Influencing your Team towards BDD & Agile Practices

Karen N. Johnson

Chicago Quality Assurance Association (CQAA) January 2016

About today's session

- Understand the terms BDD & TDD
- Learn specific tactics to help move your team towards Agile testing practices
- Develop an "elevator" pitch for promoting BDD for every layer of management

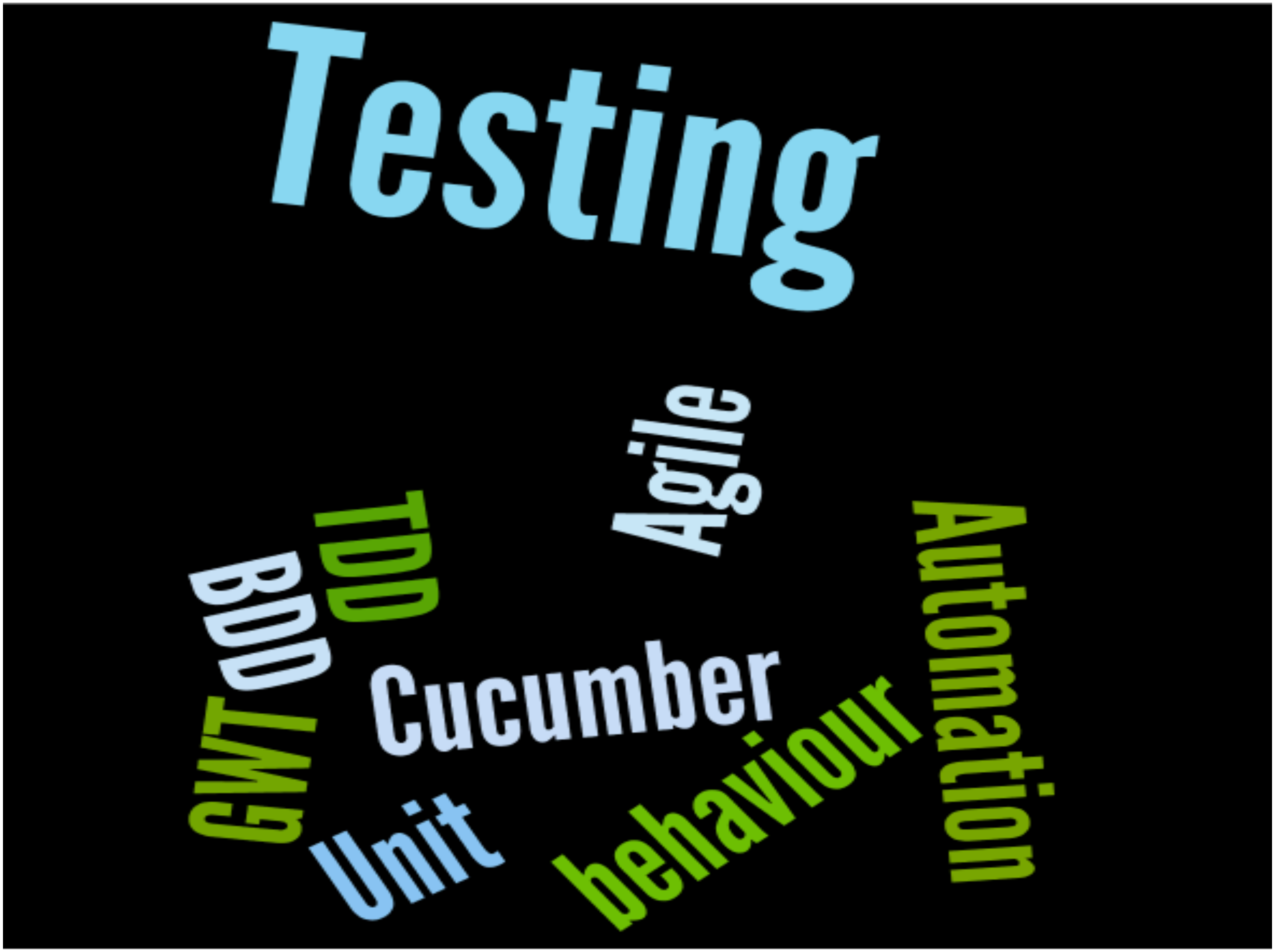
About today's session

- Understand the terms BDD & TDD
- *Meet one person today*
- Learn specific tactics to help move your team towards Agile testing practices
- Develop an "elevator" pitch for promoting BDD for every layer of management
- *Learn one thing today*

About today's speaker

- Published Author (Beautiful Testing)
- Teach Software Testing
- Speak at conferences
- Co-founder of WREST, the Workshop on Regulated Software Testing
- Website: www.karennicolejohnson.com or www.karennjohnson.com
- Twitter: @karennjohnson

Meet someone today (now)


Testing
Agile
TDD
BDD
Automation
Cucumber
GWT
behaviour
Unit

BDD: what does BDD mean?

Choose one:

1. Bold Design Decisions
2. Behavior-Documented Development
3. Behavior-Driven Development

BDD: what does BDD mean?

Choose one:

1. Bold Design Decisions
2. Behavior-Documented Development
3. [✓ Behavior-Driven Development]

TDD: what does TDD mean?

Choose one:

1. Test-Driven Development
2. Task-Documented Development
3. Test Design Development

TDD: what does TDD mean?

Choose one:

1. [✓ Test-Driven Development]
2. Task-Documented Development
3. Test Design Development

What is Cucumber?

Choose one:

1. An ingredient frequently found in salad
2. A tool designed to work in the GWT format
3. An automation tool that executes what has been written in English-understandable language

What is Cucumber?

Choose one:

1. An ingredient frequently found in salad
[✓ 2. A tool designed to work in the GWT format]
[✓ 3. An automation tool that executes what has been written in English-understandable language]

BDD basics

- Where to start in the process
- What to test and what not to test
- How much to test in one test
- What to call the tests
- How to understand why a test fails

BDD & Dan North

- Test method names should be sentences
- A simple sentence template keeps test methods focused
- An expressive test name is helpful when a test fails
- "Behaviour" is a more useful word than "test"
- JBehave emphasizes behaviour over testing
- Determine the next most important behaviour
- Requirements are behaviour, too
- BDD provides a "ubiquitous language" for analysis
- Acceptance criteria should be executable

Source: http://dannorth.net/introducing-bdd/

Story Formats

GWT

Given some initial context (the givens),

when an event occurs,

then ensure some outcomes.

As a (user) I want

As a [X]
I want [Y]
so that [Z]

X is the person (or role) who will benefit

Y is some feature

Z is the benefit or value of the feature

Story Formats

GWT example

Given a logged in user with a past order history,

when the user places a re-order

then the same products, payment form and shipping address will be used to generate an order.

As a (user) I want

As a logged in user with past order history

I want to be able to place a re-order

So that I can place an order to receive the same products and use the same payment form and shipping address so that I can place an order faster.

Influencing your team

As yourself what can you influence?

- *Test method names should be sentences*
- A simple sentence template keeps test methods focused
- *An expressive test name is helpful when a test fails*
- “Behaviour” is a more useful word than “test”
- JBehave emphasizes behaviour over testing
- Determine the next most important behaviour
- Requirements are behaviour, too
- BDD provides a “ubiquitous language” for analysis
- *Acceptance criteria should be executable*

Source: http://dannorth.net/introducing-bdd/

Offer ideas for unit tests

- The earlier upstream you can push testing and automation efforts, the lower the technical debt and the more streamlined testing will be at the end of the process.
- Help plan or highlight what “should” tests can be covered by development/unit testing.
- Coordinate what test automation is built by tester(s) and what automation is built by developers to help reduce automation efforts as well as automation run-time.

Use the word "should" for confirmation tests

- Your team will learn "should" tests as tests that are confirming something works "as expected."
- This may encourage "should" tests be developed as unit tests.
- This may also encourage developers to raise questions about "should" expectations. The use of the word should might provoke the team to think through "shoulds" and get clarity earlier.

Use the words “what if” for exploratory tests

- With “should” testing covered by unit tests, use the term (and develop the practice) that “what if” tests will be covered by Exploratory Testing.

- Start using the term “what if” tests to “train” the team to think beyond positive (confirmatory) testing.

Advocate for acceptance criteria that is executable

- As stories are written, monitor the acceptance criteria that is written (or be the person who writes the acceptance criteria) to make sure the criteria is clear and actionable.

- When acceptance criteria is written, continue to ask yourself – how will I (or my team) test to demonstrate the criteria. Avoid getting stuck with criteria that is not clear or for which you do not know how the criteria can be tested.

- In addition to monitoring acceptance criteria for each story, ask yourself how the stories for the sprint may interact with each other. Build an impact analysis.

Break the myth that only testers can test

- On an Agile Scrum board, make sure testing tasks are clearly identified and not just post-it's that say: testing. This allows other team members to pick up testing tasks.

- Break the myth that only testers can test by coaching the non-testers on the team how to test, how to explore.

- Provide "emotional & tactical" support for team mates who help test.

Advocate to prevent technical debt

- During Sprint planning and throughout a sprint, keep asking how each feature, each defect added will not add to the technical debt of the team.

Tactics to influence

Photo by Dave Fayram, Flickr

1. Offer ideas for unit tests
2. Advocate for expressive test automation names
3. Use the word “should” for confirmation tests
4. Use the words “what if” for exploratory tests
5. Keep watch that acceptance criteria is executable
6. Break the myth that only testers can test
7. Advocate to prevent technical debt

Make note of a tactic you can do.

Elevator pitch

Photo by Gideon Tsang, Flickr

A pitch should:

1. Identify Your Goal
2. Explain What You Do
3. Communicate Your USP (unique selling proposition)
4. Engage With a Question
5. Put it all Together

Source: https://www.mindtools.com/pages/article/elevator-pitch.htm

My tips on elevator pitches

- Be ready!
- Know who you're talking to
- Know what matters to them: WFFM
- Elevator pitches are more likely to happen in a hallway
- Sometimes you can only fit in one thought, one comment – seize the moment

Write an Elevator Pitch

Write an elevator pitch

Purpose: practice building an elevator pitch for BDD. Build a pitch for each of the following roles: Product Owner, Developer, and an "executive" at your company.

Time: 15 minutes!

Focus on:

- Who (how the person shapes the pitch)
- What (what matters to the person)

Done?

- Your team is done when you have a pitch for each of the three roles – or when time runs out!

Share your team’s Elevator Pitch

Thank You

for being here today.

Website: www.karennicolejohnson.com or www.karennjohnson.com

Twitter: @karennjohnson

Email: karen@karennjohnson.com