Benchmarking Reverse Engineering Tools and Using Tool Output for Further Analysis

David Cutting and Joost Noppen

University of East Anglia

david.cutting@uea.ac.uk, j.noppen@uea.ac.uk

Presentation Outline

- Introduction to Traceability Forensics Project
- Benchmarking of Reverse Engineering
- Working Further with Reverse Engineering Output for Analysis and Comparison
- Next Steps

Traceability Forensics Project

- We aim to **recover traceability links**

- Using **partial or missing documentation**

- Along with other information sources:
  - – Source Code
  - – Semantic Analysis

Reverse Engineering

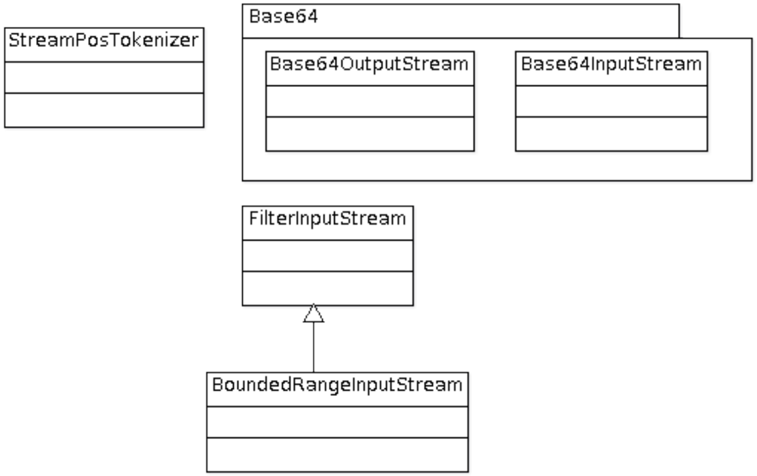
- One of the main sources of information about software is the software itself

- Reverse engineering offers a powerful tool for program comprehension

- There are a **lot** of reverse engineering tools but…

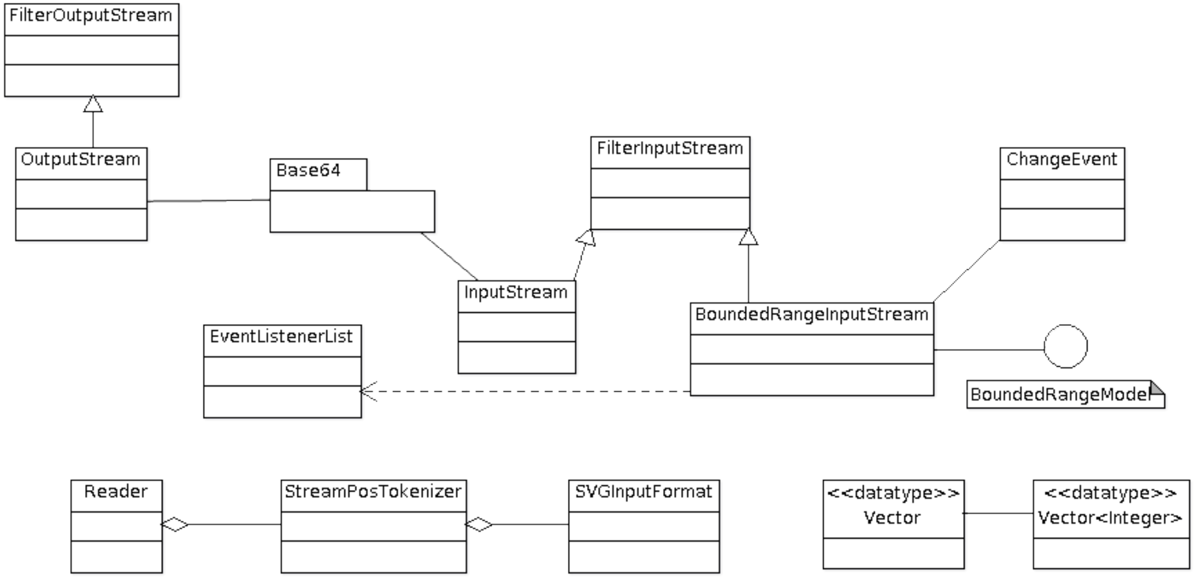
Reverse Engineering Tools

- Although there are many tools they
  - Vary in output (which is right, which is wrong?)
  - Have no standard means of comparison
- This is org.jhotdraw.io from Rational Rhapsody:

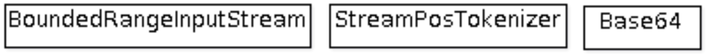


Reverse Engineering Tools

- org.jhotdraw.io from Astah Professional:



- org.jhotdraw.io from ArgoUML:



The Benchmark

- To compare and rank different tools we created a benchmark (the Reverse Engineering to Design Benchmark: RED-BM)
- 16 target artifacts
  - Varying from 100 to 40,000 lines of code
  - From 7 to 450 classes
  - Range of architecture styles and complexity
  - “Gold standard” for each in terms of contained classes and sampled relationships

The Benchmark

- Existing designs where available
- Reverse engineering output from other tools for comparison
- Initial measures for class detection, packages, and relationships:

$$Cl(s,r)=\frac{C(r)}{C(s)},Sub(s,r)=\frac{S(r)}{S(s)},\mathrm{Re}l(s,r)=\frac{R(r)}{R(s)}$$

For artifact *x*: *C(x)* is the ratio of correct classes, *S(x)* ratio of correct packages and *Rel(x)* ratio of correct relationships in system *s* for result *r*

The Benchmark

- Individual measures fed into weighted Compound Measure (CM) as function *P*:

$$P(s,r)=\frac{w_{Cl}Cl(s,r)+w_{Sub}Sub(s,r)+w_{\mathrm{Re}l}\,\mathrm{Re}l(s,r)}{w_{Cl}+w_{Sub}+w_{\mathrm{Re}l}}$$

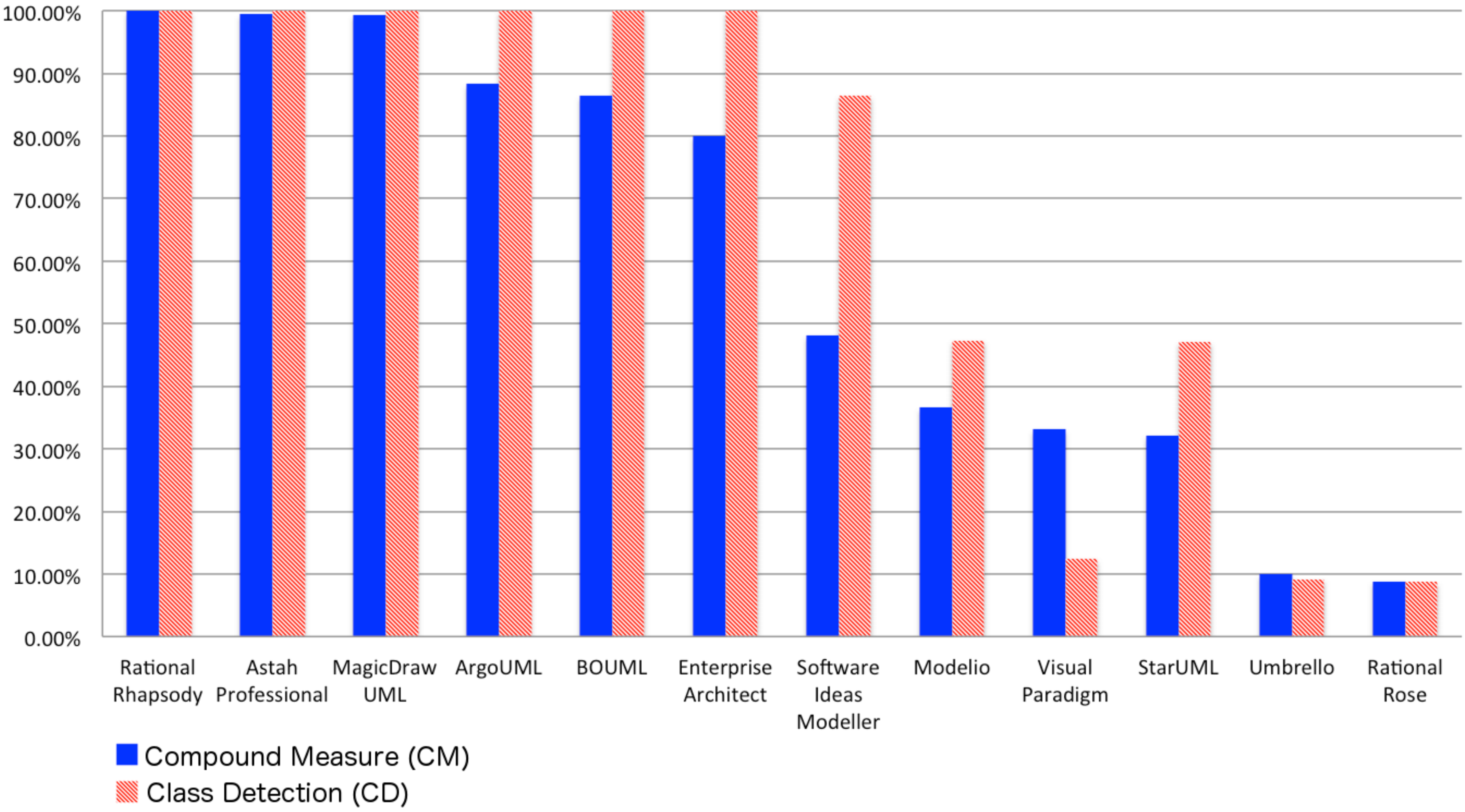
- Extensibility – existing and new measures can be combined into new or redefined (refocused) compound measure *C*:

$$C(s,r)=\frac{\sum_{i=1}^{n}w_iM_i(s,r)}{\sum_{i=1}^{n}w_i}$$

Benchmark Analysis

- We ran a 12 industry reverse engineering tools against the 16 target artifacts
- We then compared output against our "Gold Standard"
  - Rather than doing this manually we used the XMI output from tools (more on this later)
- What we found was quite surprising…

Benchmark Results



Key Findings

- Wide variance in performance between tools (8.8% to 100%)
- RED-BM is effective at differentiating tool performance
- You don’t always get what you pay for!

Working Further With Reverse Engineering Output

- Benchmarking shows clear differences but we want to be able to use output from reverse engineering for further use
  - Aggregation of output (bringing together multiple imperfect outputs)
  - Combination with other sources of information

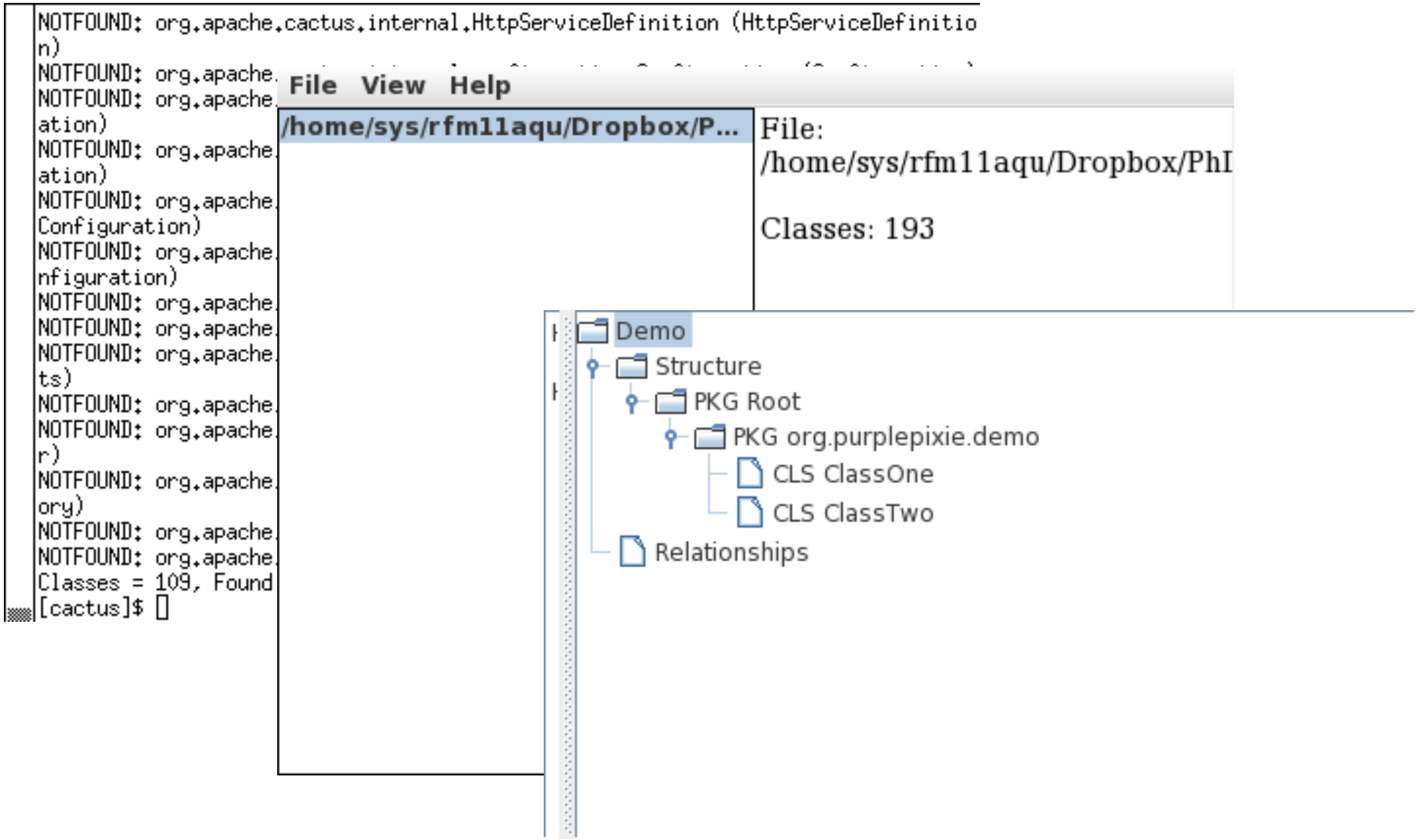
XML Metadata Interchange (XMI)

- XMI is an Object Management Group (OMG) Meta-Object Facility (MOF) for exchange of Unified Modeling Language (UML)
  - – So XMI = OMG MOF UML (OMG is right!)
- This is a standard but one offering extensibility on many levels
- So effective interchange between tools is pretty much non-existent

Working with XMI

- To create the benchmark we wanted to be able to analyse XMI rather than counting classes by hand
- This entailed the creation of a generic XMI class finder
- In turn this work led to a generic XMI parser to load XMI models into a standard format in memory

Working with XMI



```
NOTFOUND: org.apache.cactus.internal.HttpServiceDefinition (HttpServiceDefinitio
n)
NOTFOUND: org.apache.
NOTFOUND: org.apache.
ation)
NOTFOUND: org.apache.
ation)
NOTFOUND: org.apache.
Configuration)
NOTFOUND: org.apache.
nfiguration)
NOTFOUND: org.apache.
NOTFOUND: org.apache.
NOTFOUND: org.apache.
ts)
NOTFOUND: org.apache.
NOTFOUND: org.apache.
r)
NOTFOUND: org.apache.
ory)
NOTFOUND: org.apache.
NOTFOUND: org.apache.
Classes = 109, Found
[cactus]$
```

/home/sys/rfm11aqu/Dropbox/P...

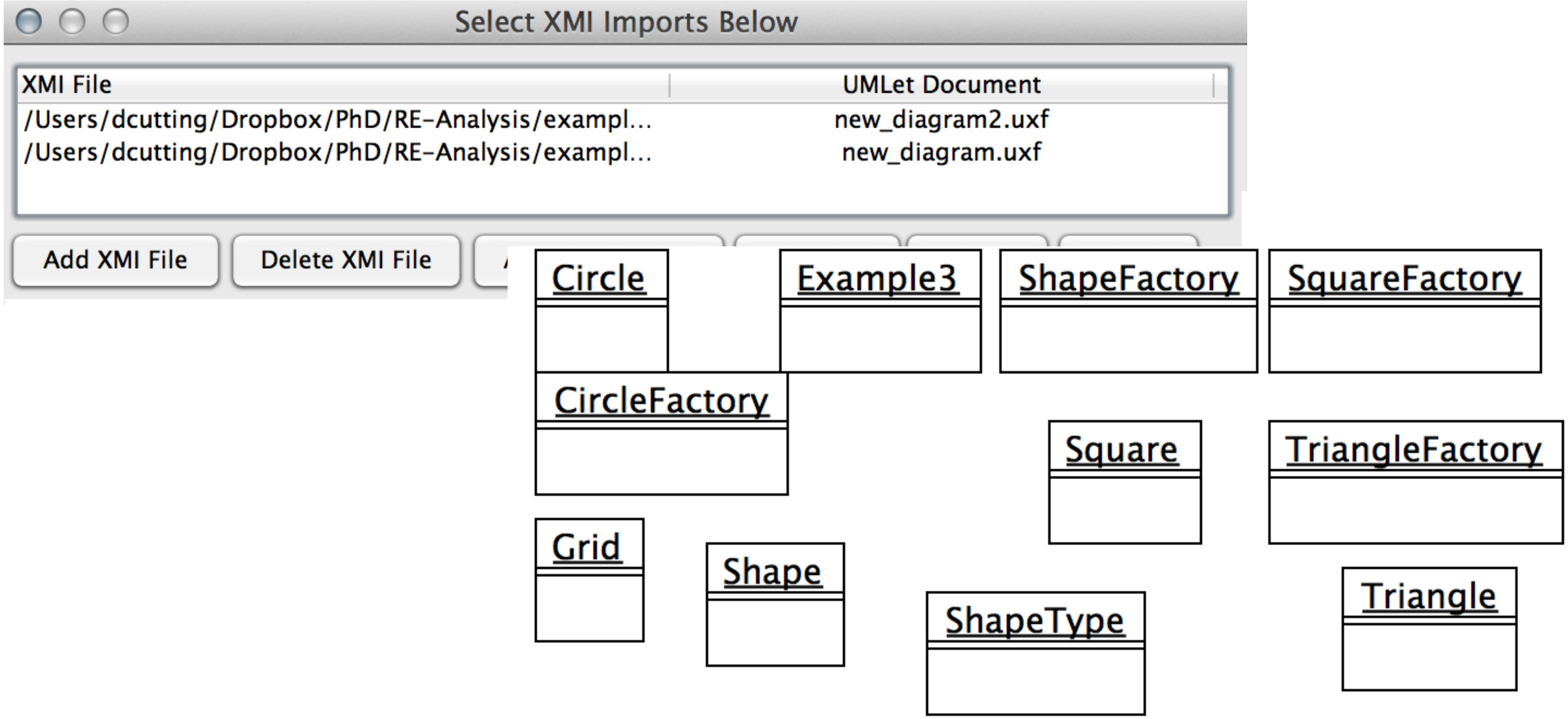
File:
/home/sys/rfm11aqu/Dropbox/PhI

Classes: 193

```
Demo
  Structure
    PKG Root
      PKG org.purplepixie.demo
        CLS ClassOne
        CLS ClassTwo
  Relationships
```

Reconstruction from XMI

- Using UMLet within Eclipse



Select XMI Imports Below

XMI File	UMLet Document
/Users/dcutting/Dropbox/PhD/RE-Analysis/exampl...	new_diagram2.uxf
/Users/dcutting/Dropbox/PhD/RE-Analysis/exampl...	new_diagram.uxf

Next Steps

- Further refine XMI parser/analyser
- Continue on UMLet Eclipse integration
- More sources of information:
  - Source Code Repository mining
  - Documentation analysis
  - Feeding into a Reasoning Component
- Base case software library for example including architectural styles

Thank You

Any questions?

Feel free to email: david.cutting@uea.ac.uk