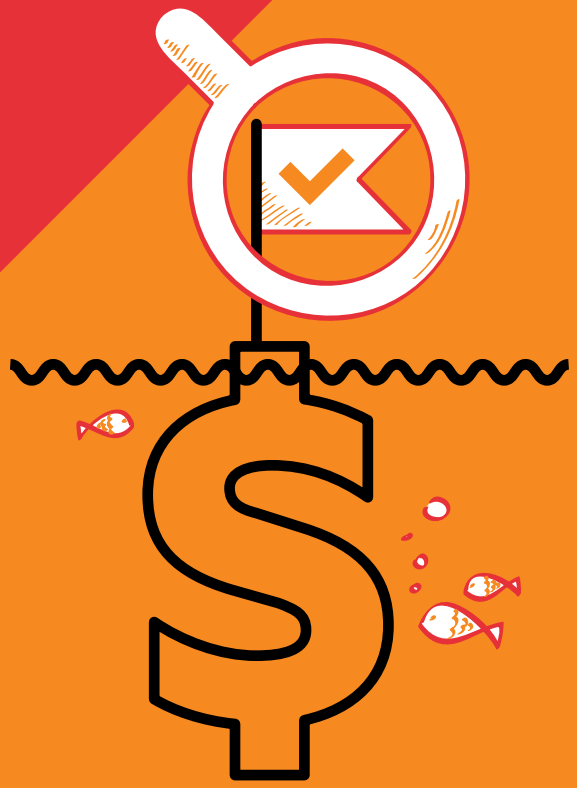
5 HIDDEN COSTS OF USING SELENIUM



Telerik Test Studio

CONTENTS

INTRODUCTION

Teams starting functional user interface (UI) test automation often prioritize free tools—such as Selenium IDE or Selenium WebDriver—ahead of paid counterparts, presuming that open source software means a free project. While both Selenium IDE (SIDE) and WebDriver are indeed free to use, ***it's a mistake to assume*** a free tool yields ***no additional*** adoption ***costs.***

This whitepaper will provide insight into some of the hidden costs testers encounter when working with Selenium IDE or WebDriver. It will also offer comparisons with Telerik Test Studio to help you understand why ***upfront costs should not be your sole evaluation criteria.***

What's Selenium IDE?

Selenium IDE (SIDE) is a tool for recording and playing back web automation tests. It's a plugin for Firefox that enables users to capture actions in the browser, perform various actions and make assertions on the state of the web page.

SIDE saves tests in HTML files, and can export tests to a number of different languages, including Ruby, C#, Python and others. Selenium IDE only works in Firefox, so it's a single-browser tool.

SeleniumHQ, the home organization for all things Selenium, specifically recommends SIDE as a tool for documenting bugs and handling one-off scripts as part of other manual test efforts. Unfortunately, its strengths and weaknesses are often misunderstood, leading to ***misuse in an effort to create large automation suites.***

What's Webdriver?

Selenium WebDriver (often just "WebDriver") is the latest incarnation of the open source Selenium web automation driver. WebDriver is explicitly a ***driver, not a framework.*** Automation drivers interface with browsers, while frameworks provide some form of grammar-based interface over the code used for the actual browser manipulation.

WebDriver is a 100% coded automation solution. It is available for many popular languages including Java, Ruby, C# and VB.NET, Python and others. WebDriver also supports playback on several different browser types.

What's Telerik Test Studio?

Telerik Test Studio is a set of tools to help teams quickly create stable, maintainable, high-value automated test suites. Test Studio provides recording and playback of scripts on Chrome, Safari, Firefox and Internet Explorer. Additionally, Test Studio allows testers to extend or customize tests with C# or VB.NET as desired.

Test Studio enables you to create automated tests for all web applications, Silverlight and desktop applications written in WPF. Additionally, mobile testing for iOS and Android is available through the Telerik Platform, a separate set of Telerik services.

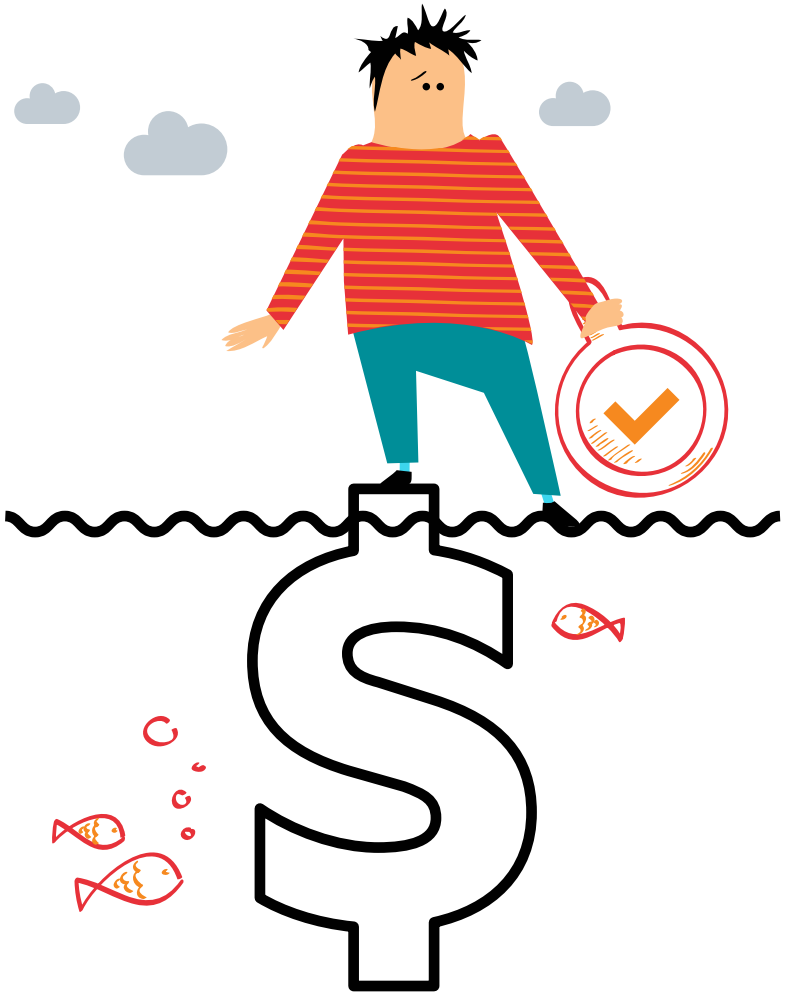
COMPARISON CHART

		Selenium IDE Free/Open Source	Selenium WebDriver Free/Open Source	Telerik Test Studio Starts at $119/month
Setup and Configuration				
	Build/Scheduling Server	3rd party integration	3rd party integration	YES Out of the box
	Execution agents	3rd party integration	3rd party integration	YES Out of the box
	Scheduling	3rd party integration	3rd party integration	YES Out of the box
	Reporting	3rd party integration	3rd party integration	YES Out of the box
Creating and Running Tests				
	Multi-browser support	Record and playback in Firefox only	No recorder. Playing back on multiple browsers requires careful code constructs.	YES Out of the box
	Customize and extend tests with code	No	No recording. Customization/ extensibility by default since WebDriver is 100% code.	Start with a recording, customize and extend in C# or VB.NET (or write 100% code if you want!)
	Parallel execution	3rd party integration	3rd party integration	YES Out of the box

COMPARISON CHART (CONTINUED)

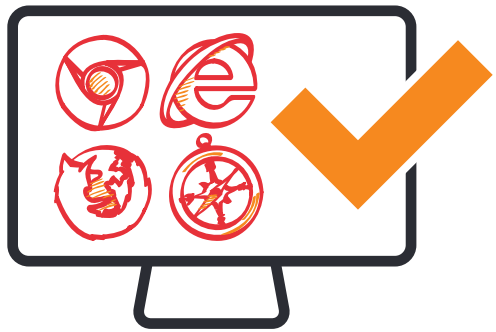
		Selenium IDE Free/Open Source	Selenium WebDriver Free/Open Source	Telerik Test Studio Starts at $119/month
Maintainability				
	Centralized storage of locators	No. Element locators duplicated in every test across the entire test suite.	Yes. Must write code that follows Page Object Pattern or similar approach.	Yes. Test Studio uses Page Object Pattern out of the box.
	Reuse of tests/modules	No	Yes. Must design well-crafted tests and software.	Out of the box. Must design well-crafted tests and software.
Training and Support				
	Adoption Training	Through 3rd party providers	Through 3rd party providers	Multiple options directly from Telerik
	Support	Through community or 3rd party providers	Through community or 3rd party providers	Multiple options directly from Telerik

HIDDEN COSTS



Open source software tools like SIDE or WebDriver are free; however, there are still costs associated with adopting “free” software. Make sure you understand what those hidden and not-so-hidden costs are, so you won’t be surprised later.

MULTI-BROWSER SUPPORT



SIDE and WebDriver

Selenium IDE only works in Firefox. If you're using SIDE you're strictly limited to that browser both for recording and playback.

WebDriver is a 100% coded automation tool, so there's no recorder. Playback is a different matter, though. WebDriver has various browser implementations, including Internet Explorer, Firefox, Safari, Chrome and headless (no browser) through HtmlUnitDriver. There are also drivers for iOS and Android which enable both native application and mobile browser testing on those platforms.

One must follow careful software design practices when creating a WebDriver test suite supporting multiple browsers. ***Because WebDriver tests are implemented purely in code, you'll need to handle proper levels of abstraction and interface implementation in your test suites.*** Generally, this is accomplished through some form of a factory helper with configuration options passed in through your build/execution environment. This isn't as hard as handling multi-threading concurrent database transactions in code, but it's not trivial. ***Your automation team may not have the appropriate skills to understand how to build this out in a flexible, maintainable fashion.***

Telerik Test Studio

While Selenium IDE supports only Firefox, and WebDriver requires additional code to support multiple browsers, ***testing across Firefox, Safari, Chrome and IE in Test Studio is a snap. You can quickly record tests in any browser and playback tests across multiple browsers***—either individually or in test lists—by using easily accessible options. No need to dive into code to manage your browsers—although you certainly can if you want to.

Why do I need it?

GET BETTER TEST COVERAGE WITH SUPPORT FOR MULTIPLE BROWSERS

Gone are the days when web sites could, with a straight typeface, say "Best viewed in <fill in browser type here>." If Internet Explorer had the lion share of internet users back in 2004 (91.35% vs. 3.66% for Firefox, and 2.09% for Netscape), users these days feel free to bounce between Chrome (43.92%), Safari (9.14%), Firefox (18.95%), Internet Explorer (23.24%) and various mobile browsers (23.41%) [Source: NetMarketShare].

Moreover, your users will likely be spread across many versions of each of these browsers. Consequently, your test tool needs to help cover your browser test matrix as efficiently as possible.

MAINTAINABILITY, THE LARGEST COST OF AUTOMATION

SIDE and WebDriver

Because ***Selenium IDE stores element locators in each step of the test***, you must interact with the one element every time the locator is defined in a different place—if it's 50, that's 50 locations. This produces a serious maintainability issue because of the duplication involved.

WebDriver's locator storage is completely up to the developer writing the tests. As with system software, you can get well-designed software that's a joy to deal with, or a horrible mess of spaghetti code that's impossible to unravel when it comes time to make changes.

Because WebDriver is 100% code, one should follow good development practices and implement the Page Object Pattern. The hard fact is that far too many workers in the UI automation space either don't know about the Page Object Pattern, or are in environments where they're not empowered to follow such an advantageous practice.

Why do I need it?

MAINTAINABILITY IS ESSENTIAL TO DELIVERING ON TIME AND ON BUDGET

It's critical to approach your automation efforts with maintainability in mind. Creating 20 small tests is one thing, maintaining those 20 tests over time as your system changes is a horse of a different color. This is especially true as you grow your test suites through hundreds or thousands of tests.

Automation tools should help you focus your valuable time and budget on creating valuable, scalable and maintainable test suites that don't drain your team's time and energy with constant maintenance. Understanding how your tools can keep your team focused on their best value contributions is a critical factor to consider when selecting a testing tool.

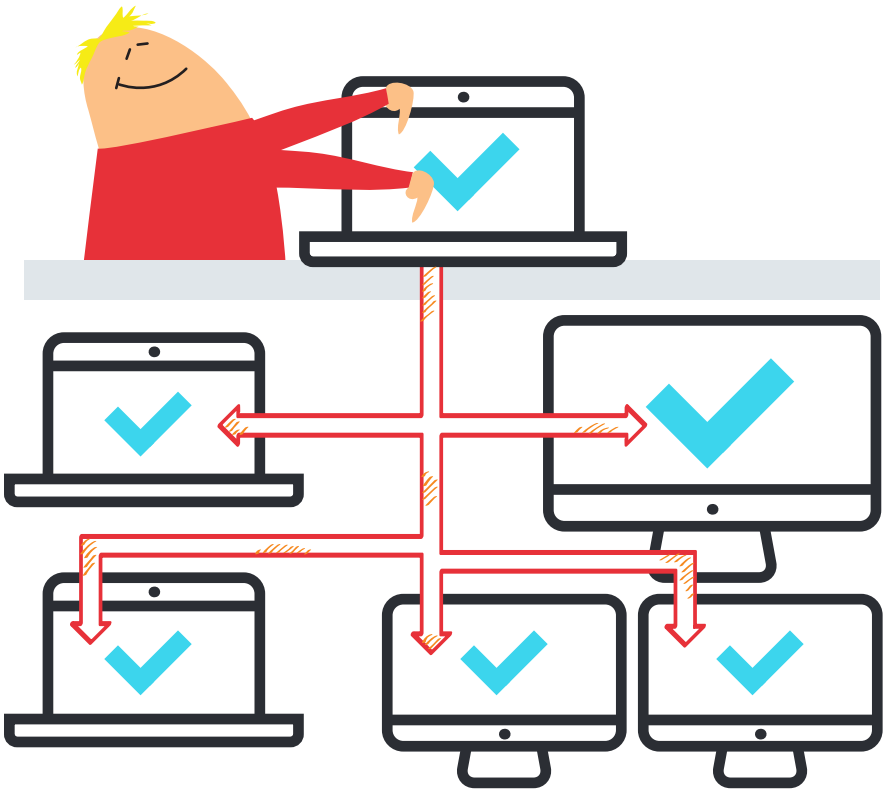
WHAT IS A CENTRALIZED ELEMENT STORAGE

Element locator definitions are the most critical aspect of a successful automation effort. You have to understand both how locators work in the system you're building, and how the testing tools store those locators.

Careful storage of your element locator definitions is crucial to the maintainability of your test suites. Your system under test will change its UI. You're in for a world of pain if your locator definitions are duplicated and scattered across each test as 60 or 300 duplicated definitions will need updating when two or three elements on a page change their location.

Several alternatives for good locator storage have evolved recently; however, the Page Object Pattern has been the clear winner for the last several years.

Abstracting locator definitions out to a single page object is a tremendous approach with huge benefits for test maintenance. When, not if, your application changes, you simply go to one place to fix your locators. Every test impacted by those UI changes is automatically updated since they don't define those locators, but instead, refer to them via the page object.



Test Studio

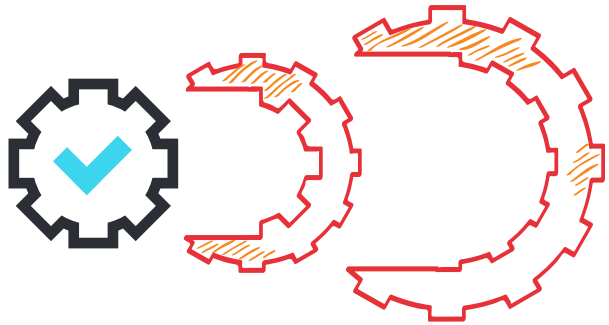
We've seen that SIDE duplicates your element definitions every time you reference an element, and we've also seen that keeping good definitions in WebDriver means you'll need to carefully follow the Page Object Pattern or another design approach. On the other hand, the Test Studio element repository handles centralized storage of element locators for you right out-of-the-box. The element repository is a variant of the Page Object Pattern previously mentioned. Locators are stored by page in the repository with no duplication at all. ***The repository is the single point for updating locators as your system evolves.***

Test Studio recorders create the repository as you build your tests. You can easily update locator definitions if you need to adjust find logic.

When your system's UI changes, head to the element repository and update the impacted elements. You'll see those updates reflected in every test that references that element—the beauty of centralized storage in action.

Another terrific benefit of the element repository is that it's available for tests not only through the UI, but also directly in code. Extend tests while still making use of the centralized definitions in the element repository. Changes to elements in the repository updates regular steps, as well as actions and references in code.

MAINTAINABILITY (CONTINUED): REUSE OF COMPONENTS



SIDE and WebDriver

Selenium IDE does not support reuse of tests as blocks. You can string things together in a series of small test cases, but you're not able to do complex flows based on smaller modules.

Component reuse in WebDriver is ***completely dependent on following good design and construction principles*** as you evolve your suite. Because you're writing 100% code, it will be up to you to compose a complex test scenario by pulling in modules, classes, methods and other constructs from other areas of your test suite.

Telerik Test Studio

Unlike Selenium IDE, ***Test Studio supports reuse of tests across your test suite right out-of-the-box.*** You're able to create small tests which can be data driven (parameterized) or otherwise flexibly configured. Pull those into larger tests to avoid duplicating functionality in multiple places.

With Test Studio, ***not only can you re-use recorded steps multiple times, you can also reuse any coded extensions or customizations you've created.***

Why do I need it?

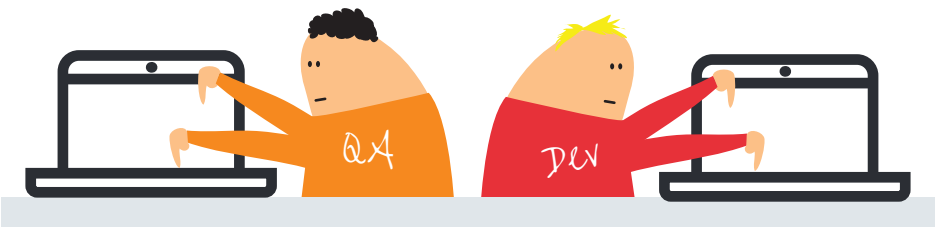
REUSE OF COMPONENTS

Now it's time to turn to the structure of the tests themselves. Good software is based on eliminating duplication (see the DRY principle), eliminating complexity and carefully designing blocks of tests so they focus on doing one thing really well (see SRP).

All these design tenets ease your creation and maintenance costs, but also lead to something wonderful: reuse of components of your test suite. By building small, carefully crafted tests, you will be able to compose larger tests by reusing the small blocks already built. Avoid having to repeatedly duplicate functionality.

Think of common tasks like logging onto a system or creating a blog post. Each of these can be used as a smaller part of a larger scenario. Reuse drastically cuts maintenance costs and improves the speed at which you create larger test suites. Having an automation test toolset that enables you to do this is critical to your long-term success—and sanity.

CUSTOMIZE IN CODE



SIDE and WebDriver

Those working with Selenium IDE, will not be able to leverage any code to help with the tasks listed above. SIDE does support invoking JavaScript, but it's in the context of the page's DOM and the current browser. Getting to your system for configuration, setup or test oracles will be a major challenge at best, impossible at worst.

WebDriver doesn't give you the productivity benefits of a recorder, so you're already in 100% code. Consequently, you'll be able to create and use backing APIs, infrastructure, helper methods and all the other things that make great software so advantageous. Of course, you'll need to follow solid software craftsmanship principles to ensure you're creating all that code in a sustainable fashion.

Why do I need it?

EXTENDING TESTS WITH CODE FOR BETTER TEST COVERAGE

Test recorders provide a great productivity boost when creating your tests; however, every significant automation project will require some level of code-level effort. Stable, reliable, maintainable automation suites rely on code for a number of critical tasks:

- Set up and teardown tasks (creating new users, test data, etc.)
- System configuration (shut off unneeded features like CAPTCHA, rich text editors, etc.)
- Environmental tweaks (loading baseline data, flushing or loading cache, etc.)

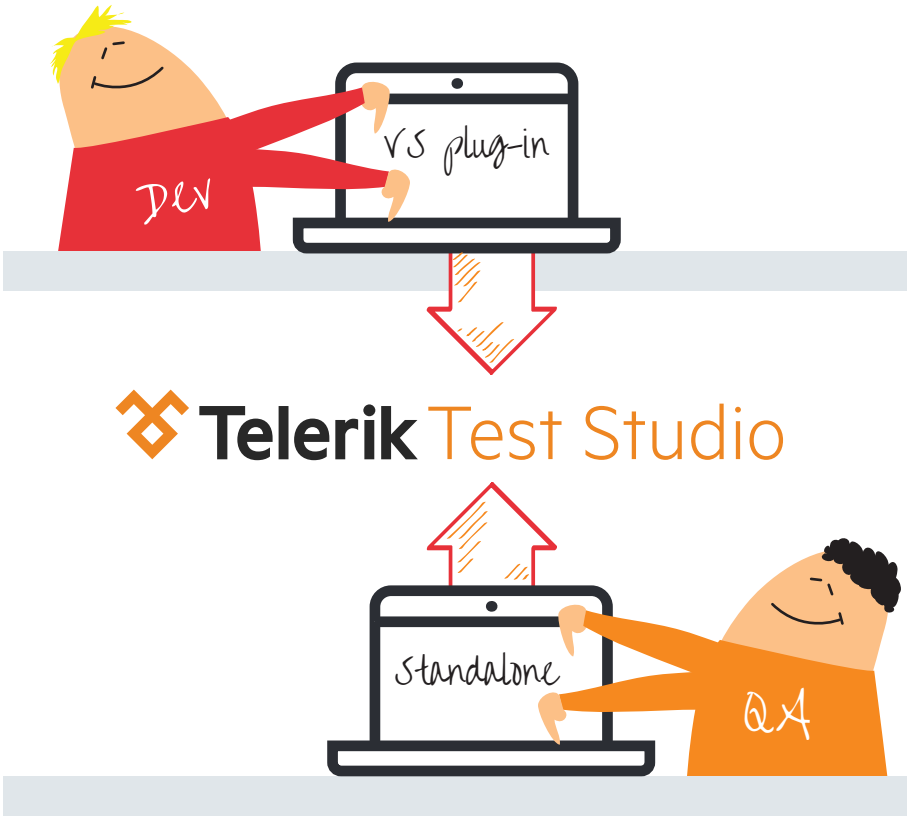
Code isn't just for major actions like these. You'll also need to leverage code for situations unique to your own application. Do you have situations with multiple concurrent AJAX operations? One way to handle synchronizing or latching for those situations is a coded step dealing with your application's particular needs.

Moreover, a good automated test will use some form of heuristic or test oracle to check a test's true state. Simply checking that your UI reflects that you've created or updated an item is not a good enough test. You need to look at your system's persistence layer to ensure the database (or whatever) was correctly updated. The only way to accomplish these oracles is to drop to code and talk directly to web services, stored procedures or internal APIs.

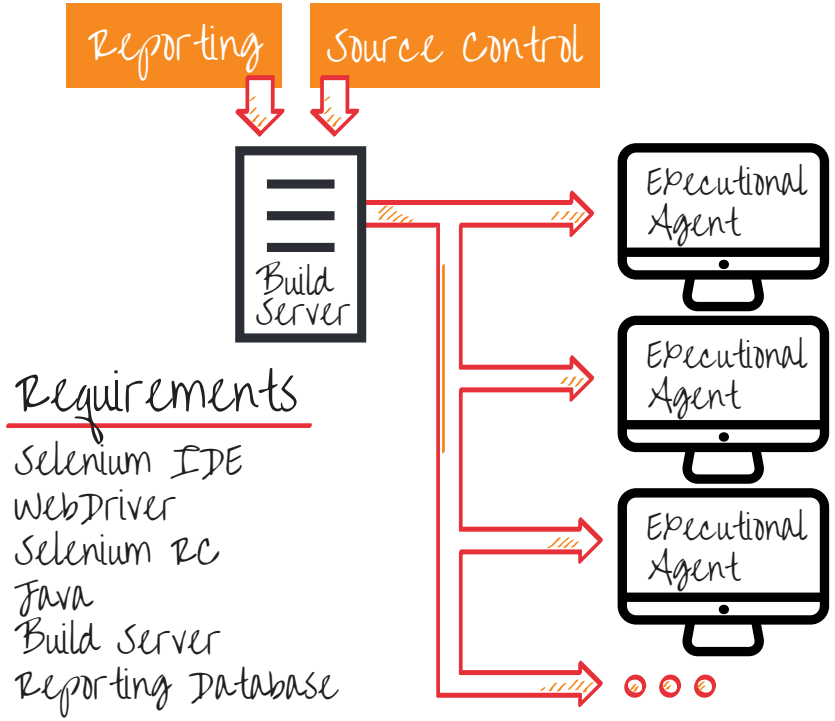
Finally, the ability to mix code together with your recorded steps enables your teams to work as efficiently as possible. Testers can quickly record tests, update locators if needed, and then collaborate with developers for creating code where needed. Developers can focus on writing APIs, configuration steps, etc., and testers can get great test cases designed and started. Both roles can leverage skills in areas playing to their strengths

Test Studio

Selenium IDE doesn't do code, WebDriver forces you to write all code. ***One of the major advantages of Test Studio is in bridging the gap between a pure test recorder like Selenium IDE and pure code solutions like WebDriver.*** Test Studio enables testers to quickly design and maintain tests through recording, then pass them over to developers via source control to assist with more complex, edge-case scenarios. Easily call out to a stored procedure or web service to initialize the system, invoke a helper method to shut off CAPTCHA or include a library of already built infrastructure features. Need to create a set of utility classes yourself? Do it right inside Test Studio standalone, or use the Visual Studio plugin.



SETUP AND CONFIGURATION



SIDE and WebDriver

If you're using SIDE or WebDriver, you'll need to find your own build/scheduling server. If your organization has no build server, then you'll need to begin the process to get one in place: authorization, evaluation, set up and configuration, gaining server proficiency, etc.

Why do I need it?

RUNNING YOUR TESTS

To run your automation suite, you'll need some form of an execution environment. Execution environments are responsible for pulling your test suite out of source control, compiling it and invoking whatever routines are involved with making the test suite actually run.

Generally speaking an execution environment involves some form of a build or scheduling server to handle the steps above, ***as well as*** execution agents to run the test suites on target systems.

BUILD/SCHEDULING SERVER

Build servers are the task masters of your automated software delivery chain. They sit at the center of your execution environment and are responsible for triggering software builds or invoking test suite runs. They of course do quite a bit more, but that's a topic for another whitepaper.

Build servers are usually able to interface with your organization's source control repository to retrieve the latest version of your tests. They also handle test runs scheduled at specific intervals or times.

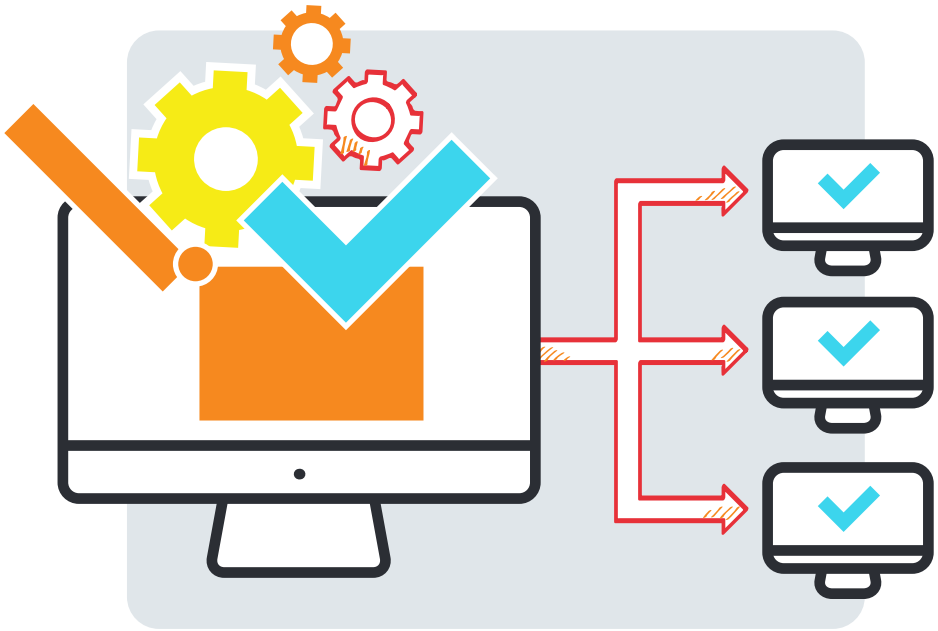
If you already have a build server in place, then getting your automation suite running is a matter of coordination. You'll need to work with the build server owners to get the right tools and frameworks in place, and then define your execution jobs. Note that few build server systems let end users create new tasks. That's usually reserved for build server administrators.

One critical aspect to understand about running WebDriver tests: WebDriver will not run your tests automatically. ***It is a browser driver, and as such WebDriver only interacts with the browser.*** You'll need to wrap all your WebDriver code in a separate framework that will manage test execution and output.

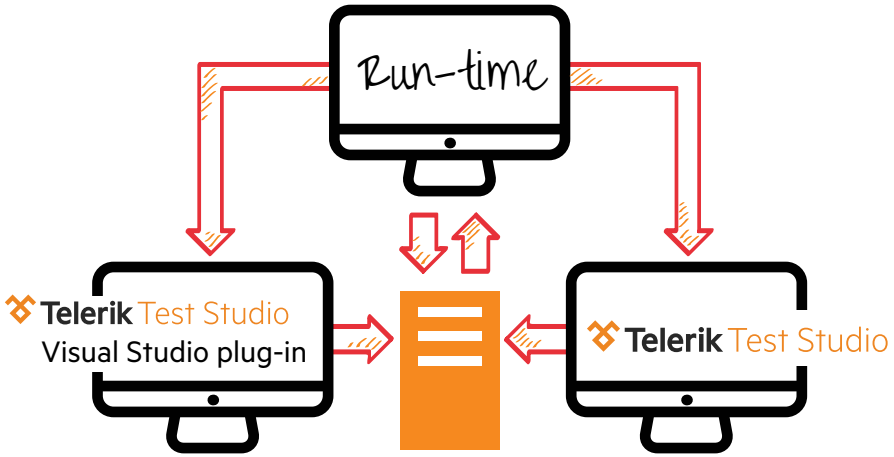
Test Studio

With Test Studio, you don't need to worry about finding or setting up other products to handle your execution. Test Studio ships with a completely integrated centralized storage and scheduling server that handles all build execution and scheduling chores for you. After installation, you can connect to the server and define your own execution/scheduling tasks. In fact, every tester can schedule their own test list runs, meaning no more bottlenecks waiting for the over-burdened build server admin.

Unlike WebDriver, with Test Studio you don't need to worry about extra frameworks, plugins or complex configuration steps to get your test suites running. Just define the tests you want, and run when ready—voila! Plus, if you already have a build server in place, by all means keep it. Test Studio will integrate seamlessly with any Windows-based build server.



SETUP AND CONFIGURATION: EXECUTION AGENTS



SIDE and WebDriver

As mentioned in the section on build servers, Selenium IDE and WebDriver don't run themselves. You need other toolsets to handle executing suites for those. This includes execution agents. Neither ***SIDE, WebDriver, nor the frameworks used to wrap them, offer any form of execution agent.***

Telerik Test Studio

Test Studio offers execution agents with simple and complete integration. Runtime agents can be added into your environment as needed to help you scale test suites and address combinations of operating systems and browsers. Test Studio scheduling and storage services have out-of-the-box support for building substantial grids of execution agents to meet your testing needs.

Why do I need it?

EXECUTION AGENTS TO SCALE OUT AND SPEED UP YOUR TESTING

Execution agents work in conjunction with your build server to execute the tests you've created. Agents are installed on client systems and perform tasks for the build server. Execution agents help you scale your environment to speed up test execution, and are also used to help flesh out testing coverage on different operating systems, browsers and devices.

Often these client systems run in a virtualized environment, either at the organization, or hosted somewhere in the cloud. Agents may also run on mobile devices, such as phones or tablets. Execution agents receive tasking from the build server, perform those tasks, and then report status back to the build server.

REPORTING

SIDE and WebDriver

Neither Selenium IDE nor WebDriver offer any reporting features. To be fair, neither was intended to. Instead, the communities building those tools expect the toolset/environment executing your test automation suite to handle reporting.

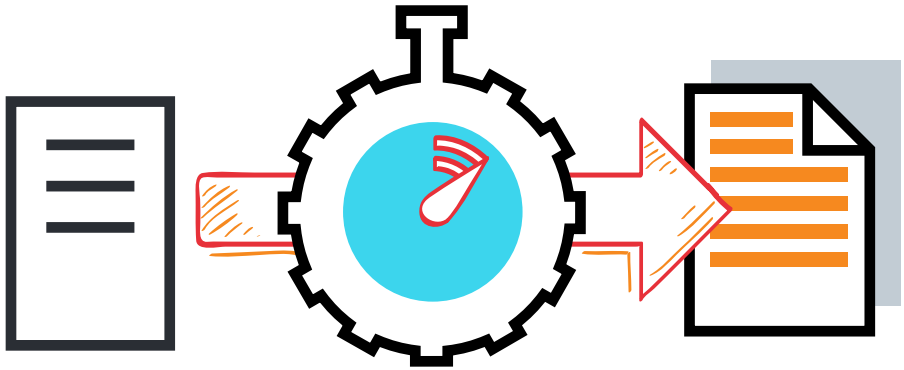
If you want advanced reporting (trends, test state, archiving, etc.), you will either need your build/CI server to support it or find an additional system to provide the reporting you need.

With reporting in mind, you'll need to get your build/CI server to recognize the output of the framework you're executing WebDriver within. Most CI servers natively understand output from the most popular frameworks. However, on occasion you will need to create an intermediary transformation step between the framework and your server.

Why do I need it?

REPORTING: SHARE STATUS AND SPOT ISSUES

How's the state of your application? Are your tests passing or failing? What do the trends look like? Is there a particular configuration or environment causing your system not to fare well? You need reporting to answer these questions and provide stakeholders the information they're looking for.

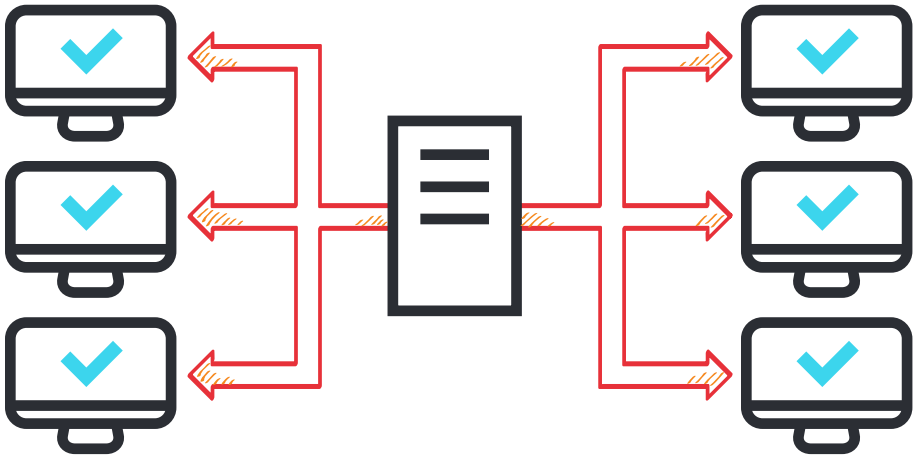


Telerik Test Studio

Test Studio has a rich reporting environment straight out-of-the-box. There's no additional integration to configure between a framework, execution environment and the reporting components. Test suites run, you get reports—it's that easy. You can even have summary information automatically e-mailed out.

Current state, historical trends, and specific status details are all just a click or two away. It's easy to get the information you need to convey to your projects' stakeholders and customers.

PARALLEL EXECUTION



SIDE and WebDriver

Both Selenium IDE and WebDriver require careful set up to support parallel execution. First, you'll have to use a testing framework that supports test suite parallelization. Some popular test frameworks such as NUnit flat don't support parallel execution, so you may encounter some limitations.

Second, your build infrastructure will have to support distributing tests for parallel execution. Your build system has to be capable of splitting out a test suite, marshaling it amongst the various agents, receiving multiple status streams back and finally stitching results from those parallel streams back into one coherent report.

In short, parallel execution is less about SIDE and WebDriver, and more about the toolsets you're using to run them in. As always, you still need well-designed tests for things to flow smoothly.

Telerik Test Studio

Distributing your test lists for parallel execution in Test Studio is simple: Install your execution agents, then check one box when running or scheduling your test lists. ***Test Studio will automatically split your tests across all available execution agents. Tests will run in parallel,*** and you'll then get all status reports automatically stitched back together. Be sure to carefully construct your test lists so there are no dependencies or side effects.

Why do I need it?

PARALLEL EXECUTION TO SPEED UP TEST CYCLES

Let's face it: functional testing at the user interface layer is slow. It's simply the nature of the beast—you've got to fire up a browser, feed it commands and wait for the browser to navigate and perform its actions. This will never be as fast as well-written unit tests. Ever.

As a result, large UI automation suites can easily take tens of hours to run. It's not uncommon to have huge suites that might take 15, 25 or more hours to finish executing in serial.

This is a significant problem because teams should always strive for quick, short feedback loops. Waiting overnight or over the weekend to discover regressions has a serious impact on your team's cost of delivery.

Splitting out test suites to run in parallel requires some careful work. First, you must ensure you have well-designed tests with no interdependencies or side effects. Since you can't be sure what order your tests will run in, you've got to be sure tests won't interfere with each other. Second, you need infrastructure that will handle splitting your test suites and distributing them across supported execution agents. That infrastructure must also handle monitoring the agents and assembling test status reported back from them.

TRAINING AND CONSULTING

SIDE and WebDriver

Finding trainers and coaches for SIDE and WebDriver can take some tricky searching. You'll need to find training providers or consultants able to meet your needs, but securing qualified, experienced coaches can be difficult. Only engage trainers who've done practical field work, and have experience solving real problems on complex systems.

Telerik Test Studio

Telerik Test Studio trainers and consultants are test automation experts with a wide range of practical experience. As experts in several toolsets—not just Test Studio—you can get help transitioning away from SIDE or a struggling WebDriver effort into a solid, valuable path with Test Studio.

Moreover, Telerik has a number of helpful training offerings. Online training will help you quickly get rolling, while experts onsite at your organization can be part of a focused effort to get your automation smoothly integrated into your overall software development process.

We also can help get your implementation work done. Telerik experts can dive in to help you rectify struggling efforts, or simply assist in getting your automation scripts built and running smoothly.

Why do I need it?

TRAINING AND CONSULTING FOR SOLUTION ADOPTION

Even with great tooling, getting a great automation suite in place is a hard effort. That's especially true with teams new to creating functional UI tests. It's a difficult problem space, and lessons and approaches from one situation often don't apply to the next.

Teams new to this sort of testing should be looking for some help and guidance. Inviting a qualified trainer and coach to help the team can be a tremendous boon, both to begin on the right foot, and later as a follow-up resource as the team works their way up the learning curve.

DEDICATED SUPPORT

SIDE and WebDriver

Because Selenium IDE and WebDriver are open source, there is no single authoritative organization to go to for help. The Selenium community is full of people with expert-level knowledge, but how do you engage them? How do you find them to have a conversation? How can you be sure you're getting good information that will sustain you over time versus a hacked together "quick and dirty" solution that's already falling apart at their organization?

There is deep expertise in the Selenium community, with conferences and mailing lists specific to SIDE and WebDriver. But will you be able to reach someone quickly when your delivery chain is falling apart due to broken UI tests?

Why do I need it?

GET YOUR TESTING DIFFICULTIES RESOLVED IMMEDIATELY

Face it: you will need help at some point in your automation effort. You'll be unable to figure out the proper element to latch onto during an asynchronous AJAX call, or get stuck solving how to find a node element in a dynamically generated tree control. Maybe it's a larger problem, like trying to figure out why your build server isn't pulling the latest versions from source control, or your test lists aren't scaling out properly.

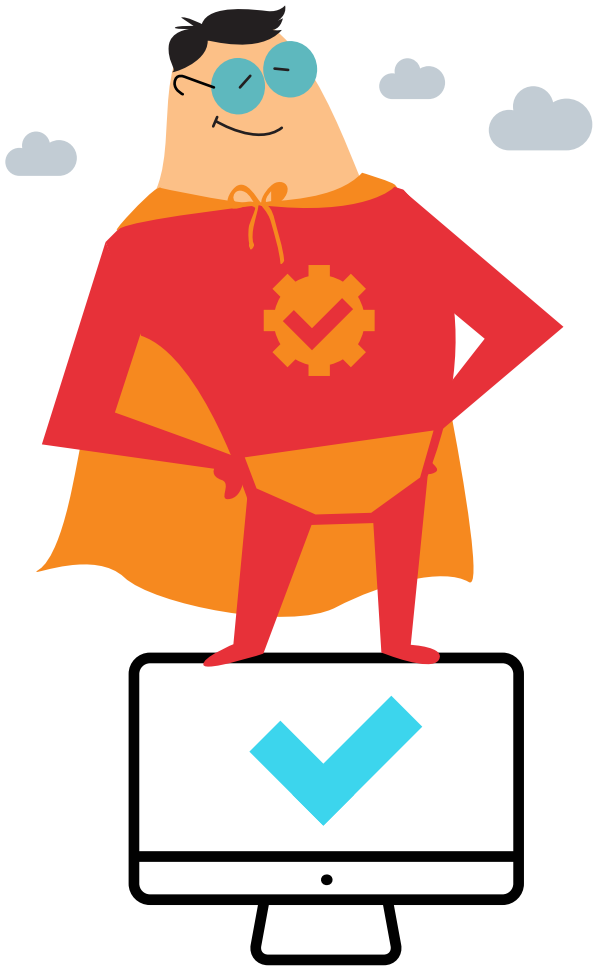
At some point your new-to-automation team will need to track down experts. Guaranteed. ***So how do you go about that?***

Test Studio

Telerik support is known, respected and outright loved across the industry. Telerik support teams regularly win awards for customer satisfaction, and are readily available to help you work through problems you encounter. Additionally, the guidance you're getting is coming from product experts with years of experience working with the toolset in a number of complex environments.

Moreover, there are a number of support channels to get you the right help you need:

- Professional support channels with quick turnaround times. Open a ticket from your Telerik.com account—or directly from within Test Studio—and you'll get speedy responses.
- Live, personal support calls. Telerik offers Test Studio customers direct, personalized support channels. You can arrange calls directly with our support staff for quick, expert help right away.
- Hosted forums where a vibrant community of Telerik users hang out and discuss issues. These forums are monitored by Telerik support staff who chime in daily with responses and guidance.



CLOSING

Selenium IDE and WebDriver are tools for functional testing that are well-suited to specific cases and can help teams deliver good software.

While free, it's important to remember there are still significant costs associated with using either as your solution for the critical functional testing required as part of your value delivery chain.

The Telerik Test Studio subscription pricing model offers an extremely cost-effective solution to help you create powerful, low-maintenance test suites that will easily grow and adapt to changes in your system. You'll deliver better tests more quickly, and with more value in the long run.

Ready to give Telerik Test Studio a try? Download your free 30-day evaluation copy now.

Download trial

About the author

Jim Homes is the Developer Advocate for Telerik Test Studio, a complete testing solution that helps teams deliver better software. He is co-author of "Windows Developer Power Tools" and Chief Cat Herder of the CodeMash Conference. Find him as @aJimHolmes on Twitter.