Unit vs. Integration Tests

Unit Vs. Integration Test

One can categorize automated tests in many different ways.

The two most important categories to understand are:

- Unit tests, and
- Integration tests.

The difference comes down to **scope**. Are you testing a single component in isolation, or something more interconnected?

Unit Tests

A **unit test** covers a specific component (function, class, method, etc.) in isolation. You test for a specific response to a precise input.

By the strictest definition, a test is a unit test only if:

- It operates on one single component, AND
- It does not interact with the filesystem, AND
- It does not do any network operations, AND
- Does not otherwise cross the process boundary.

Integration tests

In contrast, an **integration test** checks how several components and/or services work together. How they integrate.

Component means "function or class" - same as before. (Can sometimes be a group of these connected as a tight unit; a module; or a few other outliers.)

Service basically means "something outside the process"... Outside the running program.

This can be:

- Accessing the file system
- Using a database server
- Making requests to a web service (JSON over HTTPS)

If it involves data passing the process boundary, that's a service (in the sense we're talking about here).

Is It a Unit Test?

The strictest definition of unit tests:

If it uses any operating system service, for file or network access, or anything else...

It's not a unit test.

Not everyone is this strict.

A common exception: test fixtures serialized in a file, read by the unit test before it starts making assertions.

Speed

"Quantitative differences in speed create qualitative differences in how you use the program."

(From some master programmer, in a book I read years ago.)

A guiding principle: if it's ambigious, then ask whether the test runs extremely fast.

If it does, it's more likely a unit test. If not, it's more likely an integration test.

The restrictions on a unit test are designed to make them FAST, so you can run them early and often.