Squinting at the data

Investigating software entity provenance using KISS techniques

Mike Godfrey

Software Architecture Group (SWAG)

UWaterloo [visiting CWI until July 2012]

Joint work with

- Julius Davies (UVictoria, now UBC)
- Daniel German (UVictoria)
- Abram Hindle (UWaterloo, UC-Davis, now UAlberta)
- Neil Ernst (UToronto, now UBC)
- Wei Wang (UWaterloo)

Cloning in Linux SCSI drivers

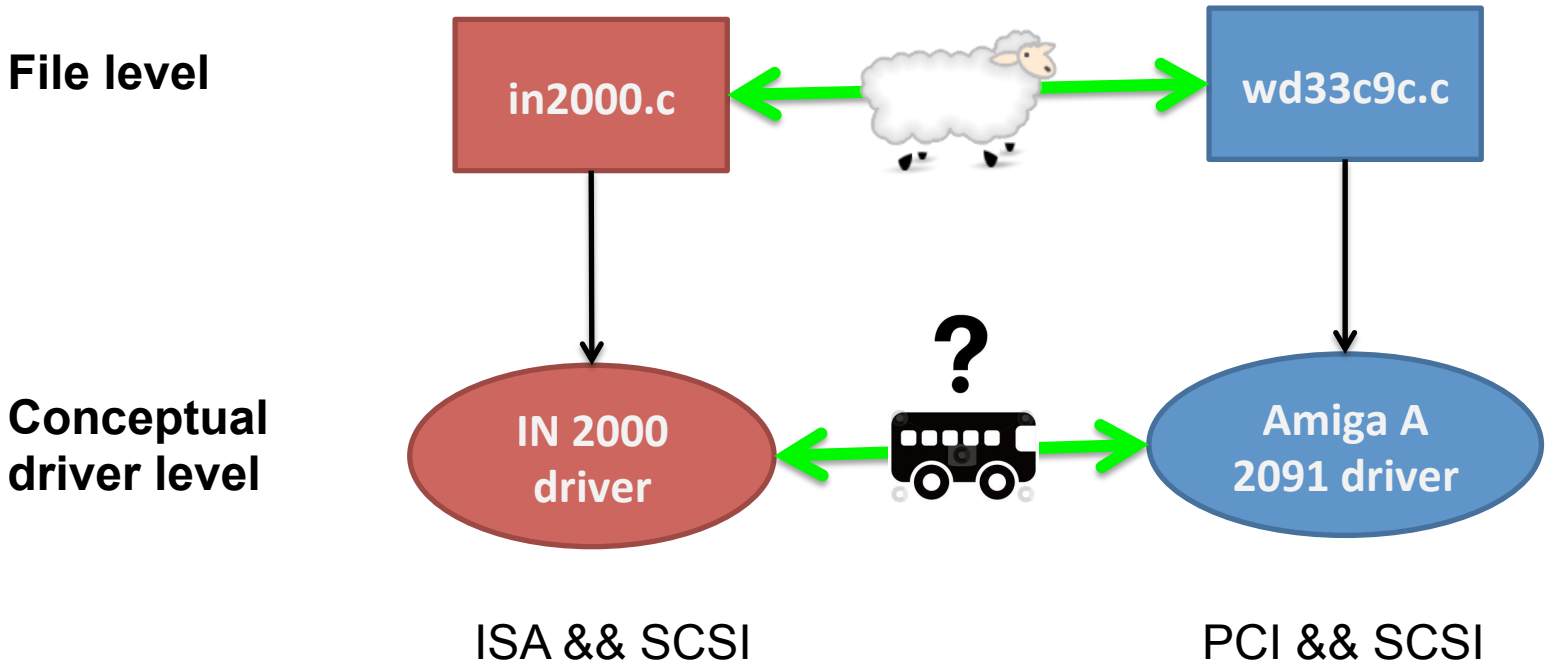
- Linux SCSI driver subsystem:
  - A large set of components that do roughly the same thing
  - Cloning is known to occur
  - 16 years of history, slowing down now
  - 549 files, 96 “conceptual” drivers, 319K SLOC
  - 75% of conceptual drivers consist of one or two files

Q: Does the presence of cloning tell us anything about the higher level design?

[Wang, Godfrey SCAM-11]

Linux SCSI driver cloning

Q: Does cloning predict compatible bus type dependencies?



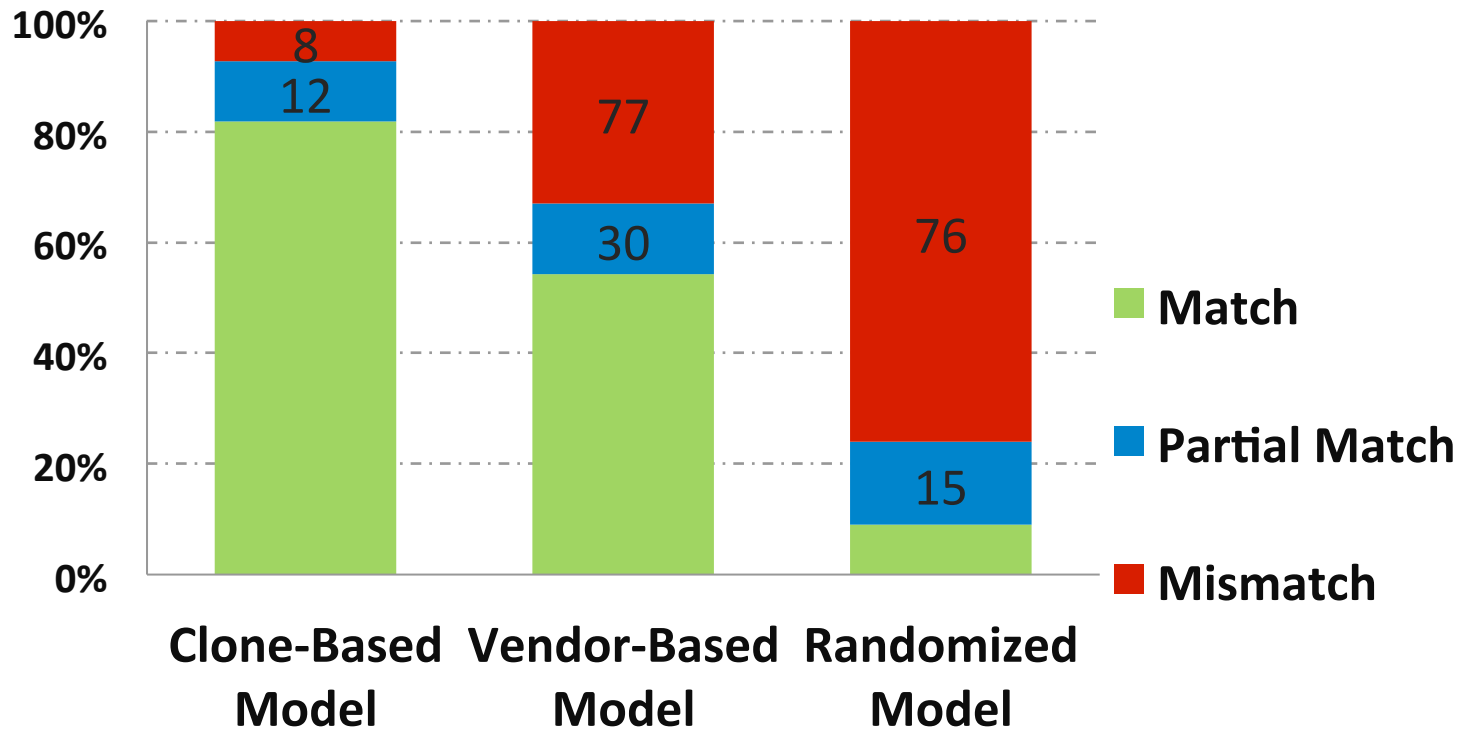
Linux SCSI driver cloning

Matching bus type dependencies:

1. Extract dependency info from config files
2. Convert each logical expression into DNF
3. Run matcher

	(ISA && SCSI) || (PCI && SCSI)	ISA && SCSI && PCI	SCSI && X86_32
ISA && SCSI	Match	Partial-match	Mismatch

Predictive power of cloning



Clone analysis beats domain knowledge!

Software clone detection

- Lots of progress in clone detection / analysis over the last 15 years!
  - Many, many techniques
  - Lots of empirical studies
- No longer just "search and destroy", instead we ask:
  - *Why are clones born? How do they evolve? When do they die? etc.*
- … but where do we go from here?

Software entity provenance

- For a given function, class, file, library, binary, bug report, feature, test suite, … we want to investigate its origin, evolution, and the supporting evidence
  - *Who are you, really?*
  - *Where did you come from?*
  - *Are there any more like you at home?*
  - *Does your mother know you're here?*

Example provenance problems

- *How was feature XXX discussed in the mailing list?*
  - *Where was it implemented in the codebase?*
  - *How much change has it undergone?*

- *How much cloning is there in my system? Why?*
  - *How should I manage the duplication over time?*
  - *Does the cloning imply high-level design similarity?*
  - *Does the cloned code violate the GPL?*
  - *Does the latest release contain at least 25% "new code"?*

Example provenance problems

- *Which version of library* `httpclient.jar` *is included in this Java application?*

- *Has anyone worked on a similar problem before?*
  - *Is this bug report a duplicate?*
  - *How "similar" is this much smaller test suite to the original?*
  - *What APIs might be useful for this maintenance task? [Mylyn]*

… and what is the evidence?

Investigating software entity provenance

Two big tasks:

1. Scoping and identifying the entity
   - *What's a feature? How big is a clone? What's a maintenance task?*
   - *What does "same" or "similar" mean?*

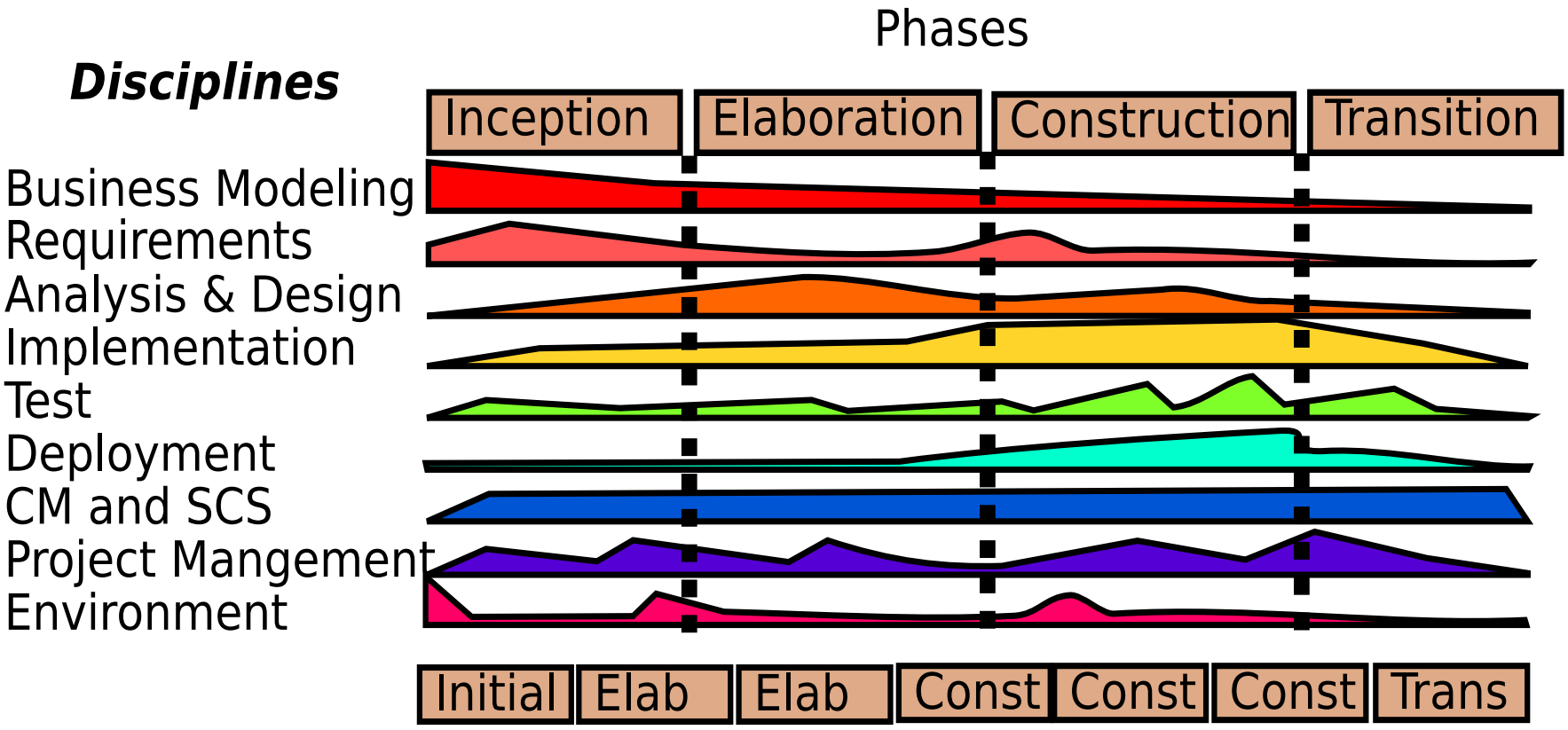
2. Extracting and analyzing the evidence
   - Many kinds of evidence, analyses
   - Ground truth? Master repository?
   - Synthesis and analysis techniques must scale!

Provenance: Mining software repositories (MSR)

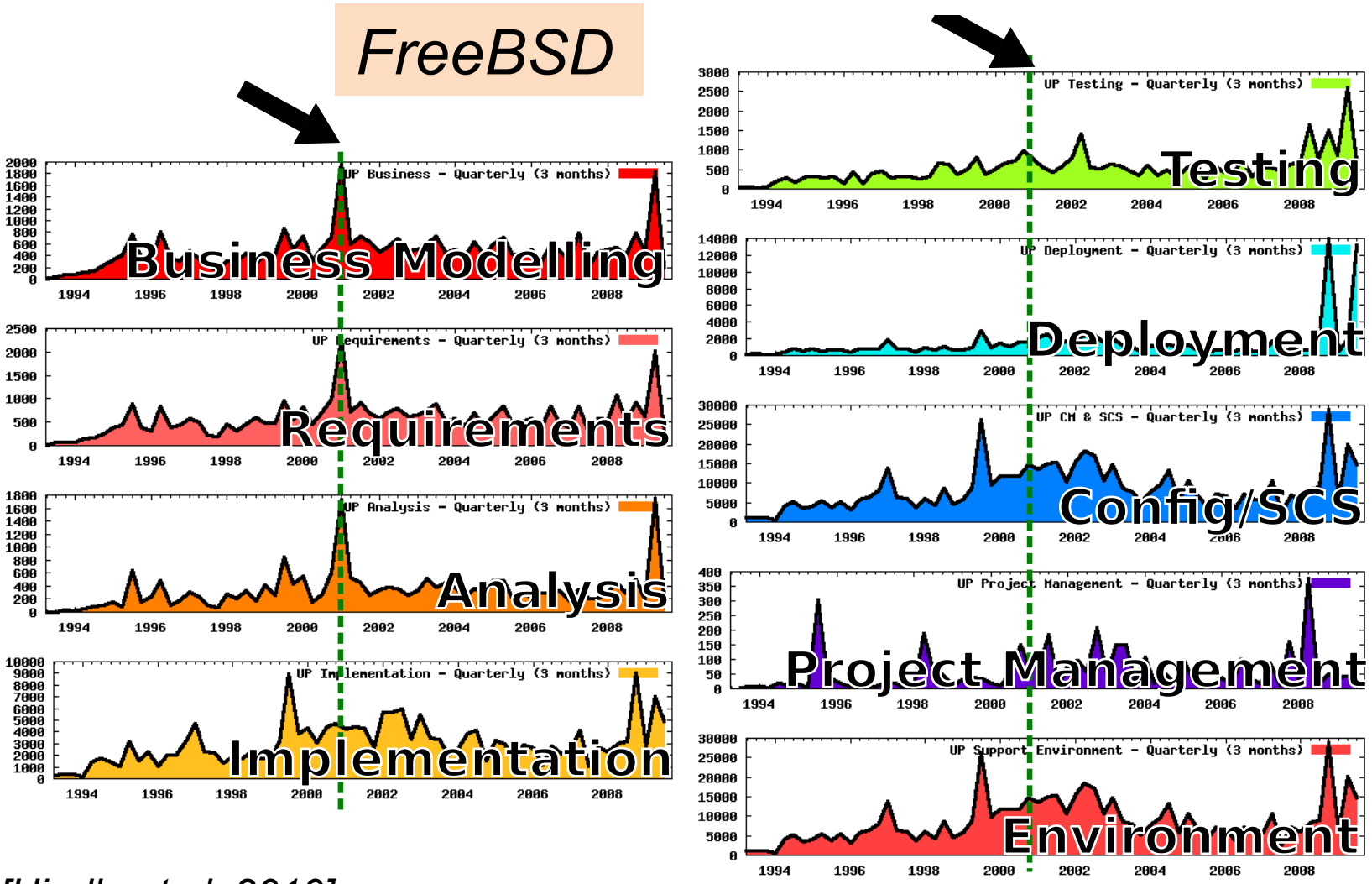
- Why?
  - Lots of artifact kinds (source code, binaries, bug reports, test suites, mailing list, documentation, requirements specs, …)
  - Often they are not well linked
  - *Can we analyze different artifact kinds within a unified context to answer questions about development?*

- Many techniques
  - Source fact extraction, meta-data extraction, clone detection, grep, …
  - AI, LSI, LDA, data mining, …
  - Ad hoc specializations + combinations

MSR: Software process extraction

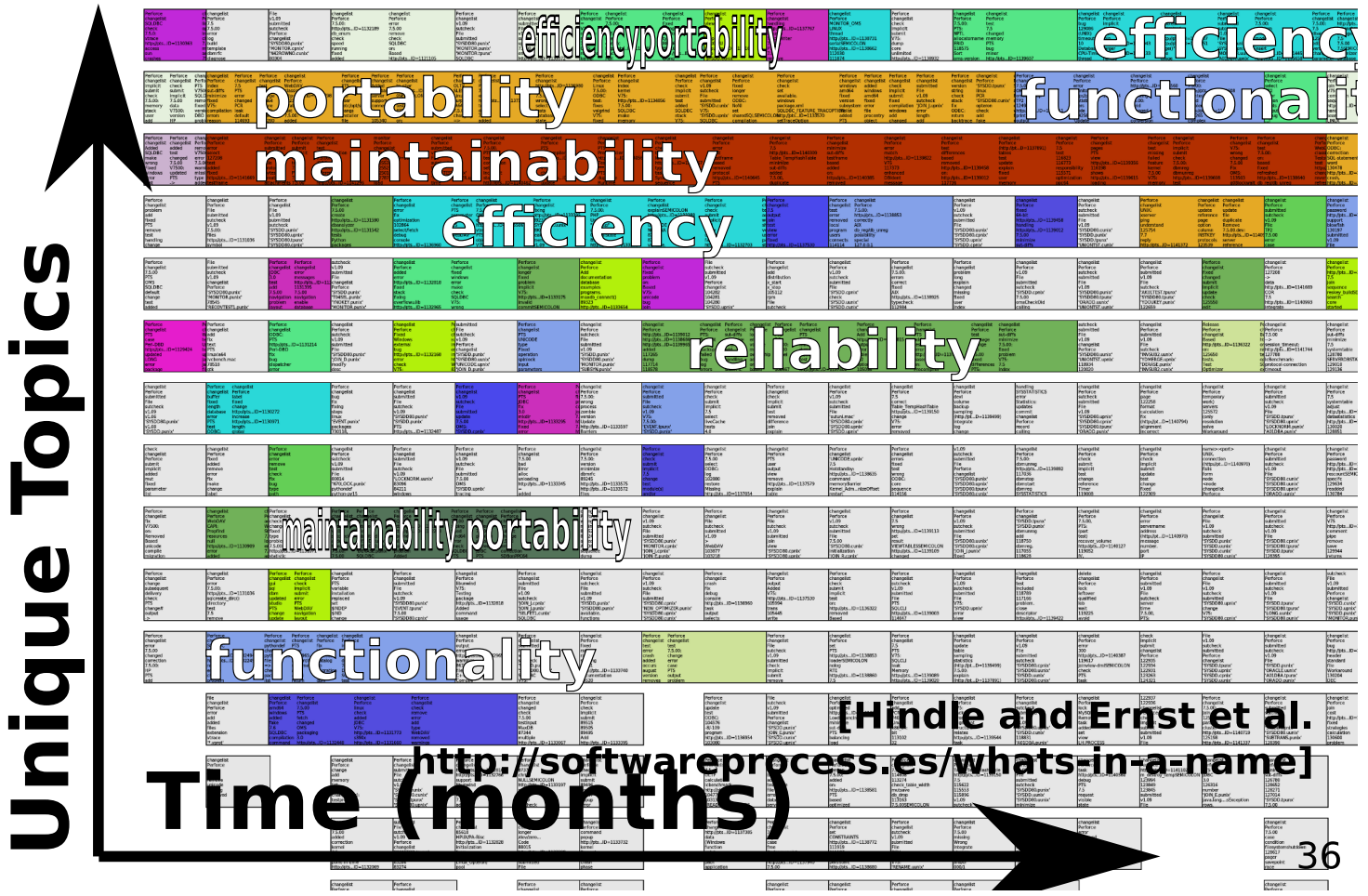


MSR: Software process extraction



[Hindle et al. 2010]

MSR: Developer email topic mining



[Hindle, Ernst, Godfrey, Mylopolous MSR-11]

Software entity provenance: The challenge

- Given recent advances in the field of Mining Software Repositories (MSR) …
  - … can we develop techniques that take advantage of a myriad of inter-related artifact kinds to establish the provenance of a given software entity?
  - … and can we minimize the amount of heavy analysis that we need to do?

KISS

Keep

It

Simple

Stupid*

*Coined by Kelly Johnson, Lead engineer at Lockheed Skunk Works
[My dad worked under him briefly in the 1960s.]

Who are you?

Alphonse Bertillon(1853-1914)

1. Taille. — 2. Envergure. — 3. Buste. —
4. Longueur de la tête. — 5. Largeur de la tête. — 6. Oreille droite. —
7. Pied gauche. — 8. Médius gauche. — 9. Coudée gauche.

Forensic Bertillonage metrics

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

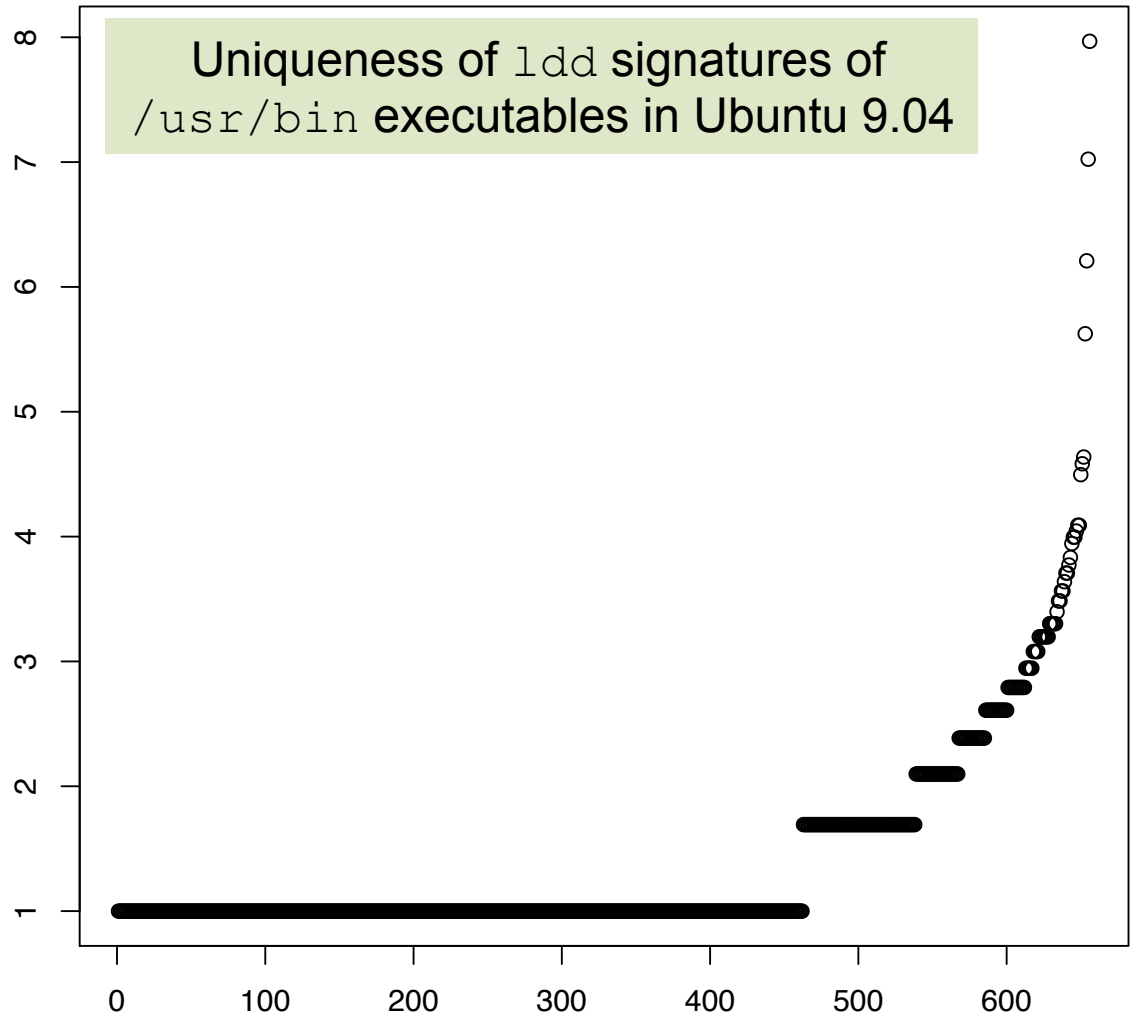
- It's not fingerprinting or DNA analysis!
  - There may be not enough info / too much noise to make positive ID
  - You may be looking for a cousin or ancestor

- 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 meta-techniques

1. Count based	size, LOC, fan-in/out, McCabe
2. Set based	contained string literals, method names
3. Relationship based	libraries included/used, calls/called-by, defines/uses, extends/implements, throws
4. Sequence based	method invocation chains, token-based clone detection
5. Graph based	AST and PDG clone detection

KISS: Matching library usage fingerprints



[Hindle, unpublished]

KISS: Matching anchored signatures

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

Our KISS approach:

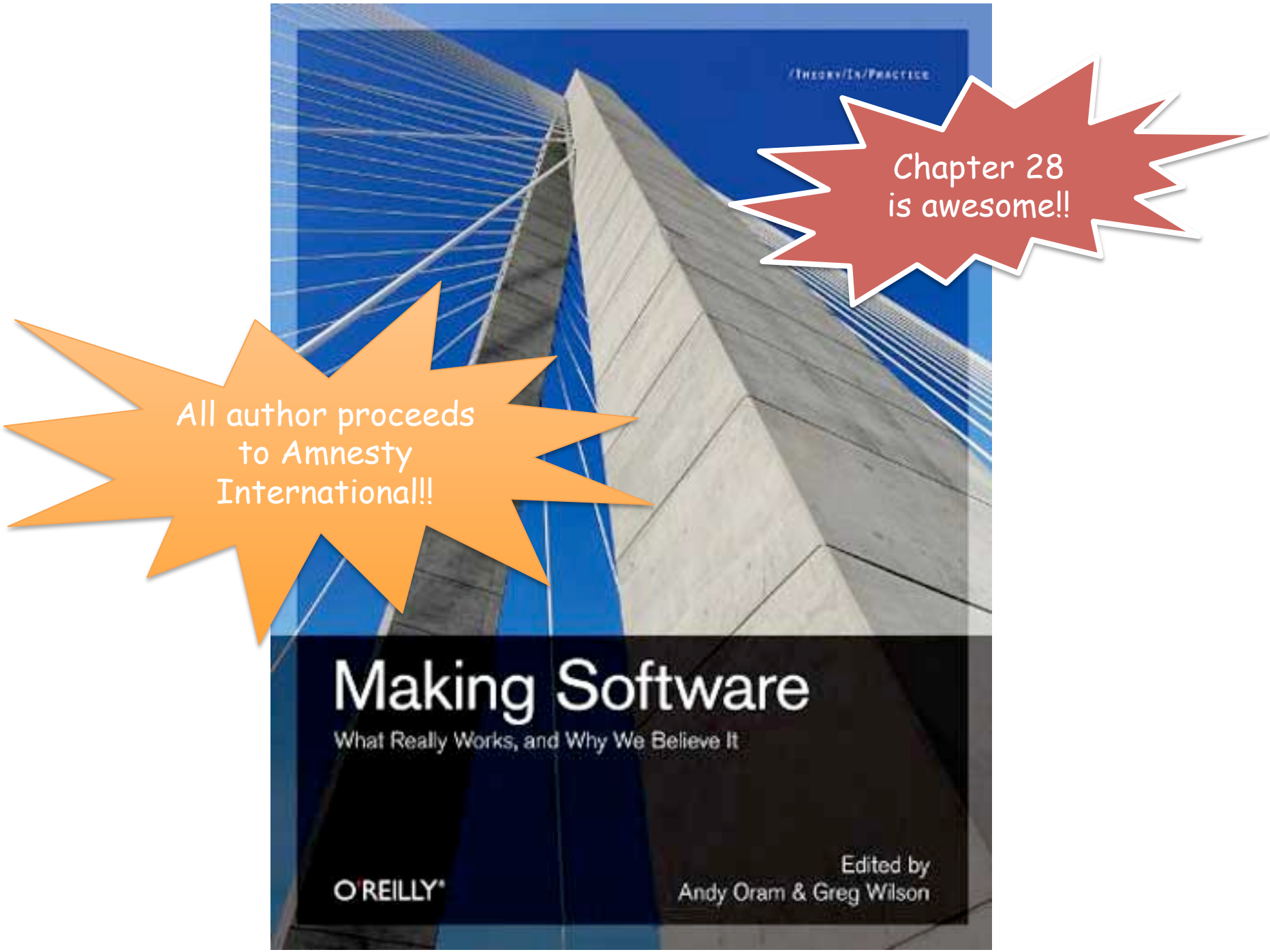
- Consider only class / method signatures
  - May not have source, compiler options may differ, …
- Build master repos of signature hashes from Maven2
  - Which has gaps, duplication, errors,
- Compare sig hashes of target appl. against master repos
  - There will be false positives when API does not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

[Davis, German, Godfrey, Hindle, MSR-11]


Maven 2

Summary

- Who are you?
  - Determining software entity provenance is a growing and important problem
- KISS / software Bertillonage:
  - Quick & dirty techniques applied widely, then expensive techniques applied narrowly
- Identifying version IDs of included Java libraries is an example of the software entity provenance problem
  - And anchored signature matching is an example of KISS / software Bertillonage


Chapter 28
is awesome!!
All author proceeds
to Amnesty
International!!
Making Software
What Really Works, and Why We Believe It
O'REILLY
Edited by
Andy Oram & Greg Wilson

20th IEEE Intl. Conference on Program Comprehension

http://icpc12.sosy-lab.org/

- To be held June 11—13, 2012, Passau, Germany
  - Right after ICSE in Zürich, a special bus is being arranged
- Abstracts due Feb 10, 2012, full papers due Feb 17, 2012
- Dedicated tracks for:
  - Industrial papers, tools, posters, plus a PhD student symposium
- Program co-chairs:
  - Arie van Deursen, Delft Technical University
  - Mike Godfrey, University of Waterloo

Squinting at the data

Investigating software entity provenance using KISS techniques

Mike Godfrey

Software Architecture Group (SWAG),

UWaterloo [visiting CWI until July 2011]