Visual(ized) Source-code Querying

Sattose 2009
Background

- Logic-based Program Reasoning
  - Declarative Code Queries

- Intensional Software Views
  - Software Evolution Support

- Tool Integration
  - Ease of use
  - General-purpose
  - Scalability
Queries and Visualizations

- **Declarative Program Queries**
  - Search program parts adhering to well-defined conditions
    - Bad smells and potential bug patterns
    - Design patterns
    - Coding conventions
    - Idioms

- **Structural Software Visualization**
  - Asses an entire system through a concise overview, exposing one or several properties
    - Modular coupling
    - System complexity
    - Metrics
Example: Constructors that call overridden instance methods that reference instance fields defined in the subclass may reference uninitialized fields (in Java)

```
?constructor unsafeConstructorAccessTo: ?var throughCallTo: ?method if
  ?constructor isConstructorDeclaration,
  ?class definesConstructor: ?constructor,
  ?constructor calls: ?selfmethod,
  ?subclass hasMethod: ?selfmethod,
  or(?selfmethod equals: ?method, ?selfmethod callsTransitiveOnSelf: ?method),
  ?class declaresType: ?classType,
  ?subclass inClassHierarchyOfType: ?classType,
  ?method reads: ?var,
  ?subclass definesVariable: ?var
```
Visual Queries

not(equals(?class,[Object]), not(equals(?sub1, ?sub2)))
Composite DP

![Composite DP Diagram]

The Intensional Views Environment

Monday 11 May 2009
To conclude

- Visualize *what* you query
  - Entities (objects)
  - Relations (predicates)
- Assess query results in a *global* context
- Narrowing focus *metaphor*
- Specify a visual query
- Example-based *source-code queries*