Does your mother know you’re here?

Understanding software artifact provenance

Mike Godfrey
Software Architecture Group, UWaterloo, Canada
[visiting CWI / SWAT, Amsterdam until July 2012]
Software artifact provenance

An emerging problem area
“Provenance”

A set of documentary evidence pertaining to the origin, history, or ownership of an artifact.

[From “provenir”, French for “to come from”]
Software artifact provenance

Origin analysis + copyright violation

$V_{\text{new}}$

$V_{\text{old}}$

OSS

[Godfrey and Zou: TSE-05]

[Many authors]
Software artifact provenance

Developer email topic mining

[http://softwareprocess.es/whats-in-a-name]

[Hindle, Ernst, Godfrey, Mylopoulos: MSR-11, EMSE to appear]
Software artifact provenance

Mylyn and inferring context

[Kersten and Murphy, AOSD-05]
Summary: Software artifact provenance

• Given a chunk of code, test suite, developer email topic, maintenance task, ... we want to investigate its origin, evolution, and the supporting evidence:
  – Who are you, really?
  – Where did you come from?
  – Does your mother know you’re here?

• Some problems:
  – Ground truth?
  – Artifact linkage?
  – Running matching algorithms on big data? **
Software Bertillonage

A metaphor for attacking the provenance problem
Who are you?

Alphonse Bertillon (1853-1914)
The nose, as it cannot be disguised, is extremely important in identification. The types above, taking them from the left, show a low, narrow nose, a hooked nose, a straight nose, a snub nose, and a high, wide nose.
Forensic Bertillonage

1. Height
2. Stretch: Length of body from left shoulder to right middle finger when arm is raised
3. Bust: Length of torso from head to seat, taken when seated
4. Length of head: Crown to forehead
5. Width of head: Temple to temple
6. Length of right ear
7. Length of left foot
8. Length of left middle finger
9. Length of left cubit: Elbow to tip of middle finger
10. Width of cheeks
Software Bertillonage

• It’s not fingerprinting or DNA analysis!
  – There may be not enough info / too much noise to make positive ID
  – You may be looking for a cousin or ancestor
  – You may be synthesizing something that doesn’t exist elsewhere

• A good software Bertillonage metric should:
  – be computationally inexpensive
  – be applicable to the desired level of granularity / prog. language
  – catch most of the bad guys (recall)
  – significantly reduce the search space (precision)
Software Bertillonage

Matching library usage fingerprints

Uniqueness of `ldd` signatures of `/usr/bin` executables in Ubuntu 9.04

[Hindle, unpublished]
Software Bertillonage

Matching anchored signatures

Q: Which version of library httpclient.jar is included in this Java application?

Our approach:

1. Build master repos of class / method sig hashes from Maven2

2. Compare sig hashes of target app against master repos

[Davis, German, Godfrey, Hindle: MSR-11 and EMSE to appear]
Software Bertillonage

Matching anchored signatures

Q: Which version of library `httpclient.jar` is included in this Java application?

Our approach:

- Consider only class / method signatures
  - May not have source, compiler options may differ, ...
- Build master repos of signature hashes from Maven2
  - Which has gaps, duplication, errors,
- Compare sig. hashes of target application against master repos
  - There will be false positives when API does not evolve
  - ... so the effectiveness of narrowing search space depends on how much APIs evolve

[Davis, German, Godfrey, Hindle: MSR-11 and EMSE to appear]
Testing the extractor, sampling the data

• Randomly picked 1000 binary jars (from the 140K) for which there was also a source jar in Maven2
  – # of classes per binary archive: median: 5, max: 2138

• Binary-to-binary matching (bin2bin):
  – Each binary archive matched itself 😊
  – # of exact matches in Maven (due to duplication or unchanging API)
    • median: 5, max: 487

• Binary-to-source matching (bin2src):
  – Correct match was among top matches (median:4, max: 158): 966 times
  – Something else was a better match (test classes): 30 times
  – No matches suggested (compiler/extractor issues): 4 times
Industrial case study

Target system: An industrial e-commerce app containing 84 jars

Q: How useful is the signature similarity index in finding the original **binary** archive for a given binary archive?

Q: How useful is the signature similarity index at finding the original **source** archive for a given binary archive?
Summary

Who are you?

Software artifact provenance is a growing & important problem

Software Bertillonage

Cheap techniques applied widely, then expensive techniques applied narrowly
Does your mother know you’re here?

Understanding software artifact provenance

Mike Godfrey
Software Architecture Group, UWaterloo, Canada
[visiting CWI / SWAT, Amsterdam until July 2012]