Behaviour-Driven Development

A road to effective design and clean code

Dan North - ThoughtWorks

My name is Dan

I am a developer

I am a coach

I am your guide

Part 1: ineffective design and ugly code

Project failures - a field guide

The project comes in too late

or costs too much to finish

The application does the wrong thing

It is unstable in production

It breaks the rules

The code is impossible to work with

How we deliver software

Top-down

Bottom-up

Why do we do this?

Part 2: effective design and clean code

If we could deliver better

Only focus on high-value features

Flatten the cost of change

of anything, at any stage

Prioritise often, change often

Adapt to feedback

Learn!

What we would need

Streaming requirements

Evolving design

Code we can change

Frequent code integration

Run all the regression tests often

Part 3: Getting there with BDD

A definition of BDD

"Behaviour-driven development is about implementing an application by describing it from the point of view of its stakeholders"

- Me :)

BDD is derivative

"Second generation" agile methodology

- XP, especially TDD and CI
- Domain-Driven Design
- Acceptance Test-Driven Planning
- Neurolinguistic Programming (NLP)
- Systems Thinking

What makes BDD?

Getting the words right

Enough is enough

...agree on "Done"

Outside-in

Interactions

People over Process!

"Getting the words right"

"When I use a word", said Humpty Dumpty in a rather scornful tone, "it means just what I want it to mean, neither more nor less"

Lewis Carroll - *Through the Looking Glass*

“Getting the words right”

Model your domain

...and identify your core domain

Create a shared language

...and make it ubiquitous

Determine its bounded context

...and think about what happens at the edges

“Enough up-front thinking”

Identify the desired outcomes

Do *enough* to feel *safe* to estimate

...and keep a note of your assumptions

Then “blink estimate” - with people you trust

...because anything else is false confidence

Estimation is fractal - don’t misunderestimate!

A story is a unit of delivery

Story 28 - View patient details
As an Anaesthetist
I want to view the Patient's surgical history
So that I can choose the most suitable gas

Try to focus on the value

Story 28 - View patient details

In order to choose the most suitable gas

an Anaesthetist

wants to view the Patient's surgical history

Try to focus on the value

Story 29 - Log patient details
In order to choose the most suitable gas
an Anaesthetist
wants other Anaesthetists to log the
Patient's surgery details for later retrieval

Agree on "Done"

Define acceptance criteria as scenarios

Scenario: existing patient with history
Given we have a patient on file
And the patient has had previous surgery
When I request the Patient's history
Then I should see all the previous treatments

Automate the scenarios

Each step corresponds to running code

Given we have a patient on file

In Ruby:

```
Given “we have a patient on file” do
  @patient = Patient.create
end
```

In Java:

```
@Given(“we have a patient on file”)
public void createPatient() {
  patient = patientFactory.create();
}
```

Code-by-Example to implement

Also known as TDD

Start at the edges with what you know

Implement outermost objects and operations

Discover collaborators, working inwards

and mock them out for now

Repeat until "Done"

Then bring it all together

Examples become code tests

…and documentation

Scenarios become acceptance tests

...which become regression tests

Automation is key

Inside-out - an example

```
Map<int, Map<int, int>> portfoliosByTraderId;

if (portfolioIdsByTraderId.get(trader.getId())
  .containsKey(portfolio.getId())) {...}
```

Becomes:

```
if (trader.canView(portfolio)) {...}
```

The team

The stakeholders

The BAs

The QAs

The developers

The project manager

The destination

Effective Design and Clean Code

...has tangible stakeholder value

...is delivered on time, incrementally

...is easy to deploy and manage

...is robust in production

...is easy to understand and communicate

BDD is a step in the right direction

Thank you

Any questions?

dan.north@thoughtworks.com

http://dannorth.net

http://lizkeogh.com

http://jbehave.org

http://rspec.info

Bibliography

Extreme Programming explained (2nd edition) - Kent Beck

Domain-Driven Design - Eric Evans

The Art of Systems Thinking

and

The Way of NLP - Joseph O'Connor