Using AspectJ in Component-Based Architectures on the Server Side

Service Level Management based on Sirius Service Monitors

– Arno Schmidmeier –
Arno.Schmidmeier@sirius-eos.com
Sirius Software GmbH

April 2002

Information about Sirius

- We sell a product suite for SLM
- This product suite consists from:
  - a general usable component server,
  - Huge assembled Components called Monitor Units, which are made up from a bunch of smaller components

Successfully used

- In Commercial Project realized by the Research Department
- TMF Catalysts
  - P&P Contract/SLA Management within UMTS
  - SLM for Wireless IP
  - UMTS-IP Interconnect & Content Settlement
- Two Customized Outsourcing Service Monitors

Initial Idea

- Write only Businesslogic in a class
- -> then apply some aspects to it and you have a server component
- -> You have the ideal separation of businesscode and server specific code
- -> You can change all server policies by simply changing some aspects
- Composition is easier, Compose the Businesslogic then apply the aspects and you have an aggregated new component

usage scenarios

- Separate Business Code and Framework Code
- Move Framework Code to Aspects
- („Standard“ AOP-Design in businesslogic)
- New Architecture Design Pattern
  - To simplify the integration with EAI-Tools

Early adopter issues

- Team Communication:
  - No common sense about good aspect oriented design available
  - No widespread- named design patterns
  - No simply graphical Modelling notation available
- Tool support urgently needed for less skilled developers:
  - No reliable debugger available
  - Aspect Aware Refactoring Browser

Achieved Results (1)

- Initial Idea could be realized
- Simpler architecture by increased flexibility
- “Code quality improved”
- Class Hierarchy reflects the business domain better
- EAI-Integration got trivial and much more performant

Achieved Results (2)

- Enhancing a commercial OODBMS to an ADBMS (aspect-oriented database management system) is quite easy
- AspectJ is a great clue between Core J2EE APIs, and Business classes
- AspectJ and Core APIs are more flexible, more efficient then “Standard” J2EE architectures

Achieved Results (3)

- Code Reduction:
  - 50-90% in Components against „plain Java“
  - 25%-60% in Components against „plain Java“+ CORBA Interceptors + Java-„Dynamic Proxies“
  - 95% in Code dealing with EAI-Integration

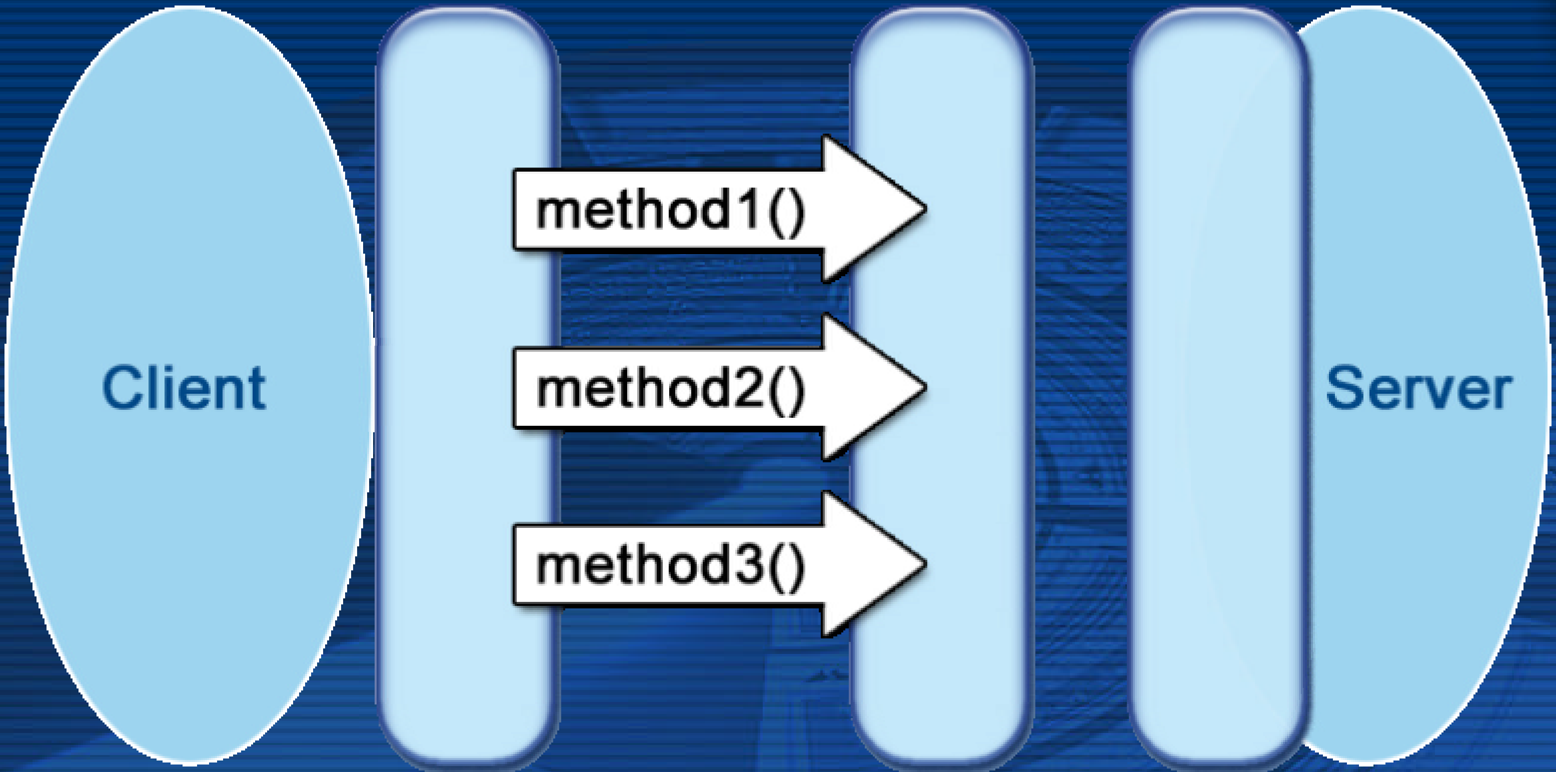
Observations (1)

- Many abstract aspects, share the same pointcuts
- We traded tangling code against tangling pointcut definitions or tangling pointcut reuse.
- Pointcut reuse could be increased by separating pointcuts in signature enumerations and signature use points

Observations (2)

- We wish therefore language extensions for:
  - a more efficient reuse of pointcut definitions
  - More flexible ways to define the importance of advices Many abstract aspects, share the same pointcuts
- Quite a lot of OOP-Pattern should be reworked or replaced by new AOP-Patterns

Signature Enumeration & Signature use points



Client

method1()

method2()

method3()

Server

Use Point
Call & this

Use Point
Call & target

Use Point
execution

Component Based compilation versus one huge concept of the compilation unit

- Place the build jobs under configuration management
- One big CU is easier to set up then several small ones, because AspectJ Code is like coffee, it should be fresh brewed

But: Separate one huge CU into several smaller ones

- These small independent build jobs is the only control for maintaining the component structure

Performance of „Framework“ Aspects

- We recognized no performance impact on the server side:
  - When we Separated Business Code and Framework Code
  - When we moved Framework Code to Aspects

- The underlying technologies have a much higher performance overhead then aspectJ

Conclusion

- AspectJ + Core J2EE APIs provide a better alternative over standard J2EE-architectures
- Quite a lot of the theoretical advantages of AOD was verified in the praxis
- Some support for tangling pointcut hierarchies should be added to AspectJ

Data Gold Service
Revenue 1083359.2
Perceived Loss 0.0

Voice Gold
Revenue 723913.2
Perceived Loss 0.0

Service Level Management Solutions

based on
Sirius Service Monitors

WORLDWIDE HEADQUARTERS

Sirius Software GmbH
Kolpingring 18
82041 Oberhaching/München
Germany
Tel: +49 (0) 89 613 676 0
Fax: +49 (0) 89 613 676 33

US HEADQUARTERS

Sirius Management Technologies Inc.
PO Box 797587
Dallas, Texas 75248
USA
Tel: +1 (972) 248 2667
Fax: +1 (972) 250 6754