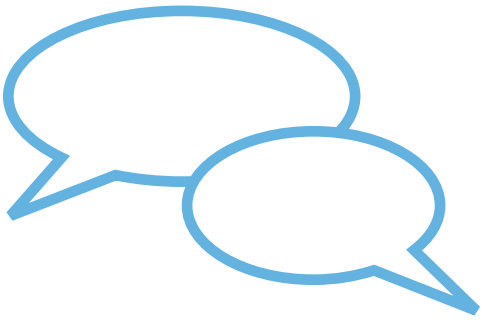
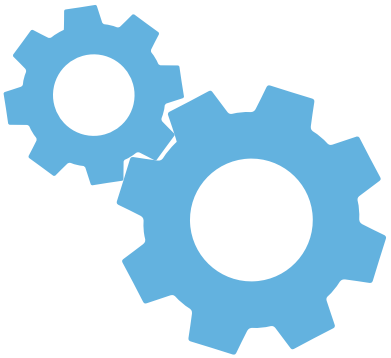


# Behaviour Driven Development

## at the heart of any DevOps transformation story



John Ferguson Smart

https://johnfergusonsmart.com

reachme@johnfergusonsmart.com

+44 7398 832273

# Behaviour Driven Development
## at the heart of any DevOps transformation story

*Speed is of the essence. How fast can you get a new product into the marketplace? How quickly can you fix a defect or tweak a feature? How long does it take you to react to consumer feedback and change? A modern organisation ignores these questions at its peril. But what is the best way to achieve this speed, without compromising on quality? Automation is essential, but increasingly organisations are finding that Behaviour Driven Development holds the key to making automation effective and enabling real quality at speed.*

## Introduction

In today's increasingly competitive world, more and more organisations are feeling the need to deliver software faster. And more and more organisations are looking to Agile and DevOps transformations to give them this capability. Indeed, organisations that have implemented effective DevOps culture and practices benefit from both accelerated delivery and better quality and business outcomes.

But DevOps transformations are challenging, and many organisations struggle to deliver as effectively as they would like. In this document, we look at four key aspects of a successful DevOps transformation, and discover why Behaviour Driven Development (BDD) is an essential practice that underpins much of what is good in DevOps, and that you won't want to do without.

## The Road to DevOps: delivering quality at speed

The goal of a DevOps transformation is to give an organisation the capability to delivery high quality software faster and more reliably, so that the organisation can seize opportunities and react to challenges more nimbly and more effectively.

Automation is what we often associate with DevOps, and automation naturally plays a key role:

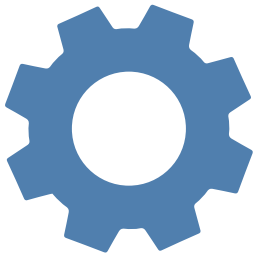
- ✓ We automate our deployment process, so that we can deploy with speed;
- ✓ We automate our tests, so that we can deploy with confidence.

Indeed, the fast pace of delivery many organisations are aiming for today simply isn't possible with long manual deployment and testing processes.

But automating our deployment process isn't all there is to a successful DevOps transformation. And test automation needs to be done with particular care to fit into a DevOps flow. But above all, it is not enough to be able to deploy features quickly; we also need to know that we are deploying the *right* features, and that these features work.

## The four levels of a successful DevOps transformation

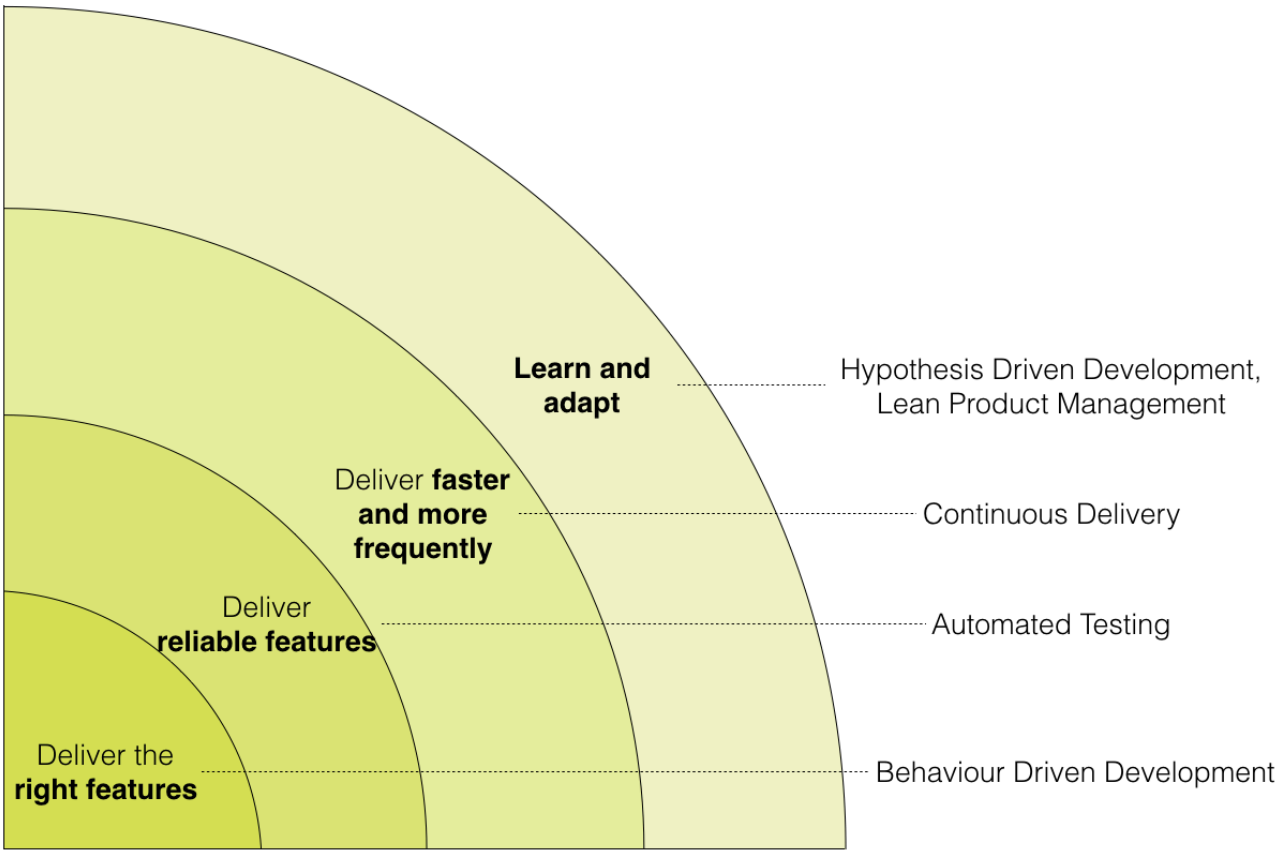
Successful DevOps transformations combine four key aspects, illustrated in the diagram below.



1. **Delivering the right features**. There is little value in delivering faster if the features are not the ones the customer wants. Understanding which features will best serve the customer, as well as the optimal way to deliver them, is at the very heart of DevOps.
2. **Delivering reliable features**: Mature DevOps teams aim for defect-free sprints; that is, sprints where no known defects are present. This can only be achieved with rigorous automation both at the unit testing and the acceptance testing level.
3. **Delivering faster and more frequently**: When we are confident that we are delivering features that the business asked for, and that they work as expected, we can automate the deployment process.
4. **Learn and adapt**: At this level, we can deploy small features quickly and get rapid feedback on their usage and impact in production. This helps us learn whether the features are actually delivering the value we expect of them, and adapt our strategies and hypotheses accordingly.

Each level builds on and leverages the previous one. For example automated acceptance testing is much more efficient and focused with the clear definition and understanding of the business needs that comes from the first stage.

It is also generally easier to introduce process improvement on one level at a time, rather than trying to introduce practices at several different levels simultaneously.



## Delivering the right features

> Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity"
>
> - George S. Patton

Teams will deliver more valuable software sooner if they have a deep shared understanding of the customer needs.

This seemingly obvious statement is not always so easy for teams to put into practice. In many organisations, teams are still presented with solutions to implement. High performing DevOps teams, on the other hand, work with stakeholders to understand the problems that need to be solved, and the value that the organisation will derive from these solutions.


This broader vision allows them to bring all of their creative skills to the table, and deliver features that not only solve the problem at hand but also delight the customer.

This is where collaborative requirements discovery practices really shine. High performing teams often use a Behaviour Driven Development (BDD) process, along with simple and practical techniques such as Impact Mapping, Story Mapping, Feature Mapping and Example Mapping to build a deep shared understanding of the features that will best serve the customer, and determine the best way to deliver them.

## What is Behaviour Driven Development (BDD)?

Behaviour Driven Development, or BDD, is a collaborative process that helps teams understand, implement and deliver the features that really matter.

BDD involves tight collaboration and communication between product owners, business analysts and the development team (including testers) to discover, understand and formulate the real business needs.

BDD also includes a range of practical techniques such as Feature Mapping and Example mapping that help structure and facilitate requirements discovery in order to make them more productive and efficient.

When these business needs are understood by the whole team, they can be automated in the form of "executable specifications", tests that will only pass when the software does everything that is expected of it.

When done well, Behaviour Driven Development and other related techniques are by far the most effective way to build creative, collaborating and engaged teams that will do their best to deliver high quality features that will make a difference for the customer.

We can keep track of our progress when adopting Behaviour Driven Development and other collaborative requirements discovery techniques using the table shown below.

- The real difference starts to happen at the ***Clarification*** level, where the benefits of building a shared understanding really start to kick in.
- At the ***Formalisation*** level, effective test automation, focused by the team's shared understanding, help reduce defects further and accelerate delivery.
- At the ***Validation*** level, the team becomes more proficient at discovering, expressing and automating requirements, freeing them up for more creative thinking, experimentation, and more innovative solutions.

| Level | At this level… | Defects are reduced by | And we see… |
|---|---|---|---|
| **Validation** | Acceptance criteria automated before or during the sprint. | 90% or more | Stories delivered early and exceed client expectations |
| **Formalisation** | Acceptance criteria expressed in an automatable format | 70-90% | Stories delivered early and with zero-defects |
| **Clarification** | Requirements are clarified through conversations about concrete examples and business rules | 50-70% | Stories delivered on time, and defects become uncommon |
| **Conversation** | User stories have acceptance criteria (though they are not always well written) | 20% | Stories delivered but defects and rework are common |
| **Presentation** | User stories are well sized and well written | 10% | Stories delivered late and/or need considerable rework |
| **Dictation** | Traditional requirements broken down into user stories | 0% | Stories often undelivered |

## Delivering reliable features

We have all worked on projects where a large list of open defects is the norm. And each sprint, new defects are discovered and added to the list. Each sprint, teams spend a considerable amount of time fixing bugs inherited from the previous sprints, and this hampers their ability to develop new features.

**44%**

High performing organisations spend 44% more time working on new features

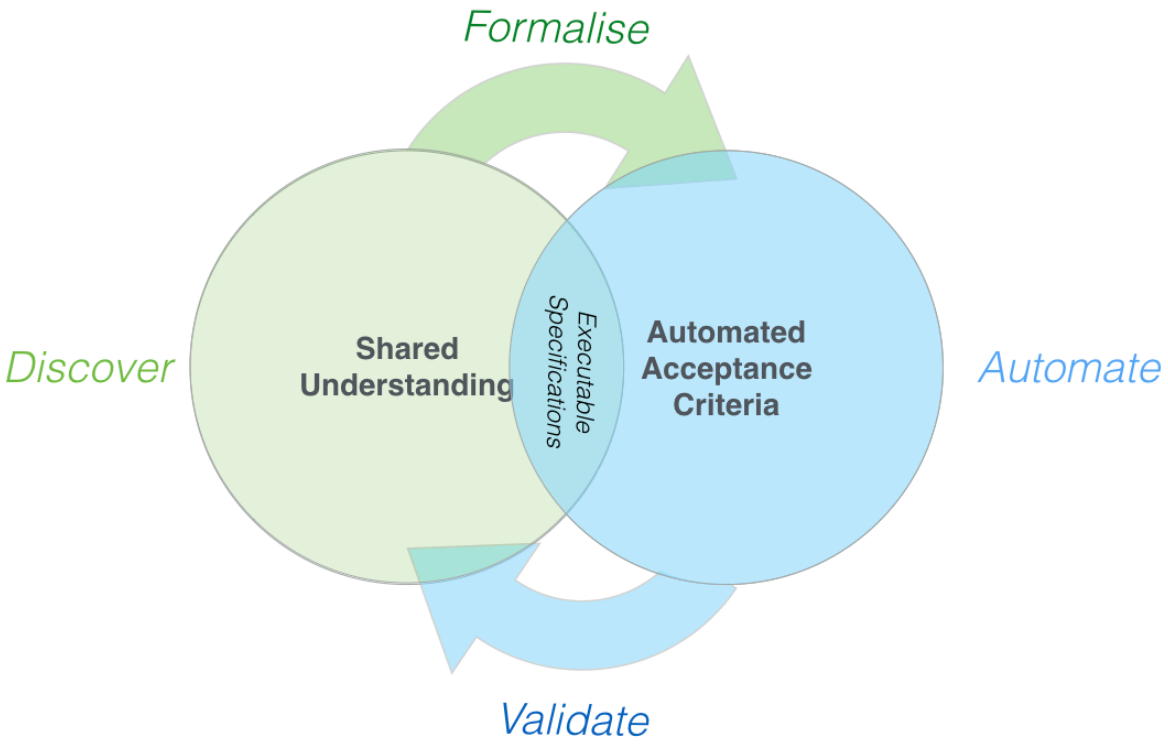
Organisations practicing effective DevOps cannot afford to function like this. High performing teams aim for, and achieve, sprints with zero known defects. And this focus on quality allows them to increase their throughput - according to the *2017 State of DevOps* Report, high performing teams spend 44% more time working on new features.

DevOps can only succeed when an organisation has the confidence that few, if any, defects will ever get into production, and the capability to roll back a release easily and quickly if something does go wrong. And the only way to achieve this level of confidence is though disciplined automation at multiple levels.

## Automated acceptance criteria are essential

In a DevOps world, effective test automation is no longer a “nice to have”. Broken software deployed faster will still be broken, and the rapid-fire pace of DevOps is incompatible with long manual testing cycles.

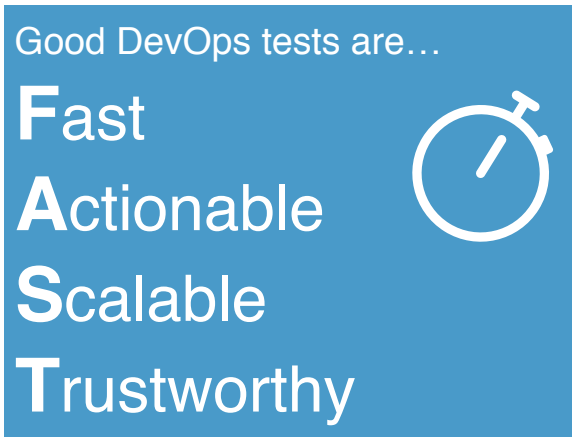
High performing teams following a Behaviour Driven Development process see automated acceptance tests as part of a multi-step process:



- ✓ First they work collaboratively with stakeholders to ***discover*** and understand the customer needs.
- ✓ Then they ***formalise*** and confirm their shared understanding, in a format that can be executed as automated acceptance tests. We call these "executable specifications". They help deepen and reenforce the team's shared understanding, and guides the development process;
- ✓ During development, they **automate** the executable specifications, which become Automated Acceptance Criteria.
- ✓ They can now run the tests to ***validate*** that the application behaves as expected, and confirms to the acceptance criteria that the team agreed upon.

## DevOps needs high quality automated tests

Just as a racing car demands reliable, high quality components, a DevOps team compromises on test automation quality at its peril. Slow-running test suites, flaky tests, or tests that are hard to maintain make it much harder to deliver on the promises of DevOps.




They write automated test suites with the following goals in mind:

- ✓ **Fast**: the longer a test suite takes to run, the longer it takes to deploy code.
- ✓ **Actionable**: When a test fails, it is easy to isolate the issue and take the appropriate course of action.
- ✓ **Scalable**: Tests should not become harder to maintain as the test suite grows, and updating the test suite when the application changes should be as smooth as possible.
- ✓ **Trustworthy**: Above all else, these are tests you can trust. If they fail, you can be confident that something is wrong. But if they pass, you can also be confident that, in the very least, all of the acceptance criteria (including happy paths and key negative scenarios) have been covered.

High performing teams realise that test automation is a software engineering discipline. They invest time to design a test automation infrastructure that will be efficient and easy to maintain. They make sure that their test automation specialists are well versed in software development skills. They design their application to be easy to test, for example with backend APIs that the tests can use to set up test data, or predictable UI elements to make web tests more reliable. They prefer API tests to UI tests where possible whenever they are not directly automating UI interactions, as this gives them major benefits in terms of speed and reliability.

This all involves planning, training, discipling and effort. But their discipline and efforts are paid off by the faster feedback and greater confidence that their test suites give them.

## Delivering faster and more frequently

Delivering faster and more frequently is the bread-and-butter of DevOps. High performing teams take advantage of a number of techniques and practices to accelerate their delivery pace, including:

- ✓ Loosely-coupled architectures and highly autonomous teams
- ✓ Automated deployment and virtualisation
- ✓ Trunk-based development, or very short-lived branches (under a day)
- ✓ Delivering new features in frequent, small batches
- ✓ Using feature toggling to turn features on and off in production
- ✓ Effective management of test data

The key focus of this stage of DevOps adoption is clearly technical, though many of the practices (such as autonomous teams and trunk-based development) also involve big organisational changes to the way the teams work. But the speed gains that come from Continuous Delivery pipelines, virtualised infrastructures and and automated deployments are only sustainable when teams have mastered the previous two levels: well understood business needs and effective test automation.

Many teams leverage some of their fast-running automated acceptance criteria as smoke tests, to check that an automated deployment went well. More and more organisations are taking this concept one step further, and running these tests continually in production.

The deeper shared understanding that comes from BDD requirements discovery also makes it easier for teams to divide larger features into smaller slices, that can be both delivered quickly and still be used to get meaningful feedback from stakeholders.

## Learn and adapt

Fast, reliable, automated deployments are not the ultimate goal of DevOps. Speed is not a goal in itself; speed is only valuable if it allows you to get where you want to go faster.

> “Speed is only of value if it helps you get where you want to go faster”

The real goal of DevOps is for an organisation to take advantage of these technical capabilities to seize opportunities and adapt to market changes faster and better than their competitors.

But to take advantage of these capabilities, organisations need to change the way they think about IT projects, and about how they interact with development teams:

### IT is an investment, not a cost

In high performing DevOps teams, IT is viewed as an investment, not a cost, and the team seeks out ways to measure the value delivered by new features. Since new features can be deployed (and rolled back) quickly, there is no need to try to specify detailed requirements too

far in advance. It is much more important to understand the problem that needs to be solved, and to articulate how each new feature might add value.

## Requirements as hypotheses, not solutions

The logical consequence of this is to express high level requirements not in terms of a specific solution or implementation, but rather as a hypothesis of how a feature might add value. If a traditional requirement describes what solution should be built, a hypothesis is an assumption the potential benefits of a particular feature that can be proved or disproved. So rather than saying:

> ***As an*** *online book store,*
> ***I want*** *to offer free shipping ,*
> ***So that*** *I can sell more books"*

we might say:

> ***We believe*** *that offering free shipping*
> ***Will result in*** *an increase in online sales*
> ***We will know this to be true*** *when we see an increase in sales of 5% or more*

We can now think about the fastest way to test this hypothesis, rather than diving into a (potentially costly) implementation without being sure of the real value.

This approach is a powerful way to avoid wasted effort on features that don't deliver real value, and to narrow in on features that really delight the user and benefit the organisation. However, its success at scale hinges on the well-understood customer needs, well-tuned quality and automation practices, and quick and frequent deployments.

## The six stages of DevOps maturity

So far we have seen a number of techniques and practices that we often see in effective DevOps teams, and we have seen how the shared understanding coming from solid Behaviour Driven Development enables and enhances many if not all of these practices. But DevOps is not just about technical practices. It is also about how the teams are organised, and how (and when) team members interact with each other.

When organisations undergo a DevOps transformation, we find that the teams themselves go through several stages of maturity.

| Name | Focus | Benefits |
|---|---|---|
| **Projecting** | *Experimentation* | Deliver value faster, and learning from it |
| **Pioneering** | *Leverage* | Delight the customer |
| **Mechanised** | *Effective automation* | Deliver the right product faster |
| **Engaged** | *Improving collaboration* | Prevent bugs and deliver the right product |
| **Colocated** | *Breaking down the silos* | Reduce delivery bottlenecks |
| **Siloed** | | |

✓ **Siloed**: At the first level, before any DevOps transformation happens, team members are typically separated according to their roles. Operations folk live in a very different world to the development teams. Product owners and business analysts may not be colocated with the rest of the team. Even the testers work in a different team and operate at a different phase of the project.

✓ **Colocated**: Business analysts, testers and developers start to work alongside each other. At this stage, we are all about reducing bottlenecks and communication gaps caused by a separate business analysis and testing phases, and preparing the ground for closer collaboration at the next level.

✓ **Engaged**: Testers become more involved in requirements discovery, and helping to build a shared understanding of the requirements and work involved earlier rather than later. They start to leverage their cross-examination skills to not only detect bugs, but prevent them from happening in the first place. This phase can see a dramatic drop in defects simply through the better shared understanding of requirements.

✓ **Mechanised**: This stage is where we can start to take advantage of automation at all levels. Testers become more skilled in automation, and understand the importance of automating the right thing. They work with developers to ensure that the tests are well written and the application is testable. Developers also automate the deployment pipeline, making it possible to deploy a new feature in minutes instead of days.

- ✓ **Pioneering**: Automation skills take time to learn and perfect, but once mastered, the time saved can be used to work on new features or experiment and innovate to improve existing ones.
- ✓ **Projecting**: Finally, at the Projecting level, the organisation learns to fully use it's new DevOps capabilities to gain a competitive advantage, seize new opportunities promptly and react nimbly to changing market conditions.

## Where to from here?

We can help your organisation make DevOps a reality and start delivering higher quality software sooner, in particular through the adoption of effective Requirements Discovery, BDD and Test Automation practices.

### Onsite training and mentoring

Much of our work involves helping teams improve their Requirements Discovery, BDD and test automation capabilities, showing them how to engage with the business and collaborate with each other more effectively, and helping them get a maximum return on investment from their DevOps and Agile transformation initiatives. We do this through a tailored combination of training, mentoring, and practical support. Some of our more popular workshops include:

- ✓ **Low Tech, High Impact Sprint Planning**: Learn how high performing teams use sprint planning and pre-planning techniques to build this understanding, and deliver software that delights the customer sooner!
- ✓ **BDD in Action: Mastering Agile Requirements**: Learn how to use BDD within a team to deliver higher value, higher quality features sooner
- ✓ **Advanced BDD Test Automation**: Learn how to deliver world-class test automation to your team by writing more robust, higher quality automated acceptance tests using state of the art test automation practices

### Training remote teams

And for offshore teams, we have a special programme that combines onsite workshops, innovative online courses and exercises, and remote mentoring to build remote teams into high performance centres of BDD and test automation excellence.

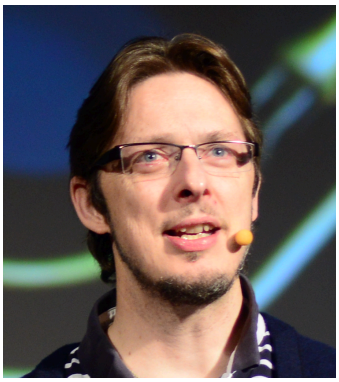
### BDD Success Stories

> *Testers and business analysts now work together to write feature files and define acceptance criteria that can be directly automated, and this has made the automation process a lot faster"*
>
> *- Sayali, Senior Software Engineer*

BDD and test automation, when done well, are highly effective development practices that result in measurable cost savings and qualitative benefits.

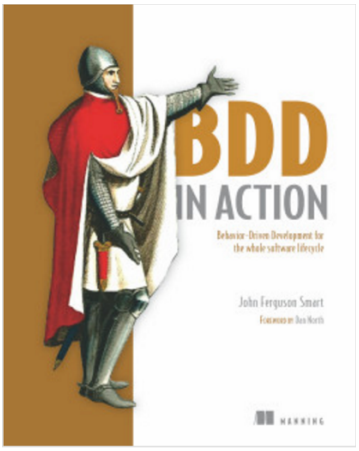
- ✓ In one large bank, we helped introduced BDD and automated testing on a major project over a period of 3 months. In the following releases, only 1 or 2 defects were found in UAT (down from over 30 in the previous release), and zero in production.
- ✓ In a multi-national investment bank, we designed and implemented a test automation framework for a complex regulatory compliance project. Designed for ease of use and sustainability, the framework allowed a team of test automation engineers, some with relatively little experience in test automation, to write over 1400 automated tests. Where these tests to be run manually, they would require some 30 full-time manual testers.
- ✓ In another large bank, we helped a mission-critical project team to adopt effective BDD and test automation practices. As a result, they were able to reduce defects by over 80% and increase overall throughput by 30%.

## John Ferguson Smart



John Ferguson Smart is an internationally recognised thought leader in the world of Agile, BDD and Test Automation. A popular speaker and published author, John has helped countless organisations and teams around the world deliver better software sooner through more effective collaboration and communication techniques, and through better technical practices.

John has been helping international banks and financial organisations with Agile and DevOps transformations for over a decade. Over that time, he has worked with international banks around the world, including HSBC, UBS, Barclays, Deutsche Bank, Credit Suisse, AIB, the Reserve Bank of Australia, Westpac, Macquarie Group Limited, National Australia Bank, MLC, as well as many smaller financial service providers.




John has also trained and mentored many offshore teams in high quality test automation and BDD practices, including teams in India, Poland, Ukraine and the Philippines. The **Serenity Dojo** programme, with its combination of onsite workshops, online training material and remote mentoring sessions is an ideal fit for turning offshore teams into high performing centres of BDD and test automation excellence.

## How John helps

- ✓ **Workshops and programmes**: for organisations and teams undergoing Agile and DevOps transformations.
- ✓ **Facilitator and Coach**: to help teams engage and collaborate more effectively, and to keep them on the right track in their BDD and test automation practices.
- ✓ **Educator and Trainer**: author of the popular Serenity Dojo online training programme, John runs tailored onsite and online training programmes for organisations wanting to keep their development and test automation teams at the top of their abilities.
- ✓ **Author**: A published author of several best-selling books, John also regularly writes articles and blog posts on https://johnfergusonsmart.com/blogs.
- ✓ **Mentor**: John helps organisations scale their DevOps, Agile and BDD capabilities by training and mentoring corporate leaders and internal champions.

**Need some guidance?**

Email: **reachme@johnfergusonsmart.com**
For more info, visit **johnfergusonsmart.com**
Learn online at **http://serenity-dojo.com**
*Follow John on:*
@wakaleo john-ferguson-smart

## Copy this the right way



## We care but you're responsible



## Consider the Environment Before Printing

We'd much prefer readers to download this document and view it digitally. Plus, it looks better on screen!