Domains of Discourse: Concern Annotations for Program Understanding

IOWA STATE UNIVERSITY
Department of Computer Science

by Curtis Clifton

cclifton@cs.iastate.edu
http://www.cs.iastate.edu/~cclifton

Central Questions

Are annotations for modeling concerns a useful mechanism for:

- communicating intent to later readers?
- automatic, static detection of mismatches between intent and implementation?
- more abstract pointcut specification?

Motivation

Traditionally type annotations have:

- conveyed meaning to readers of code
- allowed static detection of errors
- helped languages' scalability

Recently type annotations have:

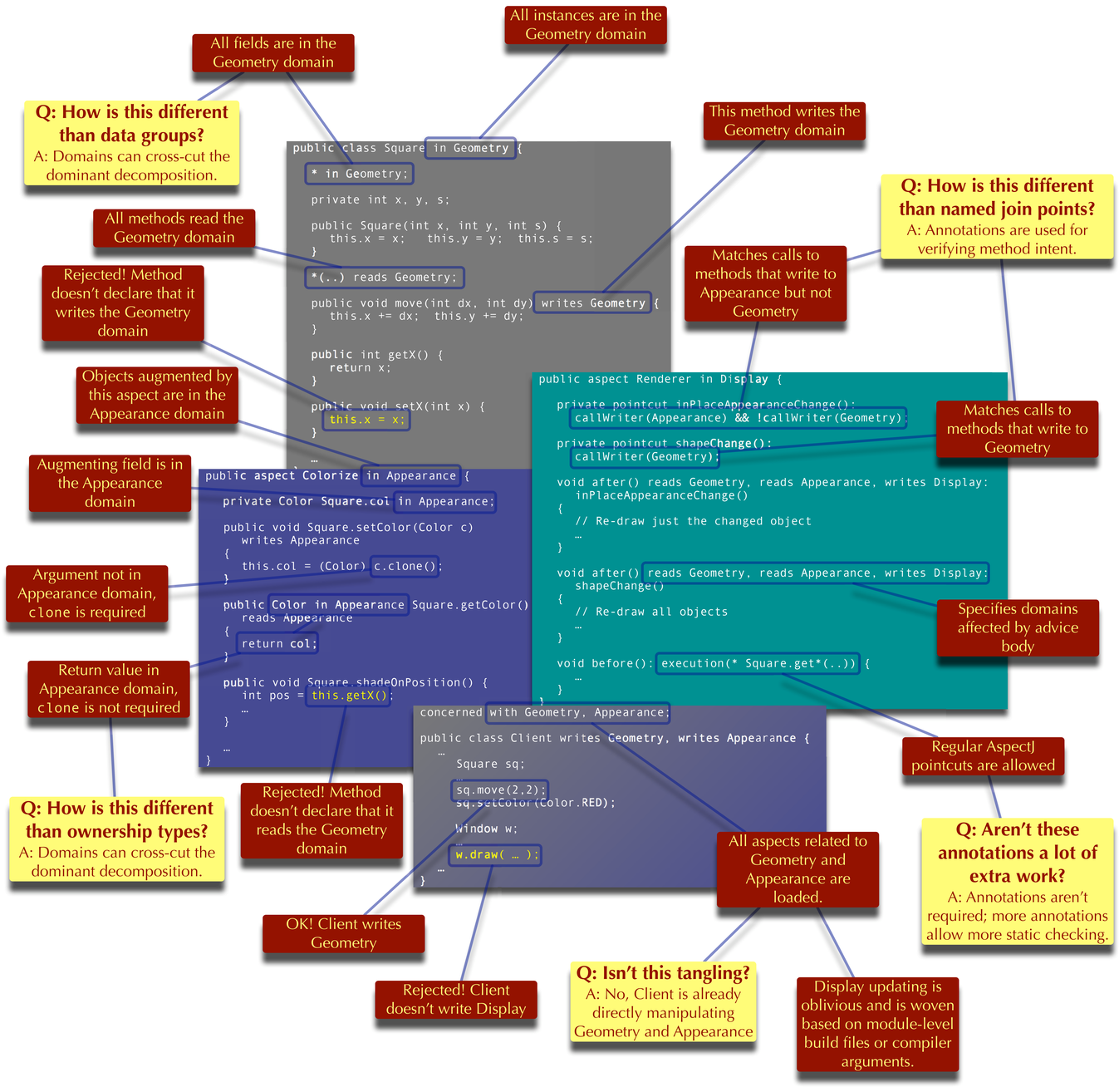
- allowed static detection of race conditions
- helped control aliasing

Objectives

Introduce annotations that:

- make aspect-oriented programs, and their intended separation of concerns, easier to understand
- provide immediate benefit to programmers

Solution Sketch



Made on a Mac.

Supported by NSF grants CCR-0097907 and CCR-0113181.

Key Ideas

- Use named domains of discourse to describe subsets of memory
- Use type system to statically check domain confinement
- Add join points based on domain access

Technical Approach

Language and type system design:

- develop *MAO*, a Modular Aspect-Oriented programming language
- target the Java Virtual Machine, interoperate with Java libraries
- include advice, dynamic join points, and open classes
- use a module interconnect language

Prototype tool implementation:

- extend the Polyglot framework using MultiJava
- use AspectJ for back-end weaving

Formal type system soundness:

- design the *MiniMAO* core calculus
- based on Jagadeesan, *et al.*, 2003

Evaluation:

- perform small case studies exploring a variety of aspect-oriented idioms
- develop a large application using the language and techniques