

## UML-Verständnis

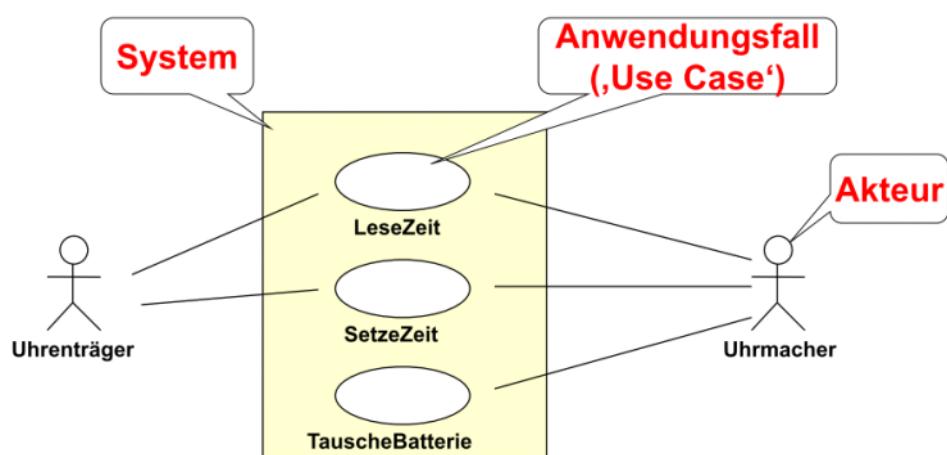
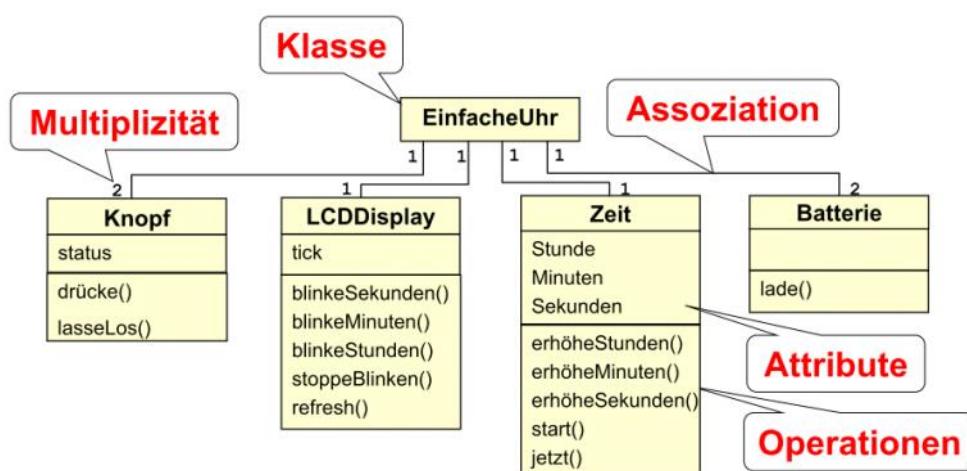
Dokumentieren Sie die Simulation einer Uhr mit Hilfe von UML-Diagrammen

- Strukturdigramm

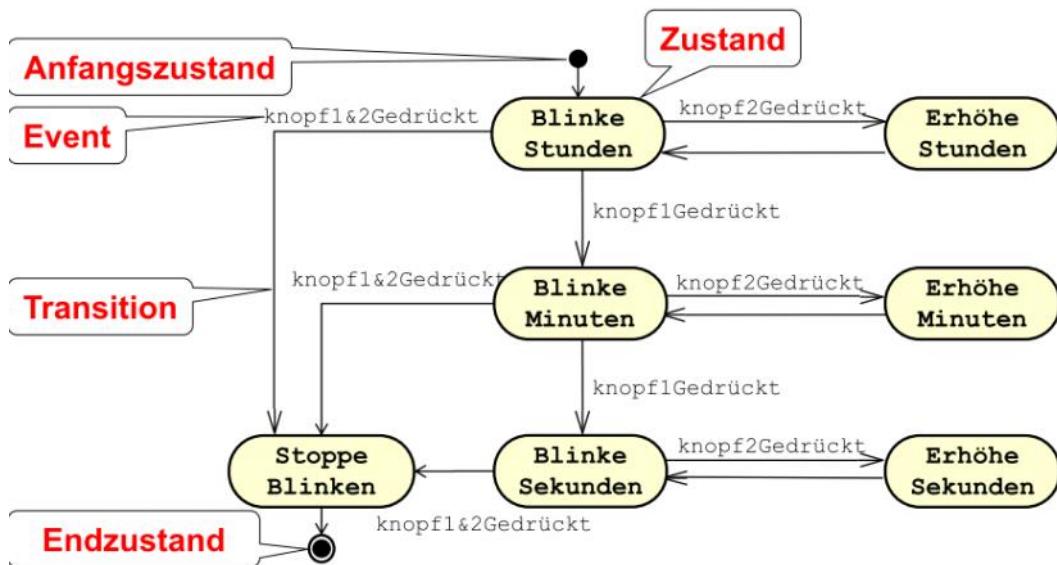
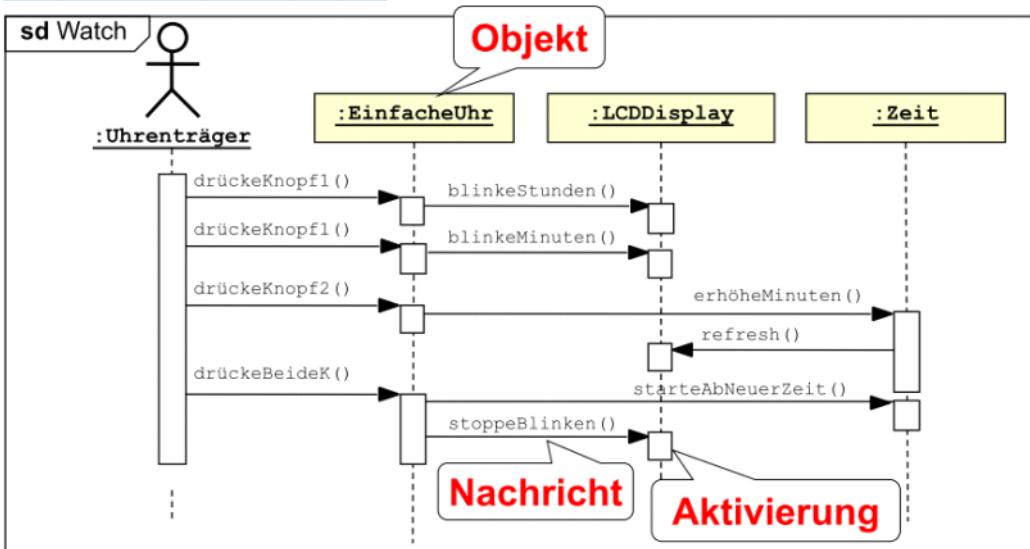


- Dynamisches Diagramm

# Lösung



# UML-Kurzübersicht: Sequenz-Diagramme



# Doku zu UseCase

Dienstag, 8. September 2020 21:17

## 1 UML-Grundlagen

### Inhaltsverzeichnis

Einführung in UML.....	3
Überblick.....	3
UML - historisch gesehen .....	5
UML-Diagramme im Überblick.....	6
UML - Tabellarischer Überblick .....	8
UML-Erfüllungsebenen.....	10
Verständnisfragen .....	10
UseCase-Diagramm .....	11
Modellelemente .....	13
Use-Case = System + Anwendungsfall + Akteur .....	13
System .....	14
Akteur .....	14
Beziehungen .....	15
Fragen zu UML.....	22
Lösungen.....	24
Sequenzdiagramm.....	26
Modellelemente .....	27
Klasse/Objekt/Lebenslinie .....	27
Nachricht .....	28
Kombinierte Fragmente.....	29
Use Case und Sequenzdiagramme .....	32
Aufgaben .....	33
Lösung zu Sequenzdiagrammen .....	38
Zustandsdiagramm .....	42
Modellelemente .....	43
Zustand .....	43
Transition.....	44
Aufgaben .....	45
Lösung .....	47
Aktivitätsdiagramm .....	49
Modellelemente .....	50
Aktivität .....	50
Objektknoten.....	50
Kontrollelemente.....	51

8. September 2020

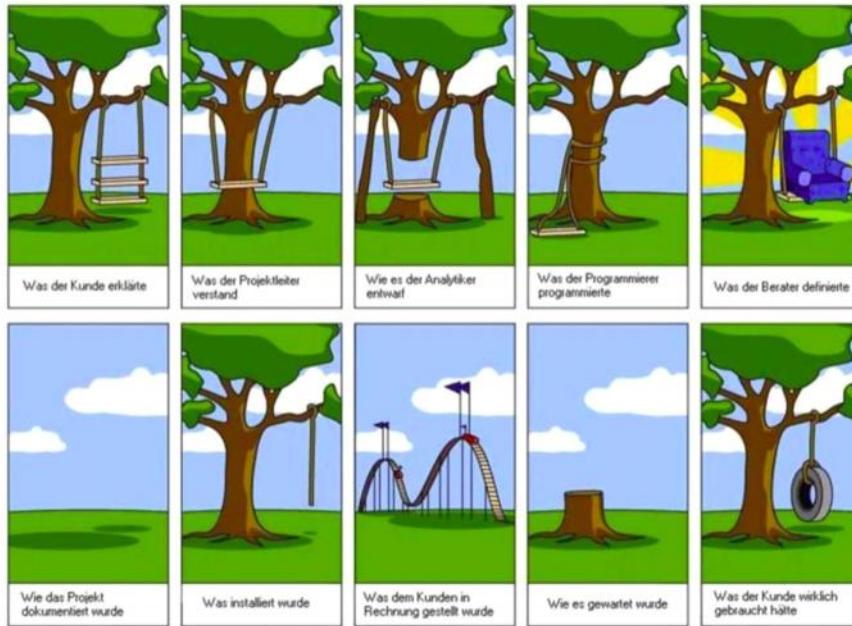
Aktivitätsbereiche (Swimlanes) .....	54
Aktion.....	54
Vor- und Nachbedingung.....	55
Signale und Ereignisse .....	55
Fragen zu Aktivitätsdiagramm.....	56
Lösungen.....	61
Verteilungsdiagramm .....	65
Modellelemente .....	66
Aufgaben.....	67
Lösung .....	68
Klassendiagramm .....	68
Analyse-Klassendiagramm.....	68
Entwurfs-Klassendiagramm .....	69
Implementierungs-Klassendiagramm .....	69
Elemente des Klassendiagramms.....	70
Notationselemente im Einzelnen.....	71
Lösungen.....	103
Index .....	<b>Fehler! Textmarke nicht definiert.</b>



8. September 2020

## Einführung in UML

### Überblick



### Arbeitsauftrag

Betrachten Sie das obenstehende Bild und diskutieren Sie in der Gruppe über die Aussage des Bildes. Halten Sie Ihre Ergebnisse stichpunktartig fest!

8. September 2020

Die Unified Modeling Language (vereinheitlichte Modellierungssprache), kurz UML, ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software-Teilen und anderen Systemen. Sie wird von der Object Management Group (OMG) entwickelt und ist sowohl von ihr als auch von der ISO (ISO/IEC 19505 für Version 2.4.1 standardisiert). Im Sinne einer Sprache definiert UML dabei Bezeichner für die meisten bei einer Modellierung wichtigen Begriffe und legt mögliche Beziehungen zwischen diesen Begriffen fest. UML definiert weiter grafische Notationen für diese Begriffe und für Modelle statischer Strukturen und dynamischer Abläufe, die man mit diesen Begriffen formulieren kann. (Quelle: Wikipedia)

Sie ist damit für die objektorientierte Programmierung das, was Struktogramme / Programmablaufpläne und ähnliche Konzepte für die strukturierte Programmierung war,

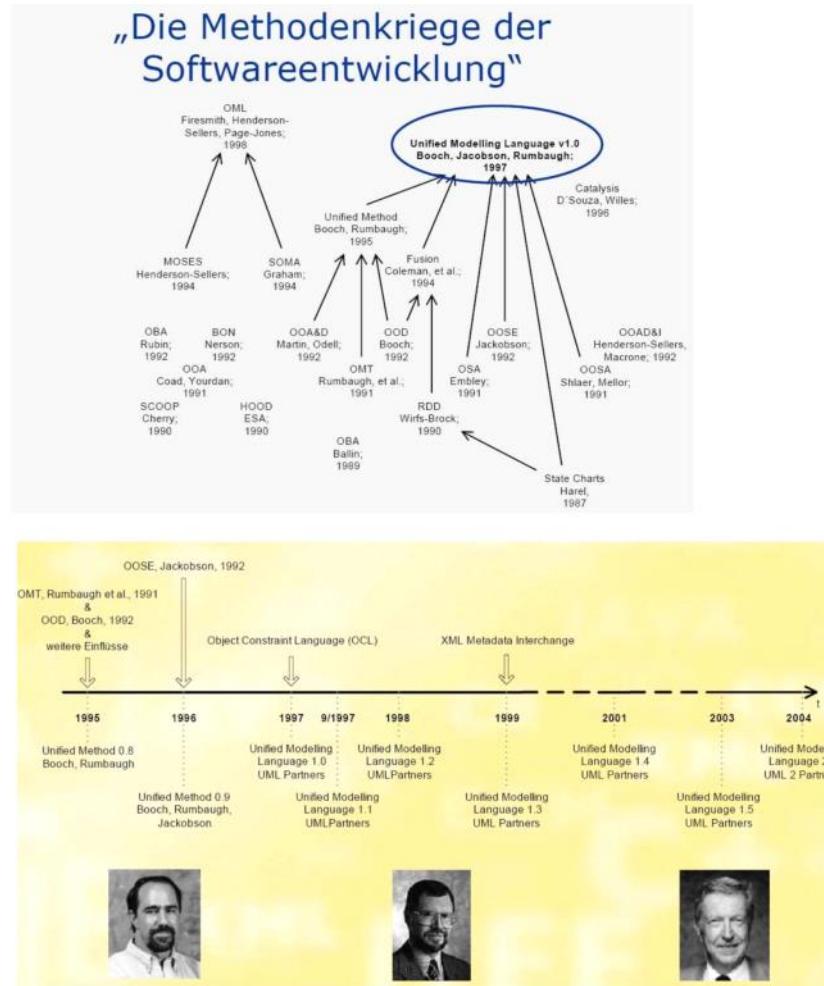
- Sie ist die verbreiteste Notation, um Softwaresysteme zu analysieren und zu entwerfen
- Sie dient zur Spezifizierung und Visualisierung komplexer Softwaresysteme unabhängig vom Fach- und Realisierungsbereich.
- Sie liefert die Notationselemente gleichermaßen für die statischen und dynamischen Modelle von Analyse, Design und Architektur und unterstützt objektorientierte Vorgehensweisen.

Die UML ist

- nicht perfekt
- nicht vollständig
- keine rein formale Sprache
- nicht spezialisiert auf ein Anwendungsgebiet
- kein vollständiger Ersatz für Textbeschreibung
- keine Methode oder ein Vorgehensmodell

## UML - historisch gesehen

Im Zuge der Entwicklung objektorientierter Programmiersprachen und deren Möglichkeiten, kam es besonders zum Anfang zu einer Vielzahl unterschiedlicher Notationsmodelle. Um diese zu vereinheitlichen, setzten sich zu Beginn der 90er Jahre mehrere Vertreter unter Führung der Firma RATIONAL zusammen, um eine gemeinsame Sichtweise (Unified) zu schaffen. Diese Arbeiten mündeten dann 1997 in die Verabschiedung der UML-Version 1.0.



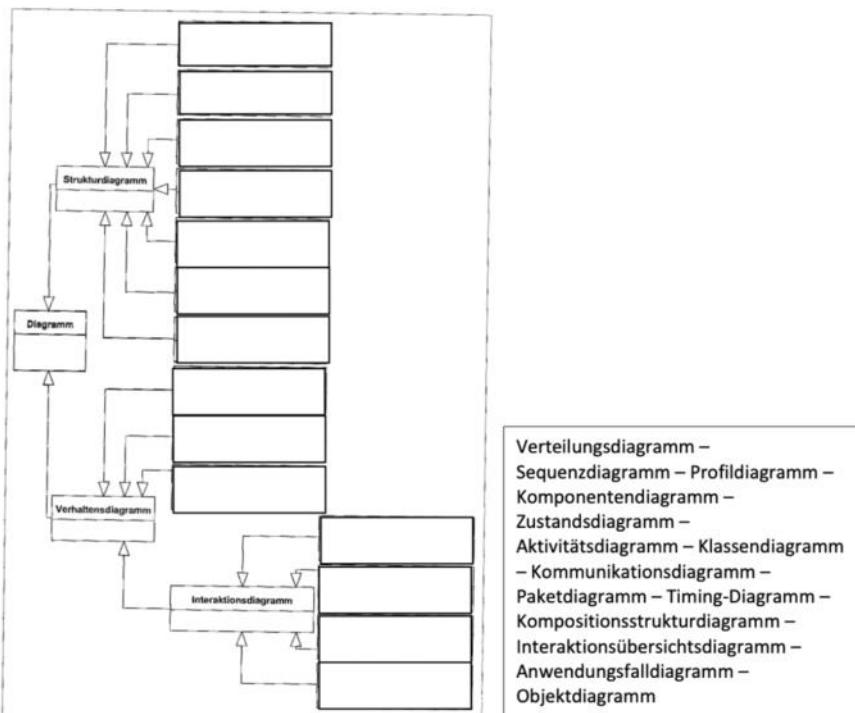
## UML-Diagramme im Überblick

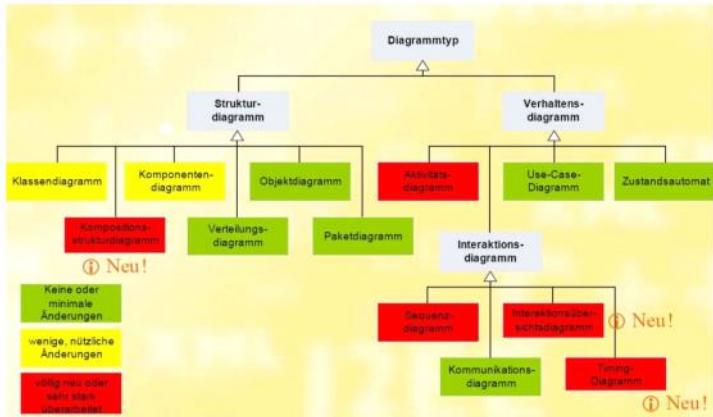
Die bisher 7 Diagramme der UML 1.x - Notation wurden in der Version 2.x auf 13 Diagramme erweitert. Darüber hinaus wurden manche (alte) Diagramme mit neuen Inhalten versehen.



### Arbeitsauftrag

Die Unified Modeling Language (UML) bietet verschiedene Diagramme zur Spezifikation, Konstruktion und Dokumentation von Software(-teilen). Die untenstehende Übersicht zeigt die verschiedenen Diagrammarten. Sortieren Sie die aufgeführten Diagramme ihren übergeordneten Diagrammarten zu.



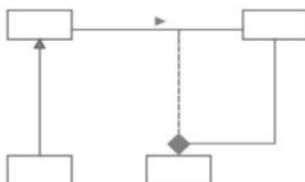


8. September 2020

## UML - Tabellarischer Überblick

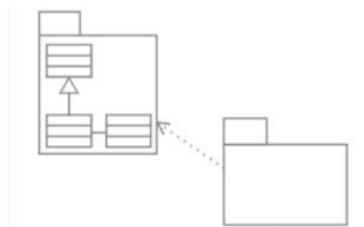
Folgende Darstellung gibt einen Überblick über die Schwerpunkte der einzelnen Diagrammarten.

### Klassendiagramm



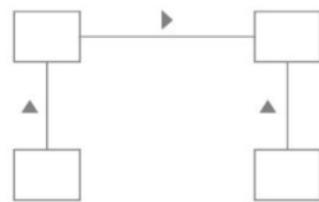
Es beschreibt die Klassen des System und wie sie miteinander in Beziehung stehen. Es ist das wichtigste Diagramm in der UML.

### Paketdiagramm



Es bündelt die Klassen zu überschaubaren Paketen und beschreibt die Beziehungen zwischen den Paketen.

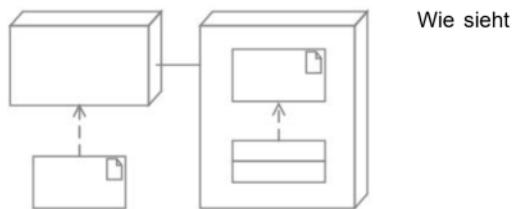
### Objektdiagramm



- Welche innere Struktur besitzt mein System zu einem bestimmten Zeitpunkt zur Laufzeit (Klassendiagrammschnappschuss)
- Zeigt Objekte und Attributbelegungen zu einem bestimmten Zeitpunkt
- Verwendung beispielhaft zur Veranschaulichung

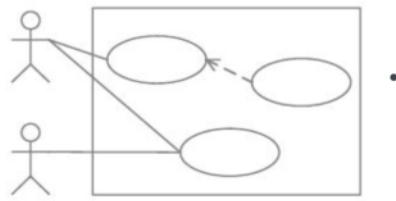


**Verteilungsdiagramm**



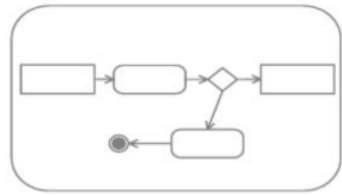
Wie sieht

**UseCase-Diagramm**



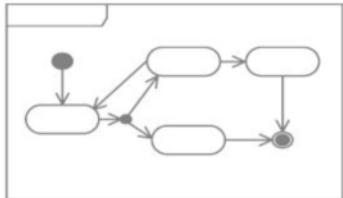
.

**Aktivitätsdiagramm**



.

**Zustandsdiagramm**

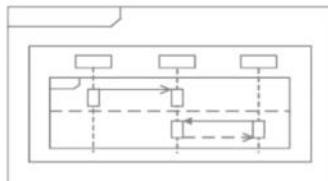


.



8. September 2020

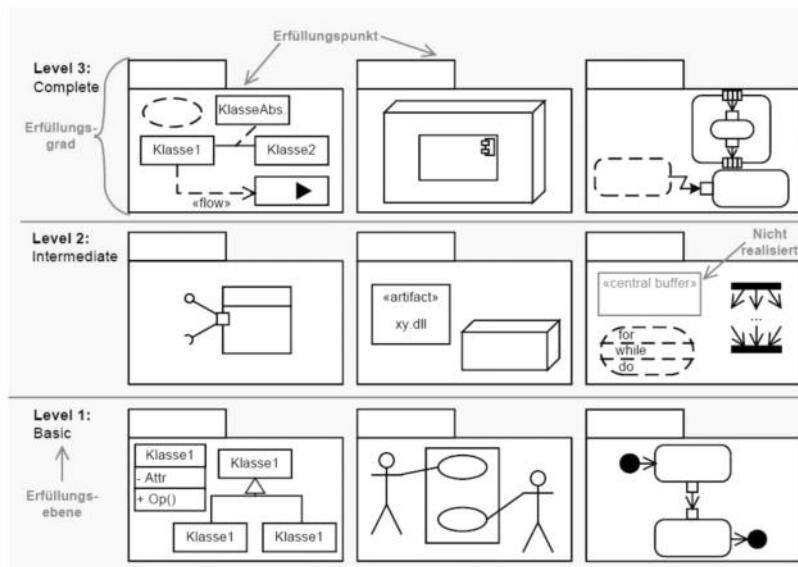
### Sequenzdiagramm



- Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus.
- Darstellung des Informationsaustausches zwischen Kommunikationspartnern
- Sehr präzise Darstellung der zeitlichen Abfolge auch mit Nebenläufigkeiten

### UML-Erfüllungsebenen

UML erlaubt durch seine Vielzahl seiner Diagramme eine sehr detaillierte Darstellung der Prozesse. Dies ist nicht immer notwendig und machbar. Folgende Grafik veranschaulicht Priorisierungen und Variationen des Detailierungsgrades.



### Verständnisfragen

- Vor welchem Problem stand anfangs die Modellierungssprache UML ?
- Welche Bedeutung haben die Begriffe Statisch/Dynamisch im Zusammenhang mit Objekten und der UML?
- Gibt es neben der UML andere Modellierungssprachen? Welche Bereiche decken diese Sprachen ab ?

## UseCase-Diagramm

Die Frage **Was soll mein geplantes System eigentlich leisten** sollte am Beginn jeder Systementwicklung stehen. **Das Use-Case-Diagramm zeigt das externe Verhalten eines Systems aus der Sicht der Nutzer.**

Das Ziel jedes Softwareentwicklungsprozesses ist es, eine Software zu entwickeln, die ganz bestimmte Anforderungen erfüllt. Die Entwicklung einer Software fängt mit der Zielsetzung an: Die Software soll, wenn fertiggestellt, die zu Beginn des Entwicklungsprozesses festgelegten Anforderungen/Ziele erfüllen.

Leider sind diese Ziele nicht immer klar definiert. Man nimmt sich zum Beispiel vor, einen Online-Shop zu entwickeln, und stellt dann während des Entwicklungsprozesses fest, dass es unendlich viele unterschiedliche Funktionen in einem Online-Shop geben kann und man sich eigentlich nie klar gemacht hat, was man denn nun für Funktionen im Detail braucht. Man muss den Entwicklungsprozess daher wiederholt unterbrechen und inne halten, um sich zu überlegen, welche Funktionen, die einem bei der Entwicklung gerade eingefallen sind, notwendig sind und welche nicht.

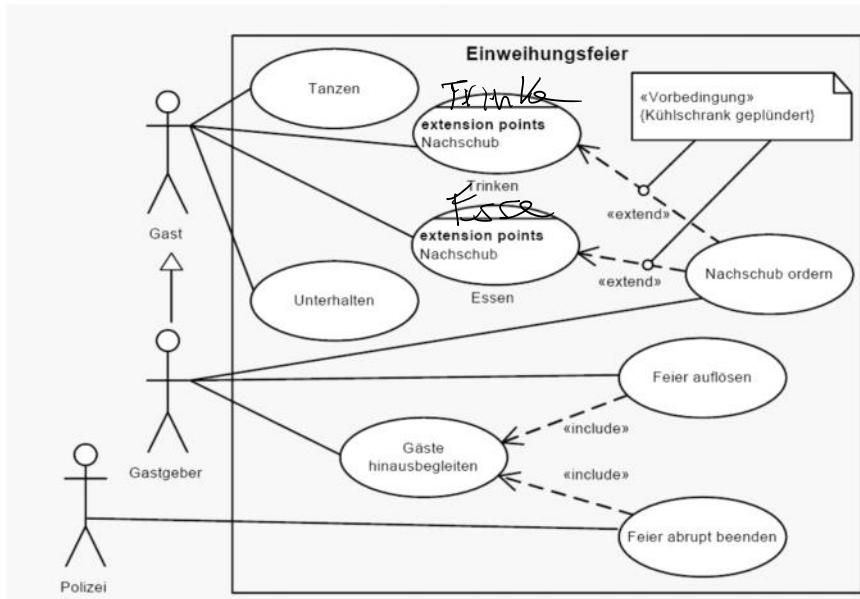
Besonders schwierig wird die Situation, wenn der Auftraggeber des Entwicklungsprozesses nicht gleichzeitig der Entwickler ist. In diesem Fall kann der Entwickler nicht entscheiden, welche Funktionen notwendig sind - dies weiß nur der Auftraggeber. Dies führt zu einem ständigen Frage-Antwort-Spiel zwischen Entwickler und Auftraggeber, wenn der Entwickler nicht - noch schlimmer - die Entscheidungen selbst trifft und hofft, dies jeweils im Sinne des Auftraggebers zu tun.

Wenn Anforderungen an die zu entwickelnde Software nicht zu Beginn des Entwicklungsprozesses klipp und klar sind, wird der Entwicklungsprozess an sich unnötig erschwert. Denn das, was Sie entwickeln, richtet sich nach den bekannten Anforderungen. Jede Anforderung, die Ihnen oder Ihrem Auftraggeber später einfällt, führt dazu, dass Sie das, was Sie bisher entwickelt haben, ändern müssen. Denn die neuen Anforderungen hatten Sie logischerweise in Ihrer bisherigen Entwicklung nicht berücksichtigt. Grundsätzlich gilt, dass je später Anforderungen in einem Entwicklungsprozess bekannt werden, umso aufwändiger und daher teurer der Entwicklungsprozess wird. Anders gesagt: Wenn alle Anforderungen von Anfang an bekannt sind, bevor der Entwicklungsprozess gestartet wird, wäre das ideal.

Im Zusammenhang mit Anforderungen setzt man als erstes Hilfsmittel das Use-Case-Diagramm ein.

8. September 2020

- Use-Case-Diagramme sind, auch bedingt durch die geringe Anzahl der Modellelemente, eingängig und übersichtlich.
- In der Projektrealität trifft man häufig eine sehr einfache skizzenhafte Verwendung von Use-Cases für erste Diskussionen.
- Sie enthalten die grafische Darstellung
  - des Systems
  - der Use-Cases
  - der Akteure außerhalb des Systems
  - der Beziehungen zwischen Akteur und Use-Case, der Akteure untereinander oder Use-Cases untereinander



## Modellelemente

Aufgabe des Use-Case-Diagramms ist es, eine grobe Ordnung in die vielen zum Teil sehr detaillierten Anforderungen zu bringen. Das Use-Case-Diagramm soll uns einen Überblick über das verschaffen, was die zu entwickelnde Software eigentlich im Wesentlichen können muss. Es werden also wichtige Funktionen der Software herausgearbeitet und zueinander in Beziehung gesetzt. Die technische Implementierung spielt bei diesem Überblick keine Rolle - vergessen Sie alles, was irgendetwas mit Programmiersprachen zu tun hat. Es geht um das Was, nicht um das Wie.

Das Use-Case-Diagramm ist ein recht einfacher Diagrammtyp, der schön anschaulich ist. Im Folgenden sehen Sie die grundlegenden Bausteine, mit denen alle Use-Case-Diagramme aufgebaut sind.

Das Use-Case-Diagramm verfügt über folgende Notationselemente:

- UseCase
- System
- Akteur
- includes-Beziehung
- extends-Beziehung

### **Use-Case = System + Anwendungsfall + Akteur**

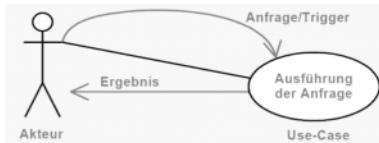
Im Use-Case-Diagramm gibt es den Akteur und das System. Der Akteur ist der Anwender, das System die zu entwickelnde Software. Im System, das als Rechteck dargestellt wird, werden verschiedene wesentliche Anforderungen platziert. Man schreibt hierzu sehr knappe Funktionsbeschreibungen in Ellipsen, wobei jede Ellipse genau eine Funktion darstellt. Die Ellipsen sind die Use-Cases - daher hat das Use-Case-Diagramm seinen Namen. Wenn Sie einen deutschen Begriff verwenden möchten: Use-Case wird für gewöhnlich mit Anwendungsfall übersetzt.



Da es beim Use-Case-Diagramm um einen ersten groben Überblick geht, beschränkt man sich auf die Darstellung wesentlicher Funktionen - alles, was nicht zum Über-

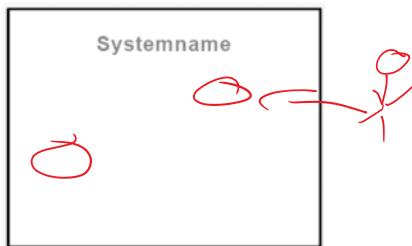
blick beiträgt, wird weggelassen. Beim Erstellen eines Use-Case-Diagramms geht es also nicht darum, möglichst viele Ellipsen in das Rechteck zu malen. Es geht darum, wesentliche

Anforderungen zu finden und diese zusammenhängend als Use-Cases in das System einzzeichnen. Die Zusammenhänge zwischen Use-Cases und dem Akteur als auch zwischen Use-Cases untereinander werden durch Verbindungslinien dargestellt.

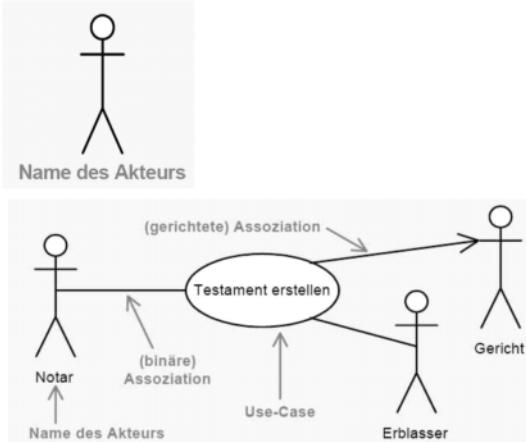


- Beschreibt eine Reihe von Aktionen, die nacheinander ein Verhalten formen
- Wird immer von einem Akteur ausgelöst
- Hat immer ein Ergebnis
- Kann gleichzeitig mehrfach instanziert werden
- Spiegelt funktionales Verhalten wider. Interne Abläufe sind irrelevant.

## System



## Akteur



## Beziehungen

Die Zusammenhänge zwischen Use-Cases und dem Akteur als auch zwischen Use-Cases untereinander werden durch Verbindungslien dargestellt.

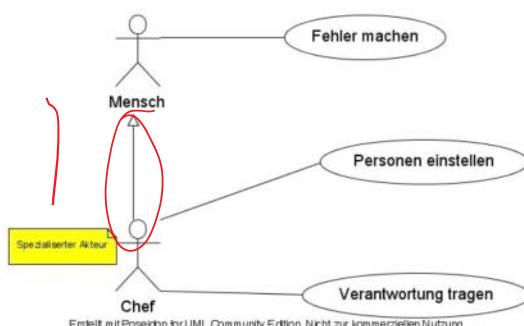
Verbindungslien in UML-Diagrammen werden Assoziationen genannt. Sie stellen Zusammenhänge zwischen Elementen an den Enden von Assoziationen dar. Welche Art von Assoziation ausgedrückt wird, hängt von der Darstellung der Verbindungslien und von Schlüsselwörtern ab, die an der Verbindungslien stehen können.

Im Use-Case-Diagramm gibt es zwei Arten von Verbindungslien: Die durchgezogene Verbindungslien stellt eine Assoziation zwischen dem Akteur und einem Use-Case dar. Sie bedeutet, dass der Akteur den Use-Case in irgendeiner Form anwendet. Der Akteur tauscht also Informationen mit dem Use-Case aus. Das kann zum Beispiel bedeuten, dass er eine Funktion des Systems startet oder ihm von einer Funktion des Systems Daten ausgegeben werden.

Die gestrichelte Verbindungslien stellt eine Assoziation zwischen zwei Use-Cases dar. Da es zwei verschiedene Arten von Assoziationen zwischen Use-Cases gibt, wird neben die gestrichelte Verbindungslien ein Schlüsselwort gesetzt. Schlüsselwörter in der UML, die zur Spezifizierung von Verbindungslien oder anderen geometrischen Formen verwendet werden, werden immer zwischen doppelte spitze Klammern gestellt. Diese Schlüsselwörter werden in der UML Stereotypen genannt.

### Beziehungen zwischen Akteuren

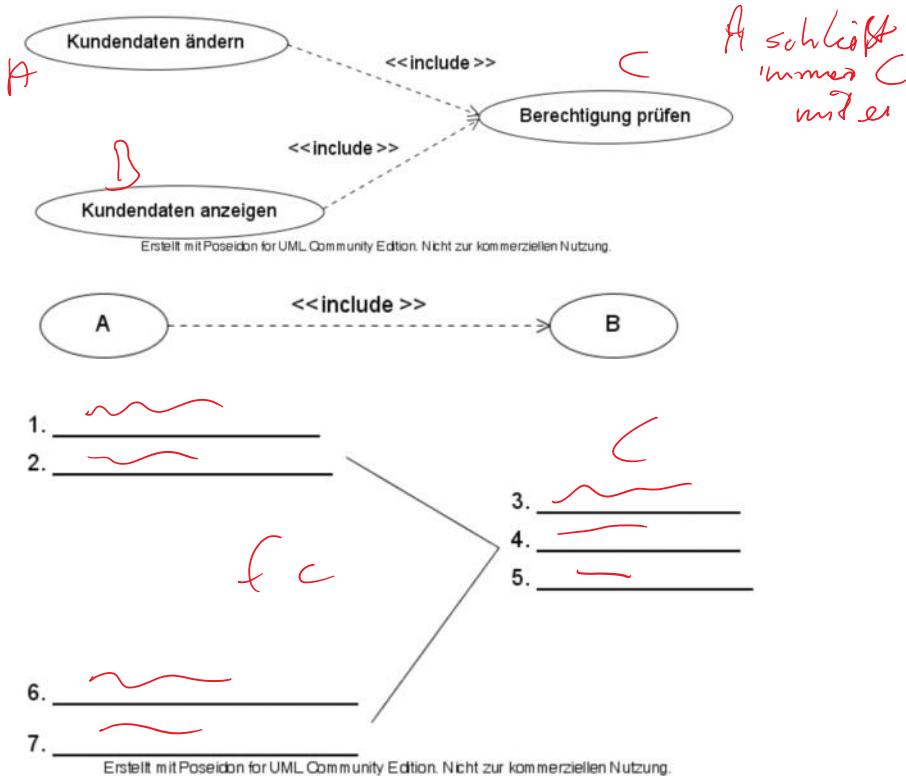
Akteure dürfen untereinander in Beziehung stehen, es sind sogar Vererbungsbeziehungen erlaubt. Der spezialisierte Akteur ist dann an den gleichen UseCase-Abläufen beteiligt wie der vererbende Akteur.



## includes-Beziehung

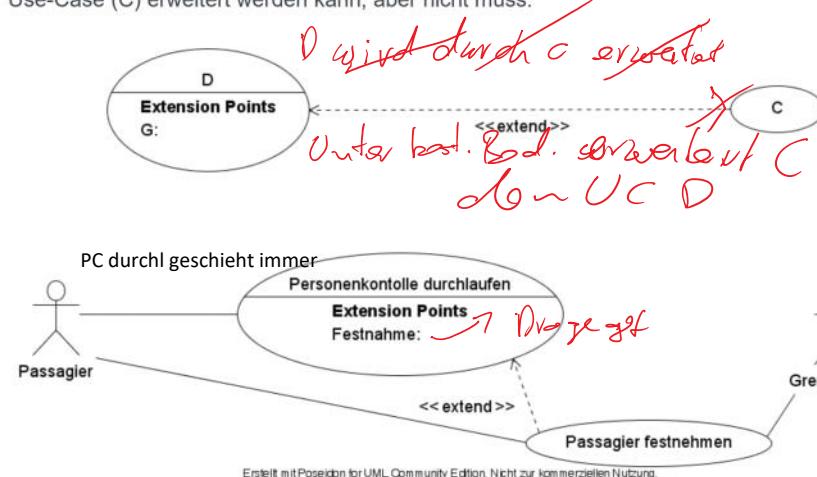
Darstellung durch Abhängigkeitsbeziehung mit Stereotyp include. Der *linke\*linke Use-Case importiert das Verhalten des \*rechten Use-Case*.

Eine include-Beziehung ist nicht optional; das Verhalten wird immer eingeschlossen. Erst durch die Inklusion ergibt sich ein (sinnvolles) Gesamtverhalten. Rekursive Include-Beziehungen sind nicht erlaubt.



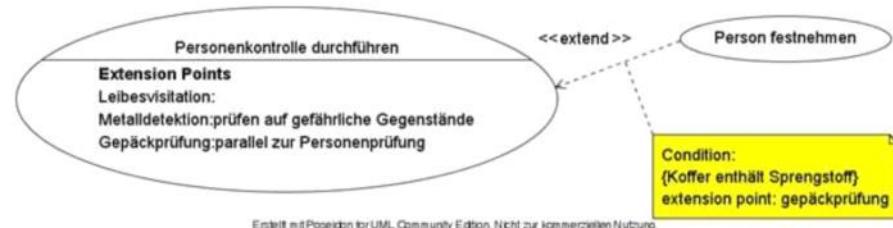
### extends-Beziehung

Eine extend-Beziehung zeigt an, dass das Verhalten eines Use-Case (D) durch einen anderen Use-Case (C) erweitert werden kann, aber nicht muss.

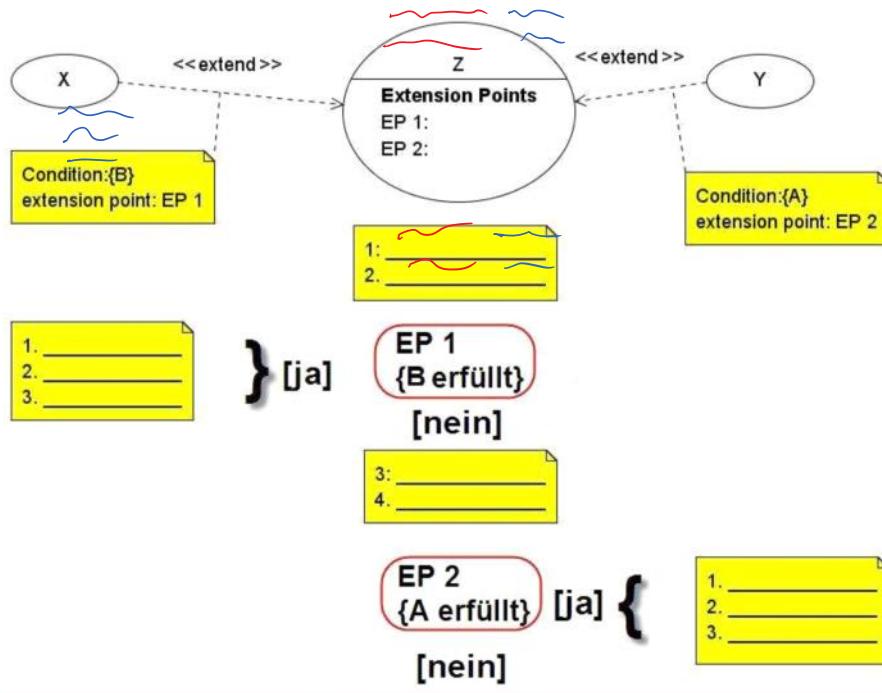


Obige Abbildung zeigt, dass der UseCase *Personenkontrolle durchlaufen* in bestimmten Fällen durch *Passagier festnehmen* erweitert wird.

Der Zeitpunkt, an dem ein Verhalten eines UseCases erweitert werden kann, wird als **Extension point** bezeichnet.

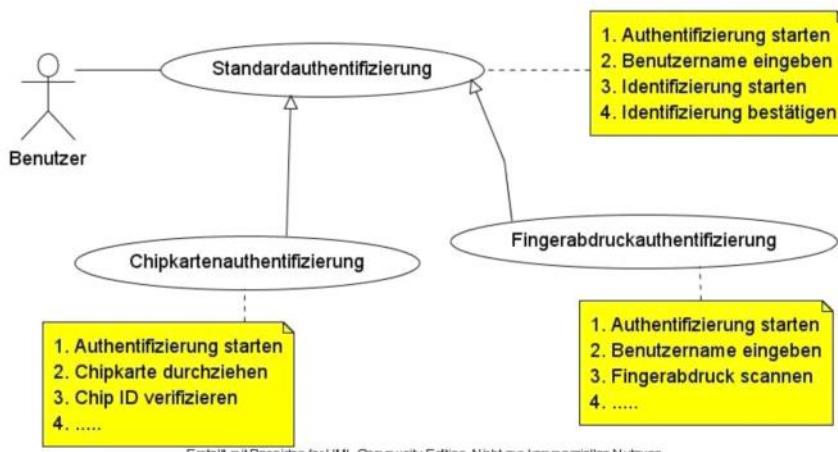


Neben dem Erweiterungspunkt kann eine Bedingung für die Erweiterung angegeben werden. Die Bedingung wird bei Erreichen des Erweiterungspunktes geprüft. Ist die Bedingung wahr, wird der Ablauf erweitert; ist die Bedingung nicht erfüllt, läuft der UseCase *normal* weiter



### Vererbung

Zwischen UseCases kann auch eine Vererbungsbeziehung modelliert werden. Sie entspricht von Konzept her der Vererbung im Klassendiagramm.



Erstellt mit Poseidon for UML Community Edition. Nicht zur kommerziellen Nutzung.

8. September 2020

In der Praxis werden Generalisierungs- und Extends-Beziehungen häufig falsch eingesetzt.

### Unterschiede include-extend

## Unterschiede include-extend

<<include>>

```
includes  
A -----> B
```

<<extends>>

```
extends  
A <----- B
```



8. September 2020

### Geschäftsprozessschablonen

Wie werden Use Cases erstellt? Es werden für ein definiertes System alle Akteure festgestellt (hierin liegt das größte Problem, dass ein Akteur übersehen und/oder vergessen wird). Alle Anforderungen der jeweiligen Akteure werden festgehalten und der Reihe nach in Schablonen übertragen. Die Schablonen werden nummeriert. Nicht alle Werte einer Schablone müssen für jeden Vorgang (Aktion) ausgefüllt sein.

- **Übergeordneter elementarer Geschäftsprozess**

Hinweis auf übergeordneten Geschäftsprozess, wenn der betrachtete GP in einer <<include>> bzw. <<extends>> - Beziehung steht.

- **Beschreibung**

Der Name des UseCase.

- **Ziel**

Was ist das Ziel des UseCase. Hier erfolgt die genaue Darstellung des UseCases.

- **Vorbedingung**

Erwarteter Zustand, bevor der GP beginnt.

- **Nachbedingung bei erfolgreicher Ausführung**

Erwarteter Zustand nach erfolgreicher Ausführung des Geschäftsprozesses.

- **Nachbedingung bei fehlgeschlagener Ausführung**

Erwarteter Zustand, wenn das Ziel nicht erreicht werden kann.

- **Beteiligte Nutzer**

Rollen von Personen oder anderen Systemen, die den GP auslösen oder daran beteiligt sind.

### Geschäftsprozess-Schablone (Balzert)

- **Auslösendes Ereignis**

Wenn dieses Ereignis eintritt, dann wird der GP initiiert.

Geschäftsprozess	Name des Anwendungsfalles
<b>Ziel</b>	Kurze Beschreibung des Zwecks
<b>Kategorie</b>	Primär, Sekundär oder Optional
<b>Vorbedingung</b>	Erwarteter Zustand vor Ausführung des Anwendungsfalles
<b>Nachbedingung Erfolg</b>	Erwarteter Zustand nach erfolgreicher Ausführung des Anwendungsfalles
<b>Nachbedingung Fehlschlag</b>	Erwarteter Zustand, wenn das Ziel nicht erreicht wird
<b>Akteure</b>	Wer ist an dem Anwendungsfall beteiligt?
<b>Auslösendes Ereignis</b>	Nach welchem Ereignis tritt dieser Anwendungsfall auf?
<b>Beschreibung</b>	Ablauf des Anwendungsfalles
<b>Erweiterungen</b>	Was läuft parallel, bzw. nach diesem Anwendungsfall ab und kann diesem zugerechnet werden.
<b>Alternativen</b>	Wie kann dieser Anwendungsfall noch ablaufen?

## Zusammenfassung

### Fazit

Anforderungen und ihre Zusammenhänge ohne Berücksichtigung der Reihenfolge

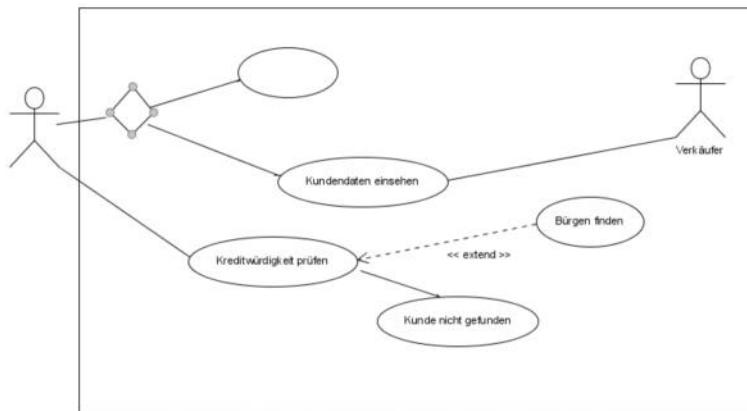
Use-Case-Diagramme sind Verhaltensdiagramme: Sie beschreiben bestimmte Aspekte, wie sich ein System verhält. Wie Sie nun gesehen haben, können im Use-Case-Diagramm wesentliche Funktionen des Systems hervorgehoben und zueinander in Beziehung gesetzt werden. Diesen Diagrammen fehlt jedoch eine Möglichkeit, die Reihenfolge der Ausführung festzulegen. Das geht im eingeschränkten Maße mit include- und extend-Assoziationen. Da ein Use-Case in diesen Fällen aber jeweils vom anderen Use-Case eingeschlossen wird - bei include immer, bei extend in Abhängigkeit einer Bedingung - handelt es sich nicht um eine strikte Reihenfolge: Ein Use-Case, der einen anderen mit include oder extend einschließt, läuft nach Beendigung des eingeschlossenen Use-Case weiter.

Dieser Nachteil des Use-Case-Diagramms ist an sich kein Nachteil, da Use-Case-Diagramme nicht für die Beschreibung einer Reihenfolge von Abläufen vorgesehen sind. Es ist also wichtig zu verstehen, dass jedes Diagramm bestimmte Aspekte eines Systems hervorhebt und beschreibt, aber nie alle. Es ist nicht die Aufgabe des Use-Case-Diagramms, die Reihenfolge einer Ausführung zu beschreiben. Zu diesem Zweck kann ein anderer Diagrammtyp verwendet werden, den Sie im nächsten Kapitel kennenlernen werden.

8. September 2020

### Fragen zu UseCase

- Was ist falsch an folgender Darstellung



Erstellt mit Poseidon für UML Community Edition. Nicht zur kommerziellen Nutzung.

- Die folgenden Anforderungen beschreiben eine einfache Artikelverwaltung, die objektorientiert zu modellieren ist.
  - Eine Firma will ihre Artikel und Bestellungen verwalten.
  - Für jeden Artikel werden die Artikelnummer, die Artikelbezeichnung und der Verkaufspreis festgehalten. Jeder Artikel gehört zu einer oder mehreren Artikelgruppen.
  - Jeder Artikel kann an mehreren Orten gelagert werden. Beispielsweise lagert die Firma Artikel in ihren Filialen Truchtelfingen, Adolfzfurt und Ahrenviölfeld. Pro Lagerort werden für jeden Artikel dessen Maximalbestand, Mindestbestand, aktueller Bestand und Name des Lagers gespeichert. Eine Lagerliste soll Auskunft über die Bestände geben.
  - Jeder Artikel kann von verschiedenen Lieferanten bezogen werden. Dabei gibt es je nach Lieferant unterschiedliche Einkaufspreise und Verpackungseinheiten. Bei jedem Lieferanten besitzt der Artikel eine eigene Bestellnummer.
  - Für jeden Lieferanten müssen dessen Name, Adresse, Telefonnummer und der Ansprechpartner gespeichert werden. Jeder Lieferant kann mehrere Artikel liefern.
  - Jeden Tag werden die Lagerbestände kontrolliert. Wird der Mindestbestand unterschritten, so wird ein Bestellvorschlag erstellt. Er enthält außer den Daten eines Artikels auch die verschiedenen Lieferanten mit ihren Lieferkonditionen. Für jedes Lager wird errechnet, wie viele Artikel nachbestellt werden müssen. Die vorgeschlagene Anzahl errechnet sich aus dem Maximalbestand und dem aktuellen Bestand. Die Anzahl ist auf ein n-faches der Verpackungseinheit eines jeden Lieferanten abzurunden.
  - Jeder Artikel kann in beliebiger Anzahl bestellt werden. Dabei wird vom System automatisch der günstigste Lieferant gewählt.

8. September 2020

8. Jeder Artikel wird einzeln bei einem Lieferanten bestellt. Für jede Lieferantenbestellung werden das Bestelldatum und das Lieferdatum festgehalten. Der Einfachheit halber werden Teillieferungen ausgeschlossen.
9. Ein Kunde kann eine oder mehrere Bestellungen erteilen, die erfasst werden müssen. Jede Bestellung erhält eine eindeutige Bestellnummer. Außerdem werden das Bestelldatum, das Lieferdatum und die Portokosten festgehalten.
10. Jede Kundenbestellung kann sich auf mehrere Artikel beziehen. Für jeden bestellten Artikel ist die gewünschte Anzahl festzuhalten. Außerdem kann der Gesamtpreis für mehrere Artikel ungleich Anzahl\*Verkaufspreis sein.
11. Jede Kundenbestellung wird von einem Sachbearbeiter bearbeitet. Für jeden Sachbearbeiter sollen dessen Personalnummer, ein Kürzel, der Name und die Telefonnummer gespeichert werden.
12. Außerdem sollen folgende Statistiken erstellt werden:
  - Welchen Umsatz hat ein Kunde X im aktuellen Jahr erzielt?
  - Welchen Umsatz hat ein Sachbearbeiter X mit seinen verschiedenen Kunden erzielt.

# Übung UseCase

Mittwoch, 9. September 2020 19:45

# UseCase

Die Auftraggeber einer Online-Videothek haben eine erste Vorstellung, wie eine Benutzerverwaltung aussehen und welche Funktionen es geben soll. Ein Entwickler macht sich in einem Gespräch diesbezüglich folgende Notizen:

Um die Funktionen der Online-Videothek zu nutzen, müssen Kunden sich zuerst einloggen. Dabei wird der Benutzerstatus überprüft. Die Kunden können anschließend Filme ausleihen, oder ihr Guthaben auffüllen.

Bei der Ausleihe wird das Alter des Kunden mit der FSK-Angabe des Films abgeglichen und gegebenenfalls die Ausleihe des Films verweigert. Für die sichere Online-Zahlung bei der Guthabensauffüllung verwendet das System den externen Dienstleister „SuperPay“.

Ein Administrator soll (ebenfalls nach einem Login) in der Lage sein, aus Kulanz direkt das Guthaben der Kunden zu erhöhen.

Der Administrator kann ferner Benutzer sperren, wenn Sie gegen die AGB verstößen haben. In diesem Fall kann ein Benutzer sich nicht mehr am System anmelden. Ein Anmeldeversuch wird mit einer entsprechenden Meldung abgebrochen.

- a) Erstellen Sie für obige Notizen ein Use Case-Diagramm, das die relevanten Anwendungsfälle darstellt und untereinander in Beziehung setzt. Identifizieren Sie die relevanten Akteure.

## Aufgabe zu UseCase

- a) Modellieren Sie die folgenden Zusammenhänge in einem *Anwendungsfalldiagramm*.

Erstellt werden soll ein System zur Vermittlung von Mitfahrtgelegenheiten. Die Nutzer des Systems können in Fahrer und Mitfahrer unterteilt werden. Fahrer können ihre Fahrten im System eingeben. Dabei kann zwischen dem Eingeben von so genannten **Etappenfahrten** und **einfachen Fahrten** unterschieden werden. Beide Eingabevorgänge weisen viele Übereinstimmungen, jedoch auch einige Besonderheiten auf.

Mitfahrer können nach Mitfahrtgelegenheiten suchen. Sowohl das Suchen als auch das Eingeben von Fahrten beinhaltet das Erfassen von Streckenpunkten. Außerdem können sowohl Fahrer als auch Mitfahrer Mitfahrtgelegenheiten absagen. (6T)

Sommer\_2010

### 2. Handlungsschritt (25 Punkte)

Ein Ziel des Projektes ist die Verbesserung des Kundenservice. Im Rahmen dieses Vorhabens soll eine neue Internetpräsenz erstellt werden.

- a) Die neue Internetpräsenz soll unter anderem folgende Funktionalität bieten:

Alle Besucher der Webseite können allgemeine Informationen der Krake AG abrufen (Leistungs- und Produktpotfolio und AGB). Ein Kunde kann Reservierungsanfragen stellen sowie Buchungen und Stornierungen durchführen. Von einem Neukunden werden zunächst die Kundendaten erfasst. Für Buchungen und Stornierungen muss sich ein Kunde einloggen; seine eingegebenen Daten werden überprüft.

Erstellen Sie ein entsprechendes Anwendungsfalldiagramm.

(15 Punkte)

So\_2012\_ga1\_awp

### 1. Handlungsschritt (25 Punkte)

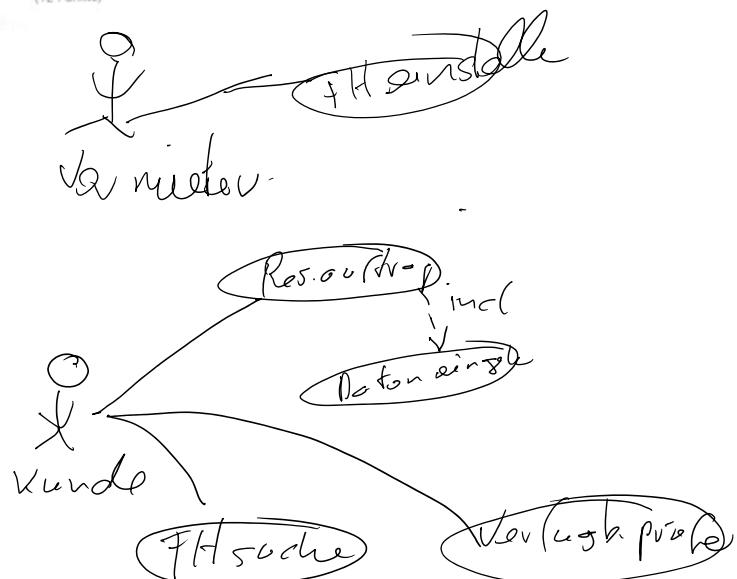
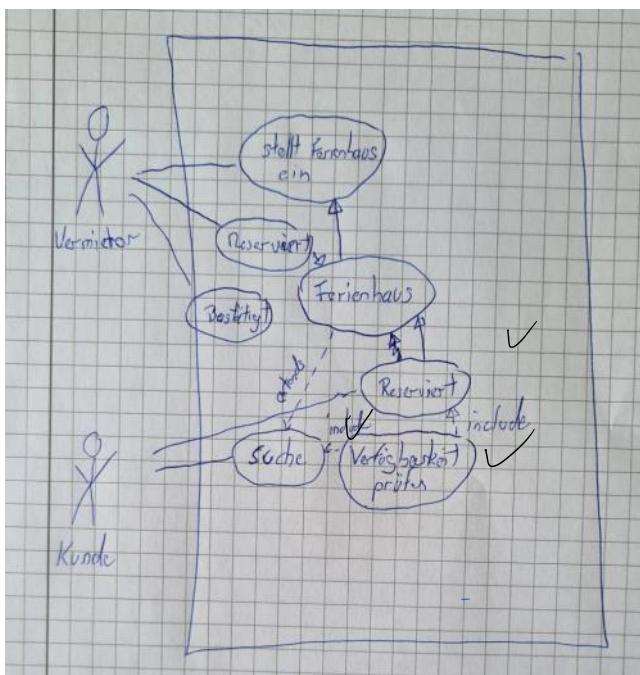
Die B&G GmbH vermietet auch exklusive Ferienhäuser. Die Immo-IT GmbH soll dazu eine Anwendung entwickeln, über die im Internet Ferienhäuser angeboten und gebucht werden können.

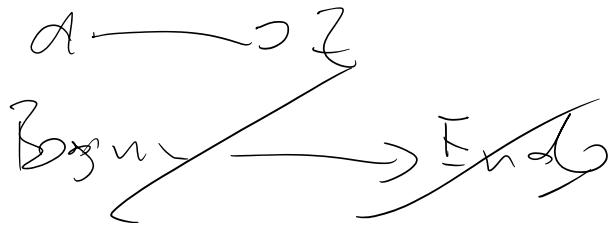
- a) Das System soll Folgendes ermöglichen:

- Ein Vermieter stellt ein Ferienhaus ein. ✓
- Ein Kunde sucht ein Ferienhaus. ✓
- Ein Kunde prüft die Verfügbarkeit eines ausgewählten Ferienhauses. ✓
- Ein Kunde stellt einen Reservierungsauftrag und gibt alle erforderlichen Daten ein. ✓
- Ein Vermieter reserviert ein Ferienhaus und verschickt eine Reservierungsbestätigung an den Kunden.

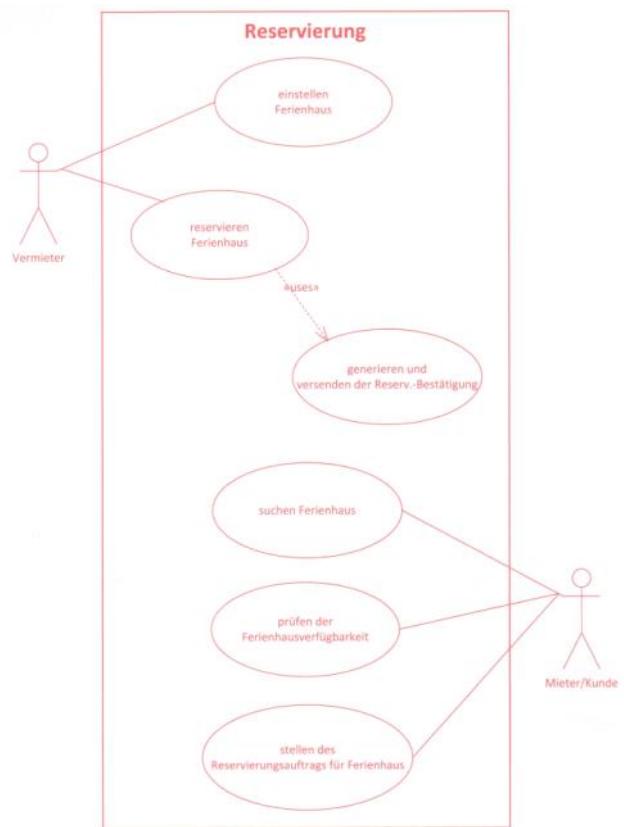
Erstellen Sie ein UML-Anwendungsfalldiagramm.

(12 Punkte)





Musterlösung Kammer:



## **2. Handlungsschritt (25 Punkte)**

Ein Ziel des Projektes ist die Verbesserung des Kundenservice. Im Rahmen dieses Vorhabens soll eine neue Internetpräsenz erstellt werden.

a) Die neue Internetpräsenz soll unter anderem folgende Funktionalität bieten:

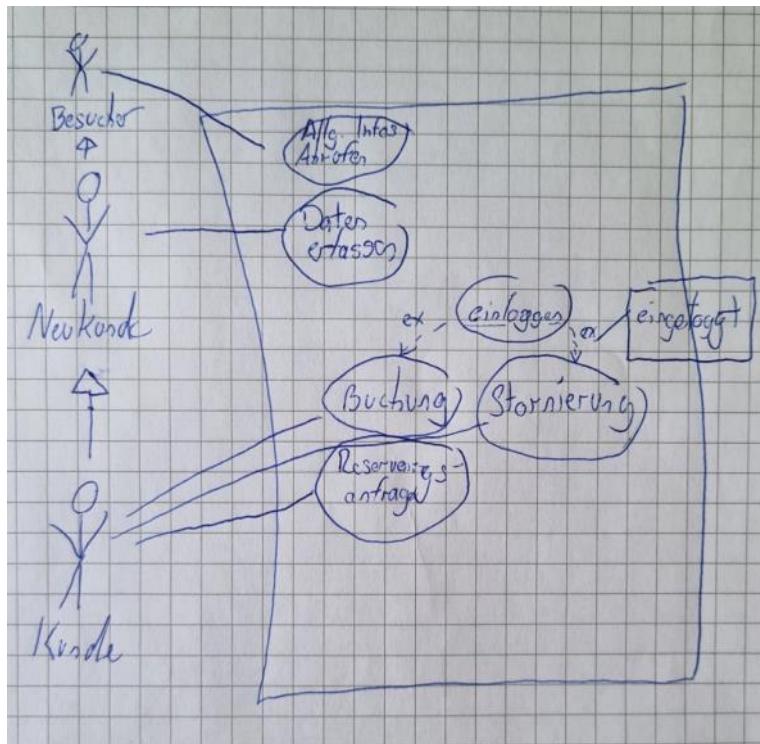
Alle Besucher der Webseite können allgemeine Informationen der Krake AG abrufen (Leistungs- und Produktpotfolio und AGB).

Ein Kunde kann Reservierungsanfragen stellen sowie Buchungen und Stornierungen durchführen. Von einem Neukunden werden zunächst die Kundendaten erfasst. Für Buchungen und Stornierungen muss sich ein Kunde einloggen; seine eingegebenen Daten werden überprüft.

Erstellen Sie ein entsprechendes Anwendungsfalldiagramm.

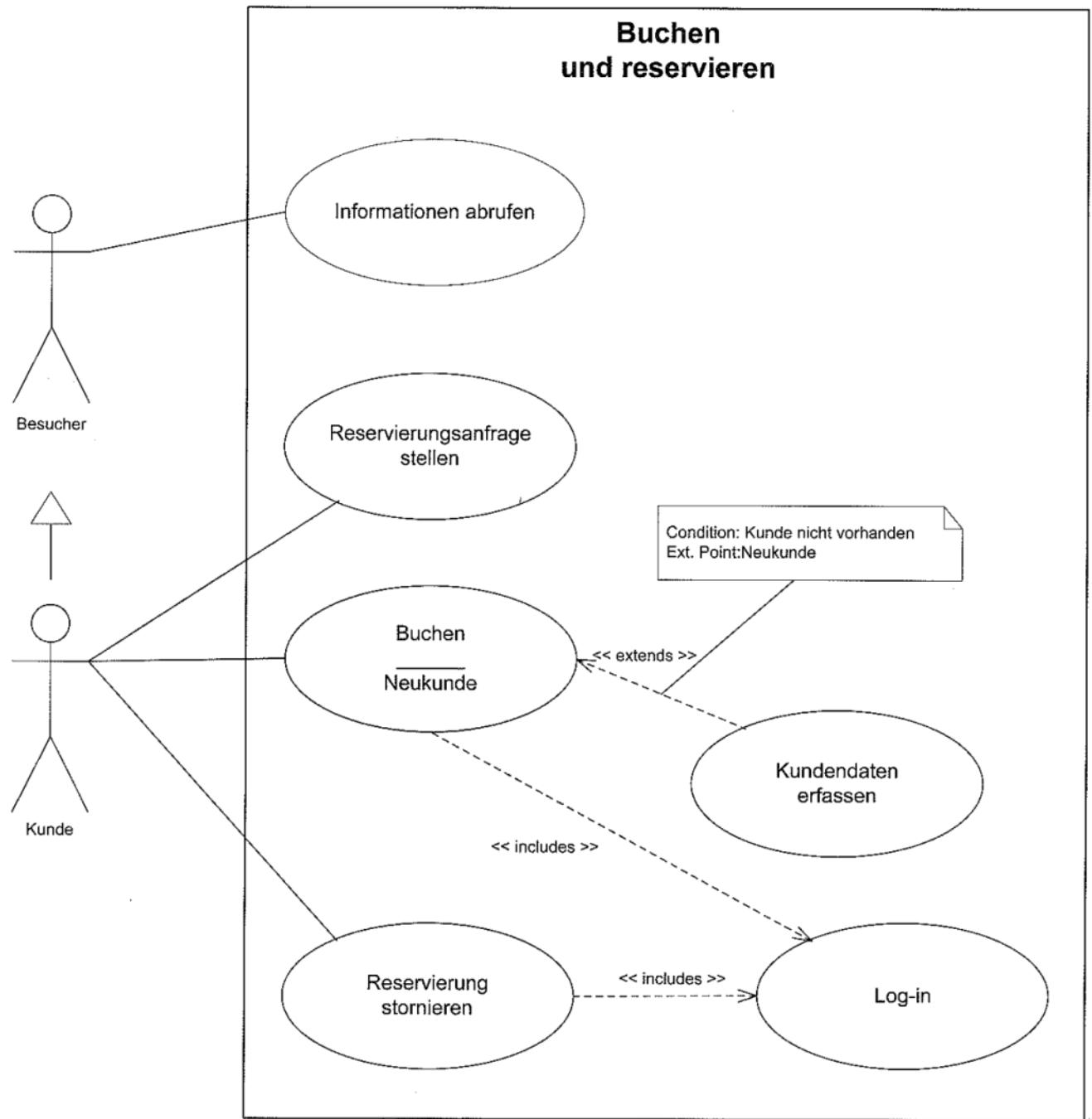
(15 Punkte)

Lösung Dennis



Musterlösung

a) 15 Punkte



# Winter 2014

Donnerstag, 10. September 2020 21:27

## Fortsetzung 1. Handlungsschritt

- c) Die Soft GmbH wurde von der FAQ GmbH mit der Entwicklung einer Software beauftragt, die den Kunden der FAQ GmbH einen Onlinezugang zu statistischen Daten ermöglicht. Folgende Anforderungen an die Software „Statistikabfragen“ liegen vor:
- Jeder Nutzer des Onlineangebotes der FAQ GmbH kann Standardstatistiken abrufen.
  - Ein Premiumnutzer kann zusätzlich Premiumstatistiken abrufen. Dazu ist ein Login erforderlich. Falls die Login-Daten nicht vorliegen (z. B. Erstanmeldung), muss dieser Nutzer die erforderlichen Daten eingeben.
  - Ein Administrator kann verschiedene Admin-Tools abrufen. Auch dazu ist ein Login erforderlich. Ein Administrator kann nur Standardstatistiken abrufen.

Erstellen Sie anhand der vorliegenden Informationen ein UML-Anwendungsfalldiagramm für die Software „Statistikabfragen“. 12 Punkte

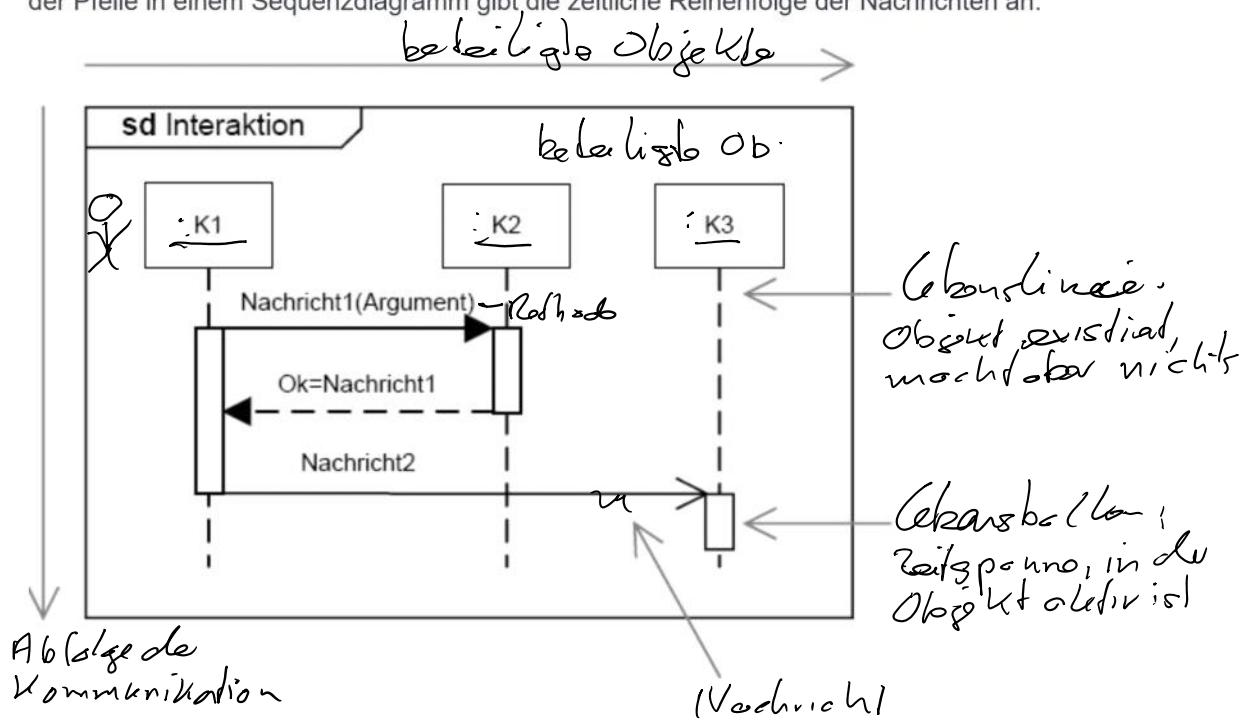
# Doku Sequenzdiagramm

Donnerstag, 10. September 2020 21:45

# Sequenzdiagramm

"Erst die richtige Reihenfolge macht aus klugen Gedanken ein gutes Ergebnis.  
A. van Rheyn"

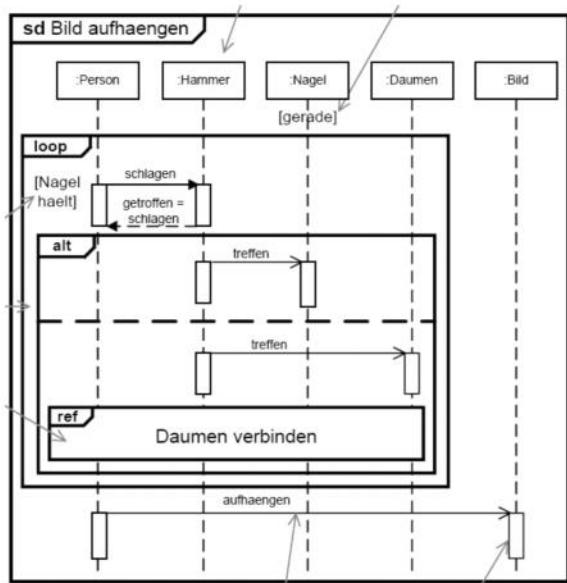
Sequenzdiagramme beschreiben die Kommunikation zwischen Objekten in einem bestimmten Szenario (UseCase). Es wird beschrieben **welche Objekte** an einem Szenario beteiligt sind, **welche Informationen** (Nachrichten) sie austauschen und in welcher **zeitlichen Reihenfolge** der Informationsaustausch stattfindet. Sequenzdiagramme enthalten eine implizite Zeitachse. Die Zeit schreitet in einem Diagramm von oben nach unten fort. Die Reihenfolge der Pfeile in einem Sequenzdiagramm gibt die zeitliche Reihenfolge der Nachrichten an.



Es wird vor allem benutzt zur

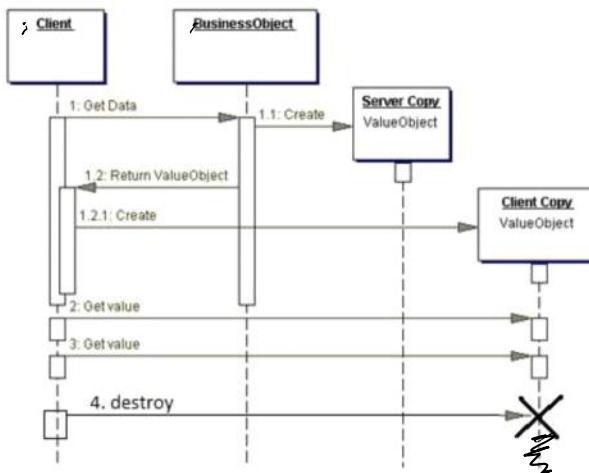
- Kommunikation und Beschreibung von UseCases
- Detailmodellierung im Feindesign
- Spezifikation von Schnittstellen, Tests und Simulation

## Modellelemente



Das Sequenzdiagramm verfügt über relativ viele Notationselemente und kann leicht unübersichtlich werden. Es ist deshalb zu empfehlen, die betrachtete Sequenz zu reduzieren.

## Klasse/Objekt/Lebenslinie



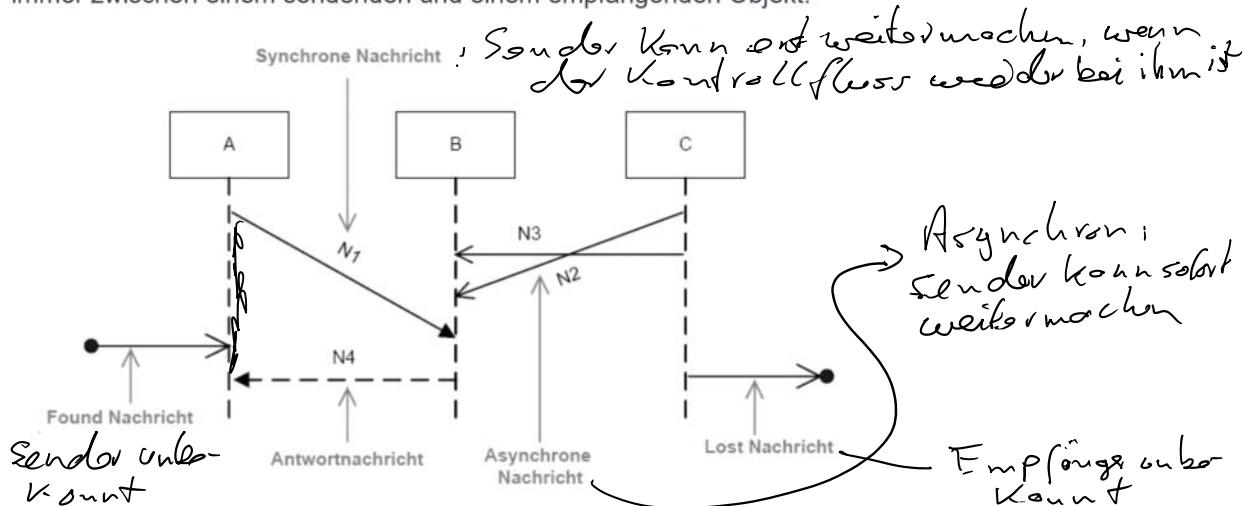
In dem Rechteck oberhalb der gestrichelten Linie wird der Objektname und der Klassename angegeben. Der Name wird unterstrichen. Die senkrechte, gestrichelte Linie stellt die Lebenslinie (lifeline) eines Objekts dar. In diesem zeitlichen Bereich existiert das Objekt. Das schmale Rechteck auf der gestrichelten Linie stellt eine Aktivierung dar. Eine Aktivierung ist der Bereich, in dem eine Methode des Objektes aktiv ist (ausgeführt wird). Auf einer Lebenslinie können mehrere Aktivierungen enthalten sein.

Das Erzeugen eines Objektes wird durch eine Nachricht, die im Kopf des Objekts endet, dargestellt (gestrichelte Linie). Sie erhält häufig den Stereotyp <<create>>.

Das Zerstören (Löschen) eines Objektes wird durch ein X auf der Lebenslinie markiert. Das Objekt existiert anschließend nicht mehr innerhalb des Szenarios.

## Nachricht

Objekte kommunizieren über Nachrichten. Nachrichten werden als Pfeile zwischen den Aktivierungen eingezeichnet. Der Name der Nachricht steht an dem Pfeil. Eine Nachricht liegt immer zwischen einem sendenden und einem empfangenden Objekt.



Der Pfeil mit der ausgefüllten Spitze bezeichnet einen synchronen Aufruf. Der Aufruf erfolgt von der Quelle zum Ziel, d. h. die Zielklasse muss eine entsprechende Methode implementieren. Die Quelle wartet mit der weiteren Verarbeitung bis die Zielklasse ihre Verarbeitung beendet hat und setzt die Verarbeitung dann fort. Ein Aufruf muss einen Namen haben; in runden Klammern können Aufrufparameter angegeben werden. Der gestrichelte Pfeil ist der Return. Die Bezeichnung des Return mit einem Namen ist optional.

Mit einer offenen Pfeilspitze werden asynchrone Nachrichten gekennzeichnet. Der Aufruf erfolgt von der Quelle zum Ziel. Die Quelle wartet mit der Verarbeitung nicht auf die Zielklasse, sondern setzt ihre Arbeit nach dem Senden der Nachricht fort. Asynchrone Aufrufe werden verwendet, um parallele Threads zu modellieren. Bei den sog. Lost-/Found-Nachrichten ist der Empfänger bzw. der Absender unbekannt.

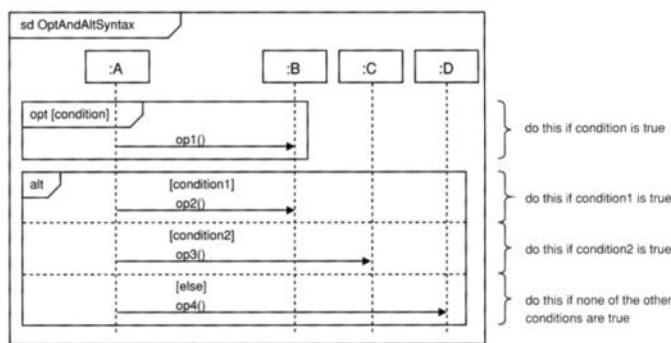
→ Übung Aufbewerksdott

## Kombinierte Fragmente

Die kombinierten Fragmente (engl. combined fragments) bieten die Möglichkeit, in Sequenzdiagrammen bedingte Anweisungen und Verzweigungen, Schleifen und bestimmte Ausführungsarten zu modellieren. Dazu wird der Ablauf in ein oder mehrere Fragmente unterteilt. Eine Auswahl verschieden Interaktionsoperatoren bietet verschiedene Möglichkeiten, wie die erstellten Fragmente miteinander kombiniert und ausgeführt werden. Untenstehend werden einige wichtige Arten vorgestellt.

### Kombinierte Fragmente – alt/opt

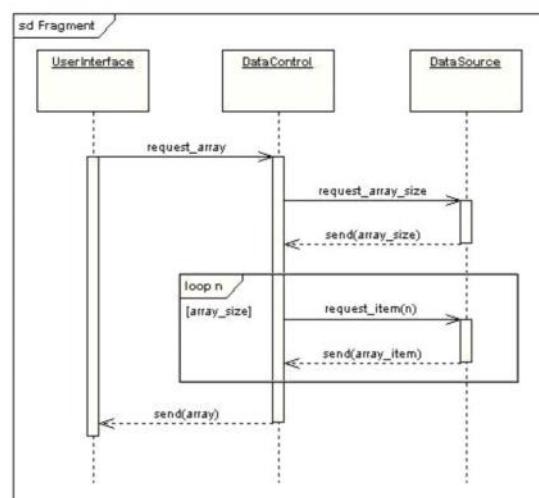
Das **alt**-Fragment ermöglicht die Existenz von mehr als einer Ablaufmöglichkeit und damit die



Modellierung von -If-else-Bedingungen-Switch/case -Anweisungen. Jeder Abschnitt kann eine eigene Bedingung enthalten. Das **opt**-Fragment steht eher für eine einzige Bedingung (if)

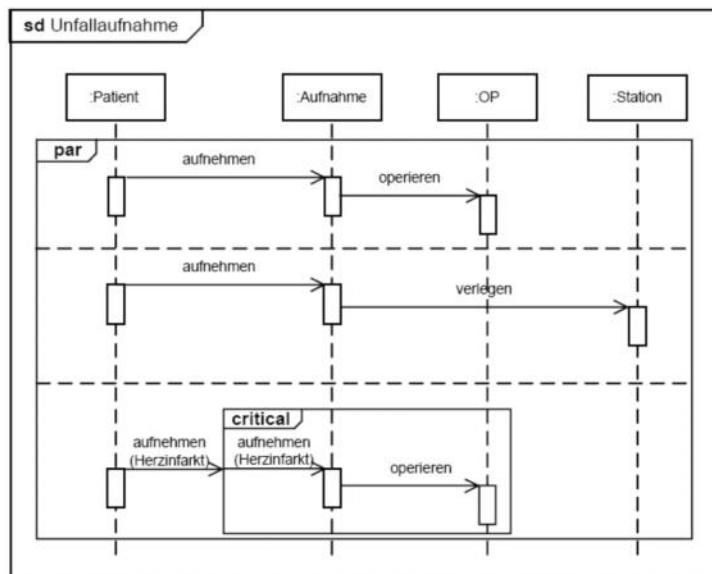
### Kombinierte Fragmente – loop

Das loop-Fragment beschreibt die Anwendung von Schleifen. Es kann durch [min, max]-Informationen erweitert werden.



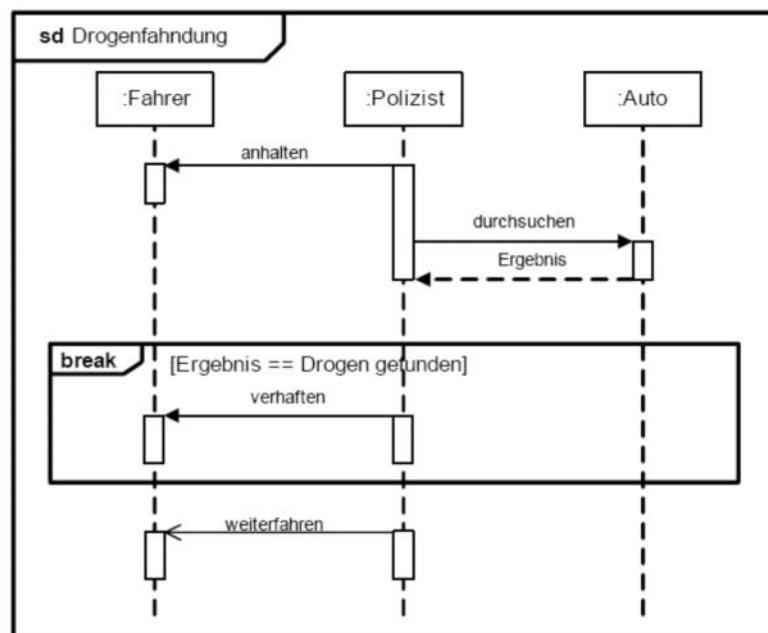
## Kombinierte Fragmente – (par — critical)

**Par** beschreibt den gleichzeitigen Ablauf mehrerer Interaktionen, während **critical** einen nicht unterbrechbaren Ablauf beschreibt.



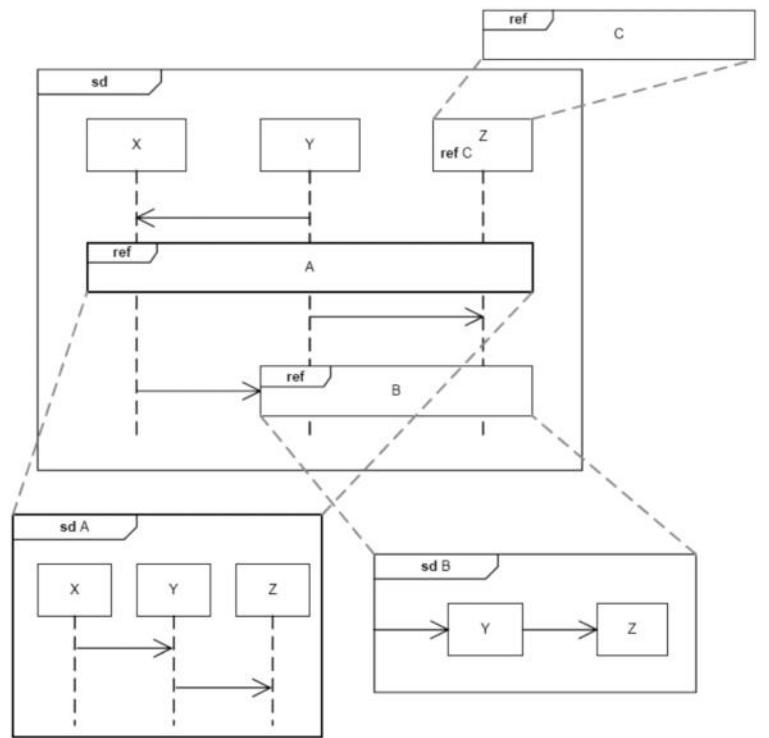
## Kombinierte Fragmente – break

Das **break**-Fragmente unterbricht den normalen Ablauf und ist geeignet zur Darstellung von Exceptions, Systemfehlern und kritischen asynchronen Ereignissen.



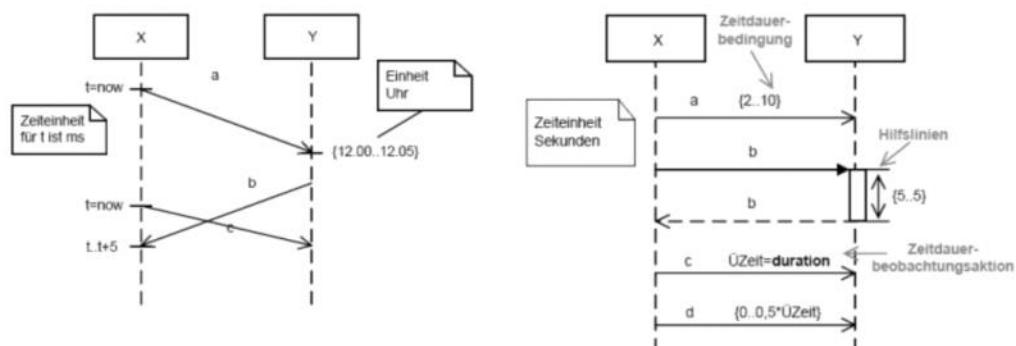
## Kombinierte Fragmente – ref

Das **ref**-Fragment verweist auf eine beliebige andere Interaktion und macht damit die mehrfache Verwendung von Sequenzdiagrammen möglich.



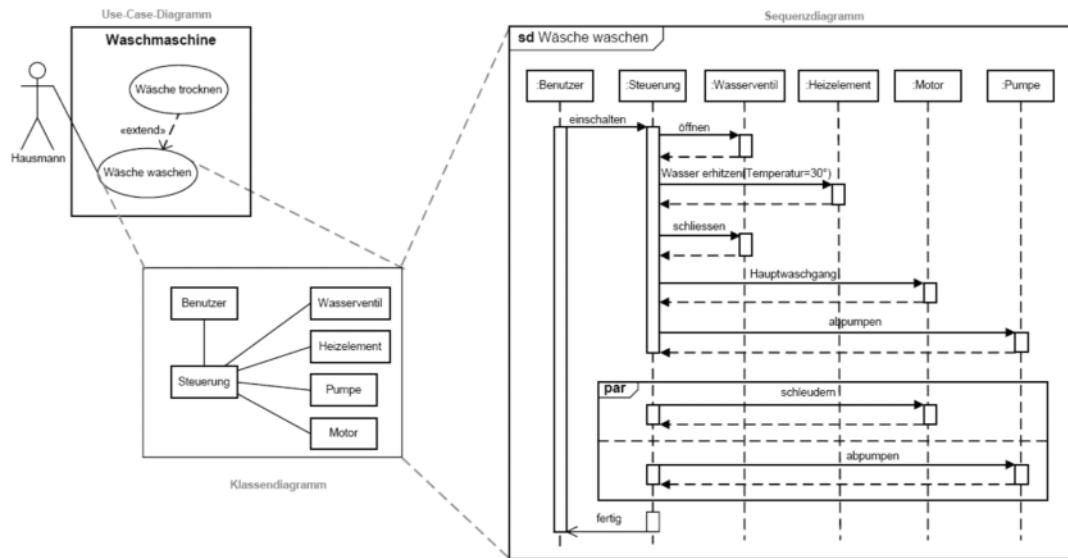
## Zeitpunkt/Zeitdauer

Sie beschreiben, wann ein Ereignis eintritt bzw. die Dauer von Nachrichten oder Aktionssequenzen



## Use Case und Sequenzdiagramme

Das folgende Bild beschreibt den Zusammenhang und das sich ergänzende Verhalten zwischen UseCase-, Sequenz- und Klassendiagramm.



*Nos Sequenzdiagramm bringt  
Use Cases und Klassen zusammen*

## Aufgaben

1. Erstellen Sie für das Szenario des Abwickelns einer Autoreparatur ein Sequenzdiagramm.

Zunächst beauftragt der Kunde die Reparatur und übergibt danach dem Mechaniker das Auto. Dieser holt den Auftrag von der Auftragsannahme ein. Nachdem der Mechaniker die Reparatur (innerhalb von 24 Stunden) beendet hat, meldet er dies der Auftragsannahme, die daraufhin den Kunden benachrichtigt. Zum Schluss holt der Kunde sein Auto beim Mechaniker ab.

2. Im Folgenden ist die Stimmenabgabe mittels eines E-Voting-Verfahrens beschrieben:



Abbildung 1: Der Wahlassistant von E-Vote - Anzeigen des Stimmzettels

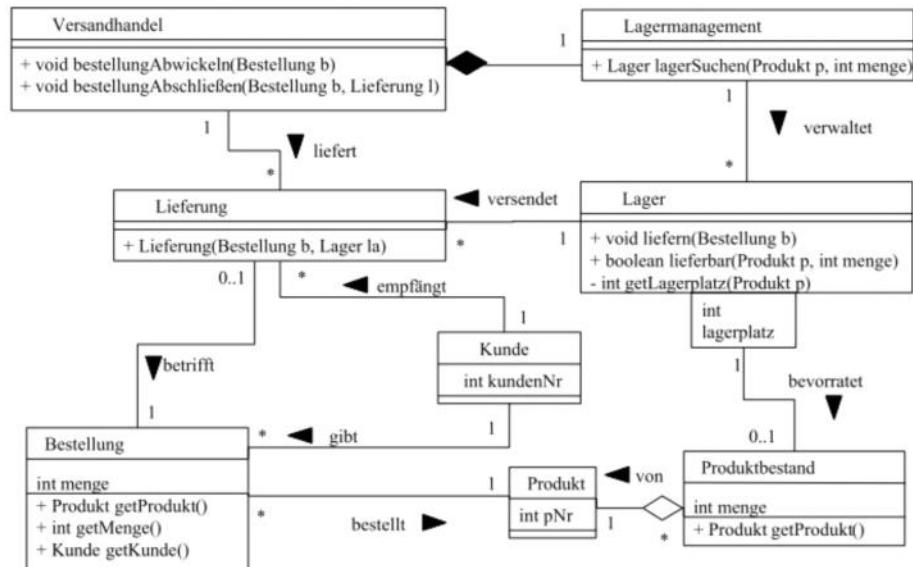
Um seine Stimme bei einer Wahl abzugeben, muss der Wähler den Wahlassistanten auf seinem lokalen Rechner aufrufen. Der Wahlassistant zeigt eine Reihe von Menüoptionen an. Der Wähler wählt die Menüoption "Stimme abgeben".

Der Wahlassistant fordert den Wähler zur Eingabe des Namens der Datei auf, in der die Registrierung gespeichert ist. Nachdem der Wahlassistant die Datei mit der Registrierung eingelesen hat, fordert er den Stimmzettel beim Wahlamt an. Das Wahlamt überprüft die Registrierung des Wählers beim Wählerverzeichnis und im Anschluss, ob der Wähler noch keinen Stimmzettel abgegeben hat. Ist dies der Fall, überträgt das Wahlamt den Stimmzettel für die entsprechende Wahl an den Wahlassistanten. Der Wahlassistant zeigt dem Wähler den Stimmzettel an (siehe Abbildung 1). Der Stimmzettel besteht aus mehreren Stimmoptionen, von denen der Wähler (je nach Wahl) eine oder mehrere ankreuzen kann.

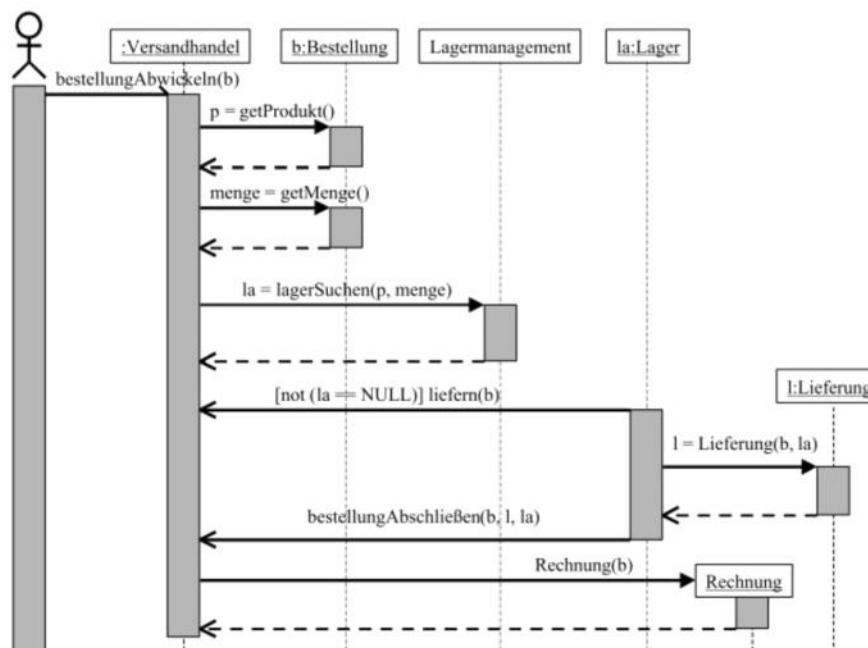
Nachdem der Wähler den Stimmzettel ausgefüllt hat, fordert der Wahlassistant ihn auf, den Wahlvorgang fortzusetzen und den Stimmzettel abzuschicken. Bestätigt der Wähler diese Aufforderung, verschlüsselt und signiert der Wahlassistant den Stimmzettel und sendet ihn an das Wahlamt, wo der Stimmzettel der Wahl zugeordnet und der Wähler als gewählt vermerkt wird. Danach wird eine Bestätigung der Stimmabgabe an den Wahlassistanten gesandt. Der Wähler hat nun seine Stimmabgabe für die Wahl abgeschlossen und kann den Wahlassistanten beenden.

Zeichnen Sie ein Sequenzdiagramm für den UseCase "Stimme abgeben". Betrachten Sie nur den Standardfall und keine alternativen Abläufe.

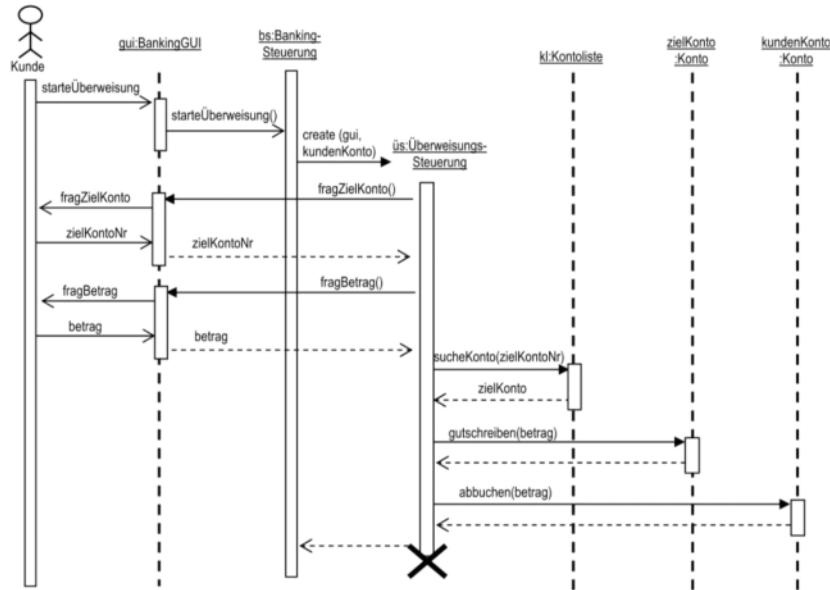
3. Gegeben sind folgendes Klassendiagramm:



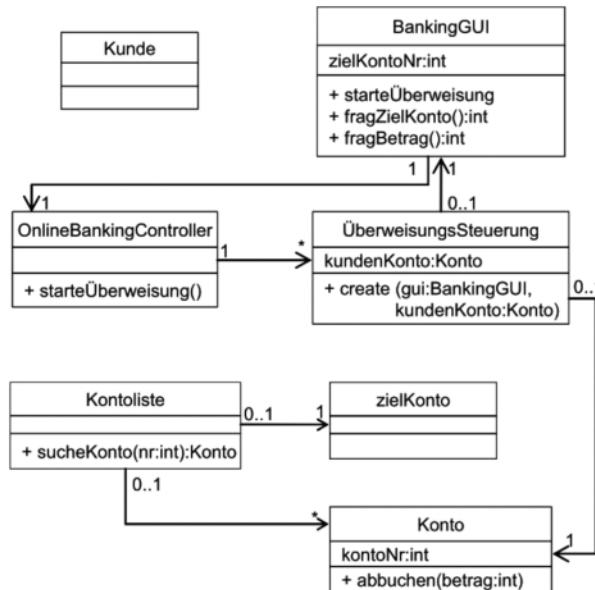
Überprüfen Sie, ob das unten dargestellte Sequenzdiagramm korrekte UML-Syntax verwendet und ob es konsistent zu dem Klassendiagramm ist. Markieren und korrigieren Sie erforderlichenfalls syntaktische Fehler und Inkonsistenzen im gegebenen Sequenzdiagramm. Begründen Sie Ihre Änderungen! (Hinweis: Sie müssen maximal 5 Korrekturen vornehmen.)



4. Gegeben sei das folgende Sequenzdiagramm aus der Feinanalyse der Produktfunktion "Überweisung ausführen":



Zu dem Sequenzdiagramm sei das folgende Analyseklassendiagramm gegeben.



Das Klassendiagramm soll nur die Dinge enthalten, die sich aus dem Sequenzdiagramm ableiten lassen. Tatsächlich enthält es aber zahlreiche Fehler und Inkonsistenzen zum Sequenzdiagramm. Markieren Sie jede fehlerhafte Stelle im Klassendiagramm und begründen Sie warum diese Stelle falsch ist.

5. Gegeben sei das folgende Szenario zur Beschreibung der Produktfunktion "Verkaufsabwicklung nach Versteigerung eines Artikels":

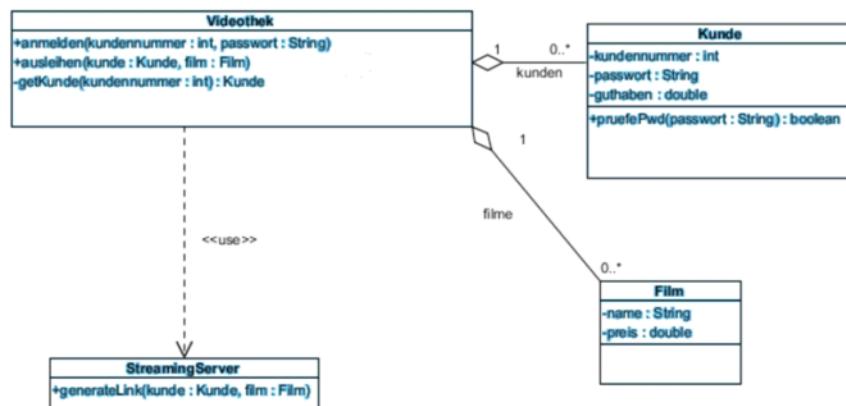
Schritt	Nutzer	Beschreibung der Aktivität
1		System teilt Verkäufer erzielten Preis mit.
2		System teilt Käufer zu zahlenden Preis mit.
3		System fordert Kontonummer vom Verkäufer an.
4	Verkäufer	Verkäufer teilt System seine Kontonummer mit.
5		System speichert Kontonummer des Verkäufers für spätere Auktionen ab.
6		System teilt Kontonummer dem Käufer zwecks Überweisung mit.

Erstellen Sie zu diesem Szenario ein Sequenzdiagramm mit den Interaktionen zwischen den Benutzern und dem System.

6. Sie sollen für die Online-Videothek den Vorgang des "Filmausleihens" modellieren.

Erstellen Sie dazu ein Sequenzdiagramm. Zur Vereinfachung können Sie davon ausgehen, dass sich das Mitglied bereits auf der Seite des gewünschten Films befindet. Wählen Sie geeignete Namen für die Elemente Ihres Diagramms.

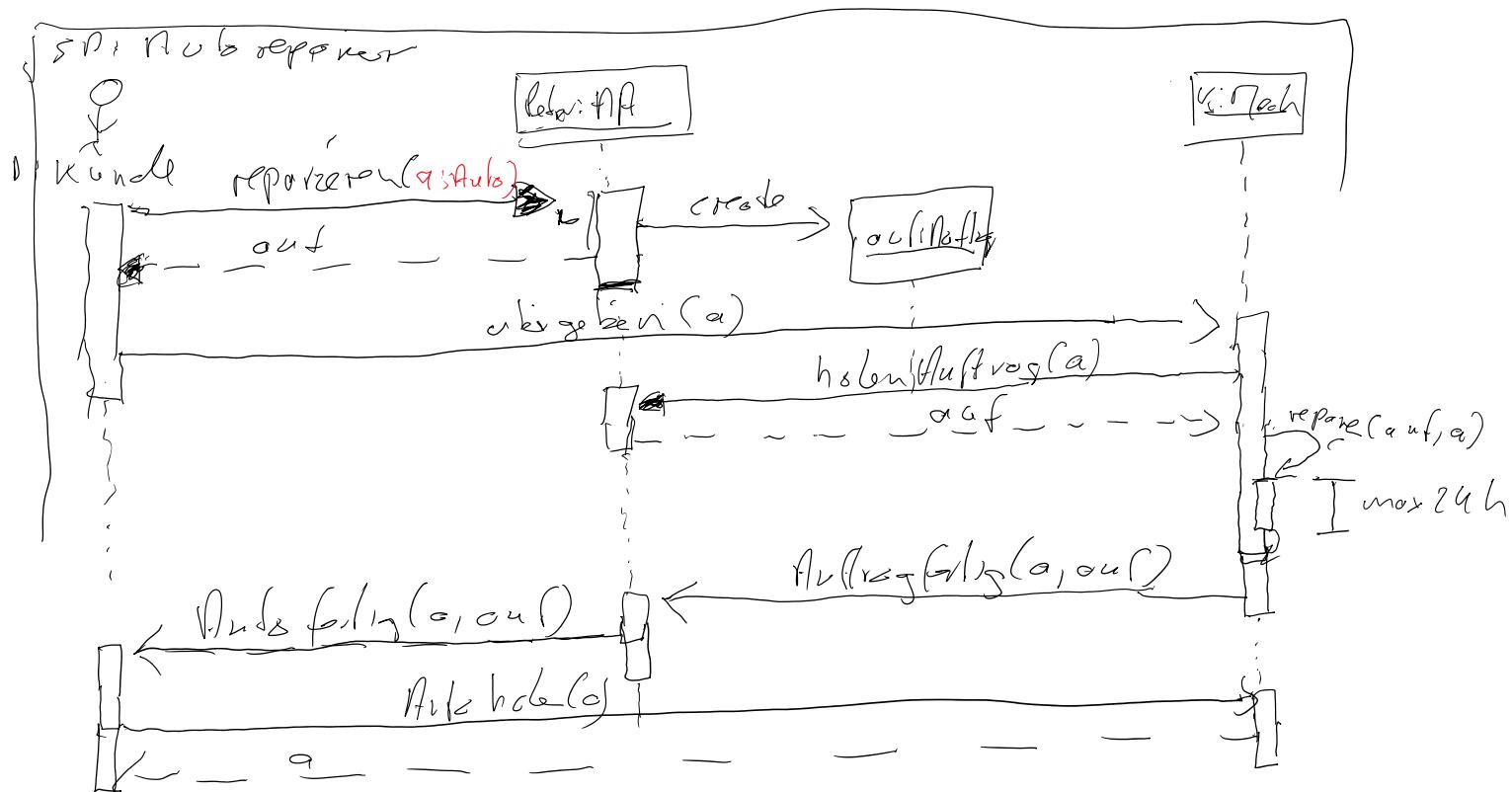
Folgendes Klassendiagramm ist zur Zeit vorhanden und sollte von Ihnen genutzt bzw. erweitert werden.



- Um den Film auszuleihen, muss das Mitglied sich zunächst erfolgreich anmelden. Dies wurde im Sequenzdiagramm SD\_Login bereits beschrieben..
- War die Anmeldung erfolgreich, versucht die Person den Film auszuleihen. -
- Die Videothek berechnet zuerst, ob das Guthaben reicht um den Film zu bezahlen.
- Reicht das Guthaben nicht aus, wird stattdessen eine Aufforderung zum Auffüllen des Guthabens angezeigt.
- Falls das aktuelle Guthaben des Mitglieds ausreicht, veranlasst die Videothek einen Streaming-Server einen Link für den Film zu generieren.
- Die Videothek zeigt dem Benutzer den Link an, unter dem der Film zugreifbar ist.

1. Erstellen Sie für das Szenario des Abwickelns einer Autoreparatur ein Sequenzdiagramm.

Zunächst beauftragt der Kunde die Reparatur und übergibt danach dem Mechaniker das Auto. Dieser holt den Auftrag von der Auftragsannahme ein. Nachdem der Mechaniker die Reparatur (innerhalb von 24 Stunden) beendet hat, meldet er dies der Auftragsannahme, die daraufhin den Kunden benachrichtigt. Zum Schluss holt der Kunde sein Auto beim Mechaniker ab.



# Übungen

Sonntag, 13. September 2020 09:23

b) Der Produktionsleiter der SoftDrink AG teilt dem Anwendungsentwickler der rapidPack GmbH mit, dass er für die Verpackungsmaschine ein neues Steuerprogramm entwickeln soll. Der Anwendungsentwickler erstellt das Steuerprogramm und weist den Maschinenführer an, das neue Steuerprogramm mittels eines Probelaufs zu testen. Der Maschinenführer führt den Probelauf durch und bestätigt dem Anwendungsentwickler die Funktionsfähigkeit des Steuerprogramms. Der Anwendungsentwickler wiederum meldet die Produktionsbereitschaft der Maschine dem Produktionsleiter zurück.

Korrekturrand

Stellen Sie die beschriebene Vorgehensweise in einem UML-Sequenzdiagramm dar.

20 Punkte

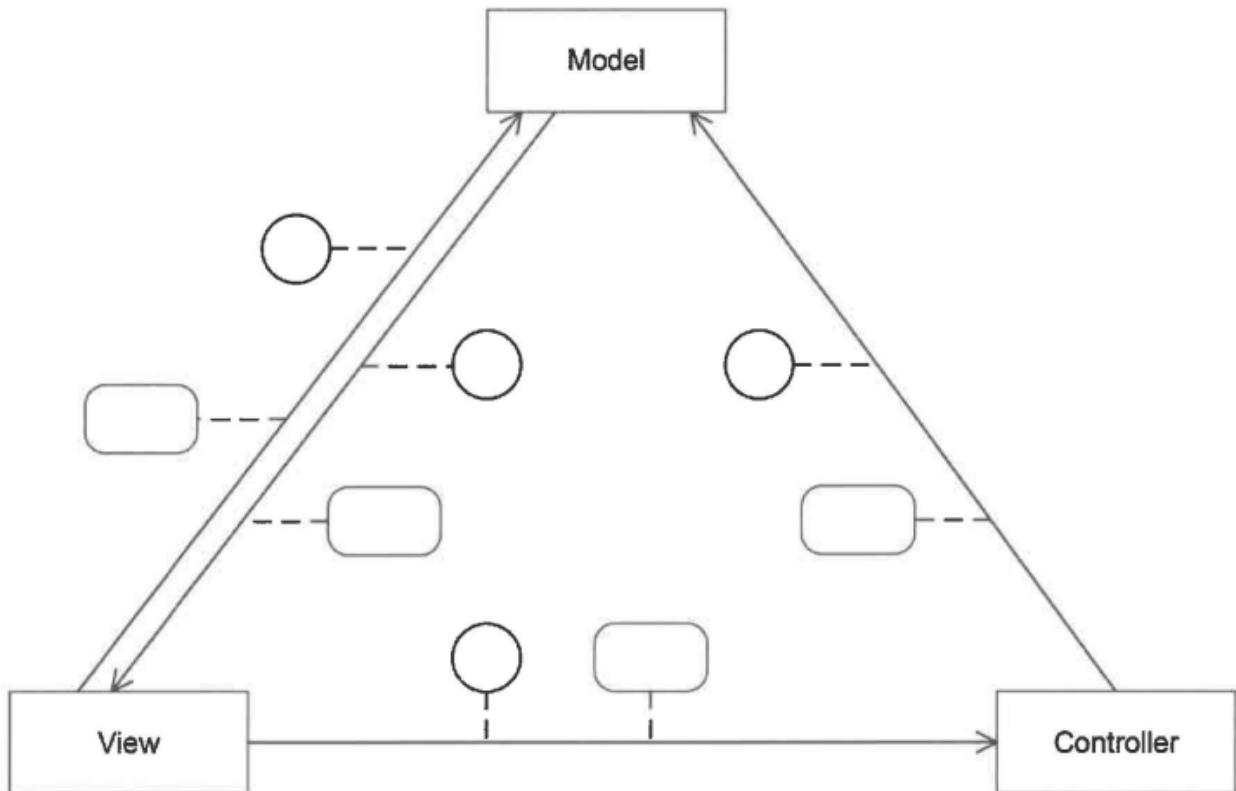
:Produktionsleiter

### 3. Handlungsschritt (25 Punkte)

Die Patientendaten (z. B. Blutdruck, Körpertemperatur) sollen im zeitlichen Verlauf in verschiedenen Ansichten (z. B. Tabelle, Säulendiagramm) dargestellt werden. Damit die Implementierung für zukünftige Erweiterungen offenbleibt, schlägt ein Teamkollege die Realisierung mit dem Model-View-Controller-Pattern (MVC-Muster) vor.

a) Für das Verständnis des MVC-Musters soll eine Reihenfolge der Benachrichtigungen angegeben werden.

Ergänzen Sie im folgenden Diagramm die entsprechenden Ziffern (Reihenfolge) in den Kreisen und die Aktivitäten durch Zuweisung der entsprechenden Buchstaben. 4 Punkte

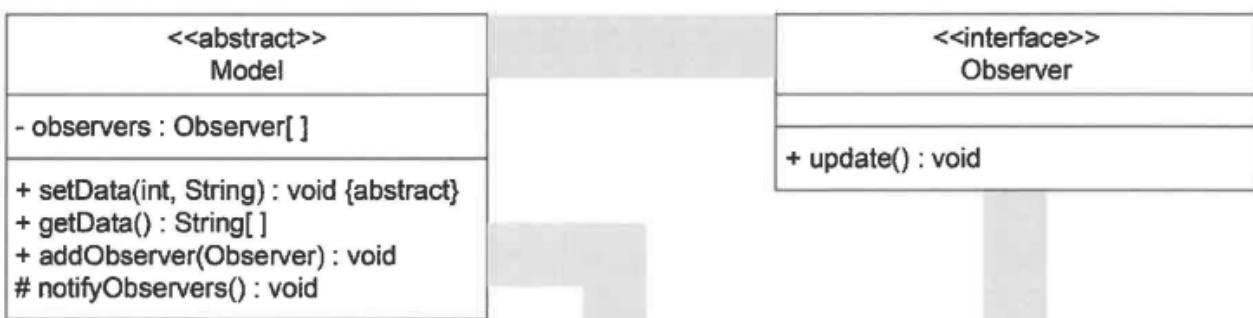


#### Aktivitäten

A	Controller fordert Model zu Zustandsänderung auf
B	Model informiert View über Zustandsänderung
C	View fordert die geänderten Daten vom Model zur Ansicht für den Benutzer an
D	View informiert Controller über Benutzereingabe

ba) Model und View werden häufig über das Observer-Pattern realisiert. Dabei erbt die konkrete Klasse „PatientModel“ von der abstrakten Klasse „Model“. Die Klasse „TableView“ implementiert das Interface „Observer“.

Ergänzen Sie im vorliegenden UML-Klassendiagramm Methoden und Klassenbeziehungen. 6 Punkte



```
+ addObserver(Observer) : void
# notifyObservers() : void
```

PatientModel

TableView

- model : Model

+ TableView(model : Model)

+ display() : void

- bb) Der Konstruktor von „TableView“ initialisiert seine Modelreferenz mit dem übergebenen Modelobjekt und registriert sich mit der Methode „addObserver“ als Observer.

Geben Sie den Konstruktor in Pseudocode an.

3 Punkte

- 
- 
- 
- 
- bc) Die Methode „notifyObservers“ sorgt dafür, dass alle registrierten Observer die Methode „update“ ausführen.

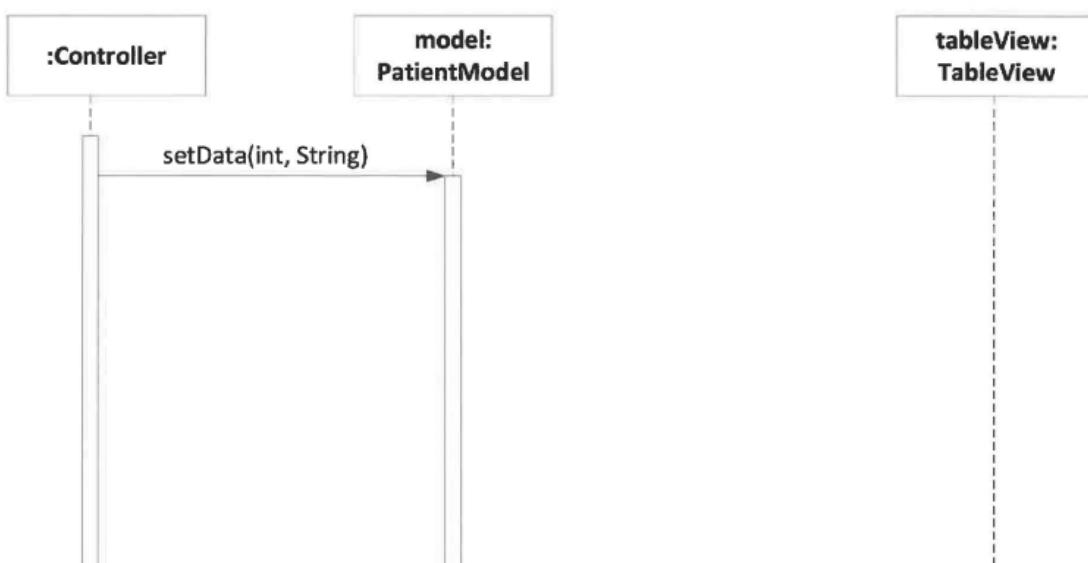
Geben Sie die Methode in Pseudocode an.

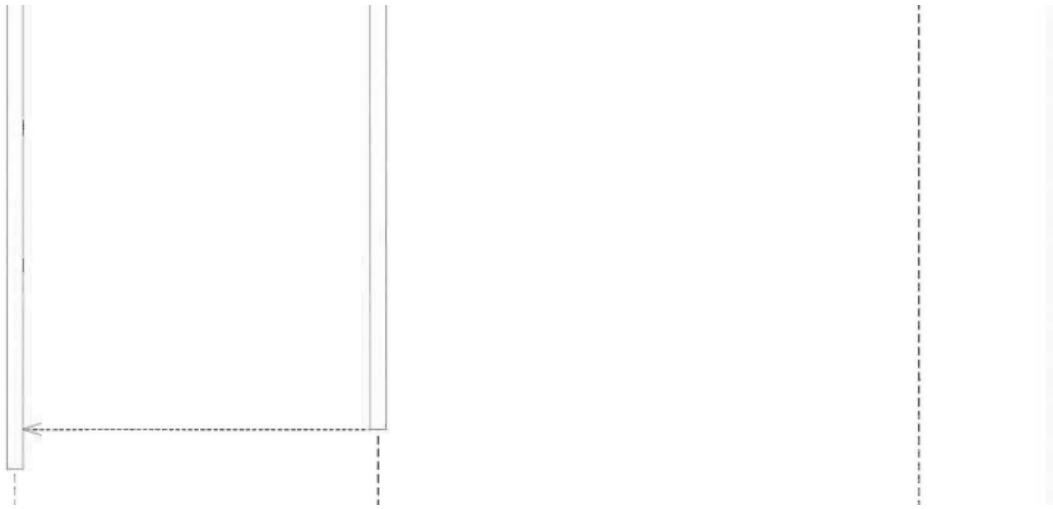
3 Punkte

- 
- 
- 
- 
- bd) Sobald ein Benutzer mit der View interagiert, ruft der entsprechende Controller „setData“ auf dem Model auf. Die Methode „setData“ aktualisiert die Daten und startet anschließend „notifyObservers“. Die Methode „update“ ruft „getData“ auf und sorgt abschließend durch Aufruf von „display“ dafür, dass die geänderten Daten des zurückgegebenen String-Arrays auf „tableView“ dargestellt werden.

Ergänzen Sie das gegebene Sequenzdiagramm entsprechend der Vorgaben.

7 Punkte





### Fortsetzung 3. Handlungsschritt

Ein Kollege schlägt vor, anstatt der Observer-Musters zur Aktualisierung des Views Datenbindung (Data Binding) einzusetzen.

- c) Erläutern Sie den Begriff.

2 Punkte

---

---

---