Praxis der Software-Entwicklung (PSE)

Prof. Dr.-Ing. Gregor Snelting (Koordinator)

WS 09/10

Praxis der Software-Entwicklung (PSE)

- Ziel: Entwicklung eines mittelgroßen Systems im Team mit moderner Softwaretechnik

Praxis der Software-Entwicklung (PSE)

- Ziel: Entwicklung eines mittelgroßen Systems im Team mit moderner Softwaretechnik
- Zielsystem: max 10kLOC
  objektorientierter Entwurf (UML), Implementierung (Java/C++/C#), Qualitätssicherung (zB Jcov, Junit)

Praxis der Software-Entwicklung (PSE)

- Ziel: Entwicklung eines mittelgroßen Systems im Team mit moderner Softwaretechnik
- Zielsystem: max 10kLOC
  objektorientierter Entwurf (UML), Implementierung (Java/C++/C#), Qualitätssicherung (zB Jcov, Junit)
- Teilnehmer: 3. Sem BA Informatik
  Voraussetzung: Grundbegriffe der Informatik, Programmieren, Algorithmen 1, Softwaretechnik 1
- Empfehlung: PSE erst, wenn *alle* Module aus 1./2. Semester vorliegen
- Umfang: 6LP, $\approx$ 200 Arbeitsstunden / Teilnehmer, $\approx$ 1.5 Arbeitstage/Woche/Teilnehmer
- Zeitplan: Mitte Oktober 2009 - Mitte März 2010; in diesem Zeitraum 15 Wochen Praktikumsbetrieb nach Absprache

PSE / Anmeldung

- vorläufige Anmeldung zur Veranstaltung: ab sofort via WebInscribe unter
  `http://pp.info.uni-karlsruhe.de/lehre/WS200910/PSE/`
  (oder auch via Softwaretechnik 1)
- Es gibt ca. 20 verschiedene Aufgabenstellungen, die aber alle einem objektorientierten Phasenmodell folgen
  Es gibt ca. 40 Teams a 5-6 Studenten

PSE / Anmeldung

- vorläufige Anmeldung zur Veranstaltung: ab sofort via WebInscribe unter
  `http://pp.info.uni-karlsruhe.de/lehre/WS200910/PSE/`
  (oder auch via Softwaretechnik 1)
- Es gibt ca. 20 verschiedene Aufgabenstellungen, die aber alle einem objektorientierten Phasenmodell folgen
  Es gibt ca. 40 Teams a 5-6 Studenten
- endgültige Anmeldung, Auftaktveranstaltung mit Aufgabenvorstellung, Teamzusammenstellung: Mitte Oktober (siehe Webseite / besondere Ankündigung)
- Wünsche zu Teamzusammensetzung / Aufgabe werden nach Möglichkeit berücksichtigt

PSE / Übersicht

- moderne Softwaretechnik ist wichtig für alle BA-Absolventen!
- vollständige Entwicklung eines größeren Systems
- Phasenmodell: Pflichtenheft, Entwurf, Spezifikation, Implementierung, Validierung
- Abschlusspräsentation
- Teamarbeit (Teams a 5-6 Teilnehmer)
- durchgehend Objektorientierung
- Toolunterstützung, zB Eclipse, Rational Architect, JUnit, JCov, ...

Pflichtenheft

Phasenziel
detaillierte Festlegung der Leistungsmerkmale eines Systems
Grundprinzipien

- Präzision
- Vollständigkeit
- Konsistenz

Vorgehen

- Systemmodell (grobe Übersicht), Systemumgebung (Hard/Software)
- vollständige funktionale Anforderungen
- GUI-Entwürfe (manuell oder programmiert)
- ausführliche Testfallszenarien

verlangt wird
Abgabe des Pflichtenheftes nach 2 Wochen;
Erläuterung im ersten Kolloquium

Entwurf & Spezifikation

objektorientiert (UML)

Phasenziel

- Festlegung der Klassenstruktur
- Schnittstellendefinition der Klassen
- Beziehungen zw. Klassen (Vererbung, Assoziationen)
- Klassendiagramm, ausgewählte Sequenzdiagramme, evtl. Zustandsdiagramm
- Einsatz von Design Patterns, MVC

Entwurf & Spezifikation

objektorientiert (UML)

Phasenziel

- Festlegung der Klassenstruktur
- Schnittstellendefinition der Klassen
- Beziehungen zw. Klassen (Vererbung, Assoziationen)
- Klassendiagramm, ausgewählte Sequenzdiagramme, evtl. Zustandsdiagramm
- Einsatz von Design Patterns, MVC

Grundprinzipien

- Geheimnisprinzip
- schwache Kopplung
- hohe Kohäsion
- Lokalitätsprinzip
- Wiederverwendbarkeit von Klassen/Subsystemen

Entwurf & Spezifikation

objektorientiert (UML)

Phasenziel

- Festlegung der Klassenstruktur
- Schnittstellendefinition der Klassen
- Beziehungen zw. Klassen (Vererbung, Assoziationen)
- Klassendiagramm, ausgewählte Sequenzdiagramme, evtl. Zustandsdiagramm
- Einsatz von Design Patterns, MVC

Grundprinzipien

- Geheimnisprinzip
- schwache Kopplung
- hohe Kohäsion
- Lokalitätsprinzip
- Wiederverwendbarkeit von Klassen/Subsystemen
- OO: Vererbung/dynamische Bindung statt Fallunterscheidung

Entwurf & Spezifikation / 2

Vorgehen

- Kombination von Top-Down und Bottom-Up Design
- Identifikation von Klassen, Vererbung, Assoziationen
- Festlegung der Schnittstellen aller Klassen
- informale Beschreibung aller Klassen
- evtl. Einsatz von Entwurfsmetriken (JMetrics)

Entwurf & Spezifikation / 2

Vorgehen

- Kombination von Top-Down und Bottom-Up Design
- Identifikation von Klassen, Vererbung, Assoziationen
- Festlegung der Schnittstellen aller Klassen
- informale Beschreibung aller Klassen
- evtl. Einsatz von Entwurfsmetriken (JMetrics)

verlangt wird

- Abgabe der UML-Diagramme nebst informeller Beschreibung nach 4 Wochen;
  Verteidigung im zweiten Kolloquium
- *Nachweis der Evolutionsfähigkeit* (zB Lokalitätsprinzip)

Implementierung & Validierung

Phasenziel

Programmierung und Test des Systems

Grundprinzipien

- Programmierung in Java (evtl. C#, C++)
- Umsetzung der Architektur
- werkzeugunterstützte Qualitätssicherung

Implementierung & Validierung

Phasenziel

Programmierung und Test des Systems

Grundprinzipien

- Programmierung in Java (evtl. C#, C++)
- Umsetzung der Architektur
- werkzeugunterstützte Qualitätssicherung

Vorgehen

- Implementierung der Methoden
- funktionaler Komponententest mit `Junit`, evtl. Überdeckungstests (zB JCov); verschränkt mit Implementierung
- Integrationstest, Robustheitstest
- Realisation der Szenarien aus Pflichtenheft

Implementierung & Validierung / 2

verlangt wird

- Implementierungsplan; Implementierung;
- Implementierungskolloquium (2 Wochen vor Abschluss)
- Testbericht; Systemabnahme (1 Woche vor Abschluss)
- Abschlusspräsentation (Februar/März 2010)