Unit Testing

ESOF 411, Software Verification & Validation

Reasons for Unit Testing

Reasons for doing unit testing:

1. They help us think about our design in a way that we might not otherwise.
2. They lead to cleaner code because it's often easier to re-factor our code to be more testable then it is to write unit tests that exercise obfuscated or otherwise hard to access code.
3. They validate that our modules are behaving correctly.
4. They provide a safety net to ensure our existing modules continue to function properly as our application evolves.

https://hackerchick.com/your-unit-tests-should-mind-their-own/

Reasons for Unit Testing

Reasons for doing unit testing:

Tests are **working specifications** of the code.

They guide someone else on what your code is doing, and how to change it without breaking anything.

Comparison: fake vs mock vs stub

Comparison : fake , mock , stub

Fake – generic term that can describe either a stub or a mock object because they both look like the real object.

Mock object – checks an interaction (i.e. is asserted against)

Stub – all the fake objects that aren't mock objects

The Art of UNIT TESTING with Examples in C# by Roy Osherove

Dummy Objects

Dummy objects – objects that the System Under Test (SUT) depends on, but doesn't actually use.

Can be:

- an argument passed to another object
- an object returned by a second object and then passed to a third object

https://code.tutsplus.com/tutorials/all-about-mocking-with-phpunit--net-27252

Test Stub

Test stub – object to control the indirect input of the tested code.

Use when:

- Object asks another object for information and then does something with that data

Spies – more capable stubs

https://code.tutsplus.com/tutorials/all-about-mocking-with-phpunit--net-27252

Dependency Injection (DI)

Dependency injection – technique that injects an object into another objects, forcing it to use the injected object.

Commonly used:

- TDD

https://code.tutsplus.com/tutorials/all-about-mocking-with-phpunit--net-27252

Using Mocks

Test mock – capable of controlling indirect input and output, and has a mechanism for automatic assertion on expectations and results.

Avoid using mocks, if possible, but necessary when accomplishing the test requires actions from objects we don't have control over

The Art of UNIT TESTING with Examples in C# by Roy Osherove

Using Fakes

Examples:

- Check that the code responds correctly to results returned for a db call. Don' t have control over the db.
- Check that a legal transaction request returns true and an illegal one causes an exception, but don't have control over the authentication process.
- Getting input from a user . Can create an interface to which the user interactions adhere, and to which some test code adheres, and the test code can fake the user.

The Art of UNIT TESTING with Examples in C# by Roy Osherove

Guideline

"One assertion per test" is a good guideline, but when an action requires several steps or states, using more than one assertion in the same test is acceptable.

- Keeps your assertions about a single concept in one place
- Eliminates duplicate set up code

https://code.tutsplus.com/tutorials/all-about-mocking-with-phpunit--net-27252

Mocking bad and good

Mocking is bad – binds too much of the tests to the implementation.

> Anti-mockist encourage bottom-up development and design. Small component parts of the system created first and then combined into a harmonious structure.

versus

Mocking is good – helps us think more from the point of view of a user

> Mockists usually use a top-down approach to implementation and design. The components of the system appear and evolve based on the mocks created at one level higher.

https://code.tutsplus.com/tutorials/all-about-mocking-with-phpunit--net-27252