

Aspektorientierte Programmierung

- Objektorientierung: Softwarestruktur folgt der Struktur der Daten.
- Prozedurale Programmierung: Struktur folgt der Struktur der Aufgaben.
- Manche Aktionen erfolgen quer zur Softwarestruktur (cross cutting concern)
- Aspektorientierte Programmierung hat das Ziel einem objektorientierten Programm weitere Aspekte hinzuzufügen, ohne dessen Struktur zu zerstören.
- Aspektorientierung wird „orthogonal“ dem Code (oder Bytecode) hinzugefügt.
- Dieses „Weben“ kann statisch (Code) oder dynamisch (ClassLoader) erfolgen.

Anwendungen der Aspektorientierten Programmierung

- Debugging: Trace, Überwachung von Variablenzugriff.
- Profiling: man misst gezielt die Ausführungszeit einzelner Aktionen.
- Protokollieren.
- Verhalten, dass mit der Programmlogik nichts zu tun hat (Transaktionen...)
- Hinzufügen von Entwurfsmustern.
- Konfiguration von Software (dependency injection in Frameworks)

Begriffe

- Crosscutting concern: Aufgabe die „quer“ zur Struktur liegt.
- Join Point: Ort an dem eine Aktion ausgeführt werden kann.
(Aufruf einer Methode, Verlassen einer Methode, Zugriff auf Variable, ..)
- Point Cut: Menge von Join Points für die Aktionen definiert werden.
- Advice: Aktion, die mit einem Point Cut verbunden wird.
- Inter-type declarations: Type Deklarationen für andere Klassen
- Aspect: Klasse mit AOP-Definitionen (Aspekte sind Singleton Objekte)
- Weaving: Verbinden der Aspekte mit dem Programmcode.
- AspectJ: Werkzeuge zur Aspektorientierung in Java. Entwickelt von Xerox Parc
- AdjT: Eclipse-Plugin für AspectJ

Anmerkung: AspectJ und ähnliche Werkzeug werden in Webframeworks verwendet.

Beispiel: Wie häufig wird eine bestimmte Methode aufgerufen?

```
public aspect MethodCounter {
    pointcut myMethodCall(): execution(* Klasse.methode(..))

    before() myMethodCall() {
        counter++;
    }

    after(): execution(static void Klasse.main(String[])) {
        System.out.println("Aufrufe: %d%n", count);
    }
}
```

Ergebnisse

- Aspekte definieren Debugging-Aktionen, etc.: keine Änderung der Funktionalität
 - Aspekte beeinflussen die Performance: keine Änderung der Funktionalität
 - Aspekte sichern ein bestimmtes Verhalten: keine Änderung der Programmlogik
 - Aspekte führen Fehlerkontrolle durch: keine Änderung der Programmlogik
 - Aspekte implementieren Muster: Erweiterung der Funktionalität
 - Meist erfolgt das „Weben“ bei der Übersetzung.
 - Es können aber auch „fertige“ Anwendungen modifiziert werden (byte code).
 - Aber: wie gewährleistet man die Konsistenz des Quellcodes?
-
- Aspekte scheinen für viele Aktionen gut geeignet.
 - Es ist aber unklar ob sich (außerhalb von Frameworks) der Einsatz durchsetzt.
 - Es gibt keine klaren Programmiermethoden und Techniken.