Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik
Computer Science

Information and Communication Systems Research Group

Dynamic AOP with PROSE

Angela Nicoară, Gustavo Alonso
IP8: System and Software Architecture

http://prose.ethz.ch

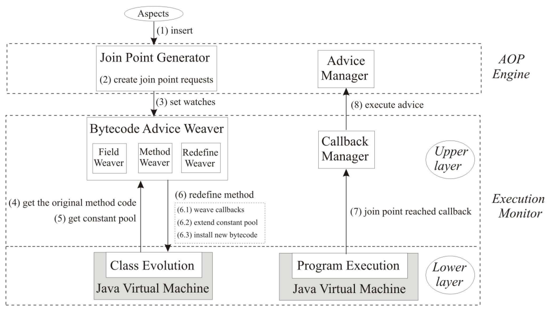
PROSE: a middleware platform for dynamic adaptation

- Applications in large distributed infrastructures, middleware, and pervasive computing environments have to adapt to changes in the environments
- They cannot behave in the same way in all possible settings
- The application should react to changes in environments for which it was not designed

- **PROSE features:**
  - Runtime weaving and unweaving of aspects
  - Weaving in local and remote VMs
  - Transactional weaving (all or nothing changes)
  - Aspects expressed in Java
  - Provide middleware tools for runtime monitoring of remote aspects

- **Key design ideas:**
  - Flexibility
    - Wide range of join-points types
    - Platform extendible with new extension constructs
    - Three different forms of weaving (JVMDI-based weaving, hook weaving, stub and advice weaving)
  - Complete API based design for added flexibility (add your own AOP engine)

PROSE stub and advice weaving architecture



- the ***AOP Engine*** accepts aspects ***(1)*** and transforms them into join-point requests ***(2)***
- it activates the join-point requests by invoking methods of the ***Execution Monitor (3)***
- the ***Bytecode Advice Weaver*** module is responsible for the bytecode manipulations and the methods instrumentation ***(4)*** - ***(6)***
- when the program execution reaches one of the activated join-points ***(7)***, the *Execution Monitor* notifies the ***AOP Engine*** which then executes the advice ***(8)***

Join-points in PROSE

- The type of join-points that can be expressed determines what can be changed in an application

```
void setX(int x) {
   this.x = x;
}
```

weaving process

weaving process

Method entry join-point

```
void setX(int x) {
   weaver.onMethodEntry();
   this.x = x;
}
```

- adds a stub to the advice before the original bytecode of a method

Method exit join-point

```
void setX(int x) {
  try {
    this.x = x;
  } finally {
    weaver.onMethodExit();
  }
}
```

- weaving done also with a stub
- more involved than weaving a method entry advice

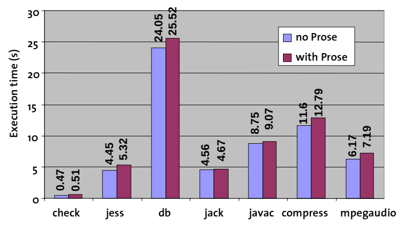
Field access/modification join-points

```
void setX(int x) {
   weaver.onFieldAccess();
   this.x = x;
}
```

- adds a stub to the advice before the field operation
- track each field reference when the class is loaded

Method redefine join-point (around) - replaces the method bodies
Exception join-points (throw and catch) - implemented by extending the runtime exceptin handler of Jikes RVM

Performance evaluation



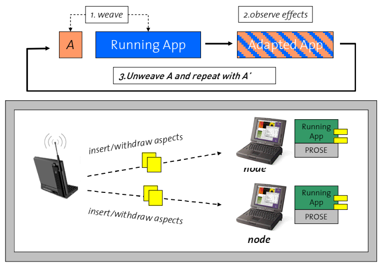
The relative overhead of the AOP enhanced JVM

Benchmark	Execution time with AOP support	
	Hook weaving	Stub and advice weaving
SPECjvm98 benchmark suite		
check	1.35 (s)	0.51 (s)
jess	34.36 (s)	5.32 (s)
db	54.23 (s)	25.52 (s)
jack	20.95 (s)	4.67 (s)
javac	37.8 (s)	9.07 (s)
compress	40.82 (s)	12.79 (s)
mpegaudio	33.62 (s)	7.19 (s)

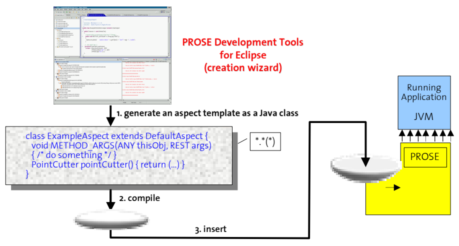
The execution times with AOP support

- Performance penalty: < 10 % overhead over non-PROSE execution using the SPECjvm98 standard benchmark

Adapting a running server



Creating aspects with PROSE


PROSE Development Tools for Eclipse (creation wizard)
1. generate an aspect template as a Java class
.(*)
2. compile
3. insert
Running Application
JVM
PROSE

```
class ExampleAspect extends DefaultAspect {
  void METHOD_ARGS(ANY thisObj, REST args)
  { /* do something */ }
  PointCutter pointCutter() { return (...) }
}
```

Schweizerischer Nationalfonds
Fonds national suisse
Swiss National Science Foundation

The National Centres of Competence in Research are managed by the Swiss National Science Foundation on behalf of the Federal Authorities

École Polytechnique Fédérale de Lausanne
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Université de Lausanne
University of St.Gallen
csem