A Modern Agile Test Automation Playbook

How To Build a Sustainable Test Automation Framework that will Turbo-Charge Your Team

The Serenity Dojo
https://serenity-dojo.com

Serenity DOJO

Do you struggle with test automation?

- Do you have trouble **keeping your test automation up to pace** with new features being built? Do you feel you don't have enough time to automate all the scenarios you need to each sprint?
- Are you **pressured by management** to deliver "100% automation", when you struggle to keep up even with the backlog of regression tests you need to write?
- Do you find your tests can be **brittle and unreliable**? Is it hard to know exactly what went wrong when a test fails?
- Do you find **developers don't want to help out** with automation tasks, and don't respect the value of the automation work you do?

IF SO, YOU ARE NOT ALONE

According to our recent industry-wide survey, these are all common problems among automation testers. We've all been there. It's frustrating.

The truth is, the classic ways that you have learned to do test automation simply are **not effective in modern, agile projects**.

Organisations today need to stay competitive and adaptable in an increasingly volatile world. They need speed and agility to adapt to changing markets and contexts.

J**ust knowing test automation tooling is no longer enough** - you need to know how to use test automation tools and practices, not just to write a bunch of automated tests, but to **make your team deliver faster**.

THE ROLE OF TESTERS IS CHANGING FOREVER, AND IF YOU WANT TO KEEP PACE, YOU NEED TO BE PREPARED

Serenity DOJO

Automation strategies that don't work #1

The Record-Replay Scam

Have you ever been tempted by **record-replay test automation**? Or maybe you have used tools like **UFT** or **Selenium IDE**, or the so-called **"codeless automation"** tools today.

At first all goes well; the tests are quick and easy to record, and need no special skills. But as time goes on, **cracks start to appear;** a single screen is updated, and suddenly your **entire test suite starts to fail**. More and more tests start failing, and you inevitably find yourself with a **pile of fragile test script**s that are a **nightmare to maintain** and **hellish to troubleshoot.**

RECORD-REPLAY WILL SLOW YOU DOWN, NOT SPEED YOU UP

The truth is, while record-replay tools are impressive in a demo, they simply **don't hold water at scale for an agile team**. Despite their slick user interfaces and seductive marketing, record replay tools can actually **slow down** your test automation. Sure, it might be faster to record an individual test case. But record-replay forces you to wait until a feature is finished and stable before you can start testing. Or if a feature changes, you need to re-record the whole script again**. Talk about a time-waster!** And this is time that you just don't have in a truly agile project.

YOU WILL MISS THE IMPORTANT TESTS

Worse-still, record-replay tools result in poor-quality tests. Record-replay tests are superficial. They test a particular path through an application's screens, but they don't go into depth at any stage. They do exactly what you recorded, no more, no less, but they are poor at checking that the application behaves the way it should.

They don't check business logic or edge cases; in other words, they **don't check the things that really matter**.

You don't get coverage; you get a **false sense of security.** And guess what: if your team is trying to move to agile, and you need to spend all this time recording or re-recording test scripts after a new feature is finished, then **you are the bottleneck**!

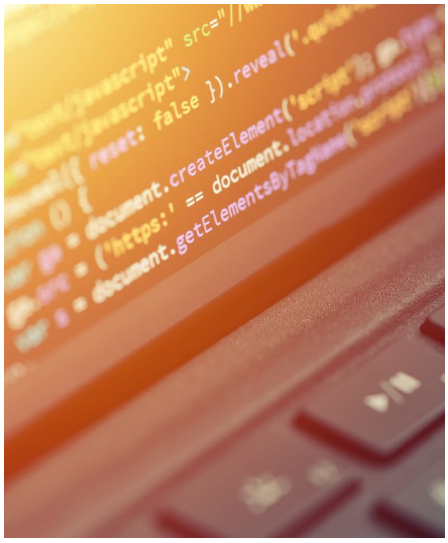
There's a darker side to record-replay tools as well. They **block your career path.** In a job market where test automation excellence is a must-have, nobody cares if you can use a tool that takes a day to learn.

Serenity DOJO

Automation strategies that don't work #2

Traditional Test Scripting

Another common myth is that test automation is simply a matter of writing "test scripts". Test scripts have been around for decades, and the basic approach hasn't changed much.



A test script is just a sequence of detailed instructions on how a user interacts with a page, expressed using vocabulary focused on their Web browser:

- *Open a web page*
- *Click on the "Today's Deals" link*
- *Click on the "Deal of the day" image*
- *Click on the "Add to Cart" button*

For many of us, this is the first sort of automation we ever encountered. It's seductive at first. It's easy to get started, and easy to relate the script to what the browser will do. And a lot of testers never really evolve past this way of thinking. But in reality, **staying at this level is a career killer**.

TEST SCRIPTS DON'T SCALE

As any experienced test automation engineer can tell you, test scripting like this simply doesn't scale.

It's simple to see why. Imagine a test scenario where you buy a book on Amazon. How many clicks would that be? 20? 30? 100? Now imagine another scenario, where you buy two books, or a book and a telephone case. Just as with "codeless" test automation tools, a minor change in a screen can bring your whole test suite grinding to a halt.

TEST SCRIPTS ARE A NIGHTMARE TO MAINTAIN

And while they might be quick to write, they are painfully slow to troubleshoot and maintain. When a screen changes in some way, you need to update every single scenario that uses that screen.

For large test suites, maintaining the existing scripts, just to keep them running, can end up taking much more time than writing new ones. And that's overhead that slows you down and prevents you from writing the new tests that you should be writing.

Don't let the outdated ideas behind the tools you use turn *you* into a bottleneck for your team. Let us help you here.

Book a call with John today!
bettertestautomation.com/apply

Serenity DOJO

Automation strategies that don't work #3

Page Objects

This is possibly the biggest trap that more experienced tester automation engineers fall into. Page Objects are an extremely common approach to UI test automation. But have you ever noticed that oftentimes, even when you use a test automation framework built around Page Objects, you *still* get brittle tests that fail for no obvious reason?

You aren't alone!

Now don't get me wrong - Page Objects are a step up from test scripts; you don't get the massive unmaintainable mess of code that you get with large-scale test scripts. The test code becomes a bit easier to read. You get a bit more reuse.

But Page Objects have some serious issues.

YOU WON'T GET THE REUSE YOU ARE LOOKING FOR

For one thing Page Objects make you **work way too hard**! They make you reproduce a model of your application's UI, where all you really want to do is to model how your users behave and the results they expect to see. The reuse you do get is a trap - it can actually make **you write too much code**, and **slow you down** instead of speeding you up.

SLOW TESTS

Page Objects are UI-centric - they force your tests to interact with the application via the UI. When your framework is build on Page Objects, everything looks like a page. But this can make your tests **slow and cumbersome**, and make you miss opportunities to speed up your tests through non-UI interactions.

BRITTLE TESTS

And worse still, Page Objects actually **make your tests brittle**! By binding your tests to the user interface, you make your tests more likely to break whenever the implementation changes.

Frustrated with slow, brittle and fragile test suites?
Need to do better? We can help you here

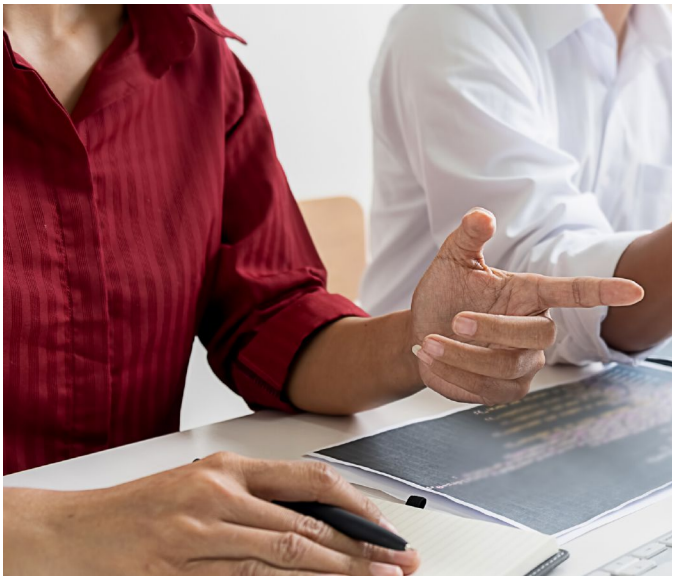
Book a call with John today!
bettertestautomation.com/apply

Serenity DOJO

Automation strategies that don't work #4

"BDD" Test Scripts

Many testers fall into what I call the "BDD trap". They hear about the great results some teams are getting with Behaviour Driven Development (BDD), so they want to try it out for themselves.



Don't get me wrong, this is a great idea. Behaviour Driven Development is one of the most powerful techniques. I've worked with dozens of teams and hundreds of testers to implement BDD correctly, and the results are spectacular. Teams practicing BDD well regularly see **80-90% fewer defects.**

WHEN BDD GOES WRONG

But 90% of testers who try to implement BDD by themselves **get it wrong**.

```
Scenario: Place an order test 1
  Given I opens the home page
  When I clicks on the "Today's Deals" link
  And I waits for deals of the day to appear
  And I clicks on the "Deal of the day" image
  Then verify that the deal details appear
  When I clicks on the "Add to Cart" button
```

It's normal, **BDD is hard**. The scripts they write are too low level, too granular, too tied to the user interface, to get any of the benefits they need.

Using Selenium with Cucumber this way is no better than the test scripts we saw earlier, and test scripts written this way quickly become **brittle and unmaintainable**.

UNMAINTAINABLE TESTS

I've seen so many teams who have had to abandon their whole Cucumber 'automation test suite, because it had become impossible to maintain. **Months of test automation work wasted!** How do you think that makes the testers look?

Have you had bad experiences with fragile BDD tests in the past? We can show you how it's meant to be done here

Book a call with John today!
bettertestautomation.com/apply

Serenity DOJO

A proven formula to building a scalable agile test automation framework

To survive and thrive as an agile test automation engineer in today's environment, you need to know how to build a **sustainable and scaleable test automation framework** tailored to your business domain.

- A framework that **runs reliably and consistently**, every build, and that doesn't fail randomly when you need it the most
- A framework that **reports what the business wants to know**, not just which test cases were executed
- A framework where many new tests can be **added without any additional code**

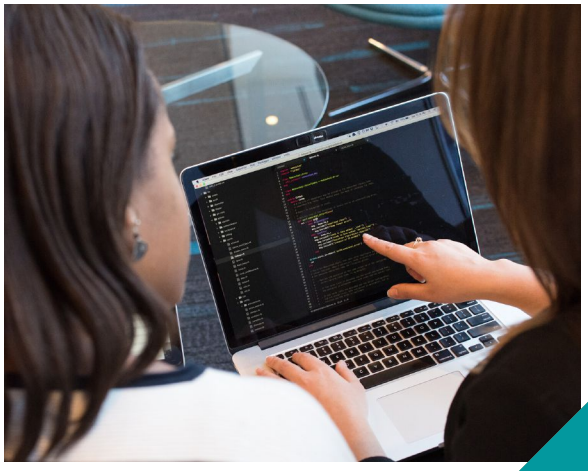
No one can sell you a framework like this for your project, because **no such framework exists.**

But we can show you how to build one.

A few simple steps can help ensure that your test automation efforts don't go to waste.

STEP 1 - THE *EXECUTABLE SPECIFICATION* MINDSET

The best agile test automation engineers don't write test scripts; they **turn business requirements into executable specifications**. And this changes everything. Executable specifications are the key to getting more efficient automation, higher quality results and higher quality reporting.



But there is an art to this, and with Serenity Dojo, we teach our clients how to write effective executable specifications that really bring the whole team together. We can help you too, **here**.

SHOULD I WRITE TEST SCRIPTS OR EXECUTABLE SPECIFICATIONS?

TEST SCRIPTS	EXECUTABLE SPECIFICATIONS
Are seen as the tester's problem	Encourage collaboration across the team
Are only written after a feature is finished	Can start before the feature even starts, and you can even get developers involved!
Report only what test cases you ran	Report on what the application is supposed to do

Book a call with John today!
bettertestautomation.com/apply

Serenity DOJO

A proven formula to building a scalable agile test automation framework

STEP 2 - THE PAVLOVA PRINCIPLE

One of the secrets to a solid, scalable test suite is layering and structure. Just like in a Pavlova cake, each layer has a purpose, and each layer needs to contain precisely the right ingredients. Without the proper layering, structure and ingredients, you just get a mess.

Organising and structuring your test code is an essential skill if you want your test automation framework to scale.

But the trick is to organise your automation code both in technical layers and also in terms of your business domain. Rather than focusing on only one level of reuse (which is what generally happens with Page Objects), the most effective, most scalable automation frameworks weave reusability across both business tasks and of technical interactions.

STEP 3 - AUTOMATION POLLINATION

Too often, testers end up **slaves to test automation**. Have you felt the pressure to automate more and more, while you struggle to automate tests for the current sprint, not to mention the growing backlog of regression tests that need automating?

But, with a well-designed executable specifications framework, you can do much better.

- You can encourage analysts and business folk to participate in test automation, so that misinterpreted requirements become a thing of the past
- You can encourage developers to participate as well, because it makes them go faster too!

The final stage, then, is to **flip the tables** and put test automation at the **service of the whole team**. This way you can leverage your automated tests to TRULY benefit your whole team, accelerate your delivery, and impress your manager.

Book a call with John today!
bettertestautomation.com/apply

Serenity DOJO

A proven formula to building a scalable agile test automation framework

WHAT'S NEXT?

It's our hope that this playbook will help you get a better idea of what really works when it comes to building a scalable, sustainable agile test automation framework.

With Serenity Dojo we have helped hundreds of testers become high performing test automation engineers, teaching the system outlined above using Java, Cucumber, Serenity BDD and other relevant tools.

If you'd like to speak with us on how we can help you incorporate this system into your projects, current or future, we'd love to talk to you. There is never any pressure on our clarity call sessions. Our objective is to figure out where you're having the most trouble in your current test automation practices and skill levels, and then help you figure out what the best plan of action is.

If you'd like to set up a clarity call session with us, follow the URL below to do that:

https://bettertestautomation.com/apply

Feel free to send us an email (if you need anything) or if you just want to say 'thanks for the guide'. (**support@serenity-dojo.com**).

Looking forward to talking soon!

John Ferguson Smart
Founder
Serenity Dojo