SUCCESS

10 PROVEN ACTIONABLE STEPS FOR AUTOMATION AWESOMENESS


Alan Page


Alan Richardson


Dave Haeffner


Rosie Sherry


Unmesh Gundecha


Jim Evans


Dima Kovelenko


Greg Paskal


John Sonmez


Cory House

From the Top TestTalks Guests

Ten Proven, Actionable Steps for Creating Automation Awesomeness

There are many ways in which you can make your testing and automation efforts better. So the last question I ask every guest on my TestTalks podcast is: "What's the one piece of actionable advice you would give someone to improve their testing/automation efforts?"

Here are ten testing and automation actionable steps you can take right now to help you succeed with automation awesomeness, from ten of the top testing and automation leaders featured on TestTalks podcasts.

Bonus

Anyone who knows me knows that I'm a bibliomaniac. I can't resist ending an interview without first asking each TestTalks guest about which of their favorite books they would recommend to help someone learn more about testing or automation. Below each featured guest you will find links to his or her recommended resources.

1. **Alan Richardson, the Evil Tester. Author of *Selenium Simplified, Java for Testers* and creator of the online training course Selenium 2 WebDriver Basics with Java (TestTalks #4)**

Alan advocates multiple, overlapping abstraction layers where you model the physical world and your logical concepts and focus on synchronization. If you were to do just one thing tomorrow, it should be to look through your abstraction layers and make sure that you have the right level of synchronization.

Wherever possible, ramp down any implicit wait times you've gotten until you are relying completely on explicit waits.

Alan's Recommended Resources:

- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Growing Object-Oriented Software, Guided by Tests
- Selenium Simplified Java for Testers

2. Dave Haeffner, author of *The Selenium Guide Book* and creator of ChemistryKit -- an open-source Selenium Framework (TestTalks #2)

Dave recommends two things. First, the best thing you can do tactically -- if you're not using page objects -- is to figure out what those are and start using them. It will make your tests immensely better, more readable and much more maintainable.

Second, in terms of something a little softer but still actionable, try to understand --if you don't already -- how your business makes money or generates value.

Next, identify critical things that users use in the system, then match that business value and see if you have automated tests that cover it. If you don't, add them. You'll be thanking yourself later, after you've avoided some catastrophic emergency release.

Dave's Recommended Resources:

- The Selenium Guide Book
- Selenium 2 Testing Tools: Beginner's Guide – David Burns
- Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing – Gojko Adzic

3. **John Sonmez, Pluralsight online trainer extraordinaire, with over 50 courses, including Creating an Automated Testing Framework with Selenium and Automated Testing with Selenium (TestTalks #3)**

John believes that the most critical element to improving your automation framework is to make sure that you have a level of abstraction between your actual browser and your tests. You should try to make your tests read like the business requirements of your application.

For example, your automation test should read like sentences with readable method names like "addUserToSystem" and "addItemToCart."

They shouldn't read like vague computer code. Some bad examples might read, for instance, like "look for button with id of login button," "look for label with id of first name," "fill in john…" You want to try to make your test move the first way rather than the latter way.

Basically, you want your abstraction layers to read like English. Anyone should be able to look at your test or code and totally understand what those tests are doing.

Try to make your tests simpler. If you start off with the goal of wanting them to communicate the business or the actual problem domain of the system and you figure out how you are going to make that happen, you aren't going to need to know how I or someone else did it; you're going to be able to work backwards from there.

The most important thing is to begin with the end in mind. If you do that, you're going to simplify your tests, making them as easy and straightforward as possible. Everything else will flow from there.

John's Recommended Resource:

- Soft Skills: The software developer's life manual

4. **Greg Paskal, author of multiple white papers on test automation and testing in general, and speaker at most of the top testing conferences (TestTalks #8)**

Greg recommends one of the most overlooked areas for automation success – getting training. Make sure that you're getting the proper training before stepping out with a test tool. Get the training you need to use the tool as it was designed to be used. Go back to your management, even if you find a webinar or something similar online that you can take, to help point you in the right direction. Be sure that the training is something you can tie back to the vendor or some other trusted source so you can be confident that it's not bad information being taught to you. It's a good place to start, and definitely something I would highly recommend.

Greg's Recommend Resources:

- How to Break Software – James A Whittaker
- Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design - James A Whittaker
- How Google Tests - James A Whittaker

5. **Cory House, founder of Bitnative and author of multiple Pluralsight video training courses, including Clean Code (TestTalks #13)**

   Cory's only piece of advice is to write your automation code with the reader in mind. Don't write your code for a computer; write your automation code for your co-workers.

Cory's Recommended Resource:

- Clean Code: A Handbook of Agile Software Craftsmanship

6. Unmesh Gundecha, author of the *Selenium Testing Tools Cookbook* (TestTalks #17)

Automation is not a silver bullet. Many teams start doing automation and then get dragged away and put too many bells and whistles into their frameworks. They spend lots of time making their framework do super-duper things, but lose their focus on the intention of the automation, which is to actually help testing.

Automation is not testing. Automation *supports* testing. You should automate things that are deterministic and automatable so your testers can focus on more interesting stuff. They should also consider the outcomes of automation. How is it helping to deliver a quality product? How is it helping you to ensure the product is built with the highest quality?

Unmesh's Recommended Resources:

- Selenium Testing Tools Cookbook
- Instant Selenium Testing Tools Starter

7. **Rosie Sherry, founder Software Testing Club and Ministry of Testing and co-founder of the testing agency Testing Ninjas. She is also the organizer of TestBash, a popular software testing conference (TestTalks #20)**

Rosie's best piece of advice is to learn about the environment you're working in and what's important for you to know about that environment. For Rosie, this was learning about startups as well as business and web design, because that's where her interests lie.

Whichever industry you work in, you should be interested enough in that industry to try and figure out how you can make that industry better. It shouldn't only be about how you can make *testing* better, but what can you do to improve the industry and the product you're working on.

It's not always about testing. Rosie thinks that's part of the problem with the testing world. Sometimes we focus so much on testing that we forget about the business side of things or the users themselves. We sometimes lose sight of what's important. If you keep the whole picture in mind you'll go a lot farther then just focusing on the bugs you're trying to find.

Rosie's Recommended Resource:

- Lessons Learned in Software Testing: A Context-Driven Approach

8. Jim Evans, one of the key contributors to the Selenium project. He is also the man behind the Selenium .NET bindings and the Internet Explorer driver (TestTalks #28)

Jim gave his actionable advice in two parts. One is to watch your configurations of Internet Explorer.

The second: if you have a language -- like Java, for instance -- that allows you to step through the execution of your tests, use your IDE. That will help you step through the execution of your tests using breakpoints.

Breakpoints will help to stop your test at the point where you think it is going to fail or right at the point it's going to fail and see and make sure the application under test is in the state you think it's in.

This will allow you to see what the variable values look like, and can also give you ideas about things that you may want to log. You also might discover an error path that you may want to exercise to get more test ideas.

Jim's Recommended Resources:

- Dave Haeffner : The Selenium Guidebook
- Selenium Design Patterns and Best Practices – Dima Kovalenko

9. Dima Kovalenko, author of the *Selenium Design Patterns and Best Practices* book (TestTalks #30)

Dima believes that you should slow down and never stop learning what good development practices are. As you improve your development skills, you will also improve your test writing skills because the two are related. Treating the test as *real* code and not as a set of instructions that are quickly slapped together is by far the best way to improve your testability and the overall healthiness of your test suite.

There are several good design books out there, and you really should learn about design patterns and how to best approach solving problems as a programmer. It seems to Dima that as software testers we tend to sell ourselves short in that area. There's no reason we can't be as good at writing code as developers are. There's no reason for us to take a back seat and believe it's okay to write poorly written code that tests a website – only to be left feeling confused and sad when the poorly written test codes are failing and becoming unmanageable. Instead, we should keep trying to improve ourselves in general. Never stop striving to be better at what you do. If you keep that goal in mind, you will find your tests and programming skills getting better, and you will invariably begin to find writing tests much more enjoyable.

Nothing is more soul crushing than having a test suite that is all over the place and just failing randomly. But when you've built a test suite that is logical and well-written, and that any developer can look at and be impressed by, it actually makes you feel good about yourself and the job you're doing.

I believe that, more than anything, striving for perfection is what will ultimately make you a more content and well-rounded software developer.

10. Alan Page, the lead author of *How We Test Software at Microsoft*. He also contributed chapters to *Beautiful Testing and Experiences of Test Automation: Case Studies of Software Test Automation* (TestTalks #44)

Alan thinks that best way to learn about testing and/or automation is by being involved -- even if it's just by reading online resources (think Ministry of Testing, Testing Planet or Software Testing Club), and following some testers on Twitter.

You also might want to start following some testing blogs. Start cramming those ideas into your head so you can figure out which ones works and which ones don't. Start interacting with the blog authors -- maybe add a comment or question or two on their posts. Ask some questions on Twitter.

If you start getting out of your office – at least in the virtual sense – and start making use of the many online resources out there, it will help you to learn and will foster some new ideas so that you can begin to throw away the bad ones (or the ones that don't apply), and figure out which ones will work best for you.

Alan's Recommended Resources:

- How We Test Software at Microsoft
- Beautiful Testing: Leading Professionals Reveal How They Improve Software
- Experiences of Test Automation: Case Studies of Software Test Automation
- The "A" Word – Under the Covers of Test Automation By Alan Page