Aspect Orientation for C: Express yourself

Aspicere

Bram ADAMS
Software Engineering Lab, INTEC, UGent

Tom TOURWÉ
CWI
= aspect-language for C (Latin: “to look at”)

History:
- Spin-off from Cobble (Kris De Schutter) [LD05]
- Developed as Master’s thesis (Stijn Van Wonterghem) [V04]
- Further development during PhD research (Bram Adams)

Characteristics:
- Based on Yerna Lindale- and Lillambi-frameworks
- Initially an AspectC-lookalike
- Source code weaving (preprocessor for gcc)

Main challenges:
- Expressive pointcut language
- Adequate weaving process

current work (SPLAT)
**Parameter checking concern**

=coding convention

```c
int do_something (char* in, int* out, double** outptr) {
    if (in == (char*) NULL) {
        /*LOG*/
    }
    if (out == (int*) NULL) {
        /*LOG*/
    }
    if (*outptr != (double*) NULL) {
        /*SPECIAL LOG*/
    }
    ...
    /*DEREFERENCE of ‘in’*/
    /*DEREFERENCE of ‘out’*/
    /*DEREFERENCE of ‘outptr’*/
}
```

BUT: only check if not already done earlier in control flow
AspectC

= subset of AspectJ [CKFS01]

- Consequences for pointcut definitions:
  - No static conditions on bound variables
  - How to capture a random method argument?
  - No explicit, static navigation through code structure
  - Name-based matching
  - Fixed set of primitive pointcuts/joinpoint types

Parameter concern needs:
- Structural pattern-based matching
- Static checks

Lack of expressivity
aspect CheckingAspect {
    pointcut pc1(queue* queue) :
        args(queue) &&
        (execution(* queue_add(..)) || execution(* queue_pop(..)));

    before(queue* queue) : pc1(queue) {
        if(queue == (queue*) NULL) {
            LOG(PARAMETER_ERROR)
        }
    }
}
Aspicere’s design

Needed: pattern matching on structure of base program

Choose logic programming language [GB03]:
- Unification
- Broad range of predicates
- Access joinpoint shadows
- Parameterisable pointcut definitions (reuse!)
- Recursion

- Selection of joinpoints
- General-purpose

AST Analysis (CFG, ...)

Hyper-primitive layer

Primitive layer:
- call, execution, ...
- general-purpose

Aspect-specific layer:
- Parameter checking
- Tracing
- ...

extensible
Taking on parameter concern

ioCheck(Jp,[Pointer,Type]):- /*gather all needed checkpoints*/.

before inout(BaseType) on(Jp):
    ioCheck(Jp,[Pointer,Type]) && pointer(Type,BaseType) {
        /*check pointer-parameters using bound BaseType*/
    }

CONCLUSION:

• AspectC-like pointcut language is not powerful enough
• Prolog allows extensible language design and expressiveness

