Outline

- My opinions/experience
- Advice for Developing Bad Tools
- Advice on Alternatives to Bad Tools
- Things the community can do to encourage good tools
Define “tool”

- Software/testbed used in research
  - AKA “Infrastructure”
  - AKA “Artifacts”
- Stuff needed to generate data for a paper
  - Or to prototype an idea
- Architecture: Simulators, emulators, FPGA designs, actual chips
- PL, OS, etc. have similar things

- What we spend most of our time on
Goal of a good research tool

IMPACT

 Scholar

10,000

Cited by 4346 Related articles
How to develop a bad tool

Some tips and tricks
Create “in-house” tools

• The idea is good enough to have impact on its own!
• Others can easily build on to of your ideas
  • It’s simple to recreate all of the details
    (it only took you a couple of weeks to build, right?)
  • Funders love to pay to recreate others’ work
• There’s no need for anyone to reproduce your results
  • Why wouldn’t they trust you? You never make mistakes
• Keeping it in house gives you a competitive advantage
  • Research is a competition and a zero-sum game
Don’t follow software best practices

• No one will see your code
  • It’s OK that the code is ugly
  • “Research-quality” code is a thing, right?
  • Monolithic designs are best!

• Who needs version control?
  • You never make mistakes
  • Your hard drive will never die

• Don’t write tests or validate
  • You’ll never add back a bug you squashed
  • No need to check to make sure you model is good

• Don’t write documentation
  • You’re the only one who will ever use this tool anyway
  • You’ll remember all of these details in 3 years, no problem
Make the tool “open source”

• Force people to register before giving code
  • If someone really wants your code, they’ll ask
  • Everyone loves registering and waiting for permission

• Provide a tarball of the source
  • With the source anyone can build and use the tool
  • Why would anyone need anything more than the source?

• Don’t provide a license or use a restrictive license
  • Clearly you own all rights to the code
  • Or, it’s “open source” so obviously it’s OK to use
  • GPL is great because it forces others to be open source (🔥🔥🔥)
Don’t market your tool

• No need to come up with a good name
  • Common English words are memorable
  • Someone else thought the name was good, let’s copy it!

• Don’t create a webpage
  • Domain names are expensive, and IT is mean

• No need to provide context beyond the paper
  • Research papers are great documentation
  • People love reading PDFs behind paywalls
Stop supporting the tool

• Once your paper’s published, no need to continue working on the tool
  • You’ve gotten everything you need out of it
  • Since the paper is accepted, the tool must be perfect

• No one will need help using your tool
  • You wrote a perfect tool, there can’t be bugs
  • The documentation you wrote is perfectly clear

• People love links to dead webpages
  • Extra points if it looks like your webpage was created in 1995
How **NOT** to develop a **bad** tool

- **Do** share your tool: **Let others use and develop**
  - **Do** share your tool as widely and easily as possible!
  - **Do** make your tool open source
  - **Market** your tool anywhere and everywhere
    Websites, tutorials, books, videos, etc.

- **Do** follow software best practices: **Make it easy for others to use your tool**
  - **Do** use git, good design practices, ...
  - **Do** use agile development practices, code review, ...
  - **Do** use the most popular tools for your tool

- **Do** support the tool: **Help others use your tool**
  - **Do** provide documentation and support
  - **Do** continue development after initial release
Make the tool Capital Open Source

- Include a LICENSE file with all distributions
- Use an OSI approved license (opensource.org)
  - Industry prefers more permissive licenses
  - Apache v2, BSD are good choices
  - Use creative commons for documentation / teaching
- As the project grows, the leadership should mature
  - Governance document defining how to make decisions
  - Committee for management
- Think about the exit strategy
  - Without an exit strategy the project will languish
  - Moving under an umbrella
  - Startup, nonprofit, etc.
Create a community around a tool

• Foster a community so that others can give back
• Answer questions when they come up
  • Mailing list, github issues, slack, etc.
• Provide answers to questions before they come up
  • Documentation is hard, but very important
  • Document for both users and developers
  • readthedocs.org is a great tool
• Include a CONTRIBUTING guide and a CODE-OF-CONDUCT
  • Make the community inclusive and accepting
  • The broader the community the more impact the tool will have
But how? (From academia)

- (Research) Incentive structure pushes us towards bad tools
  - Bean counters, not fertile soil counters
  - Need more recognition:
    - Infrastructure papers?
    - Awards? Artifact badges are a great start!
    - Count commits? Code reviews? Stackoverflow posts?

- Funding is for research not infrastructure
  - 3-year grant: papers published in years 2&3
  - The experts (students) graduate or end their internships
  - Need more “lab techs” in systems research
    - Software developers to provide continuity
    - Continuing infrastructure development funds
But how? (“Successful” projects)

• What do these projects have in common?
  • *Significant* industry uptake and support
  • Most development comes from industry

Virtuous cycle for research?

The Answer: Projects With Sustainable Ecosystems

Sustainable projects have a developer community whose technology is used in projects that produce papers for groups who in turn participate and reinvest back into the project and hire students to work in the community.

Research funding?
Intertwine tools and research

• What if the tool development leads to research? and vice-versa
• Satisfy the bean counters with agile development
• **Really** hard with current incentives
Back down to Earth

• Is your tool the next LLVM?
  • Probably not
  • But wouldn’t that be cool?!?!
• Develop like it is the next big thing

This will help you in your research, help the community’s research, help scientific progress, and increase the impact of your tool
we can all GREAT
How to Develop a Bad
Research Tools

Jason Lowe-Power  @JLowePower
University of California, Davis

Say NOPE to bad tools
Common arguments for bad tools

• I might be embarrassed
  • It’s “research quality”...
  • Some one might find a bug
  • What if my research is invalidated???

• Company will expose IP
  • Then why are you publishing?

• Faster to develop a bad tool
  • True, but what about a few years from now?