DDD and BDD

Dan North

ThoughtWorks

BDD and DDD

Dan North

ThoughtWorks

What is Domain Driven Design?

"It's about focusing on the domain and letting it affect the software very much"

- Jimmy Nilsson (ADDDP)

Identify the “Core Domain”

The thing the stakeholders care most about

A shipping company moves items around. What is their core domain?

- Logistics
- Routing
- Knowing where things are

The core domain tells us where the most useful conversations will be

- *...which are the richest seams for knowledge crunching*

Evolve a common understanding

Model the domain with the stakeholders

Establish a shared understanding
- ...which leads to a shared language

Drive this language into everything
- Classes, properties, variables, method names
- Object-orientation makes this easier

Evolve a Ubiquitous Language

Strategic DDD

How to scale domain-driven design

- to multiple teams?
- to a system-of-systems?

Ubiquitous Language isn’t so ubiquitous

It’s only valid within a Bounded Context

What is Behaviour-Driven Development?

"It's about focusing on the behaviour of an application from the point of view of its stakeholders"

- *Me :)*

Outside-in development

Express a requirement as a Story

Story 28 - View patient details

In order to choose the most suitable gas
an Anaesthetist
wants to view the Patient's surgical history

Agree on “Done”

Define acceptance criteria as Scenarios made up of Steps

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 surgical history
Then I see all the previous treatments

Automate the scenarios

Each step becomes running code

In Ruby:

```
Given “we have a patient on file” do
    # ...
end
```

In Java:

```
@Given(“we have a patient on file”)
public void createPatientOnFile() {
    // ...
}
```

Code-by-Example to implement

Also known as TDD

Start at the edges, with what you know

Implement outermost objects and services

Discover collaborators, working inwards
    and mock them out for now

Repeat until "Done"

Then bring it all together

Examples become unit tests

and documentation

Scenarios become acceptance tests

Acceptance tests become regression tests

Recap

DDD is about

- evolving a shared model of the domain
- letting the domain model drive the design

BDD is about

- establishing a shared understanding of “done”
- working from the outside in until you get there

Remind me – what’s a domain model?

- It is your stakeholder’s mental model
  - “The map is not the territory”

- *Ah! Got it. Right. Thanks.*
- *Um, so what’s a stakeholder?*

A stakeholder is *anyone who cares*

about how much the application costs

about what it does

about whether it is secure

about whether it hammers the network

about whether it complies with the law

about how easy it is to deploy and diagnose

about how well it is written and architected

and how easy it is to change

We have lots of stakeholders

So we have lots of domains

…each with their own language

…all speaking at once!

That’s a recipe for disaster, surely?

Actually it happens all the time

You want to retrieve patient records

- You're writing in Java, using Hibernate

so you define

```
class HibernatePatientRecordRepository {
```

What if your IDE did domain-specific fonts?

So how does DDD relate to BDD?

DDD is about how you explore the *domain models* your stakeholders use

- and articulate problems in that domain

BDD is about the conversations you have *in the ubiquitous languages* to produce software

And it's iterative

- In the course of different conversations the domain models emerge

In other words: DDD *enables* BDD

The domain drives your design

hence Domain-Driven *Design*

The behaviour drives what you develop

hence Behaviour-Driven *Development*

BDD helps structure the conversations for DDD

Any questions?

DDD

Domain-Driven Design by Eric Evans
Applying Domain-Driven Design and Patterns by Jimmy Nilsson

BDD

http://behaviour-driven.org
http://jbehave.org
http://rspec.info

Me

http://dannorth.net
dan.north@thoughtworks.com