"Cloning Considered Harmful" Considered Harmful

A look back

Cory Kapser and Mike Godfrey
University of Waterloo
WCRE 2006, Benevento Italy

Cory Kapser

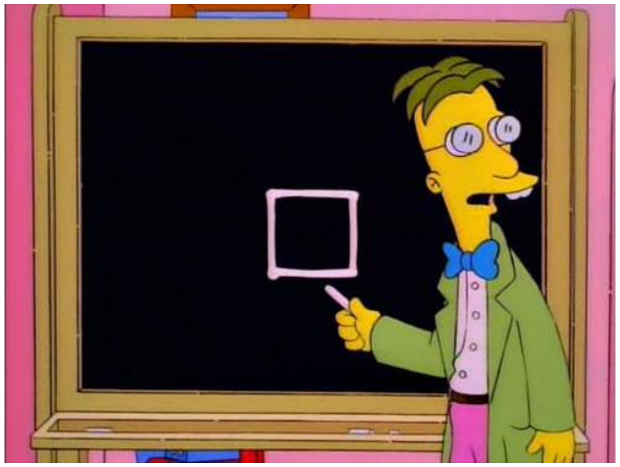
- Living in Calgary since graduating in 2009
- Working at a startup

His office, 1986

Talk at IBM Toronto, 1999

My office, 2002

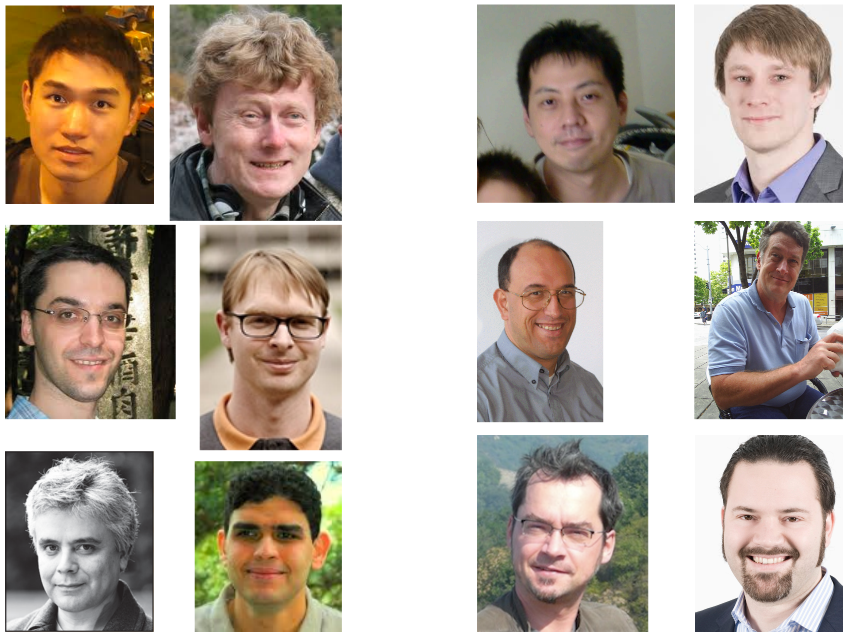
IWPSE 2004, Kyoto



Many conferences, incl. MSR 2006

IWPC keynote, 2003

WCRE 1998, and many others since



Dolly, RIP

The ages of software cloning research

A brief look back and forward

[Stolen from IWSC 2015 keynote]

Math, science, and engineering

"Where's the science in what you do?"

- *Science* concerns building reliable explanatory models of how the world works
  - Scientific models must be testable, and (reasonably) consistent with observed reality
  - Scientific models may be statistical, structural (Newtonian mechanics), …
  - Where do the models come from?
    - Experience with the domain, grounded theory, …

Math, science, and engineering

"Where's the science in what you do?"

- *Math* isn't really science, per se!
  - Its only hard requirement is self-consistency; good math need not have practical applications.
  - Math is a kind of poetry, with rigorous rules of construction.
  - Math is a *tool* used by scientists to help build and analyze models of how the world works

Math, science, and engineering

"Where's the science in what you do?"

- *Engineering* is (roughly) the practical application of science to solve real-world problems
  - Must understand "how the world works" to get stuff done
  - Engineers must also know about processes, materials, costs, risks, tools, people, law, ethics, etc.

The three overlapping ages of software cloning research

1. The Age of Math:

CLONE *DETECTION* IS POSSIBLE!

  - Algorithms exist, can scale to big systems!

[1990s: Baker, Johnson, Baxter, Ducasse, Merlo, …]

[2000s: CCFinder, iClones, NiCad, ConQAT, …]

The three overlapping ages of software cloning research

2. The Age of Science:

CLONE ANALYSIS IS POSSIBLE!

- Let's assume detection "just works", what can you tell me about the system and its clones?
  - *Some clones evolve, some don't*
  - *Type 3 clones are more stable / less buggy / ...*
- Not just source code! Other artifacts matter! We do MSR!
  e.g., StackOverflow, Bugzilla, `git` meta-data

[2000s-2010s: Krinke, Kim, Kapser, Jürgens, ... IWSC-16]

The three overlapping ages of software cloning research

3. The Age of Engineering:

CLONE MANAGEMENT IS POSSIBLE!

- Clone triage, clone refactoring, linked editing, clone recommendation, program transformation, SPLs, ...

[2000s-2010s: Robillard, LaToza, Basit, ...]

The road ahead: A look back?

- Good news: We've accomplished a lot!
  - We know what we can detect, how well, and at what scale
  - We've done many empirical studies on type 1/2/3 clones
- ... but we still aren't sure which clones are important / risky and why
  - So maybe we need more comprehensive models of cloning *as practiced by developers and experienced by managers*

Controversial statement

If cloning research is to have impact on practice, then our immediate scientific goals must be more developer oriented

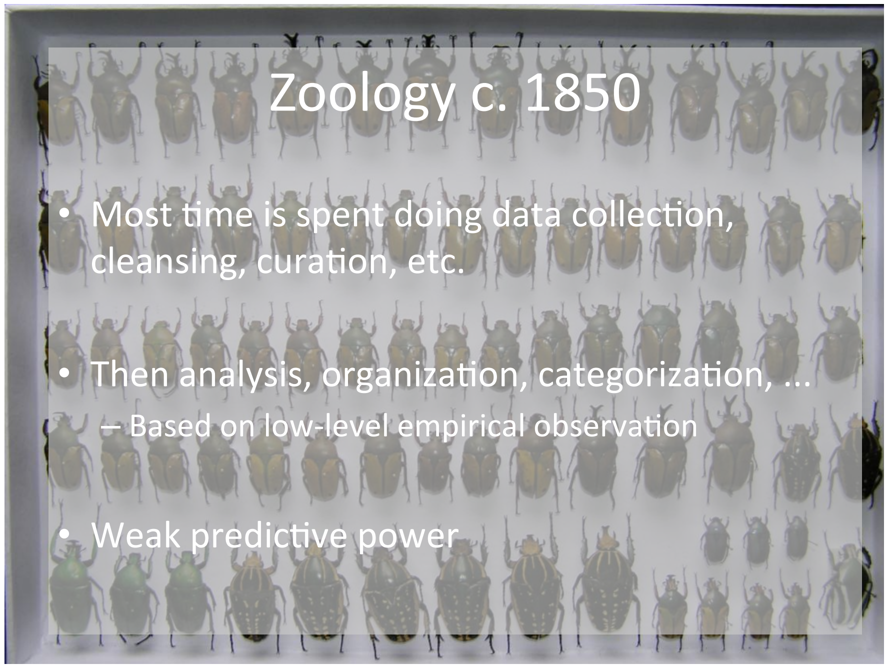
- It is not enough simply to find clones and then refactor (some of) them; rather, we must ask questions such as:
  - *Why do these clones exist in the first place?*
  - *What design decisions led to their creation?*
  - *How do developers and managers perceive them?*
  - *What possible risks do they represent to the ongoing development of the software system?*
  - *How can we recognize clones that need management?*
  - *What strategies should we use to manage them over the long term?*

"Physics is the only real science.

The rest are just stamp collecting."

Ernest Rutherford (1871 – 1937)

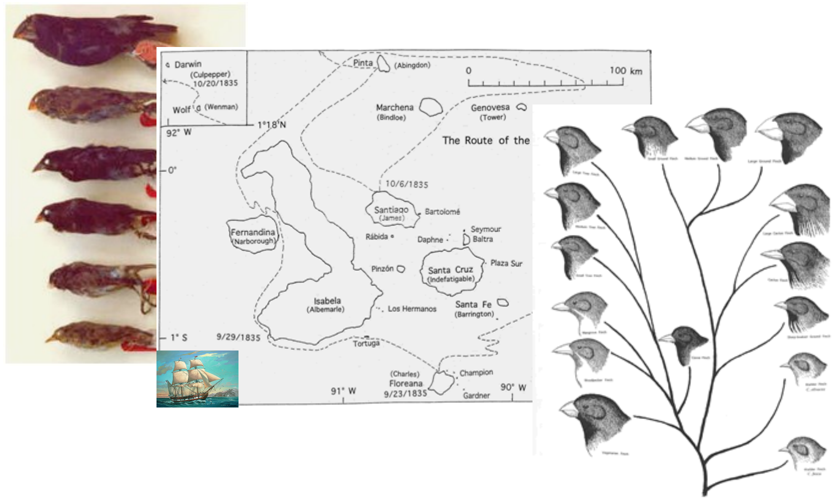
Father of atomic physics
Professor at McGill Univ.
&Univ. of Edinburgh
Nobel prize for … chemistry



Zoology c. 1850

- Most time is spent doing data collection, cleansing, curation, etc.
- Then analysis, organization, categorization, …
  - Based on low-level empirical observation
- Weak predictive power

Along comes Darwin …



A taxonomy of cloning intent

1. Forking
   - Hardware variation
     e.g., Linux SCSI drivers [SCAM 2011]
   - Platform variation
   - Experimental variation
2. Templating
   - Boilerplating
   - API / library protocols
   - Generalized programming idioms
   - Parameterized code
3. Post-hoc customizing
   - Bug workarounds
   - Replicate + specialize

"'Cloning considered harmful' considered harmful",
Cory J. Kapser and Michael W. Godfrey, *WCRE 2006*

Forking: Platform variation

- Motivation
  - Different platforms ⇒ very different low-level details
  - Interleaving platform-specific code in one place is too complex
- Well known examples
  - Linux kernel “arch” subsystem
  - Apache Portable Runtime (APR)
    - Portable impl of functionality that is typically platform dependent, such as file and network access
    - `fileio -> {netware, os2, unix, win32}`
    - Typical diffs: insertion of extra error checking or API calls
    - Cloning is obvious and well documented

Forking: Platform variation

- Advantages of cloning
  - Each (cloned) variant is simpler to maintain
  - No risk to stability of other variants
  - Platforms are likely to evolve independently, so maintenance is likely to be “mostly independent”
- Disadvantages of cloning
  - Evolution in two dimensions: user reqs + platform support
  - Change to the interface level means changes to many files

Forking: Platform variation

- Management and long-term issues
  - Factor out platform independent functionality as much as possible
  - Document variation points + platform peculiarities
  - As # of platforms grows, interface to the system hardens
- Structural manifestations
  - Cloning usually happens at the file level
    - Clones are often stored as files (or directories) in the same source directory
    - Directories may be named after OSs or similar

Cloning harmfulness: Two open source case studies

		Apache		Gnumeric	
Group	**Pattern**	**Good**	**Harmful**	**Good**	**Harmful**
Forking	Hardware variation	0	0	0	0
Forking	Platform variation	$\boxed{10}$	0	0	0
Forking	Experimental variation	4	0	0	0
Templating	Boiler-plating	5	0	6	7
Templating	API	0	0	0	9
Templating	Idioms	0	12	1	1
Templating	Parameterized code	5	12	$\boxed{10}$	$\boxed{34}$
Customizing	Replicate + specialize	12	4	$\boxed{15}$	$\boxed{16}$
Customizing	Bug workarounds	0	0	0	0
Total		**36**	**28**	**32**	**67**

Apache httpd 2.2.4 - 60 Tokens
Gnumeric 1.6.3 - 60 Tokens

The challenge for future cloning research

- Grand theories and "actionable" big ideas are a noble goal, of course!
  - It helps to avoid "yeah, OK, but who cares?" papers
- … but learning to "swim with the data" leads to higher quality research in the long run
  - It abets opportunistic exploration of the problem space
  - … which lead to deeper insights about the problem space
  - … and makes fundamental naïve mistakes less likely

Thank you