

Aspect oriented fault tolerance

Ruben Alexandersson



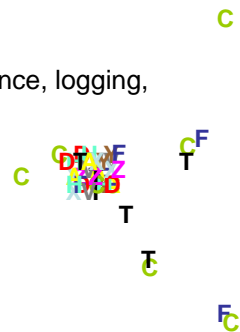
Introduction to AOP (1/2)

A program rarely do only one thing. It can, apart from its primary function include functionality for:

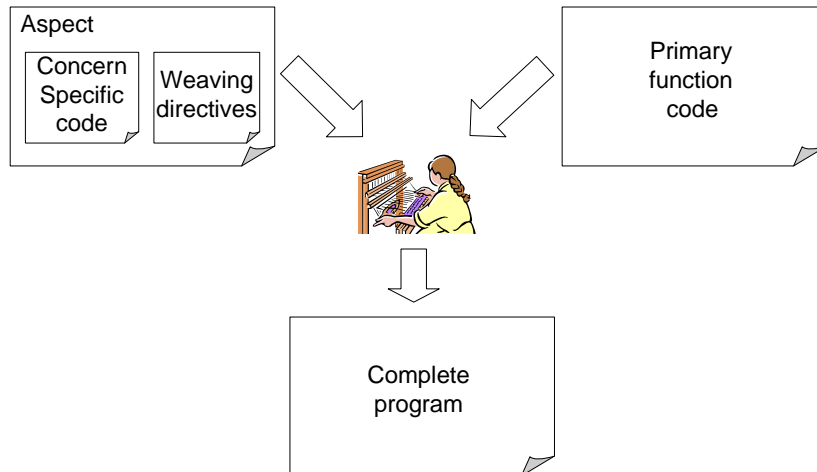
Diagnostics, fault tolerance, data persistence, logging, authentication, etc...



Bad modularization (scattering, tangling)



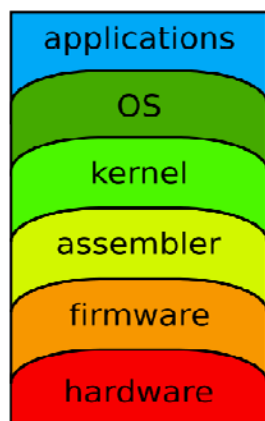
Introduction to AOP (2/2)



CEDES
Cost Efficient Dependable Electronic Systems



Setting the scope



- The idea behind AOP can be applied at any abstraction level

- We have chosen to work on application level
 - Most code scattering at this level
 - Demanding context for the AOP language
 - Potential for resource usage reduction

CEDES
Cost Efficient Dependable Electronic Systems



Using AOP for application level fault tolerance

- Both systematic and implementation specific mechanisms can be efficiently implemented
- The coverage of different mechanisms can easily be controlled
- ⇒ Allows for customized implementations for different target applications in order to reduce resource utilization
- A high level of reuse can be achieved



Work done and in progress

- Proof of concept implementations in:
 - Java
 - C++
- Language extensions to AspectC++
- Runtime performance measurements
 - Show potential benefit
 - Certify overhead cost being acceptable



Control flow checking

```
aspect ControlFlowChecking{  
  
    pointcut virtual monitoredFunctions() = 0;  
    pointcut ControlFlowExe() = execution(monitoredFunctions());  
    pointcut ControlFlowCall() = call(monitoredFunctions());  
  
    stack<int> s;  
  
    advice ControlFlowCall() : before() {  
        s.push((int)JoinPoint::JPID);  
    }  
    advice ControlFlowExe() : before() {  
        if ((int)JoinPoint::JPID != s.top()) throw "ControlFlowFaultException";  
    }  
    advice ControlFlowExe() : after(){  
        if ((int)JoinPoint::JPID != s.top()) throw "ControlFlowFaultException";  
    }  
    advice ControlFlowCall() : after(){  
        if ((int)JoinPoint::JPID != s.top()) throw "ControlFlowFaultException";  
        s.pop();  
    }  
  
};
```



Where do we go from here?

- Further research depends on performance test results:
 - Implementation of new compiler for AspectC(++)
 - Further tests on different application types

- Application in industry?
 - Dependent on commercially usable language and compiler.

