Il refactoring – 4

Stefano Mizzaro

Dipartimento di matematica e informatica
Università di Udine
http://www.dimi.uniud.it/~mizzaro
mizzaro@dimi.uniud.it

PAOO, Lezione 19
24/5/2004

Calendario provvisorio

- Lu. 24/5:
  - UMLxWeb (Zennaro)
  - Fine Refactoring
- Me. 26/5
  - J2ME (Valle e Vassena)
- Ve. 28/5
  - Qualità dei diagrammi UML (Baruzzo)
  - Agenti
- Lu. 31/5
  - Il framework EasyLocal++ (Di Gaspero e Schaerf)
- Ve 4/6
  - Metodi formali per OO (Miculan) (?)
- Lu. 7/6
  - OOA
- Me. 9/6
  - Caso di studio (con Coppola)
- Ve. 11/6
  - Chiusura corso
  - L'esame
  - Argomenti per tesi di laurea
- Seminari (?)

Riassunto

- Esempio
- Definizione e principi generali
- I refactoring
  1. Composizione di metodi
  2. Spostamenti fra oggetti
  3. Organizzazione dei dati
  4. Semplificazione di espressioni condizionali
  5. Semplificazione di invocazioni di metodi
  6. Gestione della generalizzazione

1. Composizione di metodi

- Come arrivare a metodi ben fatti
  1. Extract Method
  2. Inline Method
  3. Inline Temp
  4. Replace Temp with Query
  5. Introduce Explaining Variable
  6. Split Temporary Variable
  7. Remove Assignments to Parameters
  8. Replace Method with Method Object
  9. Substitute Algorithm

2. Spostamenti fra oggetti

- Per decidere dove mettere le responsabilità
  1. Move Method
  2. Move Field
  3. Extract Class
  4. Inline Class
  5. Hide Delegate
  6. Remove Middle Man
  7. Introduce Foreign Method
  8. Introduce Local Extension

3. Organizzazione dei dati

- Semplificare la gestione dei dati

1. Self Encapsulate Field
2. Replace Data Value with Object
3. Change Value to Reference
4. Change Reference to Value
5. Replace Array with Objects
6. Duplicate Observed Data
7. Change Unidirectional Association to Bidirectional
8. Change Bidirectional Association to Unidirectional
9. Replace Magic Number with Symbolic Constant
10. Encapsulate Field
11. Encapsulate Collection
12. Replace Record with Data Class
13. Replace Type Code with Class
14. Replace Type Code with Subclasses
15. Replace Type Code with State/Strategy
16. Replace Subclass with Fields

4. Semplificazione di espressioni condizionali

- Logica condizionale spesso intricata…
  1. Decompose Conditional
  2. Consolidate Conditional Expression
  3. Consolidate Duplicate Conditional Fragments
  4. Remove Control Flag
  5. Replace Nested Conditional with Guard Clauses
  6. Replace Conditional with Polymorphism
  7. Introduce Null Object
  8. Introduce Assertion

5. Semplificazione di invocazioni di metodi

1. Rename Method
2. Add Parameter
3. Remove Parameter
4. Separate Query from Modifier
5. Parameterize Method
6. Replace Parameter with Explicit Methods
7. Preserve Whole Object
8. Replace Parameter with Method
9. Introduce Parameter Object
10. Remove Setting Method
11. Hide Method
12. Replace Constructor with Factory Method
13. Encapsulate Downcast
14. Replace Error Code with Exception
15. Replace Exception with Test

6. Gestione della generalizzazione

- Spostamenti in una gerarchia

1. Pull Up Field
2. Pull Up Method
3. Pull Up Constructor Body
4. Push Down Method
5. Push Down Field
6. Extract Subclass
7. Extract Superclass
8. Extract Interface
9. Collapse Hierarchy
10. Form Template Method
11. Replace Inheritance with Delegation
12. Replace Delegation with Inheritance

Scaletta

- Quando fare il refactoring? Quando il codice "puzza"
  - Puzze
  - Associazioni Puzze-Refactoring
- Big refactorings (cenni)

Catalogo di puzze (Bad smells)

1. Duplicated Code
2. Long Method
3. Large Class
4. Long Parameter List
5. Divergent Change
6. Shotgun Surgery
7. Parallel Inheritance Hierarchies
8. Feature Envy
9. Data Clumps
10. Primitive Obsession
11. Switch Statements
12. Lazy Class
13. Speculative Generality
14. Temporary Field
15. Message Chains
16. Middle Man
17. Inappropriate Intimacy
18. Alternative Classes with Different Interfaces
19. Incomplete Library Class
20. Data Class
21. Refused Bequest
22. Comments

1. Duplicated Code

- Stesso codice
  - In 2+ metodi della stessa classe
    - → 1.1 Extract Method
  - In 2+ sottoclassi "sorelle"
    - → 1.1 Extract Method^2 (2 volte) + 6.2 Pull Up Method^2
    - → 1.1 Extract Method^2 + 6.2 Form Template Method
    - → 1.9 Substitute Algorithm
  - In 2+ classi non correlate
    - → 2.3 Extract Class

2. Long Method

- Scegliete buoni nomi!
  - (con buoni nomi non serve leggere il corpo…)
- 99% delle volte → 1.1 Extract Method
  - Tante variabili locali e parametri
    - → 1.4 Replace Temp with Query
    - → 5.9 Introduce Parameter Object
    - → 5.7 Preserve Whole Object
    - → 1.3 Inline Temp (?)
    - → 1.8 Replace Method with Method Object
    - → 4.1 Decompose Conditional

3. Large Class

- Indizio: tante variabili d'istanza
  - → 2.3 Extract Class
  - → 6.6 Extract Subclass
  - → 3.2 Replace Data Value with Object
- Indizio: tanto codice
  - → 2.3 Extract Class
  - → 6.6 Extract Subclass
  - → 6.8 Extract Interface
- Large GUI Class
  - → 3.6 Duplicate Observed Data

4. Long Parameter List

- OO diverso da procedurale
  - Procedurale: bisogna passare tutto come parametri, l'alternativa sono le variabili globali…
  - OO: lasciamo che sia il destinatario a prelevarsi le info che gli servono
    - → 5.8 Replace Parameter with Method
    - → 5.7 Preserve Whole Object
    - → 5.9 Introduce Parameter Object
- … ma occhio alle dipendenze…

5. Divergent Change

- Se ≠ variazione reqs ⇒ cambiamenti in punti ≠ di una classe
  - "Qs. 3 metodi della classe se nuovo database, qs. 4 se nuove regole di addebito"
- Probabilmente più oggetti sarebbero meglio di uno
  - → 2.3 Extract Class
- Boh?

6. Shotgun Surgery

- Simile al precedente (ma ≠)
- Se ogni variazione reqs ⇒ tanti piccoli cambiamenti in più punti del codice
  - → 2.1 Move Method
  - → 2.2 Move Field
  - → 2.4 Inline Class
- Idealmente dovrebbe esserci una relazione 1:1 fra tipo di cambiamento e classe da modificare
- Boh?

7. Parallel Inheritance Hierarchies

- Caso speciale di 6. Shotgun Surgery
  - → 2.1 Move Method
  - → 2.2 Move Field

8. Feature Envy

- Metodo che sembra più interessato a un'altra classe che a quella in cui è
  - Oggetto dell'invidia: dati di un'altra classe (get…)
    - → 2.1 Move Method [dopo 1.1 Extract Method]
  - Oggetto dell'invidia: dati in molte altre classi
    - → 1.1 Extract Method*
    - → 2.1 Move Method*
  - → 2.2 Move Field
- Non sempre si mette il metodo nella classe con i dati su cui lavora: Visitor, Strategy, …
- Mettere insieme le cose che cambiano insieme

9. Data Clumps

- "Grumi" di dati (gruppi di attributi/parametri)
- Meglio metterli in una classe apposita:
  - → 2.3 Extract Class
  - → 5.9 Introduce Parameter Object
  - → 5.7 Preserve Whole Object
- Ora che avete nuovi oggetti, cercate casi di 8. Feature Envy

10. Primitive Obsession

- Non abbiate paura di fare piccole classi per oggetti semplici (`Date`, `String`, `Money`, `Interval`, `ZIP`, `PhoneNumber`, …)
  - → 3.2 Replace Data Value with Object
  - → 3.13 Replace Type Code with Class
  - → 3.14 Replace Type Code with Subclasses
  - → 3.15 Replace Type Code with State/Strategy
  - → 2.3 Extract Class
  - → 5.9 Introduce Parameter Object
  - → 3.5 Replace Array with Object

11. Switch Statements

- Switch (e codice duplicato) invece di polimorfismo
  - → 1.1 Extract Method, 2.1 Move Method
  - → 3.14 Replace Type Code with Subclasses
  - → 3.15 Replace Type Code with State/Strategy
  - → 4.6 Replace Conditional with Polymorphism
  - → 5.6 Replace Parameter with Explicit Methods
  - → 4.7 Introduce Null Object

12. Lazy Class

- Classe che non sta (più) facendo molto…
  - → 6.9 Collapse Hierarchy
  - → 2.4 Inline Class

13. Speculative Generality

- "Potrebbe essere che un giorno questo serva, e anche questo, e…"
  - → 6.9 Collapse Hierarchy
  - → 2.4 Inline Class
  - → 5.3 Remove Parameter
  - → 5.1 Rename Method

14. Temporary Field

- Oggetto con attributo d'istanza non sempre settato
  - → 2.3 Extract Class
  - → 4.7 Introduce Null Object

15. Message Chains

- Lunghe catene di messaggi getXxx (con, eventualmente, variabili temporanee)
  - → 2.5 Hide Delegate*
  - → 1.1 Extract Method, 2.1 Move Method

16. Middle Man

- Troppi metodi pubblici di una classe delegano a un'altra classe
  - → 2.6 Remove Middle Man
  - → 2.1 Inline Method
  - → 6.12 Replace Delegation with Inheritance

17. Inappropriate Intimacy

- Classi troppo "intime"
  - → 2.1 Move Method
  - → 2.2 Move Field
  - → 3.8 Change Bidirectional Association to Unidirectional
  - → 2.3 Extract Class
  - → 2.5 Hide Delegate
  - → 6.11 Replace Inheritance with Delegation

18. Alternative Classes with Different Interfaces

- Classi che dovrebbero avere la stessa interfaccia (ad es., per polimorfismo) (?)
  - → 5.1 Rename Method
  - → 2.1 Move Method
  - → Adapter (?)

19. Incomplete Library Class

- Libreria a cui vorreste aggiungere metodi/servizi/funzionalità
  - → 2.7 Introduce Foreign Method
  - → 2.8 Introduce Local Extension
- N.B. 2.1 Move Method non va bene perché non possiamo modificare le classi della libreria…

20. Data Class

- Classi "stupide", con solo metodi get e set
  - → 3.10 Encapsulate Field se attributi pubblici
  - → 3.11 Encapsulate Collection
  - → 5.10 Remove Setting Method
  - → 1.1 Extract Method, 2.1 Move Method
  - → 5.11 Hide Method sui metodi get e set
- Data class = bambini: ok come punti di partenza, ma poi devono crescere…

21. Refused Bequest

- Eredità rifiutata: sottoclasse che eredita metodi/attributi ma non li vuole
  - Gerarchia sbagliata
    - → 6.4 Push Down Method
    - → 6.4 Push Down Field
  - Non è una puzzona… A meno che il rifiuto sia sull'interfaccia:
    - → 6.11 Replace Inheritance with Delegation

22. Comments

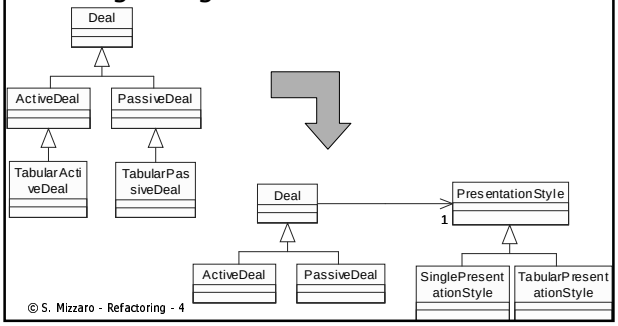
- Scrivete i commenti!!!
- Commenti usati come deodorante: li si scrive perché il codice è mal fatto. Sono superflui
  - Invece di commentare un frammento di codice, → 1.1 Extract Method
  - Invece di commentare cosa fa un metodo → 5.1 Rename Method
  - Invece di usare un commento per asserire qualche condizione → 4.8 Introduce Assertion
- invece di commentare, una buona idea è di scrivere/estrarre un metodo…

Big Refactorings

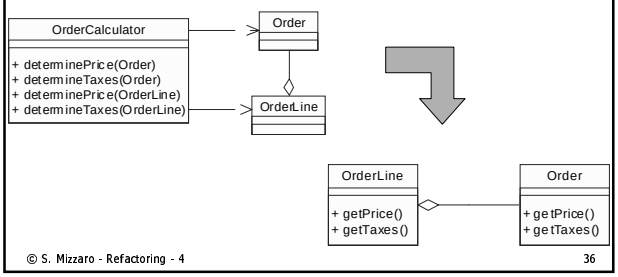
- Quelli visti finora: in minuti, al max. 1 ora
- Questi: anche mesi!
- Esempi:
  - Tease Apart Inheritance
  - Convert Procedural Design to Objects
  - Separate Domain from Presentation
  - Extract Hierarchy

Tease Apart Inheritance

- "Sbrogliare" gerarchie

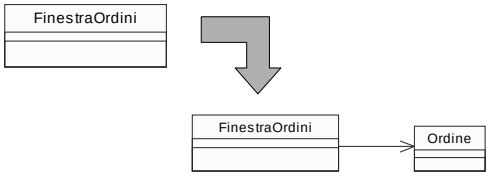


Convert Procedural Design to Objects

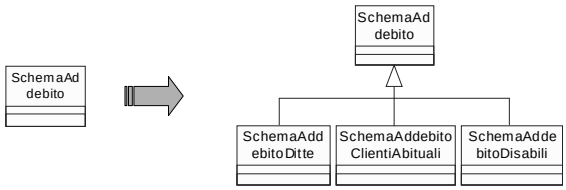


Separate Domain from Presentation

- 2 tier → N-tier
- Separare interfaccia (GUI) e presentazione (logica di dominio)



Extract Hierarchy



Riassunto

- Refactoring
- Principi ed esempi
- I 72 refactoring
- Le 22 puzze
  - E le associazioni con i refactoring corrispondenti
- Cenni ai "Big refactoring"

Commenti finali

- Mi aspetto che dal nome riusciate a ricordarvi "abbastanza" il refactoring
- Refactoring è rischioso
  - Andare con i piedi di piombo
- Tool automatici aiutano
  - Esistono tool che sentono le puzze?