**Cambia Take-Home Exercise**Jason Gill  
10/12/2018

**Tools**

1. In your opinion, what’s helpful about version control systems? What’s annoying about them?

Version control systems are helpful to keep track of changes, allow for changes to be shared, and with the right tools in place, know that a given change has caused tests to fail.

One thing that is annoying about version control systems is conflict resolution when two people are changing the same file.

2. What are some pros and cons of using Docker to develop, test, and deploy software?

One thing Docker does well is the idea of containerization: (from Docker team) "A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings."

This makes creating, testing, and deploying of software get away from the "works on my machine" situation by doing its best at not carrying about what system the application is running on.

Docker containers are an excellent way to deploy applications that are more modular and easier to manage. There are also security benefits to using containers, and containers consume system resources more efficiently.

A con of using Docker is that the containers, by design, all the data inside a container disappears forever when the container shuts down, unless you save it somewhere else first. The way to save outside of a container becomes overly complicated however.

3. How do you choose which language to use for a given task? How did you choose the language for the programming exercise above?

My general experience of chosing a language comes from what I am going to be doing with that code, and my experience using a given languange. If I were to be creating something in the web, I am likely to start using JavaScript to handle things.

I chose to create a C# application for the task above, one reason, because handling I/O in C# on Windows is easy to implement and see the result. Another reason is that I have been creating C# apps for a while, and creating a C# tool is quick and easy to understand.

**Testing Methodology**

1. What’s the right role for QA in the software development process?

I see the role of QA in software as being critically important. QA and the developers need to communicate to make sure the design is understood before either can start on their given task. QA should be there before development begins and should be one of the final roles to verify that the development done was done as planned.

2. As a QA person, you have 2 weeks to prepare before your team starts writing software. What do you do?

First thing I would do is go over the requirements and discuss with the team and the project manager to figure out a good testing plan. If there isn't a baseline of code started, then it is planning to find out how development is planning to work on the project. Could be taking time to create tests for code itself, or perhaps to test API routes. Either way, knowing how development plans to work on a project is useful and sometimes needed.

3. When is it appropriate to use automated testing? When is it appropriate to use manual testing?

Automated testing is ideal if you know exactly how a test should run. Manual tests are great to figure out how to properly create automated tests or to run through using a user interface where testing the behavior is difficult.

4. Your dev team has just modified an existing product by adding new features and refactoring the code for old features. The devs claim to have written unit tests; you’re in charge of integration testing. Dedicated teams are handling performance and security testing, so you don’t have to. As is always the case in the real world, you don’t have time to test everything. What factors do you think about as you decide where to focus your testing efforts? How do you decide what *not* to test?

There stands to be a level of trust that needs to come with working with others. As the last line of defense to stop a product from going out that isn't working correctly, I think there needs to be a system in place to run through the unit tests, compare that against code coverage, and focus on what isn't covered by the unit tests. I would put efforts into the integration tests I'm in charge of to make sure that they're modified in such a way that they will handle more code coverage and do what I can to test the new changes.

In a real-world situation, time doesn’t always allow it, but pushing back on a release is something that needs to be done at times. In Agile environments I've worked in before, this shouldn't happen too often because the QA and the devs are in communication for these things to hopefully not occur.