

Some thoughts on
“Test functionality,
not the implementation details”

Per-Olav Rusås

Pitfalls we want to avoid

“Every time I refactor, I need to change the tests”

“I can’t improve this code because it takes too much time to update the tests”

What do we want to achieve?

TDD – use tests to drive the development, focus on features

Test the features important to the user

What is an implementation detail?

Trying to define or find properties of an “implementation detail”

- The code we may want to refactor
- One of several potential ways of coding a feature
 - Some are better. This is why we want to refactor.
- Left to the developer(s) to decide
- Something the user doesn't need to know
- The non-public members of a class (or other encapsulation) are always implementation details

Avoid testing implementation details - Hide the details

Avoid leaking internal details

Too general structures:

- Don't let the internal data structures (like containers) in a class or other encapsulation be available outside the encapsulation.

Avoid revealing the construction of an object:

- Don't let any field variable, especially reference type fields, be available with getters or setters.

What about properties?

I think properties are good if they represent obvious states to be viewed or changed.

Avoid to test implementation details - Don't pair tests and methods

- Don't think like this:
 - “I need one test per method”
 - “I've written a method, now I need a test” (well, what about public APIs?)
- Better:
 - Make tests for a logical features.
Don't expect each feature to be represented by a single piece of code.
 - Use a test naming policy which says what a piece of software (an instance of a class, a web component, a web page, an application), should do.
 - Rethink what the *Unit* in ***Unit testing*** is

Exceptions to the rule?

Long data manipulating processes, with many defined (mathematical) intermediate steps.

If each step is a detail, still want to test the detail? Maybe.

A citation

I find the easiest way to avoid testing implementation details is to pretend I'm not a developer, and try to perform a specific task in my application just like a user would.

Max Rozen writing about testing React applications <https://testingreactjs.com/dont-test-implementation-details-react>

A thought

How does the TDD design and development process compare to
“Write tests. Not too many. Mostly integration”?

“Write test. Not too many. Mostly integration” is from a Tweet from Guillermo Rauch in 2016,
see also blog post by K.C.Dodds at <https://kentcdodds.com/blog/write-tests>