Aspect-Oriented Analysis and Design

The Theme Approach

Siobhán Clarke
Trinity College Dublin
What’s wrong with objects – 1?

- Concerns that crosscut multiple parts of a system cannot be modularized.
What’s wrong with objects – 2?

Class modularization encapsulates multiple concerns
The Theme Approach

Brief History Lesson!

- Early ’90’s: Subject-Oriented Programming
  - Symmetric view of decomposition
- 1997+ Subject-Oriented Design
- 2001: Subject-Oriented Design -> Theme/UML
- 2003+ The Theme Process:
  - Theme/UML – design
  - Theme/Doc – requirements analysis
Overview

Theme/Doc for analysis

View relationships between concerns

Theme/UML for design

Model concerns separately
A new game will randomly distribute crystals around the game world.

If players do not reach their initial location in time, they lose one energy point.

Dropped crystals will be re-scattered throughout the game area.

Players lose energy at two units per five minute period in a game location.

If a player enters a location that has no players or characters in it, they may pick up any crystals or magical items there.

When a game starts, players gain 10 units of energy.

Energy is gained by two units when a player picks up a crystal entering a location.

When two players meet on entering a location, they perform a duel of rock paper scissors.

When a player completes a physical test challenge successfully, they gain three units of energy and win a crystal.
R18 A new game will randomly distribute crystals around the game world.

R21 If players do not reach their initial location in time, they lose one energy point.

R35 Dropped crystals will be re-scattered throughout the game area.

R37 Players lose energy at two units per five minute period in a game location.

R38 If a player enters a location that has no players or characters in it, they may pick up any crystals or magical items there.

R40 When a game starts, players gain 10 units of energy.

R42 Energy is gained by two units when a player picks up a crystal entering a location.

R61 When two players meet on entering a location, they perform a duel of rock paper scissors.

R80 When a player completes a physical test challenge successfully they gain three units of energy and win a crystal.
Theme/Doc
graphing the relationships between the concerns
Theme/Doc

a look forward to the final goal

crosscutting themes (aspects)

base themes
Theme/Doc

graphing the relationships between the concerns
Enter a location that has no players or characters in it, they may pick up any crystals or magical items there.

Energy is gained by two units when a player picks up a crystal entering a location.
Theme too general?
- split into smaller ones
Themes too similar?
- unify synonyms, group
Theme not really useful?
- delete it!
Requirement “orphaned”?
- find a home
Requirement ambiguous?
- resolve and refine
Theme/UML

theme **design** process

Aspect-Oriented Analysis and Design with Theme
Theme/UML

model themes separately – base themes with standard UML

```
«theme» Start

GameAdmin
+ newGame(String, int)

Player
- name : String
- IPAddress : String
+ getLocation()
+ setGame()

Game
- gameDuration : int
- name : String
+ setGameLocations(Vector)
+ setThroneRoom(Location)

Vector
1..*

gameWorld

Location
+ name
+ northEast : Coordinate
+ northWest : Coordinate
+ southEast : Coordinate
+ southWest : Coordinate

activeGames

0..*

GameLocations

1

GameLocations

1

gameLocations

Vector
1..*

currentLocation

Player
+ setLocation()
+ checkOtherPlayers()  
+ takeCrystals()
+ incrementCrystals()
+ takeMagicItems()
+ setMagicItem()

Location
+ takeCrystals()
+ takeMagicItems()

Player
+ setLocation()
+ checkOtherPlayers()
+ takeCrystals()
+ incrementCrystals()
+ takeMagicItems()
+ setMagicItem()

Location
+ takeCrystals()
+ takeMagicItems()

GPSComponent
+ getCurrentGPSLocation()

Game
+ inGameLocation()
+ getOtherPlayers()

signals
leave(Player)

1..*

signals
duel()

1..*

players

Game
+ inGameLocation()
+ getOtherPlayers()

signals
leave(Player)

1..*

players

Game

MagicItem
+ takeCrystals()
+ takeMagicItems()

MagicItem
+ takeCrystals()
+ takeMagicItems()

MagicItem
+ takeCrystals()
+ takeMagicItems()

MagicItem
+ takeCrystals()
+ takeMagicItems()
```

Aspect-Oriented Analysis and Design with Theme
Theme/UML
model themes separately – aspect themes with minor UML extensions

«theme»
track-energy

< Energy.moveLocation(..) >
< Energy.energyAction(..) >
< Energy.joinGame(..) >

Template parameters used to reason about triggers of behaviour in aspect theme
- grouped by behaviour sequence within < > brackets
Distinguish between:
- execution of base operation (\_do\_op()) and
- execution of crosscutting behaviour (op())

Execution of real behavior

Crosscutting behavior specified to happen after real behavior
Theme/UML
model themes separately – aspect triggers with control flow restrictions

```
<table>
<thead>
<tr>
<th>:Class1</th>
<th>:Class2</th>
</tr>
</thead>
<tbody>
<tr>
<td>_do_op1(..)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>op2(..)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>_do_op2(..)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- `op2()` a trigger **only** within this control flow

**control flow of _do_op1()**
Theme/UML
model themes separately – aspect themes arising from detailed design

Aspect-Oriented Analysis and Design with Theme
Theme/UML
tHEME composition process

Analysis
- Requirements
  - choose
  - Themes
    - view
    - Relationships
      - show
        - Base
        - Aspects

Design
- model
  - Base
  - Aspects

Composition
- write
  - Composition Specification
    - view
    - Composed Design

Pick Themes To Compose
- Base
- Aspects
  - Identify Matching Elements
  - Specify Integration
    - Override
    - Merge
      - Resolve Conflicts
  - Bind Base Triggers to Aspect Templates
Theme/UML

compose themes – Composition Relationship and **base** themes

- **match[name]**
- **nomatch**

**Composition Relationship tags:**

- for **conflict resolution**
  - **[prec]**
  - reconcile{explicit[{}]}]
  - reconcile{default[Type {..}]}

ordering sequence for op merge
Theme/UML
compose themes – Composition Relationship and aspect themes

Binding base design elements to template parameters.
Map to AspectJ

Abstract Aspect

Concrete Aspect

Intertype Declarations

Concrete Pointcuts

Concrete Intertype Declarations

Pointcuts

Advice

Intertype

Game

«theme»

<Energy.moveLocation(..)>
<Energy.energyAction(..)>
<Energy.joinGame(..)>

bind[ < Player.setLocation() >
< Player.{incrementCrystals(), addCrystals(), completeWizardErrand(), completeWarriorTest() } >
< Player.joinGame() > ]
public aspect ConcreteTrackEnergy extends TrackEnergy {
    declare parents: Player implements EnergyEntityI;
    declare parents: Game implements GameI;

    pointcut moveLocation(EnergyEntityI energyEntity):
        this(energyEntity) &&
        execution (* Player.setLocation(..));

    pointcut energyAction(EnergyEntityI energyEntity):
        this(energyEntity) &&
        (execution (* Player.incrementCrystals(..)) ||
         execution (* Player.addCrystals(..)) ||
         execution (* Player.completeWarriorTest(..)) ||
         execution (* Player.completeWizardErrand(..)));

    pointcut joinGame(Player player):
        this(player) &&
        execution (* Player.joinGame(..));
}
Map to CME - aspects

```
<theme>
  TrackEnergy
  Energy
  Game
</theme>

bind[
  < Player.setLocation() >
  < Player.{incrementCrystals(), addCrystals(), completeWizardErrand(), completeWarriorTest() } >
  < Player.joinGame() > ]

<theme>
  Game
  Energy
</theme>

Game Theme Package

Track Energy Theme Package

Compose-game.hl

extend concern Game with TrackEnergyTheme;
extend type
  BaseTheme: Player
with
  TrackEnergyTheme: Energy;
advise method
  BaseTheme:Player.setLocation(Location)
with after:
  TrackEnergyTheme: Energy.setChangeAmount();
...
```
Map to CME – base

Start Theme Package
- Player.java
  public class Player {...}
- Location.java
  public class Location {...}
- GameAdmin.java
  public class GameAdmin {...}
- Game.java
  public class Game {...}

Enter Location Theme Package
- Player.java
  public class Player {
    ...
    public void duel(Player p) {
      // stub
    }
  }
- Location.java
  public class Location {...}
- Game.java
  public class Game {...}
- MagicItem.java
  public class MagicItem {...}
- Wager.java
  public class Wager {...}

CME Composition Specification

Compose-game.hl
merge concerns  StartTheme, EnterLocationTheme, DuelTheme => Game;
override method EnterLocationTheme: Player.duel(Player)
with DuelTheme: Player.duel(Player);
Theme publications

Siobhán Clarke, Robert J. Walker. “Generic Aspect-Oriented Design with Theme/UML“ In Aspect-Oriented Software Development, Addison-Wesley, 2005


... and a plug!
Questions...