Imhotep

Vidensformidling indenfor
Softwareudvikling

Flexibelt, pålideligt, og vedligeholdbart software.

Overblik:

I kurset vil jeg fokusere på hvorledes pålideligt, fleksibelt, og vedligeholdbart software i høj grad fremkommer naturligt, hvis man benytter nogle bestemte teknikker og en bestemt vinkel på, hvordan man designer, udvikler, og vedligeholder software. De primære teknikker er *test-dreven udvikling (TDD)* som fokuserer på hurtig fremdrift og høj pålidelighed, *refaktorisering* som fokuserer på at sikre stadig høj kvalitet i software arkitekturen, *test doubles* som definerer teknikker til at teste software moduler uafhængigt af hinanden og af eksterne ressourcer, samt *principperne for fleksibelt design* og *3-1-2 processen* som nærmest automatisk indfører gode *design patterns* i ens arkitektur og sikrer at designet får lav kobling, høj samhørighed, og tillige højner forståelsen for den overordnede arkitektur af software. Der vil blive lagt vægt på *UML* og *softwarearkitektur views* som centrale teknikker for at dokumentere software og arkitektur. Endelig vil jeg give et indblik i softwarearkitektur kvalitet og teknikker til at formulere kvalitetskrav.

Teknikkerne vil blive demonstreret på en konkret case, nemlig en betalingsautomat til en parkeringsplads. Dette software starter som en simpel applikation men i præsentationerne gennemgår jeg en proces hvor nye kundekrav og arkitektur-refaktoriseringer afløser hinanden og demonstrerer agil og test-dreven software udvikling, og hvorledes en mere og mere kompliceret kodebase forbliver pålidelig, overskuelig og fleksibel. Denne proces er også udgangspunktet for hand-on øvelserne i kurset. Jeg vil bevidst fokusere på teknikkernes kode-nære aspekter fordi alle fordelene i et design pattern baseret design alt for let kan ødelægges ved simple fejl på implementationsniveauet. Case og øvelserne er baseret på Java og simple build- og exekveringsscripts til Linux og Windows. På anden dagen er øvelserne mere baseret på ”papir og blyant”.

Format:

Kurset vil skifte mellem præsentationer og dialog med kursisterne, hands-on sessioner hvor der løses opgaver i direkte forlængelse af de gennemgåede teknikker i grupper af 3-4 personer, og diskussioner af erfaringerne fra disse øvelser. Disse øvelser er pædagogisk vigtige primært i forhold til den enkelte kursists læring og sekundært i forhold til at skabe en dynamisk og varieret undervisningssituation.

Forudsætninger:

Jeg forudsætter at deltagerne kender et procedurelt programmeringssprog og i nogen omfang også et objekt-orienteret programmeringssprog og standard objekt-orienteret terminologi. Opgaverne i kurset vil være i Java. Kursisterne forventes at have adgang til et sådant miljø på en PC'er (en per gruppe af 2-3 personer) i enten Linux eller Windows

(version underordnet), i relativ nærhed af undervisningslokalet. Det forventes, at der er slide projektor til laptop og gerne en tavle i undervisningslokalet.

Litteratur:

Slides. Slides som PDF samt opgaver fremsendes umiddelbart før kurset til jer via e-mail.
An Approach to Software Architecture Description Using UML Revision 2.0, af Christensen, Corry, og Hansen.
Kurset vil tage udgangspunkt i min bog "Flexible, Reliable Software: Using Patterns and Agile Development" fra CRC Press 2010, og det vil være en fordel hvis kursisterne har adgang til enkelte eksemplarer af denne.

Rettigheder:

Slides og opgavemateriale må frit bruges internt i Thrane og Thrane men ikke videregives til tredje-part. Jeg ejer rettighederne til alt materialet og dette må ikke anvendes i anden sammenhæng. CRC Press ejer rettighederne til min bog.

Dagsplan (udkast):

Undervisningen foregår to dage i træk, fra kl. 9.00 til 16.00. Der beregnes ca. 5 ½ time undervisning pr dag, afbrudt af frokost, samt to pauser af ca. et kvarter i løbet af undervisningsdagen.

Dag 1:

Introduktion af Case, en betalingsautomat for en parkeringsplads. Gennemgang af UML som middel til at dokumentere dette case (klasse- og sekvensdiagrammer). Test-dreven udvikling: TDD patterns og TDD rytmen. Håndtering af en ny produktvariant: Analyse af variabilitetsteknikkerne "source code copy", "parametrisk-", "polymorfisk-", og "kompositionelt" design. Udledning af Strategy pattern igennem en test-dreven process med fokus på refactoring, kompositionelt design og 3-1-2 processen. Ny produktvariant og efterfølgende udledning af State pattern.

Dag 2:

Teknikker til test med kontrol af eksterne resurser: Test Doubles. Rolle begrebet og principperne for fleksibelt design. Softwarearkitektur beskrivelser og viewpoints (specielt module, component-connector, samt deployment viewpoints). Softwarearkitektur kvalitet og Quality Attribute Scenarios. Unit testing i en Thrane og Thrane sammenhæng (kræver samarbejde med jeres tekniske stab.)