Developer Refinement of Runtime Architectural Structure

Marwan Abi-Antoun   Talia Selitsky   Thomas D. LaToza
Wayne State University   Carnegie Mellon University
SCHOLIA depicts runtime object structure

Compile-time approximation of **runtime** object structure: Ownership Object Graph (OOG)

Represents architecture as hierarchical **encapsulation / logical containment** and **points-to** relationships between objects

**Interactive OOG viewer:**
- zoom / pan / scroll diagram
- collapse / expand object
- trace to code
- search by type or identifier
Pilot study of a developer in the field

Research question
What interactive features are required to help developers edit a reverse engineered OOG to better match their mental model of the runtime architecture?

Previous study [PASTE'08]
Selected 30-KLOC module of a 250-KLOC industrial system
Experimenter reverse-engineered OOG talked to developer, added annotations, ran extraction tool

Current study method
2-hour interview with developer of module
Use interactive viewer tool; trace to code
Developer's desired edits to diagram

- **Move** object between domains
- **Abstract** low-level object
  - Push it underneath more architectural object
  - Hide it somehow (must maintain soundness!)
- **Group** an object into another object without enforcing encapsulation (**logical containment**)
- Collapse **related** instances of subtypes
- Edit object **labels**
- **Split** an object into separate objects
  - Show objects for supertype, subtype
  - Not supported by runtime view
Other features a developer might need

Navigate from type in IDE to objects in OOG

Task-specific view rather than complete view
- Hide portions of system that are not interesting
- Challenging to do while maintaining soundness

Express constraints on allowed relationships
- Show error when structural constraints violated

Supported by approach:
- Use additional layer of annotations (domain links)
- Extract-abstract-present OOG in ADL
- Use predicates to enforce constraints