Error proof your mobile app

Alexandra Marin @ I T.A.K.E. Unconference 2016

Hello! I'm a...

Mobile C# developer

Built Xamarin apps for
smartwatches, phones & tablets

Cat video enthusiast

—

Behavior driven development is a philosophy of **Outside-in Development** in which the application code is written after its requirements have been defined

A COLLABORATION TOOL

Sit down with your customer and work together!

Conversations tend to uncover assumptions between you and your user's desired result.

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

A COLLABORATION TOOL

Get to a common ground by discussing examples and using a language everyone can understand.

Also do your future self & team members a favor!

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

RESULTS

Documented features tied directly to your development and testing process.

Concrete, executable, easy-to-repeat behavior embodied in automated tests.

BDD moves the focus on the higher level purposes of the system & proving the product.

(Don't worry, unit tests still happen, but only later in the process)

FIND OUT WHAT THE USER WANTS

Involve your users to ensure that you are developing functionality they're actually going to want.

EVERYBODY CONTRIBUTES IN DEFINING THE SPECS

Don't hand your dev a sheet full of predefined requirements.

Get them together with people from the business side.

KEEP IT IMPLEMENTATION FREE

- ✔ Don't worry about the shape of the APIs just yet
- ✔ Focus on behavior and specifications
- ✔ Illustrate business rules with examples (and not just the UI)

FUNCTIONAL SPECS

- ✔ Describes how the system should behave
- ✔ Written from the user's point of view, in a step-by-step form
- ✔ Serves as an entry point into the classic TDD cycle

—

Acceptance tests are executable specifications written in a domain specific language that describe how a user will interact with the app.

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

User acceptance testing

involves writing **tests to mirror the user stories** created by and for the software's customer during development.

ACCEPTANCE TESTING

Automated tests are written in the language of the business, all while maintaining a connection to your implemented system.

If the tests pass, it means the software meets the customer's requirements and **the stories can be considered complete**.

ACCEPTANCE TESTING

- ✔ **involves performing tests on the full system** to see whether the application's functionality satisfies the specification

- ✔ have the **requirements nailed before you begin coding**

THE GOOD

- ✔ **find bugs that unit tests can't** such as wiring bugs and environment bugs
- ✔ tests are **described in plain English**
- ✔ ensures the software, as a whole, is **feature complete**

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

THE BAD

- ✔ you've moved another level up the testing pyramid
- ✔ **tests touch more code**
- ✔ tracking down a failure can be tricky

—

Don’t test business logic pieces through the UI.

Test instead flows - they’re examples of how you’re actually going to use the app.

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

UI TESTS: What can the user see?

- ✔ UI acceptance testing
- ✔ How to simulate UI interactions
- ✔ Provide some example of how your code is valuable to the user (have the rest of the tests at the class level)

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

WHY ARE UI TESTS IMPORTANT?

- ✔ UI and requirement go hand in hand
- ✔ Product is also UX, not just business logic
- ✔ Device fragmentation: do you support the specific UI of the app?
- ✔ It works on my device!

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

UI TESTS vs UNIT TESTS

- ✔ UI tests don't replace unit test, but round up the test coverage
- ✔ They run an iOS / Android application on a simulator/emulator or on a physical device
- ✔ Don't care about system internals
- ✔ Use an environment that is closest to production

UI TESTS: When?

- ✔ Continuous delivery
- ✔ Automation of UI tests
- ✔ Repeatable regression tests
- ✔ Can drive UI development as well as the application logic
- ✔ Fixing bugs

THE TOOLS

CALABASH

- ✔Enables automatic UI interactions within an application

- ✔Examples: pressing buttons, entering text, making swipe gestures

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

XAMARIN.UITEST

- ✔ Automated UI Acceptance Testing framework
- ✔ Validates functionality of **iOS and Android Apps**
- ✔ Based on Calabash

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

XAMARIN.UITEST

- ✔ Write tests in **C#**
- ✔ Execute them with NUnit
- ✔ Tests follow the **Arrange-Act-Assert** pattern

XAMARIN.UITEST: Setup

- ✔ **Build the AppBundle / APK before running the tests**
- ✔ Add to your solution an NUnit test project and install Xamarin.UITest from NuGet
- ✔ **Interact with objects on the screen** independent of screen size, orientation and layout

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

XAMARIN.UITEST: Internals

- ✔ Interactions within the mobile application occur through an instance of an **IApp** interface

- ✔ New it up before every test to prevent state from one test affecting the others

XAMARIN.UITEST: REPL

- ✔ Interact with a screen **while the app is running**
- ✔ Explore the app screens
- ✔ **Creating the queries on the fly**
- ✔ Prototype tests interactively

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

XAMARIN.UITEST: Querying the UI

✔ Locate view on screen by: id, content description, text, accessibility identifier & label

✔ Uses a fluent interface (AppQuery)

Func<AppQuery, AppQuery>

Example: x => x.Class("UITextField")

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

TEST RECORDER & TEST CLOUD

The Test Recorder is a stand alone application that records your actions within the app and transforms them in Xamarin.UI tests.

It can send the tests to run in the cloud or export a C# file to embed in local Xamarin UITest project.

Questions?

BDD
ACCEPTANCE TESTING
UI TESTING
THE TOOLS

Thanks!

@xmonodev
http://crossplatform.io