

# Software Entwicklung 1

Asha Schwegler

24. März 2022

## Inhaltsverzeichnis

<b>1</b>	<b>Software Engineering</b>	<b>3</b>
<b>2</b>	<b>Prozess und Prozess-Modell</b>	<b>3</b>
2.1	Vorgehensmodelle . . . . .	3
2.2	Agile SWE . . . . .	4
<b>3</b>	<b>Modellierung</b>	<b>5</b>
3.1	UML . . . . .	5
3.1.1	Gebrauch der UML . . . . .	5
<b>4</b>	<b>Wesentliche Artefakte</b>	<b>6</b>
4.1	Überblick Anforderungsanalyse . . . . .	6
4.2	Überblick Design . . . . .	7
4.3	Überblick Implementation . . . . .	7
4.4	Überblick Testing . . . . .	7
<b>5</b>	<b>Usability und User Experience (UX)</b>	<b>7</b>
5.1	Usability . . . . .	8
5.2	Usability Engineering . . . . .	8
5.3	Usability Anforderungen . . . . .	8
5.3.1	Aufgabenangemessenheit . . . . .	8
5.3.2	Selbstbeschreibungsfähigkeit . . . . .	9
5.3.3	Kontrolle . . . . .	9
5.3.4	Erwartungskonformität . . . . .	9
5.3.5	Fehlertoleranz . . . . .	9
5.3.6	Individualisierbarkeit . . . . .	10
5.3.7	Lernförderlichkeit . . . . .	10
<b>6</b>	<b>User-Centered Design (UCD)</b>	<b>10</b>
6.1	User & Domain Research . . . . .	10
6.1.1	GUI Design Process . . . . .	11
6.1.2	Wichtige Artefakte . . . . .	11
6.2	Anforderungsanalyse . . . . .	12
6.2.1	Use Cases . . . . .	12
6.2.2	UML Sequenzdiagramm (SSD) . . . . .	13
6.2.3	Operation Contract . . . . .	14
6.2.4	Zusätzliche Anforderungen . . . . .	14
<b>7</b>	<b>Domänenmodellierung</b>	<b>15</b>
7.1	Domänenmodell als vereinfachtes UML Klassendiagramm . . . . .	15
7.2	Vorgehen . . . . .	16
7.2.1	Kategorienliste . . . . .	16
7.3	Datentypen von Attributen . . . . .	16
7.3.1	Anti-Pattern . . . . .	16

7.4	Vorgehensweise eines Kartografen . . . . .	17
7.5	Domänenmodell vollständig Beispiel . . . . .	17
7.6	Analysemuster . . . . .	17

# 1 Software Engineering

- Herstellung / Entwicklung von Software
- Organisation und Modellierung (Zugehörigen Datenstrukturen, Betrieb von Softwaresystemen)
- Strukturiertes Projektplan f. Entwicklung
- Unterteilung Entwicklungsprozess
  - Schritte (überschaubar, zeitlich und inhaltlich begrenzt)
  - Phasen
  - Meilensteine
- Disziplinen während Entwicklungsprozess sind verzahnt.

## Disziplinen

- **Kerndisziplinen**
  - Anforderungsanalyse
  - Softwarearchitektur und Design
  - Implementierung / Test
  - Softwareverteilung
  - Softwareeinführung
  - Wartung / Pflege
- **Unterstützungsdisziplinen**
  - Projektmanagement
  - Konfigurationsmanagement
  - Risikomanagement

# 2 Prozess und Prozess-Modell

- Prozess
  - Beschreibung Aktivitäten, Rollen und Artefakte(Informationen)
  - Software-Entwicklung und Wartung
- Prozessmodell
  - Präskriptives Modell (Vorgehensmodell und Organisationsstrukturen)
  - Planung und Lenkung
  - Unified Process, V-Modell, Scrum,...

## 2.1 Vorgehensmodelle

- Code and Fix
- Wasserfallmodell
- Iterative und inkrementelle Modelle

## **Code and Fix**

- Definition
  - Codierung / Korrektur im Wechsel mit Ad-hoc Tests
- Vorteile
  - Schnell vorankommen
  - Schnelle Ergebnisse
  - Einfache Tätigkeiten (Codieren, Testen, Fixen)
- Nachteile
  - Schlecht planbar und keine Unterstützung im Team
  - Aufwand hoch für Korrekturen
  - Schlecht wartbare Software

## **Wassefallmodell**

- Definition
  - Folge von Aktivitäten/Phasen, gekoppelt durch Teilergebnisse (Dokumente). Reihenfolge ist fest definiert.
- Vorteile
  - hohe Planbarkeit
  - Klare Aufteilung der SWE (Analyse, Design, Test,...)
- Nachteile
  - Schlechtes Risikomanagement (Lösungskonzept nur auf Papier validiert)
  - Anforderungen zu Beginn nie alle bekannt

## **Iterativ-inkrementelle Modelle**

- Definition
  - Geplante und kontrollierte Iterationen inkrementell entwickelt
- Vorteile
  - Flexibles Modell bei unklaren Anforderungen
  - Gutes Risikomanagement (Mitarbeiter und Technologie)
  - Frühe Einsetzbarkeit der Software und Feedback
- Nachteile
  - Upfront Planbarkeit hat Grenzen (Funktionalität, Zeit und Kosten)
  - Braucht Involvierung und Steuerung durch den Kunden über ganze Projektdauer

## **2.2 Agile SWE**

- Basiert auf iterativ-inkrementellen Prozessmodell
- Fokussiert auf gut dokumentierten und getesteten Code statt auf ausführlicher Dokumentation
- Sammlung von Ideen SWE Prozess flexibler und schlanker zu machen
- Adressiert bekannten Probleme bei klassischen Software-Prozessmodellen

## Strategie

- Definierte Prozesskontrolle
  - Planung am Anfang, Prozess gesteuert und überwacht
  - Geeignet für gut planbare Problemstellungen
  - Strategie: Steuerung
- Empirische Prozesskontrolle (Agil)
  - Nur Grobplanung am Anfang
  - Prozess fortlaufend überwacht
  - Rollende Planung
  - Geeignet für komplexe Problemstellungen
  - Strategie: Regelung, Deming-Cycle (Plan-Do-Check-Act)

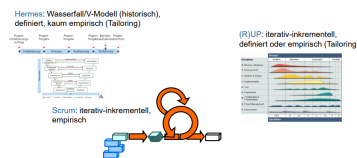


Abbildung 1: Charakterisierung Prozessmodellen

## 3 Modellierung

**Modell:** Abbild oder Vorbild für ein zu schaffendes Gebilde.

**Original:** Abgebildete oder zu schaffende Gebilde

- Modellierung
  - Software selbst ein Modell
  - Anforderungen = Modelle der Lösung
  - Testfälle = Modelle Korrektes Funktionieren des Codes

### 3.1 UML

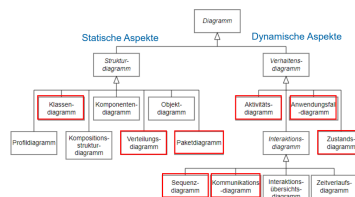


Abbildung 2: Gütereinteilung.

#### 3.1.1 Gebrauch der UML

- UML as a sketch
  - Informell, unvollständig
  - Bevorzugt von agile Community
- UML as blueprint

- Detaillierte Analyse und Design-Diagramme für Code
- Forward - und Reverse-Engineering
- UML as a Programming Language
  - Komplette, ausführbare Spezifikation eines Software-Systems in UML
  - MDA-Tools zur Modellierung und Generierung

## 4 Wesentliche Artefakte

- Anforderungsanalyse
  - Funktionale Anforderungen mit Use Cases
  - Qualitätsanforderungen und Randbedingungen
- Design
  - Softwarearchitektur
  - Use Case Realisierung (Statische und dynamische Modelle)
- Implementation
  - Quellcode (inkl.Javadoc)
- Testing
  - Unit-Tests
  - Integrations- und Systemtests

### 4.1 Überblick Anforderungsanalyse

- **User Research**
  - Personas
  - Szenarien
  - Contextual Inquiry
- Sketching und Prototyping
- **Use Cases**
  - Ableiten und Modellieren
  - Detaillierung (UML-Use-Case-Diagramm, Use-Case-Spezifikation, UI-Sketching)
- **Qualitätsanforderungen, Randbedingungen** erheben
- **Domänenmodell**
  - Konzeptuelles UML-Klassendiagramm
- **objektorientierten Analyse(OOA)**
  - Objekte/Konzepte in dem Problemereich zu finden und zu beschreiben

## 4.2 Überblick Design

- **Softwarearchitektur**
  - UML-Paketdiagramm
  - UML-Deploymentdiagramm
- **Use-Case-Realisierung und Klassendesign**
  - UML-Klassendiagramm
  - UML-Sequenzdiagramm
  - UML-Kommunikationsdiagramm
  - UML-Zustandsdiagramm
  - UML-Aktivitätsdiagramm
- Entwurf **Design Patterns**
- **Objektorientierten Design (OOD)**
  - Geeignete Softwareobjekte und ihr Zusammenwirken definieren

## 4.3 Überblick Implementation

- **Code**
  - Umsetzung Design in entspr. OOP-Sprache
- **Refactoring**
  - Code smells aufdecken und verbessern
- **laufende Dokumentierung**
  - Quellcode

## 4.4 Überblick Testing

- **Unit-Tests**
  - Laufendes Design und Implementierung
- **Teststufen Integration und System**
  - Planung, Design und Durchführung
- **Dokumentation**
  - Testkonzept und Test

## 5 Usability und User Experience (UX)

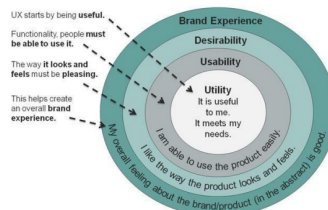


Abbildung 3: Usability.

## 5.1 Usability

**Definition:** Effektivität, Effizienz, Zufriedenheit -> Ziele erreichen im spezifischen Kontext

- **4 wichtige Aspekte**

- Benutzer
- Seine Ziele/Aufgaben
- Sein Kontext
- Softwaresystem (inkl. UI)

## 5.2 Usability Engineering

**Ziel:** Software entwickeln, die drei Anforderungen erfüllt

- **Drei Anforderungen:**

- Effektivität
  - \* Aufgaben vollständig erfüllen
  - \* Genauigkeit
- Effizienz
  - \* Mit minimalem Aufwand (Mental, Physisch, Zeit)
- Zufriedenheit
  - \* **Minimum:** nicht verärgert
  - \* **Normal:** Zufrieden
  - \* **Optimal:** Erfreut

## 5.3 Usability Anforderungen

- **7 Anforderungen:**

- Aufgabengemessenheit
- Lernförderlichkeit
- Individualisierbarkeit
- Erwartungskonformität
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Fehlertoleranz

### 5.3.1 Aufgabenangemessenheit

- Minimale Anz. Schritte f. Aufgabe
- Nur wichtige Informationen
- Kontextabhängige Hilfe
- Minimale Anz. Benutzereingaben
  - Jede Eingabe nur 1x
  - Standardwerte
  - Liste vordefinierter Werte (z.B. Länder)
  - Ableitbare Eingaben vorschlagen



### 5.3.2 Selbstbeschreibungsfähigkeit

- Benutzer ausreichend informieren
  - Wo er ist
  - Was er tun soll / kann
  - Wie er es tun soll (Formate, Werte)
- Begriffe des Benutzers verwenden (Labels, Fehlermeldungen)
- Affordanzen

### 5.3.3 Kontrolle

- Mit Interaktion Benutzer steuern
  - Initiative, Tempo
  - Dialogfluss
  - Darstellungsformate
  - Inputmodalität (Maus, Tastatur, Touch, Sprache)
- Benutzeraktionen rückgängig machen können
- Benutzeraktionen jeder Zeit abbrechen können

### 5.3.4 Erwartungskonformität

- Bezüglich
  - Design
  - Interaktion
  - Struktur
  - Komplexität
  - Funktionalität
- Konsistenz
  - Terminologie
  - Verhalten (Reihenfolge Aktionen, Änderungen)
  - Informationsdarstellung (Platzierung, Wortwahl)

### 5.3.5 Fehlertoleranz

- Benutzerfehler vermeiden
  - Klar kommunizieren (Erwarteter Input, Funktionen aktiv resp. sinnvoll)
- Benutzereingaben vor Aktion überprüfen
- Nicht unbedingt beim Tippen
- Benutzer helfen
  - Fehler zu erkennen
  - Ursache zu verstehen
  - Aus Fehlerzustand zu kommen
- Einfache Korrektur
- Kein Datenverlust

### 5.3.6 Individualisierbarkeit

- System anpassbar sein:
  - Know-How
  - Sprache
  - Kultur
  - Benutzer mit Einschränkungen

### 5.3.7 Lernförderlichkeit

- Informationen über unterliegende Konzepte und Regeln anbieten
  - Um mentales Modell anzugleichen
  - Nur auf Verlangen des Users
  - einfache Tasks ohne Vorkenntnisse
  - komplexere Konzepte bei der Verwendung zu erlernen

## 6 User-Centered Design (UCD)

- UCD Process (ISO 9241)

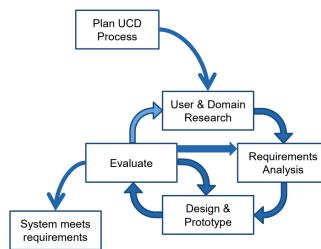


Abbildung 4: UCDProcess

### 6.1 User & Domain Research

- **Ziele bez. Benutzer:**
  - Wer ist Benutzer
  - Was ist die Arbeit (Aufgaben, Ziele)
  - Wie sieht Arbeitsumgebung aus
  - Was wird gebraucht um Ziele zu erreichen
- Welche Sprache, Begriffe
- Normen (organisatorisch, kulturell, sozial)
- Pain Points (Brüche, Workarounds)
- Für mobile Apps:
  - Nutzungskontext
    - \* Wo wird App benutzt (Umgebung)
    - \* Wann wird App benutzt (Tageszeit, involvierte Personen, Randbedingungen)
    - \* Warum wird App benutzt (Nutzen, Motivation, Trigger)
- **Ziele bez. Domäne:**
  - Business der Firma verstehen
  - Domäne verstehen (Sprache, Wichtigste Konzepte, Prozesse)

### 6.1.1 GUI Design Process

#### Methoden User & Domain Research

- Contextual Inquiry
- Interviews
- Beobachtung
- Fokusgruppen
- Umfragen
- Nutzungsauswertung
- Desktop Research (Dokumentenstudium, Mitbewerber)

### 6.1.2 Wichtige Artefakte

- **Personas**
- **Usage-Szenarien**
  - Kurze Geschichte
    - \* **Usage Szenarien**
      - aktuelle Situation
      - in User and domain research verwendet
    - \* **Kontextszenarien**
      - Zukünftige gewünschte Situation
      - in Anforderungsanalyse verwendet
- **Mentales Modell**
- **Domänenmodell**
- **Stakeholder Map**

- **Stakeholder Map**
  - Zeigt die wichtigsten Stakeholders im Umfeld der Problemdomäne

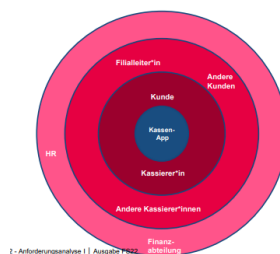


Abbildung 5: Stakeholdermap

- **Service Blueprint/Geschäftsprozessmodell**

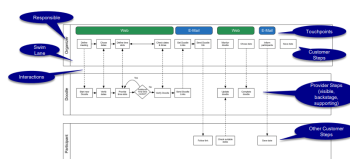


Abbildung 6: Blueprint

## 6.2 Anforderungsanalyse

**Ziel:**

- Ausgehend von den Resultaten des UCD -> User-Anforderungen ableiten:
  - Funktionale Abläufe, Interaktionen
    - \* **Kontextszenarien**
    - \* **Storyboards**
    - \* **UI-Skizzen**
    - \* **Use cases**
  - Konzepte, Beziehungen, Quantitäten
    - \* **Kontextszenarien**
    - \* **FURPS-Modell (Functionality, Usability, Reliability, Performance, Supportability)**

### 6.2.1 Use Cases

- Akteur
  - Primärakteur
  - Unterstützender Akteur
  - Offstage-Akteur
- Keine Kann-Formulierungen
- 3 Ausprägungen:
  - Kurz
    - \* Titel + 1 Absatz (Standardablauf)
  - Informell
    - \* Titel + Informelle Beschreibung (können mehrere Absätze sein, beschreibt auch Varianten)
  - Vollständig
    - \* Titel + alle Schritte und alle Varianten im Detail
    - \* UC-Name
    - \* Umfang
    - \* Ebene
    - \* Primärakteur
    - \* Stakeholders und Interessen
    - \* Vorbedingungen
    - \* Erfolgsgarantie/Nachbedingungen
    - \* Standardablauf
    - \* Erweiterungen
    - \* Spezielle Anforderungen
    - \* Liste der Technik und Datavariationen
    - \* Häufigkeit des Auftretens
    - \* Verschiedenes
  - Notation = Nomen + Verb

## Use-Case-Diagramm

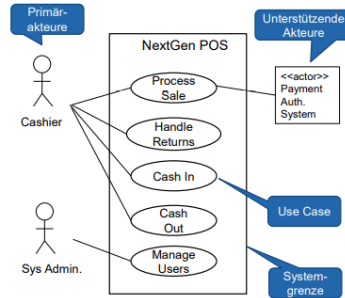


Abbildung 7: UseCaseDiagramm

## Zusätzliche Beziehungen im Use Case Diagramm



Abbildung 8: Zusätzliche Beziehungen UseCaseDiagramm

## 6.2.2 UML Sequenzdiagramm (SSD)

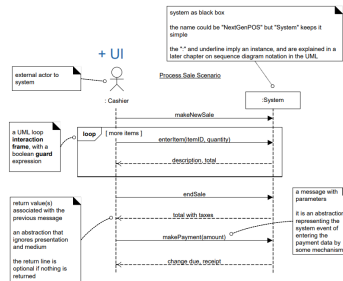


Abbildung 9: Zusätzliche Beziehungen Systemsequenzdiagramm

## SSD zwischen zwei Systemen

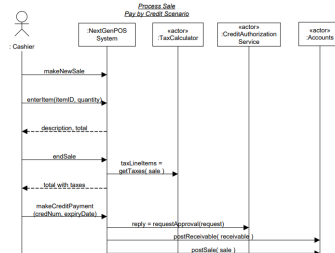


Abbildung 10: Zusätzliche Beziehungen Systemsequenzdiagramm zwischen zwei Systeme

## System Operation

- Formal wie ein Methodenaufruf

- Treffender Name
- Evt. mit Parametern
- Durchzogener Pfeil für Methodenaufruf
- Rückgabewert (Kann fehlen falls unwichtig, Gestrichelte Linie weil Update des UI)
- Definieren API des Systems

### 6.2.3 Operation Contract

**Definition:** Spezifiziert (System)Operation

- Name plus Parameterliste
- Vorbedingung (Was muss zwingend erfüllt sein damit Systemoperation aufgerufen werden kann)
- Nachbedingung
  - Was hat sich alles geändert nach Ausführung (Erstellte / gelöschte Instanzen, Assoziationen, geänderte Attribute)
  - basierend auf Domänenmodell

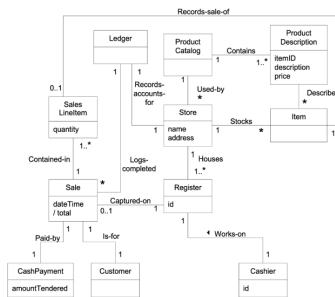


Abbildung 11: OperationContract

### 6.2.4 Zusätzliche Anforderungen

- Funktionale
- Nicht-Funktionale

#### Formulierung

- Anforderungstatements
  - Als Anforderung formuliert
  - messbar/verifizierbar
- So wenig wie nötig
  - Nur diejenige, die begründet gefordert werden
  - Keine ersten Lösungsideen als Forderungen

## Checkliste FURPS+

- **F**unctionality
  - Features, Fähigkeiten, Sicherheit
- **U**sability
  - Usability Anforderungen (Kap.5.3)
  - Accessibility
- **R**eliability
  - Fehlerrate, Wiederanlauffähigkeit, Vorhersagbarkeit, Datensicherung
- **P**erformance
  - Reaktionszeiten, Durchsatz, Genauigkeit, Verfügbarkeit, Ressourceneinsatz
- **S**upportability
  - Anpassungsfähigkeit, Wartbarkeit, Internationalisierung, Konfigurierbarkeit
- **+**
  - Implementation (HW,BS,Sprachen, Tests...)
  - Interface
  - Operations
  - Packaging
  - Legal

## Glossar

- Einfaches Glossar
  - Begriffe im Projekt und SW-Produkt
  - beliebige Elemente
- Data Dictionary
  - Zusätzliche Datenformate, Wertebereiche, Validierungsregeln

## 7 Domänenmodellierung

**Definition:** Vereinfachtes UML-Klassendiagramm.

**Aufbau:** Fachliche Begriffe mit ihren Attributen, setzt Begriffe in Beziehung zueinander. Geht nur um die Problemstellung und das Fachgebiet.

### 7.1 Domänenmodell als vereinfachtes UML Klassendiagramm

**Konzepte** = Klassen

**Eigenschaften** = Attributen (Typenangabe entfällt)

**Assoziationen** = Beziehungen zwischen Konzepten mit Multiplizitäten an beiden Enden.

Nur wenn es einen guten Grund gibt:

- **Aggregation** = Keine echte Semantik, als Abkürzung für "hat".
- **Komposition** = z.B. wenn Produktkatalog gelöscht wird, dann auch die darin enthaltenen Produktbeschreibungen. Abkürzung "bietet an".

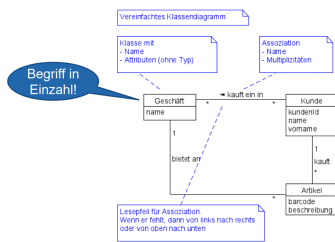


Abbildung 12: DM1

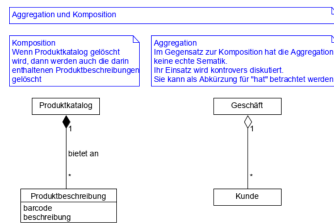


Abbildung 13: DM2

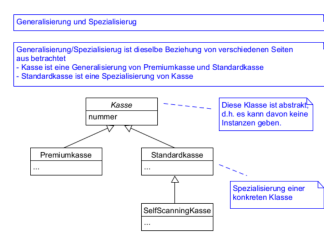


Abbildung 14: DM3

## 7.2 Vorgehen

1. Konzepte identifizieren
  - (a) Fachwissen und Erfahrung verwenden
  - (b) Substantive aus Anwendungsfällen
  - (c) Kategorienliste verwenden
2. Attributen
  - (a) Fachwissen verwenden
3. Konzepte in Verbindung zueinander setzen
  - (a) Fachwissen verwenden
  - (b) Kategorienliste verwenden
4. Auftraggeben und/oder Fachexperten beiziehen
5. Vorgehensweise eines Kartografen

### 7.2.1 Kategorienliste

Kategorie	Mögliche Konzepte für DM
Geschäftstransaktionen	
• Transaktionen als Ganzen	Sale
• Transaktionsposition	SalesItem
• Produkt, das damit verbunden ist	Item
• Wo wird Transaktion registriert?	Register
• Rollen von beteiligten Personen	Cashier
• Ort der Transaktion	Store
• Beschreibung von Dingen	ProductDescription
• Ereignisse mit Ort/Zeit	Sale

Abbildung 15: Kategorienliste1

Kategorie	Mögliche Konzepte für DM
Physische Objekte	Register
Kataloge	ProductCatalog
Container von Dingen	Store
Dinge in den Containern	Item
Andere beteiligte Systeme	CreditAuthorizationSystem
Rollen von beteiligten Personen	Cashier
Artefakte (Pläne, Finanzen, Artikel, Verträge, ...)	Receipt
Zahlungsinstrumente	Cash, Credit Card

Abbildung 16: Kategorienliste2

Kategorie	Mögliche Assoziation für DM
Transaktion	Payment - Sale
• Position	SalesItem - Sale
• Produkt	Item - SalesItem
• Rolle	Customer - Payment
Teil zum Ganzen	Register - Store
Beschreibung zum Gegenstand	ProductDescription - Item
Protokoll zum Gegenstand	Sale - Register
Verwendung	Cashier - Register

Abbildung 17: Kategorienliste3

## 7.3 Datentypen von Attributen

- Wenn nötig: eigene Datentypen als **Konzepte**
- Dann definieren wenn:
  - Typ aus mehreren **Abschnitten** (wie Tel.Nr)
  - **Operationen** darauf sind möglich (Validierung Kreditkartennummer)
  - Hat selber **eigene Attribute** (Verkaufspreis mit Anfangs & Enddatum)
  - **Verknüpft** mit Einheit (Preis mit Währung)

### 7.3.1 Anti-Pattern

#### Assoziationen statt Attribute





Abbildung 18: AntiPattern Bad

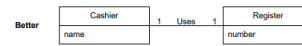


Abbildung 19: AntiPattern Good

## 7.4 Vorgehensweise eines Kartografen

- Vorhandene Begriffe oder Wissen einsetzen (Gebiete besuchen, Bewohner nach Begriffen befragen)
- Unwichtiges weglassen
- Nichts hinzufügen, was es (noch) nicht gibt
  - **Ausnahme:** System, das entwickelt wird, kann eingetragen werden
- Nur analysieren, (noch) keine Lösungen entwerfen

Keine Software Klassen im Domänenmodell :

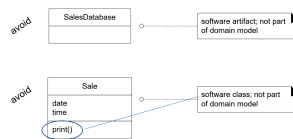


Abbildung 20: AntiPatternSoftware

## 7.5 Domänenmodell vollständig Beispiel

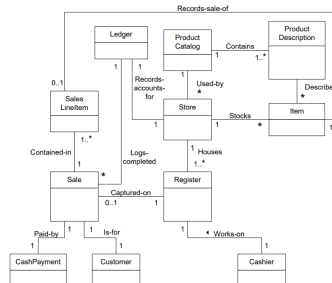


Abbildung 21: BeispielDomänenmodell

## 7.6 Analysemuster

- **Beschreibungsklassen**
  - Item = Physischer Gegenstand oder Dienstleistung
  - Mehrere Artikel desselben Typs
  - Attribute (description, price, serial number, itemID)



Abbildung 22: Beschreibungsklassen

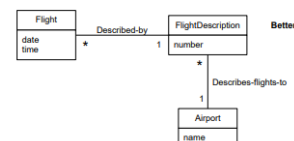


Abbildung 23: BeschreibungsklasseFlug

- Generalisierung / Spezialisierung

- Spezialisierung als is a Beziehung zu

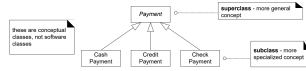


Abbildung  
GeneralisierungSpezialisierung

24:

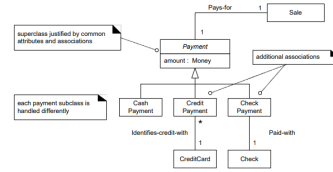


Abbildung  
GeneralisierungSpezialisierungAttribute

25:

- Komposition

- Zustände

- Eigene Hierarchie für Zustände definieren:

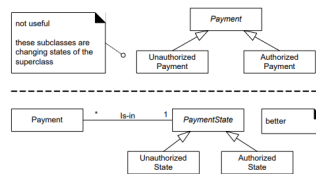


Abbildung 26: Zustände

- Rollen

- Dasselbe Konzept kann unterschiedliche Rollen einnehmen:

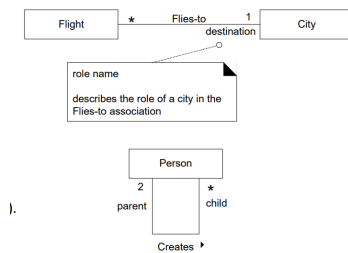


Abbildung 27: DMRolleName

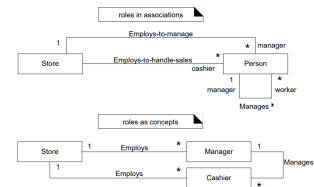


Abbildung 28: DMRolleKonzept

- Assoziationsklasse

- Wenn Assoziationen eigene Attribute haben (MerchantID für Kreditkarte Geschäft, AuthorizationService):

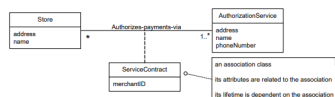


Abbildung 29: Assoziationsklasse

- Einheiten

- Manchmal sinnvoll explizit als Konzept zu modellieren :

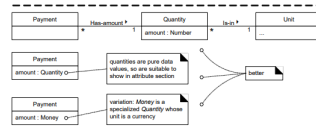


Abbildung 30: Einheiten

- **Zeitintervalle**

- Gültigkeitsintervall für sich ändernde Attribute :

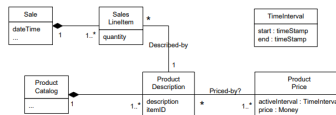


Abbildung 31: Zeitintervall