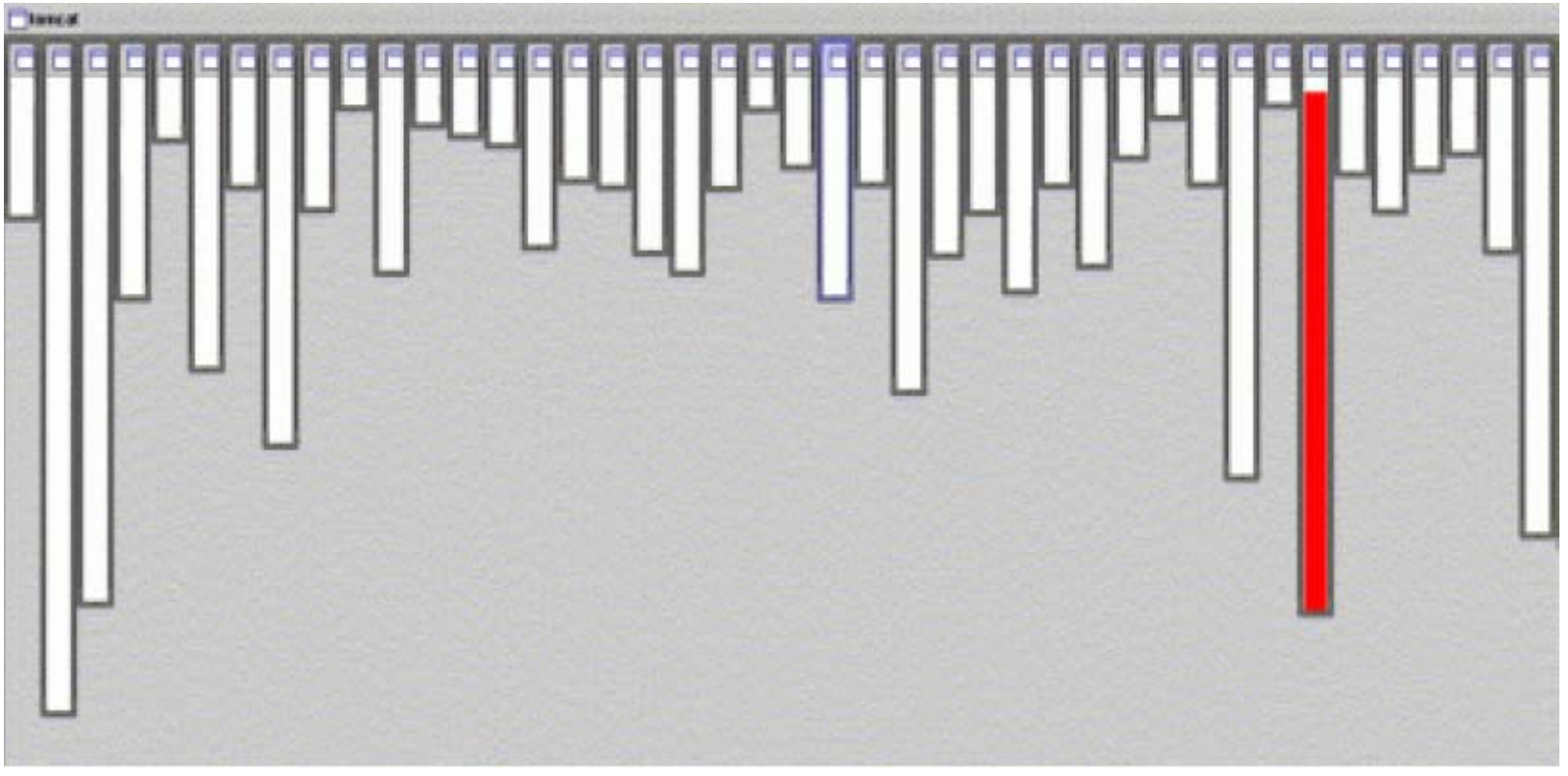
Aspect Oriented Programming

Konrad Witkowski

Programowanie obiektowe

- klasy, enkapsulacja, dziedziczenie
- pozwalają na modelowanie systemów informatycznych w kategoriach świata zewnętrznego
- jest to naturalne dla człowieka
- klasa jest jednostką modularyzacji - czasami jest to jednak niewystarczające

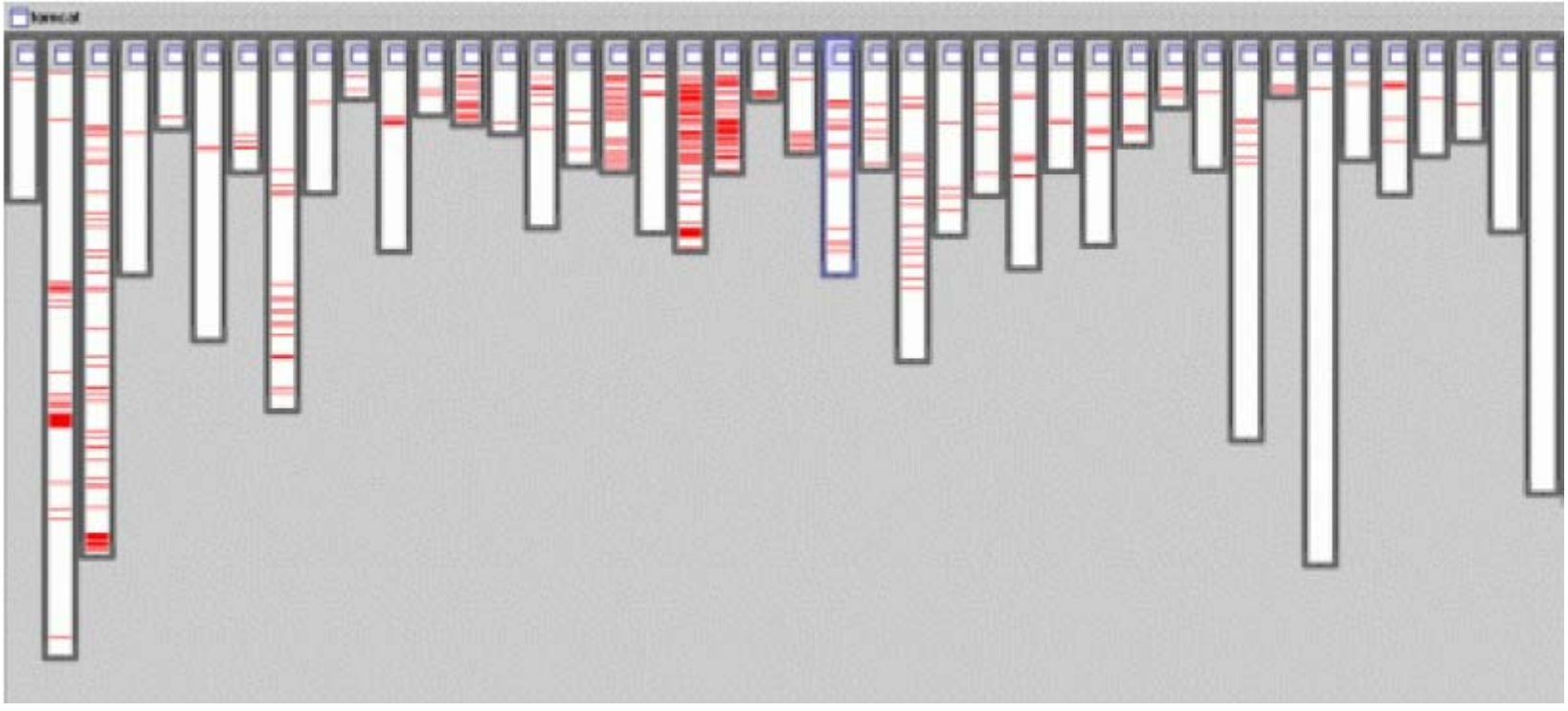
Przykład : Apache Tomcat



Podzbiór klasy serwera Tomcat. Na czerwono zaznaczony kod odpowiedzialny za przetwarzanie plików XML.

Przykład : Apache Tomcat

Są problemy, których enkapsulacja w pojedynczej klasie nie jest możliwa.



Implementacja logowania jest rozproszona po większości klas.

Prosty przykład

```
class Foo
{
        void foo(String s) {
                logger.log("enter foo : "+s);
                //logic;
                logger.log("exit foo");
        }

        int foo(int i) {
                logger.log("enter foo : "+i);
                //logic;
                logger.log("exit foo : "+result);
                return result;
        }
}
```

Dlaczego OOP się nie sprawdza
(przeplatanie realizacji poszczególnych wymagań)

- programista nie koncentruje się właściwej funkcjonalności kodu
- zmniejszone możliwości ponownego użycia danego kodu
- kod związany z pewnymi aspektami jest rozproszony w wielu miejscach
- trudno nadążać ze zmianami
- trudno wyjaśnić nowym członkom projektu

Co byśmy chcieli mieć

```
class MyClass
{
        void doSomething() {
                //do something;
        }

        void doSomethingElse() {
                //do something else;
        }
}
```

Modularyzacja funkcjonalności

- chcemy podzielić system względem realizacji konkretnych celów (wymagań) w łatwe do zarządzania części
- OOP nie wystarcza – chcemy podziału na wyższym poziomie
- np. bezpieczeństwo, śledzenie, logowanie, obsługa błędów

Historia AOP

- badania rozpoczęto we wczesnych latach 90-tych w firmie Xerox
- projektem zarządza Gregor Kiczales, założyciel *Aspect-Oriented Software Association* (AOSA), później kierujący *AspectJ*.

Programowanie aspektowe

- rozszerza paradygmat OOP – dodaje idee aspektów
- odwzorowuje aspekty funkcjonalne na aspekty implementacji
- unikamy przeplatania kodu
- możemy się skoncentrować na zagadnieniach ważnych w danym miejscu
- łatwo dodawać i usuwać funkcjonalność
- lepsza dekompozycja systemu niż w przypadku samych klas
- klasy zawierają tylko to za co odpowiadają

Mechanizmy

- przeplatanie kodu istnieje jednak jest niewidoczne dla programisty
- przeplatanie (*ang. weaving*) jest wykonywane automatycznie w czasie kompilacji (statyczne) lub wykonywania programu (dynamiczne)
- kod zawarty w aspektach jest dodawany do do klasy w punktach złączeń (*join points*)
- np. przed wywołaniem metody, po przypisaniu na zmienną

Aspekt

- *Aspekt* – jednostka modularyzacji analogiczna do klasy w OOP
- *Advice* – analogicznie do metody, zawiera kod wykonywany w aspekcie
- *Join Point* – punkt, w którym aspekt ma zastosowanie
- *Pointcut* - zbiór *join point'*ów i wartości, w których wykonywany jest odpowiedni *advice*.

Join Points

Cross-cutting concerns

- przed wywołaniem metody
- jeżeli podniesiony wyjątek
- przy każdej zmianie wartości
- zamiast wywołania
- ...

AspectJ

- dodaje konstrukcje aspektowe do Java'y
- rozwijany od 1997, pierwsze publiczne wydanie w 1998, wersja 1.0 w 2001
- kompilator (*ajc*) – *compile time weaving*
- mechanizmy AOP + modyfikacja kodu klasy (np. dodanie metody do klasy), zmiany w relacjach dziedziczenia

Poincut w AspectJ

- wywołanie metod
- wykonywanie metod
- obsługa wyjątków
- utworzenie obiektu danej klasy
- wywołanie konstruktora
- dostęp do pola

Wady

- nie widzimy jaki kod jest wykonywany w danym momencie
- trudno refaktoryzować(jak się zmieni sygnatura metody)
- narzędzia są w fazie badań i rozwoju

Rozwój AOP

- częsty temat na konferencjach poświęconych OOP
- realizacje oparte zarówno na językach obiektowych (Java, C++, Smalltalk), jak i proceduralnych jak C.
- eksperymentalne języki tworzone od podstaw
- powoli AOP znajduje zastosowanie w problemach wyższego ryzyka
- brak języka modelowania (UML)
- brak wzorców projektowych
- najbardziej brakuje programistów piszących aspektowo

Literatura

- Laddad Ramnivas, *I want my AOP!, Part 1-3, Separate software concerns with aspect-oriented programming,* JavaWorld, 2002
- Strona AOSA, *www.aosd.net*
- *http://eclipse.org/aspectj/* - strona domowa AspectJ
- *http://aspectwerkz.codehaus.org* - podobny projekt
- LOOM.NET (rapier,gripper)
- AspectC
- AspectC#
- AspectC++
- AspectS - Smalltalk