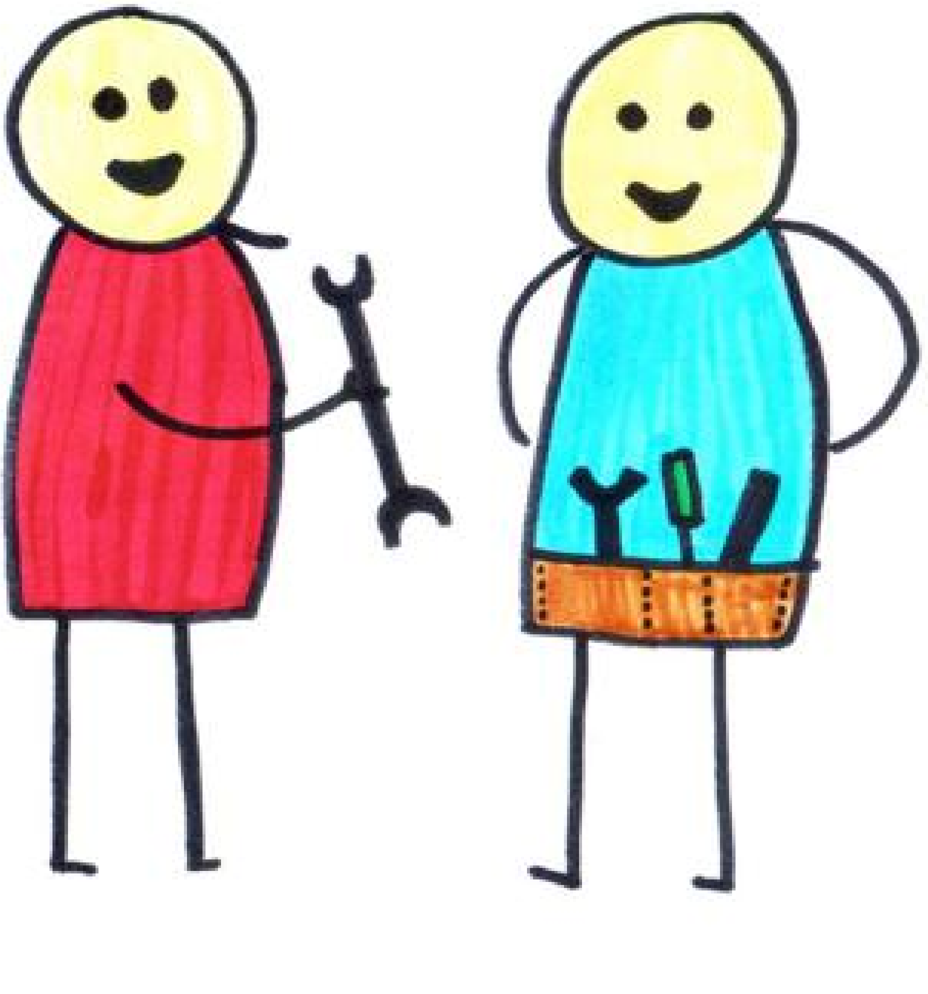
growing agile

A Coach's Guide to

Agile Testing



Karen Greaves Samantha Laing

Growing Agile: A Coach's Guide to Agile Testing

Samantha Laing and Karen Greaves

This book is for sale at http://leanpub.com/AgileTesting

This version was published on 2015-01-09

* * * * *

* * * * *

Table of Contents

Appendix

Acknowledgements

We would like to thank Janet Gregory and Lisa Crispin for their thought leading work in their Agile Testing book. Many of the ideas in this book were inspired by Janet after attending her Whole Team Test Approach course. We’d highly recommend the course if she is presenting it in your area.

We’d also like to thank Sharna Sammy for her fantastic cover designs for our “Coach’s Guide” series.

About the Authors

Sam Laing (left) and Karen Greaves (right)

We are Sam Laing and Karen Greaves. We have worked in software our whole lives. With Type A personalities and a strong work ethic, we have both done our share of overtime on death march projects. Eventually we knew we had to find another way. Agile brought us together when we worked at a company trying to do Scrum for the first time.

In 2012, we took the plunge and started our own business, Growing Agile. Since then we have been doing the work that we are passionate about -

introducing and improving agile. Best of all we have a positive impact on other people’s lives.

One of the first things we did as a company was bring Janet Gregory to South Africa to run her Whole Team Test Approach course. Her training resonated with us. We felt there where key insights about agile testing we could share with teams in short workshops, that could help them think differently about testing. We’ve been delivering talks and workshops on agile testing every since.

As always, we love feedback, so don’t hesitate to send us your thoughts via email <strong>info@growingagile.co.za</strong> or Twitter **@GrowingAgile**.

How To Use The Coach's Guide Series

As agile coaches we often find ourselves running workshops or training sessions with people we are coaching. We put a great deal of effort into creating the plans for these sessions to help the participants get value. Over the past 2 years we have collected a lot of these plans. This series is our way of sharing these workshop and training plans with other agile coaches to enable you to run similar workshops.

All the books in this series are structured in a similar way, this section explains the concepts you'll need to effectively use any of the books in the series. We've put it here at the start of the book, so that if you've used any of the other books in the series you don't need to read through this again, it's the same in each book.

4Cs Plans

Each chapter in these books includes a 4Cs plan. The technique comes from a training style called Training from the BACK of the room (TFTBOTR) developed by Sharon Bowman.

TFTBOTR is based on how adults learn and is focused on maximising learning and retention. TFTBOTR describes four parts that should be included in any training plan. These parts are known as the 4Cs and are described below.

- C1 – Connections: To get participants to connect with each other and the trainers, and to connect participants to what they might already know about the topic
- C2 – Concepts: Some facts and theoretical concepts about the topic
- C3 – Concrete Practice: An activity or simulation to experience the topic
- C4 – Conclusion: An opportunity for participants to evaluate what they have learned about the topic

Another important part of TFTBOTR is making sure you use a variety of methods to keep people engaged. Read more about it in this article on the Six Trumps by Sharon Bowman.

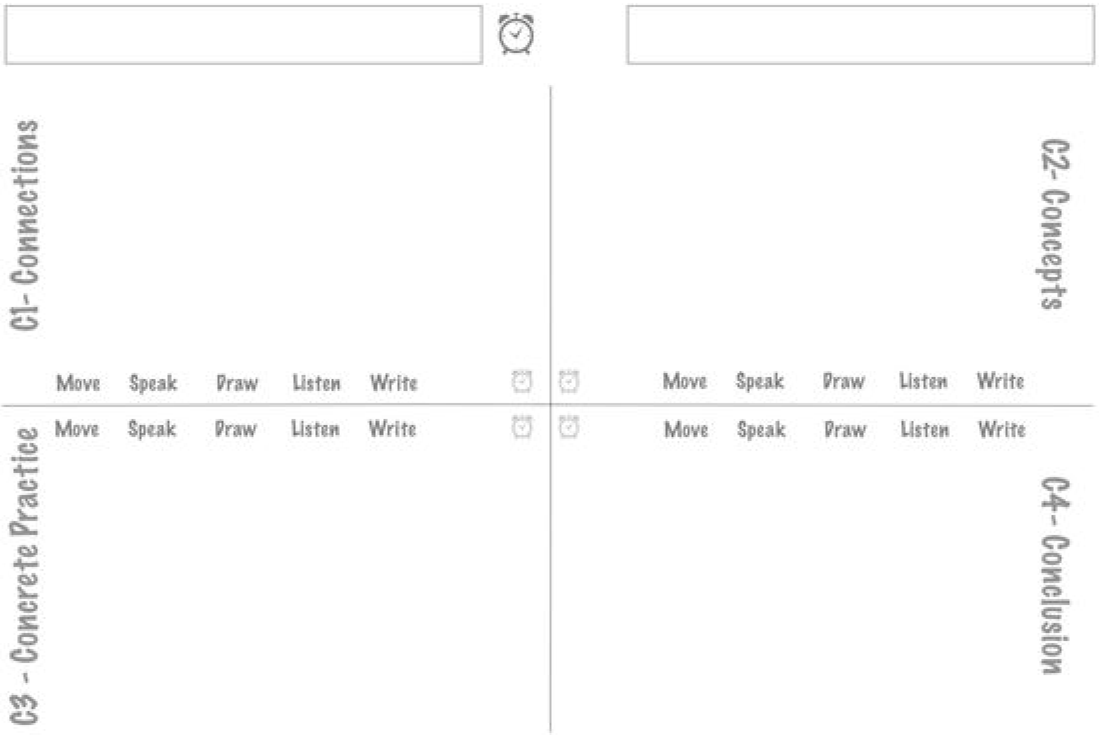
After using this technique extensively for training, we started using it for workshops as well. The 4Cs plan is a great way to weave new information or a technique into a working meeting. You can use C2, the concept stage to talk briefly about a technique, then spend time in C3, getting practice on using the technique on your work items.

Note

Occasionally it makes sense to swap the order of the C2 and C3. For some topics it is better for people to experience what you are talking about with Concrete Practice first, and then for you to teach the theory. This is especially true if you have a great game or simulation to illustrate the point. When we do this we just put the C3 in the C2 block of the template, and vice versa.

We drive all our workshops and courses from these 4Cs plans. If you usually train from slides this might take time to get used to. We print out the 4Cs plans and refer to them during the course or workshop to see what's up next and if we are on track.

We have created our own template for the 4Cs plans. The template can be found in the Coach Toolkit for each book. Use it to create your own training plans.


C1- Connections
C2- Concepts
Move Speak Draw Listen Write
Move Speak Draw Listen Write
Move Speak Draw Listen Write
Move Speak Draw Listen Write
C3 - Concrete Practice
C4- Conclusion

Here is a short overview to help you understand the template.

- The box in the top left corner is for the name of the topic.
- The big clock icon gives the time for the entire plan; the smaller clock icons in each quadrant gives the time needed for that section.
- The box in the top right corner has a space for you to enter the time for a section. For example 9:00 to 9:30 am. This helps you stay on track during the training. These are not filled in on the training plans we provide. We suggest you fill them in when you have planned your training.
- The rest of the page has a quadrant for each of the 4Cs. C1 covers connection activities. C2 is for concepts and is quite often a short lecture. C3 is for concrete practices or some activity to help people understand what they have learned. C4 contains conclusions of how people might apply the learning.
- At the bottom of each quadrant you can circle what the participants are doing in each section: *Move*, *Speak*, *Draw*, *Listen*, *Write*. This helps ensure that you have sufficient variety in each topic.

Chapter Layout

Each chapter contains the following:

- overview of the topic covered in the 4Cs plan
- 4Cs training plan
- notes on delivering each 4Cs part
- slides used for the topic
- exercises used for the topic.

Once you have a feel for what each topic covers you can structure your own workshops using one or more topics depending on your goal and time available.

Coach Toolkit

Each book in the series includes a Coach Toolkit which you can download from our website. The toolkit contains the following items.

Training plans: PDF combining all the 4Cs training plans. You should print these out and use them when you train. You will notice that these plans are handwritten, we find them much easier to create and change by hand than if they are typed.

Slides: PPTX containing all the slides used. These slides were created using scanned hand drawings. Some slides have been edited to allow you to insert your own details. For these slides we used Lauren C. Brown font as it closely matches the handwriting on the other slides. If you prefer not to use slides you can recreate these images on flipcharts.

4C template: Use this blank template to create your own 4Cs plans on new topics.

Agreement Cards: PDF of cards used in the Getting Started chapter of each book. We printed and laminated them and use them in nearly every workshop we run. You don’t need to use all the cards each time. Look through the cards before each workshop and decide which agreements are appropriate. The cards help make sure you don’t forget anything important.

Workbook: DOC containing all the pages of a participant workbook. You should print one per participant for them to fill in. Feel free to edit the order and cover page of the workbook. Many of the workbook images were created in Omnigraffle and pasted as images into the workbook.

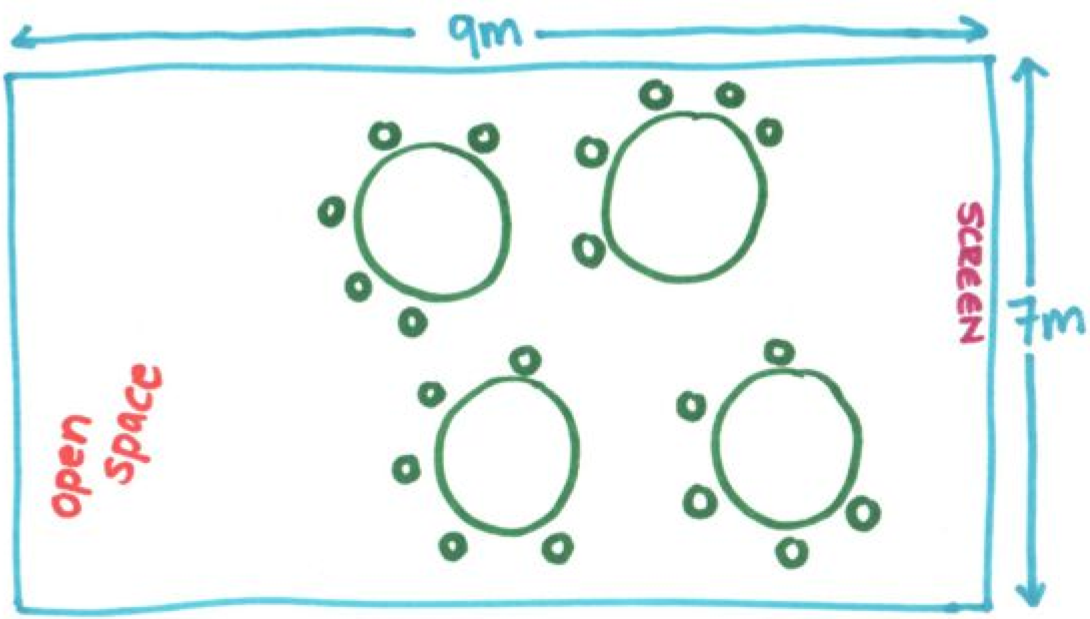
Handouts: Not all books in this series contain a workbook, since we generally only use workbooks in training courses. For topics that work well individually we provide PDFs of handouts that can be printed for each participant.

Other materials: PDFs containing materials to be printed and used in various chapters. Each chapter will reference these if they are needed. These are different in each book.

Room layout

We have trained in a variety of venues around the world, including a computer training centre, a bar and a tent! Room layout can have a significant impact on your training.

Our preferred room layout is cabaret style. i.e. small round tables seating groups of five to seven comfortably. The room should be large enough to have open space for some of the discussions. We look for a room with dimensions 7m x 9m for 20 people, with four tables. Ideally the tables should be small enough (around 1.5m–2m diameter) that people can easily talk to everyone at the table, but still have place for everyone to take notes.



Don’t worry about allocating seats when people arrive. The Getting Started chapter includes an activity for the participants to self-organise into appropriate groups.

If you are facilitating an in house workshop with only six participants, try find a room with a small round table so that everyone can sit close to each other.

When to change exercises

Feel free to change the training plans and activities to suit the class size and time available. We have delivered most of the chapters to groups varying from five to 50 people. As a result we have developed activities that scale well, but it is a good idea to be aware of the size of the group when planning your activities.

All 4Cs plans give times for each activity. These are just guidelines; any activity can be adjusted based on time available. It is often useful to have two exercises on hand, a longer and a shorter one, so that you can adjust if you find yourself with more or less time available.

If you are working with large groups, be aware that debriefing exercises can take much longer. To save time you can have teams debrief in their table group and then ask one or two table groups for their insights. Also remember that some exercises speak for themselves and don't have to be debriefed - this is the beauty of TFTBOTR.

What else do you need?

We are able to run most of the workshops in this series with our standard training kit. We keep this packed in a small suitcase on wheels so we can take it wherever we go. Below is a list of what you'll find in our kit. Some books in this series require specific items, these are listed in the Introduction for each book. Each chapter also contains a full list of materials you need for that topic's training plan, in case you plan to deliver just one topic.

Standard training kit

- flipchart with 40 sheets
- flipchart markers
- laptop and projector, including connectors and remote, if you plan to use slides
- visible timer for timeboxing activities. We use Timer+ on an iPad.
- camera to take pictures
- soft ball that can be thrown around without injuries
- masking tape for sticking up posters
- coloured markers and pens for each table
- sticky notes for each table
- index cards for each table
- one set of Agreement cards. These are available in the Coach Toolkit.

Note

If you are doing a lot of training, we recommend investing in some high-quality markers in different colours. Our favourite markers are from Neuland. They offer large, refillable, water-based markers in a great range of colours.

All the techniques referenced in the training plans are available in the Appendix. If you aren't sure what to do for a Standing Survey or Fast Pass, check the Appendix.

After the workshop or course

Whenever we train or run workshops we take photos. These include action shots during any activities and discussions as well as any flipcharts we use and posters people create.

After the workshop or course we put these photos together in a PDF, and send this to all participants as a reminder of the workshop or course. This photobook is useful if you don’t use slides and participants want some materials to reference afterwards. We also send links to further reading on any topics that came up in the Q&A that were not fully answered.

Chapter 1: Introduction

If a team believes they are agile, but nothing has changed about the way they test, then there is still much to learn. We teach 5 key principles that explain why agile testing is fundamentally different to traditional testing.

This books includes a collection of workshops to help teams grasp these principles and adopt an agile testing mindset. It's not just for testers. A key part of agile testing is that the whole team is involved, so we always run these workshops with everyone in the team.

If your team is ready for the next level we highly recommend running through the workshops in this book, it will teach them a number of simple but valuable techniques to help prevent bugs and dramatically increase the quality of your products. We provide the facilitation plans, teaching points, and even the slides you might use to help you run the workshop.

The chapters in this book each relate to a different topic on agile testing. You can use the book in a number of ways.

- You could use all the chapters together to deliver a half or full day training course on Agile Testing. This is usually how we run the workshops, and so many chapters build on things done in the previous chapter.
- You can use an individual chapter to run a workshop session on a particular topic of interest. We recommend doing the Agile Mindset first as it is reinforced in the rest of the chapters.

For each chapter, you can expand the learning by using the technique just taught on items the team are currently working with.

Unlike our previous book on Training Scrum, we don't assume you are an expert on the topics in this book. Not every coach and trainer have come across the same tools. If a topic is new to you, we have provided details of the points we teach for each topic in the C2 section. There are also links to blog posts and books we recommend on the topic on our website.

You only need the standard training kit mentioned in How to use this Series to run most of the workshops in this book. However, if you plan to run the Jenga game mentioned in the Getting Started chapter, you will also need a few Jenga sets. Two sets are enough for up to 18 people.

Since testing and requirements are closely linked together we recommend our Agile Requirements book (part of the series) as a good companion to this one.

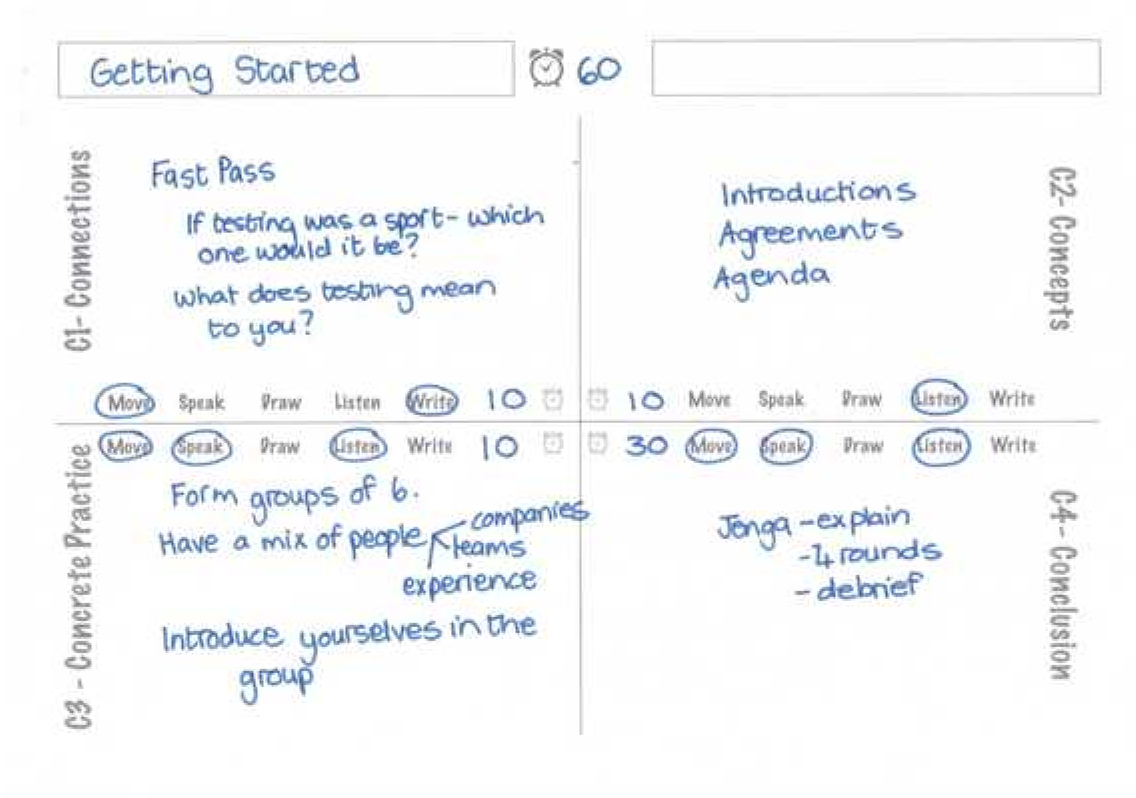
Chapter 2: Getting Started

We spend time at the start of a workshop session building connections and getting people to bond with each other and us. This immediately increases the level of trust in the room. If you are running a 90 minute workshop with people who all work together, you don't need to spend as much time on this. In that case we might only do the Jenga Game in C4. If you are doing a full day session with people who don't know each other well, we'd recommend following the full plan below.

✎ Materials needed

- coloured markers and pens for each table group
- sticky notes
- flipchart sheets
- one set of Agreement cards. These are in the Coach Toolkit.
- all items to play the Jenga game

4Cs Training plan



Getting Started ⏰ 60

C1- Connections

Fast Pass

If testing was a sport- which one would it be?

What does testing mean to you?

[✓ Move] Speak Draw Listen [✓ Write] 10

C2- Concepts

Introductions
Agreements
Agenda

10 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

[✓ Move] [✓ Speak] Draw [✓ Listen] Write 10

Form groups of 6.

Have a mix of people < companies
teams
experience

Introduce yourselves in the group

C4 - Conclusion

30 [✓ Move] [✓ Speak] Draw [✓ Listen] Write

Jenga - explain
-4 rounds
- debrief

FastPass (C1)

We start with Fast Pass posters. Ask everyone to form pairs and together write up one sticky note for each of the posters. The posters we chose for this workshop is “If testing was a sport - what would it be?” and “What does testing mean to you?”. This allows for participants to instantly identify with the topic and for us to get a better understanding of their mindset around testing.

The posters are usually flipchart sheets with a question written on them in bright colours. You can see an example of these posters on the slide below.

Slide

If Testing was a sport it would be .....
What does TESTING mean to you?

Introductions (C2)

Simply introduce yourself by saying who you are and what your experience is. Don’t bore people - keep it short and entertaining. Our names, Twitter handles and company website are shown on a slide (see below). If you are doing this training in house with people you know, this slide might not be needed. Perhaps instead share something people might not know about you. You probably also want to mention the goal of the workshop, and why you are there.

Slide

Agreements

Next go through some Agreements (like rules, but we don't like that word - too formal). For example: "Please put cellphones on silent, but feel free to leave the room if you need to take a call". This helps to set the tone for the workshop. It also allows everyone to understand the boundaries of behaviour that are expected for the duration of the workshop. The Agreement cards in the Coach Toolkit will help you remember things to mentioned. Pick the cards most appropriate for the workshop and run through each of them quickly.

Agenda

Briefly go through the agenda for the day. Most people want to know at a high level what will be covered, and most importantly when the breaks and lunch are, and what time they will finish. We put this information up on a slide, and explain that the times are approximate, although we should be within 10 minutes of them at all times.

Form groups (C3)

Ask the class to form groups of five to six people with a mix of people in each group. Encourage them to rather mix up companies or teams if possible. Once people have formed groups ask them to introduce themselves to everyone in their group.

Jenga Game (C4)

In order to illustrate some testing mindset shifts we play a little game. This is a fun and light hearted way to get people thinking differently about testing and what is possible. Introduce the Jenga game and then play it with the class.

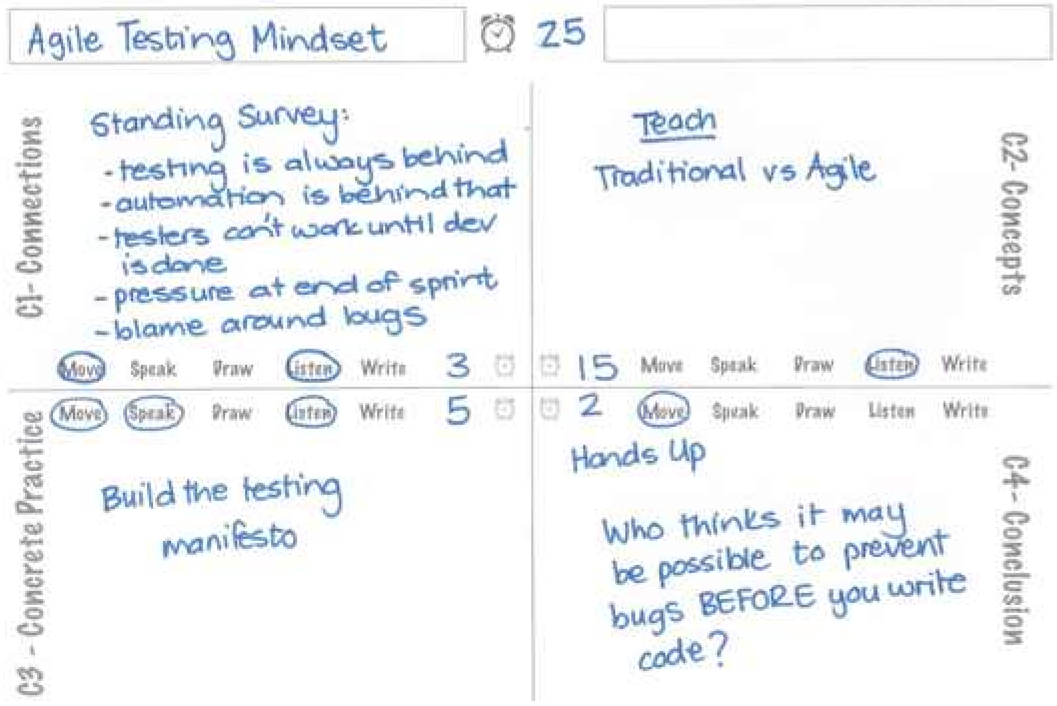
Chapter 3: Agile Testing Mindset

Before teaching any techniques or theory we like to help people understand that agile testing is fundamentally a different mindset to traditional testing. This topic presents five key differences in the form of a testing manifesto, which was inspired by the agile manifesto. We often deliver this topic as a stand alone workshop, with the Jenga game at the start or even public talk. It’s a great introduction to the concepts in agile testing. Often the phrases we refer to in this topic like “Testing is an activity not a phase” are the things teams remember most and repeat to each other after we have trained them. Proof that presenting this topic this way help it stick in their minds.

Materials needed

- Testing Manifesto Cards. These are in the Coach Toolkit.

4Cs Training plan



Agile Testing Mindset ⏰ 25

C1- Connections

Standing Survey:
- testing is always behind
- automation is behind that
- testers can't work until dev is done
- pressure at end of sprint
- blame around bugs

[✓ Move] Speak Draw [✓ Listen] Write 3

C2- Concepts

Teach

Traditional vs Agile

15 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

[✓ Move] [✓ Speak] Draw [✓ Listen] Write 5

Build the testing manifesto

C4- Conclusion

2 [✓ Move] Speak Draw Listen Write

Hands Up

Who thinks it may be possible to prevent bugs BEFORE you write code?

Standing Survey (C1)

This helps everyone in the room get to know a little bit about the testing problems others have. Ask people to stand if any of these are true for them:

- testing is always behind
- automation is even further behind that
- testers can’t work until development is done
- there is pressure at the end of a sprint
- there is blame around bugs (its his fault etc.)

Teach (C2)

Introduce the idea that agile testing is not just testing the same way you always have, except in sprints. Explain that the entire way the team thinks about testing should change. Then present the five mindset points below. Each is presented as the traditional way of thinking about testing and then the agile way of thinking. Contrasting like this can help people understand the differences, but it also helps them reflect which side they are closer to at the moment.

Testing is an activity not a phase

Traditionally people view testing as a phase that happens at the end of development. In agile most have changed it that the chunk of development done is smaller, but the testing still happens last. Nothing has fundamentally changed about how testing is done.

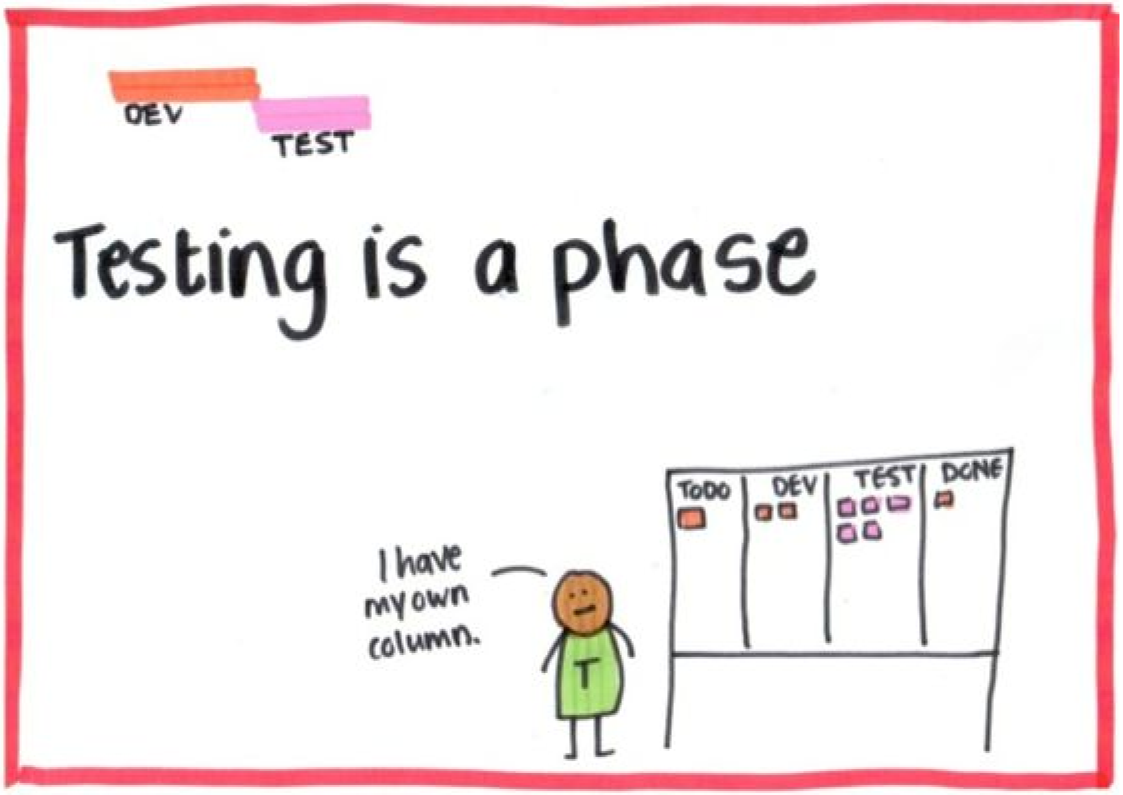
One way to see if this is the case is to ask people about their taskboards. If taskboards have a separate column for testing, it's a sure sign that testing is still being thought of as a phase.

In contrast in agile, testing is just an activity that needs to happen, along with coding, documentation and everything else. Thinking about it like this makes it possible to consider the idea of doing testing tasks before development work. A great way to visualise this on a taskboard is that instead of having a separate column for test, rather just make testing tasks a different colour sticky note. Now put all the tasks in the "To do" column together. Challenge the team to see how many of the testing tasks they can do before any development tasks happen.

For example you can create test cases before any code is written. That way you know how you are going to test it before you build it. You could even create automated acceptance tests first. These should fail since there is no code yet, but once the code is written and the tests pass, the work is done, and there are no test tasks left. Working this way will remove the hurdle of testing always being behind. For some people this is a huge step, however just breaking the mentality that testing tasks follow development is a great start.

Another useful technique is the "Show Me" column. Put it after the "In Progress" column, before the "Done" column. Most teams do code reviews, documentation reviews or even test case reviews on each story. The idea behind the "Show Me" column is to do a review on every task as soon as that task is done. If tasks are small these are micro reviews that might only take a few minutes, but they ensure that at least two people in the team have seen every piece of work, and this can help catch and fix issues much earlier.

Slides



Testing is a phase

D D T D
T D T T
D T D D
Testing is an ACTIVITY!
I'm part of the board!
T
TODO
BUSY
SHOW ME
DONE

Prevent bugs rather than finding bugs

Traditionally people think that the goal of testing is to find bugs. In fact some organisations even measure tester productivity based on the number of bugs they find (or don't find). Once again this mindset is limiting, and helps reinforce the idea that testing is something that happens at the end.

Use the star example to illustrate this point. Show the star slide or draw the star on a flipchart and ask people "How many points are there on this star?" People might offer a few numbers: 5, 10, 20. Ask people to write down the number of points they think it has. The ask people to raise their hands if they wrote anything other than 5.

Now show the next slide, and explain that the point inside the circle with a tick is a point (and only one point), and that the other circle with a cross is not considered a point. Hopefully now everyone can agree that there are only 5 points on a circle.

Explain that anyone who wrote down a number other than 5 created a bug. Ask if they could think of any way that they could have prevented that bug.

Hopefully someone will realise that they could have asked you what you mean by a point before writing down their answer. If no one mentions this, you can. Explain that this works exactly the same way in software. Often people make assumptions about requirements and implement those assumptions before clarifying them. The assumptions are only clarified once the software is tested, and the bug is then found. Imagine how much more productive it would be to have a short conversation to clarify assumptions before anyone wrote a single line of code.

Tip

Very occasionally someone might ask questions before they write down the answer. That's okay, answer the questions and see if everyone then writes down a five. Now use that to illustrate how a bug was been prevented. In all the times we have used this example, this has only happened once!

This example introduces the second agile mindset principle. Agile testing aims to prevent bugs by seeking to eliminate all assumptions and unknowns before starting to code. The goal is to make sure everyone from the customer to the developer and the tester have exactly the same understanding of how something should work. The best way to prevent bugs is to ask questions, especially stupid questions. Ask questions that everyone thinks the answer to is obvious. Remind them of the star. To some people it was obvious that their were 20 points.

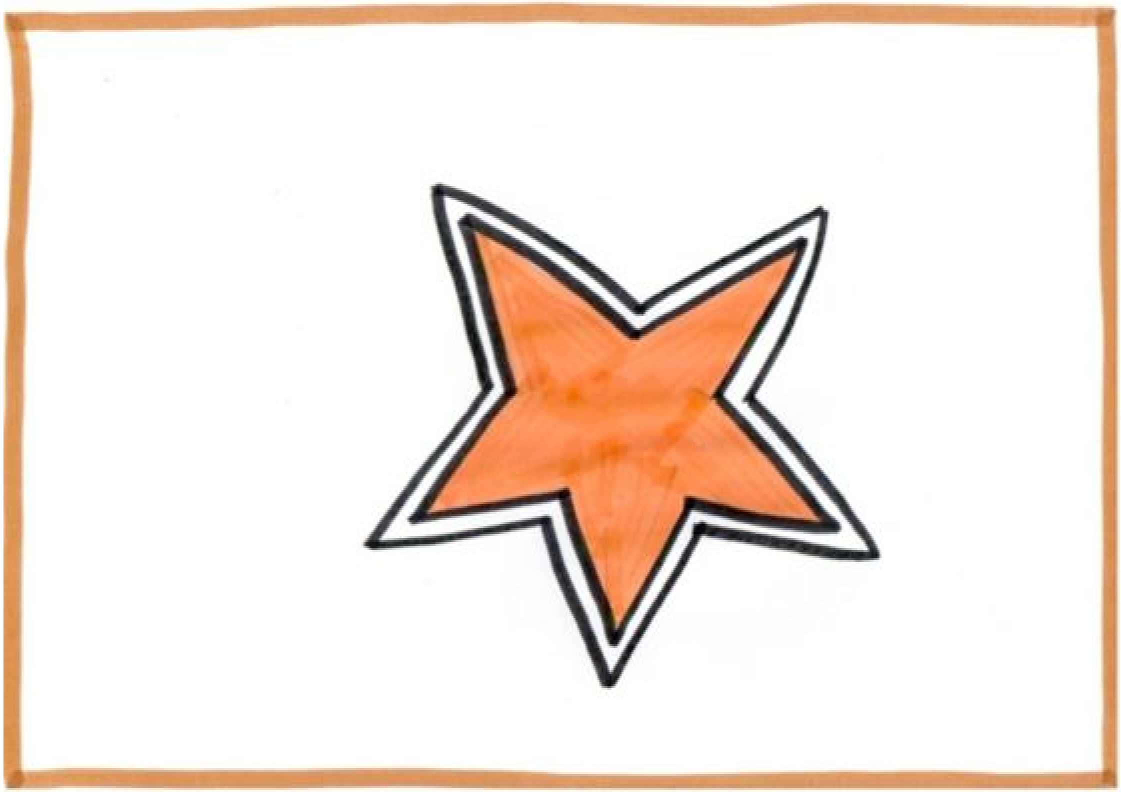
Our favourite example for this was a team that needed to create a report showing the average sales data for the last six months. Everyone thought they understood the requirement perfectly, and it didn't need much discussion. We happened to be there and we asked some questions:

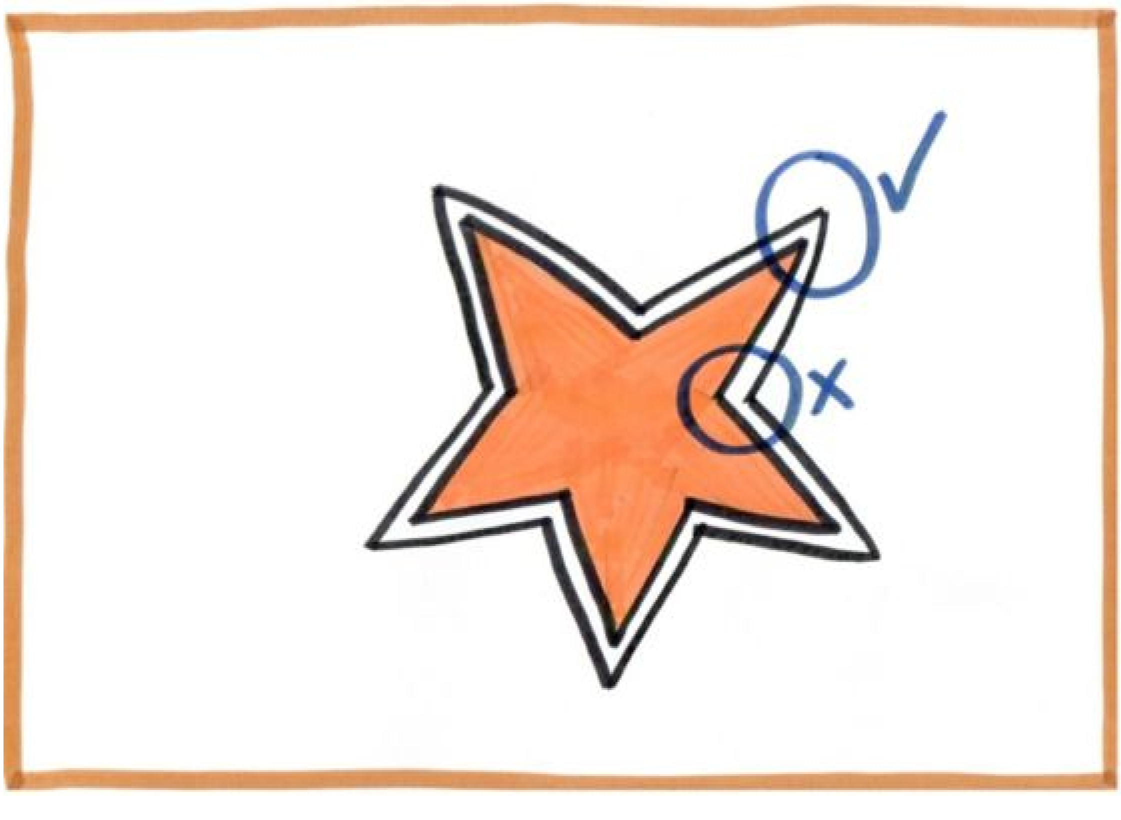
- If I run the report on 1 February is data from February included or not?
- What about if I run the report on 29 February?
- How exactly should the average calculated, as monthly average, or the average over the six months?
- Does the average need to be stored or is it calculated on the fly?
- Does the report need to be stored, or will it only be created when someone selects it?

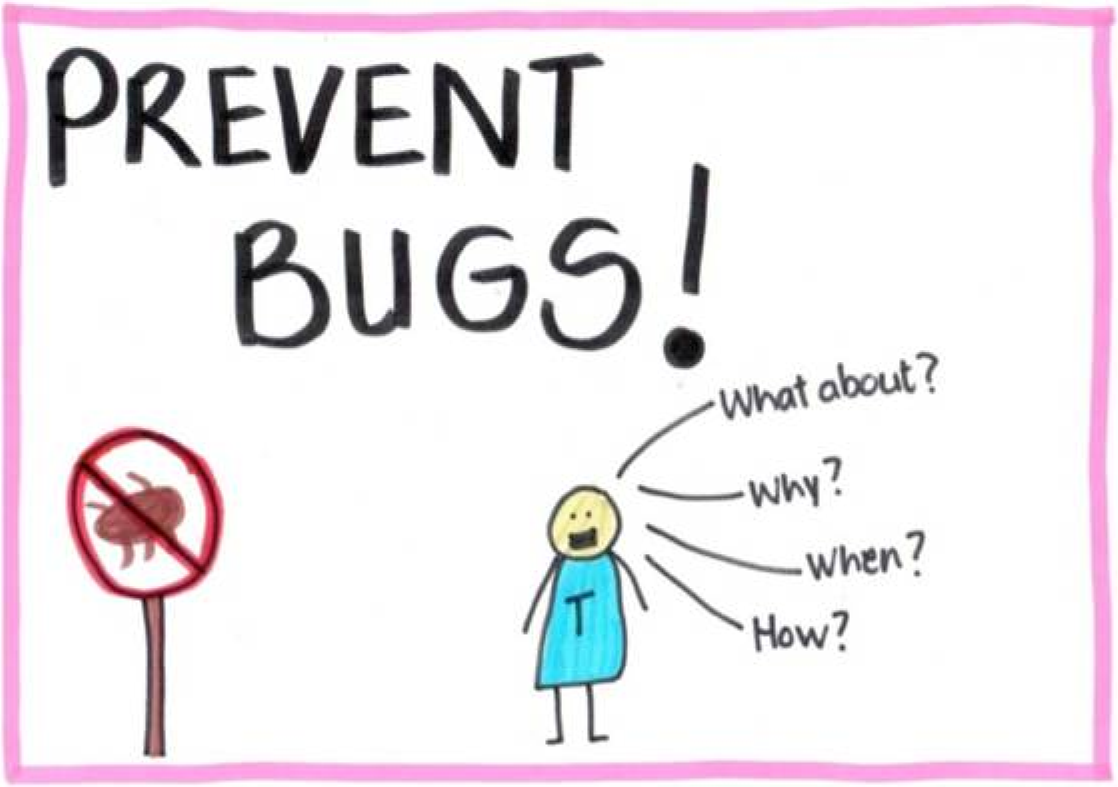
- What field in the database is the average calculated from?
- Who would be using the report and why did they need it?

It soon became very clear that no one had considered these items, and that more information was needed before they could build the report. Imagine the rework and bugs that could be created if you built this without the answers to these questions.

Slides




✓
x



Don’t be a checker, be a tester

Traditional testers often don’t like agile because without detailed specification documents they are suddenly unable to do their jobs. This is because they consider their job being to compare the working system to the specification, and report where there are discrepancies. If you think about this for a second, the only thing they are checking is how closely the developers followed the specification. This actually says nothing about the quality of a product, or more importantly if it is fit for purpose.

We call this work ‘checking’. You know what’s really good at checking? Computers! Checking that 1 + 1 = 2 is easy work for computers to do, and they will get it right every single time. They don’t get bored or tired or distracted. With agile testing simple checking should be automated so that testers can be freed up to do the kind of work computers can’t do. Things like exploratory testing or usability testing.

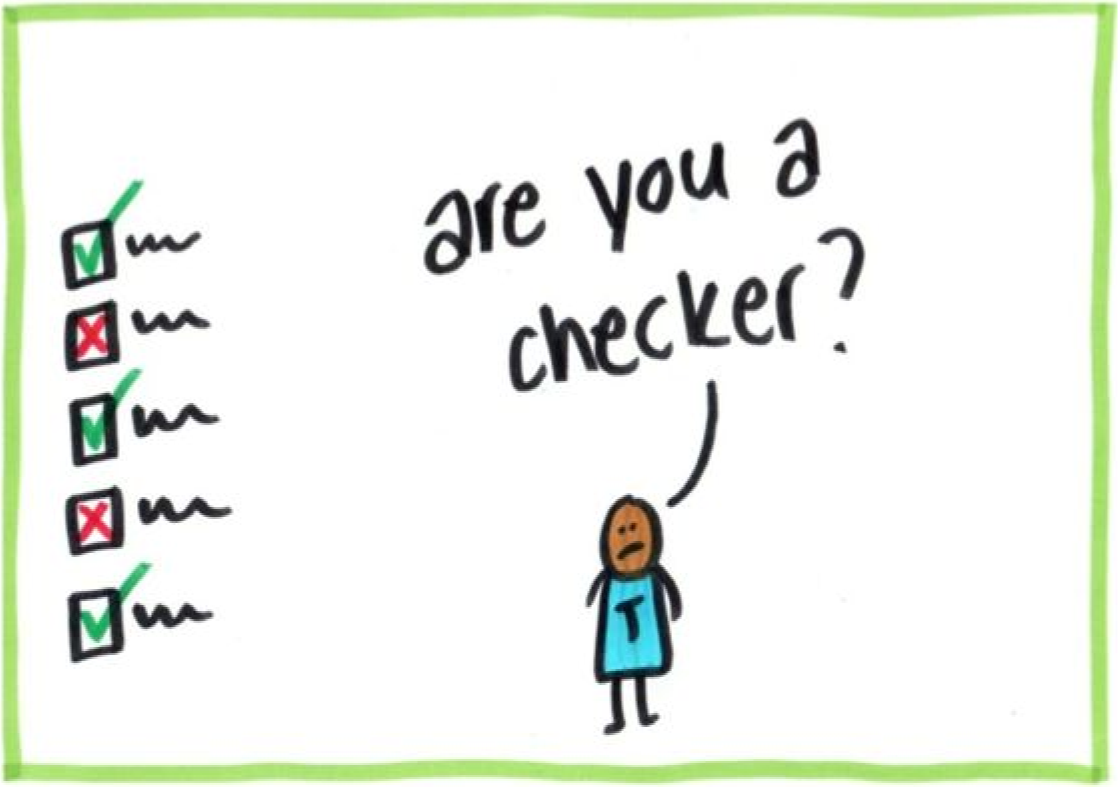
In agile, testers need to become customer advocates. They need to deeply understand who their users are and what they are trying to achieve with the product. They should be the representative of that customer in every design decision, ensuring that the feature meets the customers actual needs, not just the specification, or even what they asked for.

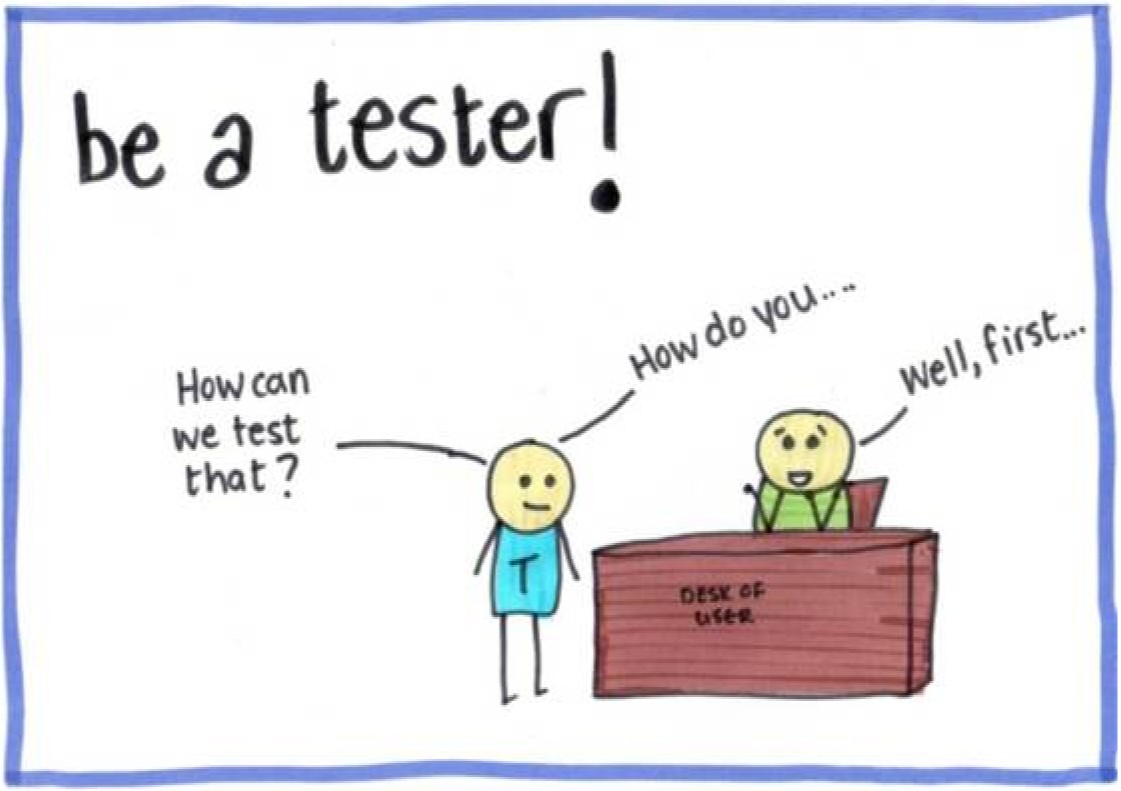
When a user asks for a feature, ask them: “How would you test that?” or “How will you know if that works?”. This can help understand the real result the customer is looking for. Translating that into acceptance criteria for the team can ensure the product does the right thing.

Note

We have an entire book full of workshops dedicated to agile requirements and acceptance tests. Please see: Agile Requirements.

Slides





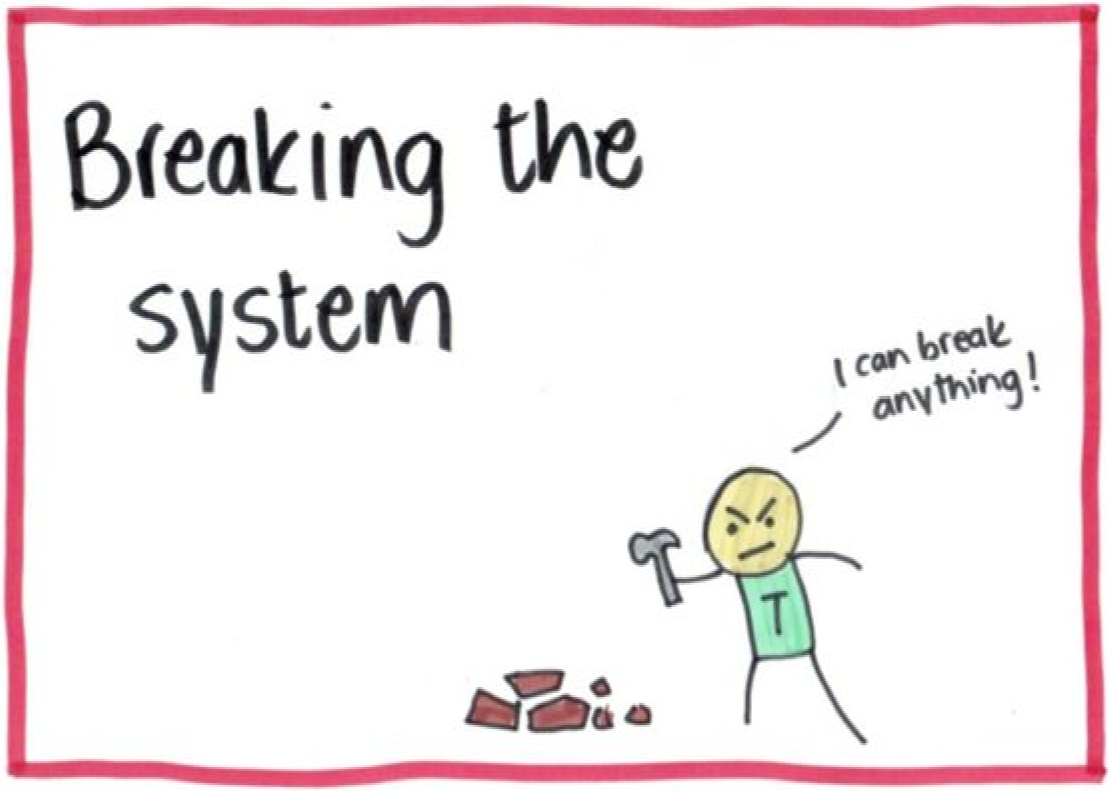
Don’t try to break the system, instead help build the best possible system

Testers like to break stuff. Yes that’s a generalisation, but it is certainly true for the majority of testers we meet. The problem with this mindset is that it creates a divide between the developers and testers. Developers build it, then testers try to break it. See how this reinforces the other traditional mindsets like testing as a phase. When this gets really extreme some strange stuff happens, like testers telling developers how they will test the product. We like to share the following story.

Many years ago I (Karen) was working as a project manager on a traditional waterfall project. We were nearing the end of development and preparing for the final test run before user acceptance testing (UAT). I was discussing the upcoming UAT with the client, and here is what they said to me. “We don’t want to share our UAT test cases with you, because then you might just build the system to make those test cases pass”. At the time I agreed and said I understood. Now I literally laugh out loud at the absurdity of this statement. Surely the client wanted software that would make their UAT test cases pass. Wasn't that in fact our joint goal!

The agile mindset is that testers should be helping to build the best system possible. They shouldn't be celebrating when they find a bug, they should be celebrating with their whole team, when the product works, and solves a business problem in a simple way. The best way to do this is to figure out how to test the system from a user point of view and then share that with developers before they start coding. Chances are high if you do this, they might actually build a system to make those tests pass.

Slides



Breaking the system

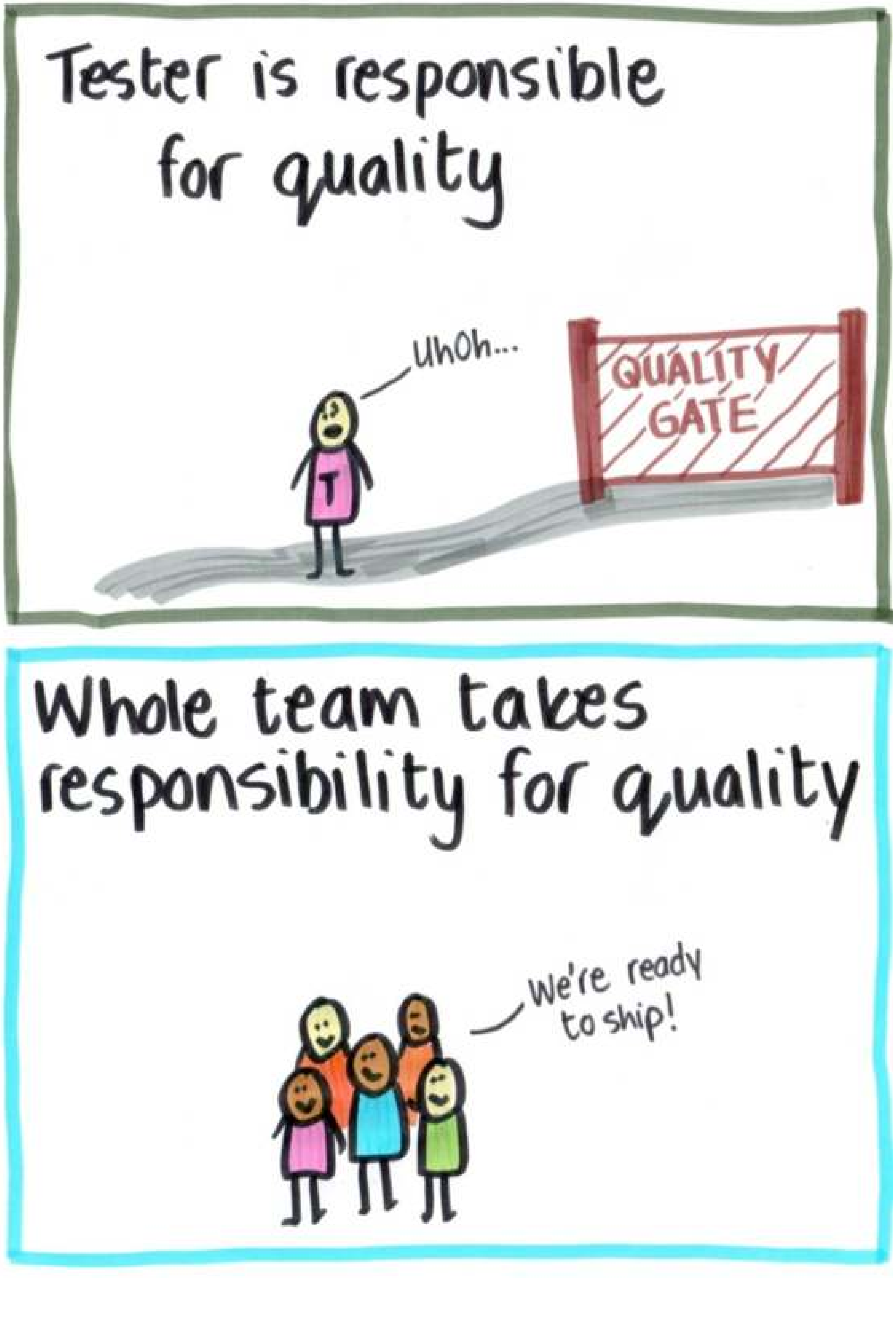
The whole team is responsible for quality, not just the tester

Traditionally it is the tester, or the test team that is responsible for quality. They get the final say in whether a product is ready to be released or not. The problem with this mindset is that it implies then that only the tester cares about quality and only the tester spends their time ensuring it happens.

Instead in agile the whole team is responsible for quality. This helps teams realise that testing is an activity they all need to take part in and that it happens throughout the work. If customers find a bug in production, no one should be asking the tester why they missed that. Instead the whole team should be discussing together how they can prevent that from happening again in the future.

Once this mindset is adopted, testers are no longer the only people busy at the end of the release, the whole team is involved.

Slides


Tester is responsible for quality
UhOh...
QUALITY GATE
T
Whole team takes responsibility for quality
We're ready to ship!

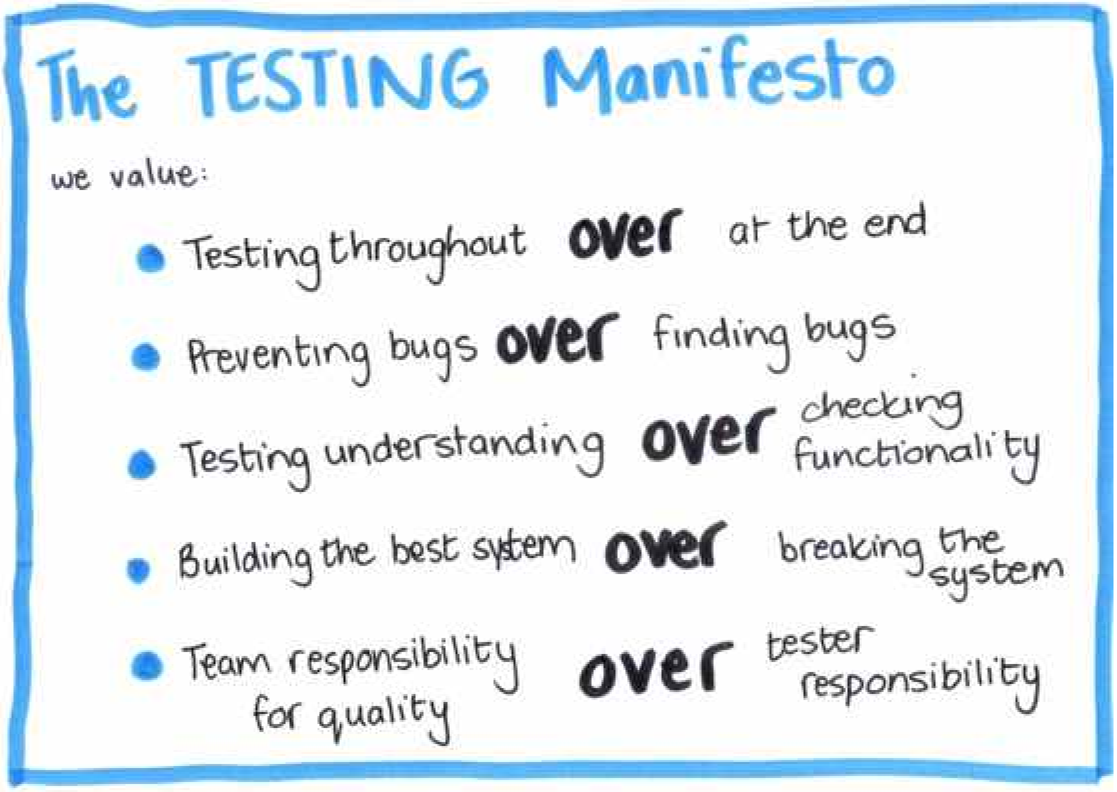
Testing Manifesto (C3)

Hand out a pack of Testing Manifesto cards to each group. Ask them to place the cards in this way: we value __ *over* __ , five times. Walk around and help with reminders if anyone seems stuck. Once all groups are done, show the slide with all five statements and quickly read through it.

Tip

If you prefer you can interchange the C2 and C3 in this topic and see if people can instinctively grasp the principles before you teach them. This can be powerful as people realise this way of thinking is just common sense. However, doing the Testing Manifesto exercise second works well as a recap exercise, and we think it helps people remember the phrases better after the workshop. Try it both ways and see what works best for you.

Slide



The TESTING Manifesto

we value:

- Testing throughout **over** at the end
- Preventing bugs **over** finding bugs
- Testing understanding **over** checking functionality
- Building the best system **over** breaking the system
- Team responsibility for quality **over** tester responsibility

Hands Up (C4)

Ask people to raise their hands if the following statement is true for them.

- Do think it might be possible to prevent bugs BEFORE you write code?

Tip

If you have played the Jenga game with the team, remind them that this is like round four in the game. Often people who didn’t think it was possible initially, change their minds after we teach this topic.

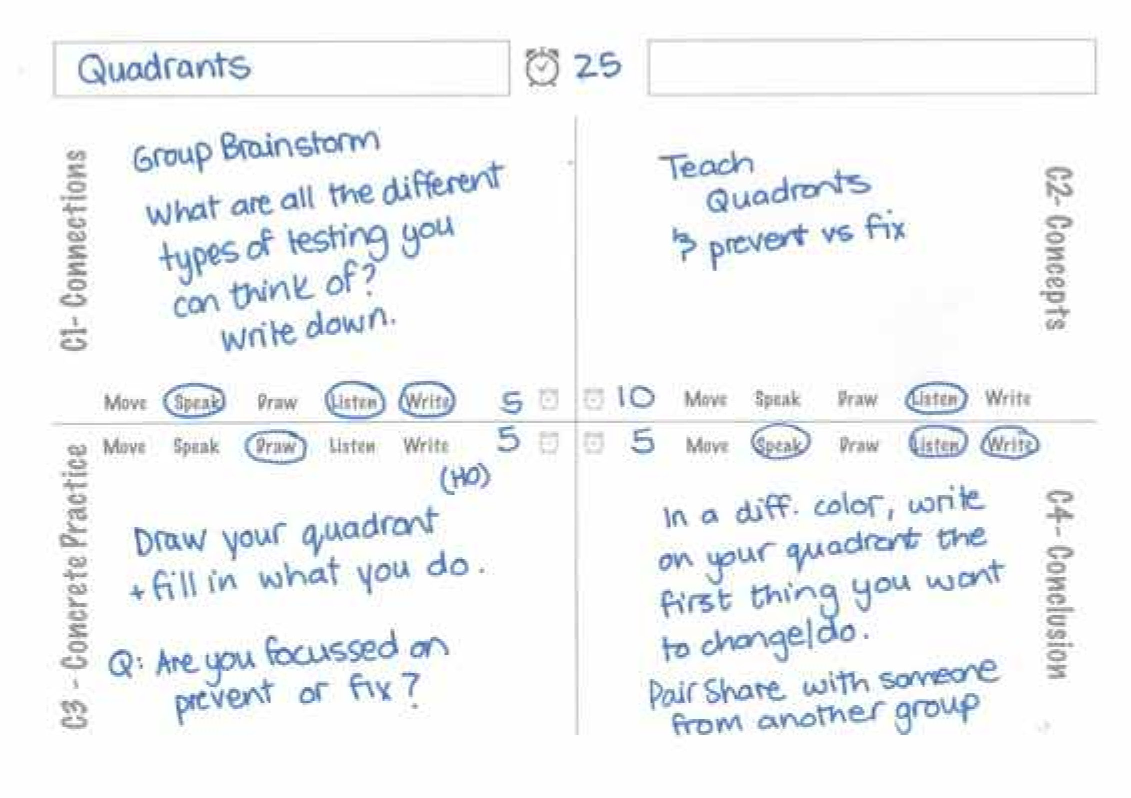
Chapter 4: Quadrants

This topic introduces the Agile Testing Quadrants created by Brian Marick. This is a useful tool to help teams think about all the different types of testing that they are or could be doing. It also helps to facilitate a discussion about how the whole team can be involved in testing activities. This is a great topic to cover with a team current doing manual functionality testing to help them see the vast possibilities of other types of testing they could do if the functional testing was automated.

Materials needed

- the Quadrant Handout
- coloured markers and pens for each table group
- paper for each table group

4Cs Training plan



Quadrants ⏰ 25

C1- Connections

Group Brainstorm

What are all the different types of testing you can think of? Write down.

Move [✓ Speak] Draw [✓ Listen] [✓ Write] 5

C2- Concepts

Teach Quadrants

↳ prevent vs fix

10 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

Move Speak [✓ Draw] Listen Write 5

(HO)

Draw your quadrant + fill in what you do.

Q: Are you focussed on prevent or fix?

C4- Conclusion

5 Move [✓ Speak] Draw [✓ Listen] [✓ Write]

In a diff. color, write on your quadrant the first thing you want to change/do.

Pair Share with someone from another group

Group Brainstorm (C1)

Give each group a piece of paper. Ask them to write down as many different types of testing as they can think of. After a few minutes ask a few people to shout out some. This gets people thinking of how many different types there actually are.

Teach (C2)

Explain that the Agile Testing Quadrants is a tool to classify different types of tests. The x-axis looks at the purpose of the tests, with one side being to support the team and the other side being to critique the product. The y-axis looks at who the tests face, with business on one side and technology on the other.

Types of tests

Explain the types of testing that happen in each quadrant. Q1 is for technical tests that support the team. Unit tests are usually the most common example here. Tests in this quadrant are usually automated. Q2 is for business facing test that support the team. Functional tests fit in this quadrant. If you are doing Behaviour Driven Development (BDD) or Acceptance Test Driven Development (ATDD) the automated acceptance tests for each feature would fit in Q2. Tests in Q2 could be manual or automated, however it is usually a good idea to have at least some automation here.

Q3 tests give feedback on the product and are business facing. Examples might be usability testing or exploratory testing. Since this type of testing is about evaluating the product critically, it is not possible to automate this kind of testing. Finally Q4 includes tests that critique the product from a technical point of view. Typical examples are performance or load testing. These usually make use of tools to help with testing.

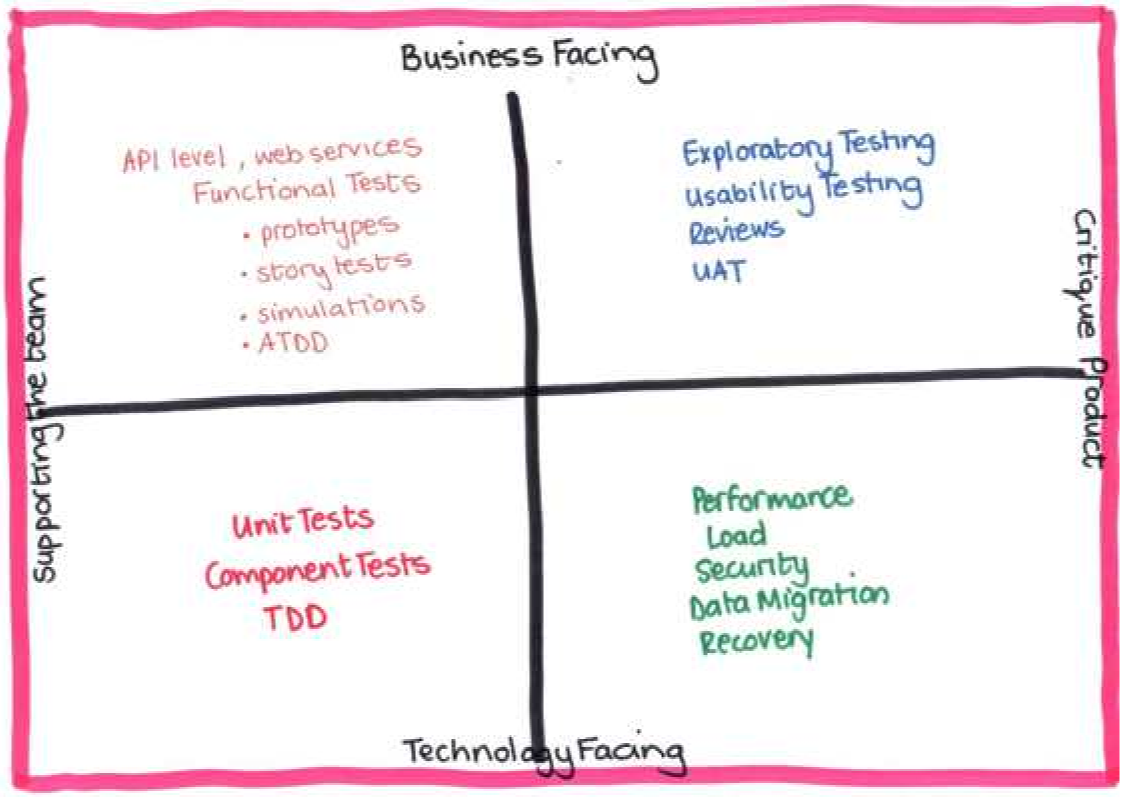
Preventing bugs versus finding bugs

If you consider the left hand side of the diagram you will notice this is the place where tests can be specified in advance of the code being written (TDD, BDD, ATDD). Test on this side of the diagram generally help prevent defects if they are done before the code. Tests on the other side usually require the product to exist before it can be critiqued and as a result, tests on the right hand side tend to be more focused on finding defects.

Tip

We have a separate chapter on automation so if you are covering both topics you might choose to not mention how automation fits into the quadrants just yet

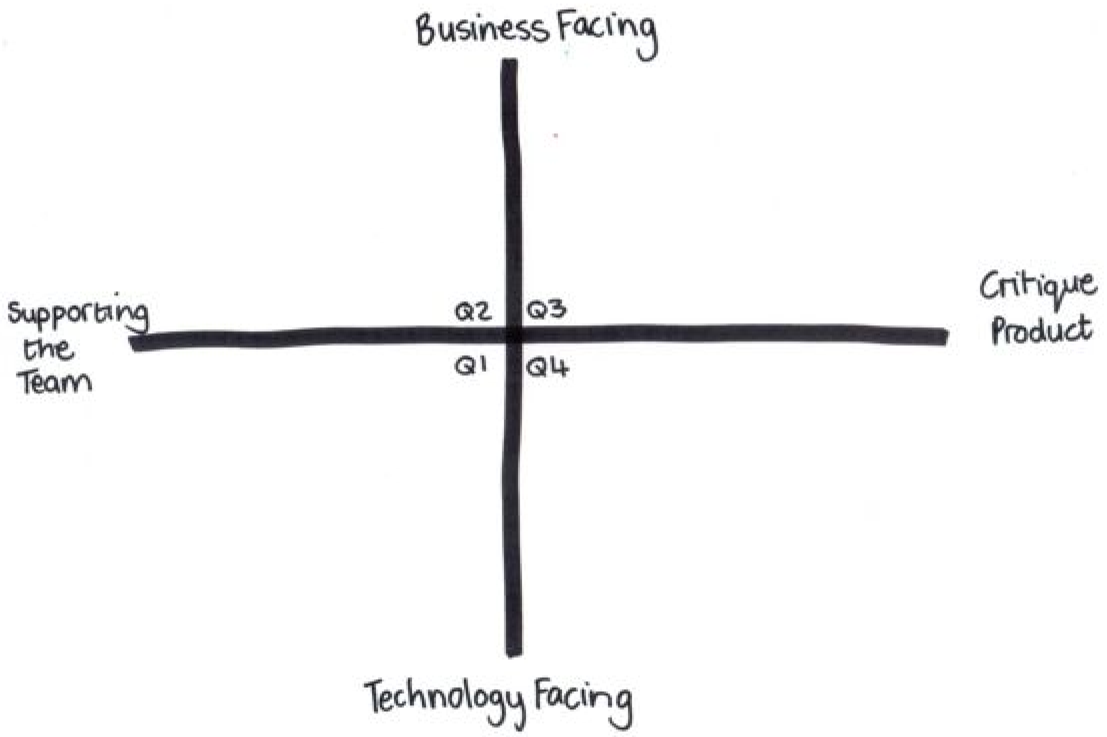
Slide



Draw (C3)

Give everyone a copy of the Quadrant Handout. You can also just ask people to draw this, if you don't have enough copies. Ask each person to fill in the types of testing they do today on the quadrant handout.

Now ask the question: "Looking at your quadrant, are you more focused on preventing bugs or fixing them?"



Pair Share (C4)

Ask people to take a different colour marker and fill in the 1 type of testing they want to start with or focus on first at work.

Now have them Pair Share their quadrants with someone from another group.

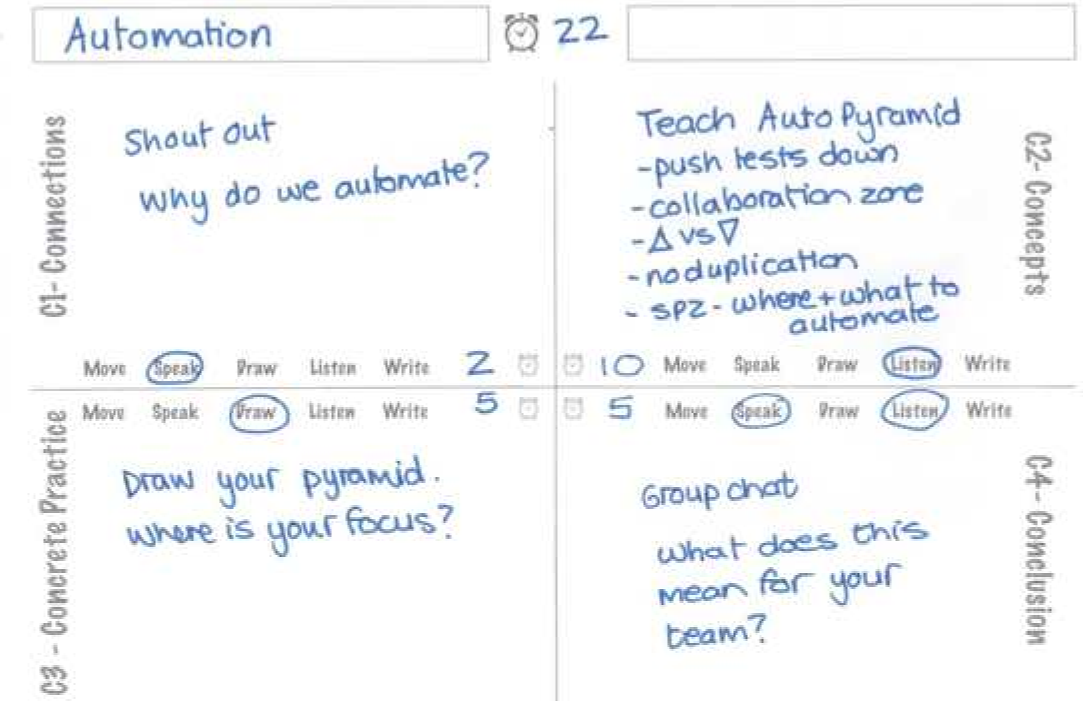
Chapter 5: Automation

It is not possible to test effectively in an agile way without automation. Yet many teams attempting agile struggle to get started with automation. This is a great topic to include in a full day course on agile testing, or in a standalone workshop with a team looking to add more automation. In this chapter we introduce Mike Cohn's automation test pyramid as a way for teams to understand how best to tackle automation.

Materials needed

- coloured markers and pens for each table group
- paper for each table group

4Cs Training plan



Automation ⏰ 22

C1- Connections

Shout out

Why do we automate?

Move [✓ Speak] Draw Listen Write 2 ⏰

C2- Concepts

Teach Auto Pyramid
- push tests down
- collaboration zone
- Δ vs ∇
- noduplication
- SPZ - where + what to automate

⏰ 10 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

Move Speak [✓ Draw] Listen Write 5 ⏰

Draw your pyramid.

Where is your focus?

C4- Conclusion

⏰ 5 Move [✓ Speak] Draw [✓ Listen] Write

Group chat

What does this mean for your team?

Shout Out (C1)

Ask people to shout out answers to “Why do we automate?” - if you like you can write these up as they shout out. The answers we’re looking for are around saving time, and making less mistakes, as well as freeing up tester’s time so they can focus on more important work.

Teach (C2)

Many teams who decide to automate their tests incorrectly think they need to automate all their tests at the user interface level. Often they take their existing manual test cases and convert then to tests using a tool like Selenium. This is a recipe for disaster. User interface tests are slow to run, difficult to maintain, and notoriously fragile.

Test Automation Pyramid

Mike Cohn’s test automation pyramid nicely illustrates a much better approach. The width of each layer of the pyramid represents how many tests should be in that layer relative to the other layers. The pyramid shows that the bottom level is made up of unit or component tests. These should be the majority of your tests. They are usually quick to run, and can be written in a robust way that prevents non deterministic tests. An example might be a test on an interest calculator class, that when passed a interest rate of x and a balance y, the interest percentage is returned correctly.

The middle level of the pyramid are tests that check business behaviour (but not via the GUI). These are sometimes called API tests, or acceptance test. If you are practising BDD or ATDD this is the level where those tests fit. Although you might need a fair amount of these tests, they should be less that the unit tests. These test that scenario's make sense from a business point of view and might run across multiple small components. An example might be that for a savings account when interest is accrued, the correct amount is calculated and added to the account balance.

The top level of the pyramid are tests that actually check the user interface. Notice that this should be much smaller than the rest of the pyramid. This is because of the reasons mentioned above. GUI tests are slow, hard to maintain and fragile. An example might be that at the end of the month after interest has been paid, my bank statement shows a line for interest with the correct amount, and my balance is updated to include this amount.

Prevent duplication

What is important about the test pyramid is to prevent duplication of tests at multiple levels. In the examples given above for the interest calculation, we might decide that the most important think to test in the GUI is how it handles large amounts. Therefore we might test a high interest rate and balance that will return over 10 digits. If we have this test at the GUI level, we do not need it at the other levels.

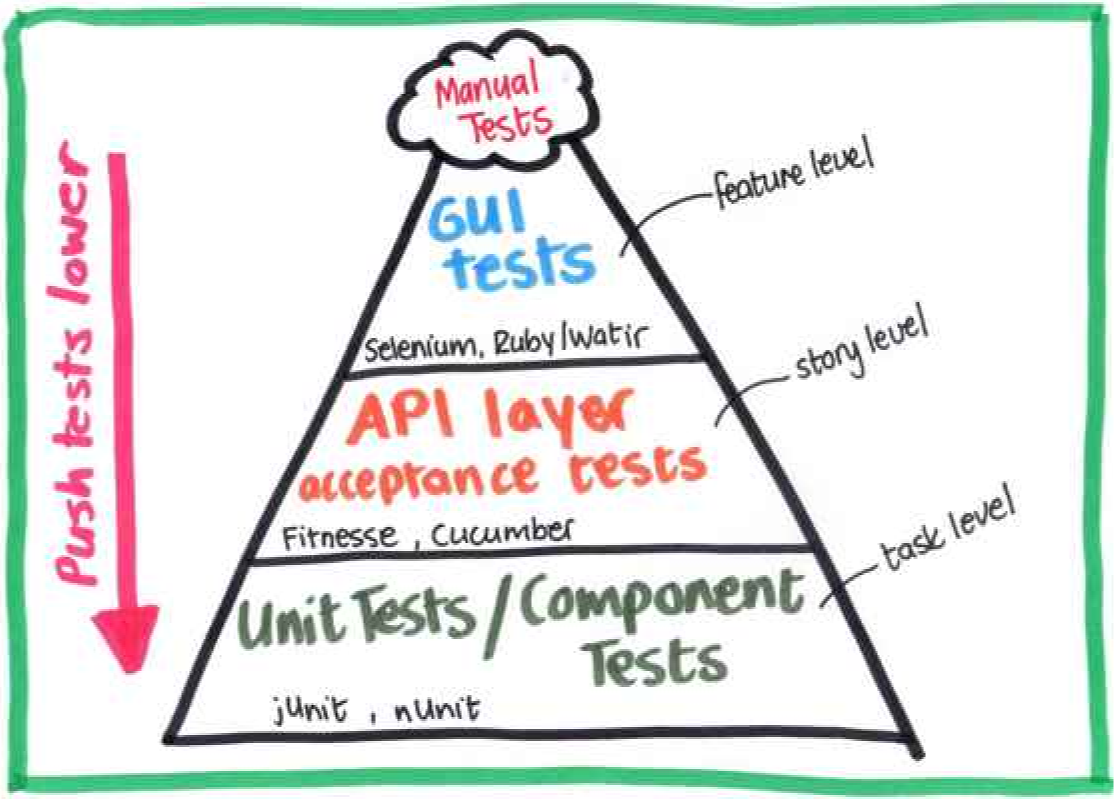
At the api level we might test a few different cases for example if we have a negative bank balance, to check that interest is debited rather than credited.

At the unit level we would do a large number of tests. All the traditional boundary values tests, negative numbers, zero's etc. A common mistake teams make is to do boundary value testing at the GUI level. This is unnecessary. Teams need to think about what the most useful thing to test is at each level, and make sure tests are not duplicated. Also where possible teams should try to push the tests lower on the pyramid.

Manual tests

Finally the pyramid acknowledges that there is a place for some manual tests. For example exploratory tests that cannot be automated. However these tests should be the exception rather than the rule.

Slide



Draw (C4)

Give everyone a piece of paper. Ask each person to draw what their automation looks like. Assist them by asking questions like: Where is your focus? Is more time spent on manual testing? Is your pyramid upside down?

Table discussion (C4)

Ask people to discuss in their groups what this means for their teams.

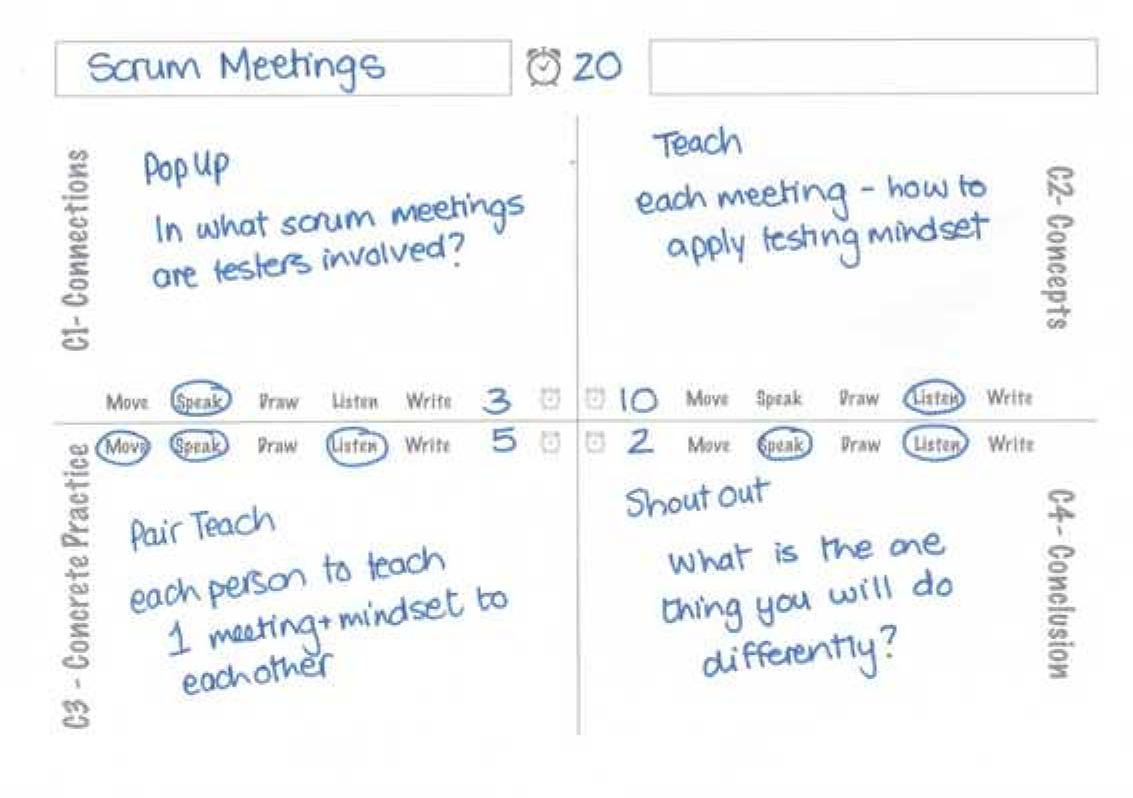
Chapter 6: Scrum Meetings

This chapter explains how to incorporate agile testing into Scrum, by looking at the types of activities that testers (or in fact everyone in the team) can be doing in each of the Scrum meetings. It’s great to include this near the end of a full day course, since it helps people think about how they will practically apply some of the tools covered in other chapters of the book. However there are enough simple techniques that you could run this workshop in isolation as a way to help teams think about incorporating testing into each part of their sprint. If the people in your workshop are not using Scrum, adapt this section to teach the activities using their process flow.

Materials needed

- none

4Cs Training plan



Scrum Meetings ⏰ 20

C1- Connections

Pop Up

In what scrum meetings are testers involved?

Move [✓ Speak] Draw Listen Write 3 ⏰

C2- Concepts

Teach

each meeting - how to apply testing mindset

⏰ 10 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

[✓ Move] [✓ Speak] Draw [✓ Listen] Write 5 ⏰

Pair Teach

each person to teach 1 meeting+mindset to eachother

C4- Conclusion

⏰ 2 Move [✓ Speak] Draw [✓ Listen] Write

Shout out

What is the one thing you will do differently?

Pop-Ups (C1)

Explain to the people in the workshop what a Pop-Up is. Then ask the question “In what scrum meetings are testers involved?”. Let a few people explain their ideas. This is to get people to connect with the topic at hand (and to move). Ideally people will realise that testers and testing should be involved in all the meetings.

Teach (C2)

Discuss each Scrum meeting in turn and explain the types of testing activities that should be happening in these meetings. This doesn’t just mean tester’s have to do them. Remember quality is the whole team’s responsibility so anyone in the team can help with these.

Backlog Grooming

The most important testing activity in grooming is to ask questions and help to surface any assumptions. This is where defects can be prevented. It is important to understand why a feature is needed, as well as how it can be tested. If there are many different ways to interpret something, it’s good to discuss all the options to make sure everyone in the room has the same picture in their head. Since Backlog Grooming is primarily about understanding requirements, some of the techniques that are useful here (like mind maps and steel threads) are covered in our Agile Requirements book from this series.

Tip

If you are covered the Agile Testing Mindset with the same group, you can remind them about the star example.

Sprint Planning Part 1

The goal of Sprint Planning Part 1 is to understand the requirements fully. This is a great place to confirm which acceptance tests are important to the Product Owner, and making sure everyone knows how the story will be tested to make sure it meets these criteria. Beware of vague criteria like “it must be fast”.

It can be useful to take a look at the Agile Test Quadrants in Sprint Planning 1 if you are struggling to decide how a story will be tested. Another important test consideration at this point is to understand if any regression testing will be needed . You don’t need to discuss the details yet, you can do that in Sprint Planning Part 2, but it’s important to know what areas the new feature interacts with and what is important to confirm as still working, especially if this feature might change the behaviour of another feature.

Sprint Planning Part 2

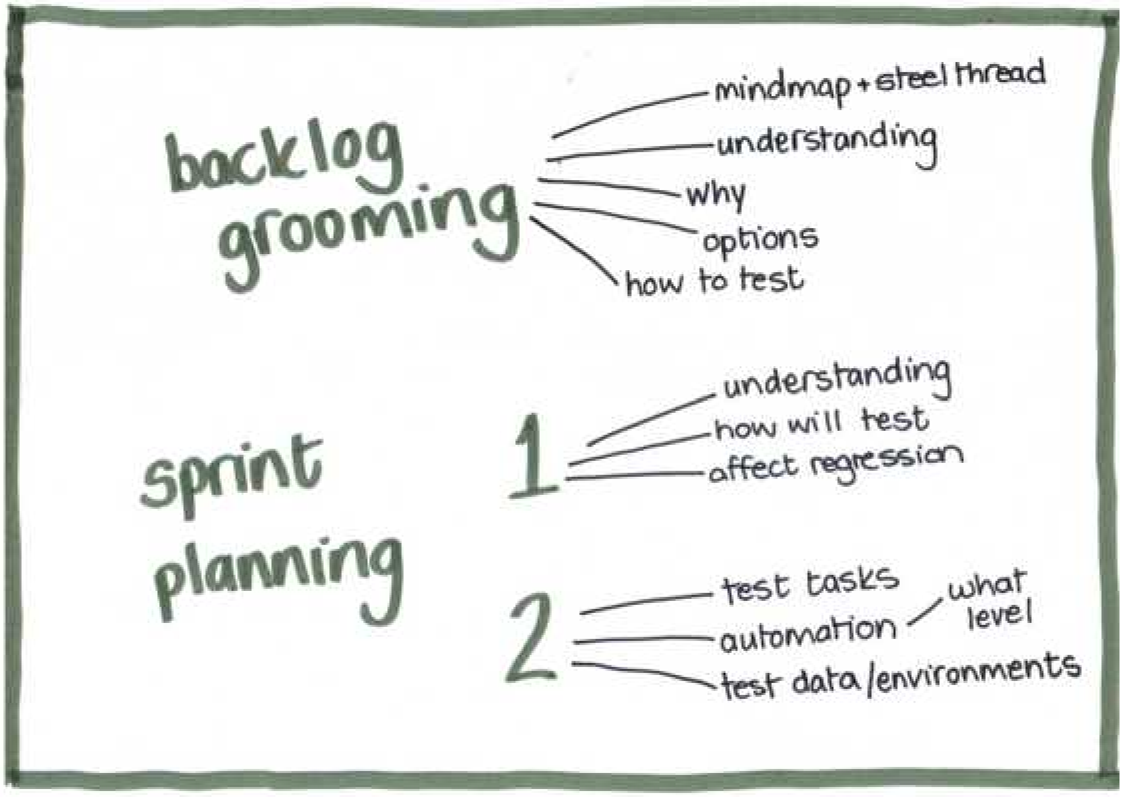
The goal of Sprint Planning Part 2 is for everyone on the team to have a shared understanding of how they will achieve their commitments. Too often developers discuss the coding to be done and testers discuss the testing. However considering quality as a whole team responsibility it is important that developers and testers discuss together what all the tasks are.

It is useful to discuss the Automation Test Pyramid in this meeting and make sure everyone is clear which tests will be covered at a unit level, which will be covered at an API level and which, if any, will be covered at a GUI or manual level. This conversation between developers and testers is crucial to prevent duplication of tests. Teams could even discuss which tools would be best for each type of test. For example JUnit might be an easy way to do API level tests, even though it is considered a unit testing tool.

It can be helpful to discuss any required test data and test environments in this meeting. Often testers require a complex data set. Developers can help create these using scripts or harvesting data from production. If performance testing needs to be done, it's important for everyone to understand what environment it can be run on.

Making sure everyone understands how quality will be ensured is a great way to help teams take ownership for quality. Sprint Planning Part 2 is the best place for this discussion to take place.

Slide



During the Sprint

Often before agile testing, testers find themselves with very little to do at the start of the sprint, and overwhelmed at the end. That should not be the case is they adopt an agile testing mindset. Using the review or “Show Me” column can help everyone start reviewing tasks as soon as the first task is completed.

It’s also important for bugs to be fixed as soon as they are found. This reduces the cost of fixing the bugs since there is less context switching. If a tester finds a bug in code the developer worked on 30 minutes ago, it will take them much less time to fix it, since the code is fresh in the developers mind. If the developer only gets around to fixing that bug two weeks later, it will almost certainly take longer.

The Definition of Done is the team’s way to ensure quality on each story. It’s up to each team member to ensure each point in the Definition of Done is met before they consider a story complete. It’s very easy to forget that last code review task if it is the only item left on a story. It’s up to the whole team to hold themselves accountable and make sure it gets done.

The best use of testers early in the sprint is to pair them with developers as they are starting to code. Testers can think about all the tests that need to pass, and the developer can make sure those tests are implemented.

Daily Scrum

The Daily Scrum is a good time for stupid questions. It’s often when people make assumptions. Tasks might be taking longer than expected because someone has assumed it needs to be done in a certain way. Question that. If no one remembers what a particular task was about question that.

Looking for things people are confused about is a great way to prevent bugs. Listen for phrases like “I thought…” or “I expected…”. This usually means reality didn’t turn out like their assumption. Question it to understand the assumption and make sure everyone in the team is clear on it to prevent future confusion.

The Daily Scrum is a great opportunity to see how much testing can be done before development. If you use different coloured sticky notes for test tasks, then each day you can look at the taskboard and see is testing is lagging behind development, or if testing is happening in advance. Whenever possible see which test tasks can be done first.

Tip

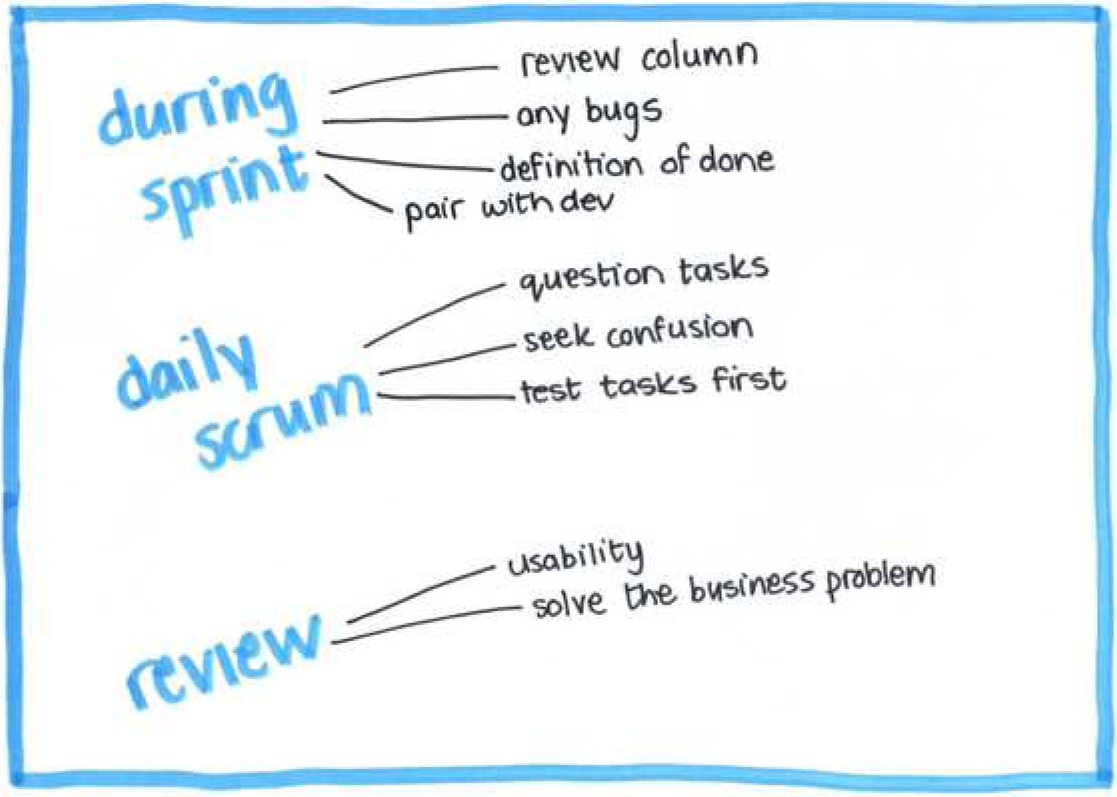
If you have played the Jenga game with the team remind them that it does not take more time to test earlier, and in fact makes them much faster in the long term.

Review

The Sprint Review is a great opportunity to execute tests from Q3 on the Agile Test Quadrants. For example usability or User Acceptance testing. Instead of demonstrating the feature, get real users or stakeholders to use the product and then take notes about what you learn.

This is also the place to confirm that the solution actually solves the business need, and to question users to see what their expectations are about how this feature might be enhanced in the future. This all helps become closer to understanding your users' needs, which is critical for agile testing.

Slide



Teach Back (C3)

This topic includes a lot of ideas in a short time, so follow it with a pair teach-back to reinforce the ideas. This is similar to a Pair Share, each person explains a concept to the other. Ask one person to explain testing activities that should happen in Backlog Grooming and Sprint Planning, and the other to explain testing activities that should happen during the sprint.

Shout Out (C4)

Ask people to shout out answers to “What is the one thing YOU will do differently” - listen to around 5 people to get some variety. The question will get everyone thinking and that is the aim with this exercise.

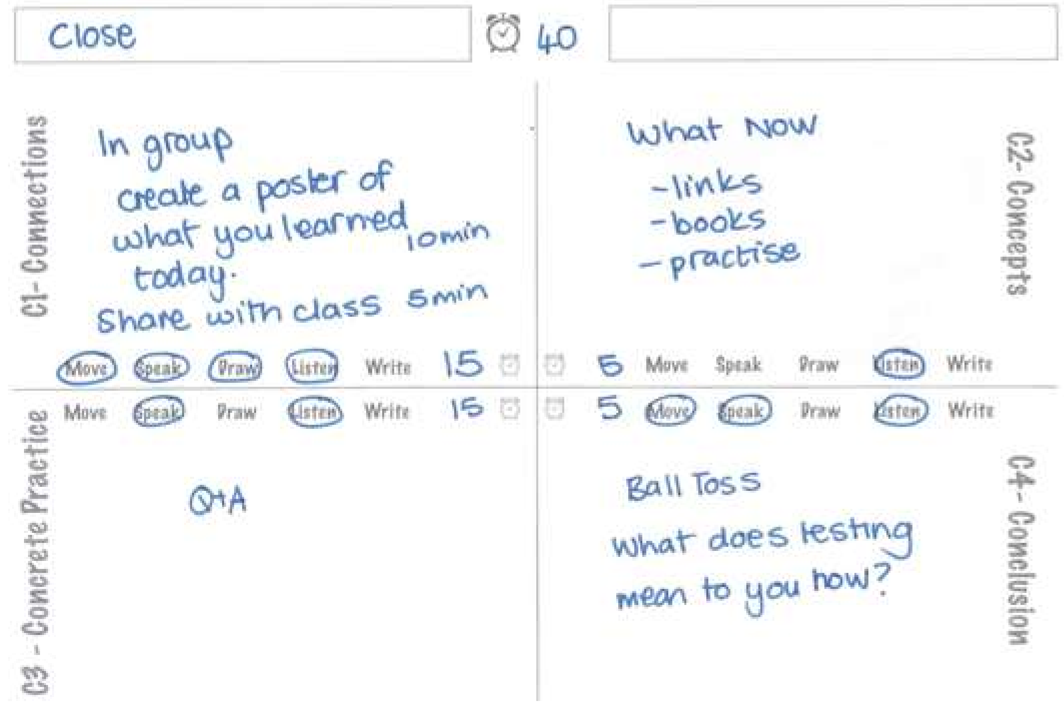
Chapter 7: Close

This topic deals with closing the workshop. It’s the time to remind the class of what they have learned, where they can get more learning from and finally to end the day. If you are running each topic as a separate workshop you probably don’t need to do this full close each time, rather incorporate some of the activities from here in the C4 of the topic.

Materials needed

- coloured markers and pens for each table group
- sticky notes
- flipchart sheet or cardboard for each table group
- a soft ball

4Cs Training plan



Close ⏰ 40

C1- Connections

In group
create a poster of
what you learned today. 10min
Share with class 5min

[✓ Move] [✓ Speak] [✓ Draw] [✓ Listen] Write 15 ⏰

C2- Concepts

What Now
-links
-books
-practise

⏰ 5 Move Speak Draw [✓ Listen] Write

C3 - Concrete Practice

Move [✓ Speak] Draw [✓ Listen] Write 15 ⏰

Q+A

C4- Conclusion

⏰ 5 [✓ Move] [✓ Speak] Draw [✓ Listen] Write

Ball Toss
What does testing
mean to you now?

Review Poster (C1)

Hand out a sheet of flipchart paper or cardboard to each table group. Ask them to create a poster of what they have learned about Agile Testing during the workshop (see slide below). Depending on time and the number of groups, you can either ask each group to present their poster, or you can ask people to put up their posters and do a Gallery Walk.

Slide

What Now (C2)

Here you can talk about other learning materials available for Product Backlogs. We usually list our blog - you might want to add yours. Talk about other training courses, blogs and books you recommend. Also mention any upcoming conferences in the area and worldwide.

For more links to great blog posts on this topic please go to Agile Testing on our website.

Slide

Question & Answer (C3)

Ask each group to write down any questions they still have and prioritise them. Allow 3-5 minutes for this. Then go round and answer 1 question for each group until your time runs out. As this section is almost the end of the workshop, you can cut it out if you run out of time. In that case, simply ask the class to email you questions if they have any. Should you find yourself with extra time, you might be able to answer all questions. We usually limit each answer to 2 minutes. This allows us to get to the point and for each group to get some questions answered.

Ball Toss (C4)

To close you can choose to do Pop-Ups or a Ball Toss. We usually do a Ball Toss with a smaller group (<25 people) and Pop-Ups with larger groups.

Explain the technique you are going to use and then ask people what testing means to them now. Be aware that some people might take some time to share, so this is a good time to practise being comfortable with silence. Use your judgement on when to end it as not everyone will always speak. Once you are done, thank people for their time.

APPENDIX

Agreements

⏱	👥	🎭
5 minutes	any number people	Communication

What you can learn

This sets a tone and expectations near the start of a session. It helps the attendees know what the boundaries of the session are, and what behaviours are acceptable.

What you need

It is best to have each agreement on a card and to go through them near the start of the session.

How to do this

Decide which agreements are appropriate for your audience and meeting. Explain them clearly and simply near the start of the session.

You can also ask participants if there are any agreements they would like to add.

How we’ve used this

We change these depending on the session we’re running. Over time you will learn more techniques and so this list will keep evolving.

Here are some of the cards we have:

- **Take Care:** Take care of your own needs. You don’t need to ask permission to go to the bathroom, or get coffee.

- **Cellphones:** Keep your phones on silent please. If you need to take a call, just leave the room. We'd rather you were paying attention than worrying because your boss/wife/child is calling.
- **Right to Pass:** You have the right to pass in any activity or exercise we do. Just sit to the side and observe.
- **Workbooks**: These are yours to keep. Please take notes. We will let you know when we are doing specific exercises in the books.
- **Timeboxing:** We give a specific end time for each break. We will start at that time whether you are back or not. It's up to you to choose to be on time or not.

Who shared this with us

Various people over the years, many from Sharon Bowman. We came up with the concept of using cards to remember all of the things we wanted to say.

Fast Pass

⏱	👥	🎭
10 - 15 minutes	6 - 20 people	Movement Trust

What you can learn

An activity to connect participants to each other through content related to the session. This is a great technique to use at the start of a session, so people who arrive early have something to do.

What you need

Flipchart pages stuck up on a wall, with questions. Have a minimum of three (for six participants) and a maximum of five (for 20 participants).

Some questions might be:

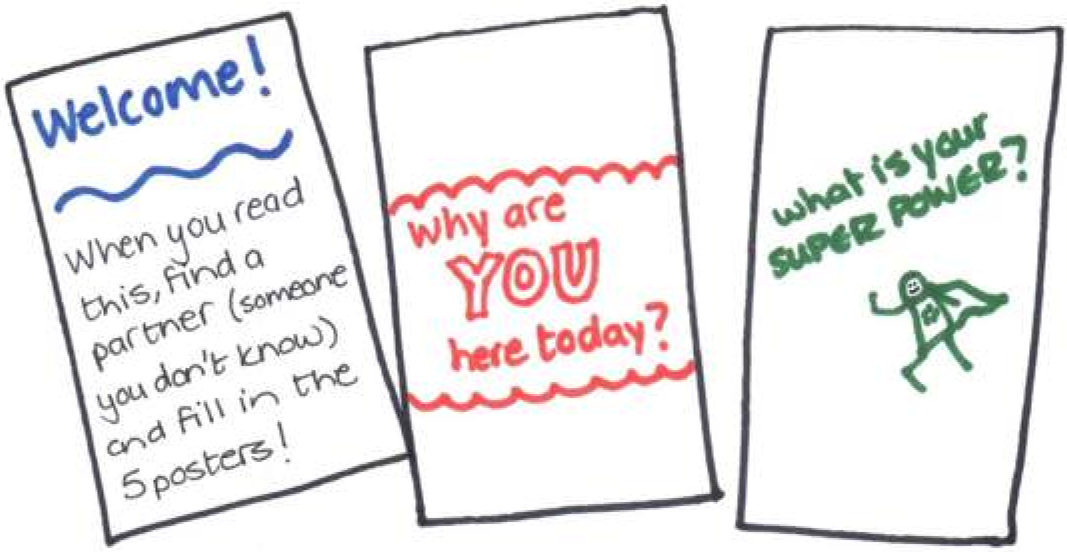
- What are your pets' names?
- What do you know about <topic of="" session="">?</topic>
- Why are you here today?
- What is your biggest strength?
- What is your company's greatest challenge?

Instruction flipchart:"After reading this, introduce yourself to a stranger and fill in the flipchart questions around the room with them."

Marker for each participant.

How to do this

At the start of a session stick up the prepared flipcharts around the room and place the instruction flipchart near the front of the room.


Welcome!
When you read this, find a partner (someone you don't know) and fill in the 5 posters!
Why are YOU here today?
what is your SUPER POWER?

Encourage people to read the instructions if they don't notice them, and let them know they can start whenever they like.

How we've used this

We often use this at the start of training courses, or large group meetings, especially if people don't know each other. It is a great way to get strangers talking at the start of the day.

Who shared this with us

Sharon Bowman

Jenga

30 minutes	4 - 30 people	Testing Quality

What you can learn

This game is a great way to teach people the benefits of testing early and in smaller batches, before development is complete. This simple game demonstrates how it is actually faster to test earlier.

What you need

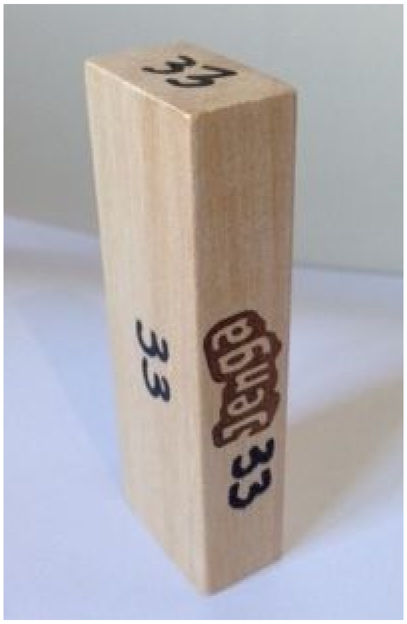
- 36 Jenga blocks per group of 6, number each block from 1 to 36. Be sure to write the number with a permanent marker on each side of the block (i.e. 6 times per block).
- Handouts with 4 bug numbers for each round. Pick any 4 numbers between 1 and 36 for each round. For example: 2, 17, 21, 36. You need one copy of the numbers for each group.
- A stopwatch or timer
- A flipchart or whiteboard to write up the results

Note

A Jenga set has 54 blocks per set, so if you buy 2 Jenga sets you will have enough blocks for 3 groups (i.e. 18 people). Try to buy a light coloured wooden set because you need to write numbers on the blocks.

How to do this

Split people into groups of 4 to 6. Try to have the same number of people in each group. Explain that the goal is to build a tower with Jenga blocks. The requirements are that the tower must use all 36 Jenga pieces and must be at least 4 Jenga blocks high, with the blocks stood up on the small end - see the picture below.



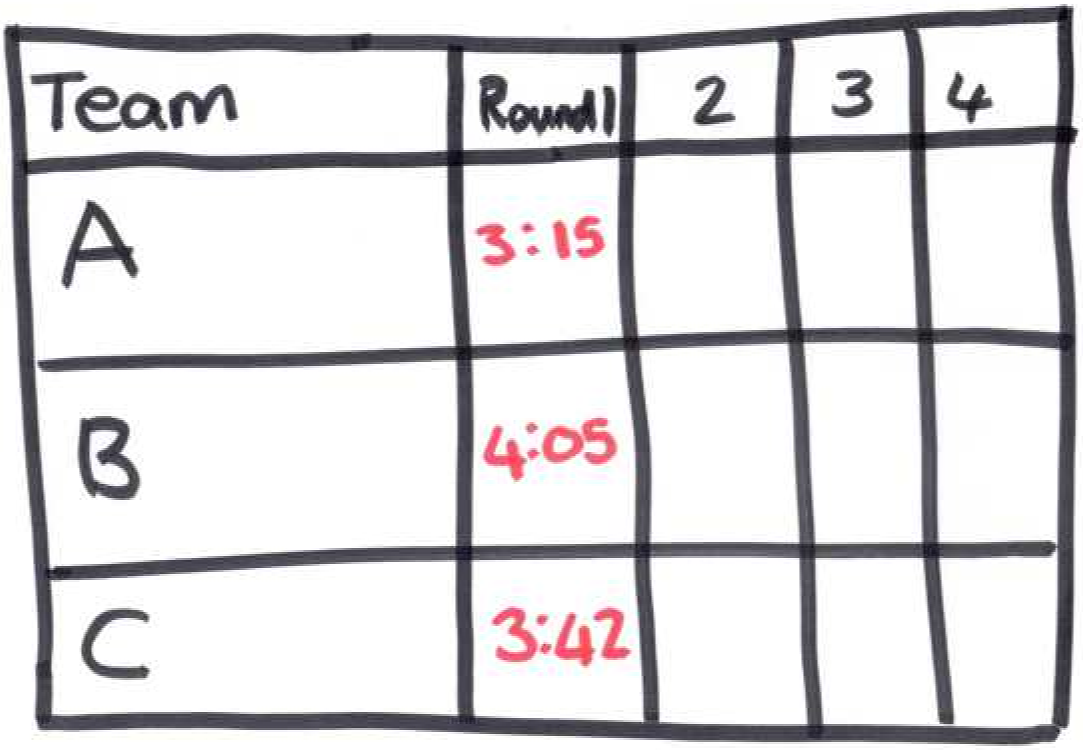
The tower does not have to be built like it is in Jenga when the game starts. Some people assume this, if they do, don't correct them. It leads to a great teaching point since it makes fixing the bugs particularly hard. If people ask tell them they can build the tower however they want as long as it meets the requirements.

Round 1

Before the teams start to build ask each team to nominate two people on their team to be testers. Ask the testers to come forward and brief them separately. Hand the testers from each group the list of 'bugs' for the first round. This is just a list of 4 numbers between 1 and 36. Explain that they can't show the bugs to the builders on their team, but once they have finished building the tower, the testers must find these 'bugs' and remove those pieces from the tower. If the tower collapses when they are removed, it needs to be rebuilt. The end result needs to meet the original height requirement, but the tower must be built from only 32 pieces (i.e. the original 36 without the four 'bug' blocks).

Once the testers return to their teams, let everyone know they can start and that they should let you know when they are complete. Start a timer at this point, so you can track how long it takes for each team to finish. Make sure testers don't reveal the bugs until the end. Once people finished, capture the time it took for each group. Only take the time once the tower is complete without the 'bugs'.

You can write the results on a poster like those shown below.



Team	Round1	2	3	4
A	3:15			
B	4:05			
C	3:42			

Round 2

Ask everyone to breakdown their towers for round two. Again give the testers four numbers for the 'bugs'. This time tell them that they can test the tower after the builders have placed nine blocks. They still can't show the team the bug numbers, but after the builders have placed nine blocks, they can tell them if any of the blocks are bugs. The builders can then remove them and continue to build. Once against start a timer as teams begin, and update the times on the score sheet for round two. In most cases the time should be less than the time for the first round.

Round 3

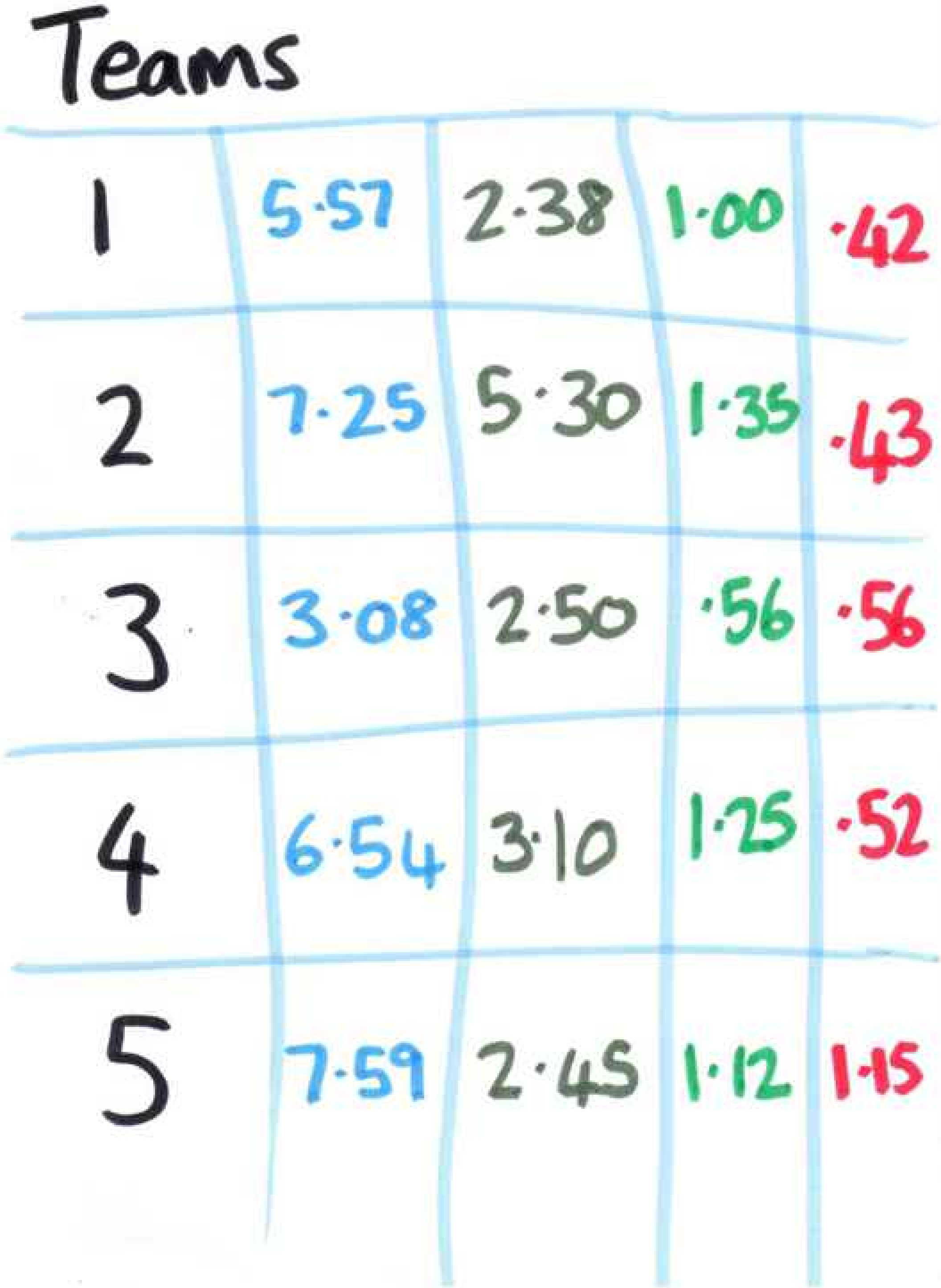
Again ask people to breakdown their towers. Hand out the bugs to the testers. Make sure you use different numbers for each round so that builders don’t start to guess what the bugs will be. It’s okay for teams to all get the same bug numbers though. This time let people know that the testers can check after each block is placed, and that block can be removed immediately if it is a bug. Again keep track of the time. It should now be significantly less that the first round.

Round 4

Let people know this is the final round. Sometime people can start to wonder how many times they need to build a tower at this point. Ask people to break down their towers, and hand out the bugs. This time tell the testers they can share the bug numbers with the builders at the beginning, and that they can use that information however they like. Most team immediately remove the four bugs and then build the tower. Double check though because occasionally team forget to put the bugs aside and end up including them in the tower even though they knew they are bugs in advance. Write up the time for the final round.

Debrief

Now debrief the game. Start by asking people what they notice with the times. In our experience the times for the last two rounds are fractions of the original time. Below is a copy of the results from one of the times we have run this game.



Teams				
1	5·57	2·38	1·00	·42
2	7·25	5·30	1·35	·43
3	3·08	2·50	·56	·56
4	6·54	3·10	1·25	·52
5	7·59	2·45	1·12	1·15

Some other questions which are useful for the debrief are:

- Which round feels like how you work today?
- Which round is how you would like to work?

- How did it feel to fix bugs in the first round?
- How did it feel to fix bugs in the other rounds?
- Do you think testing early takes more time, if so do they results change your mind?
- Do you think round four (knowing the bugs in advanced) is possible in software?
- In which rounds were testers more involved? ###How we’ve used this We use this simulation whenever we introduce agile testing as a concept to a team. It demonstrates really easily how much of a time saver it is to test early. It’s a great way to start a workshop on agile testing. We especially love the last round (our own addition to the game), which helps people think differently about testing, and introduces the agile testing principle of preventing bugs rather than finding them. ###Who shared this with us Nanda Lankalapalli

Pair Share

⏱	👥	🎭
2 minutes	any number people	Movement Speaking

What you can learn

A great way to get people to talk about their thoughts, or to recap what they have just learned. You can use this early on before trust is established as it is easy for people to share with only one other person. It’s also an easy way to introduce movement into any session.

What you need

Nothing.

How to do this

Explain that Pair Share means that you need to find a partner and share with them. After a short time swap and have the partner share with you.

Ask people whatever question you want them to discuss, then ask them to find a partner and Pair Share.

If you want to include movement ask people to pair with someone from another table.

How we’ve used this

Ask people to share an action they will take after a session. It is easy to share this with one person, rather than a whole group.

Use this early on in public training to help connect people who don’t know each other.

Use this to recap a section of training by asking people to share what they have learned in a particular section.

Who shared this with us

Sharon Bowman

Pop-Ups

⏱	👥	🎭
1 minute	any number people	Movement

What you can learn

An easy way to introduce movement into any session.

What you need

Nothing.

How to do this

In any section where you want to introduce movement, and you need to get individuals to speak, introduce Pop-Ups. Explain that a Pop-Up means that you need to stand up before you speak. Ask people whatever question requires input from them. Remind them to stand before they speak. Often this results in a Shout Out without standing. To counter this we act dumb and ask “Sorry, what was that?”, “I can’t seem to hear you?” until the person realises and stands up. Also good for a few laughs in the class.

How we’ve used this

You can ask questions where people have to stand if they think the answer is yes. Use Pop-Ups to brainstorm ideas. Ask a question like, “What are some aspects of a great team?”

Who shared this with us

Sharon Bowman

Standing Survey

5 - 10 minutes	any number people	Visualisation Movement

What you can learn

This is a great technique to introduce movement into a session as well as visualising information.

What you need

Decide what questions you will ask, and how you will ask people to arrange themselves in the room.

Having some open space in a room without tables and chairs is useful.

How to do this

Ask people to stand. Explain that you want them to organise themselves in the room according to some criteria (e.g. amount of Scrum experience).

Explain how to organise themselves (e.g. a single line, with no experience near the door, and most experience near the other side of the room).

Allow time for people to move around the room.

Remind people to speak to others to see where they should stand relative to each other.

Ask people to notice where other people are relative to them.

How we’ve used this

Some ideas for criteria to organise by:

- how easy you think something will be to implement (easy: one side of the room, impossible: the other)
- how well you know people in the room (close to those you know, far from those you don’t)
- people’s roles within an organisation (a quadrant with a different role in each corner of the room)
- where people are from (in the centre: close by, edges of the room: far away).

Who shared this with us

Lyssa Adkins

Gallery Walk

⏱	👥	🎭
5 - 15 minutes	6 - 50 people	Movement Sharing

What you can learn

An activity to allow participants to learn from each other by sharing information created on posters. It is also a good way to introduce movement into a session.

What you need

- masking tape to stick up posters
- posters created by each participants in a previous exercise.

How to do this

Ask people to stick up their posters around the room. Make sure each poster is easy to get to, especially if there is a large group.

Tell people they can now walk around the room and look at the posters created by the other groups. If people have questions about posters they can ask those who created them. Let participants know how much time they have, and end the activity once people stop looking at the posters and just start chatting.

How we’ve used this

Use this technique right after a section where people created posters with their personal input.

Who shared this with us

Sharon Bowman

Ball Toss

⏱	👥	🎭
5 minutes	5 - 25 people	Movement Feedback

What you can learn

A good way to close a session with some movement, and to give people an opportunity to share their thoughts and give you some feedback.

What you need

A soft ball.

How to do this

Ask people to stand up and form a circle. If you have the space to do this away from tables then do so, but if not create a circle around the tables.

Explain that the ball is the speaking token. If you would like to speak, signal to the person with the ball and they will throw it to you. They can then answer the question you have posed.

Remind people that once they have spoken they should look for someone else signalling that they want the ball. Often people get flustered and just throw the ball when they are done speaking.

Ask people a question about the session. Wait for someone to ask for the ball, then throw it to them. Once the ball stops moving for a while, you can signal for the ball again, and close the session by thanking people for their time.

How we’ve used this

Use this to close the day on a one-day training course, by asking people to share one word that best describes the day for them.

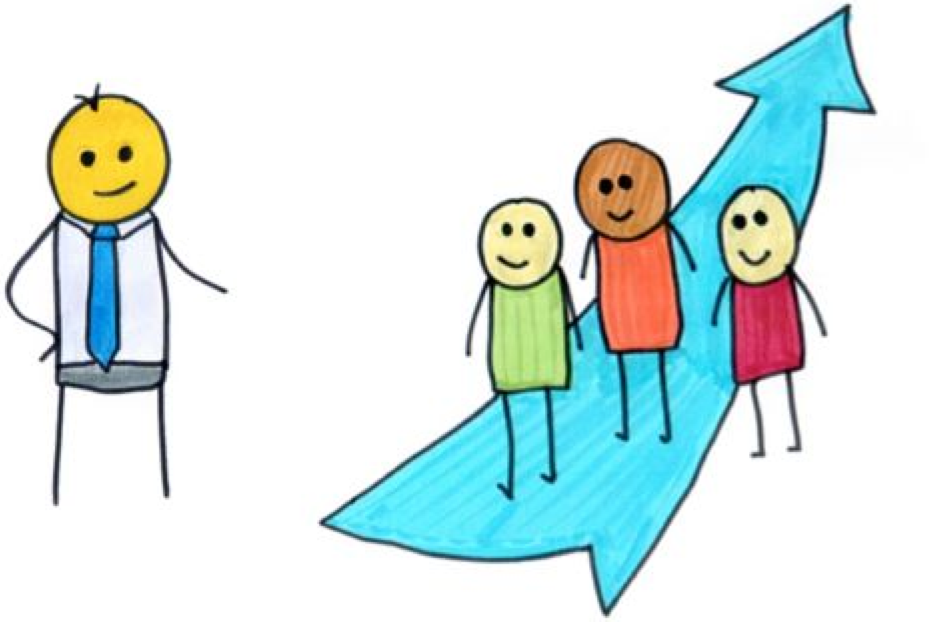
Use this to close retrospectives by asking people what their hope is for the next sprint.

Who shared this with us

Sharon Bowman

Growing Agile Online Courses

Agile Management - The art of servant leadership



Great leaders aren't born, they're grown. They cultivate their own skills over time. Become a great leader with this course!

The best leaders magnify their team's productivity. They inspire their teams to always be better, always be learning, and always solve issues before they become problems. They cultivate collaboration, not competitiveness. They help everyone combine their diverse talents and skills into innovative solutions and better products.

At the end of this course you will:

- Be a better leader
- Facilitate more effective, shorter meetings
- Give actionable feedback

- Build trust and respect within your team
- Recognise your own strengths and weaknesses, and improve on them

This course covers several topics. We recommend focusing on one topic per week. Each topic has a video and exercises to help drive the lessons home, as well as resources to explore further. Finally, for each topic, we give you a practical technique to implement in your business.

Coursework takes approximately 3 hours per week, for 8 weeks.

We give you our personal guarantee that this course will take you to the next level as a leader. If not, you have 30 days to get a full refund, no questions asked, from Udemy.

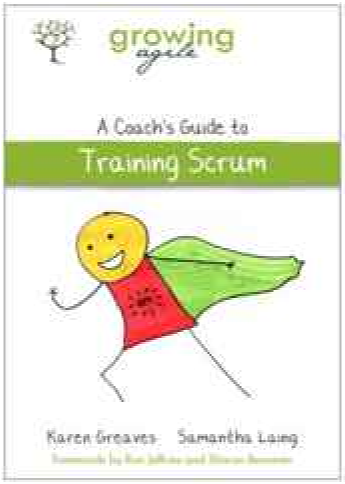
The course is available online on Udemy.

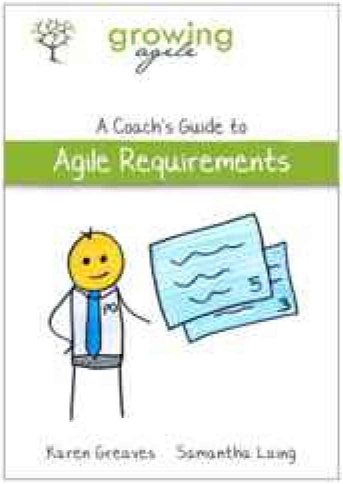
Growing Agile Books

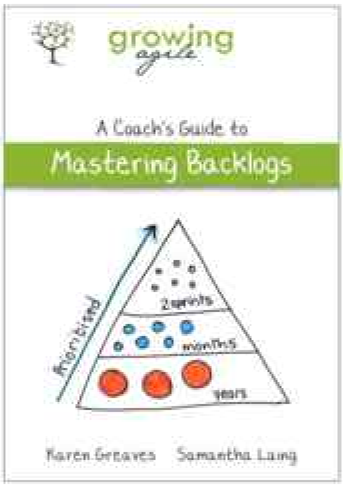
The Growing Agile Coach's Guide Series

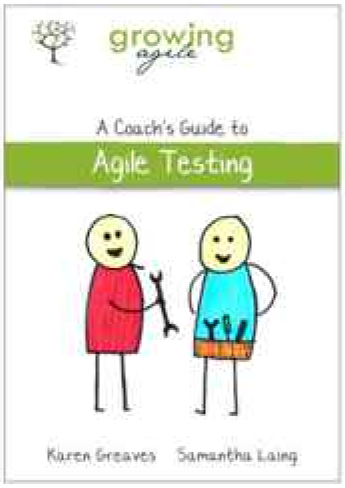
This series provides a collection of training and workshop plans for a variety of agile topics. The series is aimed at agile coaches, trainers and ScrumMasters who often find themselves needing to help teams understand agile concepts. Each book in the series provides the plans, slides, workbooks and activity instructions to run a number of workshops on each topic. The interactive workshops are all created using techniques from Training from the Back of the Room, to ensure participants are engaged and remember their learnings after the workshop.

The series is available in a bundle on Leanpub, else you can purchase the books individually.









Growing Agile: A Coach's Guide to Training Scrum

Part of the ScrumMaster role is to ensure that everyone on their team is educated about Scrum and to evangelise Scrum to their organisation. Over the past few years we have come across many ScrumMasters who have great intentions of running training but then get bogged down in the planning and preparation and don't ever get round to actually doing it.

We have been training teams in Scrum for about three years. Over the past year we have trained Certified ScrumMaster classes worldwide. During this time we have spent many hours preparing training plans and creating workbooks, flipcharts and slides. Our materials have been continually refined from feedback after each course. All our training uses Training from the Back of the Room principles.

This book will help you plan and deliver interactive, fun Scrum training for anything from a short workshop on a particular topic to a full two-day course.

Growing Agile: A Coach's Guide to Training Scrum is available on Leanpub

Growing Agile: A Coach's Guide to Agile Requirements

Have you ever looked at a requirement document thats 50 plus pages long and wondered if there was an easier way to communicate? Or have you ever received a 1 liner story without any context and thought 'huh?'.

We have and in our years of work in traditional waterfall organisations and newer agile organisations the trend continues. Very seldom is the sweet spot in the middle achieved. In every single case the path to the sweet spot started with conversations.

Over the last 3 years we have developed a number of workshops to help people start these conversation. The workshops are aimed at different stakeholders raging from business, to Product Owners and teams. This book is a collection of some of those workshop and can be used to help improve the way you think about and communicate agile requirements.

Although we talk about Product Owners, this book isn't just for Scrum teams. You can apply most of the lessons in here to any project you are on. We have used these techniques and ideas for renovating houses, redesigning websites and even writing this book!

Growing Agile: A Coach's Guide to Agile Requirements is available on Leanpub

Growing Agile: A Coach's Guide to Mastering Backlogs

Scrum is great for getting teams to deliver good quality software regularly, but that doesn't really help if you are building the wrong thing! The Product Backlog is a key artefact to helping steer the team in the right direction. Yet sometimes the Product Backlog is just a long list of tickets logged by the business. Often Product Owners can't see the forest for the trees and there are so many items in their backlog the team have no idea what direction the product is actually headed. Even if you know what your backlog should look like, finding the time to get it in order seems impossible.

If this sounds familiar, this book is for you. We have worked with a number of business analysts and Product Owners who feel the same way. We found that sending Product Owners on 2 day theoretical training courses is not the answer. Instead we run short workshops where we work with the Product Owner's actual backlog. The workshop is a working session, and an hour later the Product Owners emerge with an improved backlog.

We have combined a number of these workshops into this book. We provide all our workshop plans, tips for facilitation, and teaching points to cover for each topic. Use these workshops to help the Product Owners you work with to master their backlogs.

Growing Agile: A Coach's Guide to Mastering Backlogs is available on Leanpub

Growing Agile: A Coach's Guide to Release Planning

We often hear people say "We're agile, we don't need a plan"! or even worse "We can't plan". This is just not true. Release Planning in agile is as important as it is in traditional projects, the only difference is there are a few techniques that help make sure the plans bear some relation to reality. We've all worked on projects where you know from day 1 you will be late, and yet no one does anything about it. Agile planning is different. We give up the illusion of control of traditional rigid plans, and replace it with a clear view of where we actually are, even if it's badly behind schedule, so that together we can make decisions based on what is actually possible.

We have run Release Planning workshops with many organisations. Sometimes starting with C-level executives to set the direction and vision for a product suite, other times starting with teams and making sense of what they are currently working on. This book is a collection of our workshops that will help you run similar workshops to create agile release plans. We include teaching points on a range of techniques like Story

Mapping and release burnups to help you explain to other's how to use these methods effectively.

Growing Agile: A Coach's Guide to Release Planning is available on Leanpub

Growing Agile: A Coach's Guide to Agile Testing

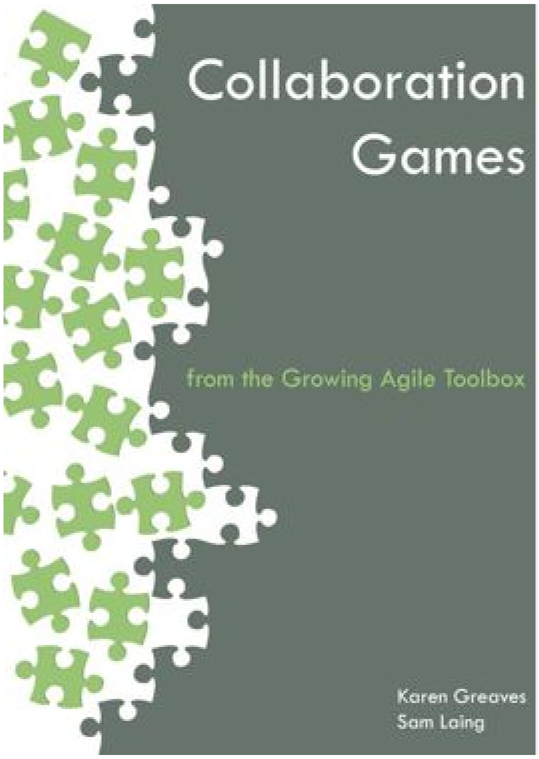
If a team believes they are agile, but nothing has changed about the way they test, then there is still much to learn. We teach 5 key principles that explain why agile testing is fundamentally different to traditional testing.

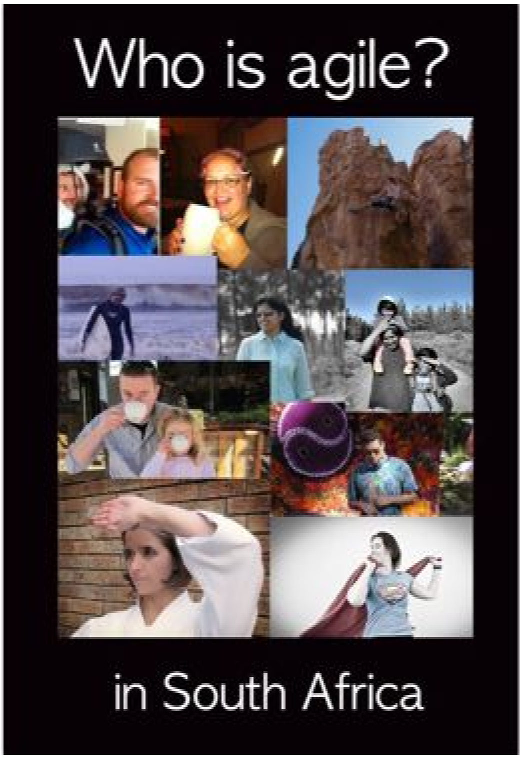
This books includes a collection of workshops to help teams grasp these principles and adopt an agile testing mindset. It's not just for testers. A key part of agile testing is that the whole team is involved, so we always run these workshops with everyone in the team.

If your team is ready for the next level we highly recommend running through the workshops in this book, it will teach them a number of simple but valuable techniques to help prevent bugs and dramatically increase the quality of your products. We provide everything you need to run the workshops, from facilitation plans, to teaching points, and even the slides you might use.

Growing Agile: A Coach's Guide to Agile Testing is available on Leanpub

Other books by Growing Agile





Collaboration Games

Over the years we have played various games with individuals and teams to illustrate the value of collaboration. The games are fun and non-threatening and allow for great learning experiences. Many people have AHA moments when playing games.

You know how you learn how to ride a bike as a kid, and then years later, you are able to hop on and cycle? That's muscle memory. Your muscles remember what they are supposed to do and how they need to work together to keep you on the bike and keep the bike going. We like to think of the games as emotional memory. During the game each person experiences certain emotions that they remember. One day a situation will arise and those emotions will emerge, they will recall the game and their emotions, and be able to apply the lessons they learned during the game.

Collaboration Games is available for free on Leanpub in English, Spanish and Russian

Who is Agile in South Africa

This book is based on the original Who Is Agile book, only this is a regional version for South Africa. It’s a collection of interviews with passionate South African agilists.

Who is Agile in South Africa is available on Leanpub

About Growing Agile

At Growing Agile we help companies create great teams that build exceptional software. We are agile coaches passionate about helping you get the results you are looking for.

Here are examples of how we've helped our clients:

We have helped teams combine **Kanban and Scrum** effectively to manage both production support and new feature development in a sustainable and predictable way.

Our **agile kickstart** has helped software companies adopt Scrum and transform their teams. They are now more focused, enthusiastic, and delivering quality software regularly.

We have coached **ScrumMasters** to better understand their role. Their teams are now more effective, through better facilitation, visibility of impediments, and team ownership.

We have coached **Product Owners** to effectively manage their backlogs to incorporate stakeholder needs, technical debt, and realistic release dates.

If you enjoyed the training plans in this book, you will love our interactive training and coaching. We deliver private courses throughout South Africa for companies looking to train whole teams. We coach teams getting started with Scrum, as well as those who've been doing it for years looking to get to the next level.

Find out more about us at www.growingagile.co.za.