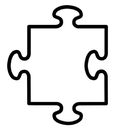
TDD vs BDD CHEATSHEET

"Testing the spark plug"

… is a test of a component in isolation. It's an example of a…

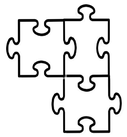
UNIT TEST



"Testing for a spark"

… is a test of a collection of components. It's an example of a…

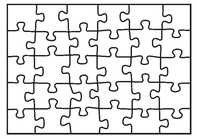
FUNCTIONAL TEST



"Trying to start a motorbike"

… is a System Test/Blackbox Test. It's also a…

BEHAVIOURAL TEST



Unit Tests are the building blocks of

TEST DRIVEN DEVELOPMENT

Behavioural Tests are the building blocks of

BEHAVIOUR DRIVEN DEVELOPMENT

Focus

TDD is an "inside-out" process. The focus is on **quality**.

BDD is an "outside-in" process. The focus is on **value**.

Reader/Writer

Who writes the tests? The tests are written at the same time as the code. So it is the Developer that writes the tests. **Who reads the tests?** Developers. And Testers. Its unlikely that anyone else will read them.

Who writes the tests? The plain English format means that the tests can be written by the person that understands the customer best: the Product Owner. **Who reads the tests?** Almost anyone: Developers, Testers, Stakeholders, Business Owner, Product Owner.

Specificity

When I test a spark plug in isolation - and the test fails - I know that spark plug is faulty. When a Unit Test fails, you know exactly what has failed.

When my motorbike doesn't start, I know that something is wrong. But I don't know what is wrong.

Speed

A Unit Test is a test of a component in isolation. This may require mocking of (potentially slow) external dependencies. As a result, Unit Tests are super-quick to run.

Behavioural Tests are tests of the system as a whole. The system must be "put into a known state" before each and every test. Not particularly difficult to do… but not quick either.

Coverage

The higher the level of code coverage, the better you sleep at night. But 100% code coverage is rare/difficult to achieve where Unit Test are concerned.

Code coverage percentages for Behavioural Tests tend to be high. 100% code coverage is not uncommon.

Maintainability

Any change to the functionality of a system will require a change to one or more Unit Tests. (Remember: the tests change first!)

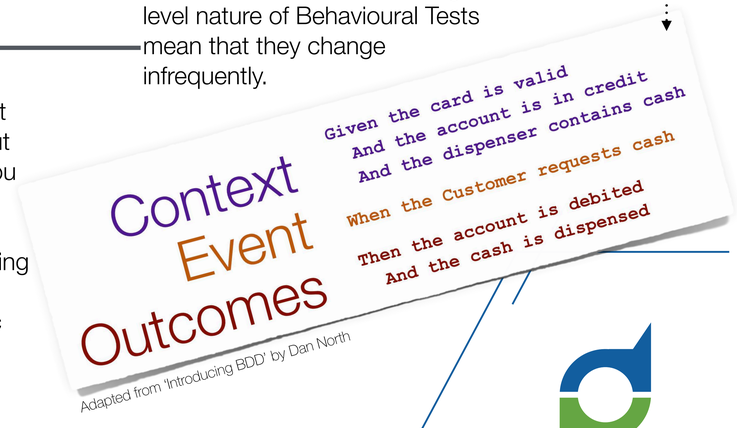
Not all functional changes impact the external behaviour; the high level nature of Behavioural Tests mean that they change infrequently.

Portability

Unit Tests are highly specific to the code that they cover; they're intertwined. Unit Tests are not portable.

Behavioural Tests are not coupled to code. Without changing a single test you could:

- rewrite your application in a different programming language;
- refactor your monolithic application into a set of microservices.



DEVELOPMENT THAT PAYS