Convergence...
1. What I learned this summer
   My thoughts on program observation

2. What I was assigned this summer

3. I beg to differ

4. Relevance to PLT

5. Relevance to MzTake
Modes of Computer Science

• CS as engineering: system *construction*
  
  Program design, implementation
  
  Information hiding $\Rightarrow$ implementation independence
Modes of Computer Science

• CS as engineering: system \textit{construction}
  
  Program design, implementation
  
  Information hiding $\Rightarrow$ implementation independence

• CS as experimental science: system \textit{observation}
  
  Debugging, profiling, testing, monitoring, ...
  
  Information hiding gets in the way
Borrowing some Credibility...

People have taken abstraction...and believed that it’s always right to hide what’s going on behind this little curtain. But I don’t think that’s true if you’re looking at the behavior of a system and trying to figure out what’s going on. There you want to be able to see everything.

- Martin Rinard

[Computer scientists] are especially good at dealing with situations where different rules apply in different cases; they are individuals who can rapidly change levels of abstraction, simultaneously seeing things "in the large" and "in the small."

- Donald Knuth
Enforcement

- Pythonistas, Smalltalkers, AOPers, etc.

  Boundaries are for wimps

  Development conventions, analysis tools
Enforcement

• Pythonistas, Smalltalkers, AOPers, etc.
  
  Boundaries are for wimps
  
  Development conventions, analysis tools

• Semanticists
  
  Language support for enforcement of conventions
Enforcement

- Pythonistas, Smalltalkers, AOPers, etc.
  Boundaries are for wimps
  Development conventions, analysis tools
- Semanticists
  Language support for enforcement of conventions
- Who’s right?
  If you have to subvert the language, the language is flawed
  Information hiding needs to be refined
1. What I learned this summer

2. What I was assigned this summer

   Origins of the research project

3. I beg to differ

4. Relevance to PLT

5. Relevance to MzTake
Filman and Friedman 2005

AOP = Quantification + Obliviousness
Filman and Friedman 2005

AOP = Quantification + Obliviousness

• Quantification

Join-point model: universal quantification over program elements
Filman and Friedman 2005

AOP = Quantification + Obliviousness

- Quantification
  
  Join-point model: universal quantification over program elements

- Obliviousness
  
  Modular advice ⇒ no scattered modifications
All-seeing aspect language

"Complete" aspect language for Java

• Quantification
  Should be able to observe *everything*

• Obliviousness
  Should be able to do anything to it
Kinds of Program Elements

- Syntactic
  
  All string literals

  All `for` loops
Kinds of Program Elements

• Syntactic
  
  All string literals
  
  All for loops

• Semantic
  
  All methods of class Foo
  
  All singleton abstract value sets
Kinds of Program Elements

- **Syntactic**
  
  All string literals
  
  All for loops

- **Semantic**
  
  All methods of class `Foo`
  
  All singleton abstract value sets

- **Dynamic**
  
  All assignments to `Foo.x` where new value is 42
  
  Whenever LTL formula $\Phi$ is violated
Kinds of Transformations

• Before/after
• Delete/Replace
• Modify
• Wrap
Source-to-Source Translation

apply

Foo.java

mumble.rules

Foo.java
Applications

- Scriptable debugging, profiling, testing
- Scheduler scripting language
- Runtime verification
- Runtime monitoring, fault tolerance
Interesting design questions

- Mutation

Can/should we formulate this functionally?

Orthogonality vs. sequentiality
Interesting design questions

• Mutation

  Can/should we formulate this functionally?

  Orthogonality vs. sequentiality

• Automation

  Inference of relevant fields for monitored formulae
Interesting design questions

• Mutation
  
  Can/should we formulate this functionally?

  Orthogonality vs. sequentiality

• Automation
  
  Inference of relevant fields for monitored formulae

• Lessons from macros
  
  $\alpha$ -conversion

  Template language
Interesting design questions

• Mutation
  Can/should we formulate this functionally?
  Orthogonality vs. sequentiality

• Automation
  Inference of relevant fields for monitored formulae

• Lessons from macros
  $\alpha$ -conversion
  Template language

• Interactive instrumentation
  Compilation makes this hard/impossible
  Structural changes can’t be done dynamically
1. What I learned this summer
2. What I was assigned this summer
3. I beg to differ
   Where I depart from my mentors’ view
4. Relevance to PLT
5. Relevance to MzTake
Wand 2003

Obliviousness ⇒ pain
Wand 2003

Obliviousness $\Rightarrow$ pain

- Any advice might completely change a module’s behavior
Wand 2003

Obliviousness $\Rightarrow$ pain

- Any advice might completely change a module’s behavior
- Destroys local reasoning
Wand 2003

Obliviousness $\Rightarrow$ pain

- Any advice might completely change a module’s behavior
- Destroys local reasoning
- What happened to information hiding?
Obliviousness $\Rightarrow$ pain

- Any advice might completely change a module’s behavior
- Destroys local reasoning
- What happened to information hiding?
- Proposal
  
  Aspects as modular units of *conjunctive* specification
  
  Whereas AspectJ aspects are *exceptional*
"Open-box" AOP is better suited to system observation
"Open-box" AOP is better suited to system observation

• Problematic for system construction

  All the usual problems of violation of abstraction boundaries
"Open-box" AOP is better suited to system observation

- Problematic for system construction
  
  All the usual problems of violation of abstraction boundaries

- Killer apps are for system observation
  
  How does instrumenting for loops help build systems?
1. What I learned this summer
2. What I was assigned this summer
3. I beg to differ
4. Relevance to PLT
   Why should Schemers care?
5. Relevance to MzTake
Relevance to PLT Scheme

- `(module ...) language`
- White-box testing (e.g. SchemeUnit)
- Development environments
  - Programs can’t observe their host IDE
    - Aspect scope
- `struct` inspectors
1. What I learned this summer
2. What I was assigned this summer
3. I beg to differ
4. Relevance to PLT
5. Relevance to MzTake

Comparison, questions
Comparison to MzTake

- Trace points, join points
- Promoting debugging sessions to repeatable tests
- Compilation model?