The Future of Software Engineering: Tools

Michael Ernst
MIT Lab for Computer Science
http://sdg.lcs.mit.edu/~mernst/
Tools matter

The most productive people are the most effective tool users

Tools are a change agent
Reproducibility

Replication is key to the scientific method
  Papers may gloss over details
Insist on published systems
  Permits additional evaluation
Stop accepting papers without supporting data and implementations
  Central repository for such information
Need support for infrastructure-building
Counterargument to publication

Responsibility to have commercial impact

Can have commercial impact without IP
  If it's a good idea, it will be adopted
Secrecy is antithetical to science & education
Evaluation of implementation

Tools should work

• for users other than implementers
• when run on production systems

Correct the impression that software engineers do not build real systems
Evaluation of tools

Humans complicate systems
   Theorems much easier to understand!
Case studies, not experiments
Devise rewards for reproducing experiments
   Special publication venues?
   Do not raise the bar for new ideas
Industry can assist academia
Ignore the average programmer

Practitioner/researcher gap

What is the job of researchers?

Researchers must assume best practices, in order to have long-term impact
Static and dynamic analysis

How to obtain information:

• Programmer-supplied

• Static analysis: examine the program text
  • properties are guaranteed to be true
  • pointers are intractable in practice

• Dynamic analysis: run the program
  • efficient, precise
  • complementary to static techniques
Combining techniques

Use static to help dynamic and vice versa
Transfer from one domain to the other
Example: Purify and LCLint
Combine in a principled way
Select the desired efficiency/soundness
Lightweight tools

No need to produce exact results
Useful results that people can check
The return of formal specifications

Full formal specifications do not work

Partial formal specifications:
  • Specify only certain aspects of behavior
  • Generated automatically

Other examples from languages and compilers
  Example: functional programming
Summary

Tools are key
Publication must include tools
Case studies
Assume best practices
Static and dynamic analysis
Lightweight tools