OSGiMOP
Monitoring-Oriented Programming in OSGi

Yufang DAN, Nicolas STOULS and Stéphane FRENOT
Université de Lyon, INRIA
INSA-LYON, CITI
{yufang.dan, stephane.frenot,nicolas.stouls}@insa-lyon.fr

Introduction

MOP

- Behavior monitoring approach
- Software development technique
- Instrumentation of the implementation with the specification

OSGi

- Service oriented platform
- Implements a complete and dynamic component model

JavaMOP

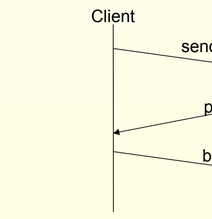
- An instantiation of MOP
- Java for declarations and event/handler actions
- Relies on AspectJ for code injection
- Logic plug-ins : FSM, ERE, CFG, PTLTL, LTL

Logos

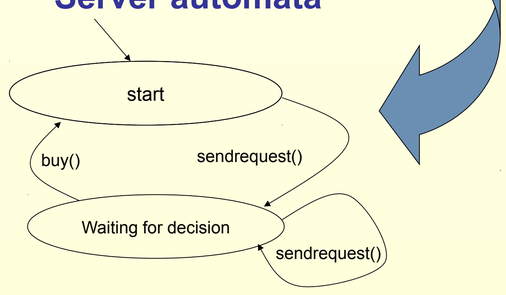
- CITI developed OSGi service
- Provide mechanisms to observe and record system calls

MOP Example

Client / Server sequence diagram



Server automata

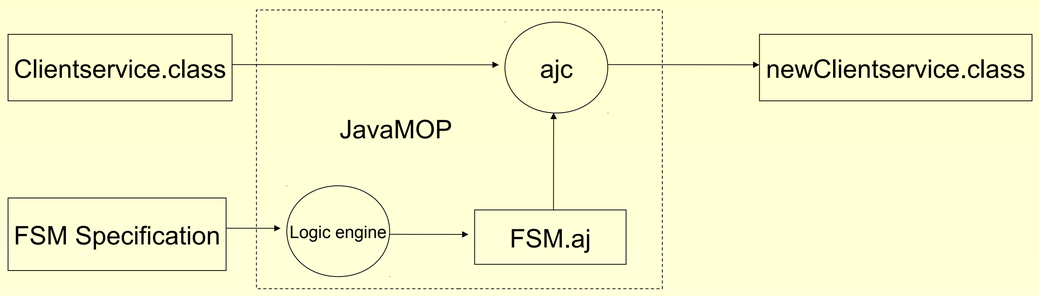


JavaMOP FSM Specification

```
package mop;
import dan.Server;
Test() {
        event sendrequest before(Server s) :
            call(* Server.sendRequest(String)) && target(s) {}
        event buyit after(Server s) :
            call(* Server.buyIt()) && target(s) {}
        fsm :
            start [
                sendrequest -> Send
            ]
            Send [
                buyit -> start
                sendrequest -> Send
            ]
        @fail {
            System.out.println("But client couldn't call buyIt method after called buyIt!");
        }
}
```

Automata Injection

Automata injection process : FSM example



AspectJ injection

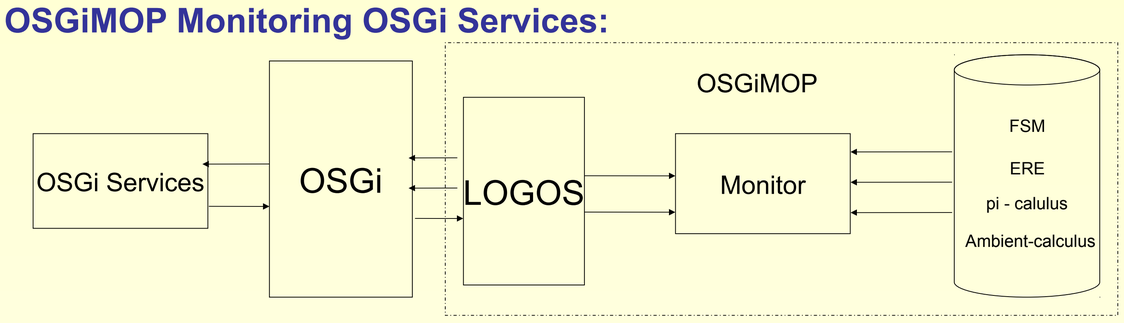
```
package mop;
import dan.Server;
Test() {
        event sendrequest before(Server s) :
            call(* Server.sendRequest(String)) && target(s) {}
        event buyit after(Server s) :
            call(* Server.buyIt()) && target(s) {}
        fsm :
            start [
                sendrequest -> Send
            ]
            Send [
                buyit -> start
                sendrequest -> Send
            ]
        @fail {
            System.out.println("But client couldn't call buyIt method after called buyIt!");
        }
}
```

OSGiMOP

What is OSGiMOP:

- Monitor for OSGi framework which dynamically checks properties
- No needs of AspectJ
- Using the existing LOGOS logging passive monitor

OSGiMOP Monitoring OSGi Services:



Advantages of OSGiMOP:

- Based on OSGi.
- Component-based development of high quality and less costly software solutions.
- Checks OSGi services conformity w.r.t their specification
- Relies on provided JavaMop plugins

Add new plug-ins:

- PI - calculus
- Ambient - calculus

Conclusions

MOP is a program behavior monitoring approach:

- Java for declarations and event/handler actions, enriched with AspectJ for event definitions.
- OSGi to monitor OSGi services.

OSGiMOP: portability, flexibility, special interest for popular Java application framework.

References

[1] Kiczales, Gregor, J.L.A.M.C.M.C.L.J.M.L.J.I.: Aspect - oriented programming, The European Conference on Object -Oriented Programming. pp. 220-242 (1997)
[2] Chen, F.: Towards monitoring - oriented programming: A paradigm combining specification and implementation. In: Electronic Notes in Theoretical Computer Science, Elsevier Science. pp. 106-125 (2003)
[3] Chen, F.: Checking and correcting behaviors of java programs at runtime with java - mop. In: Electr. Notes Theor. Comput. Sci, Springer (2005)
[4] Meredith, P.O., Jin, D., Chen, F., Rosu, G.: Efficient monitoring of parametric context - free patterns. In: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. ASE '08, Washington, DC, USA, IEEE Computer Society. pp.148-157 (2008)