BDD Primer

ONE DAY OVERVIEW

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

Course Introductions

Name?

Role?

Background?

Primary Objective in Taking Course?

Recent Book or Movie?

Questions

Daily schedule

- Time frame?
- Breaks?
- Lunch?

Any other questions before we get started?

BDD Primer

GOAL: ESTABLISH A BASELINE OF KNOWLEDGE SURROUNDING BEHAVIOR-DRIVEN DEVELOPMENT

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

BDD Origins

Test Driven Development (TDD)
- *Techniques*
- *Principles*

Domain Driven Design (DDD)
- *Concepts*

Object Oriented Analysis & Design (OOAD)
- *Concepts*

Natural Language Constructs

Domain Specific Language (DSL)

Shared Process

Shared Tools

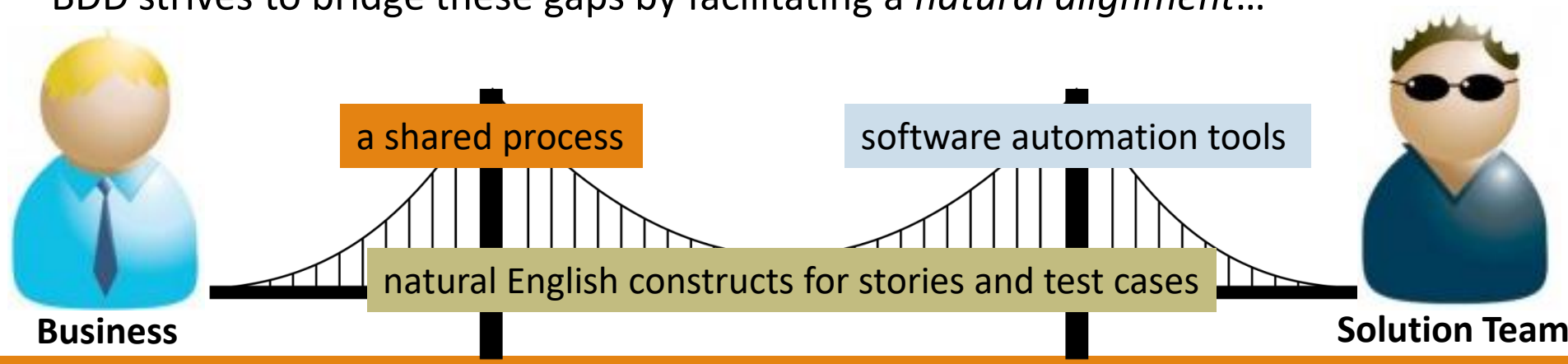
Automation

The Digital Disconnect

Software development projects experience common challenges:

- The business struggles to completely define desired outcomes
- The development team has a flawed understanding of what needs to be built
- The business is unaware of the technical challenges incurred by the requirements they identify
- The translation of business needs into technical software loses much in translation and ultimately leads to gaps in expectation and delivery

BDD strives to bridge these gaps by facilitating a *natural alignment*…



What's that spell?

B - *Behavior*, the desired *behavior* of the software to be developed.

D – *Driven*, the project is oriented around and aligned with the desired behavior

D – *Development*, software is mapped directly to the declaration of behavior, supported by automated tooling.

In short, BDD is an approach for building a shared understanding regarding what software should be built by working through specific examples.

How's it work?

BDD is pretty simple:

- **describe** what you want the system to do by talking through example behavior
- work from the "outside-in" to **implement** those behaviors using the examples to validate what you are creating

Bring in different perspectives for validation:

- The customer (or Product Owner) describes what they want.
- Team members ask questions to flesh out "just enough" in order to implement the behavior.
- Business analysts and quality assurance specialists can also help to identify and elaborate specific details of various scenarios.

Principles

Business Traceability – All development work can be traced back directly to business objectives.

Needs Focused – Software development meets user needs and ultimately supports current and future business objectives.

Efficient Prioritization – Business-critical features are delivered first.

Shared Understanding – All parties (business and technical) have a shared understanding of the project and thorough visibility into the project's progression.

Imagine if you could choose…

A logical, thorough, content-heavy training course on a new topic

- Detailed definition of terms
- Well-structured thought process
- Extensive historical context
- Comprehensive philosophy regarding the motivations and drivers for this new subject
- Itemized implementation steps
- Best practices and recommendations

A pragmatic, example-rich course that aimed to SHOW you as much as TEACH you the new topic

- Establish a baseline of terminology, principles, and process
- Highlight a few benefits and motivations behind the topic's core philosophy
- Practical, real-world examples
- Collaborative exercises
- More practice, less theory

Which type of course would you prefer to participate in?

You can **always** learn by *doing*. You can **never** do by *learning*.

Simple Example

Context: Your team is working on the quote functionality within the rating engine for an insurance business.

Task: Implement a risk premium factor to quotes based upon a prospect's credit score.

Business Rules:

- Quote premium for sub-prime (less than 580 credit score) is 25%
- Quote premium for prime (more than 580 credit score) is 0%
- Quote premium for super-prime (more than 720 credit score) is -10%

Can you use this information to deliver a working feature?

How BDD Helps

How do we address the ambiguity left by the initial scenario setup? Through concrete, behavior-driven examples.

Given a credit score of 570, when a quote is run, then the premium is 25%

Given a credit score of 580, when a quote is run, then the premium is 0%

Given a credit score of 719, when a quote is run, then the premium is 0%

Given a credit score of 720, when a quote is run, then the premium is -10%

If you take these examples, convert them into tests, and use those tests to drive (and scope) your development activity, then you've doing BDD!

Lost in Translation

Most any project (and software development in particular) has the potential to fall prey to miscommunication due to ambiguous terms.

- **Example 1** – Writing software for a bank and being asked to implement basic account management functionality. *Great! Umm… did you want 'user accounts' or 'credit/debit accounts' to be managed?*
- **Example 2** – Delivering a *policy validation* function for an insurance business. *Terrific! So…you were referring to 'security policies,' right?*

There is an inherent "cost of translation" (also known as "the feedback delay") which can range from weeks (using Agile) to months/years (using Waterfall) as we deliver working software and discover the extent of our miscommunication.

BDD aims to close the size of the feedback loop through **ubiquitous language** (a concept borrowed from domain-driven design).

Conventional BDD Format

Given-When-Then

"Given" – describes the initial context for the example

"When" – describes the triggering event performed by an actor

"Then" – describes the expected outcome (which can be validated)

This aims to mirror the natural way people describe expected behavior. When a stakeholder is asked to provide examples of what they expect for a particular capability, the naturally say something like: *"when I do X, then Y should happen."*

BDD Story Format

CLASSIC AGILE STORY (USER FOCUS)	BDD STORY (VALUE FOCUS)
As a [type of user]	In order to [achieve some value]
I want [some goal]	As a [type of user]
So that [some reason]	I want [some goal]

BDD shifts the emphasis away from performing *functions* required by users and toward achieving *value* required by the business.

Complete Story – 1/2

Story: *New cars receive the same coverage as old cars*

In order to easily manage my policy

As a member

I want to extend the same coverage to a new car when I add it to my policy

Example: *Coverage limits should be extended to a new car*

Given a member has an existing policy

And the policy has at least one covered car

And the same coverage limits can be extended to the newly added car

When she adds a car to the policy

Then the same coverage limits should be extended to the new car

Complete Story – 2/2

Example: *Policy add-ons should be extended to the new car*

Given a member has an existing policy

And the policy has at least one covered car

And the coverage includes at least one add-on

When she adds a car to the policy

Then the add-ons should be extended to the new car

And the quote should be updated to reflect the cost of the add-ons

Ubiquitous Language

The story identifies a stakeholder, business outcome, and value.

Each example defines a precondition, trigger, and expected outcome.

Some keywords are present (Given-When-Then).

Some ambiguous, domain-specific terms exist as well. These would ultimately need to be identified, defined, and agreed-upon in collaboration between development team and business.

- Member
- Coverage
- Policy
- Quote
- Add-on
- Etc…

BDD Structure

Title: *The story should have a clear, explicit title.*

Narrative: *A short, introductory section that specifies three things…*

- who (which business or project role) is the driver or primary stakeholder of the story (the actor who derives business benefit from the story)
- what effect the stakeholder wants the story to have
- what business value the stakeholder will derive from this effect

Acceptance criteria or scenarios: *a description of each specific case of the narrative. Such a scenario has the following structure…*

- specify the initial condition that is assumed to be true at the beginning of the scenario. This may consist of a single clause, or several
- state which event trigger initiates the scenario
- state the expected outcome, in one or more clauses

Group Exercise – BDD Basics

1. Organize into teams of 2-3.
2. Define the scope of your project (choose a specific system, process, or feature-set you are responsible for developing).
3. Craft two user stories which could realistically fit within that scope.
   - **In order to**… (WHY)
   - **As a**… (WHO)
   - **I want to**… (WHAT)
4. Select one story and create one behavior-driven scenario:
   - **Given**… (describe the initial condition)
   - **When**…(triggering event performed by actor)
   - **Then**…(expected outcome)

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

Two Main Parts

The BDD approach can largely be divided into two main parts.

First, the practice of using examples written in *ubiquitous language* to illustrate behaviors (how users will interact with the product).

Second, the practice of using those examples as the basis of *automated tests*.

This approach ensures alignment with business needs and user expectations, with a direct traceability to system behavior as defined by the business throughout the project lifetime.

TDD Recap

Typically broken up into five different stages:

1. First the developer writes some tests.
2. Next the developer runs those tests and they fail (by design), because none of those features are actually implemented yet.
3. Next the developer actually implements those tests in code, but *just enough* to pass the tests, thus validating the tests.
4. Now the developer fleshes the implementation out to be complete, using the tests to ensure that the code remains on track.
5. Finally the developer refactors their code, adds comments, cleans it up, and is able to do so with confidence that the tests provide an objective regression meter to identify if anything breaks in the process.

From TDD to BDD

BDD borrows much of its initial philosophy from TDD and utilizes a test-centric approach as an essential thread throughout the methodology. However, it moves beyond TDD to rethink the testing approach.

Classic TDD

For each unit of software…

- define a test set for the unit first
- then implement the unit
- finally verify that the implementation of the unit makes the tests succeed

Testing in BDD

Behavior-driven development specifies that tests of any unit of software should be specified in terms of the ***desired behavior of the unit***.

What is that “desired behavior” in question? It’s the business requirements which spell out a valuable business outcome.

Within BDD practice, this is referred to as BDD being an "outside-in" activity.

Re-framing our Tests

BDD aims to rethink how we define our unit tests and acceptance tests.

Unit Tests

- BDD suggests that unit test names be whole sentences starting with a conditional verb ("should" in English, for example) and should be written in order of business value.

Acceptance Tests

- Conventionally, BDD practitioners write acceptance tests using the standard agile framework of a User story (but with a twist): “In order to [benefit] As a [role] I want [feature]".
- *Acceptance criteria* is most often written in terms of scenarios and implemented as classes: Given [initial context], when [event occurs], then [ensure some outcomes].

The ability to read tests like a sentence is an important cognitive shift which enables you to *naturally* write better and more comprehensive tests.

Show me!

Instructor Demo #1 – Flipper Demo

Instructor Demo #2 – Breaking the Demo

Reflecting on TDD / BDD

What aspects of test-driven appeal to you?

Where do you see potential challenges or pitfalls?

Any personal experiences with a test-centric approach?

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

JBehave

A Java-based BDD framework developed by Dan North.

There are two varieties:

- **Core** – Includes all the essential JBehave libraries, command-line runner, and hooks for various tools (Eclipse, Maven, Ant, etc.)
- **Web** – An extension of core, providing support for web-related access or functionality. It includes a web runner and also a selenium integration module for driving verification of web application behavior

Unit Structure

Story Class – describe the expected behavior using natural language

Object Class – describe an object which implements the desired behavior

Unit Test – bind the story elements to the object elements to facilitate traceability, scoping, and automated testing

Story Structure

```
Sample story

Narrative:
In order to communicate effectively to the business some functionality
As a development team
I want to use Behaviour-Driven Development

Scenario:  A scenario is a collection of executable steps of different type
Given step represents a precondition to an event
When step represents the occurrence of the event
Then step represents the outcome of the event

Scenario:  Another scenario exploring different combination of events
Given a precondition
When a negative event occurs
Then a the outcome should be captured
```

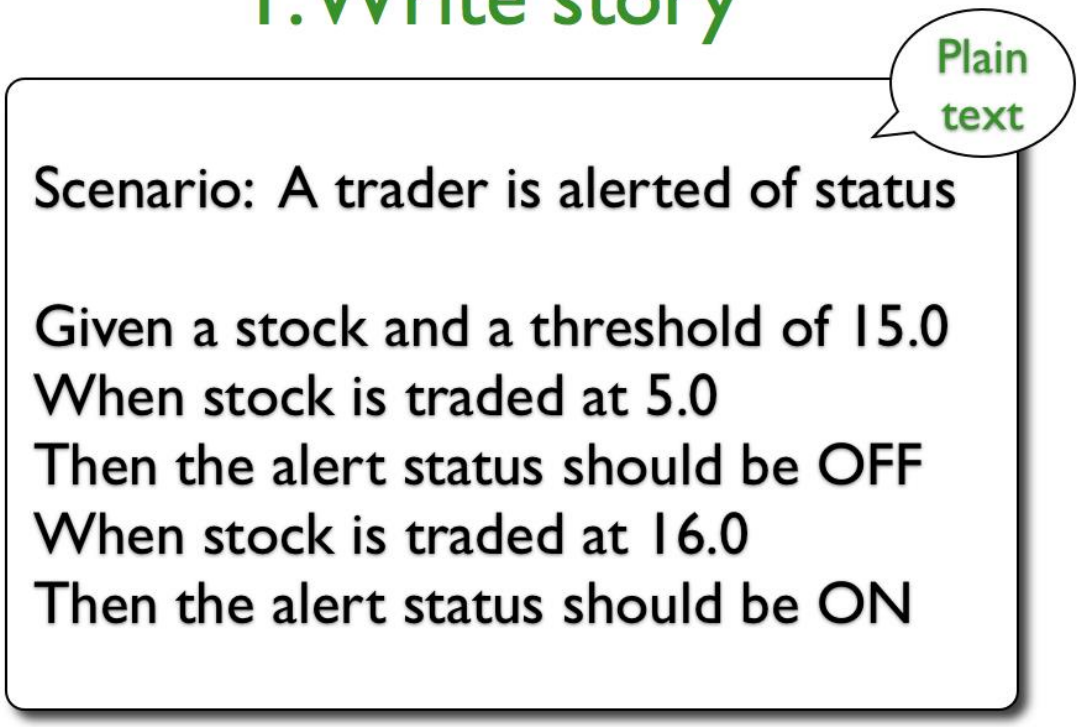
Group Exercise – Refactoring

1. Get back into your teams.
2. Refactor your previous BDD scenario (if needed) now that you have a stronger understanding of the BDD structure.
3. Take your other story and elaborate two BDD scenarios (refactor the story if needed).

THE TRADER EXAMPLE

The Trader Example – 1/5

1. Write story



The Trader Example – 2/5

2. Map steps to Java

POJO

```
public class TraderSteps {
    private TradingService service;   // Injected
    private Stock stock; // Created

    @Given("a stock and a threshold of $threshold")
    public void aStock(double threshold) {
        stock = service.newStock("STK", threshold);
    }
    @When("the stock is traded at price $price")
    public void theStockIsTraded(double price) {
        stock.tradeAt(price);
    }
    @Then("the alert status is $status")
    public void theAlertStatusIs(String status) {
        assertThat(stock.getStatus().name(), equalTo(status));
    }
}
```

The Trader Example – 3/5

3. Configure Stories

Only once

```
public class TraderStories extends JUnitStories {

    public Configuration configuration() {
        return new MostUsefulConfiguration()
            .useStoryLoader(new LoadFromClasspath(this.getClass()))
            .useStoryReporterBuilder(new StoryReporterBuilder()
                .withCodeLocation(codeLocationFromClass(this.getClass()))
                .withFormats(CONSOLE, TXT, HTML, XML));
    }

    public List<CandidateSteps> candidateSteps() {
        return new InstanceStepsFactory(configuration(),
             new TraderSteps(new TradingService())).createCandidateSteps();
    }

    protected List<String> storyPaths() {
        return new StoryFinder().findPaths(codeLocationFromClass(this.getClass()),
"**/*.story");
    }
}
```

The Trader Example – 4/5

The Trader Example – 5/5

Demo

Let’s look at the Trader code from JBehave…

NOTE: Pay special attention to handling of variables, tables, etc.

Group Exercise – Software Binding

1. Get back into your teams.
2. Take one BDD scenario and use pseudo code or annotated comments to create a wireframe of what the corresponding object class and testing class would look like for this scenario.
   - Identify a method for each element of the story (GIVEN-WHEN-THEN) that would need to exist within the testing class.
   - For each identified method in the testing class, briefly describe how you would perform that step.
   - For each THEN clause, make sure you have a testable outcome to measure against.
   - Define the basic structure of the object class (fields and methods).

Group Exercise

Take one BDD scenario and use pseudo code or annotated comments to create a wireframe of the corresponding object class and testing class.

- Identify a method for each element of the story (GIVEN-WHEN-THEN) that would need to exist within the testing class.

```
@Given("a customer exists")
customerExistsFunction()
```

- For each identified method in the testing class, briefly describe how you would perform that step.

```
//Here we will create a Customer object
```

- For each THEN clause, make sure you have a testable outcome to measure against.

```
@Then("the account balance will be 100")
checkAccountBalance() {
     AssertEquals( 100, account.getBalance() );
}
```

- Define the basic structure of the object class (fields and methods).

```
Customer { name, account balance, account number }
Bank { routing number, account number, customer list}
```

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

Outside-in Development

Focus – Satisfy the needs of stakeholders.

Premise – In order to create successful software, you must have a clear understanding of the goals and motivations of your stakeholders.

Aim – Produce software that is highly consumable, meeting or even exceeding the needs of your stakeholders.

Four categories of stakeholder for outside-in projects

- **Principals** – people with the money
- **End users** – people who interact with your product
- **Partners** – people who make your product work in real life (ops and SI teams)
- **Insiders** – people within your company who impact how your team develops

Importance of Goal Alignment

If you break it down, **software development is merely a way of supporting human behaviors** by automating them, replacing them, or creating them so a human doesn't have to perform that activity. Moreover, **every change in behavior should have a tangible impact by helping the organization to achieve one or more of its goals** (e.g. increasing revenue, reducing waste, improving quality, reducing time-to-market, etc.).

Through BDD, projects deliver feature sets. These features should enable users to be more efficient. Perhaps by automating a manual process or performing a complex calculation.

If we are going to satisfy the business requirements, we will need to understand precisely what the software needs to do from a business point of view. **Whatever that business outcome looks like should be the driving force behind your project**. Every story and scenario should tie back to one or more business goals.

Good Goals vs Bad Goals

A good business goal is specific, measurable, achievable, realistic and time-bound (SMART).

Bad example: We are going to generate more revenue by working faster

Good example: We are planning to achieve a 15% increase in revenue by reducing our average time-to-market for new capabilities from 3 months to 6 weeks

Having a SMART goal provides focus, measurable impact, and a direct linkage back to what drives the business.

How do we achieve our goals?

Once you have a defined goal, you're ready to plan how you will go about achieving that target state.

Classic answer? *"We can achieve our goals by delivering new features on our applications."* What could possibly go wrong? As it turns out…a lot.

As Dan North states: *"Delivering features for the sake of delivering features is a common problem in software development, and is the reason behind many of the most infamous project failures. Instead of trying to deliver as many features as possible, you need to identify how the features will support the user's behaviours or the business."*

Since the hundreds of potential features are likely to require different levels of time, energy, and skill, it is important to systematically navigate through you growing backlog. An increasingly popular strategy for this is known as **Impact Mapping**.

Impact mapping – The ‘HOW’

Impact Mapping is a technique that helps you to outline all of the alternative ways to reach your decided goal and works by analyzing users’ behavior. Since software cannot achieve business goals independently; the only way software can get us closer to the business goal is by supporting particular human behaviors.

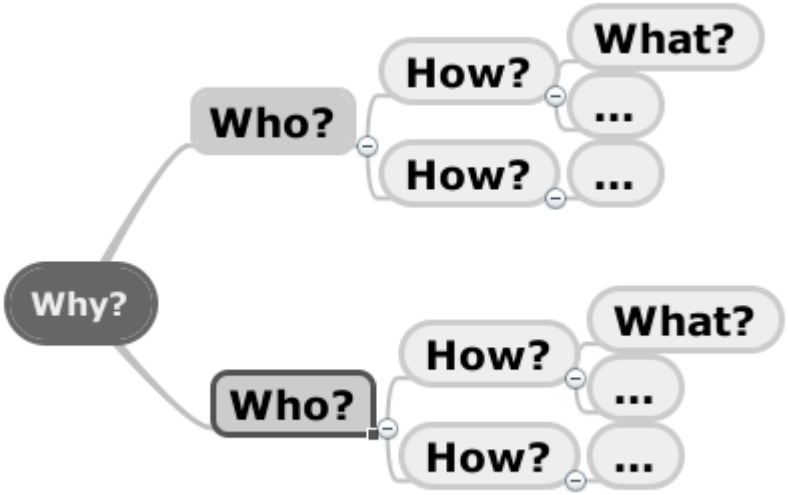
For example, an online shop doesn’t bring you money on its own - people bring you money by purchasing products from it. Similarly, better system logging doesn’t improve performance of your website on its own - technical specialists using it do. Looking at software through a prism of those who use it is a very valuable way to plan and direct the project delivery.

Impact Mapping is a simple technique built on top of conventional mind mapping, with a bit of twist.

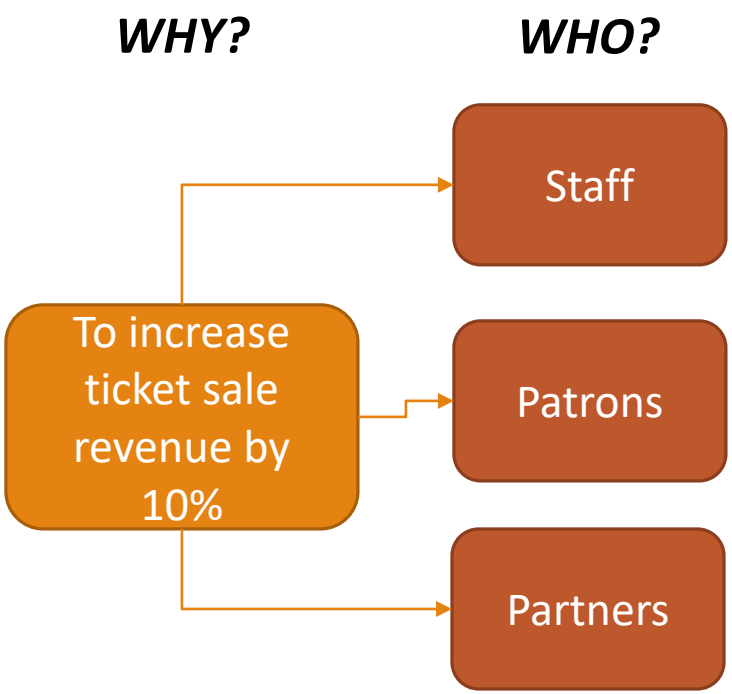
An impact map includes four distinct levels of detail:

- Business goal [WHY]
- One or more actors [WHO]
- One or more impacts [HOW]
- Ways to support/prevent these impacts [WHAT]

Impact Map Structure



Impact Example – 1/2

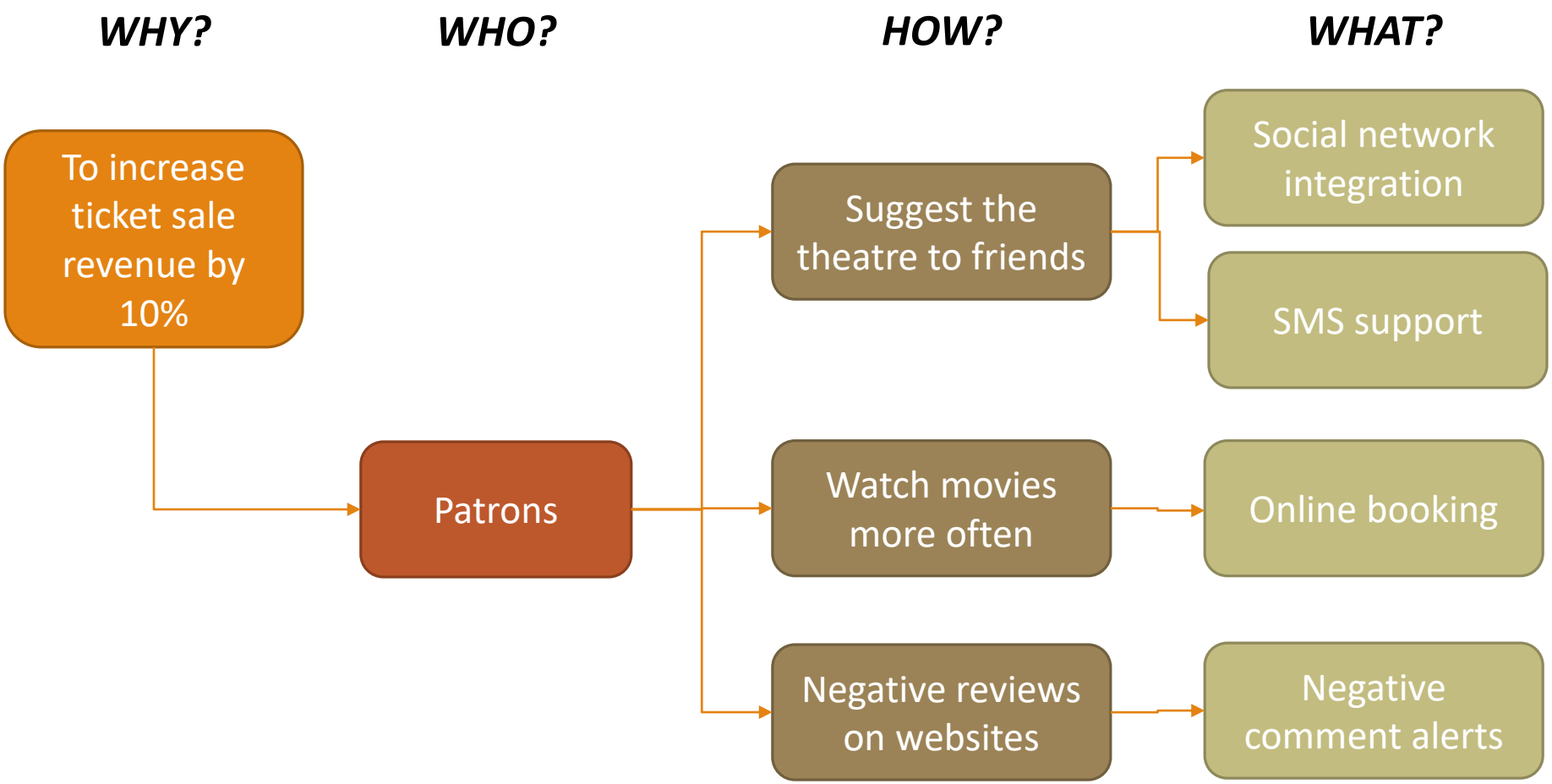


HOW?

*How can this actor <u>positively</u> **impact** my goal?*

*How can this actor <u>negatively</u> **impact** my goal?*

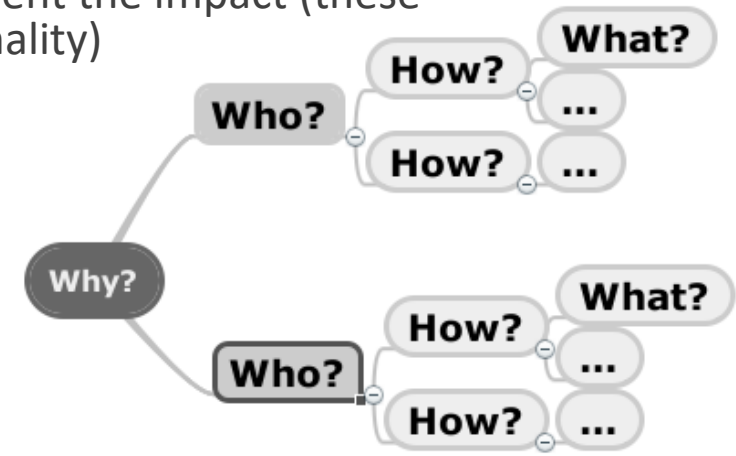
Impact Example – 2/2



OPTIONAL (if time allows)

Group Exercise – Goal Focus

1. Get back into your teams.
2. Define a SMART goal within the context of your project scope.
   Specific, Measureable, Achievable, Realistic, Time-bound
3. Build out an Impact Map
   - Select at least two actors
   - Identify 1-3 impacts for each actor
   - Capture at least one way to support / prevent the impact (these ultimately map to epics/stories of functionality)



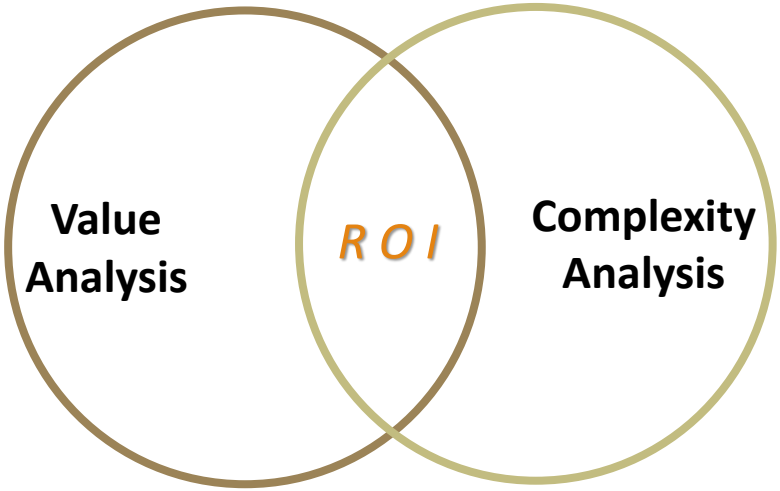
Food for thought – *How does this change the way you think about identifying stories and scenarios?*

Value and Complexity Analysis

To do BDD right, requires a lot of collaboration. Consequently, it is essential you clearly distinguish and agree upon priorities.

In BDD, it is recommended you achieve this through a value analysis and complexity analysis.

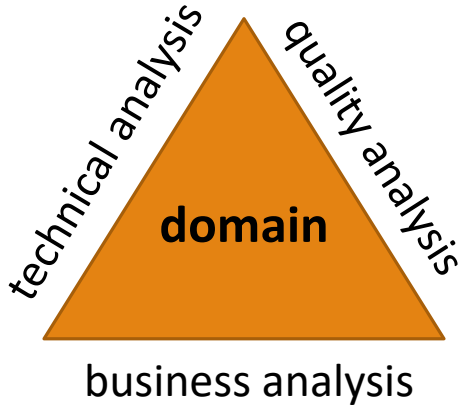
- **Value analysis** – quickly identify low-cost, high-value features in the backlog
- **Complexity analysis** – helps us to choose the right development and collaboration approach to the project as a whole, as well as each individual feature



The Three Amigos

BDD is intended to be a **highly collaborative process**. Talking in examples requires multiple inputs and varied perspectives in order to be efficient.

In order to effectively strike the right balance of breadth and depth, business and technical elements, you need three fundamental perspectives.



Why do you think it is so important to have all three of these roles at the table?

Course Agenda

Course Overview

BDD Basics and Principles

From TDD to BDD

Seeing it in Action

Domains and the Role of Business

Steps to Incorporate BDD into a Project

Ways to include BDD

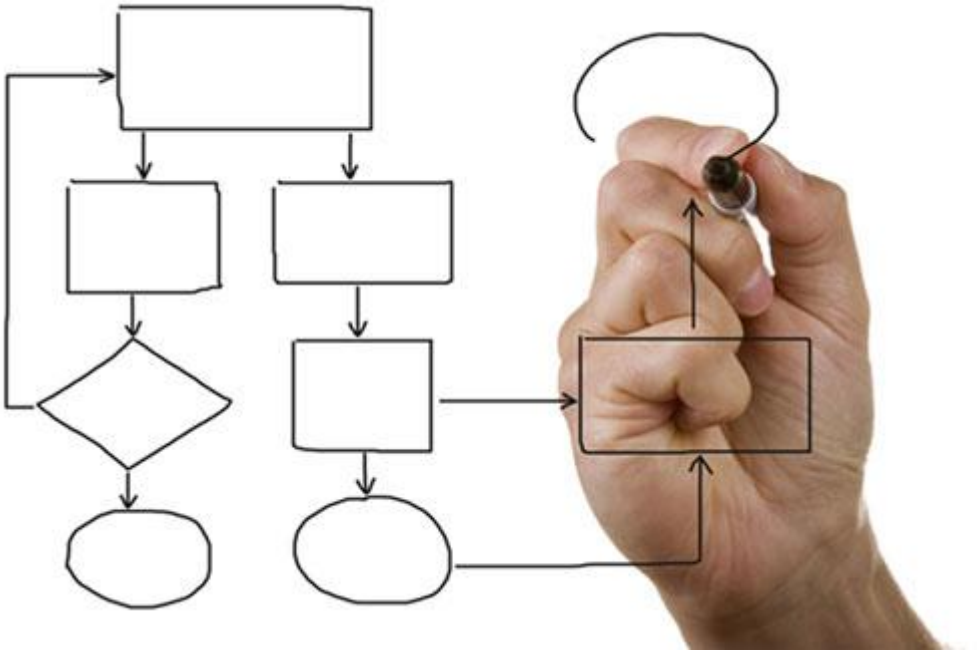
Incorporate elements

- BDD as a way to drive a shared understanding
- TDD by itself
- BDD with fully automated tooling
- With / without the goal-focus
- With / without domain modeling
- With / without impact analysis

Remember, it’s a spectrum!

What does your current / evolving process entail?

Let’s talk…



How do you quantify BDD benefits?

In order to define an ROI for BDD adoption, you need a set of baseline metrics to compare against

- Cost of development
- Cost of maintenance
- Time to market
- Code quality

Without baseline data, you can't craft a case for the degree to which BDD will improve the status quo.

In the absence of baseline data, the value proposition must be defined in terms of business/IT alignment, story prioritization, and reducing the feedback loop between business and IT.

Next Steps

What might make sense as next steps to embracing BDD?

- One or two techniques you incorporate on an up-coming or current project?
- Full-blown BDD pilot with a right-sized project
- Spike to explore BDD tooling options
- Deeper hands-on learning (i.e. “BDD Workshop”)

course retrospective

Today's Goal

establish a baseline of knowledge surrounding behavior-driven development

Do you feel like we've accomplished that goal today?

What aspect of today's training most resonated with you?