PATTERNS IN DE ECHTE WERELD

Veerle Ongenae, Wijnand Schepens

Patterns in de echte wereld

- Definitie pattern
- Catalogus van patterns
- Classificatie van patterns
- Gebruik van patterns
- Anti-patterns

Definitie patterns

- Een pattern is een oplossing voor een probleem in een context
- Context
  - Situatie die vaker voorkomt
- Probleem
  - Doel te verwezenlijken binnen de context
  - Beperkingen vanuit context
- Oplossing
  - Algemeen ontwerp
    - Voor iedereen bruikbaar
  - Oplossing voor probleem + beperkingen

Catalogus van patterns

- Beschrijving van verschillende patterns
  - Naam
  - Context
  - Probleem
  - Oplossing
  - Bedoeling
  - Motivatie
  - Toepassingen
  - Gevolgen
  - …

Catalogi van patterns

- GoF: “Gang of Four”
  - Fundamentele patterns
- Domeinspecifieke patterns
  - EJB patterns
- Application Patterns
  - Meerlagen architectuur, MVC, client-server
- User Interface Design Patterns
  - Interactieve software
- Andere
  - Portland Pattern Repository (http://c2.com/ppr/)
  - Hillside Group (http://hillside.net/europlop/HillsideEurope/)

Oefening

- p. 574

Patterns classificeren

7

- Creational
  - Aanmaken objecten
  - Ontkoppelen client en aan te maken objecten
- Behavioral
  - Samenwerking klassen en objecten
  - Verantwoordelijkheid delen
- Structural
  - Klassen en objecten in structuren onderbrengen

Patterns classificeren

8

- Creational
  - Singleton
  - Abstract Factory
  - Factory Method
- Structural
  - Decorator
  - Proxy
  - Composite
  - Facade
  - Adapter
- Behavioral
  - Template Method
  - Command
  - Iterator
  - Observer
  - State
  - Strategy

Tweede classificatie

9

- Klassenpatterns
  - Relaties tussen klassen
  - Compile time
- Objectpatterns
  - Relaties tussen objecten
  - Compositie
  - Vaak at runtime gerealiseerd
- Klassenpatterns
  - Template Method
  - Adapter
  - Factory Method
- Objectpatterns
  - Composite
  - Decorator
  - Proxy
  - Strategy
  - Facade
  - Command
  - Iterator
  - Observer
  - State
  - Abstract Factory
  - Singleton

Denken in patterns

10

- Hou het eenvoudig
- Een pattern is geen wondermiddel
- Wanneer gebruiken
  - Het lost een probleem op
  - Eenvoudige oplossing niet afdoende
  - Sommige delen zullen veranderen
- Refactoring
  - Verbetering structuur code
- Haal weg wat je niet nodig hebt
  - Te complex
- Als je het nu niet nodig hebt, doe het niet nu.
- Eerst ontwerpprincipes, dan patterns

Denkwijzen over patterns

11

- Beginnersdenkwijze
  - Overal patterns
- Gevorderde denkwijze
  - Ziet wanneer nodig en wanneer niet
  - Aanpassen patterns
  - Combineren
- Zen denkwijze
  - Ziet waar patterns op een natuurlijke wijze passen

Kracht gedeelde woordenschat

12

- Woordenschat gedeeld tussen ontwikkelaars
  - Betere communicatie
  - Betere ontwerpen

Anti-Patterns

13

- Andrew Koenig (1995)
- Vertelt je hoe je van een probleem naar een SLECHTE oplossing gaat
  - Andere ontwikkelaars behoeden voor dezelfde fout
  - Waarom slechte oplossing aantrekkelijk is
  - Waarom slecht
  - Wijst richting voor een goede oplossing aan

Enkele anti-patterns (wikipedia)

14

Organizational

- **Analysis paralysis**: Devoting disproportionate high effort to the analysis phase of a project
- **Escalation of commitment**: Failing to revoke a decision when it proves wrong
- …

Project Management

- **Software bloat**: Allowing successive versions of a system to demand ever more resources
- **Smoke and mirrors**: Demonstrating unimplemented functions as if they were already implemented

Enkele anti-patterns (wikipedia)

15

Software design

- **Big ball of mud**: A system with no recognizable structure.
- **Gold plating**: Continuing to work on a task or project well past the point at which extra effort is adding value.
- **Magic pushbutton**: Coding implementation logic directly within interface code, without using abstraction.

Object-oriented design

- **BaseBean**: Inheriting functionality from a utility class rather than delegating to it
- **God object**: Concentrating too many functions in a single part of the design (class)
- **Object orgy**: Failing to properly encapsulate objects permitting unrestricted access to their internals

Code smell

16

- **code smell** is any symptom in the source code of a program that possibly indicates a deeper problem.