Software-Engineering 3

Im folgenden sind die Lernziele aus den Unterlagen von Graessle/Schacher zusammengefasst. Sie dienen als Grundlage für die Prüfungen.

Requirements Engineering

Sie kennen

- die Bedeutung und das Wesen von Anforderungen
  - *Anforderungen beschreiben eine oder mehrere gewünschte Eigenschaften oder gewünschte Verhaltensweisen eines Systems. Anforderungen werden oft mit Zielen verwechselt.*
  - *Anforderung:*
    - beschreiben eine Eigenschaft oder Verhalt eines Systems
    - ist auf den Betrachtungsgegenstand gerichtet
    - Blickrichtung ist nach „innen“
  - *Ziel:*
    - Beschreibt eine Wirkung oder Folge die aus der Verwendung des Systems entsteht
    - ist auf das Umfeld des Betrachutngsgegenstands gerichtet
    - Blickrichtung nach „aussen“
- Möglichkeiten zum Festhalten von Anforderungen
  - *z.B. Word-, Excel-Datei, Flipchart, DB Repository etc.*
  - *Anforderungen an einem Ort verwalten und von dort verteilen! (am besten ein Repository benutzen)*
- verschiedene Techniken aus dem Requirements Engineering
  - *Gespräche mit Wissensträgern (intern und extern), Fragebögen, Interviews, Workshops*
  - *Beobachten der aktuellen Arbeit*
  - *Analyse vorhandener Systeme und Dokumente (intern und extern)*
  - *Validierung von Zwischenresultaten*
  - *Analyse von Unternehmensmodellen, insbesondere von Prozessen*
  - *GUI-Prototyping*

Sie können

- ein fachliches Glossar erstellen
  - *Mindestens die wichtigsten fachlichen Konzepte werden: gesammelt, mit einem eindeutigen Begriff in ein Glossar eingetragen, definiert.*
  - *Wird das fachliche Konzept in einer Anforderung verwendet so soll dies immer mit den eindeutigen Begriff aus dem Glossar geschehen.*
- textuelle Anforderungen von guter Qualität formulieren
  - *Die folgenden Tipps zur Verbesserung textueller Anforderungen haben sich in der Praxis bewährt:*
    - *atomare Anforderungen verwenden (nicht verschiedene Anforderungen zusammenfassen)*
      - *X "Die Rechnung muss am Bildschirm geprüft und ausgedruckt werden können."*
      - *√ "Die Rechnung muss am Bildschirm geprüft werden können."*
      - *√ "Die Rechnung muss ausgedruckt werden können."*
    - *fachliche Begriffe auf Glossar basieren*
      - *√ „Die <u>Kundenrechnung</u> muss ausgedruckt werden können."*
    - *deklarative Beschreibung mit Attribut "Modalität" erstellen*
      - *√"Das System erlaubt dem Sachbearbeiter, Kundenrechnungen auszudrucken." MUSS-Anforderung*

Advanced UML

Sie kennen

- die Bedeutung von Strukturmustern für die Validierung

  Muster (Pattern) erlauben es, ohne fachliches Wissen mögliche Probleme und Unstimmigkeiten in Modellen zu erkennen

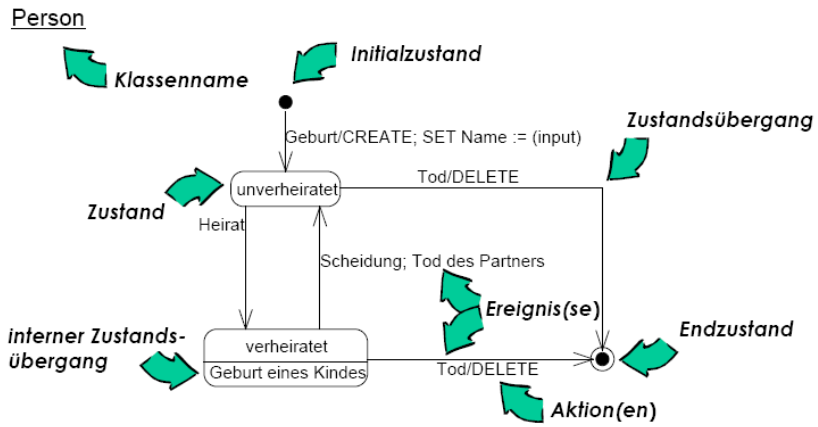
  Muster weisen auf ***mögliche*** Probleme und Unstimmigkeiten hin und beruhen auf gemachten Erfahrungen

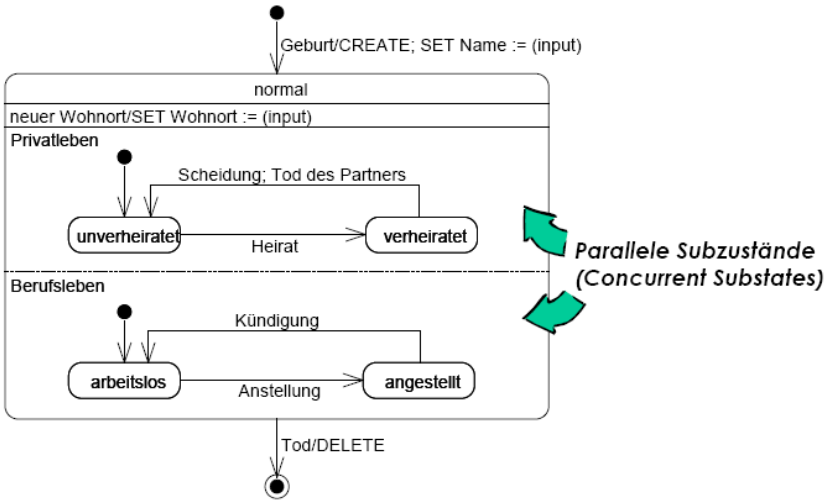
  - ***M:M Beziehung:*** *Bei einer M:M-Beziehung sollte hinterfragt werden, ob sich hinter der Beziehung nicht noch weitere Informationen verstecken*
  - ***Dreieck-Struktur****: Bei einer Dreieck-Struktur sollte hinterfragt werden, ob die direkte Master-Detail-Beziehung redundant ist*
  - ***Doppel-V-Struktur****: Wenn zwei Detailklassen direkt oder indirekt dieselben zwei Master besitzen sollte hinterfragt werden, ob zwischen den zwei Details auch eine Beziehung besteht.*
- die Bedeutung von Zustandsdiagrammen für die Verhaltensmodellierung
  - *In der Verhaltensmodellierung werden die für die Klassen relevanten Ereignisse identifiziert. Das Verhalten der einzelnen Objekte, in Reaktion auf diese Ereignisse, wird mit Zustandsdiagrammen spezifiziert:*
    - *Durch **Zustandsübergänge** und **Aktionen** werden die **Reaktionen** eines Objektes auf ein Ereignis spezifiziert (die Folgen des Ereignisses).*
    - *Zustandsdiagramme dienen zur Spezifikation des **Lebens eines Objektes** von dessen Geburt (Kreation) bis zu dessen Tod (Löschung), mit sämtlichen möglichen Zuständen dazwischen.*
    - *Aus dem Zustandsdiagramm können **zugelassene** und **nicht zugelassene Ereignissequenzen** abgeleitet werden.*
    - *Zustandsdiagramme enthalten somit die **dynamischen Geschäftsregeln**, die für ein Objekt.*
- erweiterte Konzepte der Zustandsmodellierung

  Top-Down/Bottom-Up Analyse, Death Models, Vererbung von Zustandsdiagrammen, Pseudo Ereignisse

Sie können

- Strukturmuster in Klassendiagrammen erkennen und hinterfragen
- fachlich vollständige Zustandsdiagramme für Klassen erstellen




Geburt/CREATE; SET Name := (input)
normal
neuer Wohnort/SET Wohnort := (input)
Privatleben
Scheidung; Tod des Partners
unverheiratet
Heirat
verheiratet
Berufsleben
Kündigung
arbeitslos
Anstellung
angestellt
Tod/DELETE
Parallele Subzustände
(Concurrent Substates)

OCL

You will know

- the typical usages of OCL
- *OCL is a side-effect free textual expression language, used to model:*
  - *data constraints*
  - *complex constraints*
  - *operation behaviour*
  - *derivations*
  - *complex computations*
- *On model elements*
  - *default values for attributes and parameters, derivation rules for attributes and directed associations, guards as well as change and time events in state diagrams, conditions in actions, constraints (invariants) on model elements, pre- and postconditions of operations, (pre- and postconditions of use cases)*
- *On models (and metamodels)*
  - *global constraints, target expressions for messages, queries, model transformations*
- the core concepts of OCL
  - *OCL supports the invocation of (predefined) operations of types, collections, and UML classes:*
    - *Original OCL*
    - *attributes of UML classes:*

      t.attribute
    - *roles of UML classes:*

      t.role
      t.role1.role2.…
    - *operations of types and UML classes:*

      t.operation(…)
      t::operation(…)
      (class operations)
    - *operations of collections:*

      coll->operation(…)

You will do

- some exercises with OCL
- You will be able to
  - decide when OCL is helpful/useful
  - enrich UML models using OCL in a pragmatic way

  OCL may be used for many purposes:
  • default values for attributes and parameters
  • derivation rules for attributes and directed associations
  • guards as well as change and time events in state diagrams
  • conditions in actions
  • constraints (invariants) on model elements
  • pre- and postconditions of operations
  • (pre- and postconditions of use cases)

  OCL makes UML models more precise
  - *enrich UML models using OCL in a pragmatic way*

Executable UML

You will know

- the typical usages of xUML
  - *Goals*
    - *Represent complex functionalities as easily as possible*
    - *Perform simulations for end users to validate specifications*
    - *Perform test cases to stabilize specifications and verify implementations*
    - *Use UML as far as possible*
  - *Typical Applications*
    - *Information Systems with complex time-dependent behavior*
    - *Information Systems with complex derivation rules and/or constrains*
    - *Complex safety-critical systems*
    - *UML education*
- the core concepts of Xuml
  - *Black-box modeling:*
    - *Are actors representing user roles*
  - *Glass-box modeling:*
    - *Are classes that are instantiable and destroyable, and so on…*
    - *Are furthermore represented by state diagrams*

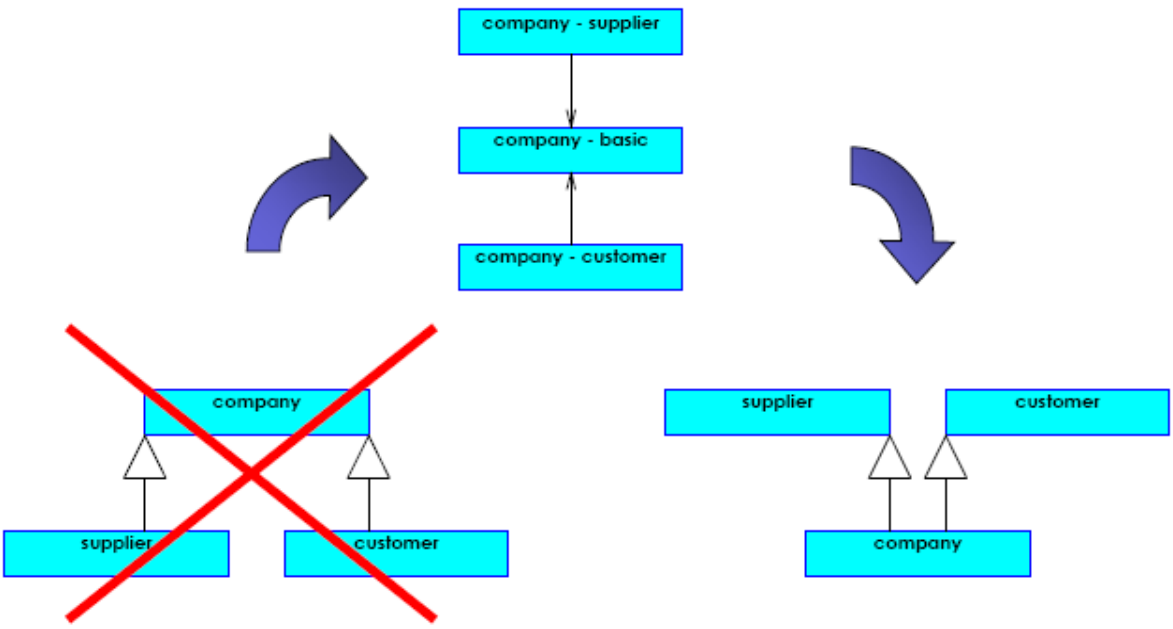
You will do

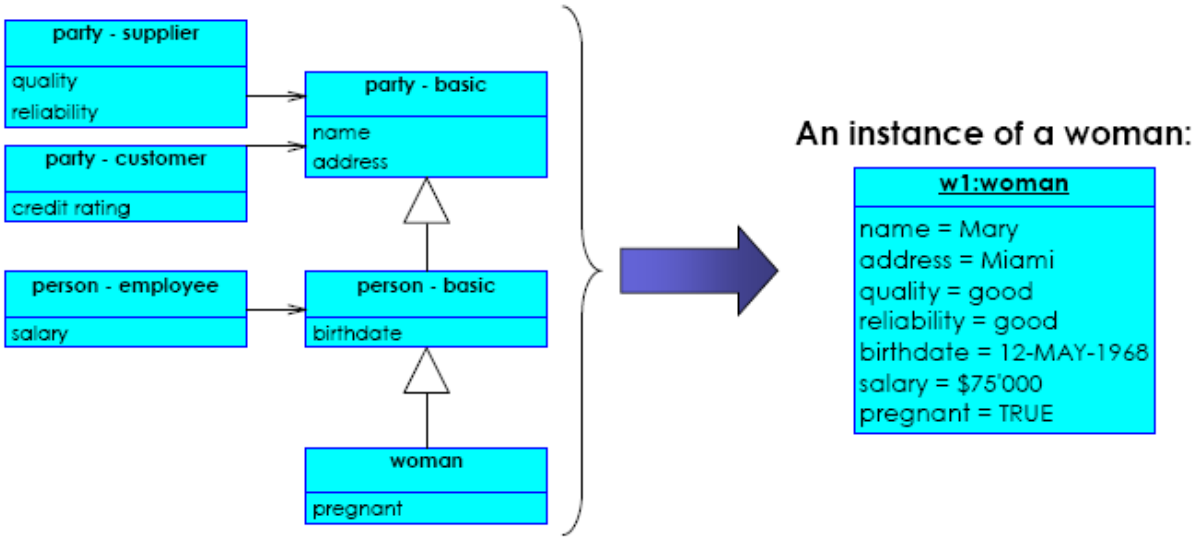
- some exercises with CASSANDRA/xUML
- You will be able to
  - "feel" the UML
  - model (complex) functionalities using xUML

Aspect Oriented Analysis

You will know

- common pitfalls with inheritance
  - *Modeling of different views on an object using inheritance is difficult. Inheritance might lead to different implementation of the same feature. An object can be in more than one state at the same time.*
  - *(Supplier is Customer example)*





  - *Für jeden Aspect kann ein eigenes Zustandsdiagramm modeliert werden*
- AOA as an approach to enterprise-level object modeling
  - *An object in an enterprise could be of interest to more than one party. However, different parties have different view of this object(e.g. Sales, human res...). These different views are best implemented using aspects.*
  - *Modeling customers, suppliers and employees*
- several implementation patterns for AOA
  - Aggregation, weaving, multiple inheritance, parallel specialisation, distributed implementation
  - guggsch du obe...

You will be able to

- decide, when AOA is applicable

  Wenn es mehr als nur eine Sicht auf ein und dasselbe Objekt gibt.
- understand inheritance much better

Metamodeling

You will know

- What a metamodel is
  - *a metamodel is a model that describes a model*
  - *the UML metamodel describes the Unified Modeling language, i.e. the UML language specification uses an UML class model (the UML metamodel) to describe the UML language*
  - *an UML case tool implements the UML metamodel (or parts of it) in its data base*
  - *UML users can safely ignore the UML metamodel (In fact don't mention it at all, it will only confuse them!)*
  - *A lot of constraints on the metamodel are defined in OCL.*
- What metamodels are used for
  - *[Wikipedia]Common uses for metamodels are:*
    - *As a schema for semantic data that needs to be exchanged or stored*
    - *As a language that supports a particular method or process*
    - *As a language to express additional semantics of existing information*
- How the UML metamodel can be extended
  - *A Profile is a kind of Package that extends a reference metamodel. The primary extension construct is the Stereotype*
  - *UML Stereotype*
    - *A stereotype defines how an existing metaclass may be extended and enables the use of platform or domain specific terminology or notation.*
    - *Stereotype is a kind of Class that extends Classes through Extensions.*
    - *Just like a class, a stereotype may have properties, which may be referred to as tag definitions.*
    - *When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.*

You will be able to

- Make your own extensions to UML

Model Driven Architecture

You will know

- The fundaments of OMG's Model Driven Architecture
  - *MDA bezeichnet die Verwendung von Modellen und Generatoren zur Verbesserung der Softwareentwicklung. Dieses Konzept ist nicht neu. Schon in den Anfangszeiten der Informatik stützte man sich auf Modelle und entwickelte Generatoren, um schematischen Quellcode daraus zu erzeugen. Es wird dabei nicht eine hunderprozentige Codegenerierung angestrebt.*

  - ***Definition**: „The MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. To this end, the MDA defines an architecture for models that provides a set of guidelines for structuring specifications expressed as models.“*
  - ***MDA Models***
    - ***CIM: Computation Independent Model***
      - *represents a model that purely describes the business*
      - *serves as the basis to decide about IT support of the business*
    - ***PIM: Platform Independent Model***
      - *represents a model that defines an IT system's functionality in implementation-independent form*
      - *expressed in UML*
    - ***PM: Platform Model***
      - *represents a metamodel of a particular platform (J2EE, .NET, RDB, etc.)*
      - *typically expressed as a UML profile*
    - ***PSM: Platform Specific Model***
      - *represents a model that specifies the mapping of a PIM onto a particular platform (J2EE, .NET, RDB, etc.)*
      - *expressed in UML and applies a UML platform profile*
    - ***PSI: Platform Specific Implementation (Code)***
      - *represents the final implementation artifacts (source code, configuration files, etc.) of an implemented IT system*
      - *expressed in a textual (programming) language*
  - ***MDA Roles***
    - ***Business/IT Analyst***
      - *Builds the CIM*
      - *Builds the PIM*
    - ***Architect***
      - *Defines the architecture*
      - *Defines coding style*
    - ***Designer***
      - *Understands the PIM and the architecture*
      - *Tags the PIM*
    - ***Transformation Builder***
      - *Implements M2M transformations*
      - *Implements M2T transformations*
- Approaches and issues to code generation
  - **Code Generation Issues**
    - **Name mangling**
      - Model names must be transformed into names that conform with the target language syntax

- **Type mapping**
  - Generic UML types must be mapped to types of target language
- **X% of 100% versus 100% of X%**
  - Generating everything of a little bit is better than generating a little bit of everything

- **Protected regions**
  - Incompletely generated components require manual completion, which must be protected at regeneration
- **Reverse/round trip engineering**
  - Incompletely generated components require manual completion, which may compromise compatibility with model and thus difficult resynchronization

You will be able to

- conceptually understand MDA tools
- judge different approaches

Geschäftsprozessmodellierung

Sie kennen

- Den Nutzen der Geschäftsprozessmodellierung

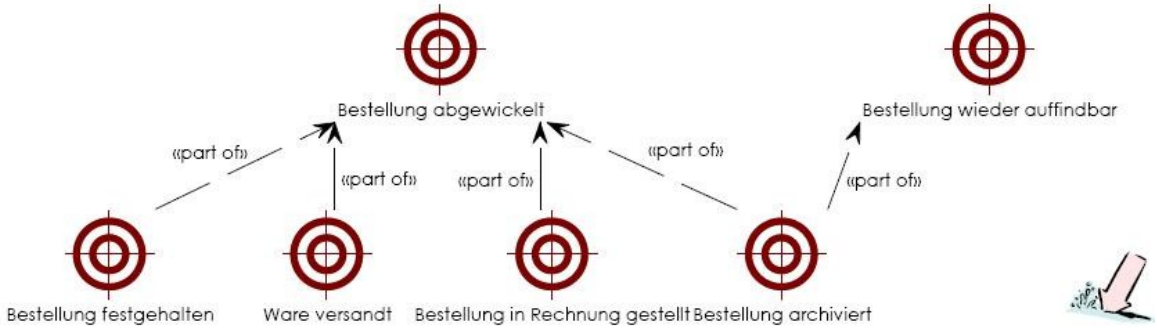
  Die ***Prozesse*** sind eine ***wertvolle Ressource*** eines Unternehmens, sie müssen ***gemanaged*** werden

  - *Mit der Modellierung von Geschäftsprozessen werden ganz unterschiedliche Ziele verfolgt. Geschäftsprozesse werden beispielsweise modelliert, weil*
    - *bestehende Prozesse dokumentiert werden müssen (z.B. wegen ISO 9001)*
    - *bestehende Prozesse überwacht werden müssen (Durchlaufzeiten, Fehlerquoten, etc.)*
    - *Verantwortlichkeiten für Prozesse und Aktivitäten festgelegt werden müssen*
    - *bestehende Prozesse verbessert/vereinheitlicht werden müssen*
    - *bestehende Prozesse zur Verbesserung simuliert werden müssen*
    - *grundsätzlich neue Prozesse identifiziert und gestaltet werden müssen*
    - *das Zusammenspiel verschiedener Prozesse gestaltet werden muss und weil*
    - *bestehende Prozesse als Grundlage für die Gestaltung der IT-Unterstützung verwendet werden sollen*
    - *bestehende Prozesse so weit wie möglich automatisiert werden müssen*
- Die Begriffe der Geschäftsprozessmodellierung
  - ***Prozessor***
    Ein Prozess wird durch einen Prozessor ausgeführt. Der Prozessor kann ein Mensch oder eine Maschine sein.
  - ***Geschäftsaktivität***
    Ein einzelner Schritt eines Geschäftsprozesses, die typischerweise durch einen Prozessor in entsprechender Rolle ausgeführt wird.
  - ***Workflow***
    Die vollständige oder teilweise Automatisierung eines Geschäftsprozesses, bei der Dokumente, Informationen oder Aufgaben von einem Teilnehmer gemäss einer Menge von Regeln zu einem anderen zur Weiterbearbeitung weitergeleitet werden.
  - ***Modell***
    Eine auf bestimmte Zwecke ausgerichtete vereinfachende Beschreibung der Wirklichkeit
    - ***Deskriptive Modelle***
      werden zur Analyse eines existierenden Systems verwendet
    - ***Präskriptive Modelle***
      werden zur Gestaltung eines neuen Systems verwendet

- ***Geschäftsprozessmodell***
  *Eine **zweck**orientierte, vereinfachte Abbildung eines oder mehrerer Geschäftsprozesse*

- Klassische detaillierte Prozessmodelle
  - *Klassische detaillierte Prozessmodelle beschreiben die Ausführung von Prozessen:*
    - *Sie optimieren Effizienz und Produktivität.*
    - *Sie enthalten viele Regeln und stellen ihre Einhaltung sicher.*
    - *Sie sind ein Instrument des Qualitätsmanagements.*
    - *Sie eignen sich für Routinearbeiten, die häufig wiederholt werden.*
    - *Sie können mit Hilfe von Workflow Management Systemen automatisiert werden.*
  - ***Grenzen detaillierter Prozessmodelle***
    - *Es gibt in der Realität nicht nur repetitive Routinearbeiten, sondern eben auch wissensintensive Geschäftsprozesse. Diese*
      - *sind oft lang laufend*
      - *sind im Ablauf variabel*
      - *haben viele Ausnahmen*
      - *stellen hohe Anforderungen an Fähigkeiten und Expertise der ausführenden Person*
    - *Für solche Prozesse ist die klassische Darstellung in einem detaillierten Prozessmodell ungeeignet.*
      - *Beispiel: Betreuungsprozess für vermögende Privatkunden*

- Leichte, regelbasierte Prozessmodelle
  - *Um wissensintensive Geschäftsprozesse zu modellieren, bevorzugen wir leichte Prozessmodelle.*
    - *leicht bedeutet:*
      - *nur so wenige Details wie nötig (aber nicht weniger)*
      - *nur die wichtigsten Kontroll- und Datenflüsse*
      - *möglichst wenig Regeln in Prozessmodellen zu "betonieren"*
  - *Dadurch können die Modelle auch in der Praxis gepflegt werden.*
  - *Leichte Prozessmodelle lassen sich zielorientiert aufbauen.*
    - *Eine Geschäftsaktivität*
    - *darf nicht zum Selbstzweck ausgeführt werden*
    - *muss ein Geschäftsziel verfolgen*
  - *Die Geschäftsziele können ebenfalls modelliert werden. Hier ein*
    - *Beispiel zur Bestellungsabwicklung:*



- Den Unterschied zwischen den detaillierten und leichten Prozessmodellen
  - *Leichte Prozessmodelle sind Ziel orientiert-> jeder Aktivität muss ein Geschäftsziel verfolgen.*
  - *Schwere Prozessmodelle sind sehr Detailliert und enthalten viele Regeln, quasi mehr in der Form einer Anleitung. Dies kann benutzt werden für Automatisierung oder QA.*

- Die Vor- und Nachteile der detaillierten und leichten Prozessmodelle
  - *Klassische Prozesse*
    - *Vorteil:*
      - *kann als "Bedienungsanleitung" verwendet werden*
      - *nichts wird vergessen (?)*
      - *kann von Ausführenden genau nachvollzogen werden*
      - *dokumentiert (nahezu) IST-Zustand*
    - *Nachteil:*
      - *Intention des Prozesses ist nicht einfach ersichtlich*
      - *Änderungen an Implementationsdetails müssen in allen betroffenen Diagrammen nachgeführt werden*
      - *unübersichtlich wenn alle Varianten auf einem Diagramm*
      - *Gefahr von uneinheitlichen Abläufen, wenn jede Variante einzeln aufgezeichnet wird: der Blick fürs Gemeinsame geht verloren*
  - *Leichte Prozesse*
    - *Vorteil:*
      - *auch für komplexe Themen lassen sich Prozessdiagramme übersichtlich halten*
      - *ähnliche Abläufe die sich nur in den Entscheiden unterscheiden lassen sich zu einem Prozess zusammenfassen*
      - *Entscheidungen können als Regeln angepasst werden, eine Diagrammanpassung entfällt*
      - *die meisten "Prozessänderungen " sind Änderungen an Entscheidungen, nicht am Ablauf deshalb weniger Änderungen an Diagrammen*
    - *Nachteil:*
      - *Um den Prozess im Detail nachvollziehen zu können, müssen weitere Dokumente beigezogen werden, die zugreifbar sein müssen.*
      - *Prozessreviews mit Prozess ausführenden sind schwieriger, wenn diese die etwas höhere Abstraktion nicht gewohnt sind.*

Sie werden

- Als Übung ein Prozessmodell aus einer Fallstudie erstellen

Business Rules Ansatz

Sie kennen…

- die grundlegende Philosophie des Business Rules Ansatzes (BRA)
  *Das **Business**, nicht Technologie muss die **treibende Kraft für die ITEntwicklung** sein.*

  ***Eine Menge von Techniken und Technologien, die sowohl Business** Engineering als auch IT Systementwicklung abdecken, mit dem Ziel, agile Geschäfts- und IT-Systeme zu betreiben, welche direkt durch Geschäftsleute kontrolliert werden.*

  ***Business Rules Mantra**:*
  ***Regeln** bauen auf **Fakttypen** auf, Fakttypen bauen auf **Konzepten** auf*

  - *Geschäftswissen muss **transparent** sein, d.h. es muss*
    - *externalisiert, bzw. **explizit***
    - *für das Business **verständlich***
    - *flexibel **auffindbar** sein.*
  - *Geschäftswissen muss **anpassbar** sein, d.h. es muss*
    - *durch das Business **diskutierbar***
    - *einfach **änderbar***
    - *in sich **konsistent** sein.*
  - *Geschäftswissen muss **automatisierbar** sein, d.h. es muss für IT-Systeme*
    - *anwendbar, bzw. **ausführbar***
    - *in der Anwendung **überwachbar***
    - *verständlich **erklärbar** sein.*

- Sinn und Zweck eines Faktenmodels mit Vokabular
  - *Das Faktenmodell (auch "**Vokabular**" genannt) bildet die Grundlage für die Formulierung von Geschäftsregeln. Es definiert*
    - *In der betrachteten Domäne wichtige **Konzepte***
    - *Gängige Begriffe für diese Konzepte*
    - *Wichtige **mögliche** und **interessante** Aussagen (**Fakttypen**) über diese Konzepte*
  - *Das Faktenmodell wird typischerweise als **Faktendiagramm** dargestellt. Wir verwenden hier eine erweiterte Version der UML-Notation (Unified Modeling Language), welche ein weltweiter Standard der OMG (Object Management Group) ist.*
- verschiedene Arten von Geschäftsregeln
  - *Geschäftsregeln sind **fachliche Aussagen**, die **immer wahr sind** oder immer wahr sein **sollten**.*
    - *Beispiele:*
      - *Eine Rechnung muss innerhalb von 30 Tagen bezahlt sein*
      - *Der Gesamtbetrag einer Bestellung berechnet sich aus der Summe der einzelnen Elemente abzüglich einem allfälligen Rabatt*
      - *Wenn ein Kunde mindestens 20 Einheiten eines Produktes bestellt, erhält er 10% Rabatt*
      - *Falls ein Kunde ein "Fass Guinness" bestellt, muss ihm eine "Flasche Hürlimann" offeriert werden*
  - *Arten von Geschäftsregeln*
    - ***Ableitungsregeln***
      - *Ableitungsregeln sind Regeln, die Aussagen aus anderen (ggf. zusammengesetzten) Aussagen ableiten. Beispiele:*

        - *"Eine Person ist ein bevorzugter Kunde, falls ihr Umsatz in den vergangenen 12 Monaten mindestens 500 CHF ist und sie nicht auf der schwarzen Liste ist."*
        - *"Eine Person ist auf der schwarzen Liste, falls eine an sie gelieferte Bestellung nicht innerhalb von 30 Tagen bezahlt wurde."*
    - ***Einschränkungen***
        - *Einschränkungen sind Ausdrücke, die immer wahr sein müssen.*
          Beispiele:
            - *"Ein Kunde darf nie seine Kreditlimite überschreiten."*
            - *"Jede aktive Bestellung darf nur aktive Produkte enthalten."*

    - ***Prozessregeln***
        - *Prozessregeln sind Regeln, die eine bestimmte Aktion kontrollieren, falls ein bestimmtes Ereignis eintritt und/oder eine bestimmte Bedingung wahr wird.*
          Beispiele:
            - *"Wenn ein neuer Kunde eine Bestellung aufgibt, muss seinE Kreditwürdigkeit geprüft werden."*
            - *"Guinness muss empfohlen werden, falls Heineken bestellt wird."*
- verschiedene Darstellungen von Geschäftsregeln
    - ***Formulierung von Geschäftsregeln***
        - *Geschäftsregeln sollten idealerweise so formuliert sein, dass sie*
            - *für den (**Business-)Menschen verständlich** sind*
                - *in möglichst natürlicher Sprache*
                - *in Deutsch, Französisch, Italienisch, Englisch, …*
            - ***deklarativ** und **nicht prozedural** sind*
                - *das Ziel, nicht den Weg formulieren ("what not how")*
                - *keinen Anwendungszeitpunkt und keine Reihenfolge spezifizieren*
            - *für die **Maschine** verständlich sind*
                - *präzise, d.h. formal*
                - *effizient ausführbar*
    - ***Darstellungsmöglichkeiten für Geschäftsregeln***

      Geschäftsregeln können auf verschiedene Arten formuliert werden:
        - *in Deutsch*
        - *in formalem Deutsch*
        - *als Entscheidungstabellen*
        - *als Entscheidungsbäume*
        - *als Klassifikationsbäume*
        - *in formaler Logik*
        - *in einer deklarativen Computersprache (z.B. SQL)*
        - *in einer prozeduralen Computersprache (z.B. PL/SQL, Java)*
- wichtige technische Realisations-Varianten für BRA-Applikationen
    - *Service-Architecture*
    - *Layer-Architecture*
    - *Database-Architecture*
    - *Push-/Pull-Interface*

Sie können…

- ein Faktenmodel lesen
- einfache Geschäftsregeln in formalem Deutsch formulieren
- grundsätzliche Architektur-Varianten unterscheiden