigt worden sind, wird die Bestellung als "bestätigt" markiert, eine Bestellnummer vergeben und an den Kunden zurück gemeldet.

PostCondition: die Bestellung ist abgespeichert und als "bestätigt" gekennzeichnet

Spezielle Anforderungen:
Das System muss Antwortzeiten von weniger als einer Sekunde aufweisen!

10.2.1.7. Präsentation der Ergebnisse

Szenarios werden in der Regel "Freistiel" geschrieben, also ohne fixen, formalen Aufbau. Der Stil der Beschreibung muss dem Zielpublikum angepasst sein:

Mögliche Leser unseres Berichtes sind sicher die Endbenutzer, also Personen, die in der Regel wenig von Compilerbau oder den Details des neusten IP Vorschlages wissen.

Sie wählen Ihren Stil selber. Hier einige Beispiele:

Informeller Text als Beschreibung eines Use Cases

Bestellung rückgängig machen Primärszenario

Sobald der Kundenvertreter (Vertrieb Innendienst) die Rücknahme einer Bestellung erfährt, muss er sie im System suchen und als "gecancelt" kennzeichnen. Dann muss er eine Bestätigung an das Finanz- und Rechnungswesen senden, damit der Kunde sein Geld zurück erhält.

Listenbeschreibung des obigen Use Cases

Bestellung rückgängig machen Primärszenario

1. Der Use Case beginnt sobald der Kundenbetreuer (Verkaufsinnendienst) eine Anforderung für einen Bestellungsabbruch erhält
2. Der Kundenbetreuer gibt die Bestellnummer ein
3. Der Kundenbetreuer startet die Suchfunktion
4. Das System zeigt die Bestellung an
5. Im System wird die Bestellung als "gecancelt" markiert
6. Das Buchhaltungssystem wird informiert, dass der Kunde eine Gutschrift oder eine Überweisung erhält

Der Use Case endet hier

Pseudocode Form des selben Use Cases

Bestellung rückgängig machen Use Case

```
Bestellung = Kundendienst.BestellungRückgängigMachen(Bestellnummer)
        Bestellung.Status = "cancel";
        Buchhaltung.ÜberweiseGutschrift(Kunde, Bestellung.Betrag)
```

Das ist doch toll, dass man einmal nicht stur ein bestimmtes Format einhalten kann. Der Pseudocode zeigt schon in Richtung Klassen und Objekte beziehungsweise deren Methoden und Datenfelder.

10.2.2. Zusammenfassung

Ergebnisse der Entwurfsphase

abgeschlossen	**Ergebnisse**
ok	detaillierte primäre Szenarios
	sekundäre Szenarios
	Aktivitätsdiagramme
	(optional) Benutzerschnittstellen
	Architektur
	Projektplan

10.2.3. Sekundäre Szenarios

Damit wir einen Use Case umfassend und abschliessend beschreiben können, benötigen wir die sekundären Szenarios, also Beschreibungen für alle Ausnahmesituationen, alternativen Pfade und den Szenarios für allfällige Fehlersituationen.

10.2.3.1. Finden der Sekundär-Szenarios

Ein Alternativszenario ist ein Szenario, welches eine andere als die Standardabfolge von einzelnen Use Case Schritten erlaubt. Den typische Fall haben wir im Primärszenario beschrieben.

Ein Ausnahmeszenario ist ein Szenario, welches einen Ausnahmezustand beschreibt. Was kann alles schief gehen und wie wollen wir darauf reagieren?

Ein mögliches Vorgehen zum Finden der Sekundärszenarios sieht wie folgt aus:

- besteht eine Alternative zum Ausführungspfades?
- Kann etwas schief gehen bei der Ausführung des Primärszenarios?
- Kann etwas zu irgend einem Zeitpunkt eintreten?

Jeder Sekundärszenario wird kurz mit Namen und Effekt beschrieben.

Plaziere Bestellung sekundäres Szenario

Bezahlung ist nicht bestätigt
Bestellung ist unvollständig
Bestellung geht verloren
Kunde kann nicht ins System einloggen
Produktcode stimmt mit keinem aktuellen Produktcode überein
Produkt wird nicht mehr geführt
Kreditprüfung schlug fehl
Kunde bestellt mit Check
Kunde bestellt per Post
Kunde telefoniert um zu bestellen

Fangen wir mal so an; den weiteren Ausbau machen wir später!

10.2.3.2. Detaillierung wichtiger Abläufe

Wichtige Szenarios oder Szenarios, welche sehr komplex und schwer zu verstehen sind, müssen schrittweise verfeinert werden.

Sie haben mehrere Möglichkeiten vorzugehen: es bietet sich oft an, dass man die Informationen des sekundären Szenarios gleich ins primäre einträgt.

Natürlich könnten Sie auch versuchen von Anfang an gleich alles mit allen möglichen Alternativen zu beschreiben. Aber das gelingt in der Regel bei komplexen Abläufen nicht!

Wie viele sekundären Szenarios gibt es typischerweise?

Wichtig und wesentlich ist, dass eine Klartext Beschreibung zwar gut und hilfreich ist; aber zu lange wollen wir damit dicht aufhalten:

Wir werden die Zeit sinnvoller im Bereich des Architektur-Designs einsetzen. Es gibt auch noch keine Transpiler, die natürliche Sprache in Java umsetzen....

10.2.3.3. Überarbeitung der Use Case Beschreibung

Oft findet man Gemeinsamkeiten in den verschiedenen Use cases, also bestimmte Muster, die sich in verschiedenen Use Cases wiederholen.

Die UML Notation unterstützt solche Muster, in dem es möglich ist, abstraktere Use Cases zu definieren und daraus konkrete Use Cases analog zu Klassen und Objekten herzuleiten.

Wichtiger als die Nutzung dieser Techniken ist die Klarheit der Beschreibung.

10.2.3.3.1. Extends

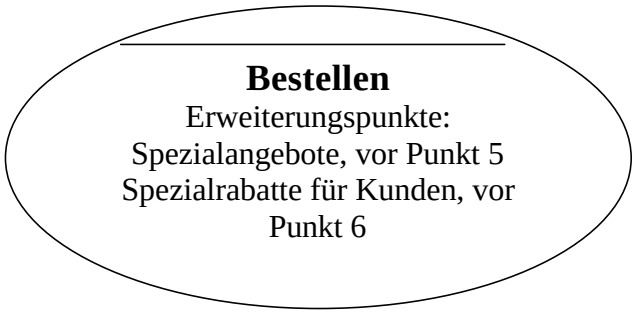
Extends wird dann eingesetzt, wenn in einem Use Case eine Sequenz optional ist, also unter Umständen ausgeführt werden muss, aber auch fehlen kann.

Betrachten wir ein Beispiel:
Unser Firma bietet Spezialrabatte, falls der Kunde ein bestimmtes Geschäftsvolumen erreicht. Jetzt müssen wir unseren Use Case *Bestelleingang* ergänzen, aber eben optional.

Bestelleingang Use Case

1. Der Use Case startet sobald der Kunde die Option Bestellen auswählt
2. Der Kunde gibt seinen Namen und seine Adresse ein
3. Der Kunde gibt die Bestellnummer für seinen gewünschten Artikel ein
4. Das System schaut die Produktbeschreibung nach und den Preis und zeigt diese an
5. Das System merkt sich den Artikel, berechnet das Gesamttotal und trägt diese Informationen in die "Shopping Card" ein
6. Der Kunde gibt seine Kreditkarteninformationen ein
7. Der Kunde schickt die Bestellung weg
8. Das System verifiziert alle Informationen, sichert die Bestellung als "pendent" und schickt die Kreditkarteninformation an die Buchhaltung
9. Sobald die Kreditinformationen bestätigt worden sind, wird die Bestellung als "bestätigt" markiert, eine Bestellnummer vergeben und an den Kunden zurück gemeldet.

Diagramm mit Extension Points



Nun müssen wir noch unsere Use Case Beschreibung entsprechend anpassen.

Erweiterter Use Case Ausverkauf

Falls der gewünschte Artikel ein Ausverkaufsartikel ist, dann beachte die "Ausverkauf" Erweiterung

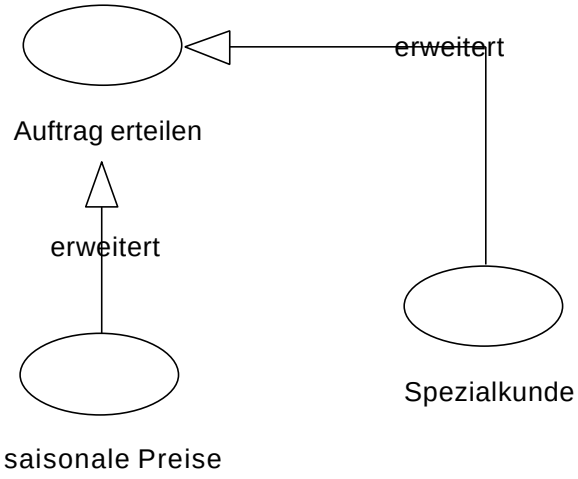
1. Ein Teilsystem bestimmt ob ein Discount gegeben werden kann
2. Das System zeigt den Discount dem Benutzer an
3. Das System berechnet den Discount, indem es den Preis mit Hilfe des Discount neu berechnet
4. Das System subtrahiert den Discount vom regulären Kaufpreis

Erweiterter Use Case für "gute Kunden"

Falls der Kunde in der Spezialkundenliste eingetragen ist, dann verwendet das System die Erweiterung "Spezialkunden"

1. Das System bestimmt den Kunden Discount
2. Das System zeigt dem Kunden den Discount auf den Totalbetrag
3. Das System berechnet einen Discount durch Multiplikation des regulären Preises mit der Ermässigung in Prozent
4. Das System berechnet die Differenz zwischen ursprünglichem Kaufpreis und en Spezialpreis für gute Kunden

Wie stellen wir diesen Sachverhalt mit Hilfe von UML dar?

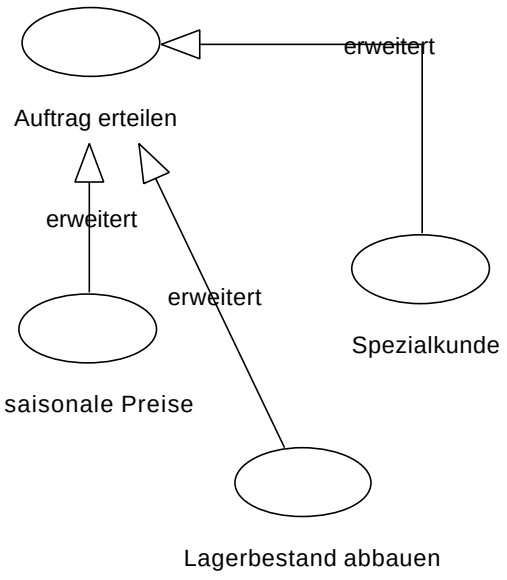


Betrachten wir noch ein weiteres Beispiel: Abverkauf von Ware, deren Lagerbestand den optimalen Lagerbestand überschreitet.

Erweiterung : Use Case Lagerabverkauf

Falls das Produkt in der Abverkaufliste ist und die Ware an Lager für diesen Artikel grösser als der maximal geplante Lagerlevel ist, dann **Verkaufsartikel Erweiterung** 1. Das System bestimmt den Verkaufsdiscount für diesen Artikel 2. Das System zeigt den Discount an 3. Das System berechnet den neuen Preis 4. Das System ermittelt den neuen Verkaufspreis auf Grund des Discount

Jetzt müssen wir unser Use Case Diagramm entsprechend nach führen:



Jetzt wollen wir uns nach dem Fall widmen, dass ein Use Case Debugging Information enthalten kann:

Erweiterter Use Case mit Debugging

If (debugging) then
bei der Sonderartikel Erweiterung
1. Drucke "Prüfung, ob ein Sonderartikel!"

bei der Spezialkunde Erweiterung
1. Drucke "Prüfung, ob ein Kunde = Spezialkunde!"

10.2.3.3.2. Uses

Wie im Falle von Klassen kann man auch bei Use Cases neben der Erweiterung *(extends)* einen Use Case ausbauen, indem man bestehende Use Cases zu einem neuen Use Case kombiniert und ihn noch zusätzlich ausbaut.

Dieser Fall wird mit dem Schlüsselwort "***USES***" beschrieben.

Betrachten wir einige Beispiele:
Unser Bestellvorgang soll zusätzlich eine Suchfunktion erhalten, mit deren Hilfe bei der Eingabe die Aufträge eines bestimmten Kunden gesucht werden können.

Finde Bestellung Use Case

1. Der Kunde gibt seine ID, seine Kundennummer oder seinen Namen ein
2. Der Kunde startet die Suche
3. Falls der Kunde eine gültige Bestellnummer eingab
   a) dann zeigt das System diese Bestellung an
4. Falls der Kunde einen Kundennamen oder Kundennummer eingibt
   a) das System zeigt eine Liste an mit allen Bestellungen dieses Kunden
   b) der Kunde wählt eine der Bestellung aus
   c) das System zeigt diese Bestellung an

Dadurch können wir diese Suche aus dem ursprünglichen Use Case entfernen und einen neuen selbständigen Use Case definieren. Hier ein Beispiel dafür, wie dieser Use Case in einen andern eingebunden werden kann:

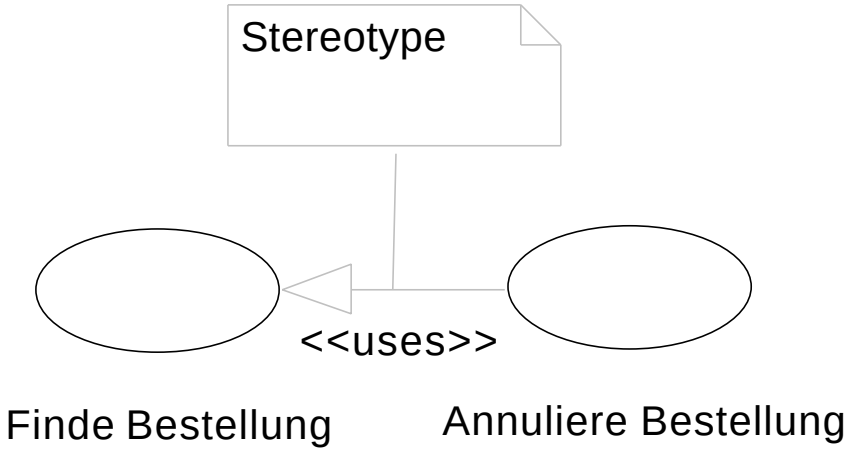
Cancel Bestellung Use Case mit Uses Relation

1. Der Kunde canceld eine Bestellung
2. **USE Finde Bestellung**
3. Falls der Bestellstatus "bestätigt" wird:
   a) merkiere die Bestellung als "annuliert"
   b) informiere die Buchhaltung, so dass der Kunde eine Gutschrift erhält
4. Falls die Bestellung bereits "versandt" wurde:
   a) informiere den Kunden über unsere Rücknahmegarantie

Im obigen Use Case wird die Aktivität im Use Case "Finde Bestellung" ausgeblendet.

Wichtig
Der Use Case "Bestellung annulieren" ist jetzt als solcher nicht mehr vollständig! Er hängt von einem andern Use Case ab.

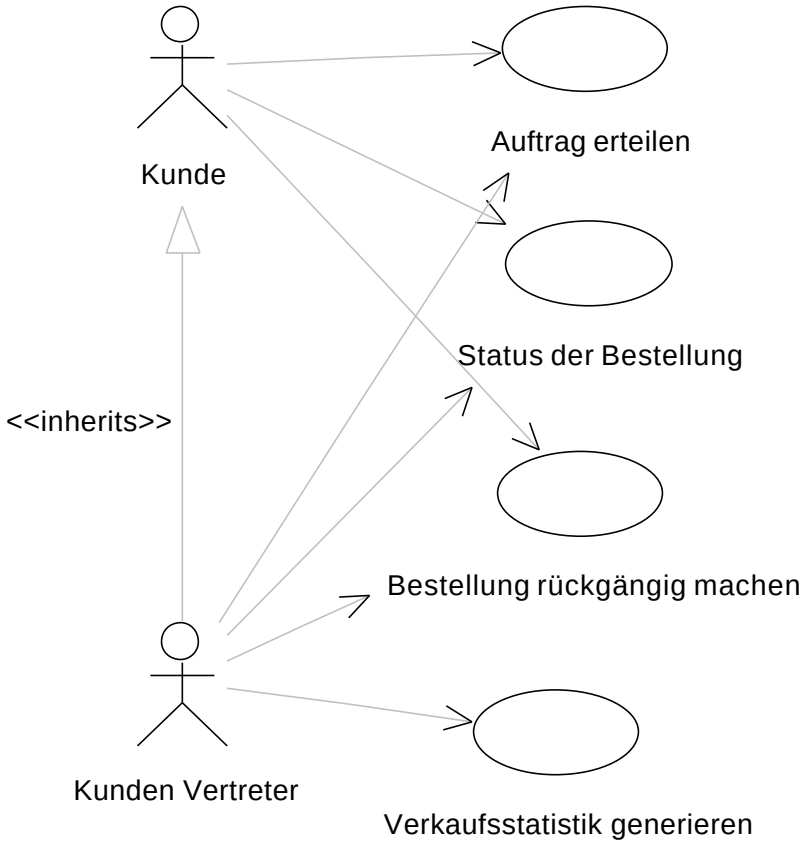
In UML verwendet man für diesen Sachverhalt folgende Konstrukt:



10.2.3.3.3. Vererbung / Inheritance

Analog zu "*uses"* wird die Vererbung zwischen Aktoren dargestellt. Bei Use Cases gibt es KEINE Vererbung im Sinne der Objektorientierung. Die Symbole sind jedoch nicht unterschiedlich und in Rose stellt sich alles gleich dar.

Ausweg : Stereotype angeben (siehe Diagramm).



In diesem Beispiel haben der Kunde und der Innendienstmitarbeiter der MailOrder Firma viele Use Cases gemeinsam. Es bietet sich daher an, den Innnendienstmitarbeiter und den Kunden zu verknüpfen.

10.2.3.3.4. Interfaces / Schnittstellen

Interfaces kann man für Use Cases, Aktors oder beide definieren. Ein Interface beschreibt, was eine bestimmte Grösse tun soll. Das Interface ist nicht Teil des Aktors oder des Use Cases. Es gibt vielmehr Hinweise darauf, wie ein Aktor und ein Use Case zusammen arbeiten können.

Betrachten wir ein Beispiel:

Interface Beispiel für den Bestellungs Use Case

Gibt Bestellung ein() lies Produkt Beschreibung und Preis(ProduktID) an Zeige Beschreibung, Preis Addiere Preis zu Total(Preis) return Total Submit Bestellung() return BestellID

Interface Beispiel für Kunden Aktor Use Case

Eingabe Name und Adresse() return Name, Adresse **Eingabe ProduktCode() return produktCode()** **Eingabe KreditInfo() return Kartennummer, ExpirationDate**

Wie sieht ein Interface generell aus:

- Ein Interface hat einen Namen
- Ein Interface besitzt eine Signatur
- Die Signatur zeigt , welche Daten verwendet werden, welche Daten zurück gegeben werden nach Abschluss der Operation
- Die Operation beschreibt, was diese Entität erledigen soll

Interfaces lassen sich nicht eindeutig definieren:

- Wir können Aktivitäten in Use Cases hinein schieben
- Wir können Aktivitäten aber genauso in Schnittstellen / Interfaces kapseln

Noch zwei Beispiele für Schnittstellen:

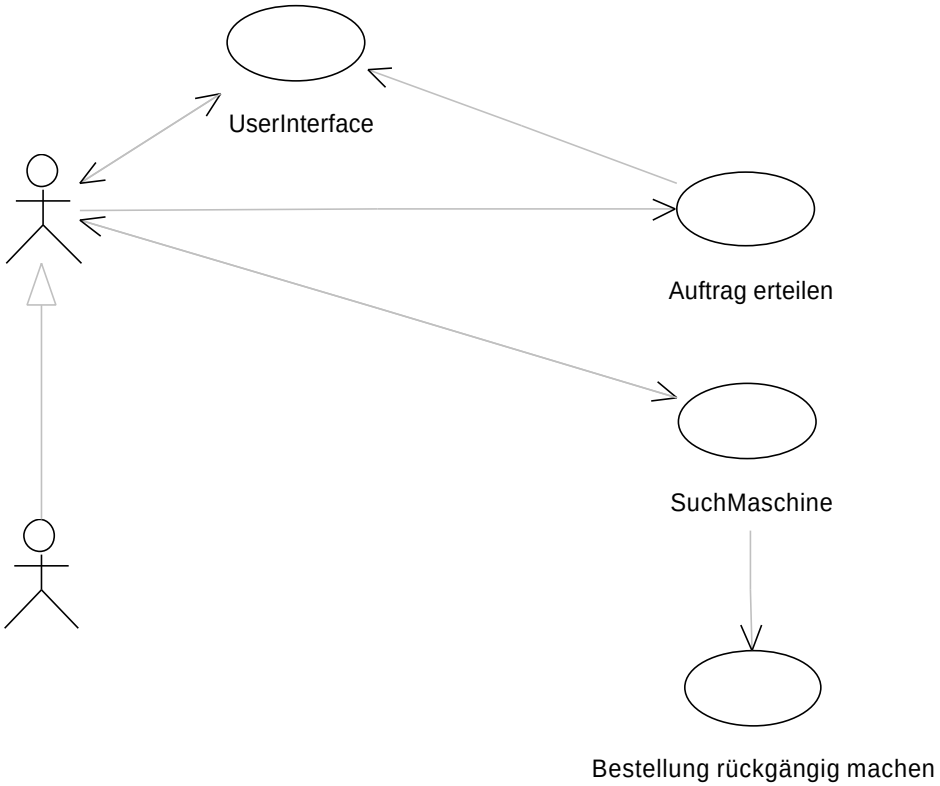
Bestellsuche Schnittstelle

Eingabe BestellungsID()	return BestellungsID
Eingabe KundenName()	return KundenName
Eingabe KundenID()	return KundenID

Ausbau einer Schnittstelle

Eingabe Name und Adresse()	return Name, Adresse
Eingabe ProduktCode()	return ProduktCode
Eingabe KreditkartenInfo	return Kartennummer, Expiration Datum
Eingabe BestellID()	return BestellID
Eingabe KundenName()	return KundenName
Eingabe KundenID	return KundenID

In der Praxis ist es oft besser kleine Use Cases zu definieren und mit Hilfe von Schnittstellen zu verbinden, immer in der Hoffnung dadurch Teile des Systems wieder verwendbar machen zu können.



Zur Notation:
Teilweise wird, nicht in Übereinstimmung mit UML1.0 unterschieden:

- Verbindungslinien sind entweder gestrichelt, falls sie die Kommunikation mit einem Interface betreffen
- Verbindungen anderer Art sind ausgezogen

10.3. Anhang : Benutzt (USES) und Erweitert (EXTENDS)

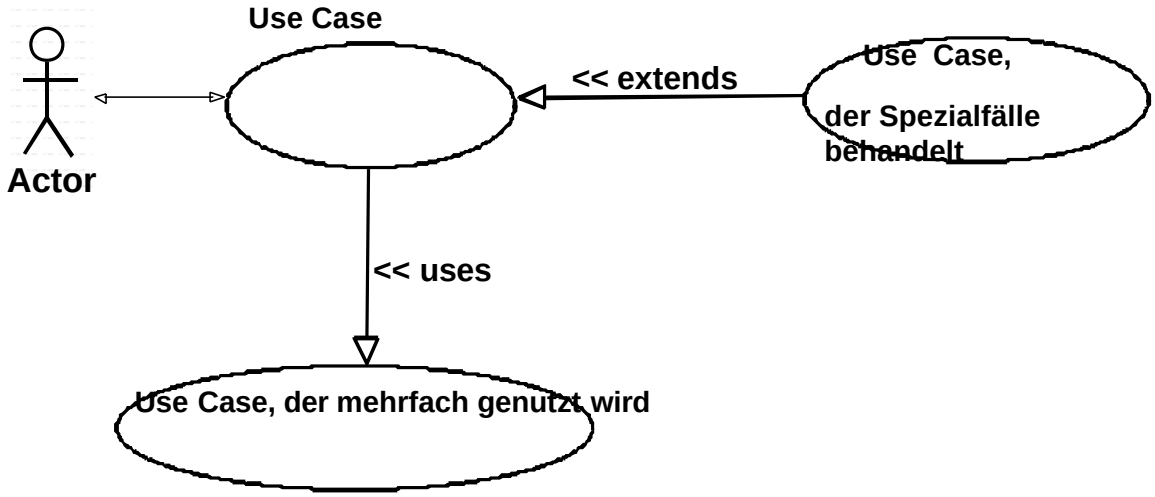
Man benutzt *Erweitert* bei einem Anwendungsfall, falls die Anwendungsfälle ähnlich sind, aber der eine doch noch etwas mehr bewerkstelligt.

1. Man beschreibt zuerst den einfachen, normalen Anwendungsfall
2. Für jeden Schritt in ihm fragt man sich "Was kann hier schiefgehen?" und "Wie könnte dies anders funktionieren?"
3. Man skizziert alle Variationen als Erweiterungen des Anwendungsfalles. Üblicherweise gibt es sehr viele Erweiterungen. Eine Auflistung schafft ein vertieftes Verständnis und hilft ganz beträchtlich bei der Programmierung!

Die *Benutzt* Beziehung wird dann verwendet, wenn ein Verhaltensanteil in verschiedenen Anwendungsfällen gleich sind. Man erspart sich das Kopieren der Beschreibung dieses Verhaltens.

Faustregel:

- Man verwendet "ERWEITERT", wenn man eine Variation eines normalen Verhaltens beschreibt
- Man verwendet "BENUTZT", wenn man sich in zwei oder mehreren Anwendungsfällen wiederholt und diese Wiederholung vermeiden möchte.



10.4. Anhang : Rational Rose Templates

Im Verzeichnis :
Rational Unified Process 5.1
wordtmpl
templates

finden Sie Word Templates für die unterschiedlichen Phasen und Aktivitäten im ROP Modell.