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.
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, they win three units of energy and gain 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
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.

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

getting there.. base themes

Theme too general?
   \textit{split into smaller ones}
Themes too similar?
   \textit{unify synonyms, group}
Theme not really useful?
   \textit{delete it!}
Requirement “orphaned”?
   \textit{find a home}
Requirement ambiguous?
   \textit{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

```
GameAdmin
+ newGame(String, int)

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

Vector
+ name
+ gameWorld

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

Player
- name : String
- IPAddress : String
+ setLocation()
+ checkOtherPlayers()
+ takeCrystals()
+ incrementCrystals()
+ takeMagicItems()
+ setMagicItem()

Player
+ getLocation()
+ setGame()

Game
- gameDuration : int
- name : String
+ inGameLocation()
+ getOtherPlayers()

Location
- name

Player
- name
- IPAddress
- invention : String

Game
+ inGameLocation()
+ getOtherPlayers()

Location
- crsytals : int
+ takeCrystals()
+ takeMagicItems()

MagicItem
- crsytals : int
+ takeCrystals()
+ takeMagicItems()
```

Aspect-Oriented Analysis and Design with Theme
Theme/UML

model themes separately – aspect themes with minor UML extensions

«theme»
track-energy

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

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

\[
\begin{align*}
\text{op1(..)} & \quad \text{before()} \\
\_do\_op1(..) & \quad \text{after()} \\
\text{op2(..)} & \quad \text{sd restrictedTrigger}
\end{align*}
\]

\[\text{control flow of } \_do\_op1()\]

\[\text{op2(\ldots) a trigger } \textbf{only} \text{ within this control flow}\]
model themes separately – aspect themes arising from detailed design
Theme/UML

theme composition process

Aspect-Oriented Analysis and Design with Theme
Theme/UML
compose themes – Composition Relationship and base themes

Composition Relationship tags:
- match[name]
- nomatch
- for conflict resolution
  - [prec]
  - reconcil[explicit[{{..}}]]
  - reconcil[default[Type {{..}}]]
- ordering sequence for op merge

Aspect-Oriented Analysis and Design with Theme
Theme/UML
compose themes – Composition Relationship and **aspect** themes

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

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

```
<theme>
track-energy
```

```
<theme>
game
```

**Binding base design elements to template parameters.**
Map to AspectJ

- «theme» TrackEnergy
  - Energy
    - Game
      - Game:Type
        - _do_templateOp()
        - crosscuttingOp()

- «theme»
  - bind[<Player.setLocation()>
    - <Player.incrementCrystals(),
      addCrystals(),
      completeWizardErrand(),
      completeWarriorTest()>
    - <Player.joinGame()>

Abstract Aspect

Intertype Declarations

Pointcuts

Advice

Concrete Aspect

Concrete Pointcuts

Concrete Intertype Declarations
Small flavour – concrete aspect

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

```
<TrackEnergy.moveLocation(..)>  
<TrackEnergy.energyAction(..)>  
<TrackEnergy.joinGame(..)>  
bind( <Player.setLocation() >         
     <Player.{incrementCrystals(), 
          addCrystals(),  
          completeWizardErrand(),  
          completeWarriorTest()} >  
     <Player.joinGame()>)  
```

```
Game:templateOp()  
crosscuttingOp()  
_ do_templateOp()  

Player:templateOp()  
_ do_templateOp()  
```

```
Game:Type

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

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...