Aspect Orientation in the Procedural Context of C

ir. Bram Adams

http://users.ugent.be/~badams

Problem Statement

Enterprise software systems ...
= “virtualized” business processes
• massive investments
• acceptable ROI only after long deployment periods

... should be highly and safely evolvable.

BUT:
• first generation maintainers are gone
• legacy hardware and programming languages (Cobol, C, ...)
• huge, ever-growing, undocumented code base

Aspect Orientation (AO)

A new(ish) paradigm tackling the problem of crosscutting concerns, i.e. scattered concerns which are tangled between others. [Fig. 1]

Terminology:
• advice: crosscutting functionality, part of aspect
• join point: interesting event in control flow of base program
• pointcut: defines set of join points by quantification on base program’s properties
• aspect: advice linked to join points by pointcut

Benefits:
• better SoC, irrespective of paradigm (OO, procedural, ...)
• obliviousness: base program unaware of any aspects
  obliviousness: base program unaware of any aspects ➞ unobtrusive reverse engineering of legacy systems

Aspicere, AO for C

Eligible join points: procedure calls and variable accesses

Combines multiple languages/paradigms: [Fig. 2]

• C: - leverage existing concepts and scoping rules A
  - less steep learning curve D
• Prolog: - bindings enable generic advice bodies D
• robust pointcuts C using structure & semantics
• AO: - advice signature B, pointcut C and body D

Weaver transforms base program and aspects into ANSI C code, by chaining relevant advice. [Fig. 3]

Validation

Case Study:
• industrial system with 407 C-modules and 269 Makefiles

Goal:
• generate traces with tracing aspect + check pointer args

Results:
• all modules were woven and relevant traces obtained without altering the original source code
• we could replace some hacks with clean, modular code

Problems:
• slow, complex type inference (large pre-ANSI C chunks)
• slow join point matching ➞ scalability or implementation?
• deployment requires altering existing makefile-hierarchies

Legacy systems are killer apps for Aspect Orientation (AO).