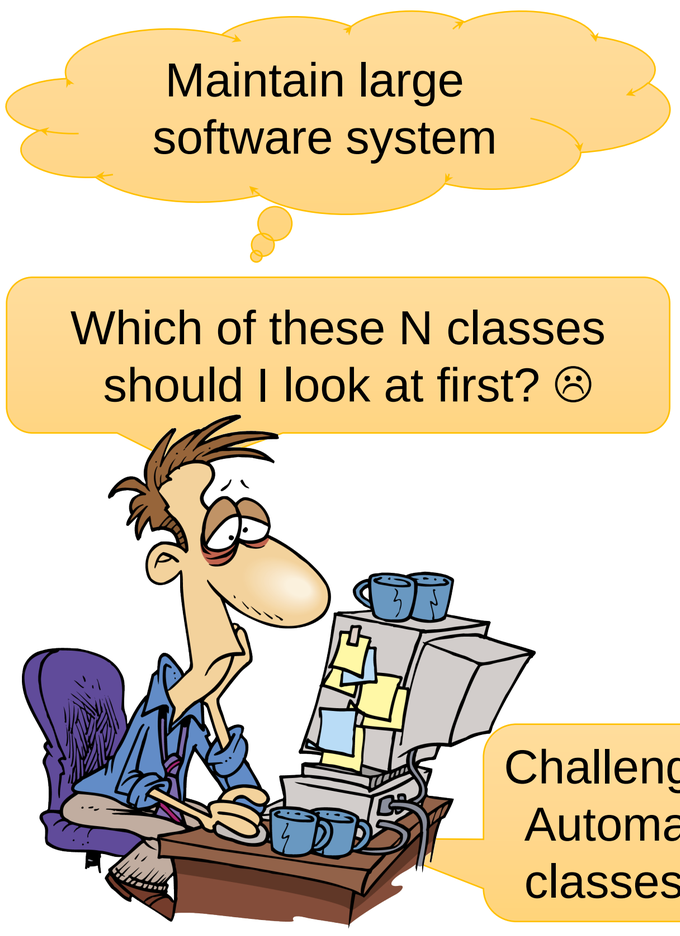
Reverse Engineering Object-Oriented Applications Into High-Level Domain Models With Reoom

Tuan Anh Nguyen, Christoph Csallner
tanguyen@mavs.uta.edu, csallner@uta.edu
Computer Science and Engineering Department
University of Texas at Arlington (UTA), USA



Problem

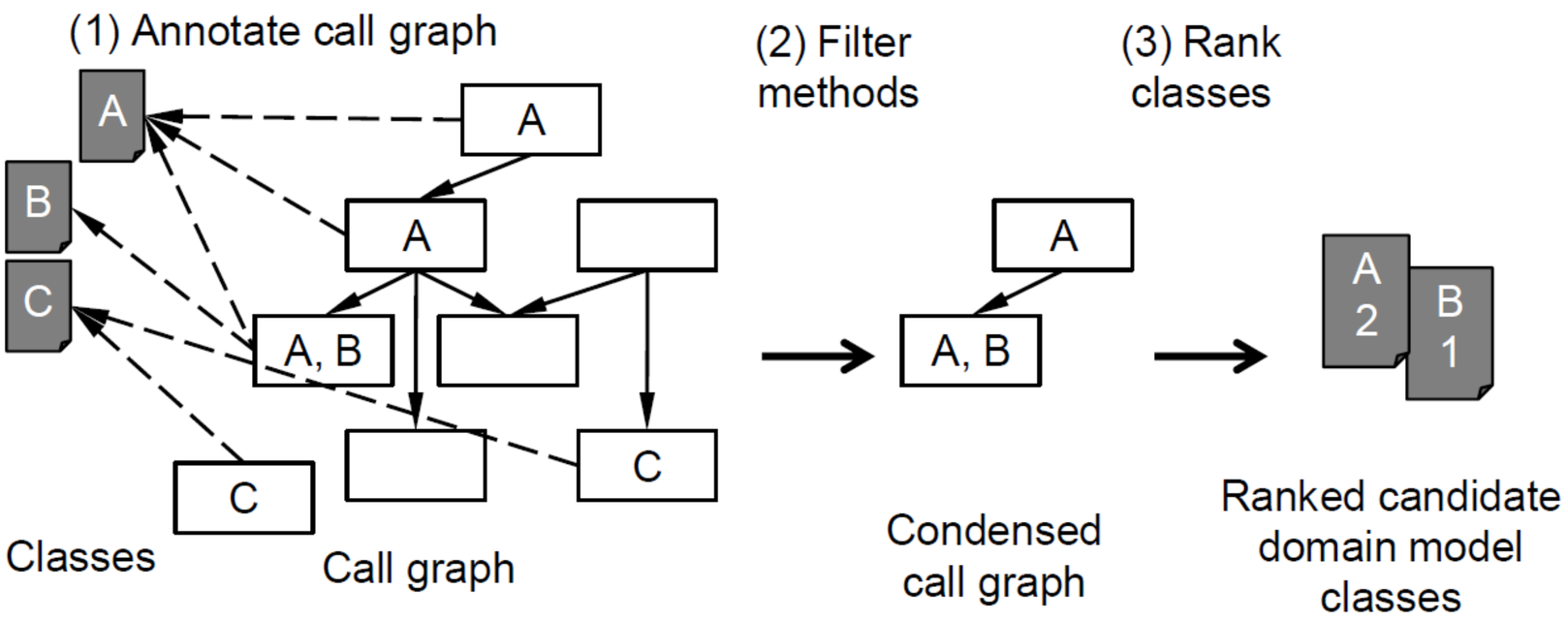
- Design documents typically not up to date with code.
- Business concepts not readily available in the code.
- Requires reasoning about large complex code base.

Observations / Heuristics

(O1) 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

(O2) 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) Check O1: Annotate each method *m* with classes *m*'s code refers to explicitly. Example: 8 methods, 3 classes
(2) Check O2: Remove *m* from call graph if *m* does not appear in a call chain that explicitly refers to ≥2 classes
(3) Rank classes that annotate remaining methods, 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 Light: Reoom without step (2)

Subject				Womble							Reoom Light							Reoom		
					∩		∪		∅			Locals		Params		Return				
	kLOC	c	d	t[s]	p	r	p	r	p	r	t[s]	p	r	p	r	p	r	t[s]	p	r
jMusic 1.6.2	23.2	272	5	timeout							29	7	100	16	100	25	100	397	19	100
pdf-sam 2.2.1	8.8	156	9	270	100	11	16	100	45	14	15	13	67	23	56	17	56	158	38	89
pizza_wo	1.1	18	4	1	50	25	57	100	52	31	6	36	100	100	75	40	50	30	100	100
SH	28.3	161	27	timeout							28	20	85	28	85	30	74	662	37	56

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) ; ∩/∪ = 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.

(*) D. Jackson and A. Waingold: "Lightweight extraction of object models from bytecode," in *Proc. 21st ACM/IEEE International Conference on Software Engineering (ICSE)*. ACM, May 1999, pp. 194—202.