EY self-healing automation

Building a better working world

TABLE OF CONTENTS

Executive summary

While test automation has long been recognized as an engine for efficiency and competitive advantage, each new wave of technology transformations and application upgrades has posed questions for leaders: How will this help us, and to what extent? Where and when should we implement? What upfront investment is required to realize optimal benefits in terms of streamlined processes and long-term savings?

The latest test automation tools and frameworks, while holding great promise, are accompanied by a significant obstacle. When scripts are confronted with the constant changes that come from the fast pace of application development (especially in an agile and DevOps world), most fail to work, and the advantages of automation can be quickly offset by the need to conduct manual troubleshooting, debug, fix and generally maintain test automation scripts. Such interruptions are enough to sour executives on the cost-benefit case for investing in test automation scripts. Automation is therefore perceived as something that:

- Slows down testing due to constant maintenance
- Requires more people to perform maintenance
- Shouldn't be adopted until late in development for fear of too much change and resulting maintenance
- Ultimately costs more than it's worth

But what if the need for manual maintenance could be removed?

Current state of manual maintenance

Unfortunately, "fragile" is a word all-too-commonly associated with test automation scripts. When a script breaks, manual object identification maintenance can take up to 15 minutes per occurrence. A script breaks when object properties change, and an automation engineer must stop developing new scripts to troubleshoot and fix the broken one. The team manually inspects or spies the object to see the new property value or find new properties to use, then updates the script or object repository accordingly and reruns the script.

The math is daunting: One application deployment per week could encounter around 35 object changes (which varies greatly based on application maturity, development methodology, size of project, etc.). At 15 minutes per manual fix, the result is more than one person's full workday — 8.75 hours — spent per week on basic automation maintenance.

Self-healing automation

But it doesn't have to be that way. Self-healing automation is a solution that addresses the No. 1 cause of test automation script maintenance: object changes. The "object," in this context, is an item in a script — such as a button or text box on a webpage — that the script (or the user) would interact with to perform tasks. Scripts must be able to unique identify which object it needs to perform an action on — which text box should it put your username into? Just as a person can be identified by physical attributes such as size, hair color or eye color — or by other relative means ("that person we saw at the store yesterday") — objects must also be uniquely identified in some way. And, just as people's appearances can change to the point that they aren't recognizable to others, objects that no longer fit their original "description" can confuse traditional automation scripts. When that happens, scripts break and downtime accumulates.

Self-healing employs data analytics to identify objects in a script even after they have changed. The result is a system that goes far beyond the "Band-Aid" approach often written into scripts, such as the use of wildcards or regular expressions to handle variation in object names or identifiers. Rather than relying on those methods — and allowing productivity to grind to a halt anytime they fail — the self-healing approach introduces a higher level of intelligence and analysis.

When your script fails due to being unable to find the object it expected, the self-healing mechanism provides a fuller understanding and analysis of options. Rather than shutting down the process, it examines objects holistically, evaluates attributes and properties of all available objects and uses a weighted scoring system to select the one most similar to the one previously used. Self-healing can scrape, evaluate and choose among 10 objects in less than 0.05 seconds. Stop-and-go syndrome is effectively cured.

The self-healing difference can be fully realized as it:

- Changes the mindset regarding automation approaches
- Allows automation efforts to start earlier as fears of maintenance subside
- Automates the maintenance process itself in real time
- Improves or preserves the return on investment

Changing leading practices

The implications of self-healing are wide ranging. Test automation leading practices say you should not use especially fragile methods to identify objects such as absolute XPaths and certainly never use something as lengthy and specific as the full outerHTML. These would constantly change and be very difficult to maintain manually.

With self-healing, the opposite can be true: absolute XPaths and other very specific methods of identification can be — maybe should be — used. Since self-healing could fully maintain object identification automatically in fractions of a second, why not automatically build absolute XPaths (yes, we can do that) to use that to solely identify objects in your framework? It would standardize and simplify methods and mechanisms in your framework, provide very specific and accurate objects and handle circumstances where properties no longer exist (i.e., a text box used to have a name "txtUserName" and now has no name property at all) or perhaps there's no unique identifier at all.

This wouldn't work in all circumstances, but an interesting application of self-healing is that it can fundamentally change the way your framework handles objects. Self-healing would run more often, but would you care? How much time would you save from having to follow normal processes for object identification and maintenance? When self-healing runs in a fraction of a second, will you even notice?

How does self-healing work? How can this be possible?

Changing the mindset

Now, what if I told you that you didn't have to uniquely identify objects anymore?

You take the blue pill: you go back to your everyday life, writing automation scripts like you always have. Manually identifying objects, endless maintenance and updates. Poor development practices sometimes even making it impossible to even identify objects at all.

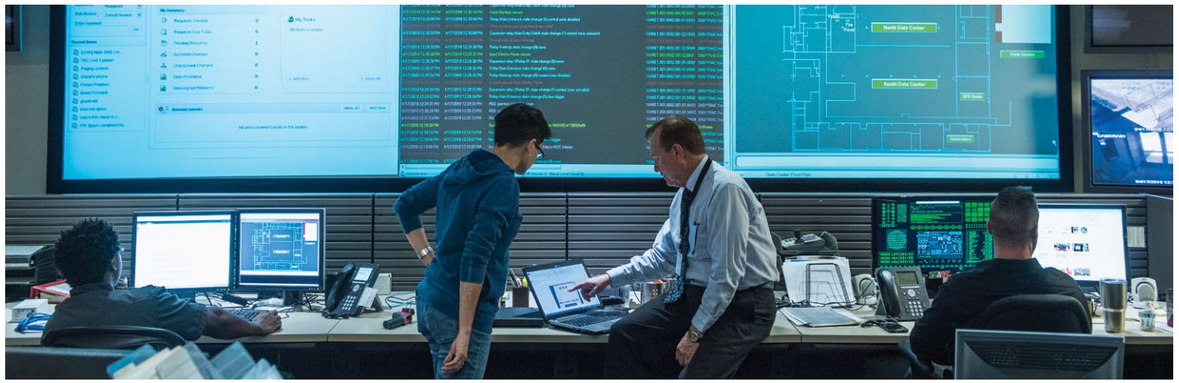
You take the red pill: let's see how far this rabbit hole goes. You are freed from the painstaking maintenance, and now your scripts can be used across different applications without making any changes.

The status quo in test automation is that every object must be identifiable based on any number of specific ways. Going back to the human analogy, a person can be identified by height, hair color, eye color, clothing or other characteristics. But what if you don't need to pinpoint a specific person, but rather someone who demonstrates some of those characteristics (6 feet tall with brown hair)?

Self-healing automation represents a change in the mindset of test automation, dispensing with the need to always uniquely identify every specific object. Self-healing can take more general information — size, general location on a page, text and other characteristics — and find the closest match for the object.

You could start to write generic scripts to perform tasks as long as you know enough about an object. For instance, a script to log in could be written to simply look for a text box with a name similar to "username" and another text box below it with a name similar to "password" and then find a button with a name or text similar to "submit" or "log in." Now you have a login script that could work on thousands of applications.

This capability effectively turns today's concept of test automation on its head.



How self-healing works

With self-healing automation, the same number of object changes per week mentioned earlier (35) — which could take more than a full workday to handle manually — can be remediated in a fraction of a second. When an object's properties change, the self-healing tool springs into action, scraping similar objects on the page and comparing them to the previously stored historical data on that object. Data analytics and custom property weighting are deployed, enabling the tool to predict the most probable object being sought. The automation script continues running, using new object properties and an automatically updated object repository to head off future errors. The default weights assigned to each property name can be modified.

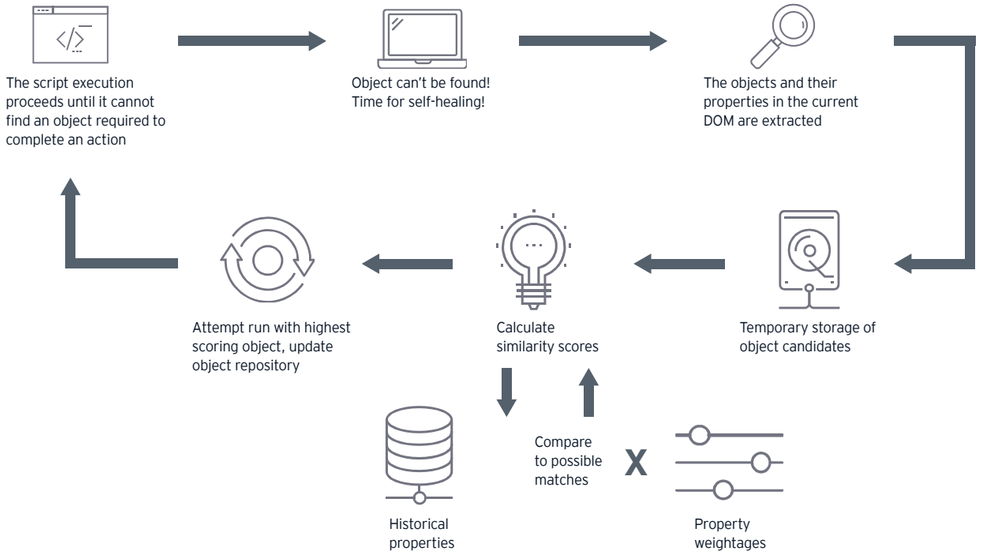
Ultimately, empowered automation engineers will turn their attention to use cases that seemed too aspirational in the past. They can:

- Create generic scripts regardless of application or objects, resulting in much higher reusability
- Reduce the effort to update automation scripts to work after application upgrades
- Reuse more automation scripts based on out-of-the-box (OOTB) functionality
- Use the same scripts between environments
- Use the same scripts between clients/projects (important for consulting firms and our clients to accelerate testing by reducing initial ramp-up time for automation efforts!)

The self-healing process

The process is straightforward. When an object cannot be found (due to a property name or value change), the failed object is fetched from a historical object repository file, along with all its property names and values. All similar objects (such as all other text boxes) that do exist on the page are scraped, including all their properties and values and saved into an "Object Capture" table. Self-healing will use various similarity scoring algorithms to evaluate how similar each property is between the missing historical object and the available objects on the page. Each property is given a similarity score (while also considering a customizable weightage so you could say that "name" is more useful to identify a match than "color," for example) and ultimately each possible match is given a total score.

Self-healing will return the object and its properties with the highest score for use. The script can then attempt to identify the new object, continue on in execution and update framework repositories with the new object information, as long as everything works.



Integration ability

One of the hallmarks of the self-healing solution is the flexibility it offers in terms of implementation. If a business opts for the pre-built EY (Ernst & Young LLP (EY) framework, it can reap the benefits of a ready-to-use system and sharply reduce the initial setup time. Our pre-built framework also offers the advantages of pre-built features for items such as common test management integrations, more than 200 functions for web, mobile, API, DB, PDF, accessibility; and other testing needs right out of the box, such as reporting and popular behavior-driven development (BDD) components. The result is a sizable head start for your business, where as much as 50%-60% of your automation needs can be started within three hours from saying "go." At the same time, a pre-built framework does not mean everything has to be reinvented. It still allows potential reuse of existing scripts if written in a compatible language such as Gherkin.

Unlike proprietary automation tools that claim to use machine learning or other self-healing mechanisms, this solution also can be integrated with an existing framework by being directly integrated into Java-based frameworks or via microservice/API for other tools, such as MicroFocus Unified Functional Testing (UFT). Such efforts will vary based on the existing framework's architecture and maturity, but will still allow the reuse of current scripts and will not require new tool knowledge. The tool has been crafted to be flexible and universal — usable with any nonproprietary test automation tools.

Common questions

Discussions of this solution frequently give rise to questions such as the following.

How does it handle objects that are extremely similar, such as Address 1 and Address 2 fields?

The first thing to try is adding more properties to compare that may help differentiate these objects. In this example, they are both text boxes, have similar names and sizes and locations — but what is different about them? Perhaps considering which is mandatory would help correctly identify them, or more heavily weighting location.

How configurable is it?

It is entirely configurable, either by using config files or by directly editing the code. In the configuration files, you can configure properties to scrape and determine their weight. In the code, you can change the similarity scoring mechanism, the input/output formats (JSON by default) and other mechanisms.

Other tools claim to have self-healing capabilities. How is this different?

Other tools are proprietary and siloed — they must be purchased from the vendor, with everything done the vendor's way. To use their "self-healing" mechanisms, you must totally transition to their tool, which is a significant effort for large organizations. Many of these tools, while more modern in their approach and capabilities, are lacking the robustness needed by most large organizations. Our self-healing utility can be integrated with existing tools, reused and customized. It provides a more direct, scalable solution that has the flexibility to work differently with other tools or applications.

Are there any known limitations or issues?

The primary application is for web-based applications, so further adaptation would need to be done to apply to other platforms (native mobile apps, desktop, etc.). In concept, this applies to those as well as robotic process automation (RPA), since they identify objects in similar ways for the most part (some tools support image cognition-based identification that would not be covered here). However, most RPA tools are proprietary and closed off and do not allow for integration of external utilities such as this. Even if they allow custom code to be run, that code cannot manipulate their internal object repository information.

Why not use machine learning or other concepts for self-healing?

While it could be accomplished using machine learning, it introduces a greater level of complexity to the process. Machine learning (ML) requires large amounts of training data to be accurate and then is only accurate based on that training data. Different applications, projects and automation frameworks will have different quirks — or "features" as they are called in the data science world — that the ML model may not adapt well to. If a new property needs to be assigned to an object, machine learning requires a change to its model, new training data and a comparatively cumbersome procedure. The self-healing automation solution, by contrast, provides a lighter-weight, more direct approach that is eminently more controllable and flexible.

What if it doesn't work?

While we've found that our self-healing mechanism is accurate more than 80% of the time, much depends on the application, the scripts and other factors. It's flexible enough to add more properties to aid in comparison accuracy, as well as weight the importance of properties (such as determining that a similar name is more important than a similar color). In a worst-case scenario, if self-healing identifies the wrong object, you are no worse off than before — you'll have to perform maintenance manually for that one instance. So why not try it? We have not encountered this, but even if it worked only 25% of the time, that's 25% less manual maintenance.

What's the future of self-healing?

While the benefits of self-healing can be realized today, we are working to broaden its reach and utility.

New implementation approaches and use cases continue to come up. For example, once you realize and apply the power of self-healing to your framework, you may realize that your approach to automation can change entirely, as mentioned earlier. We are applying self-healing in unique ways to help our clients accelerate test automation in ways that were previously not possible:

1. Start a project with 50% or more of testing already automated by using self-healing to adapt OOTB platform scripts to work with their specific implantation

2. Create generic scripts that can be used on any application (logging in, for example)

3. Start automation earlier, since the fear of maintenance is reduced

Furthermore, as more RPA tools are developed in the future that don't have the proprietary restrictions of those used today, you can look for this unique technology to be extended to RPA and other robotics concepts.

Right time, right place

By creatively and aggressively confronting the traditional problems associated with object changes, our self-healing approach differentiates itself in the world of test automation. If you've struggled with getting value out of test automation or shifting testing left in the SDLC in DevOps or agile, self-healing could help. The great promise of next-generation automation echoes that of previous automation advances — to boost operating efficiency and liberate people from tedious tasks. By removing the frequent manual interventions that accompany test script object changes, self-healing automation can deliver on that promise and provide companies with the competitive advantage they need to thrive in a digitally transformed marketplace.

Want to learn more?

If you have specific questions related to your situation, or would be interested in a demonstration, please reach out to John McEvoy at john.mcevoy@ey.com

EY | Assurance | Tax | Transactions | Advisory

About EY
EY is a global leader in assurance, tax, transaction and advisory services. The insights and quality services we deliver help build trust and confidence in the capital markets and in economies the world over. We develop outstanding leaders who team to deliver on our promises to all of our stakeholders. In so doing, we play a critical role in building a better working world for our people, for our clients and for our communities.

EY refers to the global organization, and may refer to one or more, of the member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients. Information about how EY collects and uses personal data and a description of the rights individuals have under data protection legislation is available via ey.com/privacy. For more information about our organization, please visit ey.com.

For more information about our organization, please visit ey.com.

Ernst & Young LLP is a client-serving member firm of Ernst & Young Global Limited operating in the US.

US SCORE no. 06363-191US
1904-3108678

ED None

ey.com