0. Einführung

Ziel: Ein Softwaresystem in Teamarbeit entwickeln (möglichst gut, zusammenhängend über zwei Semester)

Plenum: Meilensteininhalte, Phasenmodell, OO-Analyse, Pflichtenheft, studentische Präsentationen
Im nächsten Semester: OO-Entwurf, Datenbankanbindung, Implementierung.

Wöchentliche Rücksprachen: Diskussion, Meilensteinabgaben auf Papier, Abnahme der Technik-Prototypen

Literatur: **Heide Balzert**: Lehrbuch der Objektmodellierung. Analyse und Entwurf, Spektrum Ak.Vlg.1999
Eric Evans: Domain-Driven Design, Addison-Wesley, 2003
Johannes Siedersleben: Moderne Softwarearchitektur, dpunkt.verlag, 2004
Techniküberblick: JavaStarter: Sonderheft von JavaMagazin, 20.06.2006, 9,80 €, http://javastarter.de/
oder gleichartig: iX KOMPAKT; Java. Java auf einen Blick, 1/2009, 130pp., 9,90 €

Werkzeuge: Für **UML-Modellierung**: Allgemeines Zeichenwerkzeug oder dediziertes UML-Werkzeug, siehe http://www.tfh-berlin.de/~knabe/java/UmlDiagramsInText.shtml. Für **Oberflächenprototyp**: Aktuelle Java-IDE für Swing oder HTML-Editor für JSP/JSF.

0.1 Was für Projekte?

Gruppengröße: 4-5 Personen möglichst gleichen Niveaus
Anwendung: Typischerweise mit Aspekten Buchungen, Persistenz, Multi-User/Multi-Tasking, Verteilung.
Analyse-Klassendiagramm: max. 5 persistente Klassen, max. eine viele:viele-Assoziation
Softwaregröße: ca. 5-10 Klassen pro Person à 1-20 Operationen: 5-10 Tausend LoC gesamt

SW-Architektur:

Benutzeroberfläche (user interface)	ui
Fachkonzept (business logic)	lg
Datenbasis + Dienste (utilities)	db, ut

Programmiersprache: objektorientiert, informatisch sauber: Scala, Java mit AspectJ, C#, C++

Nach Belegung provisorischer **Zusammenschluss** zu Teams. Für die Abgabe der Meilensteine vereinbart jedes Team mit mir einen regelmäßigen **Termin** im Rahmen der Rücksprachezeiten. Darüberhinaus müssen Sie als Team eine wöchentliche, **interne Projektsitzung** durchführen.

1. Meilensteinabgabe in Übung
Projektstudie: 1 A4-Seite Grobkonzept, gewünschte Basistechniken, soweit beurteilbar. **Teamprofil**: 1 A4-Seite Personenliste mit jeweiliger Erfahrungsangabe in Basistechniken, sowie Wunschtechniken. Basistechniken sind: DB-Zugriff z.B. JPA/Hibernate | iBatis mit RDBMS; UI z.B. Java ServerFaces | Swing | SWT; DI-Container z.B. Spring; Systemverteilung z.B. HTTP/RMI/CORBA/REST sowie AspectJ, Java-Reflection, weitere Fremdsoftware, verteilte Entwicklung mit Subversion, Entwicklungsumgebung z.B. Eclipse/IntelliJ IDEA; Build-Tool Maven

0.2 Warum Software-Engineering (SE)?

SE: Ingenieurmäßige Erstellung großer Softwaresysteme.

Welche Probleme bei großen SWS (z.B. 30 MA x 2 Jahre)?

- Arbeitsteilung, Mitarbeiter-Kommunikation
- Mitarbeiter-Fluktuation
- Unüberschaubarkeit
- Kostenabschätzung
- Wartungskosten

Qualitätskriterium „es läuft" reicht nicht aus, „es ist wartbar" (leicht änderbar) zusätzlich erforderlich.

Wechsel von „Kunst des Programmierens" (D.E. Knuth) zu „Software Engineering" (1968 NATO-Fachtagung in Garmisch).

Viele, z. T. einander widersprechende SE-Projektmodelle; dennoch Problemdruck, methodisch vorzugehen.

0.3 Das Phasenmodell (nach Helmut Balzert, alias Wasserfallmodell) als Projektmodell

- Zeitlicher Rahmen für Erstellung großer Softwaresysteme
- Jeder Phase zuordnen:
  - ➔ Zerlegung in Teilaufgaben mit Aufwandsschätzungen
  - ➔ Ergebnis
  - ➔ Verantwortlichkeiten
  - ➔ Ressourcen (Personen, Geräte u.a.)

Planung → Definition → Entwurf → Implementierung → Test+Abnahme → Wartung+Pflege
------dieses Sem.------ ----------nächstes Semester---------- ------privat-------

Begleitend: Projektführung und Qualitätssicherung
Kritik: Nicht alles synchron, Iterationen nötig, Überforderung des Menschen durch detaillierte Dokumente

0.3.1 Planungsphase

- Erstellung eines Grobkonzepts (Was soll das System leisten?)
- Feststellung der Machbarkeit (wirtschaftlich, technisch, personell, zeitlich)

Ergebnis: Projektstudie, Entscheidung: Ja/Nein.

0.3.2 Definitionsphase

Was soll das System aus Kundensicht genau leisten?
Qualitative Anforderungen (Benutzerfunktionen, Schnittstellen nach außen)
Quantitative Anforderungen (Mengengerüst, Zeitverhalten, max. Ressourcenverbrauch)
Entwicklungsbedingungen (evtl. Methoden, Programmiersprachen, Plattform, Betriebssystem)
Abnahmekriterien

Alles in kundenverständlicher Form, modellieren mit OOA.

Ergebnis: Systemmodell, Oberflächenprototyp, Pflichtenheft.

Achtung: Jede Funktion sollte stornierbar sein!

0.4 eXtreme Programming (nach Kent Beck) als Projektmodell

- Gehört neben z.B. Scrum zur Gruppe der agilen Prozesse [www.agileAlliance.org]

- Versucht, Kunden- und Entwicklerinteressen zufriedenzustellen:
  - ➔ Kunden haben das Recht auf Irrtum und Meinungsänderung.
  - ➔ Entwickler haben das Recht, Aufwände selbst zu schätzen und Qualität zu liefern.

- Einsatz von 12 guten Praktiken in extremem Maße, **Bsp.**: Refactoring.