Team Leader Simulation Cost-Benefit Analysis: Dynamic vs Static Testing

Scenario A (simplistic, testing-only)

You are team leader, managing the development of a NEW software app.

It is near release.

You must decide on your testing strategy.

The app has 10 modules of equal size, around 2000 LOC each.

Estimated Bugs (typical industry rate is 6% of untested lines of code have some error):

- 6% * 2000 = 120 bugs/module
- = 1200 estimated bugs in whole app.

Strategies:

- Unit Testing (Automated): $1000/module, finds around 50% of bugs.
- System Testing (Manual): $100/hour on whole system, finds 10 bugs/hour, especially the more severe/common bugs.

With $5000 to spend, what is your testing strategy?

- What is the minimum bugs of bugs you could release with?
- What is the best practical strategy that you would recommend?

Scenario B: Testing-Only, release 2, 3, 4…

In each release, about half the modules (5/10) have changed.

Any existing automated unit tests will still catch 50% for free.

- In reality, we should consider the test update costs, reduced test effectiveness, etc.

Manual testing can focus on new/changed behaviours.

But note danger of not regression testing – major bugs might slip into existing behaviours!

=> benefits of having some automated regression tests...

Scenario C: Static Analysis

Setup cost $1000. Once-off, so view this as an investment.

Analysis tool is fully automatic, so run cost is $0. (Ignoring compute cost).

- Will find ALL bugs of a given type, like #1 SQL-injection, buffer overflow. NPE, etc.
- NOTE: In 2019, 'input manipulation' attacks like SQL-injection and buffer overflow were 66% of reported vulnerabilities. And 34% 3771/11092 of vulnerabilities in first half of 2019 were still unpatched in Aug 2019. [TechRepublic.com Brandon Vigliarolo, 23 Aug 2019.]

Looks like a silver bullet!

But what are the hidden costs?

What are the limitations?