**Challenge:** Automatically pinpoint the domain model classes: The high-level business concepts

**Problem**
- Design documents typically not up to date with code.
- Business concepts not readily available in the code.
- Requires reasoning about large complex code base.

**Observations / Heuristics**
(01) If an intermediate result is a domain model object, the code more likely refers to it explicitly:
- Assign to local variable, field, etc.
- May aid debugging
- May be seen as more stable over time

(02) A domain model class is likely used together with other domain model classes
- To navigate domain relations
- To provide business functions

**Reoom Approach: Light-weight Static Analysis**

1. **Annotate call graph**
   - Classes: A, B, C
   - Call graph: A → B, A → C

2. **Filter methods**
   - Condensed call graph: A, B, C
   - Ranked candidate domain model classes: B, A, C

3. **Rank classes**
   - By how often they are referenced explicitly

**Implementation**
- On top of static inter-procedural Java analysis framework MoDisco
- Call graph: Explicit method and constructor calls in analyzed public methods and constructors
- Over-approximates virtual calls
- Not captured: Calls via reflection, bytecode, or native code

**Research Questions (RQ)**
- How do Reoom and Womble compare in runtime performance (RQ1) and precision and recall (RQ2)?
- What is the benefit of step (2), which requires relatively expensive inter-procedural analysis (RQ3)?

**Subjects**
- jMusic: "These [five] classes form the backbone of the jMusic data structure"
- pdf-sam: Identified domain classes with our own domain knowledge
- pizza_wo: Plain Java version of well documented pizza shop tutorial
- SweetHome3D: "This UML diagram should help you understand which classes are available [..]"

**Reoom vs. closest competitor—Womble*: Higher precision (p) and recall (r) values are better; SH = SweetHome3D 1.5; c = classes and interfaces; d = domain model classes in c; t = runtime (Womble seeded: sum of d runs, Reoom Light: sum of three runs); n/U = results for classes identified by each or any seeded Womble run; Ø = average precision and recall of Womble’s seeded runs. Experimental setup: 16 GB RAM 2.6 GHz Core i7 MacBook Pro running OS X 10.10.2.