CSC 509 Lecture Notes Week 5

The Promise of Spec-Based Test Generation

-- *a brief oveview of of a new direction* --

I. **Again, common refrains about manual test gen:**

A. It’s boring and tedious.

B. It’s error prone.

C. May leave important things untested.

D. There’s got to be a better way.

II. A Possible Better Way

A. The slick pitch --

"Suppose all you have to do is write a couple simple boolean expressions per method, and a decent set of unit tests will be automatically generated.

Would you use a tool that does this?"

B. Some skeptical questions.

B. Some skeptical questions.

1. Is this really possible, i.e., can someone build a working version of this tool that I can use as part of my normal program development workflow?

B. **Some skeptical questions.**

1. Is this really possible, i.e., can someone build a working version of this tool that I can use as part of my normal program development workflow?

2. When you say *"a couple simple boolean expressions"*, how really "simple" are they.

B. **Some skeptical questions.**

1. Is this really possible, i.e., can someone build a working version of this tool that I can use as part of my normal program development workflow?

2. When you say *"a couple simple boolean expressions"*, how really "simple" are they.

3. If this really is doable and simple enough, how come it hasn’t happened yet?

C. Some non-Skeptical Answers

C. **Some non-Skeptical Answers**

1. Yes, several such tools have been built in the last twenty years.

C. Some non-Skeptical Answers

1. Yes, several such tools have been built in the last twenty years.

2. The boolean expressions are simple enough for most competent programmers.

C. Some non-Skeptical Answers

1. Yes, several such tools have been built in the last twenty years.

2. The boolean expressions are simple enough for most competent programmers.

3. It hasn’t happened yet because ...

It hasn’t happened yet because ...

a. The tools were written for languages and environments that aren’t or weren’t widely used.

It hasn’t happened yet because ...

a. The tools were written for languages and environments that aren’t or weren’t widely used.

b. There’s a chicken-and-egg problem with convincing programmers to adopt a new notation when there’s no immediate tangible benefit.

It hasn’t happened yet because ...

a. The tools were written for languages and environments that aren’t or weren’t widely used.

b. There’s a chicken-and-egg problem with convincing programmers to adopt a new notation when there’s no immediate tangible benefit.

c. There remain technical challenges in generating genuinely "decent" tests.

D. Some new ideas that just might work:

D. Some new ideas that just might work:

1. Provide a mixed-language notation for popular languages like C, C++, Java, Python, and Ruby.

D. Some new ideas that just might work:

1. Provide a mixed-language notation for popular languages like C, C++, Java, Python, and Ruby.

2. Simplify the spec-language to the bare minimum necessary for test generation.

D. Some new ideas that just might work:

1. Provide a mixed-language notation for popular languages like C, C++, Java, Python, and Ruby.

2. Simplify the spec-language to the bare minimum necessary for test generation.

3. Generate tests that are very readable.

D. Some new ideas that just might work:

1. Provide a mixed-language notation for popular languages like C, C++, Java, Python, and Ruby.

2. Simplify the spec-language to the bare minimum necessary for test generation.

3. Generate tests that are very readable.

4. Generate tests that are 100% executable in any environment for the specified language.

III. Further details in the 509 project "white paper" at

```
http://users.csc.calpoly.edu/
   ~gfisher/classes/509/project/
      summary.html
```