Software Quality Workshops

With Amir Barylko

MavenThought Inc.

Why use Kanban

A Kanban board is a great way to ensure that the status for the project is visible to everybody, manage your throughput and be able to configure it against demand, improve the predictability of your business and make software development a real team effort.

The term Kanban is a Japanese word whose English translation means signboard or visual signal. A well-timed Kanban system works exactly like a traffic signal in managing the flow of traffic and meeting the real time needs of customers by sending clears signals on when to start, slow down, and stop production

Using a *Pull system* will change the way you develop software by focusing on *capacity* and not on *demand*. Some of the benefits are:

- Sustainable pace of development
- Rate of new requirements set based on delivery
- Create time to tackle technical debt
- Empower team and encourage ownership

Who should attend?

- As a **manager** you will learn to discover your team capacity and current status with just a glance at the board!
- As an **analyst** you will learn to improve your requirements and estimations!
- As a **tester** you will learn to participate early in the acceptance criteria definition avoiding recurring bugs!
- As a **developer** you will learn to display blockages and dependencies that affect performance and are usually hidden!
- As a **team** learn to identify bottlenecks and become more productive by reducing the lead time!

Contents

- The benefits of a lean process
- Push vs Pull systems
- Why WIP limit is so important?
- Buffers and bottlenecks
- Handling exceptions
- Cumulative Flow and other metrics
- Requirements with user stories
- Planning
- Sizing
- Accurate estimation
- Stakeholder commitment
- Healthy teams = happy life

Requirements

Notepad, pens, Laptop (optional if you want to follow the materials).

Sticky notes and sharpies will be provided :) !

Why learn Test Driven Development

Struggling with high coupled classes? Every time you fix a bit of code new bugs appear? No confidence in your current code base? Then this workshop is for you!

Programming is hard, and coding it right the first time even harder! **Test Driven Development** is a methodology that will help you discover the design of the solution to your problem making you write high quality code while at the same time getting as a byproduct tests that will document how your system behaves.

Add a dash of automation to the list of ingredients and you will have a recipe for success!

Benefits of TDD are:

- Let the methodology drive
- It will save your bacon more than once
- High quality code
- Very few bugs (no more regression)
- Increases developer confidence

Who should attend?

- As a **manager** you will understand why testing is important and how to keep you team producing high quality code
- As a **developer** you will learn to prove that your code does what it is supposed to do, discover common smells and use SOLID principles
- As a **tester** you will learn how to identify bugs quickly and how to fix them producing high quality code
- As an **architect** you will learn how to setup the testing harness from day one, setup database testing and manage acceptance criteria

Contents

- Benefits of “test first approach”
- Integration tests vs.unit tests
- Quality as a driver
- Red-Green-Refactor
- Common testing frameworks
- Using Given-When-Then
- Test automation
- Regression testing
- SOLID principles
- Mocking hardcoded dependencies
- The case for BDD
- Applying the outside-in approach

Language

Though the concepts can be applied to any language, most of the examples will be done in C#.

Full Java code of all the examples will be available for download upon request.

Requirements

- Basic knowledge of programming.
- Laptop with VisualStudio installed if you are doing C#
- Laptop with a Java IDE (Eclipse, NetBeans, other) if you are doing Java

Why learn Behavior Driven Development

Good requirements are essential to any software project. However, how we make sure they are *good?* How can we have certainty that they are working for the whole team and every team member is on the same page?

Behavior Driven Development is a methodology that helps capturing requirements and transform them in actual living documentation making sure the whole team is on the same page. Knowing what to expect will increase the confidence of the whole team and the ability to deliver valuable working software on each build.

Benefits of using **BDD** are:

- Encourages stakeholder participation
- Clear expectations
- Traceability from code to feature
- Live documentation
- Fewer cross feature bugs
- Improves complexity estimation

Who should attend?

- As a **manager** you will understand how defining acceptance criteria for each feature is a key to success
- As an **analyst** you will learn how to write requirements that can be converted into runnable stories
- As a **developer** you will learn how write and follow runnable features that will be used to validate your code and define when the feature is *done*
- As a **tester** you will learn how to write acceptance scenarios in collaboration with the developer before writing any code, making sure that the team delivers the expected functionality
- As an **architect** you will learn how to setup the acceptance test harness from day one, automate builds and implement continuous integration

Contents

- The case for BDD
- The outside-in approach
- The BDD cycle
- Combining BDD and TDD
- Keeping it Agile
- Integration tests vs. unit tests
- User stories
- Traceability
- Test automation
- Regression testing
- Writing sturdy scenarios
- Working with databases
- Stakeholder participation

Language

Though the concepts can be applied to any language, most of the examples will be done in C#.

Full Java code of all the examples will be available for download upon request.

Requirements

- Basic knowledge of programming.
- Laptop with VisualStudio installed if you are doing C#
- Laptop with a Java IDE (Eclipse, NetBeans, other) if you are doing Java