

Wprowadzenie do programowania aspektowego

Michał Stochmiątek



Plan prezentacji

Problem

- Wymagania klienta
- Ortogonalność

Programowanie obiektowe

- Apache Tomcat OOP?
- Problem na przykładzie

Programowanie aspektowe

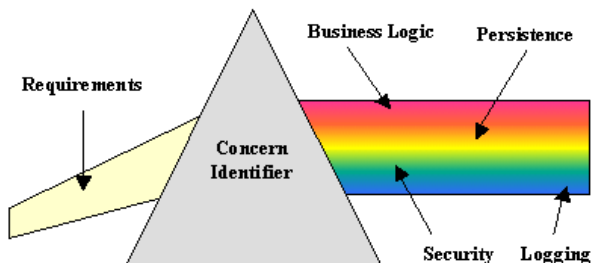
- Idea
- Przykład
- Tkacz

Wielowymiarowość wymagań klienta

Klient wymaga oprócz realizacji głównego zagadnienia systemu, również inne poboczne np:

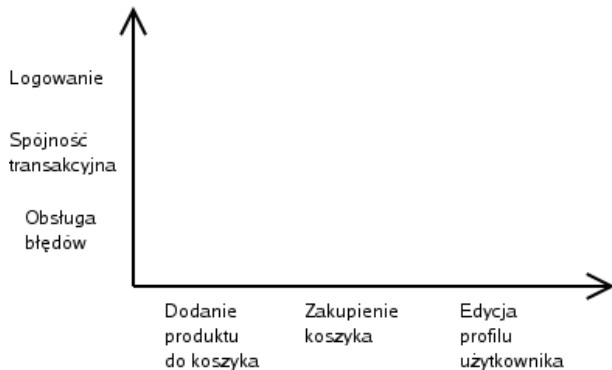
- ▶ logowanie transakcji do dziennika systemowego
- ▶ spójność transakcyjną
- ▶ autoryzację użytkowników
- ▶ rozproszenie obliczeń

Metafora pryzmatu



[Źródło: Laddad, I want my AOP!, part 1]

Ortogonalność wymagań?



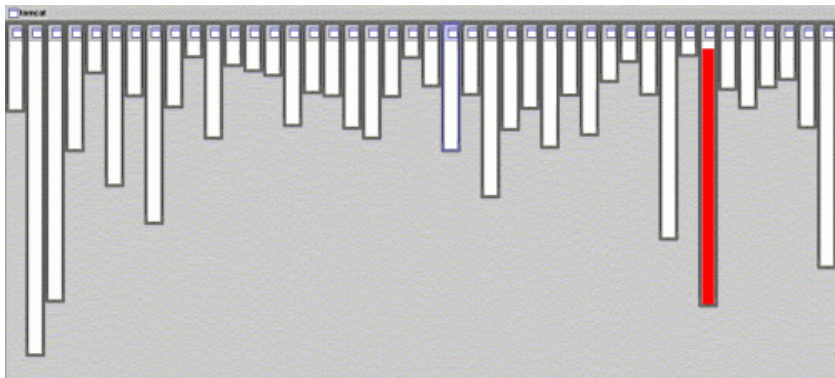
Czy implementacja jest na to przygotowana?

Programowanie obiektowe

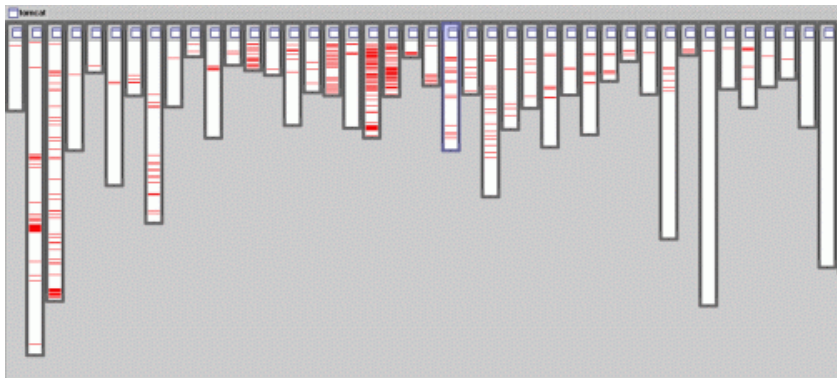
Pozwala na modelowanie systemów informatycznych w kategoriach świata rzeczywistego, dostarczając:

- ▶ klasy
- ▶ generalizacja
- ▶ enkapsulacja

Apache Tomcat - realizacja przetwarzania plików XML



Apache Tomcat - obsługa logowania



Problem na przykładzie

```
public void sellSelfDefenceTShirts(Client client, int num)
{
    super.getLogger().info("Klient " + client.getFullName()
        + " rozpoczął transakcję zakupu koszulek.");
    UserTransaction ut = getUserTransaction();
    try {
        ut.begin();

        // *** wykonaj operację zakupu koszulek ***

        ut.commit();
        super.getLogger().info("Klient " + client.getFullName()
            + " pomyślnie zakończył transakcję zakupu koszulek.");
    } catch (Exception e) {
        ut.rollback();
        super.getLogger().info("Przerwana transakcja zakupu koszulek"
            + " (klient: " + client.getFullName() + ").");
    }
}
```

Pomysł na refaktoryzację?

Czy metoda nie powinna wyglądać tak?

```
public void sellSelfDefenceTShirts(Client client, int num)
{
    // *** wykonaj operację zakupu koszulek ***
}
```

Konsekwencje przeplatania kodu

- ▶ programista nie skupia się na właściwej odpowiedzialności danej klasy czy metody
- ▶ zmniejszenie zdolności do ponownego użycia danego kodu
- ▶ rozproszenie kod związanego z ortogonalnymi wymaganiami klienta
- ▶ trudna do refaktoryzacji duplikacja kodu źródłowego

Rozwiązanie

Paradygmat programowania aspektowego

Aspect-Oriented Programming - AOP

- ▶ rozszerza paradygmat programowania obiektowego
- ▶ dodaje nowy wymiar do programowania
- ▶ **Aspekt** - jednostka modularności analogiczna do klasy



Anatomia programu aspektowego

- ▶ Aspekt (*aspect*) - analogiczna do klasy, jednostka modularności
- ▶ Rada (*advice*) - analogiczna do metody, zawiera kod wykonywany przez aspekt
- ▶ Punkty łączy (*join points*) - miejsca uruchomienia rad w kodzie obiektowym (połączenie kodu aspektowego i obiektowego)
- ▶ Linie podziału (*pointcuts*) - zbiór punktów łączy

Nieśmiertelny HelloWorld

```
public class HelloWorld {  
    public static void say(String message) {  
        System.out.println(message);  
    }  
  
    public static void sayToPerson(String message, String name) {  
        System.out.println(name + ", " + message);  
    }  
}
```

[Źródło: Laddad, I want my AOP!, part 2]

Aspekt dodający uprzejmości

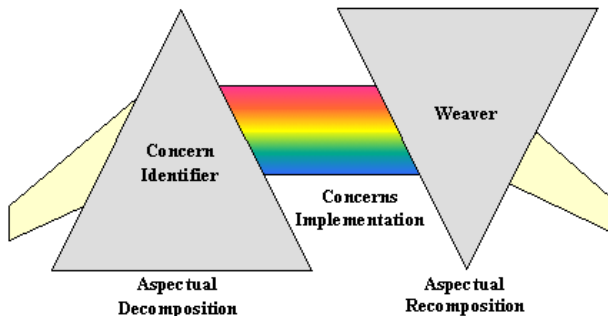
```
public aspect MannersAspect {
    pointcut callSayMessage()
        : call(public static void HelloWorld.say*(..));

    before() : callSayMessage() {
        System.out.println("Good day!");
    }

    after() : callSayMessage() {
        System.out.println("Thank you!");
    }
}
```

[Źródło: Laddad, I want my AOP!, part 2]

Tkacz - silnik AOPu



[Źródło: Laddad, I want my AOP!, part 1]

Implementacje AOP

- ▶ AspectJ - rozszerzenie Java (<http://eclipse.org/aspectj/>)
- ▶ AspectC - rozszerzenie C++ (<http://www.aspectc.org>)
- ▶ RAPIER-LOOM.NET - biblioteka .NET

Podsumowanie




Do zapamiętania

- ▶ problem ortogonalności wymagań klienta
- ▶ paradygmat programowanie aspektowe - AOP
 - ▶ rozszerzenie paradygmatu obiektowego
 - ▶ przejrzysty sposób na modelowanie ortogonalnych wymagań

Do przemyślenia :-)

- ▶ rozwiązanie problemu refaktoryzacyjnego?
- ▶ brak przyjętego języka modelowania (jak UML)

Bibliography

-  Laddad Ramnivas, *I want my AOP!, Part 1-3, Separate software concerns with aspect-oriented programming*, JavaWorld, 2002
-  AspectJ Team, *The AspectJTM Programming Guide*, <http://eclipse.org/aspectj/>
-  Stochmiątek Michał, *Wprowadzenie do programowania aspektowego*, CodeGuru.pl, 2004