Discussion of Steimann’s Essay: “The Paradoxical Success of Aspect-Oriented Programming”

Gary T. Leavens
Department Of Computer Science, Iowa State University

OOPSLA, October 26, 2006

Supported in part by the US NSF
Outline

- Summary
- Important points
- Perspective: what questions?
- Questionable points
- Conclusion
Summary: Definition and Paradox

- Need definition of AOP languages
- Definition offered ≈ implicit invocation
- Paradox:
  - Ok in moderation, but then no contribution
  - Hurts modularity if more developed
Summary: Modularity of AOP

- Information hiding (Parnas)
- “Main concern” (5.1.2):
  - Strong coupling of aspects to base
  - Implicit interface, often not public
  - Impairs independent development
- Explicit interfaces for aspects in base:
  - Less obliviousness
  - More scattering
Summary: Locality of AOP

- Implicit invocation
  = less direct control flow

- Context exposure
  = more global access to variables
Summary: Utility of AOP

- Generated code weaving
- Forming new components (glue code)
  - Dismantle components
  - Reassemble to form new components
- Observed uses:
  - Logging
  - Security
  - Runtime assertion checking
Summary: Prospects for AOP

- “Tremendous success”
  - Cachet
  - OOP is old, need something new
- Paradox (section 8):
  - Aims to “modularize crosscutting concerns”
  - “Its very nature ... breaks modularity”
Important Points: What is AOP?

- Focus on mechanisms (vs. goals)
- Need definition of AOP languages
  - Mechanisms
  - Independent of goals
- Implicit invocation definition sensible
Important Points: Modularity of AOP

- Interfaces are crucial
- Explicit interfaces for aspects in base:
  - Less obliviousness
  - More scattering
Important Points: Utility of AOP

- AOP especially useful as noted
- *Generated code weaving*
  - Aids modularity of generator
  - AspectJ = Assembly language of 2000’s
- *Glue code is important*
Important Points: Prospects for AOP

- AOP has been a success
Perspective

- Flon’s axiom (*SIGPLAN, Oct. 1975)*: “There does not now, nor will there ever, exist a programming language in which it is the least bit hard to write bad programs.”
Perspective

- Flon’s axiom (*SIGPLAN*, Oct. 1975):
  Can write bad programs in any language
Perspective: What are the right questions?

- Flon’s axiom (*SIGPLAN*, Oct. 1975):
  Can write bad programs in AOP languages

- Can we write good ones?
  - Without scattering and tangling of crosscutting concerns
  - Without excess coupling of aspects to base
Perspective: All or Nothing?

- Modularity
  - System is modular or not

- Obliviousness
  - Mechanisms are oblivious or not
Perspective: Questions of Degree

Coupling

Tangling

Scattering
Perspective: Perfect AOP Measures

- Coupling
- Tangling
- Scattering
Perspective: Without AOP

Coupling

Tangling

Scattering
Perspective: Compromise

Coupling

Tangling

Scattering
Perspective: N-Dimensions of Modularity

- Coupling
- Reasoning Difficulty
- Heap interference
- Tangling
- Scattering
Questionable Points: Coupling of Aspects and Base

- Section 5.1.2:
  - Code “moved out of its context ...”
  - “it must take (a reference to) the context that it depends on with it,”
  - “thereby establishing coupling”

- How often?
- How much coupling?
Questionable Points: Reducing Coupling

- XPIs (Griswold et al.)
  - Adding interfaces for crosscutting
  - Extra indirection reduces coupling
- Open Modules (Aldrich)
  - Explicit interface to advice
  - Reduces obliviousness
Questionable Points: Paradox

- Due to all-or-nothing usage of:
  - Modularity
  - Quantification and obliviousness
  - AOP’s contribution
- Due to multiple senses of “modularity”
  - Information hiding (lack of coupling)
  - Encapsulation (lack of scattering)
Questionable Points: Lack of Imagination

- “I find it difficult to imagine ...” (4.2)
- “I cannot see how ... “ (4.3)
- “it may be my lack of imagination” (5.1.7)
Questionable Points: Lack of Imagination

- Most technology nay-sayers wrong
- Some promising directions
  - Tool support (Eclipse AJDT)
  - XPIs
  - Open Modules
  - Static analysis
  - Model-driven architecture
  - Annotations
Conclusions

- Thought-provoking
- Read it!
- Agree: AOP $\approx$ implicit invocation
- Perspective on “paradox”:
  - N-dimensions of modularity
  - Each a scale, not all-or-nothing
  - Compromise!
- Research: how to do better
Time for Discussion

- Coupling
- Reasoning Difficulty
- Heap interference
- Tangling
- Scattering