Bachelor of Science (BSc) in Informatik

SWEN1/PM3 Team:
R. Ferri (feit), D. Liebhart (lieh), K. Bleisch (bles), G. Wyder (wydg)

Modul Software-Entwicklung 1 (SWEN1)

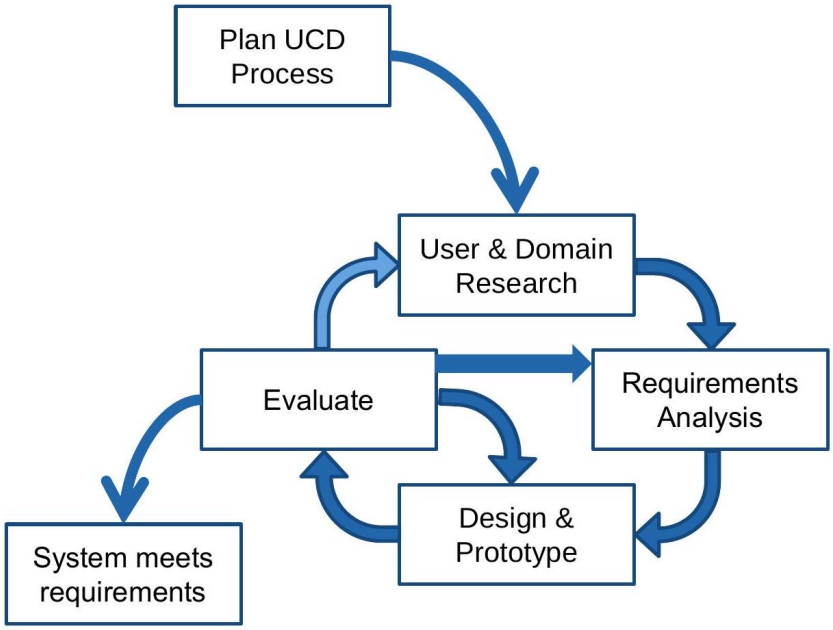
LE 03 - Anforderungsanalyse II Zusammenfassung

Warm-Up

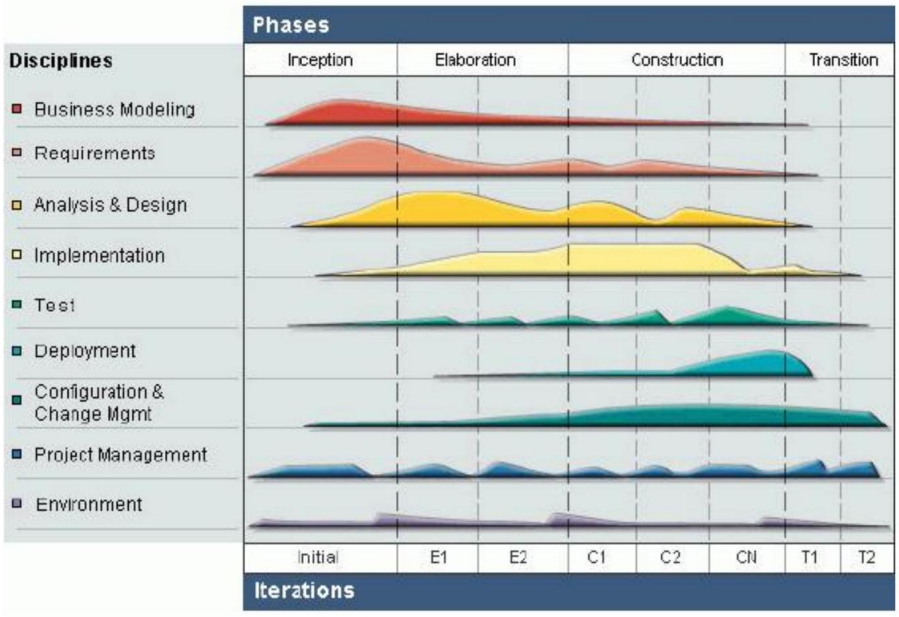
- Wo stehen wir im Software-Entwicklungsprozess nach der Anforderungsanalyse I?
- Was sind die 3 wichtigsten Usability-Anforderungen?
- Welche Artefakte werden typischerweise im UCD erstellt und was beschreiben sie?

Um was geht es?

- Wie bringt man UCD in SWE-Prozesse ein?

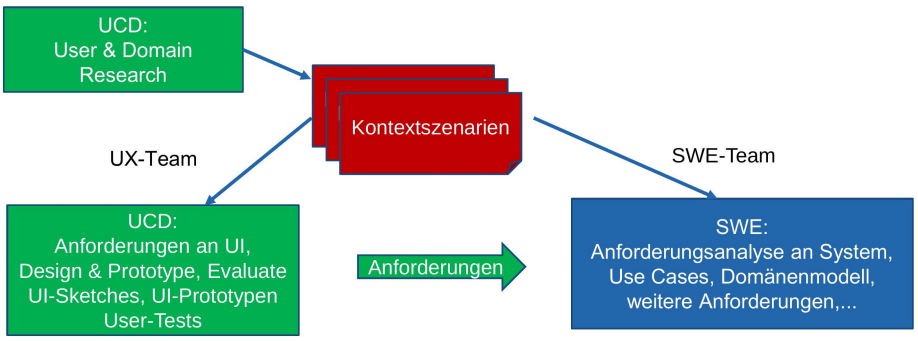


UX-Team



SWE-Team

Anforderungsanalyse: Benutzer-Anforderungen



Lernziele LE 03 - Anforderungsanalyse II

Sie sind in der Lage,

- aus den Artefakten des UCD detaillierte Anforderungen an das Softwareprodukt und dessen UI abzuleiten.
- funktionale Anforderungen in Form von Use Cases in verschiedenen Granularitätsstufen aus Kontextszenarien abzuleiten.
- weitere funktionale und nicht-funktionale Anforderungen aus den Artefakten des UCD abzuleiten.

1. Wie erfasst man (dynamische) funktionale Anforderungen mit Use Cases
2. Wie schreibt man gute Use Cases
3. System-Sequenzdiagramme, Systemoperationen und Verträge
4. Wie erfasst man zusätzliche funktionale und nicht-funktionale Anforderungen
5. Wrap-up und Ausblick

SWE: Anforderungsanalyse

- Anforderungen (Requirements)
- Forderungen bezüglich (Leistungs-) Fähigkeiten oder Eigenschaften, die das System/Projekt unter gegebenen Bedingungen erfüllen muss
- Können explizit oder implizit sein
- Kurze Diskussion

- Woher kommen die Anforderungen an ein System und wieso?

1. Benutzer (möchte einen Job erledigen in einem bestimmten Kontext, hat bestimmte Bedürfnisse, Skills, Ziele)
2. Weitere Stakeholder (Einkauf, Projektleiter, IT-Abteilung, ...)
3. Regulatorien, Gesetze, Normen

SWE: Anforderungsanalyse

- Anforderungen (Requirements)
- Sind (fast) nie im Vorneherein vollständig bekannt
- Müssen zusammen mit den Benutzern und anderen Stakeholdern erarbeitet werden
- Sie haben häufig implizite Anforderungen nicht explizite
- Explizite Anforderungen sollten hinterfragt werden (wieso bestehen sie genau so)
- Können kaum je zu Beginn vollständig erhoben werden, sondern entwickeln sich im Verlaufe des Projekts
- «I don't know what I want but I'll tell you when I see it!»
- Problematisch bei nicht iterativen Prozessen

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- Textuelle Beschreibung einer konkreten Interaktion eines bestimmten Benutzers mit dem zukünftigen System
- Aus Sicht des Akteurs
- Enthalten implizite und explizite Anforderungen
- Beschreiben das Ziel des Benutzers (= Grund für die Anforderungen)
- Beschreiben den Kontext

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- Use Cases (UCs) bilden in iterativen SWE-Prozessen eine zentrale Rolle
- Funktionale Anforderungen werden hauptsächlich mit UCs dokumentiert
- Mittels UCs können Anforderungen einfach mit dem Kunden diskutiert werden
- UCs sind ein wichtiger Teil der iterativen Projektplanung
- Projekt wird entlang von UCs geplant
- UC-Realisierungen bestimmen die Lösungsarchitektur und treiben das Lösungsdesign
- UCs werden für funktionale Systemtests eingesetzt
- UCs bilden die Basis für Benutzerhandbücher

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- 3 Ausprägungen von Anwendungsfällen
- Kurz (Brief UC)
- Titel + 1 Absatz
- Beschreibt Standardablauf (keine Varianten, Problemfälle)
- Informell (Casual UC)
- Titel + informelle Beschreibung in ein bis mehrere Absätze
- Beschreibt auch wichtige Varianten
- Vollständig (Fully dressed UC)
- Titel + alle Schritte und Varianten werden im Detail beschrieben
- Enthalten weitere Informationen zu Vorbedingungen, Erfolgsgarantien, etc.

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- Umfang eines guten Use Case
- Muss einen konkreten Nutzen für den Akteur erzeugen
- Eine Handlung, die eine Person, an einem Ort zu einer Zeit mit dem System ausführt
- Sollte mehr als eine einzelne Interaktion umfassen
- Titel eines UC
- Aktiv formulieren
- Verb! + evtl. Objekt vorangestellt (z.B. „Kasse eröffnen")
- Sollte Ziel des Akteurs beschreiben
- Boss-Test
- Falls dein Boss dich fragt, was du den ganzen Tag gemacht hast und du sagt, du hast die ganze Zeit den einen Use Case ausgeführt, sollte er zufrieden sein!
- EBP-Test (Elementary Business Proc.)
- Eine Aufgabe, die von einer Person an einem Ort zu einer bestimmten Zeit ausgeführt wird, als Reaktion auf einen Business Event.
- Size-Test
- Mehr als eine einzelne Interaktion
- Fully dressed meist mehrere Seiten lang

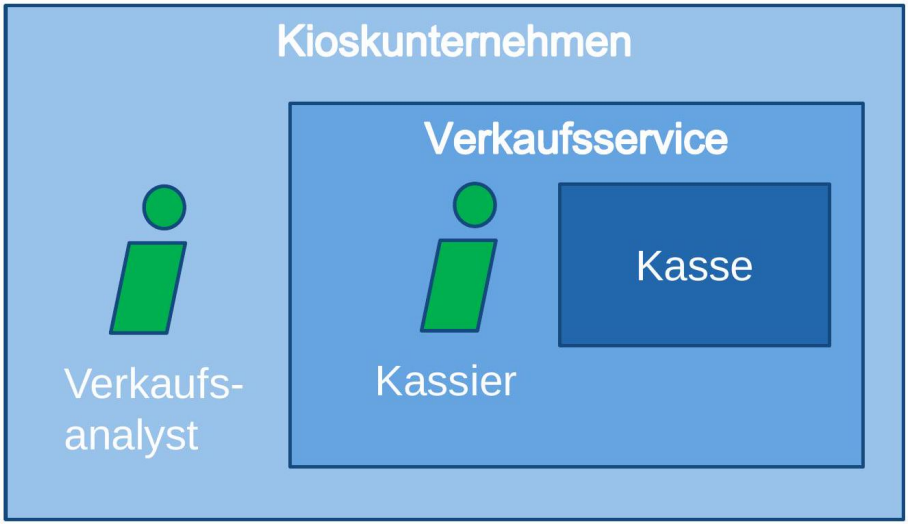
Finden von Anwendungsfällen

- Schritt 1: Systemgrenzen definieren
- Kasse
- Schritt 2: Primärakteure identifizieren
- Kassier

- Frage: Wieso nicht Kunde?

Steuerbehörde

Kioskkunde



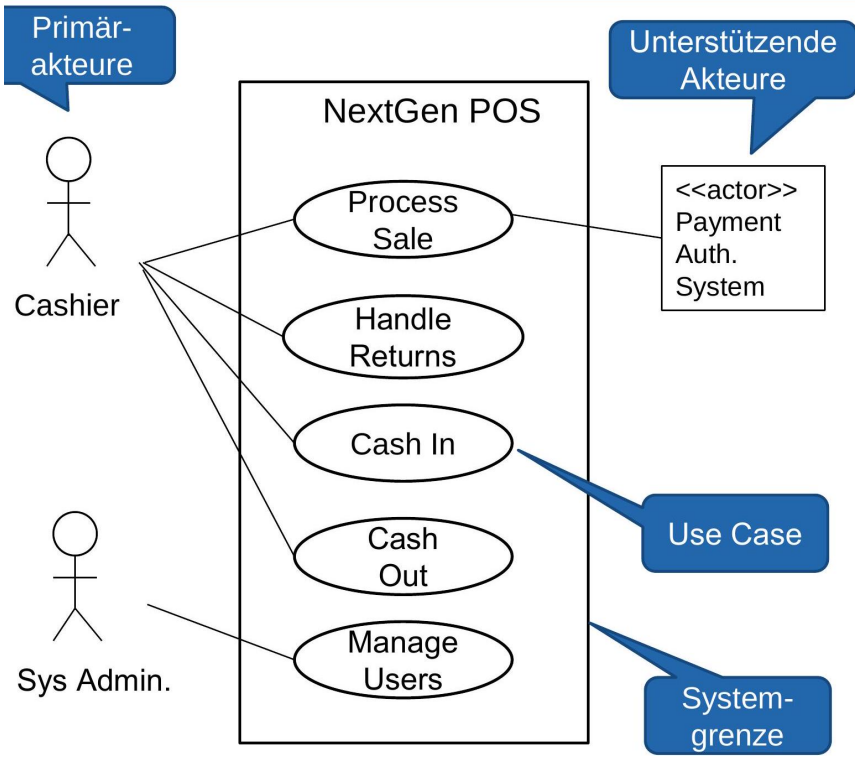
Finden von Anwendungsfällen

- Schritt 1: Systemgrenzen definieren
- Kasse
- Schritt 2: Primärakteure identifizieren
- Kassier
- Frage: Wieso nicht Kunde?
- Schritt 3:
- Jobs (Ziele, Aufgaben) der Primärakteure identifizieren
- Kassier
- Kasse eröffnen
- Verkauf abwickeln
- Kasse abrechnen
- Retouren entgegennehmen
- Kassenadministrator
- Kasse aufsetzen
- Kasse warten
- Kasse ausser Betrieb nehmen
- Kassenbenutzer administrieren

Anwendungsfalldiagramm (Use-Case-Diagramm)

- Zeigt
- Systemabgrenzung
- Akteure
- Primärakteure initiieren einen UC
- Unterstützende Akteure sind beteiligt an einem UC
- Liste der Anwendungsfälle

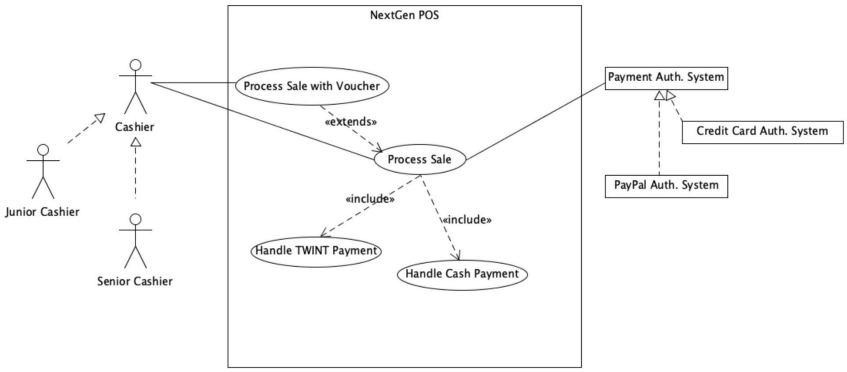
1. Process Sale
2. Cash In
3. Cash Out
4. Handle Returns
5. Manage Users



Anwendungsfalldiagramm (Use-Case-Diagramm)

School of

- Zusätzliche Beziehungen im UC-Diagramm
- «include»
- UC "Handle Cash Paymentünd UC „Handle TWINT Paymentßind enthalten im UC „Process Sale"
- Sie sind hier aber keine eigenständigen UCs (keine Verbindung zu Akteuren)
- Included UCs können auch selbständige UC sein.



Anwendungsfalldiagramm (Use-Case-Diagramm)

- Zusätzliche Beziehungen im UC-Diagramm
- «extends»
- Eigenständiger UC, der eine Erweiterung eines anderen darstellt, und
- ursprünglicher UC nicht verändert werden soll
- Sonst besser als Erweiterung im UC-Text einfügen
- Akteur-Generalisierung
- Um Akteure zusammenzufassen
- Kann als „ist-einBeziehung modelliert werden

1. Wie erfasst man (dynamische) funktionale Anforderungen mit Use Cases
2. Wie schreibt man gute Use Cases

3. System-Sequenzdiagramme, Systemoperationen und Verträge
4. Wie erfasst man zusätzliche funktionale und nicht-funktionale Anforderungen
5. Wrap-up und Ausblick

Anwendungsfälle: Brief UC

- Kurze Beschreibung des

Anwendungsfalls in einem Paragraphen

- Nur Erfolgsszenario
- Sollte enthalten
- Trigger des UCs
- Akteure
- Summarischen Ablauf des UCs
- Wann?
- Zu Beginn der Analyse
- Process Sale (brief)

Kunde kommt mit seinen Waren zur Kasse. Kassier erfasst alle gekauften Produkte. Am Ende berechnet Kasse den Totalbetrag. Kassier zieht das Geld von Kunden ein und gibt den Betrag in die Kasse ein. Diese berechnet das Rückgeld. Kassier gibt Kunde das berechnete Rückgeld zurück.

Anwendungsfälle: Casual UC

- Informelle Beschreibung des Anwendungsfalls in mehreren Paragraphen
- Erfolgsszenario plus wichtigste Alternativszenarien
- Sollte enthalten
- Trigger des UCs
- Akteure
- Interaktion des Akteurs mit System
- Wann?
- Zu Beginn der Analyse

- Process Sale (casual)
- Kunde kommt mit seinen Waren zur Kasse. Kassier scannt den Produktcode jedes Produkts ein. Kasse zeigt Artikel und Preis. Kassier korrigiert Menge, falls nötig. Bei Produkten ohne Code gibt der Kassier das Produkt und den Preis, sowie die Menge manuell ein.
- Am Ende schliesst Kassier den Einkauf ab. Kasse zeigt den Totalbetrag. Kassier nimmt das Geld vom Kunden entgegen und gibt den bezahlten Betrag in Kasse ein. Kasse berechnet den Retourbetrag und öffnet die Geldschublade. Kassier entnimmt den Retourbetrag und übergibt das Retourgeld dem Kunden. Kassier schliesst Geldschublade und beendet damit den Verkauf.
- Kasse druckt die Quittung aus. Kassier übergibt Quittung dem Kunden.

Anwendungsfälle: Fully-dressed UC

- Detaillierte Beschreibung des Ablaufs

mit allen Alternativszenarien

- Wann?
- Ende der Inception- und v.a. in ElaborationPhase (Anforderungsdisziplin)
- Nachdem die meisten UCs identifiziert und kurz beschrieben worden sind
- Die wichtigsten UCs (10%), die die Architektur bestimmen, werden im Detail ausformuliert
- Formaler Aufbau
- UC-Name
- Umfang (Scope)
- Ebene (Level)
- Primärakteur (Primary Actor)
- Stakeholders und Interessen
- Vorbedingungen (Preconditions)
- Erfolgsgarantie/Nachbedingungen (Success Guarantee)
- Standardablauf (Main Sucess Scenario)
- Erweiterungen (Extensions)

- Spezielle Anforderungen (Special Requirements)
- Liste der Technik und Datavariationen (Technology and Data Variations)
- Häufigkeit des Auftretens (Frequency of Occurance
- Verschiedenes (Miscellaneous)

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- Eigenschaften guter Anwendungsfälle
- Aussagekräftiger Titel
- Beschreibt Anwenderziel, aktiv formuliert
- Beispiel (Akteur Kassier): „Process Sale"(Einen Verkauf abwickeln)
- Essentieller Stil (nicht konkreter Stil)
- Beschreibt Logik der Interaktion, nicht konkrete Umsetzung
- Beispiel (Akteur Kassier):
- Konkret: „Kassier tippt die Produkt-ID ein. System zeigt Produktnamen."
- Essentiell: „Kassier erfasst das Produkt. System bestätigt Produkt."(z.B. durch Eintippen, Wählen, Scannen, oder per Sprache)

Anforderungsanalyse: Anwendungsfälle (Use Cases)

- Eigenschaften guter Anwendungsfälle
- Knappe aber präzise Beschreibung der Interaktion des Standardablaufs
- Keine kann-Formulierungen
- Alternative Interaktionen sind unter Erweiterungen aufgeführt
- Nur Aussensicht (Benutzersicht), keine systeminternen Interaktionen

Anwendungsfälle, und wie weiter?

- Frage
- Wie kommen wir von den Anwendungsfällen, die auf abstrakter Ebene die funktionalen Anforderungen an das System beschreiben, zu den konkreten Funktionalitäten, die das System aufweisen muss?
- Antwort
- Systemsequenzdiagramme
- Operation Contracts

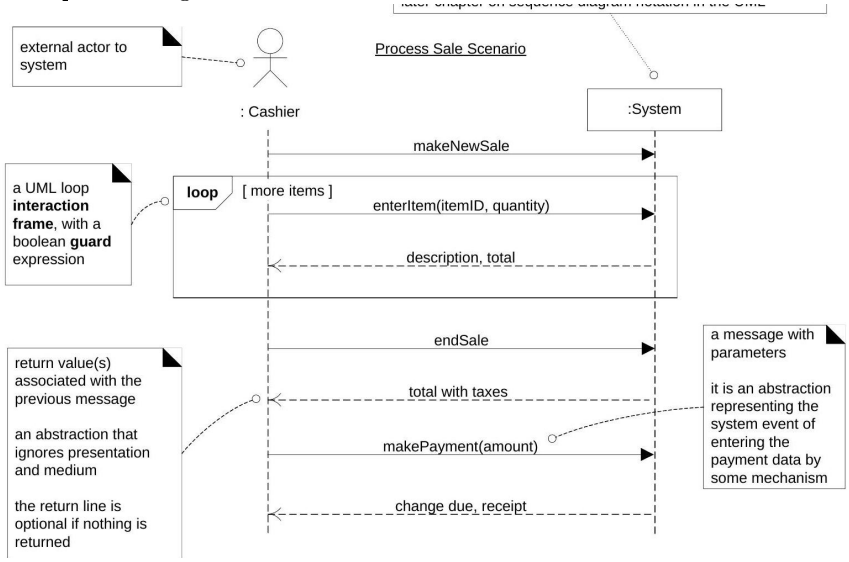
1. Wie erfasst man (dynamische) funktionale Anforderungen mit Use Cases
2. Wie schreibt man gute Use Cases
3. System-Sequenzdiagramme, Systemoperationen und Verträge
4. Wie erfasst man zusätzliche funktionale und nicht-funktionale Anforderungen
5. Wrap-up und Ausblick

Systemsequenzdiagramm (SSD)

School of Engineering nIT Institut für angewandte Informationstechnologie

- Ist formal ein UML Sequenzdiagramm
- Zeigt Interaktionen der Akteure mit dem System
- Welche Input-Events auf das System einwirken
- Welche Output-Events das System erzeugt
- Ziel
- Wichtigste Systemoperationen identifizieren, die das System zur Verfügung stellen muss (API) für einen gegebenen Anwendungsfall
  system as black box
  the name could be "NextGenPOS"but SSystem"keeps it simple
  the :änd underline imply an instance, and are explained in a later chapter

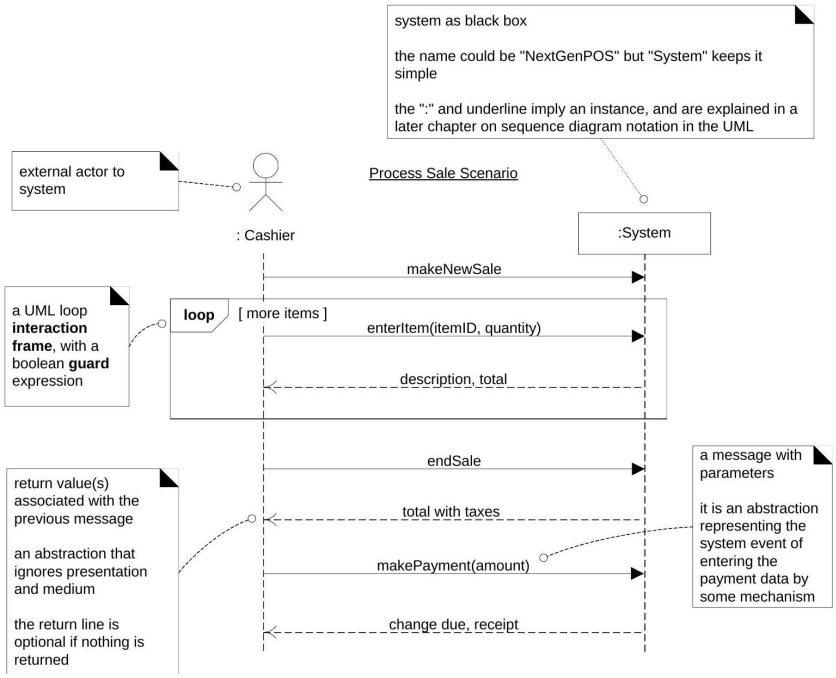
on sequence diagram notation in the UML



Systemoperation

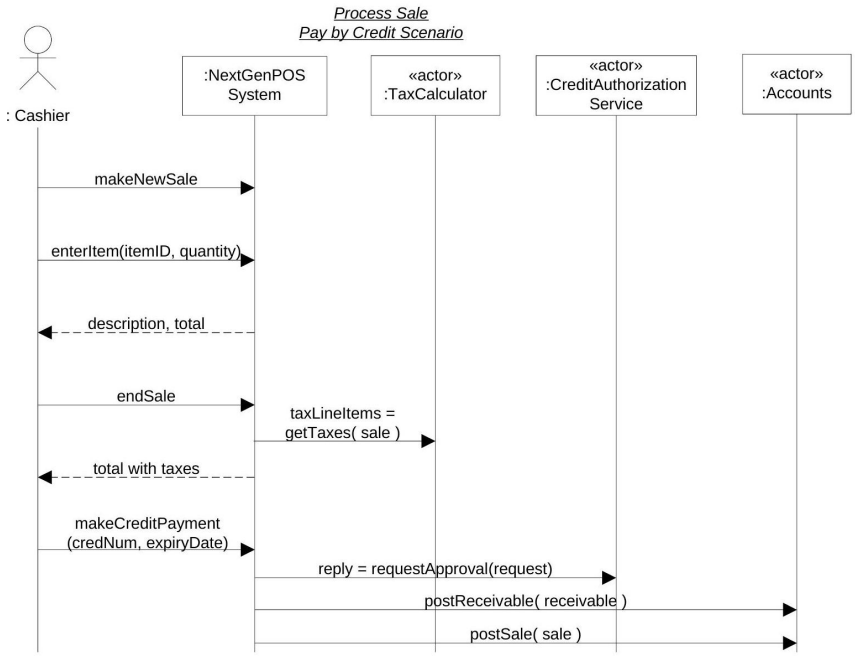
School of Engineering

- Formal wie ein Methodenaufruf
- Treffender Name, der die Absicht des Akteurs repräsentiert
- Evtl. mit Parametern
- Information, die für die Ausführung der Systemoperation nötig sind, aber noch nicht im System vorhanden sind
- Details zu den Parametern sollten im Glossar erläutert werden
- Durchgezogener Pfeil für Methodenaufruf
- Rückgabewert
- Kann fehlen, falls unwichtig
- Kein Methodenaufruf, sondern indirektes Update des UI (deshalb gestrichelte Linie)



SSD für Interaktionen zwischen Systemen

- SSD können auch Interaktionen zwischen SuD und externen unterstützenden System zeigen



Operation Contract

- Eine (System) Operation kann mit einem Vertrag noch genauer spezifiziert werden
- Name plus Parameterliste
- Vorbedingung
- Was muss zwingend erfüllt sein, damit Systemoperation aufgerufen werden kann
- Nachbedingung
- Was hat sich alles geändert im System nach Ausführung der Systemoperation
- Erstellte/gelöschte Instanzen, Assoziationen
- Geänderte Attribute
- Basiert auf Domänenmodell

Contract CO2: enterItem

- Operation:

```
enterItem(idemID: ItemID,
quantity: integer)
```

- Querverweis: UC Process Sale
- Vorbedingungen:
- Verkauf muss gestartet sein

  Zeitform beachten!

- Nachbedingungen
- SaleLineItem-Instanz sli (ist) erstellt
- sli mit aktueller Sale-Instanz verknüpft
- sli.quantity auf quantity gesetzt
- sli mit entsprechender ProductDescription verknüpft (gemäss itemID)

Operation Contract

- Wann Operation Contracts?
- Nur wenn aus einem Anwendungsfall nicht klar wird, was die Systemoperation genau machen muss
- Meist nur bei sehr komplizierten Operationen und/oder
- Wenn Entwicklung der Systemoperation ausgelagert wird (anderes Team, externe Entwickler)
- Erst gegen Ende des Meilensteins Lösungsarchitektur oder kurz vor Start des Designs der Systemoperation

Nutzen von SSD und Systemoperationen

- Systemoperationen definieren die Schnittstelle (API) des Systems
- Während dem Design
- wird das System ausgehend von den Systemoperationen entwickelt (anhand der Verträge)
- Das UI-Team kann parallel das UI entwickeln unter Verwendung der vereinbarten Systemoperationen (und ihren Verträgen)

- SSD können auch zur Darstellung der Kommunikation von Subsystemen verwendet werden (z.B. bei Client-Server-Architektur)
- Achtung Frameworks!
- UI- und andere Frameworks geben häufig gewisse Systemoperationen vor, die das System implementieren muss
- Sollte man bei den SSD bereits berücksichtigen

1. Wie erfasst man (dynamische) funktionale Anforderungen mit Use Cases
2. Wie schreibt man gute Use Cases
3. System-Sequenzdiagramme, Systemoperationen und Verträge
4. Wie erfasst man zusätzliche funktionale und nicht-funktionale Anforderungen
5. Wrap up und Ausblick

Weitere Anforderungen

- Die UCs beschreiben einen grossen Teil der funktionalen Anforderungen aus Benutzersicht
- Es gibt aber weitere funktionale und nicht-funktionale Anforderungen (Qualitätsanforderungen und Randbedingungen), die schlecht in UCs beschrieben werden können
- Diese werden als zusätzliche Anforderungsspezifikation formuliert (Supplementary Specification)
- Zusätzliche Anforderungen
- Können als Anforderungsstatement oder als User-Story (agile SWE) formuliert werden
- Bsp:
- Anforderungsstatement:
  „Das Kassensystem muss in weniger als 1 Minute aufgestartet sein"
- User-Story:

Als Kassier möchte ich, dass bei mehreren gleichen Artikeln der Einzelpreis, die Anzahl und der Gesamtpreis angezeigt werden, damit ich einen schnellen Überblick habe.

Weitere Anforderungen

- Anforderungsstatements
- Sollten als Anforderung formuliert werden
- Das System muss/soll mindestens/darf nicht...
- Sollten messbar/verifizierbar sein
- Sie müssen dem Auftraggeber irgendwann belegen, dass ihr System diese Anforderung erfüllt
- So wenig wie nötig
- Nur Anforderungen stellen, die auch wirklich von jemandem begründet gefordert werden
- Keine ersten Lösungsideen als Forderungen formulieren
- „Das System muss eine Web-App sein"

- User-Stories

- Sagen in einem Satz wer, was, warum fordert
- Erfüllen damit einige der Bedingungen automatisch
- Die anderen sollten aber auch bei UserStories erfüllt sein
- Messbarkeit/Verifizierbarkeit (z.B. durch Akzeptanzkriterien)

Weitere Anforderungen: FURPS+

- Checkliste für zusätzliche Anforderungen

- Functionality (Funktionalität)
- Features, Fähigkeiten, Sicherheit
- Usability (Gebrauchstauglichkeit)
- Siehe Usability-Anforderungen (LE02)
- Accessibility (Benutzer mit spez. Bedürfnissen)
- Reliability (Zuverlässigkeit)
- Fehlerrate, Wiederanlauffähigkeit, Vorhersagbarkeit, Datensicherung
- Performance (Performanz)

- Reaktionszeiten, Durchsatz, Genauigkeit, Verfügbarkeit, Ressourceneinsatz
- Supportability (Unterstützbarkeit)
- Anpassungsfähigkeit, Wartbarkeit, Internationalisierung, Konfigurierbarkeit
- Implementation
- HW, Betriebssysteme, Sprachen, Tests, Werkzeuge,..
- Interface
- Schnittstellen von ext. Systemen, Protokolle
- Operations
- Betriebliche Aspekte
- Packaging (Verpackung)
- Auslieferung physisch, logisch (Container, Plugin,..)
- Legal
- Lizenzen, rechtl. Rahmenbedingungen

Glossar

- Einfaches Glossar
- Definiert die Begriffe, die in diesem Projekt und im SW-Produkt verwendet werden
- Kann beliebige Elemente enthalten
- Wichtige Begriffe, Konzepte des Domänenmodells, Attribute, Parameter von Operationen
- Data Dictionary
- Definiert zusätzlich Datenformate, Wertebereiche, Validierungsregeln
- Empfehlungen Glossar
- So früh wie möglich mit Glossar beginnen, sobald der erste Begriff auftaucht, der nicht sofort allen klar ist
- Fortwährend aufdatieren, sobald weitere Begriffe, Konzepte etc. auftauchen, die nicht allen sofort klar sind
- In der Kommunikation untereinander, die Begriffe des Glossars verwenden

- Fördert Klarheit in der Kommunikation unter den Projektmitgliedern
- Für dieselbe Sache nur einen Begriff verwenden (wenn möglich)

1. Wie erfasst man (dynamische) funktionale Anforderungen mit Use Cases
2. Wie schreibt man gute Use Cases
3. System-Sequenzdiagramme, Systemoperationen und Verträge
4. Wie erfasst man zusätzliche funktionale und nicht-funktionale Anforderungen
5. Wrap-up und Ausblick

- Ausgehend von den Kontextszenarien werden die funktionalen Anforderungen immer mehr konkretisiert.
- Anwendungsfälle (Use Cases)
- Beschreiben die konkrete Interaktion der Akteure mit dem System (lösungsneutral)
- Systemsequenzdiagramme (System Sequence Diagram)
- Identifizieren die notwendigen Systemoperationen, die das System für die Anwendungsfälle zur Verfügung stellen muss
- Systemoperationen und Verträge (Operation Contracts)
- Spezifizieren die Systemoperationen im Detail

Ausblick

- In dieser Lerneinheit haben Sie gelernt, wie Sie «dynamische» funktionale Anforderungen aus den Interaktionsbedürfnissen der Benutzer ableiten können.
- In der nächsten Lerneinheit werden Sie lernen,
- wie Sie die Problemdomäne an sich im Detail verstehen und kommunizieren können.

Quellenverzeichnis

[1] M. Richter and M. D. Flückiger, Usability und UX kompakt: Produkte für Menschen, 4th ed. Springer Vieweg, 2016.
[2] Larman, C.: UML 2 und Patterns angewendet, mitp Professional, 2005
[3] Seidel, M. et al.: UML @ Classroom: Eine Einführung in die objektorientierte Modellierung, dpunkt.verlag, 2012
[4] Martin, R. C.: Clean Architecture: A Craftsman's Guide to Software Structure and Design, mitp Professional, 2018