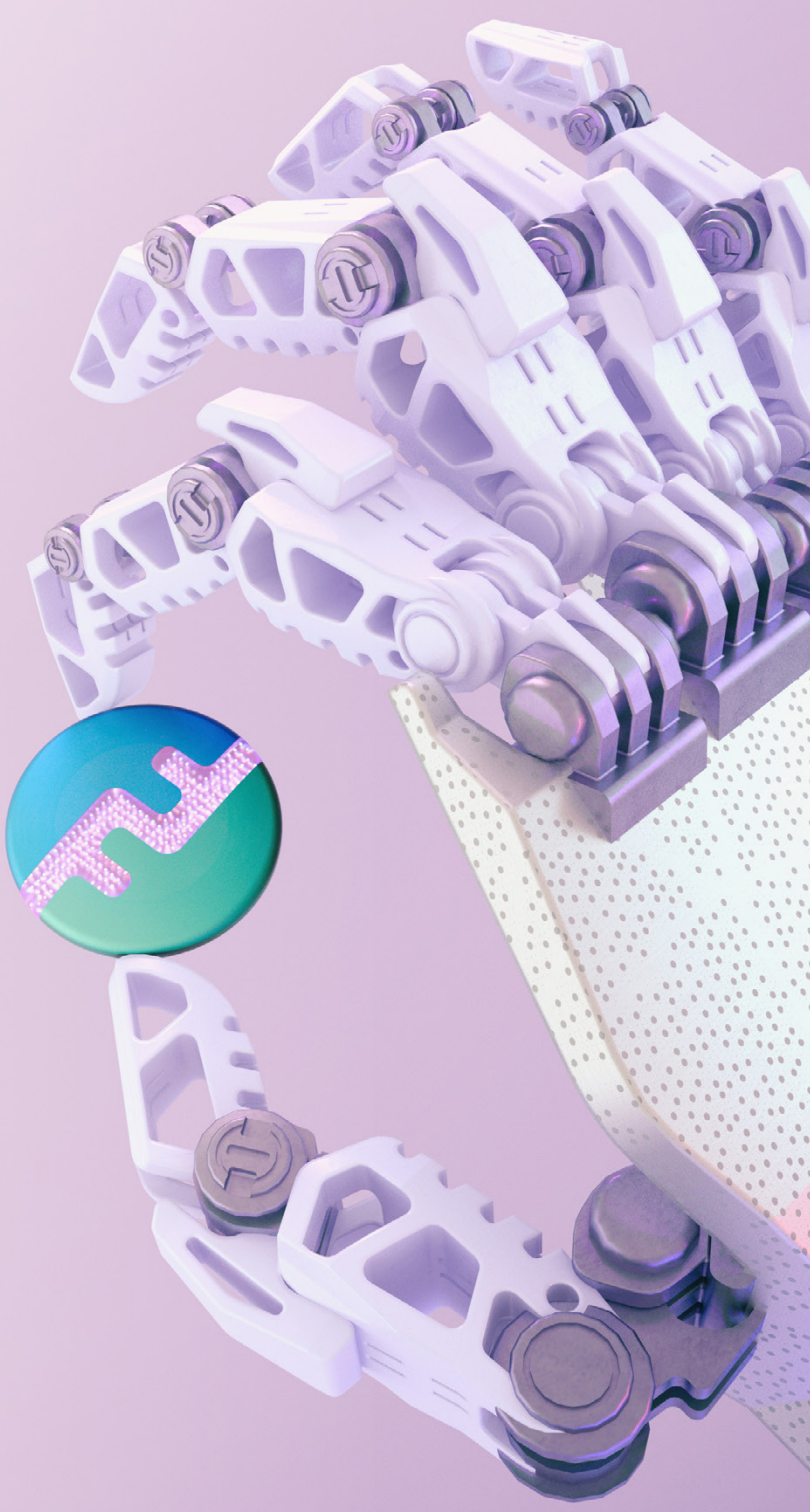
functionize



eBook

# **How Autonomous** Should Your **Testing** Be?

> **Automation (noun)**
> the technique of making an apparatus, a process, or a system operate automatically

Automation is revolutionizing the worlds of business and IT. The world of testing is no exception. From simple scripts to recorders, and systems that use artificial intelligence, test automation allows testing to be done at scales that were impossible just a few years ago. This is driving the creation of more complex applications and enabling new development models such as CI/CD.

In this eBook we discuss how you can assess the level of automation your test tool provides, and we introduce a new model for levels of automation in testing. Using this model will allow you to compare automated testing solutions on a like-for-like basis.

# Creation, Execution & Analysis, and Maintenance are all candidates for test automation

# Introduction

Automation, or the process of getting a machine or computer to perform its tasks without the need for human intervention, has revolutionized manufacturing over the past two centuries. Automated processes control the manufacture of computer chips, pick and place machines assemble circuit boards with a precision no human can achieve, robots assemble cars so fast that some car plants can produce a new car every 2 minutes.

> A new car leaves the Nissan factory every 2 minutes

Since the turn of the Millennium, automation has been used to solve problems in many new fields. Robotic Process Automation is now routinely used by big business to improve the efficiency of routine and repetitive tasks by using semi-intelligent 'robots' to perform the tasks. The world of testing is no exception. From simple scripts to recorders and even systems that use artificial intelligence, test automation now allows testing to be done at scales that were impossible just a few years ago. In turn, this is driving the creation of more and more complex applications and enabling new development models such as Continuous Integration/Continuous Delivery.

> Nowadays, test automation systems often leverage AI and ML

But how can you know what your automated testing solution is actually achieving? How can you compare different solutions from different providers? One way is to use a formal taxonomy or model for the level of automation being provided. In this eBook we will introduce several models, including the well-known SAE taxonomy for levels of automation in self-driving vehicles.

> Test automation ranges from manual to autonomous testing

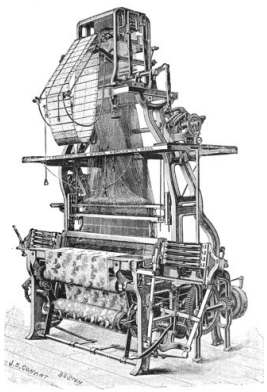
Our model is loosely based on the SAE one, and ranges from level 1 (manual testing) to level 5 (autonomous testing). To add to the utility of the model we define 3 areas where automation can be applied in testing. These are Creation, Execution & Analysis and Maintenance. Thus, a system may achieve level 4 for test execution, but only level 1 for creation.

# A Brief History of Automation

As an idea, automation isn't new. However, the idea has been slow to evolve, and it is only recently that it has started to reach its full potential.

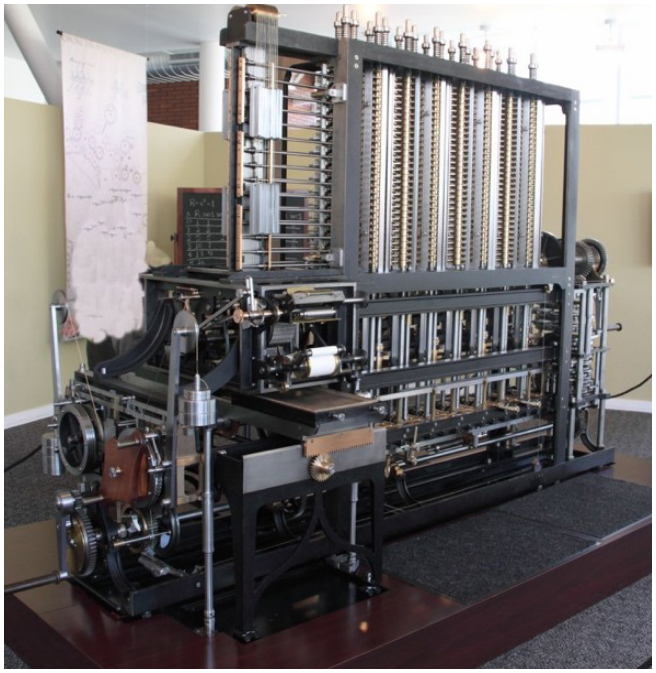
## Automation before 1900

Before the 20th Century, automation was relatively rare. Partly this was for technical reasons (the technology was simply too basic) and partly for socio-economic reasons (labor, both skilled and unskilled, was cheap and plentiful). But there were a few notable examples of automation where machines were able to replace relatively skilled workers.



In 1725, Basile Bouchon invented a new form of loom that could be controlled using a punched paper tape. However, the technology wasn't very reliable, and it wasn't a success. Then in 1804, Joseph Marie Jacquard created his own programmable loom based on a chain of wooden blocks with holes. These controlled the order in which threads were woven. This process allowed highly complex patterns to be automatically woven into cloth, and the process is so good that Jacquard looms are still used to this day.

Another significant advance was Charles Babbage's difference engine. This mechanical calculator was meant to speed up the calculation of astronomical and mathematical tables. These were key for nautical navigation but calculating them by hand took dozens of skilled "computers" (as people that did the calculations were called). Moreover, doing them by hand led to a real risk of errors. The project was so important, it was funded by the British Admiralty to the tune of £17,000 (well over a million in today's money). Ultimately, Babbage's engine never worked as expected, but it paved the way for later working difference engines, such as that developed by Per Georg Scheutz in 1853.



## Saboteurs and Luddites


From the earliest days of automation, workers have viewed the technology with suspicion and fear. In the 19th Century, this fear bubbled up into outright violence and sabotage. Indeed, the word sabotage was coined at this time and referred to French workers who deliberately wrecked industrial machinery by throwing in their wooden clogs or sabots. People frequently believe that sabotage derived from this use of sabots. However, it is now thought it comes from a different root, “saboter”, meaning to botch, bungle, or wreck.

Sabotage was a common tactic in industrial disputes for many decades. Famously, it was one of the tactics used by the Luddites, a secretive brotherhood of textile workers. The Luddites were protesting at the perceived use of automation to harm their job prospects and to get around labor laws. Of course, we now know that automation transformed the whole labor market. There was no mass unemployment and it enabled everyone to get access to goods that were previously luxuries. This suspicion against automation continues even today. People routinely distrust decisions made by machines on their behalf. They believe AI will cause a mass loss of jobs and instinctively distrust anything that is powered by it.

## Automation in the 20th Century

The 20th century saw a huge surge forward in automation, especially in manufacturing and industry. The first step was the invention of the production line and the realization that most complex tasks could be broken down into simple individual steps. One of the pioneers of this approach was Henry Ford. At first, this approach meant previously skilled jobs could be done by unskilled workers. But it paved the way for some of those tasks to be fully automated.

The Second World War saw a concerted push to improve manufacturing, and much effort was spent on developing automated production lines, especially for armaments and aircraft production. The war also spurred the development of programmable computers and saw control theory become an established approach for systems design.

After the war, automation developed apace. Robots began being developed to work on production lines from the 1950s. At first these were relatively simple, but they became increasingly complex, and autonomous. At the same time the concept of computer numerical control (CNC) was being developed. This allows machines to be programmed to complete their tasks automatically, for instance milling highly complex machine parts with no human input.

### Automation Today

Automation has come a long way. Nowadays we have reached the stage where factories can produce a new car every 2 minutes, autonomous mining vehicles are allowing coal mines to operate without the need for humans, and electronic fabrication labs can assemble circuits from components so small a human simply cannot handle them.

But automation is no longer just the preserve of manufacturing and industry. Automation is now routinely applied to other business uses. One example of this is Robotic Process Automation. Here, simple, routine, and repetitive business tasks are done by software robots, replacing humans who are liable to make mistakes, need rest breaks, and can only work a few hours a day. RPA is still a developing field, but RPA is now capable of replacing complex ERP (enterprise resource planning) systems with a completely autonomous system.

## Levels of automation

As we see, automation is nothing new. But how do we actually measure it properly? How can you compare two systems that both claim to achieve automation? A number of researchers have created models to assess the level of automation a given process or system achieves. Here we will describe some of these.

### Decision-based models

In their 1999 paper A Model for Types and Levels of Human Interaction with Automation, Parasuraman, Sheridan, and Wickens identified the four major tasks that need to be

completed in order to take any action. These are:

- information acquisition or sensing
- information analysis
- decision and action selection
- action implementation

**Endsley and Kaber** present a model for automation based on whether these tasks are performed by the human, the computer, or both.

| Level | Monitoring | Generating | Selecting | Implementing |
|---|---|---|---|---|
| **Manual control** | Human | Human | Human | Human |
| **Action support** | Human/Machine | Human | Human | Human/Machine |
| **Batch processing** | Human/Machine | Human | Human | Machine |
| **Shared control** | Human/Machine | Human/Machine | Human | Human/Machine |
| **Decision support** | Human/Machine | Human/Machine | Human | Machine |
| **Blended decision-making** | Human/Machine | Human/Machine | Human/Machine | Machine |
| **Rigid system** | Human/Machine | Machine | Machine | Machine |
| **Automated decision-making** | Human/Machine | Human/Machine | Machine | Machine |
| **Supervisory control** | Human/Machine | Machine | Machine | Machine |
| **Full Automation** | Machine | Machine | Machine | Machine |

Similarly, in his 1980 paper, Computer Control and Human Alienation, Thomas B. Sheridan looked at the 10 levels of automation when it comes to decision making:

1. The human considers the alternatives, makes and implements the decision.
2. The computer offers alternatives which the human may ignore in making the decision.
3. The computer offers a restricted set of alternatives, and the human decides which one to implement.

4 The computer offers a restricted set of alternatives and suggests one, but the human still makes and implements the final decision.

5 The computer offers a restricted set of alternatives and suggests one, which it will implement if the human approves.

6 The computer makes the decision but gives the human an option to veto the decision prior to implementation.

7 The computer makes and implements the decision but must inform the human after.

8 The computer makes and implements the decision and informs the human only if asked.

9 The computer makes and implements the decision and informs the human only if it feels this is warranted.

10 The computer makes and implements the decision if it feels it should and informs a human only if it feels this is warranted.

In both models, the process of automation involves making and acting on decisions. This is important, since this decision-making is usually what sets humans apart from machines.

## Locus of control models

Several models look at automation from the point of view of who is in overall control of the process. For instance, in his 1995 model, Draper defines five levels of automation:

| Level of Automation | Description |
|---:|---|
| **Human control** | Total human control of process |
| **Manual control with intelligent assistance** | Human control and teaching with machine modification of control inputs. |
| **Shared control** | Human control and monitoring with machine control of routine subtasks |
| **Traded control** | Assignment of subtask control to the human or machine depending on characteristics of task and server capability. |
| **Strategic control** | Human long-term planning with machine performing tasks |

In a similar fashion, Milgram et al. defined a model for levels of automation in manufacturing (specifically looking at the control of manufacturing robots):

| Level of Automation | Description |
|---|---|
| **Manual Teleoperation** | The human has to remain continuously in the control loop |
| **Telepresence** | Master-slave systems where human initiates all actions |
| **Director/agent control** | Human directs the robot which uses limited intelligence to do the task |
| **Supervisory control** | Human now in a supervisory role with robot performing all tasks |
| **Autonomous robotics** | Human has no role in operating the system at all |

# The SAE Taxonomy for Self-Driving Vehicles

One of the most famous machine-centered models is the Society of Automotive Engineers' **taxonomy** for the levels of automation in self-driving vehicles. They define 6 levels of automation if you include level 0 which is no automation. This is a locus of control model.

0 **No Automation.** Here the driver is responsible for all vehicle interactions and receives no assistance from the vehicle.

1 **Driver Assistance.** Here the vehicle can only assist with things like cruise control and lane keeping.

2 **Partial Automation.** Here the vehicle takes over more elements of steering and acceleration/braking, but only in tightly controlled circumstances. For instance, automated lane changing or self-parking.

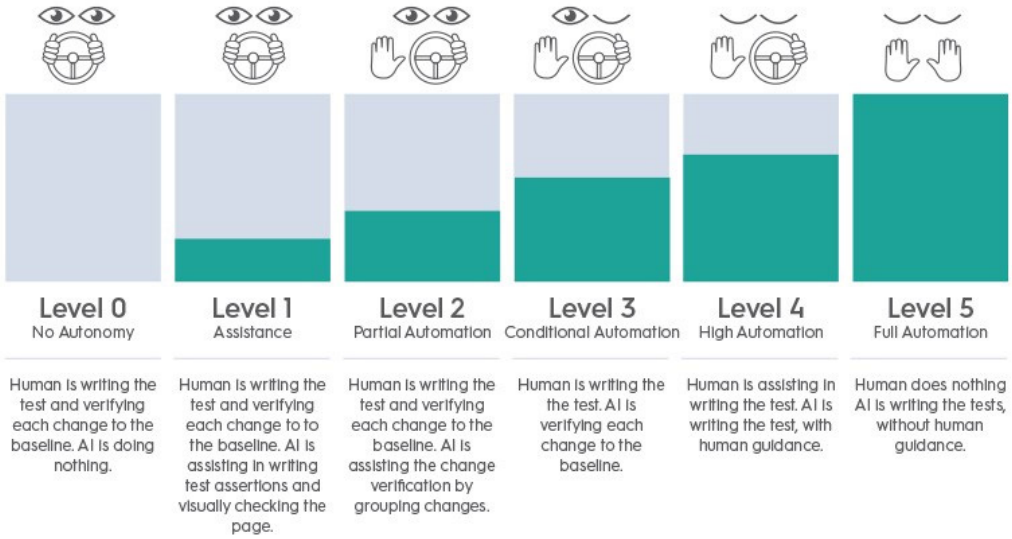
3 **Conditional Automation.** Here the vehicle will not only control steering and acceleration/braking, it will also monitor the environment and warn the driver if they need to take back control. This is similar to the autopilot mode in Tesla vehicles.

4 **High Automation.** Here the vehicle performs most of the driving tasks itself, so long as it stays within defined use cases like driving between two known locations. The human driver takes no active role at all but is still able to take back control.

5 **Full Automation.** At this level, the vehicle performs all driving operations completely autonomously. There is now no human driver and, indeed, there are no human-operated controls in the vehicle.

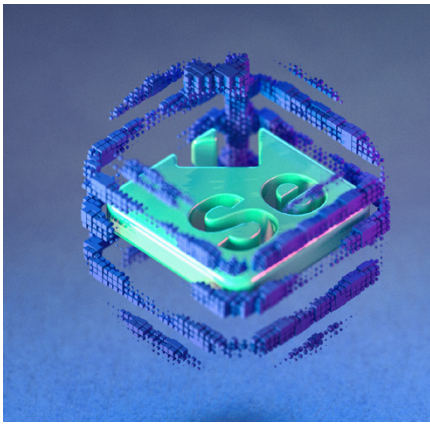
This taxonomy has the benefit of being simple, but clearly it has a very narrow focus. While the job of creating self-driving vehicles is undoubtedly complex, the actual action of driving

is relatively straightforward—after all a human can be taught to be a competent driver with just a few hours intensive tuition.




Clearly this is a very simplified model and only addresses some aspects of test automation.

# Test Automation



Automation took a long time to enter the world of software development. One of the most significant areas where automation has made an impact is software testing. From its creation in 2004, Selenium began to revolutionize the world of software testing. Rather than test complex UIs manually, Selenium offered a new approach that allowed you to script tests and run them automatically. This transformed how UI testing was done, and, arguably, helped drive the smartphone apps revolution. Prior to Selenium, manual UI testing was the single most-intensive job for any QA engineer. After Selenium, routine UI regression testing suddenly became far easier, and testing at scale became feasible.

## Modern Test Automation

Over the years we have seen increasingly complex systems that are able to perform cross-platform testing, test at scale, and where you can record tests using UI plugins. Nowadays, test automation systems increasingly use AI to help create, run, and analyze your tests.

Some companies aim to reach a stage where all routine testing (e.g. regression testing) can be done completely automatically, freeing up QA engineers to focus on manual progression testing. Other companies aim to demystify automated testing, making it accessible to everyone on a development team and thus achieving the goal of shifting testing left. Yet other companies see a world where testing becomes entirely automated end-to-end.

## Test Creation 

Creating tests is one of the hardest parts of test automation. First, you have to define your test plan, being cognizant of the limitations of the system you are using. So where in a normal manual test plan you might say “search for XYZ”, in a plan for test automation you need to spell this out in steps: “Locate the search box at the top right; Enter ‘XYZ’; Click on Search”. Second, you have to take this plan and convert it into a test script. With the original version of Selenium, you had to be a competent programmer to do this step, since it involved hand-coding a complex script that tells the system exactly what selector to choose at each stage, and what action to take.

Because of the complexity of doing this manually for any but the simplest of tests, recorders were developed to help. One of the earliest of these was Selenium IDE, which used a Firefox plugin to enable you to record the steps in a UI interaction and then generated the correct Selenium script from this. Since then, numerous improvements have been made, and modern test recorders are robust and produce scripts that run across multiple platforms. These often rely on some element of AI or machine learning.

## Test Execution 

Automated test execution is what most people think of when they hear test automation. Here, the system takes a test script that has been defined by a human and runs it automatically. This may be on a single system with a given browser and OS combination. or it may use something like Selenium Grid to test across multiple browsers and even OSes simultaneously.

In general, people tend to think of test automation for UI testing. Firstly, that is the problem Selenium solved and, for many people, Selenium is synonymous with test automation. Secondly, UIs lend themselves well to automated testing. Thirdly, it is relatively easy to create general-purpose systems for automated testing of UIs. Thus, there are a large number of companies offering such systems.

### Test Analysis 

Analyzing the results of a test is a critical step. Without this there would be little point in actually doing the testing! Automated testing tends to do this simplistically. Generally, the system checks that the final result of the test is as expected, either using screen shots, or by capturing some aspect of the final state of the UI (e.g. is the correct item in the shopping cart). In other words, tests are purely pass/fail. Some systems go further and use screenshots to highlight to the user where the failure is. For instance, highlighting if a button is missing. In most systems the analysis and the execution of the tests are closely linked.

### Test Maintenance 

Test maintenance is an interesting case. While test plans have always needed some element of maintenance, it was only the introduction of automated testing that made test maintenance a significant issue. This is because with most automated tests, even minor changes to things like page layout or button style can cause tests to fail. Suddenly, test failures are more likely to be because of test maintenance issues than because of bugs. Anecdotally, this sort of maintenance takes more of a test team's time than creating new tests.

## Models for automation in testing

Now let's look at how the idea of levels of automation can be applied to automated testing.

### The Gartner model for test automation

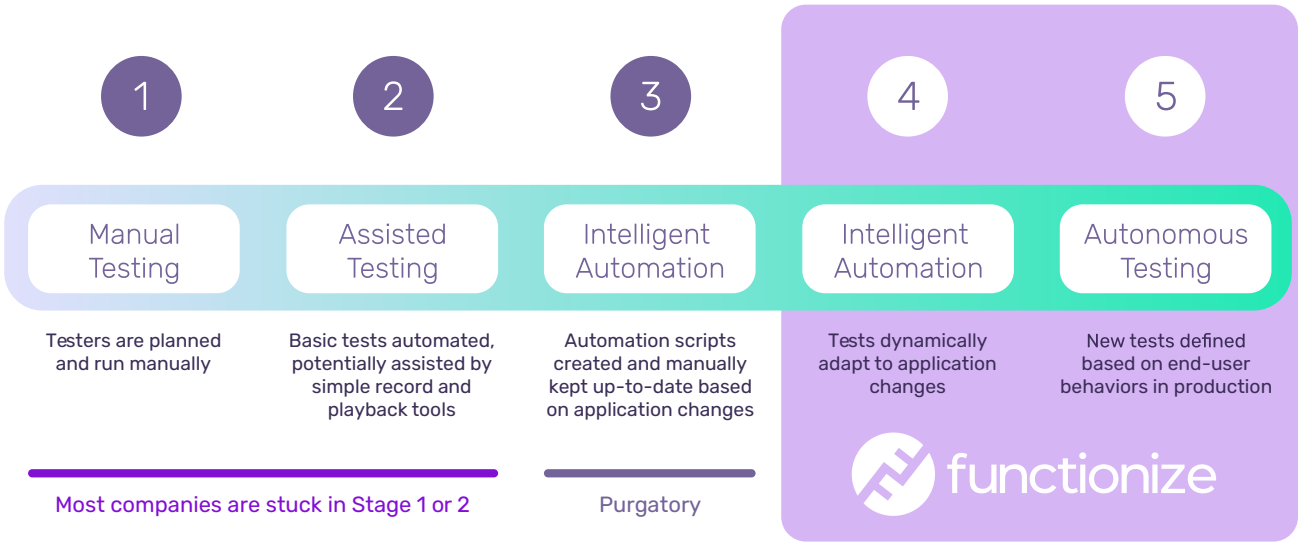
In 2020, Gartner introduced their model for assessing test automation. Like the SAE model, it consists of five levels (or six if you include manual testing). These levels are:

0 Manual testing
1 Assisted testing
2 Partially automated testing
3 Integrated automated testing
4 Intelligent automated testing
5 Autonomous testing

As with the SAE taxonomy, each level of automation gets closer to full autonomy.

## The Functionize model for test automation maturity

The aim of creating a new model for automation is to allow you to more accurately understand the level of automation that your testing tools are providing. Unlike the Gartner model, the Functionize model reflects the fact that there are multiple areas in which automation is applied to testing. These are creation, execution & analysis, and maintenance. As a result, our model has an additional dimension.




Our model is designed to make it easier to argue rationally about different solutions that are on the market. We show the detailed model on the next page.

| Automation Level | Test Creation (C) | Test Execution & Analysis (E) | Test Maintenance (M) |
|---|---|---|---|
| **Manual testing** | A human is responsible for creating the detailed test plan, setting out the test steps in detail. They may be assisted by a test management system. | The human completes each test completely manually. They work through the test plan step by step. Results are recorded in their test management system. | Whenever the application or code changes, a human tester updates all the test plans accordingly. They also review old tests to establish which ones to remove. |
| **Assisted testing** | A human is able to record a test using point and click. This is then converted into a test script. This only allows for simple interactions with the system under test. | The system is able to playback the test and provide simple pass/ fail results. A human has to manually collate the results and add them to the test management system. | Any update to the code or UI requires a human to re-record the test. All test failures need to be analysed to check for false positives. |

| | | | |
|---|---|---|---|
| **Partial automation** | A human creates a test script that can perform advanced actions, such as data-driven testing. The script can be modified to run on any browser or device. | Tests can be triggered automatically from CI/CD or the test management system. The results are directly transferred to the test management system. | The system can identify potential false positive failures where a selector changes or is unavailable. A human has to manually update the test script. |
| **Intelligent automation** | A human can create tests in different ways: NLP, intelligent recorders or hybrid approaches. All the tests are then used to train a machine learning model of the system under test. | The test can be run automatically from CI/CD or a test management system. Results can be analyzed visually and the whole UI is tested each time. Tests are run from individual VMs to ensure realism. | All simple test maintenance is handled by the system thanks to the ML model. This allows it to identify the correct selector even if the UI or underlying site code is updated. |
| **Autonomous testing** | The computer creates new tests based on observing user interactions with the system. Tests can also be bootstrapped from user journey descriptions or test templates. | The computer is responsible for the complete test cycle from detecting code changes to canary testing the feature in production. A human is presented with detailed results. | The computer performs all test maintenance tasks automatically. It identifies all issues and corrects them, testing potential solutions and showing these to the user. All tests are then updated. |

# Applying the Functionize model in practice

The power of this model is it allows you to make very detailed assessments of test automation tools and frameworks. In our experience, all tools achieve different levels of automation for different aspects. So, let's see how we apply the model in practice.

## Manual testing

Clearly, manual testing is always at level 1 in this model. However, manual testing remains an essential part of any test regime. That's because it is uniquely suitable for so-called progression testing. That is, testing of new features or the exploratory testing needed to establish the steps to recreate a known bug.

## Selenium

Selenium exhibits different levels of automation for different aspects. For instance, most Selenium-based test frameworks only achieve **M1** for maintenance. On the other hand, a skilled test engineer can use it to reach level **C3** for test creation. Test execution for Selenium is generally **E2**. However, certain test management tools and integrations allow it

to reach **E3**. Typically, it will average between level 2 and level 3. We like to describe this as purgatory, because you get the worst of both worlds: a complex system that is both hard to use and time consuming to maintain.

## Functionize

Functionize Architect is an example of a level **C4** test recorder. Architect allows you to record tests simply by interacting with the system under test. However, it also offers advanced features, like random test data, test variables, and data-driven testing. Test execution and analysis is at level **E4**, with tests able to run automatically and with full visual testing of the UI. The Functionize platform lies somewhere between **M4** and **M5** for test maintenance. Self-healing tests are an example of a level **M4** feature. The underlying ML model is able to seamlessly adapt to most UI and site code changes. However, sometimes the logic changes. That triggers a SmartFix—the system identifies the problem and offers suggestions for the best fix. This feature exceeds **M4**, but isn't quite true autonomous testing.

## Future systems

In the near future, we are expecting to see the first systems that truly reach level 5 on at least one of these measures. For instance, we are close to seeing systems that are able to analyse real-life user interactions and use these to create tests without any manual intervention. That would be the first **C5** system we have seen. For test execution, we will see systems that trigger end-to-end testing as soon as a new feature is pushed. This would include testing the feature in production to ensure it is stable. Finally, we are on the brink of seeing maintenance-free test automation. These systems will be similar to our Smart Fix approach, but they will use more detailed models to allow them to test each possible fix and reliably select the correct one. Where no fix seems to work, for instance, if a test user's credentials were updated, the system will still need to alert a human though.

# Conclusions

Test automation is a widely used term that is at risk of being abused. In particular, we see many systems claiming to offer some level of intelligent test automation. As a result, it can be very hard to compare test automation systems in a fair manner. Here, we presented a new model for assessing test automation. What sets our model apart is that it looks separately at each of the main test functions, creation, execution & analysis, and maintenance.

We are confident that level 5 autonomous testing is just around the corner. At that time, your whole team will become empowered to drive your test automation. This will result in more robust software, faster releases, and better customer satisfaction. However, there may be roadblocks in the way. Some of these are technical, but others are human, created by existing testers who are afraid of what the future holds for them. But if you confront these issues head-on you can look forward to a world where test automation becomes a key driver for your company's success.

functionize

# Test faster.
# Test with Functionize.


## Applications change over time. So should your tests.

Functionize provides a suite of intelligent test automation tools powered by machine learning. Our tools allow anyone to build automation, test smarter, and release faster. Traditional automated tests are brittle, constantly break, and require manual upkeep. Empower your teams to build AI-Powered tests that don't break and reach peace of mind. You focus on delivering great software, and let machine learning supercharge your tests.

## Functionize Recognized as a Gartner Cool Vendor

### Leading the industry in innovation

Gartner's Cool Vendor research recognizes innovative vendors in agile and devops products. Read the report to see what differentiates Functionize from others in the agile and devops market.

Gartner
COOL VENDOR 2020

## How it works

**Functionize tests run differently.** Unlike traditional test automation, our system is constantly fine-tuning a detailed machine learning model of your application based on your tests. We capture every piece of detail for every test step, including screenshots, DOM data, CSS, and even timing data. These millions of data points are collected and analyzed every time your test runs.

**Why does this matter?** Your test definition is an ever-expanding dataset from every execution. This allows the system to understand the intent behind your tests and adapt intelligently to change. When you update your application, your tests self heal, saving your team precious time to create more automated tests!

## Machine Learning You Can Trust

Our expert team of data scientists analyzed models from hundreds of terabytes of data from millions of tests, maturing our machine learning models over the past 6 years. We combined numerous machine learning techniques with traditional data science practices, resulting in 99.9% model accuracy, the highest in the industry! Machine learning techniques embedded in Functionize include:

- Supervised and unsupervised learning
- Computer vision - image recognition
- Natural language processing
- Template recognition

# End-to-end Capabilities to Accelerate Testing

## Support for complex flows

Build tests for even the most complicated end-to-end scenarios:

- Create orchestrations for sequential test suites or large-scale data-driven tests
- Generate random test data (to validate forms)
- Verify Email, Text Message, Database, APIs, and File Testing (PDF, HTML, CSV, XLS)

## Highly scalable

Eliminate testing infrastructure costs. All tests run in the Functionize Test Cloud, which means you can launch:

- Up to tens of thousands in parallel
- Across browsers, including mobile
- From different geographies for localized tests

## Smart creation

We give you 3 ways to create AI-based tests in minutes:

1. **Architect** is our super-smart test recorder: anyone can create tests rapidly, using an intuitive interface, easy drag and drop editing, and the ability to extend tests with code
2. **Natural Language Processing:** create tests using plain-text English, making migrations super easy!
3. **Autonomous Test Cases:** generate tests from how real users interact with your system

## Low maintenance

Save time wasted by test maintenance. Our platform autonomously self-heals your tests or recommends 1-click SmartFixes, so you can trust your tests and more easily find real bugs in your application.

Save time editing tests instead of a full rerun with:

- **Smart Screenshots:** select elements or add steps directly from screenshots
- **Live Debug:** edit tests directly in a clean container

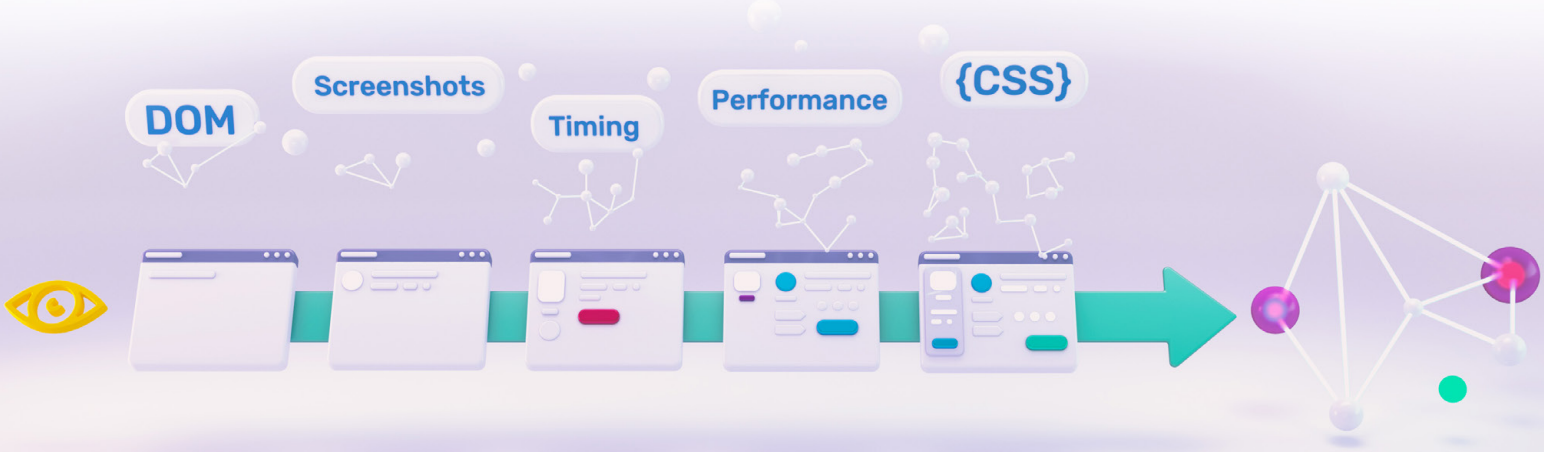
## Migrations

Need help converting your tests? Leverage our network of global service providers to help you migrate your existing tests from Selenium into Functionize.

## Rich analysis

Our visual testing analysis verifies full pages and elements to highlight what changed between runs. Visual completion time metric shows when a user can actually interact with your application.

# AI-assisted tests are dynamic and adapt with every execution



# Challenges with Test Automation

At Functionize, we speak to customers every day who struggle with test automation. They thought that Selenium would be the solution. Instead, it became the problem. Legacy tools and even other next gen solutions still fall short. Here are just some of the limitations:

| | | Functionize<br>AI-Powered Test Automation | Selenium | Legacy Tools* | Other Next Gen* |
|---|---|---|---|---|---|
| Platform | Web / Saas solution | ✓ | ✗ *desktop* | ✗ *desktop* | ✓ |
| | Multi-Deployment Options (Cloud, On Prem) | ✓ | ✗ *OnPrem only* | ✗ *OnPrem only* | ✗ *Cloud only* |
| Test Creation | No Code / Low Code | ✓ | ✗ | ✓ | ✓ |
| | Natural Language Tests (NLP) | ✓ | ✗ | ✗ | ✗ |
| | Autonomous Usage Based Creation | ✓ | ✗ | ✗ | ✗ |
| | Custom Reusable Code | ✓ | ✗ | ✗ | ✗ |
| Intelligence | Big Data - Non Scripted Tests | ✓ | ✗ | ✗ | ✗ |
| | Automate manual tests written in plain-text English using natural language processing | ✓ | ✗ | ✗ | ✗ |
| | Dynamic self-healing tests | ✓ | ✗ | ✗ | ✓ |
| | A/B Test Learning | ✓ | ✗ | ✗ | ✗ |
| | Trend Analysis - Anomaly Detection | ✓ | ✗ | ✗ | ✗ |
| | Integrated Visual Testing | ✓ | ✗ | ✗ | ✗ |
| | SmartFix Update Suggestions | ✓ | ✗ | ✗ | ✗ |
| | Screenshot Selection Updates | ✓ | ✗ | ✗ | ✗ |
| | Lazy Loading Support | ✓ | ✗ | ✗ | ✗ |
| Advanced Execution | Mobile Web | ✓ | ✗ | ✗ | ✓ |
| | Cloud Execution | ✓ | ✗ | ✗ | ✓ |
| | Live Debugging on the VM | ✓ | ✗ | ✗ | ✗ |
| | Built-in Timing Analysis | ✓ | ✗ | ✗ | ✗ |
| Extensible Testing | Email, Text Message, Database, API Testing, File Testing | ✓ | ✗ | ✗ | ✗ |
| | Native test data generation | ✓ | ✗ | ✗ | ✗ |
| Support | Industry Leading Global Customer Support | ✓ | ✗ *Community support* | ✓ | ✓ |

* Criteria: if more than 1 competitor in the category has the capability, the category will be marked as ✓

# Results

**85%**
**Time Saved**
From fixing broken tests

**10x**
**Faster**
Create at least 10 Functionize tests in the time to create 1 Selenium test

**5x**
**Coverage**
Increase % automated from 10% to 50%

**80%**
**Cost Reduction**
Save infrastructure costs with Functionize Test Cloud

## Trusted by Industry Leading Enterprises Worldwide

salesforce, Hewlett Packard Enterprise, mastercard, wipro, cisco, FARMERS INSURANCE, Google, The General Insurance

> Functionize offers an easy way for us to test a highly dynamic and complex production site, ensuring that not only the functionality, but the look and feel for our site is world class.

> I am so happy my company chose to work with Functionize. I quickly learned a choice to use Functionize is a choice for success.

> Functionize has a fast learning curve and therefore anyone, right from automation experts to business users can utilise this tool to write effective test cases, thus speeding up the testing cycle of an application.

## Seamlessly integrate with your existing tools

circleci, HEROKU, Spinnaker, Bamboo, Jira Software

functionize

**Welcome** to the
**Future of Testing**



functionize.com
1-800-826-5051