University of Applied Sciences and Arts Northwestern Switzerland
School of Business

Betriebliche Informationssysteme

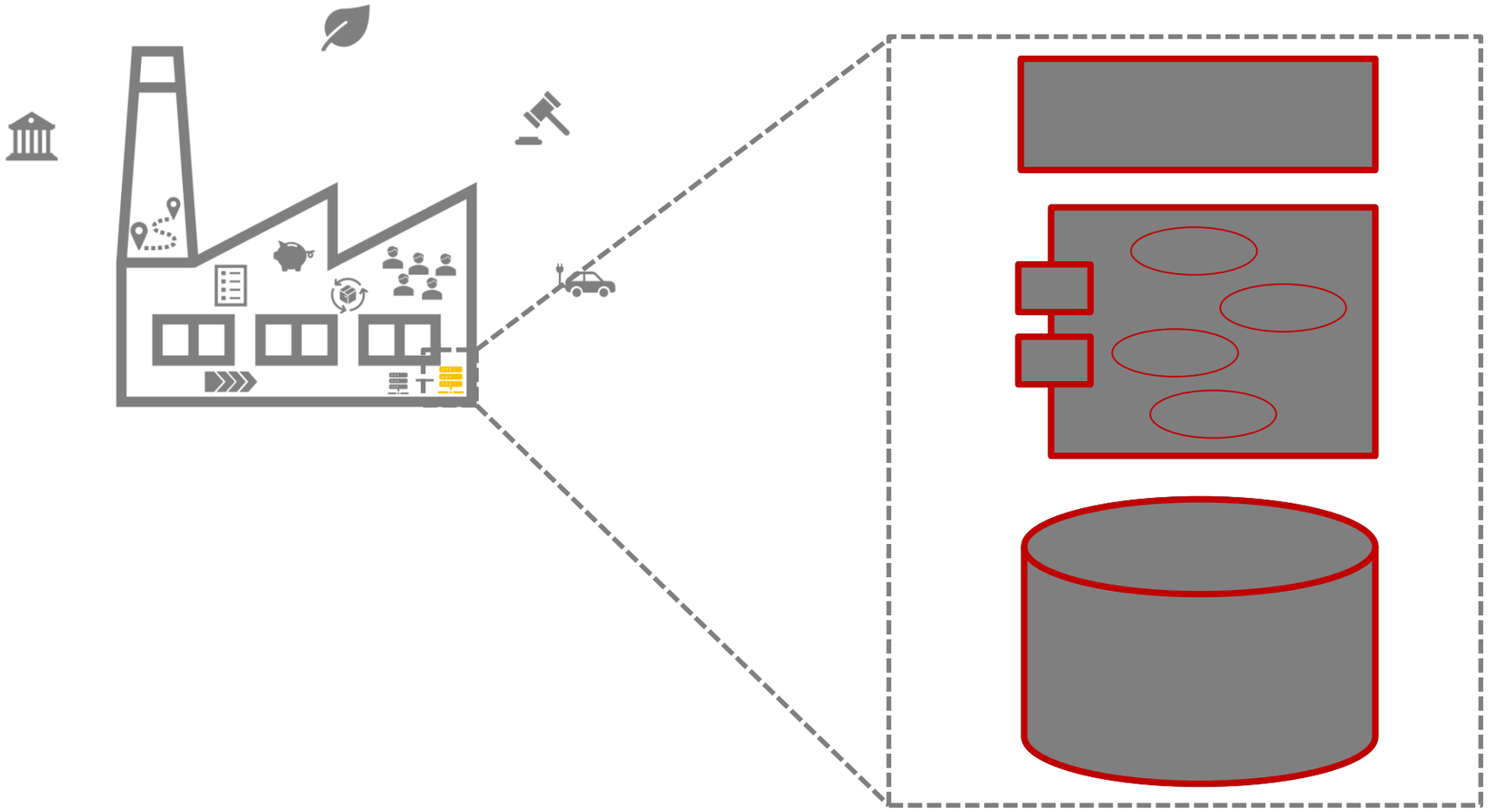
L7 – Systemdesign

L7 – Systemdesign

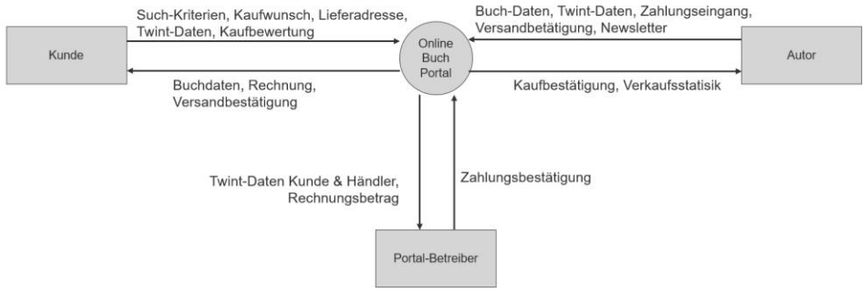
„Wie kann ein Systemverhalten sowie dessen -komponenten dokumentiert werden?“

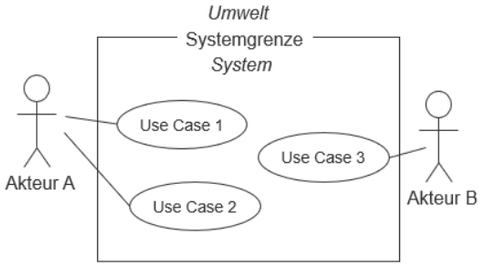
- **Zusammenhang zu Requirements Engineering & Systemarchitektur verstehen**
- **Einfache UML Klassendiagramme & Sequenzdiagramme modellieren können**
- **Einblick in Referenzarchitekturen bzw. –designs und Notation von Cloud-Anbietern erhalten**

Big Picture



Repetition „Requirements Engineering“





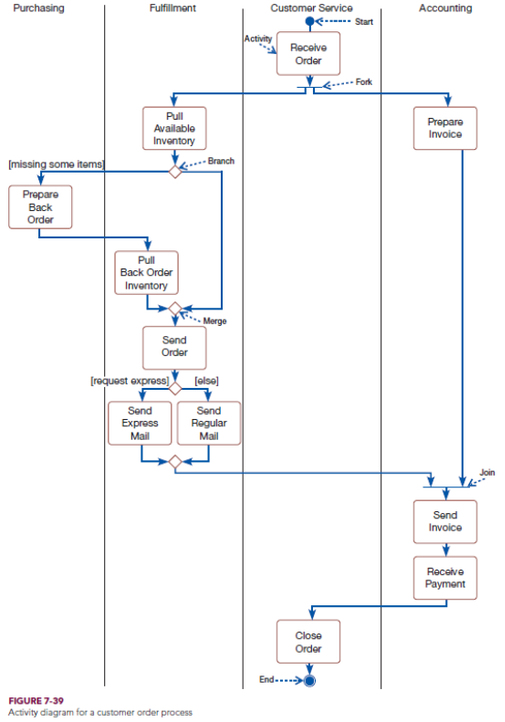
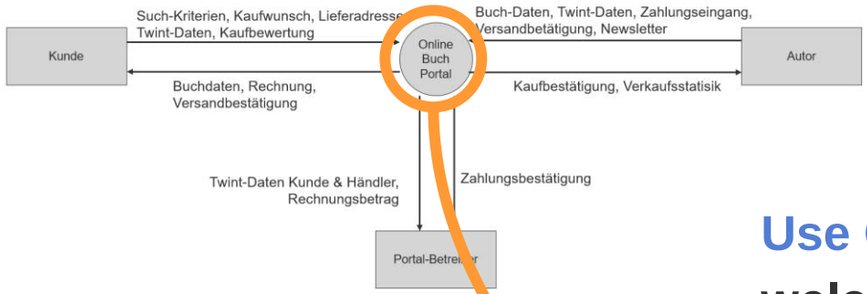


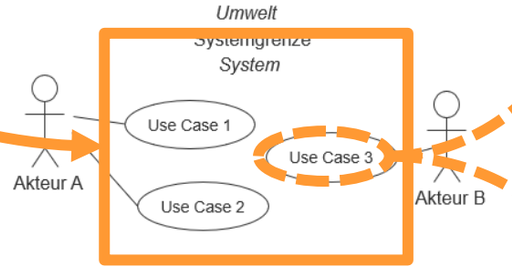
FIGURE 7-39
Activity diagram for a customer order process

Repetition „Requirements Engineering“

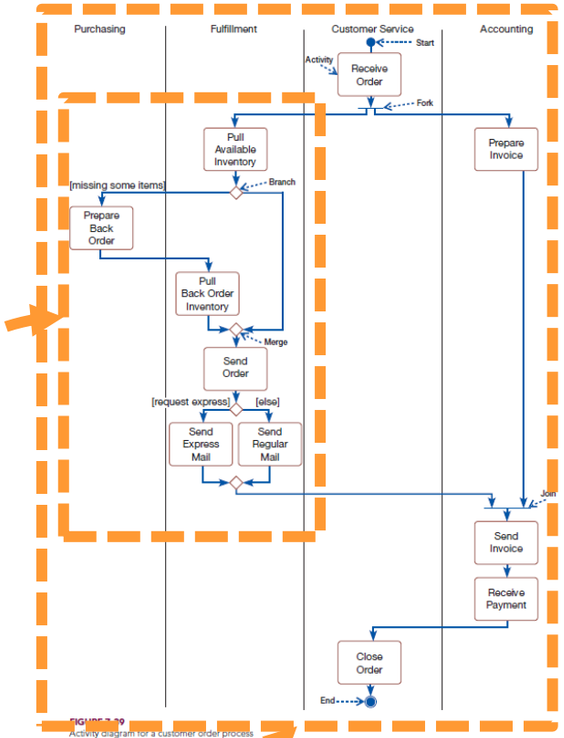
Kontext Diagramm: Zeigt **welche externen Systeme und Akteure** mit dem System interagieren.



Use Case Diagramm: detailliert, **welche Aktionen die Akteure im System durchführen** können.

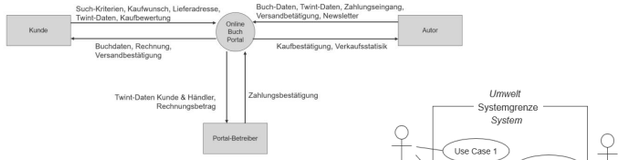


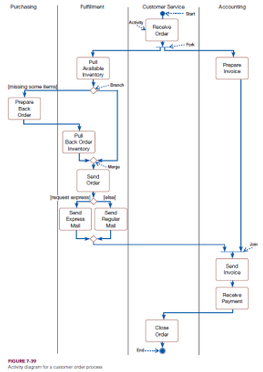
Aktivitäts-Diagramm: Zeigt den **Ablauf** der Aktionen der Use Cases.



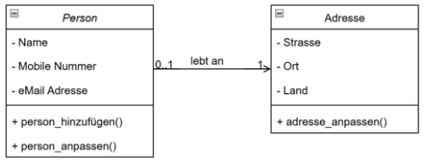
Von Was zu Wie

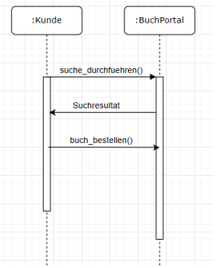
Mittels dem Kontextdiagramm, Use-Case-Diagramm, Aktivitätsdiagramm (**Anforderungsanalyse**) ist beschrieben, **was** das System tun soll.





Diese Information fliesst in die **Systemdesign:** Klassendiagramm, Sequenzdiagramm beschreiben, **wie** es technisch umgesetzt wird.





Das Klassendiagramm beschreibt die statische Struktur…

…während das Sequenzdiagramm zeigt, wie diese Klassen zur Laufzeit interagieren.

Systemdesign?

Systemdesign bezieht sich auf die **spezifische Ausarbeitung** und Strukturierung der einzelnen Komponenten eines Systems **sowie ihrer Interaktionen**.

Fokus von Klassendiagramm, Sequenzdiagramm

Das Systemdesign geht sehr ins Detail und legt fest, wie jede Komponente implementiert wird. Es enthält **Entscheidungen über Programmiersprachen, Frameworks, Datenbanken**, Schnittstellen und andere technische Aspekte. Es enthält detaillierte Dokumentation zur Unterstützung der Entwickler während der Implementierung.

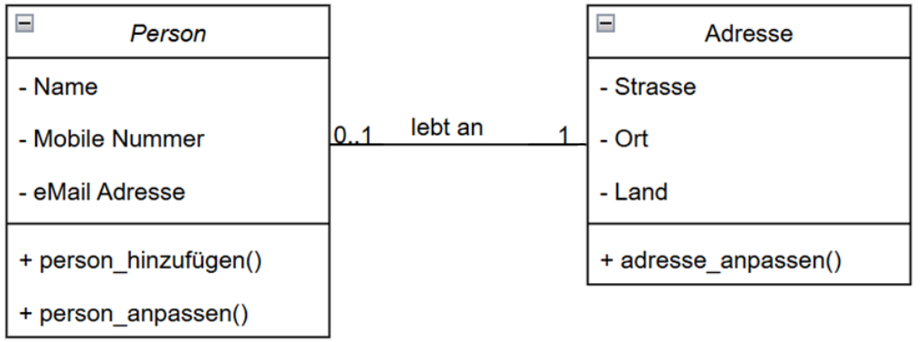
Vergleich zu **Systemarchitektur**. Diese betrachtet das System aus einer höheren Perspektive. Es ist eine high-level Dokumentation zur Unterstützung von Architekten und Stakeholdern:

Systemarchitektur bezieht auf die Struktur eines Systems, die deren Komponenten, ihre Interaktion und ihre Beziehungen zueinander beschreibt. Es handelt sich um ein **konzeptionelles Framework**, das dazu dient, die **Gesamtheit eines Systems** (Hardware, Software, Netzwerk, etc.) **zu verstehen**.

UML Klassendiagramm

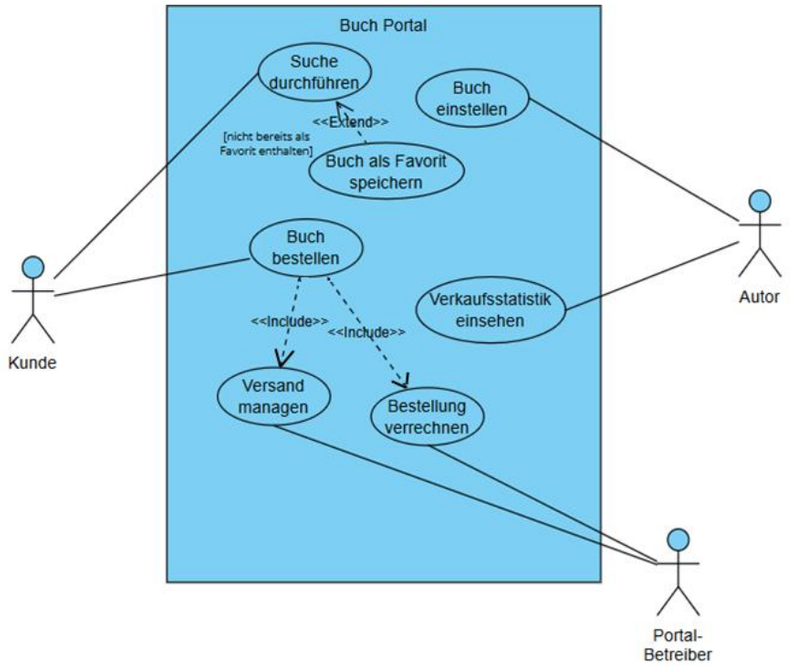
Erläuterung Klassen Diagramm

- ✓ Ein Klassendiagramm definiert die Datenstrukturen und deren Beziehungen.
- ✓ Eine Klasse hat einen **Namen, Attribute** und **Funktionen/Methoden**
- ✓ Die Klassen werden über **Assoziationen** miteinander verbunden
- ✓ Die **Kardinalität** (0, 0..1, 1, 0..*, 1..*) detailliert die Beziehung
- ✓ Es gibt weitere Modellierungsmöglichkeiten (Interface, Package, Aggregation, Komposition, … welche wir hier nicht behandeln



Vom Use Case zum Klassendiagramm

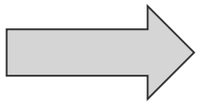
Die identifizierten Akteure, Prozesse und Datenflüsse des Kontext-, Use Case- & Aktivitätsdiagramm helfen zu bestimmen, welche Klassen benötigt werden und wie sie strukturiert sein sollten.

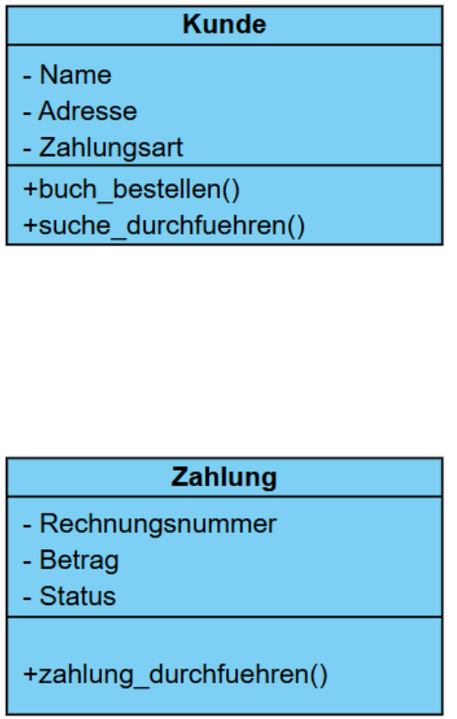


- **Kunde**
  - *Attribute*: Name, Adresse, Zahlungsart
  - *Methoden:* suche_durchfuehren(), buch_bestellen()
- **Autor**
  - *Attribute: Name, Kontoangaben*
  - *Methoden*: buch_einstellen(), verkaufsstatistik_ansehen()
- **Buch**
  - *Attribute: Titel, ISBN, Preis*
  - *Methoden*: buch_einstellen()
- **Zahlung**
  - *Attribute: Rechnungsnummer, Betrag, Status*
  - *Methode*: zahlung_durchführen()

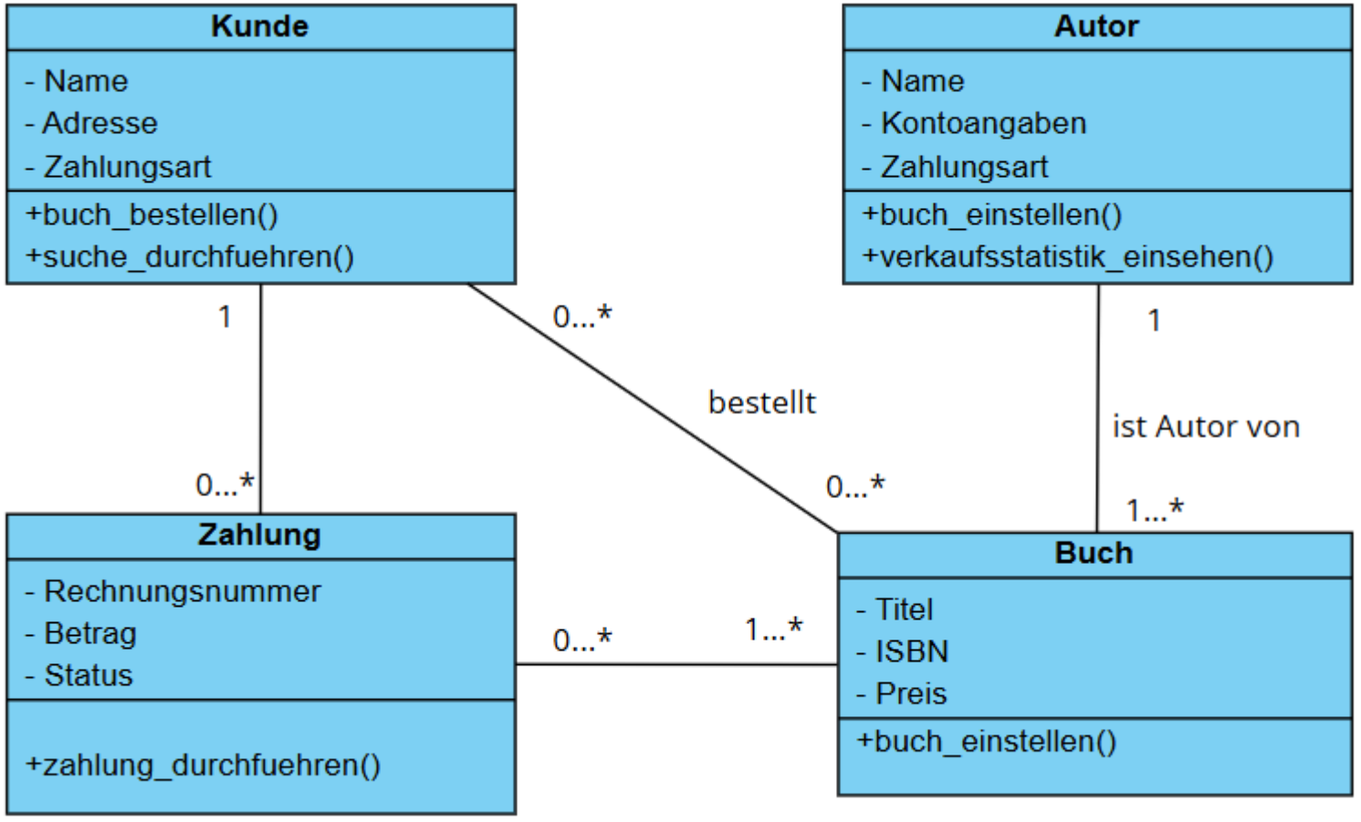
Klassendiagramm mit Attributen und Methoden

- **Kunde**
  - *Attribute*: Name, Adresse, Zahlungsart
  - *Methoden:* suche_durchfuehren(), buch_bestellen()
- **Autor**
  - *Attribute: Name, Kontoangaben*
  - *Methoden*: buch_einstellen(), verkaufsstatistik_ansehen()
- **Buch**
  - *Attribute: Titel, ISBN, Preis*
  - *Methoden*: buch_einstellen()
- **Zahlung**
  - *Attribute: Rechnungsnummer, Betrag, Status*
  - *Methode*: zahlung_durchführen()





Ergänzung um Relationen



Das Klassendiagramm müsste zur Umsetzung weiter verfeinert werden:

- **Zwischen Kunde und Buch würde eine Zwischentabelle benötigt**
- **Zwischen Zahlung und Buch würde eine Zwischentabelle «Bestellung» benötigt**
- **Mit dieser Modellierung kann ein Buch nur einen Autor haben,**
- **…**

UML Sequenz Diagramm

Erläuterung Sequenz Diagramm

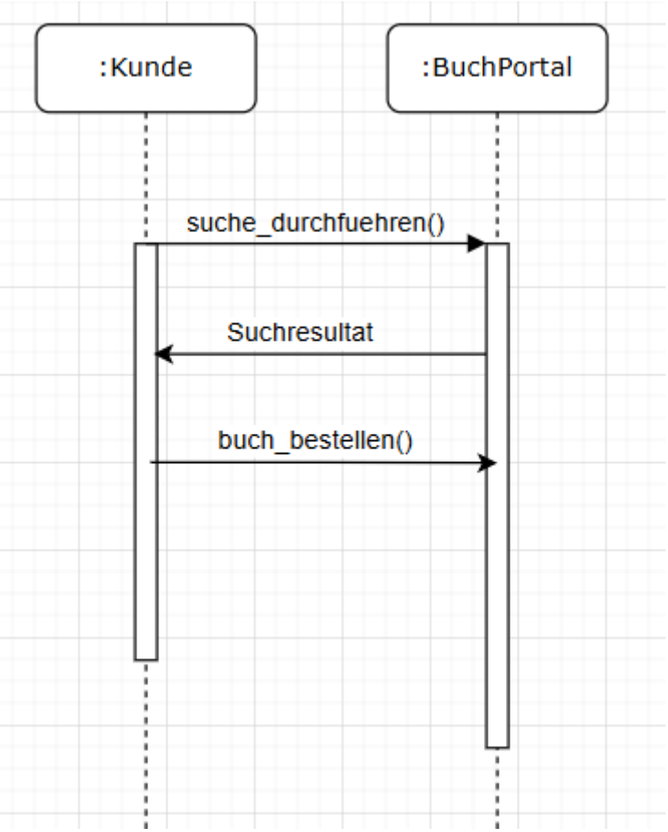
- ✓ Ein Sequenzdiagramm visualisiert die **zeitliche Abfolge** der Nachrichten zwischen Objekten eines Systems. Die Zeitachse startet oben und steigt allmählich ab.
  Es **zeigt, wie Objekte interagieren**, um eine bestimmte Funktion umzusetzen.

- ✓ **Lebenslinien** repräsentieren Objekte oder Komponenten im System. Sie werden als vertikale gestrichelte Linien dargestellt, mit dem Namen des Objekts am oberen Rand.

- ✓ **Nachrichten** stellen die Kommunikation zwischen Objekten dar. Dargestellt durch Pfeile.

- ✓ **Aktivierungsbalken** (auch Fokus der Kontrolle genannt) zeigen an, wann ein Objekt aktiv ist oder eine Aktion ausführt. Sie werden als schmale Rechtecke auf der Lebenslinie dargestellt

- ✓ Es gibt weitere Modellierungsmöglichkeiten (Asynchrone vs synchrone Nachrichten, Loops, Optionale Nachrichten, …) welche wir hier nicht behandeln.



Übung: Führt das Sequenzdiagramm weiter

- Die Bestellung geht an den Portalbetreiber
- Dieser stellt die Rechnung und überweist nach Bezahlung den entsprechenden Betrag dem Autor weiter
- 2-3er Gruppen / 15 min

Model-Driven Development (MDD)

Model-Driven Development (MDD)

- **Model Driven Development (MDD) basierend auf Klassendiagrammen ist ein Ansatz in der Softwareentwicklung, bei dem UML-Klassendiagramme als Ausgangspunkt für die automatische Codegenerierung dienen.**

- **Nutzen:**
  - **Schnellere Entwicklung: Grundlegende Codestrukturen werden automatisch erstellt.**
  - **Konsistenz: Das Modell bleibt die zentrale Quelle der Wahrheit.**
  - **Wartbarkeit: Änderungen im Modell können schnell in den Code übertragen werden.**

- **Vorgehen:**

MDD

Erstellung des Klassendiagramms

```text
+-------------+        +-------------+
|   Product   |        |    Order    |
+-------------+        +-------------+
| - id: int   |        | - id: int   |
| - name: str |  1..*  | - date: Date|
| - price: dec|<-------| - total: dec|
+-------------+        +-------------+
    ^                       |
    |                       | 1
    |                +-------------+
    |                |  Customer   |
    |                +-------------+
    |                | - id: int   |
    |                | - name: str |
    |                | - email: str|
    |                +-------------+
    |
+-------------+
| DigitalProd |
+-------------+
| - url: str  |
+-------------+
```

Quelle: Perplexity.ai. Prompt «MDD basierend auf Klassendiagramm am Beispiel Online-Buch Portal» Zugriff 31.01.2025

MDD

Codegenerierung

```java
public class Product {
    private int id;
    private String name;
    private BigDecimal price;
    // Getter und Setter
}

public class Order {
    private int id;
    private Date date;
    private BigDecimal total;
    private List<Product> products;
    private Customer customer;
    // Getter und Setter
}

public class Customer {
    private int id;
    private String name;
    private String email;
    // Getter und Setter
}

public class DigitalProduct extends Product {
    private String url;
    // Getter und Setter
}
```

Quelle: Perplexity.ai. Prompt «MDD basierend auf Klassendiagramm am Beispiel Online-Buch Portal» Zugriff 31.01.2025

MDD

Anpassung und Erweiterung

```java
public class Order {
    // ... generierte Felder und Methoden ...

    public void addProduct(Product product) {
        products.add(product);
        recalculateTotal();
    }

    private void recalculateTotal() {
        this.total = products.stream()
            .map(Product::getPrice)
            .reduce(BigDecimal.ZERO, BigDecimal::add);
    }
}
```

Quelle: Perplexity.ai. Prompt «MDD basierend auf Klassendiagramm am Beispiel Online-Buch Portal» Zugriff 31.01.2025

Referenzarchitekturen bzw. -designs & Cloud-Notationen

Notationen

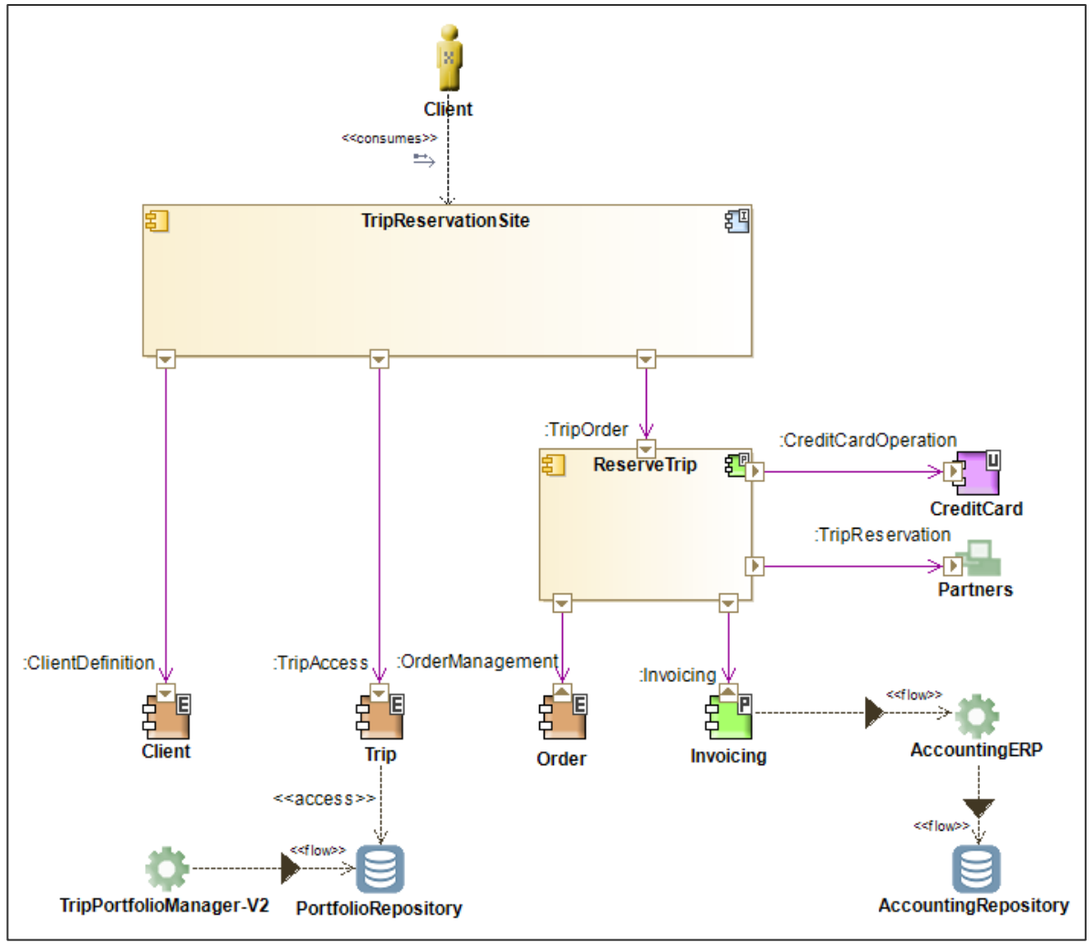
- **Analog zu ArchiMate® für Unternehmensarchitektur, gibt es Notationen zur Modellierung des Systemdesign.**
- **Je nach vorhandenen Systemen wird eine andere Notation eingesetzt, oft auch eine spezifische Zeichnung / Visiodiagramm.**
- **Notationen von Cloud „Systemen“ (Hyperscaler) werden vermehrt eingesetzt.**

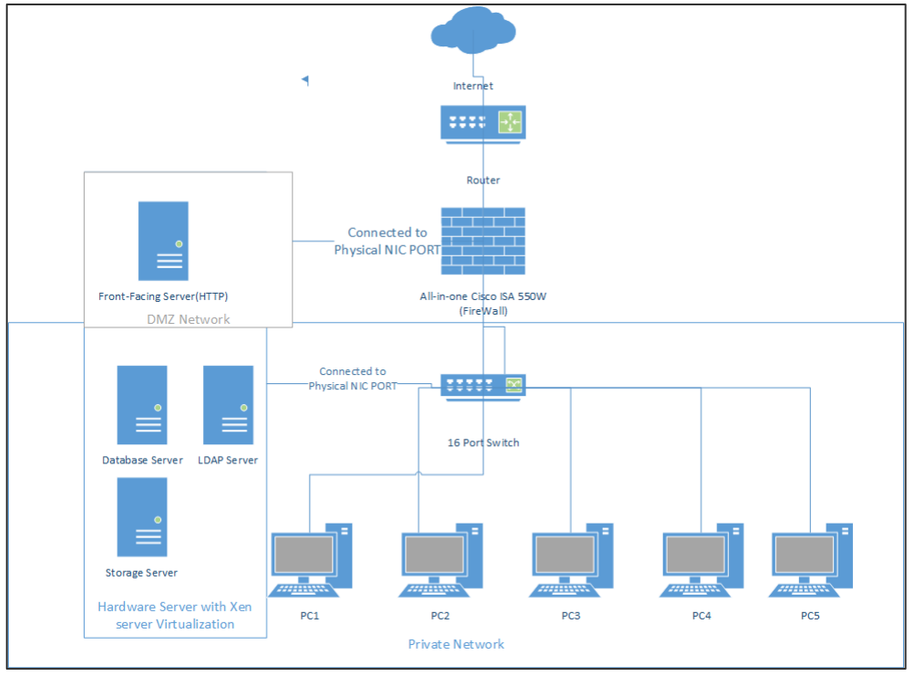
Systemdesign bezieht sich auf die **spezifische Ausarbeitung** und Strukturierung der einzelnen Komponenten eines Systems **sowie ihrer Interaktionen**.

Fokus von Klassendiagramm, Sequenzdiagramm

Das Systemdesign geht sehr ins Detail und legt fest, wie jede Komponente implementiert wird. Es enthält **Entscheidungen über Programmiersprachen, Frameworks, Datenbanken**, Schnittstellen und andere technische Aspekte. Es enthält detaillierte Dokumentation zur Unterstützung der Entwickler während der Implementierung.

Beispiel Visiodiagramme

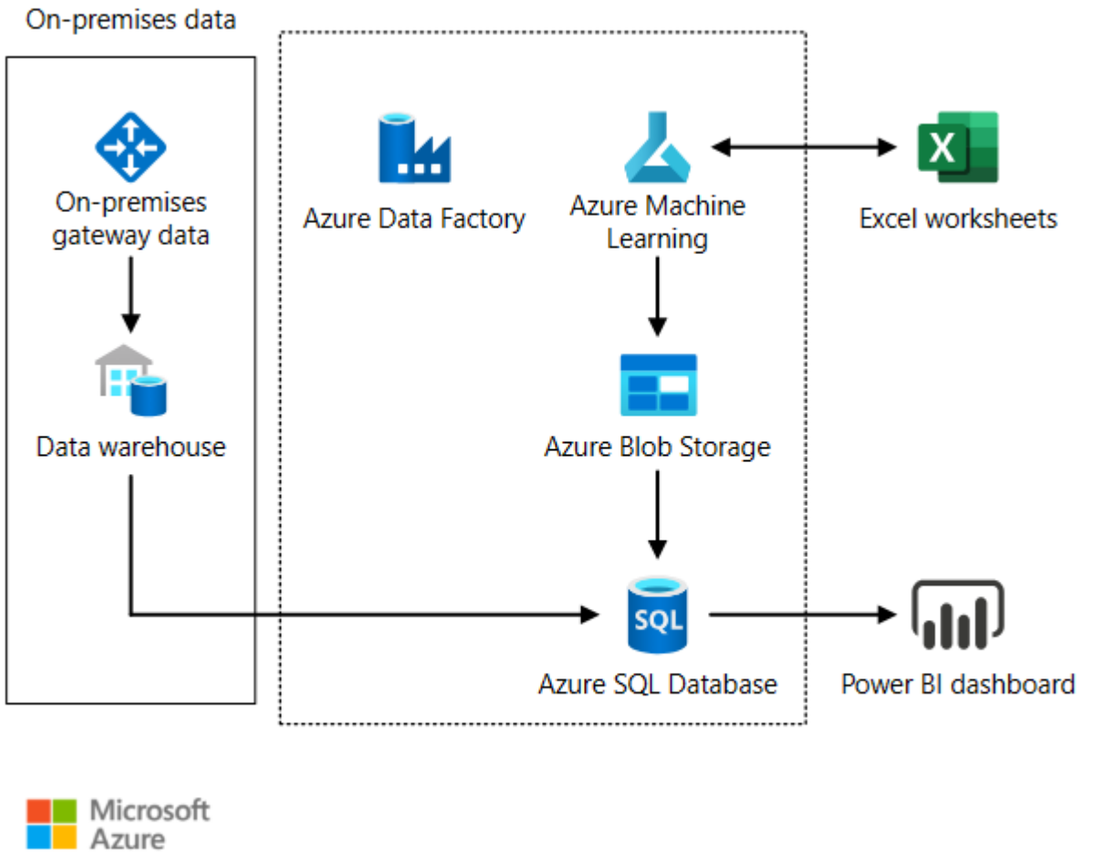




Quelle: https://serverfault.com/questions/529028/proposed-network-design-for-dmz-with-server-virtualization

Beispiel Azure

1. Azure Machine Learning ermöglicht das Erstellen von Preismodellen.
2. Azure Blob Storage speichert das Modell und alle erzeugten Zwischendaten.
3. Azure SQL-Datenbank speichert Transaktionsverlaufsdaten und alle erzeugten Modellvorhersagen.
4. Azure Data Factory wird verwendet, um regelmässige (z. B. wöchentliche) Modellaktualisierungen zu planen.
5. Power BI bietet eine Visualisierung der Ergebnisse.
6. Excel Kalkulationstabellen verwenden Vorhersagewebdienste.



Quelle: https://learn.microsoft.com/de-de/azure/architecture/solution-ideas/articles/interactive-price-analytics

Referenzarchitekturen bzw. -designs

Referenzarchitekturen werden in der Informationstechnologie verwendet, um bewährte Methoden, Muster und Designprinzipien zu definieren, die als Grundlage für die Entwicklung und Implementierung von IT-Lösungen dienen.

Standardisierung: Referenzarchitekturen fördern die Standardisierung von Design- und Implementierungspraktiken innerhalb einer Organisation oder einer Branche. Dies ermöglicht eine konsistente und wiederholbare Herangehensweise an die Entwicklung von IT-Lösungen.

Beschleunigte Entwicklung: Durch die Verwendung von Referenzarchitekturen können Entwicklungs- und Implementierungszeiten verkürzt werden, da Organisationen auf bewährte Muster und Best Practices zurückgreifen können, anstatt Lösungen von Grund auf neu zu entwickeln.

Bessere Interoperabilität: Referenzarchitekturen können dazu beitragen, die Interoperabilität zwischen verschiedenen Systemen und Anwendungen zu verbessern, indem sie Standards und Schnittstellen festlegen, die eine nahtlose Integration ermöglichen.

Risikominderung: Da Referenzarchitekturen bewährte Methoden und Designprinzipien enthalten, können sie dazu beitragen, potenzielle Risiken und Fehler bei der Entwicklung und Implementierung von IT-Lösungen zu reduzieren.

Skalierbarkeit und Flexibilität: Referenzarchitekturen sind in der Regel so konzipiert, dass sie skalierbar und flexibel sind, sodass Organisationen ihre IT-Systeme an sich ändernde Anforderungen und Arbeitslasten anpassen können.

Wissensaustausch: Referenzarchitekturen dienen als Mittel zum Wissensaustausch innerhalb einer Organisation oder einer Branche. Sie ermöglichen es Fachleuten, bewährte Praktiken und Erfahrungen zu teilen und von den Erfahrungen anderer zu lernen.

Ergänzung: Durch die Verwendung einer entsprechenden Notation wird die gemeinsame Sprache gesprochen.

Orange: Etwas weit her geholte Vorteile (Anmerkung Dozent)

Quelle: chat.openai.com, Prompt „Wozu werden Referenzarchitekturen verwendet? Vorteile? “, 25.03.2024

Notationen und Referenzarchitekturen bzw. -designs

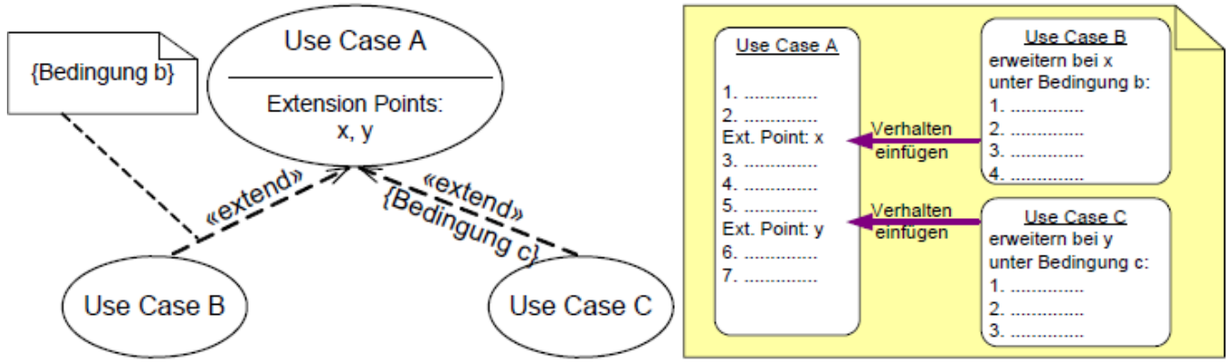
	Zeichnung / Visio	Microsoft Azure	GCP (Google Cloud Platform)	AWS (Amazon Web Services)
Notation	-	~Overview Full Set Visual paradigm	Google Cloud Cheat Sheet	AWS Architecture Deck
Referenz-architekturen	-	Reference Architectures	Reference Architectures	Reference Architectures Weitere

- **Betrachtet die unterschiedlichen Notationen und Referenzarchitekturen. Ziel ist es, solche zu erkennen und die Möglichkeiten und das Potential zu verstehen.**

- ***Zeit: 15 Minuten***

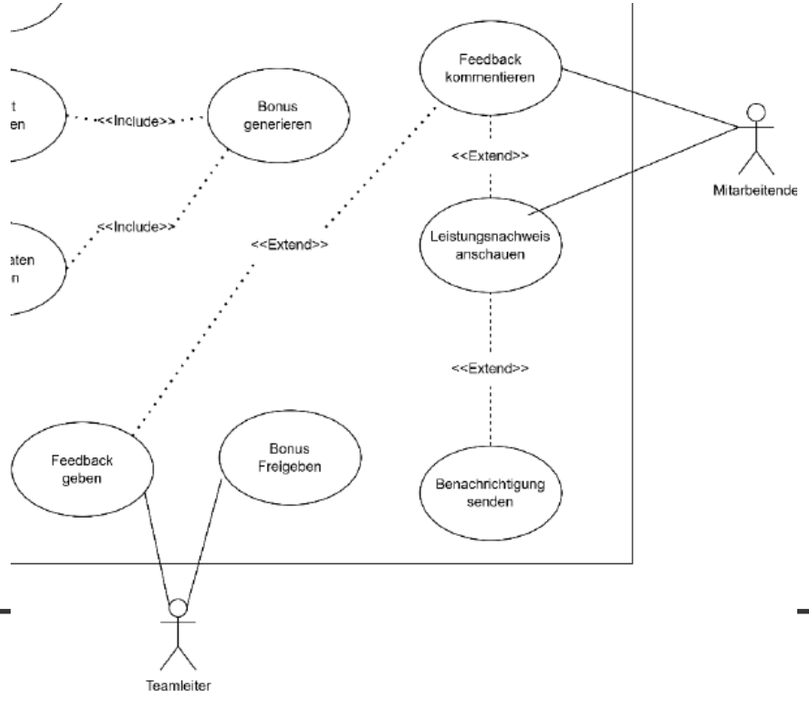
Auftrag 6: Klassenfeedback

- **Extension Points werden verwendet, um die Stelle im Ablauf eines Anwendungsfalls zu kennzeichnen, an der ein erweiternder Use Case optional eingefügt werden kann.**



- **Generell: Pfeile, Systegrenze, Beschreibung aller Use Cases**

- **Feedback kommentieren extends Feedback geben? Und auch Leistungsnachweis anschauen?**



Auftrag 6: Feedback pro Gruppe

Vormittag
Gruppe 2
Gruppe 4

Nachmittag
Gruppe 8
Gruppe 10
Gruppe 12
Gruppe 5
Gruppe 7
Gruppe 1 (kein Feedback)
Gruppe 2 (kein Feedback)

Gruppenarbeit

Auftrag 7