Why do we still have bugs?

an almost economic perspective
on the cost of software quality

Ras Bodik UC Berkeley

Two conflicting trends?

Huge reductions in cost of software quality possible.

Yet bugs still cost us about 50%, as they did 50 years ago

Where have all improvements gone?

A revealing question:

If given a magic debugger/verifier, how would you use it?

a) reduce the cost of your product

b) build a more advanced product ← typical answer

Clue: Who switches from Apple to Linux because Linux is cheaper?

The hypothesis

You translate advances in software quality to competitive advantage. Of course.

… by building better products, not cheaper products.

Better products are more ambitious and have more bugs.

In the process, cost of your bugs will grow back to 50%.

Why do we still have bugs?

Because customers are willing to pay for them.

Stable state?

Is spending 50% on bugs an equilibrium?

-- a cost we can recoup from the profit margin?

In case you ask: yes, we can build software more cheaply, eg with Rails, but it’s not good enough.

Corollary

The seemingly alarming cost of bugs is a **good sign**.

It means we are not (yet) a **commodity** industry, where margins are lower, and so are salaries.

So, a healthy cost of bugs means that we are pushing the boundary of what we can build.

We could almost end with this message

How I learned to stop worrying and love the bugs.

But there are some technical observations…

Cost of Software Quality, short term

Short term, make sure the cost is going down, thanks to technological and process innovations, or else there will be no room for better products

Cost of Software Quality, long term

Long term, make sure the cost is staying stable (and high), otherwise you are entering a commodity business

If software is easy to build, anyone can do it and margins go down.

The panel question revisited

Do we still have the same bugs?

What bugs can we expect tomorrow?

Can we steer things so that future bugs don’t choke us?