



Universität Stuttgart

Institut für Software Technologie (ISTE)

Abteilung Software Engineering

Einführung in die Softwaretechnik (EST)

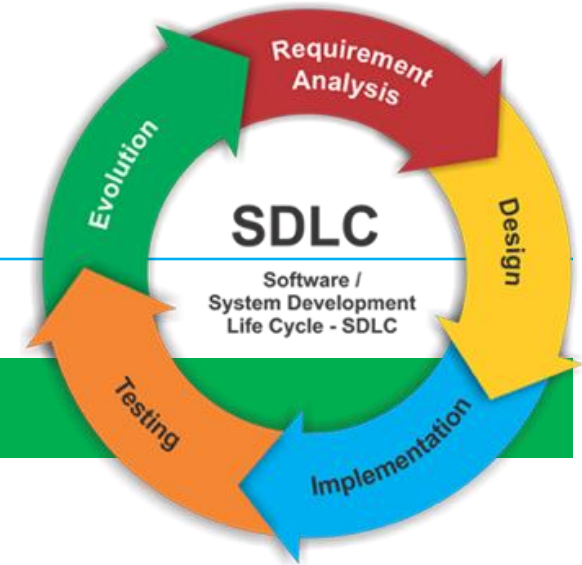
Übung 2: Spezifikation

27.04.2020

Justus Bogner

Vorläufige Übungstermine

Übung	Termin	Thema
Ü0	20.04.2020	Organisation & Analyse
Ü1	27.04.2020	Spezifikation
Ü2	11.05.2020	Testfall-Erstellung
Ü3	25.05.2020	Implementierung
Ü4	15.06.2020	Wartung & Evolution
Ü5	22.06.2020	Architektur & Qualitätssicherung
Ü6	30.06.2020	Langlebige Software
Ü7	14.07.2020	Prüfungsvorbereitung



Vorläufige Abgabetermine der Übungsblätter

Blatt	Thema	Veröffentlichung	Abgabe
ÜB1	Analyse	17.04.2020	24.04.2020
ÜB2	Spezifikation	27.04.2020	06.05.2020
ÜB3	Testfall-Erstellung	11.05.2020	19.05.2020
ÜB4	Implementierung	25.05.2020	09.06.2020
ÜB5	Wartung & Evolution	15.06.2020	21.06.2020
ÜB6	Architektur & Qualitätssicherung	22.06.2020	28.06.2020

Letztes Übungsblatt 1: Analyse

Unklarheiten in der Systembeschreibung:

- Attribute der Domain Entities (Buch, Buchkopie, Kunde)?
- Datei-Import (Format, Attribute, etc.)?
- Input für Löschen und Suche (welche IDs und nach was kann gesucht werden?)
- Wie sieht die Ausgabe bei Suchen oder Reports aus?
- Genaue Kriterien für die Ausleihe?
- Bezahlvorgänge innerhalb oder außerhalb des Systems?
- Häufigste Prozesse für effiziente Menüanordnung?
- Andere nicht-funktionale Qualitätsattribute?

Übungsblatt 2: Spezifikation

Universität Stuttgart, Institut für Softwaretechnologie

Einführung in die Softwaretechnik (EST) / Grundlagen des Software Engineerings (GSE)

Übung, Sommersemester 2020

Justus Bogner<justus.bogner@iste.uni-stuttgart.de>

Übungsblatt 02: Spezifikation

Veröffentlichung: 27.04.2020

Abgabe: 06.05.2020

Mit Hilfe der durch die Kundenbefragung erhaltenen Antworten haben Sie ihr Verständnis der gewünschten Software verfeinert. Diese Antworten sowie die initiale Systembeschreibung sind die Basis für die jetzige Aufgabe.

Aufgabe

Im zweiten Schritt sollen Sie für den Kunden zwei Artefakte erstellen, um Ihr Angebot vorzustellen.

→ Sie haben den Kunden befragt und sollen nun eine Software Requirements Specification (SRS) und einen UI-Prototyp erstellen (Abgabe in GitLab).

Übungsblatt 2: Wichtige Konzepte

- Spezifikation (SRS)
 - Domain Model / Class Diagram
 - Use Case Diagram (alle Use Cases des Systems)
 - Use Case Table (für einen einzelnen Use Case)
- Java UI-Prototyp
 - Git / GitLab
 - Console Interaction

Software Requirements Specification (SRS)

- Im Deutschen: Spezifikation oder Pflichtenheft
- Ziel: Gemeinsames Verständnis der Anforderung schaffen und dokumentieren
- [IEEE 830](#): Recommended Practice for Software Requirements Specifications
 - Ausführliche Gliederungsvorschlag
 - Mittlerweile abgelöst durch [ISO/IEC/IEEE 29148](#)
- Oft Teil eines Vertrags zwischen Kunde und Dienstleister
- Für dieses Übungsblatt: Erweiterung der SRS durch Entwurfselemente
- Nutzen Sie eines der bereit gestellten [Templates](#) (Word oder LaTeX)!

Domain Model / UML Class Diagram

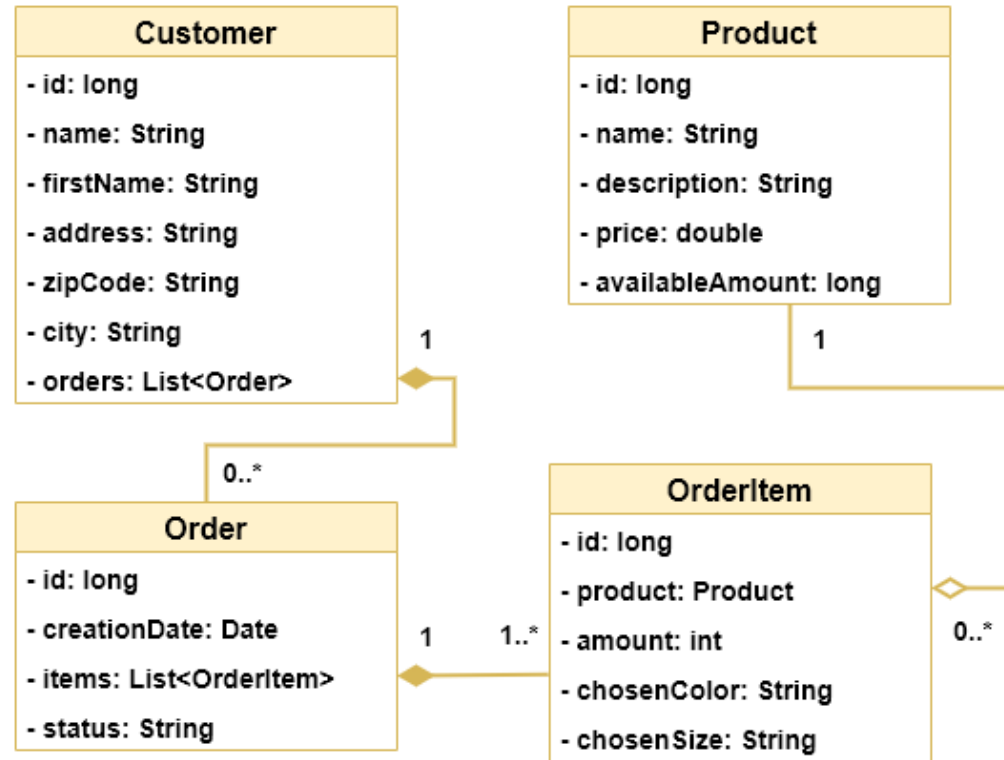
- Großteil existierender Softwaresysteme manipuliert Daten (CRUD)
- Geschäftsprozesse-unterstützende Software bildet Konzepte aus der realen Welt ab
- Objekt-orientiertes Design: Klassen (classes) und Objekte (objects / instances)
- Domain Model [1][2] spezifiziert wichtigste Konzepte der abzubildenden Prozesse
 - Core Entities und ihre Beziehungen
 - Objekte aus der Geschäftsdomäne (keine reinen Softwarekonzepte!)
 - Kommunikationsmittel zwischen Kunde und Entwicklern
 - Gut gewählte Namen sind wichtig (“ubiquitous language” [1])

[1] E. Evans, *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[2] M. Fowler, *Patterns of Enterprise Application Architecture*. Pearson Education, 2002.

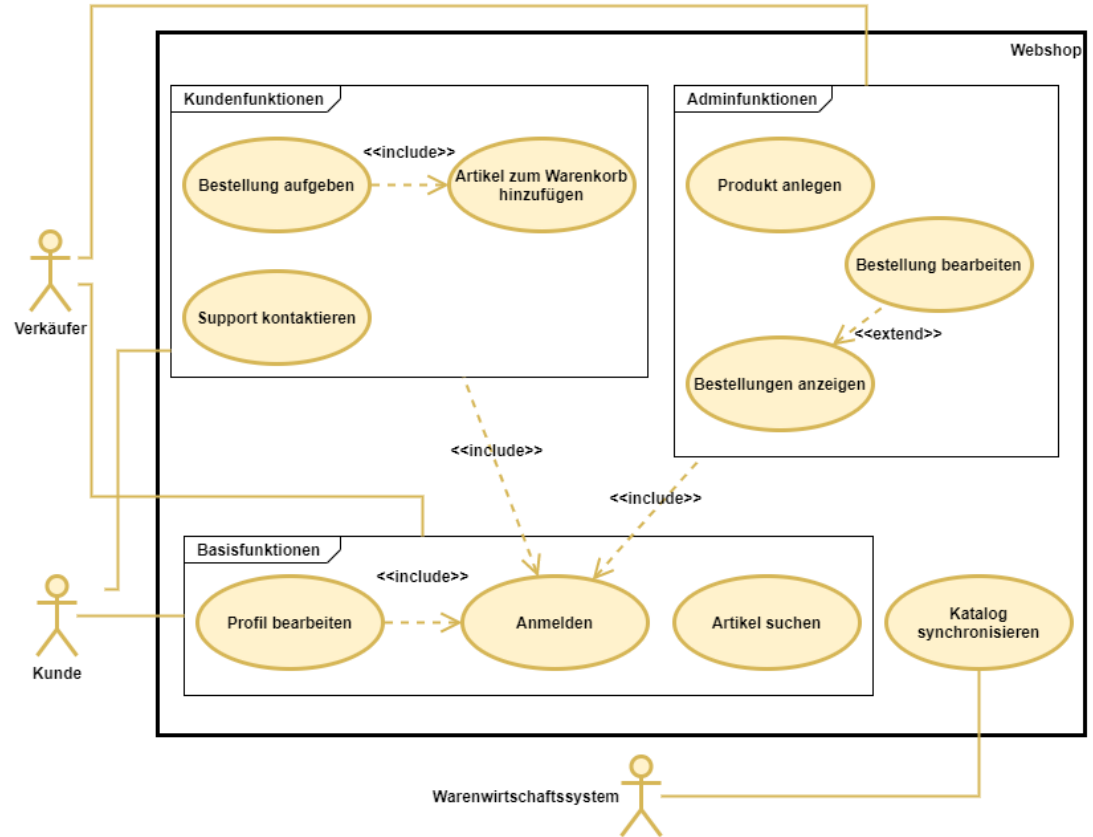
Domain Model Beispiel

- Statische Darstellung der Domain Entities mit Attributen, Methoden und ihren Beziehungen als UML Class Diagram
- Beispiel Onlineshop: Verkauf und Versand von Artikeln über eine Webseite
- Referenz:
<https://www.uml-diagrams.org/class-diagrams-overview.html>



Use Case Diagram

- Verhaltensorientierte Abbildung der Systemfunktionalität (Überblick)
- Externe Interaktionssicht (Akteure, i.e. Nutzer oder Nachbarsysteme)
- Beispiel Onlineshop: Verkauf und Versand von Artikeln über eine Webseite
- Referenz:
<https://www.uml-diagrams.org/use-case-diagrams.html>



Use Case Table

- Systematische Beschreibung zu einem bestimmten Use Case (Details und Ablauf)
- Beispiel Onlineshop: Profil bearbeiten

Name:	Profil bearbeiten
Ziel:	Kunde ändert seine persönliche Daten über das Profil
Akteure:	Kunde
Vorbedingung:	Kunde hat Account im Webshop und ist angemeldet
Nachbedingung:	Geänderte Daten des Kunden sind gespeichert, Erfolgsmeldung wird angezeigt, aktualisierte Profilansicht wird angezeigt
Nachbedingung im Sonderfall:	Kundendaten wurden nicht aktualisiert, Fehlermeldung wird angezeigt, weiterhin im Bearbeiten-Modus des Profils
Normalablauf:	1. Auswahl des Menüpunkts "Profil" 2. Klick auf Button "Profil aktualisieren" 3. Eingabe der zu ändernden Daten 4. Klick auf Button "Speichern" 5. Validierung der neuen Daten (z.B. korrekte E-Mail-Adresse) 6. Erfolgsmeldung anzeigen 7. Navigation zur normalen Profilansicht
Sonderfall 5a:	5a.1 Anzeige des genauen Fehlergrunds 5a.2 Weiter mit Punkt 3



Git




- Große Softwaresysteme bestehen aus sehr viel Source Code
 - Versionsmanagement?
 - Koordination der Entwickler untereinander? Konflikte?
- → Version Control Systems (VCS)
- Git als De-facto-Standard (dezentralisiert, Branch-Modell)
- Nutzung über die Konsole, aber auch über Webplattformen oder Desktop-Clients
 - GitHub: <https://github.com>
 - GitLab: <https://gitlab.com>
- Referenz: <https://git-scm.com/book/en/v2>










GitLab benutzen (1)


EST-GSE Exercise SS20 > team-01 > Details






**team-01** 
Project ID: 297






  Star 0  Fork 0


 1 Commit  1 Branch  0 Tags  154 KB Files


master  team-01 /  


 **Initial commit**
JREB authored 5 days ago

 README  Add LICENSE  Add CHANGELOG  Add CONTRIBUTING  En

History  Find file  Web IDE   Clone 

Clone with SSH
git@sopra.informatik.uni-stuttg 

Clone with HTTPS
https://sopra.informatik.uni-st 

 Copy URL

```
JREB@JREBs-PC MINGW64 /c/dev/workspace
$ git clone git@sopra.informatik.uni-stuttgart.de:est-gse-ss20/team-01.git
Cloning into 'team-01'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

GitLab benutzen (2)

```
JREB@JREBs-PC MINGW64 /c/dev/workspace/team-01 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        src/

nothing added to commit but untracked files present (use "git add" to track)

JREB@JREBs-PC MINGW64 /c/dev/workspace/team-01 (master)
$ git add .

JREB@JREBs-PC MINGW64 /c/dev/workspace/team-01 (master)
$ git commit -m "Added UI prototype for library software"
[master 068ec79] Added UI prototype for library software
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/LibrarySoftware.java

JREB@JREBs-PC MINGW64 /c/dev/workspace/team-01 (master)
$ git push origin master
```

Änderungen machen → git add → git commit → git push

Java Console Interaction

- Text-basierte Benutzerschnittstelle (Console)
- Ausgabe von Informationen über `System.out.println()`
- Navigation über Eingabe von Zahlen oder Text
- Empfohlene Java APIs: `java.io.Console` oder `java.util.Scanner`
- Auswahloptionen anzeigen und aussagekräftig benennen
- Dauerschleife für Input (bis zur Abbruchbedingung)
- Häufige Outputs in Methoden auslagern (z.B. Hauptmenü)

Java Console Interaction

```
JREB@JREBS-PC MINGW64 /c/dev/workspace/team-01/src
$ javac catpicmanagement/CatPicApp.java && java catpicmanagement/CatPicApp
```

Welcome to the CatPicManager!

Please select one of the following options. Type 'q' to exit.

1. Import CSV file
2. Upload cat pic
3. Delete cat pic
4. Share cat pic

Your selection: 2

Please specify the path to the cat pic: /c/pics/cat01.jpg

Uploading /c/pics/cat01.jpg...

Cat pic successfully uploaded!

Please select one of the following options. Type 'q' to exit.

1. Import CSV file
2. Upload cat pic
3. Delete cat pic
4. Share cat pic

Your selection:

```
package catpicmanagement;

import java.io.Console;

public class CatPicApp {

    Run | Debug
    public static void main(String[] args) {

        System.out.println("\nWelcome to the CatPicManager!");

        displayMainMenu();

        Console console = System.console();
```


Schlusswort zum abzugebenden Programmcode

- Schreiben Sie Source Code auf **Englisch!**
- Alle Bezeichner (Variablen, Methoden, Packages, etc.) und Kommentare
- Tutoren haben Anweisung, notfalls Punktabzug zu geben



Universität Stuttgart

Vielen Dank!



Justus Bogner, PostDoctoral Researcher

E-Mail justus.bogner@iste.uni-stuttgart.de

Telefon +49 (0) 711 685-88306

Fax +49 (0) 711 685-88380

Universität Stuttgart

Institut für Softwaretechnologie (ISTE) – Abteilung Software Engineering

Universitätsstraße 38

70569 Stuttgart

<https://www.iste.uni-stuttgart.de/institute/team/Bogner-00002/>

https://www.researchgate.net/profile/Justus_Bogner