

Modern, objektorienterad programvarudesign - 4 dagar

kurser 1801

- Du får lära dig att**
- Leverera programvara inom tid och budget med iterativa och lätttrörliga (agila) metoder
 - Fånga korrekta krav genom användarberättelser och förädling av användningsfall
 - Använda UML-modellering för förbättrad design
 - Skapa återanvändbara och komponentbaserade objektorienterade arkitekturer
 - Producera flexibla och anpassningsbara system med hjälp av iterativ och inkrementell design
 - Garantera robusta lösningar genom testdriven utveckling, omfaktorisering och designmönster

Sammanfattning I dagens snabbt föränderliga affärsmiljö är det en konkurrensfördel att snabbt kunna leverera programvara som är anpassad till ny teknik och ändrade användarkrav. Att använda UML-modellering och lätttrörliga metoder är god industripraxis för att utveckla sådan programvara. På kursen lär du dig analysera, designa och implementera programvara med högeffektiva, iterativa och inkrementella metoder.

Vem bör delta Programmerare och programvarudesigners, teamledare, projektledare och kravanalytiker. Grundläggande kunskaper om objektorienteringsbegrepp förutsätts.

Praktiska övningar Praktiska övningar ger erfarenhet av att använda UML och lätttrörliga metoder iterativt och inkrementellt. Övningar och demonstrationer innehåller bland annat:

- Utveckla användarberättelser till användningsfall
- Modellera dynamiken i användningsfall med UML:s sekvens- och aktivitetsdiagram
- Beskriva komplexa beteenden med tillståndsdigram
- Utforma systemarkitekturer med klass- och komponentdiagram
- Producera och förbättra kod med TDD
- Extrahera och identifiera designmönster i kod

Modern, objektorienterad programvarudesign - 4 dagar

kurser 1801

Introduktion

- Anpassa metoden till projektets storlek
- Uppnå lätttrörlighet med hjälp av iterativ utveckling
- Designa effektivt med hjälp av UML
- Omsätta design i färdig lösning med testdriven utveckling

Anpassa metoden till projektet

Utvärdera traditionella metoder

- Granska vattenfalls- och V-modellivscykeln
- Hantera förändring iterativt och inkrementellt

Utforska iterativa och lätttrörliga alternativ

- Identifiera riskerna med alternativ som endast bygger på lätttrörliga metoder
- Minska risker med UML-baserad design

Samla in korrekta krav

Förbereda för iterativ och inkrementell utveckling

- Identifiera och involvera intressenter
- Fånga användarberättelser och fylla backlogen
- Förädla krav genom att utveckla användarberättelser till användningsfall

Planera en iterativ cykel

- Bedöma arbetet för design och utveckling av användarberättelser
- Få intressenterna att prioritera
- Hantera ofullständiga och inbördes beroende användarberättelser

Designa användarberättelser med UML

Använda lämplig andel modellering

- Undvika för mycket eller för lite modellering
- Modellera statiska strukturer: klassdiagram och komponentdiagram
- Representera systemets dynamik med aktivitetsdiagram

Designa den dynamiska arkitekturen

- Utforma trelagslösningar på användningsfall
- Beskriva dynamiken i användningsfall med sekvensdiagram
- Kontrollera alternativa flöden med tillståndsdigram
- Översätta beteenden till MVC-arkitekturen

Representera den statiska arkitekturen

- Skapa en entitetsmodell från klasser och associationer
- Verifiera datastrukturen mot beteendemodellen

Bygga programvaran

Dokumentera designen i detalj med UML

- Implementerade klassdiagram
- Beskriva kodbeteende med sekvens- och tillståndsdigram
- Ta med modeller från CASE-verktyg i iterationsresultatet
- Specificera och designa metod-algoritmer
- Förbättra effektiviteten genom att skapa restriktioner

Etablera god, teststyrd praxis

- Skriva körbara tester för användarberättelser och användningsfall
- Välja rätt enhetstester: ekvivalensklasser och gränsvärden
- Automatisera testprocessen med enhetstestning och härmramverk (mocking frameworks)
- Isolera komponenter med Mock Objects

Omfaktorisera till bättre lösningar

- Förbättra återanvändbarhet med öppen/stängd-principen
- Minska koppling och öka sammanhållning genom enkelt ansvar
- Skilja ut gränssnitt och vända beroenden
- Segregera gränssnitt för att maximera anpassningsbarhet

Öka granulariteten i designen genom mönster

- Separera beteenden med strategimönstret
- Isolera de tre lagren med MVC och observatörmönster
- Centralisera skapande av objekt till fabriker

Integrera delsystem för att skapa ett fungerande system

- Isolera tester genom att härma underlydande moduler
- Konstruera och köra integrationstester
- Minimera coupling med fasad- och proxy-mönster

Stödja den iterativa processen

Slutföra iterationen

- Validera slutförda användarberättelser och användningsfall
- Leverera modeller och koder till versionskontroll

- Finjustera processen för kommande iterationer

Använda rätt verktyg

- Jämföra automatiserade testverktyg
- Stödja förändringar och buggar med spårverktyg
- Duplicera versionshantering för krav, modeller och kod

Implementera god praxis för UML och iterativa metoder

- Identifiera praxis som kan användas på arbetsplatsen
- Bedöma vilken praxis som bäst passar i din organisation