Wzorce projektowe i refaktoryzacja

Paweł Kozioł
p.koziol@students.mimuw.edu.pl

18.01.2005

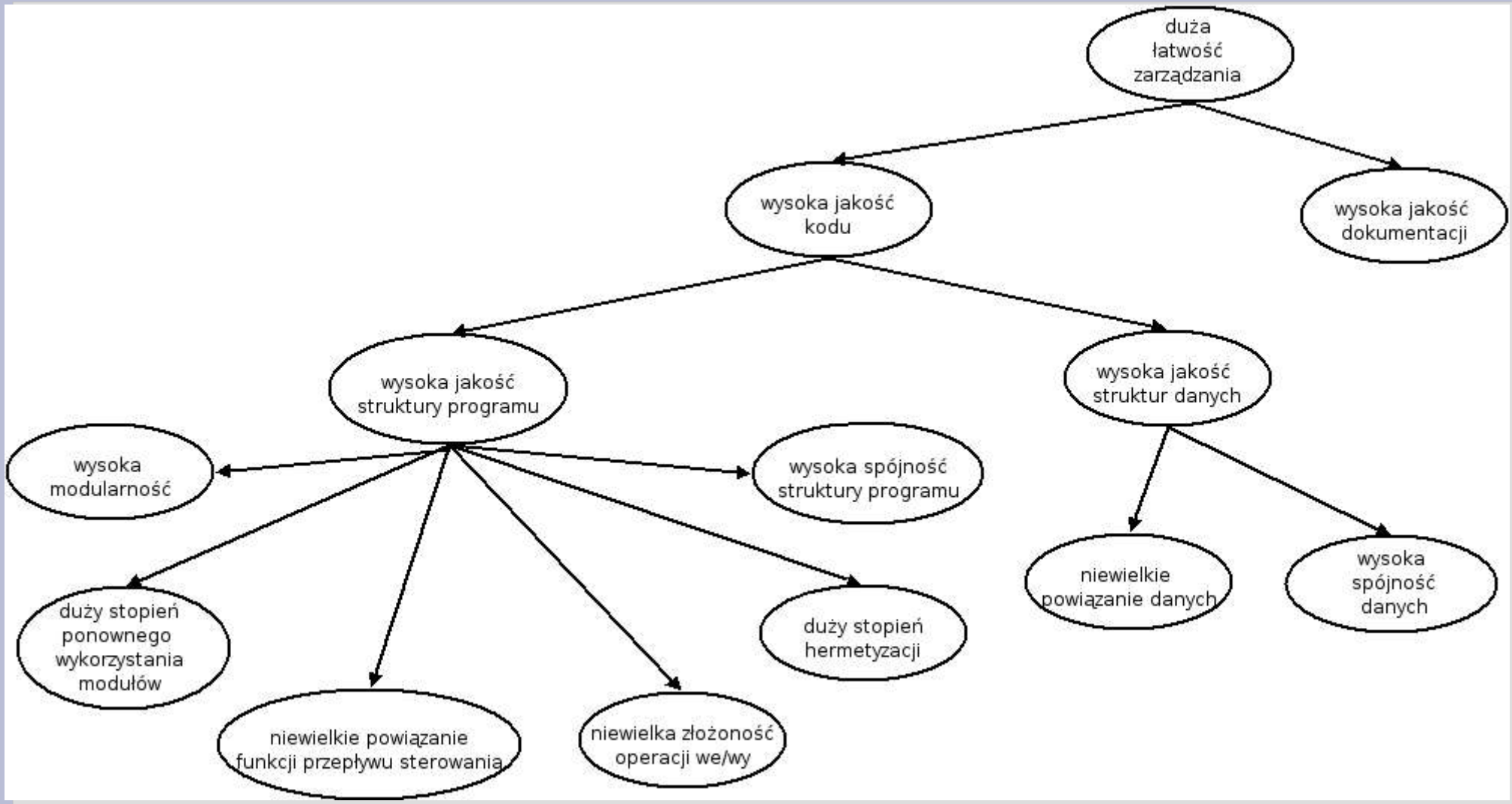
Moja praca magisterska

- “Narzędzie dla środowiska Eclipse wspierające stosowanie wzorców projektowych J2EE”
- Prowadzący: dr Janusz Jabłonowski

Cele:

- Wsparcie pracy programisty prowadzące do:
  - Przyspieszenia pisania kodu
  - Wzrostu:
    - Rozszerzalności
    - Skalowalności
    - Niezawodności
    - Terminowości tworzenia
  - Polepszenia **jakości kodu**

Jakość kodu


duża łatwość zarządzania
wysoka jakość kodu
wysoka jakość dokumentacji
wysoka jakość struktury programu
wysoka jakość struktur danych
wysoka modularność
wysoka spójność struktury programu
niewielkie powiązanie danych
wysoka spójność danych
duży stopień ponownego wykorzystania modułów
duży stopień hermetyzacji
niewielkie powiązanie funkcji przepływu sterowania
niewielka złożoność operacji we/wy

Sposób osiągnięcia celów

- Wprowadzenie w kodzie wzorców projektowych
  - Przez generowanie wzorcowego kodu w konkretnych zastosowaniach (nowy kod tworzony jest od zera)
  - Przez refaktoryzację do wzorców projektowych (modyfikowany jest istniejący kod)

Wzorce projektowe - definicja

- “Każdy wzorzec opisuje problem,
- który powtarza się wielokrotnie w naszym środowisku,
- a następnie opisuje modelowe rozwiązanie tego problemu
- w taki sposób, że można to rozwiązanie wykorzystać milion razy,
- tak, że żadne dwa wykorzystania nie będą identyczne”

Co dają wzorce

- Wzrost jakości oprogramowania
- Wspólny język dla programistów
  - Ułatwiają wymianę doświadczeń
  - Ułatwiają specyfikację wymagań
- Spadek czasu tworzenia oprogramowania:
  - Projektowania
  - Kodowania
  - Testowania
- Stanowią zbiór dobrze zdefiniowanych i sprawdzonych (przetestowanych) gotowych rozwiązań

Założenia pracy

- Duża część kodu w zastosowaniach wzorców projektowych jest **zawsze taka sama**
- Duża część pozostałego kodu zależy od niewielu parametrów i **może być generowana automatycznie**

Generowanie kodu

- Wspomaga pracę programisty
- Znane np. z narzędzi programistycznych
  - Szablony klas
  - Kreatory klas
- Stosowane także w narzędziach typu Javadoc, Xdoclet czy Embeded SQL
- Jest wygodne, przyspiesza pracę, pozwala uniknąć pomyłek

Przykłady wzorców projektowych

- Singleton
  - Zapewnia istnienie dokładnie jednej instancji (jednego obiektu) danej klasy w ramach jednej maszyny wirtualnej
- Fabryka
  - Umożliwia tworzenie obiektów o identycznym interfejsie i realizujących te same zadania, ale w różny sposób. Np. dostęp do plików
- Dekorator
  - Dostarcza sposobu na zmianę własności indywidualnych obiektów bez potrzeba tworzenia klasy pochodnej. Np. Lampa

Przykłady wzorców projektowych (c.d)

- Fasada
  - Upraszcza złożoność skomplikowanych podsystemów dostarczając dla nich prostego interfejsu
- Iterator
  - Pozwala przechodzić po elementach kolekcji bez wiedzy o jej implementacji
- Obserwator
  - Pozwala na informowanie zależnych obiektów o zmianie stanu pewnego innego obiektu

Wzorce – literatura

- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison–Wesley, 1995.
- Allan Shalloway, James R. Trott, ***Projektowanie zorientowane obiektowo: Wzorce projektowe***, Helion, 2001.
- James W. Cooper, ***Java: Wzorce projektowe***, Helion 2001.

Refaktoryzacja – definicja

- Przekształcenie zmieniające strukturę kodu nie zmieniające jego działania (program przed i po przekształceniu uruchomiony na tych samych danych da dokładnie ten sam wynik)
- Jak w matematyce - liczbę 6432 można reprezentować jako 32 * 3 * 67. Refaktoryzując ja do 8 * 12 * 67 otrzymamy dokładnie ten sam produkt (6432) ale o innej strukturze.

Refaktoryzacja - cele

- Ułatwia ona budowę i pielęgnację oprogramowania, umożliwiając utrzymanie systemu komputerowego w stanie pozwalającym na łatwą rozbudowę.
- Refaktoryzacja jest znacznie tańsza niż przepisywanie kodu od nowa
- Zmiana struktury programu ma na celu poprawę jakości kodu

Refaktoryzacja - automatycacja

- W niektórych przypadkach możemy ręcznie zmienić kod.
- W dużych programach niezbędna jest automatyzacja refaktoryzacji.
- Bardziej złożone refaktoryzacje można wykonać jako złożenie kilku prostszych

Refaktoryzacja - przykłady

- Zmiana nazwy
  - metody,
  - pola,
  - parametru

Mało mówiącą nazwę parametru

```
void C::cos() {
   String zm150 = “jakiś”;
    zm150 += “ napis”;
   wypisz(zm);
}
```

zmieniamy na:

```
void C::cos() {
  String informacja = “jakiś”;
  informacja += “ napis”;
  wypisz(informacja);
}
```

Refaktoryzacja – przykłady (c.d.)

- Wydzielenie metody

dzieli metodę jeśli jest ona za długa, robi więcej niż jedną rzecz lub miesza nisko- i wysokopoziomowe operacje, np:

```
void C::cos() {
   for(int i; i<a.length(); i++)
      a[i] = rand();
   wypisz(a);
}
```

zmieniamy na:

```
void C::wylosuj() {
   for(int i; i<a.length(); i++)
      a[i] = rand();
}

void C::cos() {
   wylosuj();
   wypisz();
}
```

Refaktoryzacja – przykłady (c.d.)

- Przeniesienie metody
- Wydzielenie wartości wyrażenia w postaci stałej.
- Przeniesienie funkcji w górę hierarchi dziedziczenia
- Rozwinięcie metody (inline)
- Wyodrębnienie interfejsu z klasy

Refaktoryzacja - literatura

- Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts, ***Refactoring: Improving the Design of Existing Code***, Addison-Wesley, 1999.
- refactoring.com