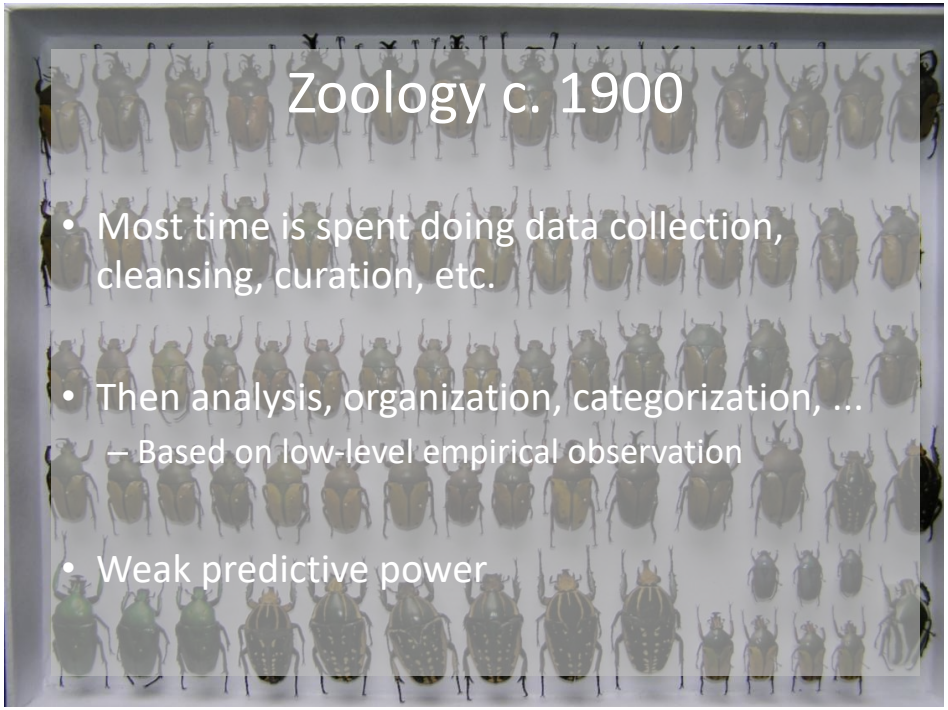
CS846 — Empirical Software Evolution

Fall 2025, Tues 2:00-4:50pm

Mike Godfrey, DC2340
migod@uwaterloo.ca
@migod on twitter

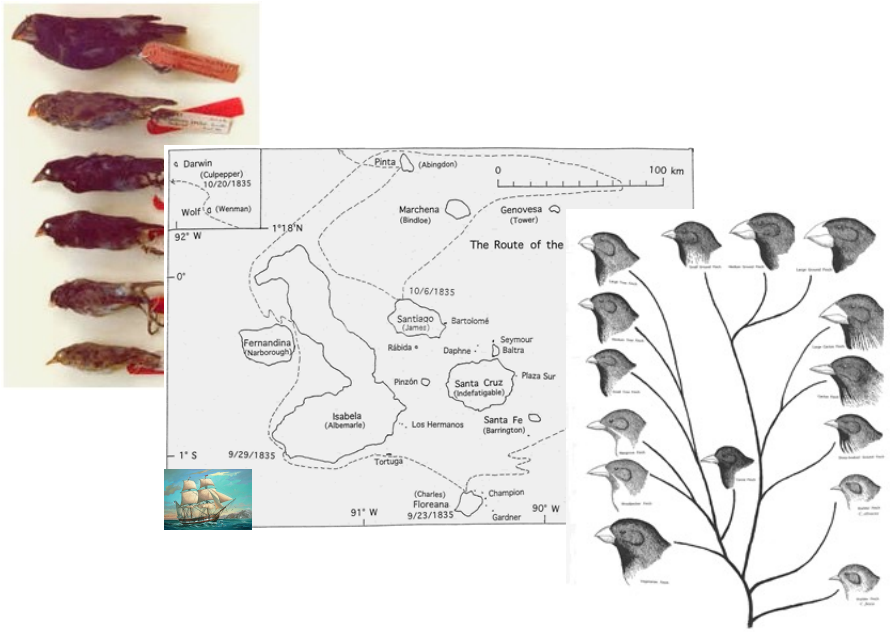


Topics and themes

"Physics is the only real science.
The rest are just stamp collecting."

Ernest Rutherford (1871-1937)

Father of atomic physics
Nobel prize for … chemistry



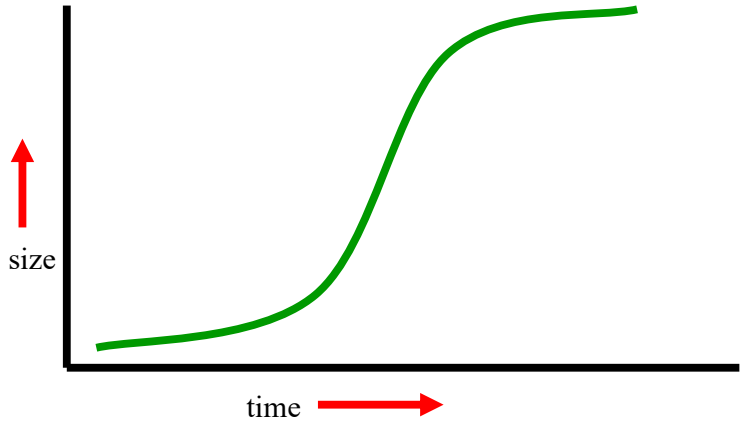
stackoverflow

Gerrit Code Review

Jenkins

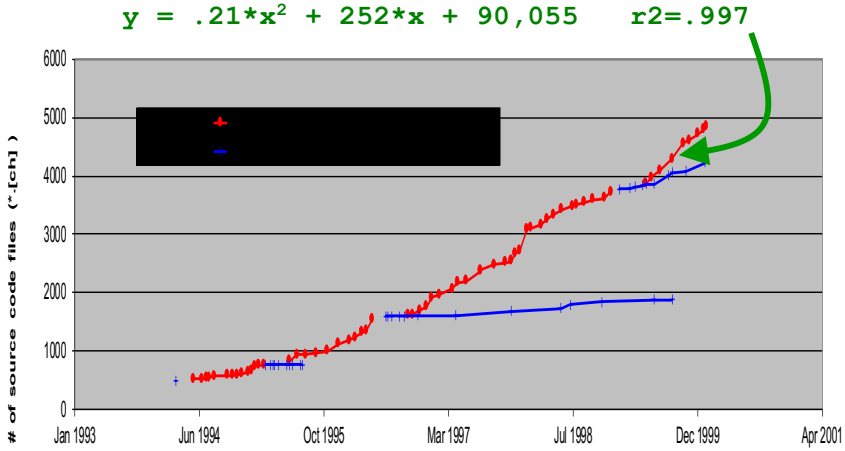
Travis CI

The "S" curve of successful growth

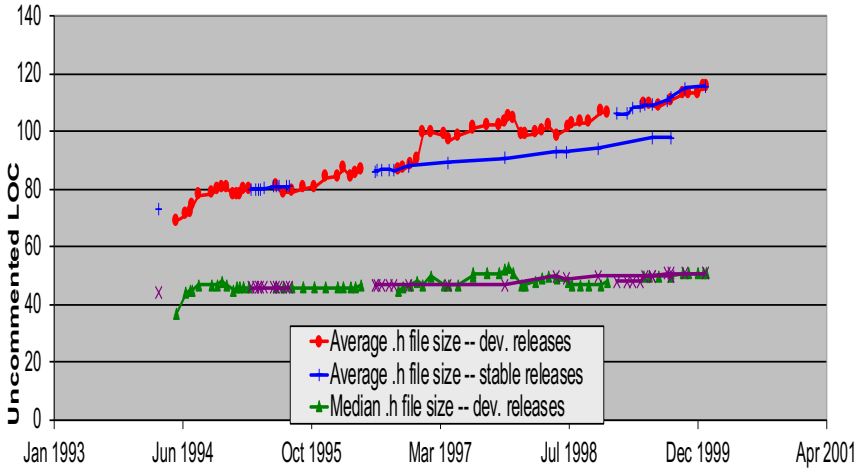


Growth of the Linux kernel source tree

(# of src files)



Average / median `.h` file size



Consider this code…

```
const char *err = ap_check_cmd_context(cmd, GLOBAL_ONLY);
if (err != NULL) {
   return err;
}
ap_threads_per_child = atoi(arg);
if (ap_threads_per_child > thread_limit) {
   ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
            "WARNING: ThreadsPerChild of %d exceeds ThreadLimit "
            "value of %d", ap_threads_per_child,
            thread_limit);
   ....
   ap_threads_per_child = thread_limit;
}
else if (ap_threads_per_child < 1) {
   ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
            "WARNING: Require ThreadsPerChild > 0, setting to 1");
   ap_threads_per_child = 1;
}
return NULL;
```

and this code …

```
const char *err = ap_check_cmd_context(cmd, GLOBAL_ONLY);
if (err != NULL) {
   return err;
}
ap_threads_per_child = atoi(arg);
if (ap_threads_per_child > thread_limit) {
   ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
            "WARNING: ThreadsPerChild of %d exceeds ThreadLimit "
            "value of %d threads,", ap_threads_per_child,
            thread_limit);
   ....
   ap_threads_per_child = thread_limit;
}
else if (ap_threads_per_child < 1) {
ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
            "WARNING: Require ThreadsPerChild > 0, setting to 1");
ap_threads_per_child = 1;
}
return NULL;
```

Major theme

Evidence-based program comprehension

- Understanding a software system means understanding its parts, their inter-relationships, and their histories
- A software system is more than just compiled source code
  - Myriad source artifacts + deployment environments + development processes + supporting tools + developers + user community
- To understand the history of a software system, you have to see the big picture through the lens of data analytics and socio-technical context

Major theme

Evidence-based program comprehension

- To better understand people, we can use techniques from social sciences (to produce data)
  e.g., interviews (pre and post study), surveys, grounded theory
- To understand tools and processes, we can use instrumentation (to produce data)
- To better understand software artifacts (and other data), we can use techniques from data science
  e.g., machine learning and LLMs, data mining, NLP, statistics

Major theme

Evidence-based program comprehension

- There are many, many kinds of development artifacts!
  e.g., internal docs (requirements, design, testing), git commits (+ meta-data), issue tracking histories, build/deploy scripts, test suites, developer mailing lists, execution logs, …
- Development artifacts have explicit internal structure and both explicit and implicit interrelationships
- All of these artifacts have histories, so we can track evolution, measure long term effects and costs, … over time

Other topics

- Data science applied to sw development artifacts:
  - aka Mining Software Repositories / software analytics
  - What can we do now? How accurate is it? How useful?
  - Can we make sense of / link up the many kinds of artifacts?
  - Is there enough signal in the data?
  - Are we enabling bad management?
  - Actionable advice: Is that the gold standard?

Other topics

- AI4SE
  - i.e., using AI techniques such as LLMs, NLP, ML to aid in software engineering tasks: code review, defect prediction, code summarization, code completion and recommendation
- SE4AI
  - i.e., the design of specialized SE techniques to aid in developing software systems that build around LLMs ("FMware")
- What is program comprehension?
  - Mental models and cognition
  - How can we evaluate comprehension?
  - What is the value of software visualization?
  - Does instrumenting IDEs provide useful knowledge?

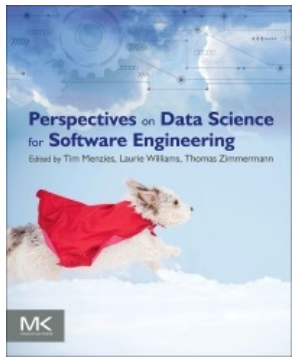
Meta-topics

- What to do when the data is messy, incomplete, noisy, ambiguous, wrong, …
- Measuring and metrics
  - "Not everything that can be counted counts; not everything that counts can be counted."
- Use of statistics, machine learning, LLMs, …

Some of the readings

- *"Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?"*
- *"Can LLMs replace manual annotation of software engineering artifacts?"*
- *"Software Bertillonage: Finding the provenance of a software artifact"*
- *"Code today, deadline tomorrow: Procrastination among software developers"*
- *"With great humor comes great developer engagement"*
- *"Wolves in the repository: A software engineering analysis of the XZ utils supply chain attack"*

Course text book



- *Perspectives on Data Science for Software Engineering*, 2016
  Tim Menzies, Laurie Williams, Thomas Zimmermann (eds.)
  - Very short, very readable chapters explaining key ideas, techniques, and experiences applying data science techniques to software development artifacts (aka software analytics aka Mining Software Repositories)

Logistics

http://plg.uwaterloo.ca/~migod/846

Some student feedback

- *"This was probably the best class I've been in all the 17 years of my education."*
- *"[Student] presentations were a good tool to improve one's speaking skills."*
- *"[Prof] leads discussions [which were] helpful & motivating."*
- *"Very interesting material. Strong departure from the stuff I normally see. Interesting to get a different set of opinions / topics / points of view."*

Some student feedback

- *"Great selection of papers. [Prof] very knowledgeable; excellent at leading interesting / informative discussions."*
- *"Paper selection was very good. The textbook was very helpful to introduce the subject of empirical sw eng."*
- *"Interesting and diverse topics."*
- *"[Class atmosphere] was positive and calm."*
- *[The prof interrupted too much to add his own two cents / background / opinions. Also, he nitpicked on presentation details, which I found a bit off-putting. But I liked the course!]*

I need (three + three) volunteers for next week

1. *"No silver bullet: Essence and accidents of software engineering"*, Fred Brooks, *IEEE Computer*, April 1987.
   - Presenter: Jacie Jermier
   - Scribe: Youssef Souati
2. *"The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluation"*, Blackburn et al., *ACM Trans. on Programming Languages and Systems (TOPLAS), 38(4), Oct. 2016.*
   - Presenter: Brian Do
   - Scribe: Tongwei Zhang
3. *"Towards AI-native software engineering (SE 3.0): A vision and a challenge roadmap"*, Hassan et al., arXiv, Oct. 2024
   - Presenter: Asim Waheed
   - Scribe: Michael Ogezi

CS846 — Empirical Software Evolution

Fall 2025, Tues 2:00-4:50pm

Mike Godfrey, DC2340

migod@uwaterloo.ca

@migod on twitter