Nachschub: Systemsicherheit, Deployment & Co.

- https://www.op-marburg.de/Mehr/Welt/Panorama/Panne-bei-McDonald-s-Gewinnspiel-Kunden-gewinnen-versehentlich-100.000-Euro?fbclid=IwAR2Dkcr9EEqcTSZFuYw4gRFLG3BjKi58tqFAOf13cRO_nmwOIlfDqRy4JT8

2.4 Domänenmodellierung

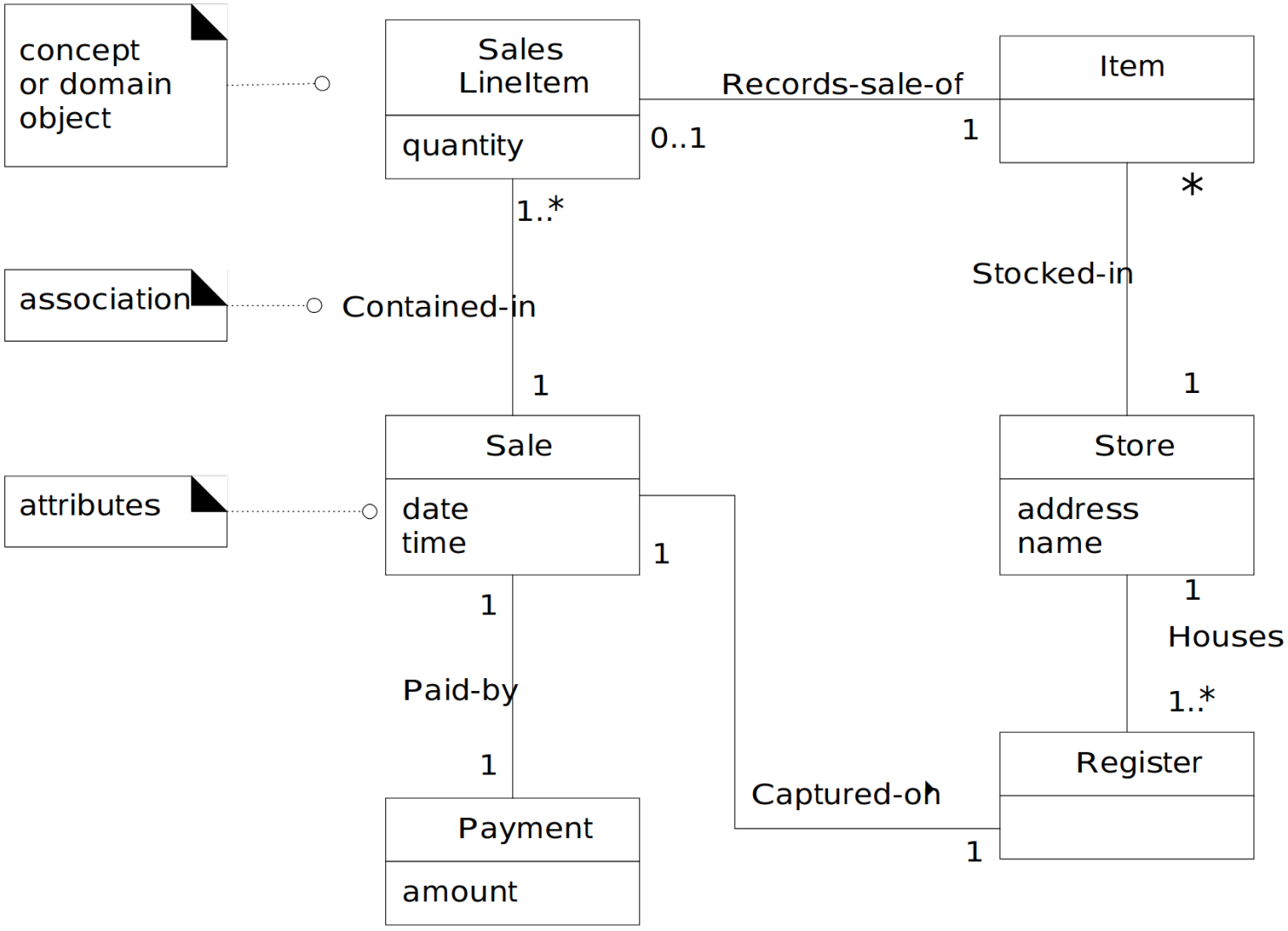
- Ein Domänenmodell veranschaulicht (die für Modellierer) aussagekräftigen konzeptionellen Klassen in einer Anwendungsdomäne. Es ist eine Darstellung von realen konzeptionellen Klassen, nicht von Softwarekomponenten. Es handelt sich *nicht* um eine Reihe von Diagrammen, die Softwareklassen oder Softwareobjekte mit Verantwortlichkeiten beschreiben.
  → Domänenmodelle sind weder Daten- noch Designmodelle

- Mit Hilfe der UML-Notation wird ein Domänenmodell mit einem Satz von Klassendiagrammen dargestellt, in denen keine Operationen/Methoden definiert sind. Es kann zeigen:
  - Domänenobjekte oder konzeptionelle Klassen
  - Assoziationen zwischen konzeptuellen Klassen
  - Attribute von konzeptionellen Klassen

- Das Domänenmodell kann als visuelles Wörterbuch der wesentlichen Abstraktionen, des Domänenvokabulars und des Informationsinhalts der Domäne betrachtet werden.

- Der Unified Process (UP) definiert ein sogenanntes Domänenmodell, das durch die UML-Notation veranschaulicht wird. In der offiziellen UML-Dokumentation ist jedoch kein Begriff "Domain Model" zu finden. Dies deutet auf eine wichtige Erkenntnis hin:
  → Die UML beschreibt lediglich Rohdiagrammtypen wie Klassendiagramme und Sequenzdiagramme. Es wird keine Methode oder Modellierungsperspektive auf diese angewendet. Vielmehr wendet ein Prozess (wie etwa der Unified Process) die reine UML im Kontext von methodisch definierten Modellen an.

2.4 Domänenmodellierung

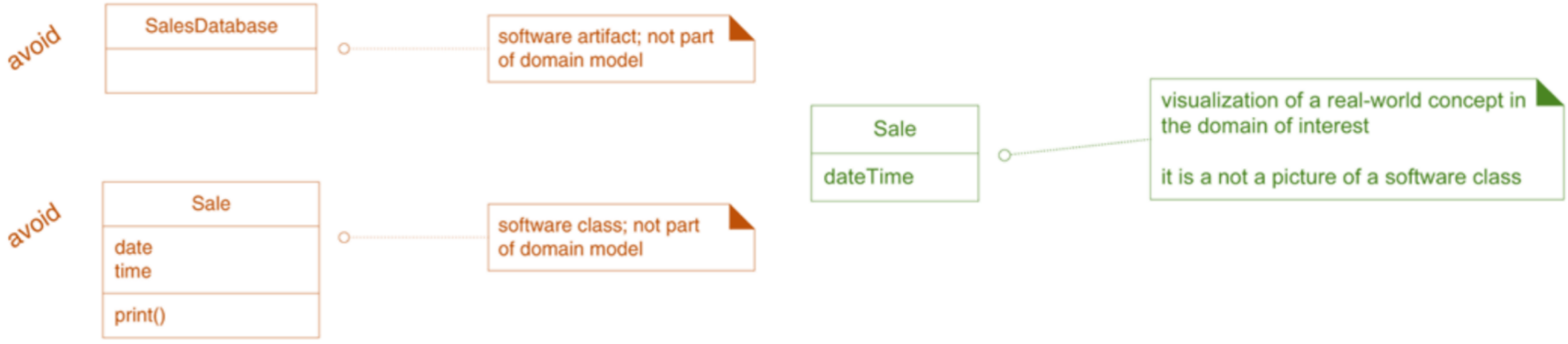


2.4 Domänenmodellierung

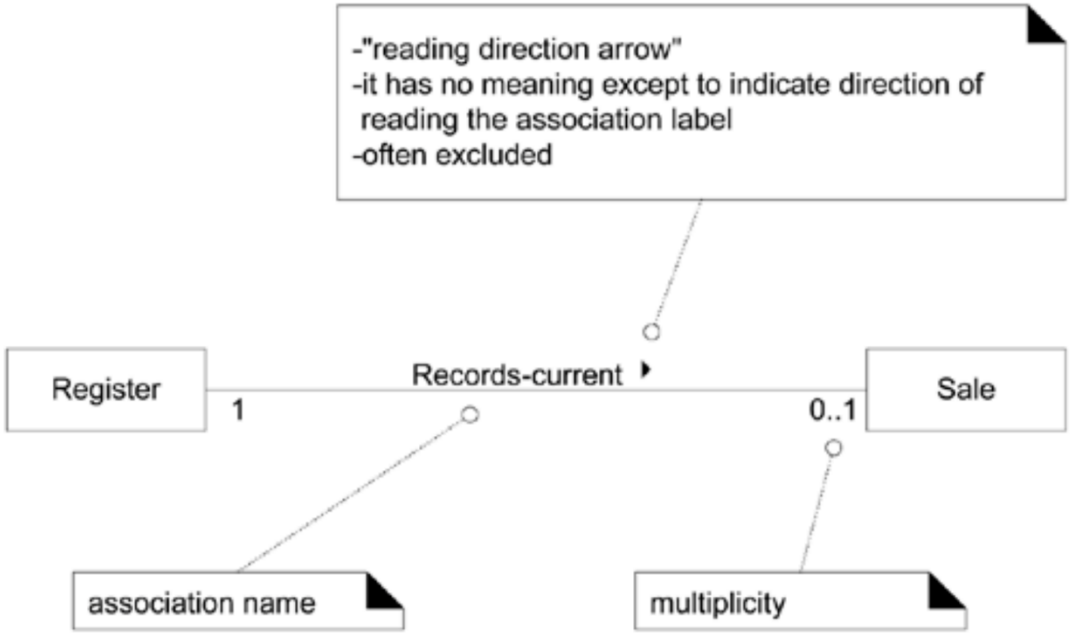
- Mit den folgenden Schritten kann (iterativ!) ein Domänenmodell erstellt werden:
  1. Anhand der Anforderungen/Stories können wichtige Substantive ermittelt werden
  2. Diese werden in einem Domänenmodell eingezeichnet
  3. Verknüpfungen dokumentieren die Beziehungen zueinander
  4. Attribute dienen der Angabe von notwendigen Informationen
- Erstellen Sie ein Domänenmodell im Sinne der Arbeitsweise eines Kartographen:
  - Die vorhandenen Namen des Gebietes wiederverwenden
  - Irrelevante Merkmale ausschließen
  - Keine Dinge hinzufügen, die nicht vorhanden sind.
- Wenn wir nicht an eine konzeptuelle Klasse X als Zahl oder Text in der realen Welt denken, ist X wahrscheinlich eine konzeptuelle Klasse und kein Attribut.
  - Im Zweifelsfall ist es immer ein Konzept, kein Attribut.
  - Attribute sollten in einem Domänenmodell selten sein.
  - Konzeptionelle Klassen werden mit einer Assoziation verknüpft, nicht mit einem Attribut.

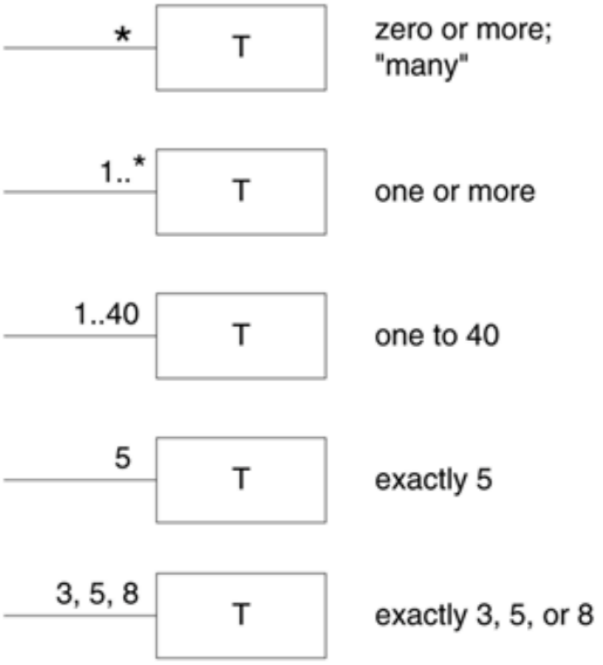
2.4 Domänenmodellierung

- Domänendiagramme bilden die reale Welt ab, sie beschreiben keine Softwarekomponenten! Folgende Elemente gehören daher nicht in ein Domänendiagramm:
  - Software-Dinge wie “Fenster” oder “Datenbank”, sofern die zu modellierende Domäne nicht von diesen Konzepten handelt, z.B. eine grafische Benutzeroberfläche.
  - Verantwortlichkeiten und Methoden
- Eine Ausnahme für Verantwortlichkeiten ergibt sich nur, wenn eine menschliche Rolle (z.B. ein Kassierer) in der Domäne wichtig ist. In diesem Fall kann dessen Verantwortlichkeit in dem Modell dargestellt werden.

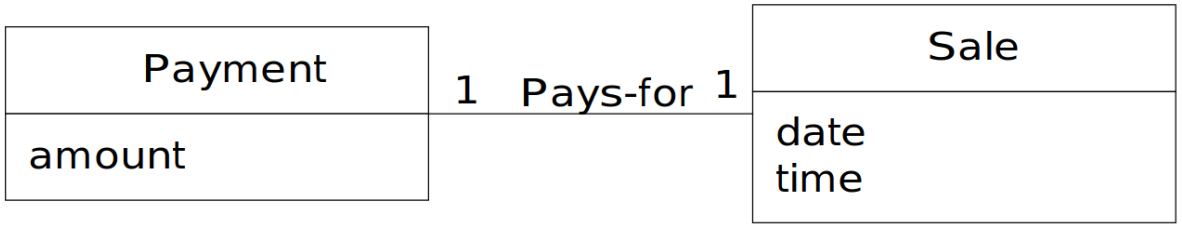


2.4 Domänenmodellierung



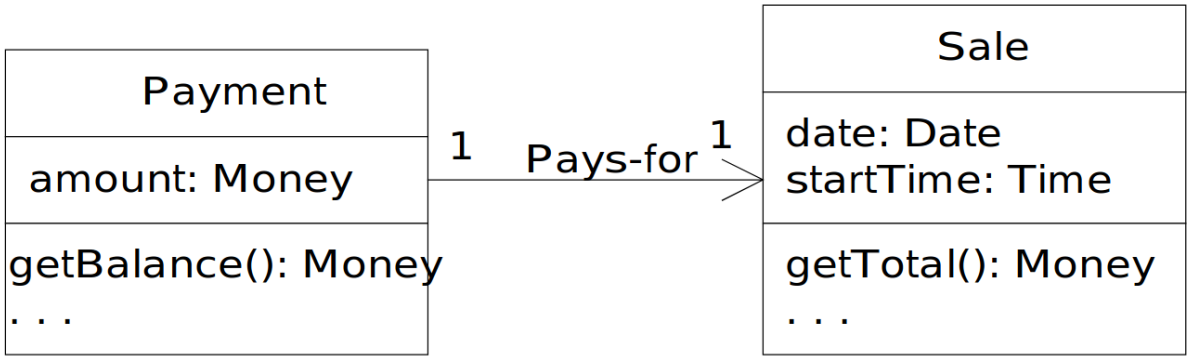


2.4 Domänenmodellierung



UP Domain Model

Raw UML class diagra notation used in an essential model visualizing real-world concepts.



UP Design Model

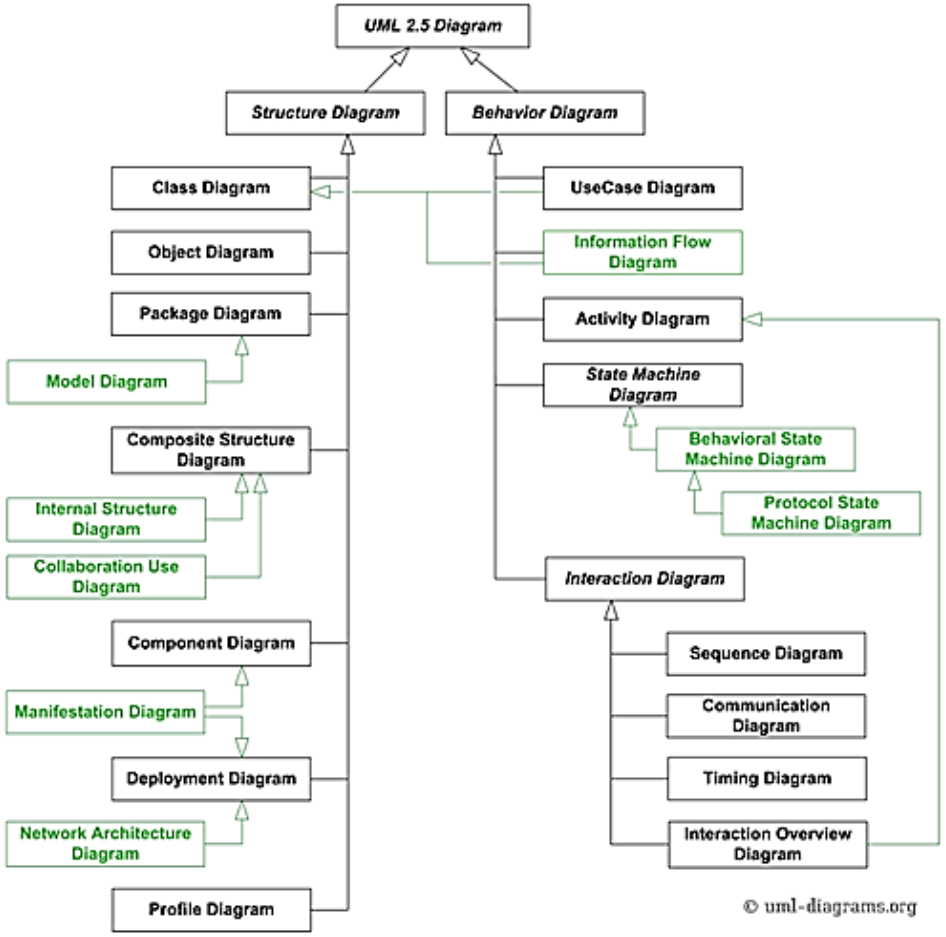
Raw UML class diagra notation used in a specification model visualizing software components.

2.5 UML-Überblick

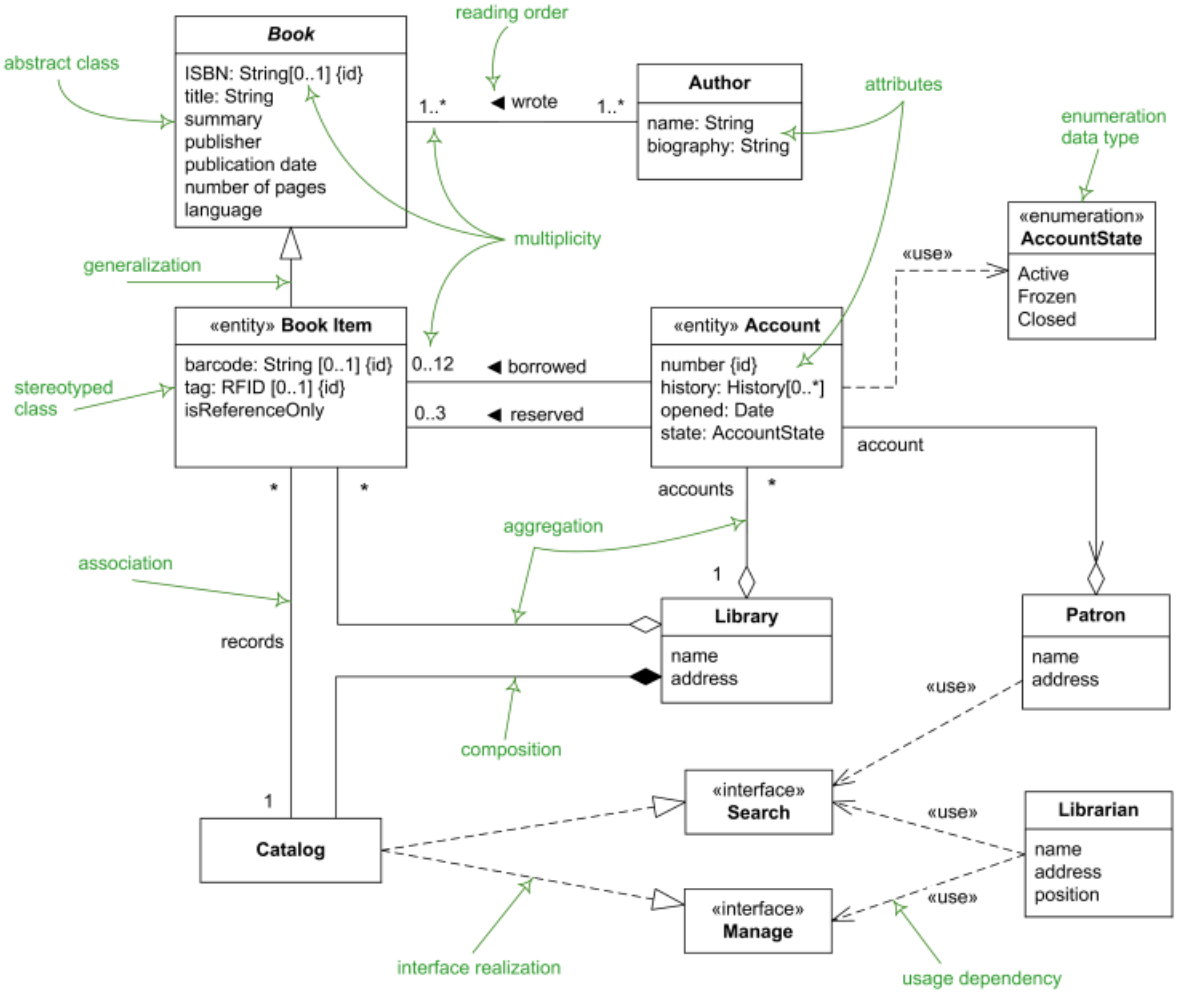
- Unified Modeling Language
  - Entstanden Ende der 90er
  - Aktuelle Version 2.5
  - Allgemein verwendbare (Modellierungs-)Sprache → General-Purpose Language
- **Standardisierte** Modellierungssprache um statische und dynamische Aspekte beliebiger Anwendungsgebiete unter Zuhilfenahme von Diagrammen abstrakt zu beschreiben
  - →Keine Beschränkung auf Softwareentwicklung
  - →Eindeutige Notationselemente (Bestandteile von Diagrammen)
  - →Bessere Verständlichkeit durch abstrakte Notationselementen und visuelle Darstellung
  - →Technologieunabhängig
  - →Können in praktisch jedem Vorgehensmodell verwendet werden

2.5 UML-Überblick

- Mit UML können in Struktur- und Verhaltensdiagramme erstellt werden
  - Strukturdiagramme
    - Klassendiagramm
    - Objektdiagramm
    - Kompositionsstrukturdiagramm
    - Komponentendiagramm
    - Paketdiagramm
    - …
  - Verhaltensdiagramme
    - Anwendungsfalldiagramm
    - Aktivitätsdiagramm
    - Zustandsdiagramm
    - Sequenzdiagramm
    - …

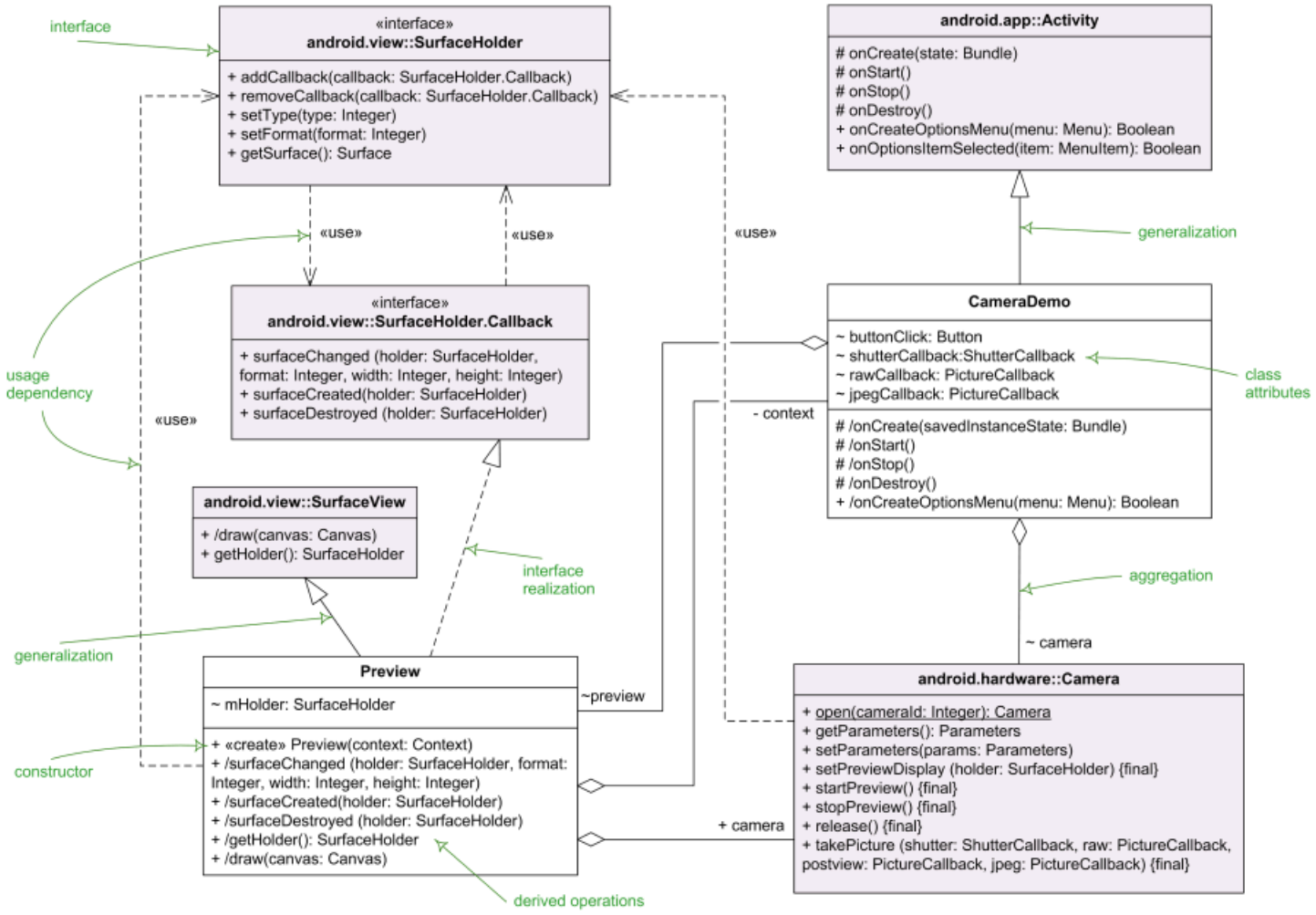


2.5 UML-Überblick – Klassendiagramm (Domänenmodell)



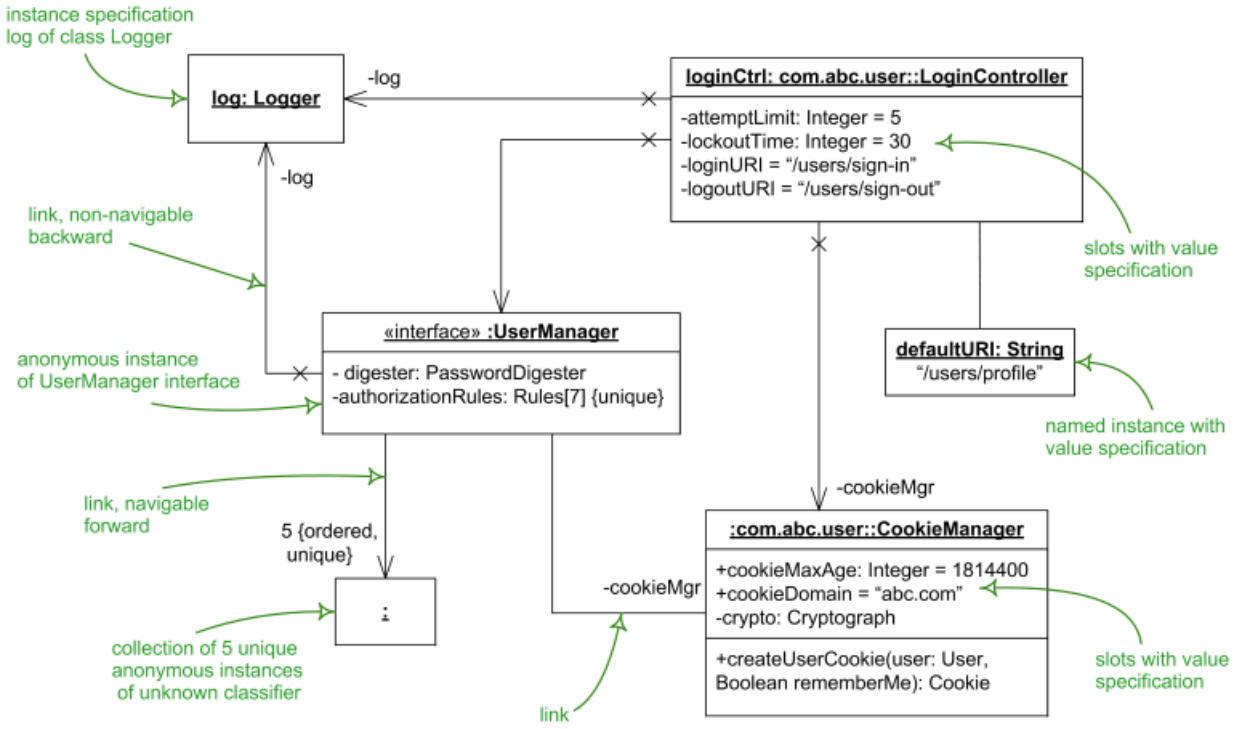
Technologieunabhängig!

2.5 UML-Überblick – Klassendiagramm

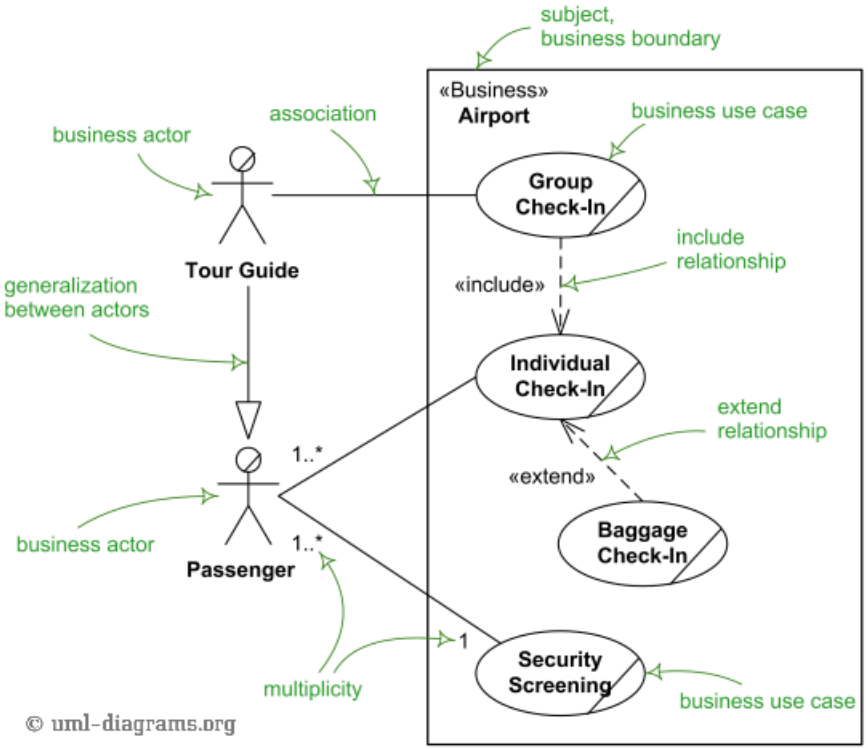


Technologieabhängig!

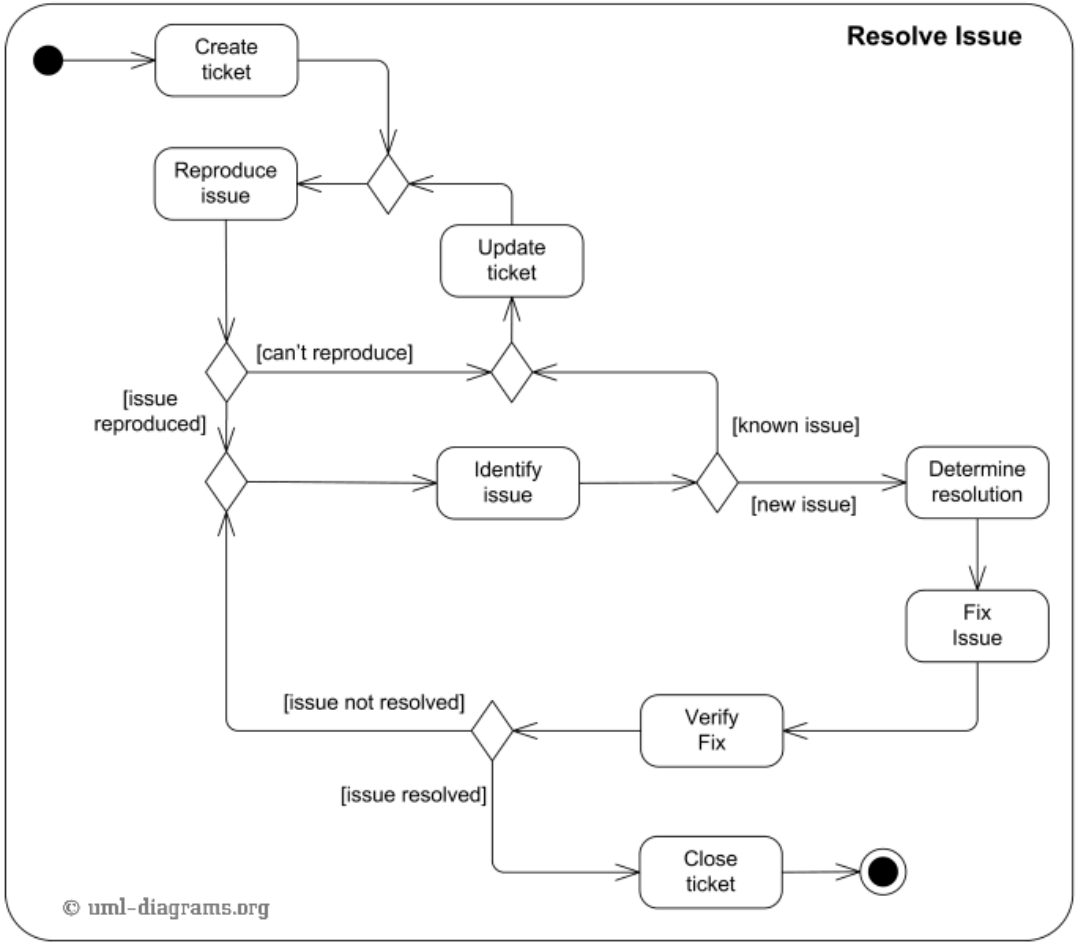
2.5 UML-Überblick – Objektdiagramm



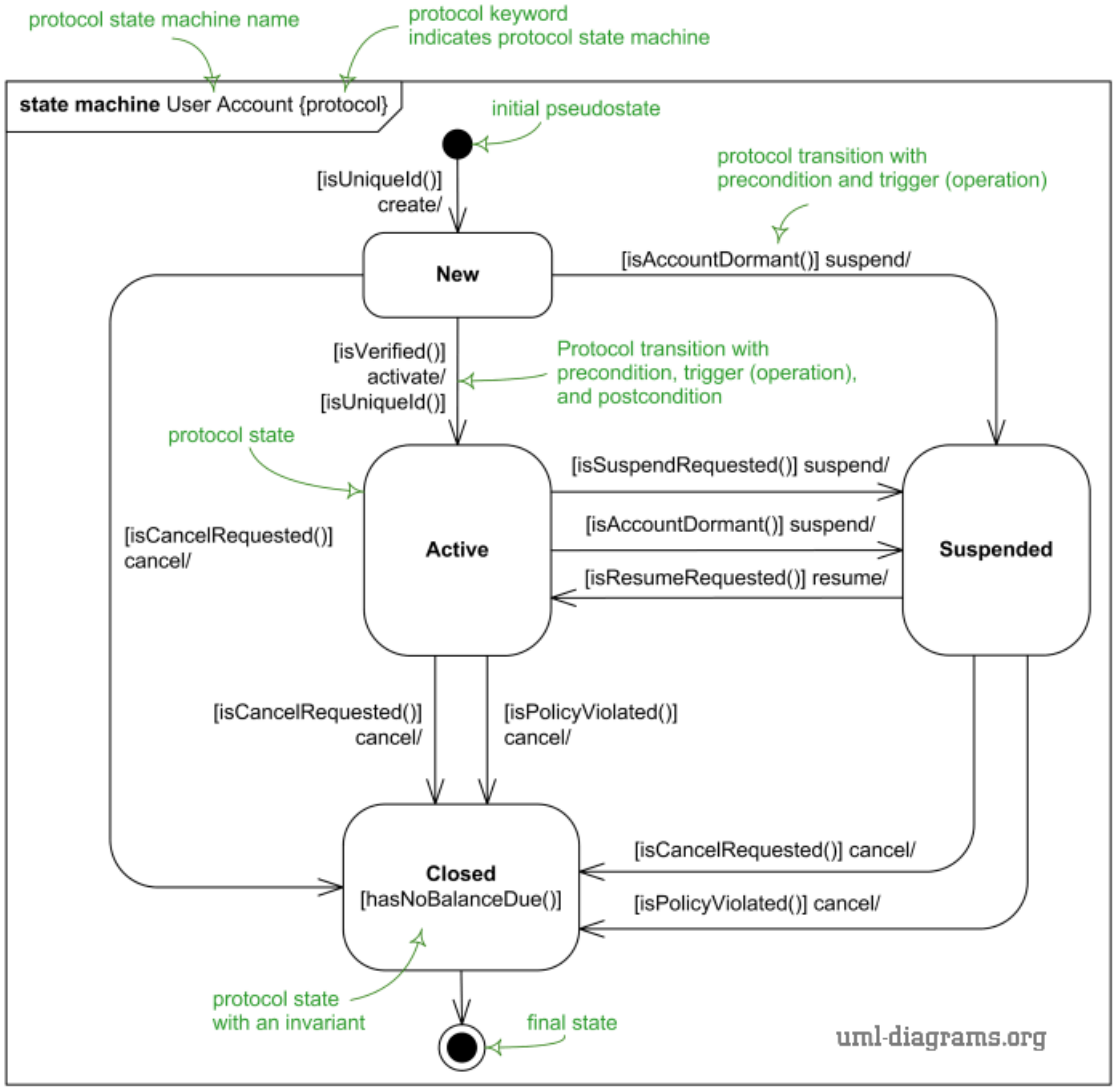
2.5 UML-Überblick – Anwendungsfalldiagramm



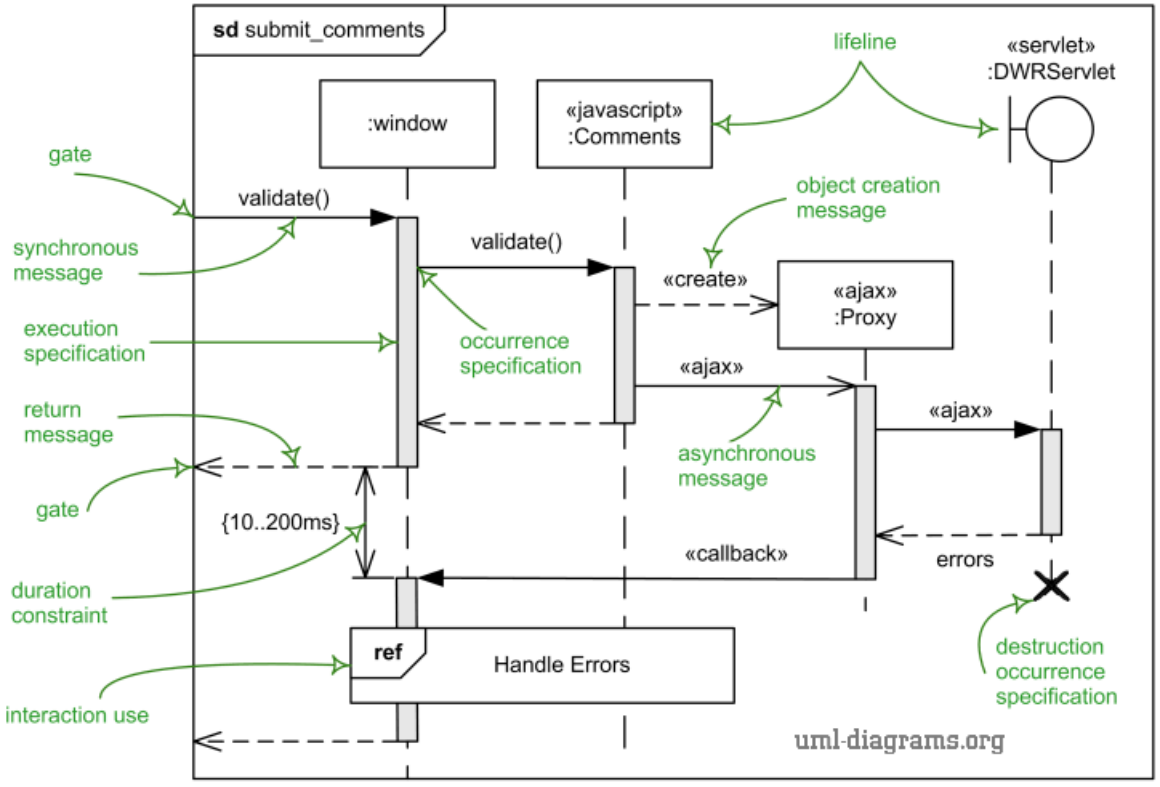
2.5 UML-Überblick – Aktivitätsdiagramm



2.5 UML-Überblick – Zustandsdiagramm



2.5 UML-Überblick – Sequenzdiagramm



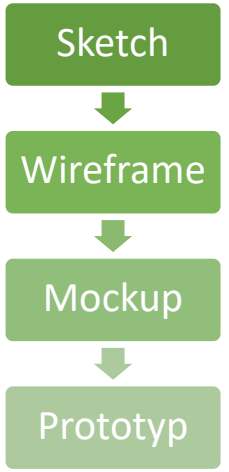
2.6 Projekt: MyCargonaut

- Selbstständige Umsetzung der Plattform „MyCargonaut“
  - Vermittlungen von Mitfahr- und Lieferdienstleistungen
  - „Ich suche jemand, der mein Klavier von Köln nach Berlin bringt.“
  - „Ich fahre von Köln nach Berlin und habe eine große Ladefläche.“
- Schlüssel-Use-Cases:
  - Ich suche Fahrer/Spediteur
  - Ich biete Fahrten/Ladefläche an
  - Abwicklung/Bezahlung
- Features
  - Registrierung von „Cargonauten“ (Nutzern):
    - Name, Geburtsdatum, Profilbild, E-Mail, Passwort
  - Profilansicht
    - Inkl. Bewertung von anderen Cargonauten
  - Fahrzeugverwaltung
  - Angebot/Gesuch erstellen
  - Tracking
  - Bewertung nach Abschluss der Fahrt

Designed by LuM4x Enterprise Design

2.6 Projekt: MyCargonaut

- Freie Wahl der Technologien: Programmiersprachen, Frameworks, Tools usw.
- Arbeiten nach Softwaretechnik-Standards (OOAD, agile Entwicklung, Continuous Delivery, Software-Patterns usw.)
- Iteratives Vorgehen in 4er-Gruppen:
  - **1. Sprint:** Domänendiagramm (UML), Wireframes, Technologieauswahl, Mockup, Infrastruktur (Git, CI usw.)
  - **2. Sprint:** Klassenentwurf, Teststrategie, erste Version (Prototyp)
  - **3. Sprint:** Kundenpräsentation der fertigen Version
- **Abschlusspräsentation am 06.07.2020**
- Abnahmekriterien:
  - Dokumentation (Infrastruktur, UML-Diagramme, Teststrategie, Mockups usw.)
  - Akzeptanztests in der CD-Infrastruktur
  - Lauffähige Software gemäß Definition-of-Done



Ende VL 8