ADE400 : Design Patterns

Code : ADE400 **Duur :** 3 dagen **Categorie :** Analysis and Design

Doelgroep :

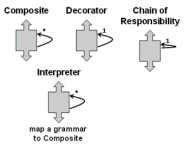
Ervaren developers en software architecten met kennis van object georinteerd programmeren en systeemanalyse die Design Patterns willen toepassen bij het ontwerpen van deze systemen.

Voorkennis :

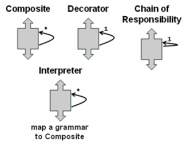
Kennis van een object georinteerde programmeertaal zoals C++, C# of Java en ervaring met object georinteerde analyse en design met UML is vereist.

Uitvoering :

De concepten worden behandeld aan de hand van presentatie slides. De theorie wordt gellustreerd met demo's van patterns in C++, C# en Java. Er zijn oefeningen in design problemen waarbij patterns worden toegepast.



Design Patterns



Inhoud :

In deze cursus leert u hoe design patterns kunnen worden toegepast bij het object georinteerd ontwerpen van systemen. Na een inleiding over de rol die design patterns spelen en hoe ze kunnen worden gebruikt om de non functional requirements van systemen te realiseren, wordt aandacht besteed aan hoe design patterns beschreven en gecatalogiseerd worden. Ook de rol van design patterns in de architectuur van applicaties wordt besproken evenals de verschillende categorien van design patterns die kunnen worden onderscheiden. In de module Creational Patterns komen de Factory patterns en het Builder, Prototype en Singleton pattern aan de orde. U leert uit welke classes, relations, responsibilities en collaboraties een typische design pattern oplossing kan bestaan. Vervolgens worden in de module Structural Patterns van het Adapter, Composite, Bridge, Decorator, Proxy en FlyWeight pattern besproken. U leert wat de gevolgen zijn van de toepassing van de design patterns, de voordelen en mogelijke nadelen wat betreft tijd en ruimte en welke overwegingen u kunt hanteren om een bepaald attern te gebruiken. Vervolgens worden in de module Behavioral Patterns de Chain of Responsiblity, Interpreter, Iterator, Mediator, State en Strategy patterns besproken. En tot slot worden in de module Architectural Patterns patterns besproken die betrokken zijn bij de architectuur van software, waaronder Operating Systems en Frameworks. Deze module richt zich onder andere op het Layer pattern, het Micro Kernel pattern en het Model View Controller (MVC) pattern.

Module 1 : Intro Design Patterns

What is a design pattern?
Describing design patterns
Reuse through design patterns
Structure of patterns
Classification of patterns
Catalog of Design Patterns
Creational Patterns
Structural Patterns
Behavioral Patterns
Sample design patterns
Selecting Design Patterns
Solving problems with design patterns

Module 2 : Creational Patterns

Factory Patterns
Factory Method Pattern
Provide hooks for subclasses
Connect parallel class hierarchies
Abstract Factory Pattern
Encapsulating Implementation Dependencies
Concrete Class Isolation
Exchanging Product Families
Promoting Consistency
Builder Pattern
Vary Internal Representation
Isolate construction code
Controlling the build process
Prototype
Adding products at run-time
Reduced Subclassing
Dynamic configuration
Singleton Pattern
Controlled access
Reduced name space

Module 3 : Structural Patterns

Adapter Pattern
Pluggable Adapters
Two-way adapters
Bridge Adapters
Decoupling interface and implementation
Improved Extensibility
Hiding Implementation Details
Composite Pattern
Explicit parent references
Sharing Components
Decorator Pattern
Improved flexibility
Lots of little objects
Faade Pattern
Reducing client-subsystem coupling
Public subsystem classes
Private subsystem classes
Flyweight Pattern
Reducing number of instances
Proxy Pattern
Remote Proxies
Virtual Proxies
Copy-on-write

Module 4 : Behavioral Patterns

Chain of responsibility
Reduced Coupling
No guaranteed receipt
Command Pattern
Decoupling Objects
Use of callback functions
Supporting undo and redo
Avoiding error accumulation
Interpreter Pattern
Implementing Grammars
Grammar Changes
Abstract syntax tree
Iterator Pattern
Controlling iteration
Traversal algorithm
Mediator Pattern
Limiting subclasses
Simplification of object protocols
Memento Pattern
Preserving Encapsulation Boundaries
Storing incremental changes
Observer Pattern
Subject and receiver
Broadcast Communication
State Pattern
State transitions
Sharing State Objects
Strategy Pattern
Context interfaces
Template parameters
Template Pattern

Module 5 : Architectural Patterns

Architectural patterns versus design patterns
Patterns for real-time software
Layers
Pipes and Filters
Blackboard
Broker
Model-View-Controller
Presentation-Abstraction-Control
Microkernel
Reflection