What types of testing are typically performed in Agile projects (e.g., unit testing, integration testing, acceptance testing), and how do you decide on the appropriate mix?

How do you incorporate test automation into Agile testing to achieve faster feedback and regression testing?

What metrics or indicators do you track to assess the quality and progress of testing in Agile projects?

Agile projects and methodologies generally use many forms of testing. In my experience, the amount of a given type of test will most likely vary from project to project, organization to organization, or some other combination of factors. The most common type I see is acceptance testing since it is what is written when doing Test Driven Development (TDD). Some of the bigger projects that I have worked on take advantage of this general concept, although we did not, necessarily, explicitly document the tests. As mentioned in “Introduction to software testing” (Ammann, P., & Offutt, J. 2016), a user story is usually converted to an acceptance test and then smaller tests are broken out in order to create smaller tests that will eventually satisfy the acceptance test. At the end of development, there are different types of tests - generally unit testing - that support the acceptance tests.

Test automation works well when using agile. I find this is best done through the use of CI/CD pipelines when doing automated deployments. This is helpful as a developer because you can get near instant feedback during development. I worked on one project that had a pretty thorough automated pipeline. Everytime a commit was pushed to the Gitlab repo, it would automatically run tests and build the software. This was helpful because I didn’t have to wait until I was done developing the use case before I got feedback on the accuracy of the code. I could also continue to develop while the regression testing was running so there was limited wasted time waiting for tests to complete.

A good metric to measure the quality of code that I have used in the past is test coverage. The higher that your percentage of coverage is, then the more of your code is tested. Higher test coverage means more code is verified for a given input. Having high test coverage doesn’t necessarily mean the tests are well designed, but it is a good indicator of how much testing there is overall. A good metric to understand the progress of a project is by comparing completed acceptance tests to the total number of user stories. Usually an acceptance corresponds to a user story, so if you have 7 completed acceptance tests and 10 total user stories, then you are about 70% through the project. This assumes that each acceptance test is weighted equally, but this can be accounted for with the use of story points.