Quiz

CSC131 The Beauty & Joy of Computing

05 September 2018

1. "TDD" is "Test Driven Development." It is part of a discipline for creating software. Software engineers who use this approach write the tests that their code must pass before they write the code.

   Engineers who write tests after creating a product are subject to bias—knowing how they wrote the software, they write tests that exercise the functions that they know they built into the product, but might miss testing cases that they neglected to cover in their design.

   How might this way of writing software correspond to the ways in which the authors of our books suggest that we go about learning any subject?

2. Regression testing is another widely used discipline. Software engineers subject their code to many tests. They do not apply a test only until the code passes. They continue repeating the test even after the code passes.

   Software engineers use regression testing because they understand that errors can recur.

   Here's how—imagine that one person on a team interprets a specification incorrectly, makes a false assumption, or makes an error in logic. A second person fixes the error. A third member of the team, in the course of adding another feature or fixing another defect, might then repeat the mistake that the first person made. The same ambiguity in a specification or complexity in an algorithm that confused the first contributor can confuse another that contributor's teammate.

   Regression testing guards against the reintroduction of errors.

   How might this way of writing software correspond to the ways in which the authors of our books suggest that we go about learning any subject that we might study?

3. Software engineers learned that the composition of a detailed design at the outset of a project, followed by months or years of work to build according to that design, and then testing at the end of the project often leads to failure.

Experience taught them to recognize great risk of failure in plans that proceeded by taking instructions, working alone for a long time, and then emerging (they hoped) with a finished and acceptable product.

Software engineers today prefer the agile approach: build a small prototype, put it in the hands of customers (or prospective customers), and improve the product with the help of the customers' suggestions. Repeat this process over and over again. Construct the product in small steps. Solicit feedback at each step.

How might you adapt this aspect of the agile approach in your studies at Cornell College?

4. Agile developers prefer short deadlines. For example, they might set for themselves a goal that they can meet in two weeks.

   The division of a task into many smaller subtasks and the commitment at each stage to modest goals and frequent due dates give a team a better way of measuring progress. It makes steady progress easier to achieve. A team that sets for itself a single big goal and a deadline far in the future will often face panic and a need to work overtime at the end of the alloted time.

   What might be a suitable interval for benchmarks in projects that you undertake at Cornell College?

5. Software engineering is a younger profession than civil, mechanical, electrical, or even aeronautical engineering. The first programmers invented their own ways of working. Many worked without any formal discipline, guided by intuition. The writing of software was a black art—mysterious, forbidding, dangerous. Software engineers soon recognized the need for discipline. They borrowed methods from older professions. They invented methods and measures to meet the special challenges of writing software.

   For example, they learned that, while many of them had learned programming through solitary practice, success in professional practice requires teamwork. Small teams work better than large teams. A common rule of thumb says that a team is too big if it takes more than two pizzas to feed it.

   How might these lessons on teamwork apply in your studies?