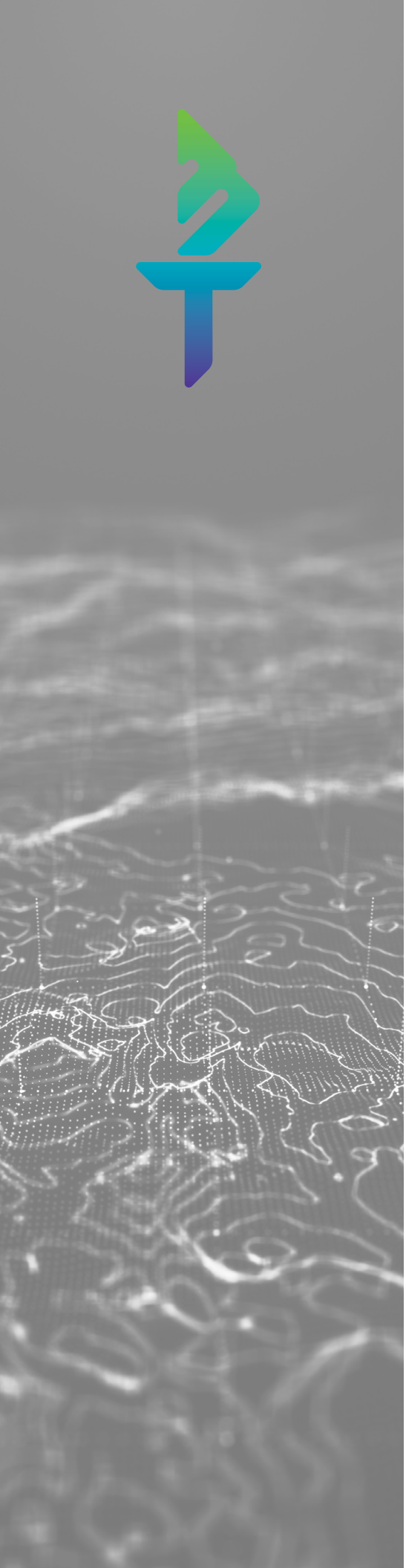
# White Paper

# RPA Platforms

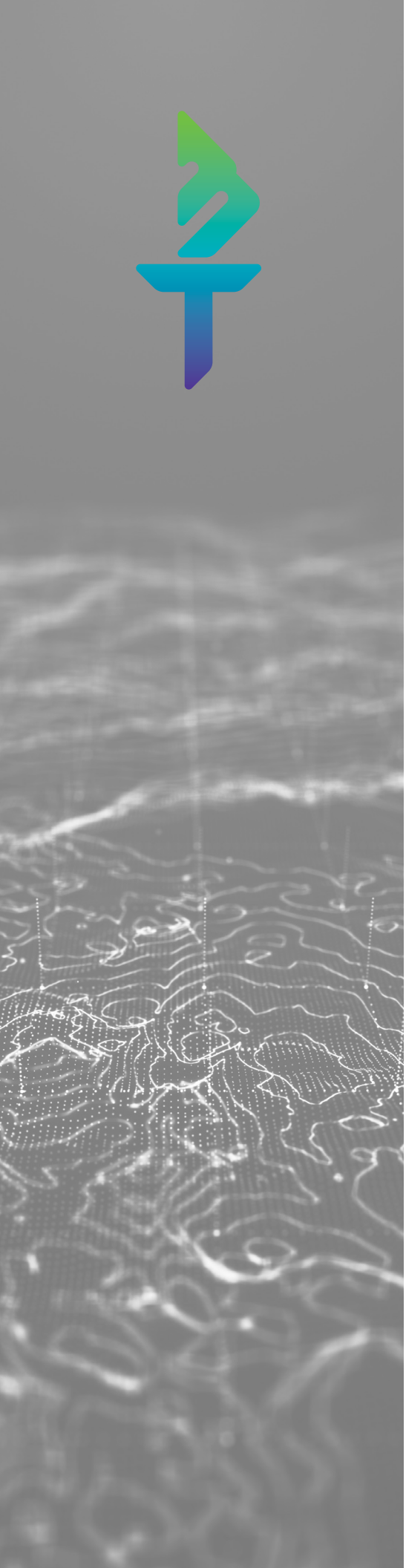
## Automate repetitive tasks and save time with the right RPA tool

By Victor Le

MACEDON TECHNOLOGIES



# Table of Contents



## Introduction

Robotic process automation - or RPA - yields time and cost savings by having a robot perform steps quickly, consistently, and accurately. Many industries exhibit several pain points that can be alleviated through RPA. For example, in healthcare, scarcity of electronic medical record system APIs necessitates a lot of manual data entry. Report generation involves gathering data from multiple systems. Patient and provider lookups require particular search criteria and careful matching against multiple results.

Macedon has built automations using UiPath - a leading automation platform - to address these areas. Now that Appian has grown in the RPA space, it is important to weigh these technology options. Some workflows can be performed without any human input. For others, humans drive some of the process, with the structured and predictable portions being automated. Appian and UiPath are capable of solving both scenarios to varying degrees.

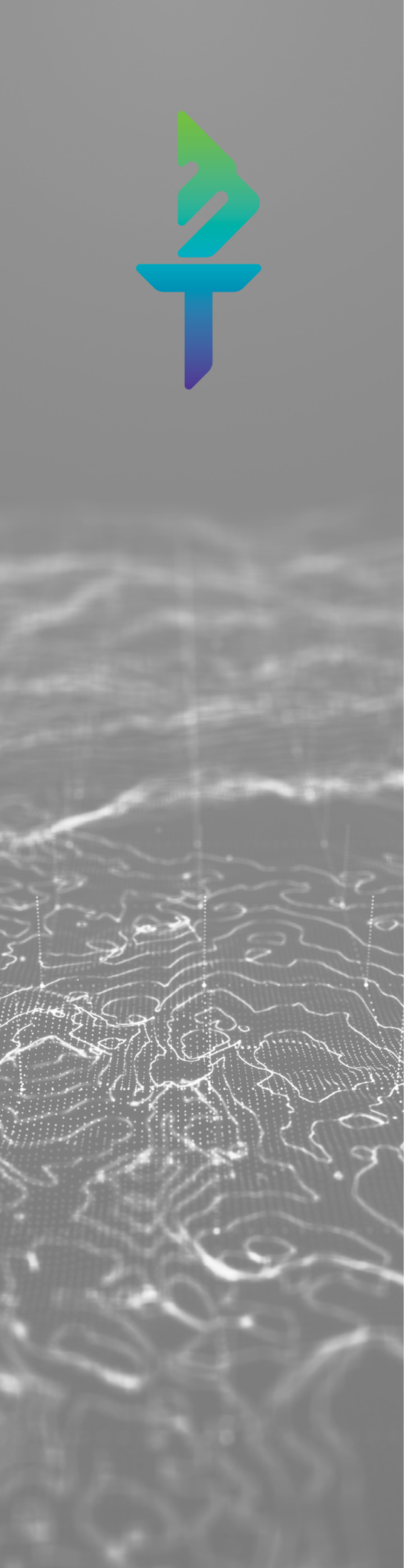
Appian is a powerful BPM platform capable of automating certain functionality even without its dedicated RPA feature. With RPA, Appian can automate workflow steps in external systems that lack APIs. UiPath has vast automation capabilities and also facilitates user-driven processes, most prominently via applications built in App Studio.

With both platforms able to achieve similar goals, we will examine which scenarios are better suited for each. This paper is based on Appian 22.4 and UiPath 22.10.

## Fully Automated Processes

For completely hands-off automations, UiPath is the platform of choice. There is a plethora of out-of-the-box components for common actions, greatly accelerating development. This includes both automation of user interactions within applications and background activities like file system operations, Excel sheet modification, sending emails, script execution, etc. As will be explored later in this paper, UiPath's UI and image-based automation is quick and straightforward to configure compared to Appian's implementation. Moreover, workflows can be built in one editor for a unified developer experience.

It is possible to build a fully automated process in Appian RPA as a standalone robotic process. For example, a robot might log into a system, pick up work items from a queue, and complete those items, repeating said actions throughout the day. Since all the steps are just interactions with various controls, the workflow can be designed using available low-code actions in the robotic process editor. More complex functionality, not feasible via the low-code actions, will require developers to write custom Java methods in a separate IDE to then deploy and import into Appian for use.



With more involved automations, developers may need to additionally leverage tools from the base Appian platform. The Process Modeler is a workflow designer within the base Appian platform that ties together business rules, integrations, database operations, and user interfaces. It can even execute robotic processes. When used within the Process Modeler, Appian RPA primarily automates steps involving repetitive, predictable, and well-defined user interactions. Although the Process Modeler is a versatile tool, the development process can involve a lot of context switching. Business rules, integrations, and robotic processes are designed in separate editors within the platform. Troubleshooting robotic processes is especially cumbersome since the step-by-step execution of a robotic process cannot be viewed from the Process Modeler; instead, one would have to go into the RPA Console and view the logs for the process of interest.

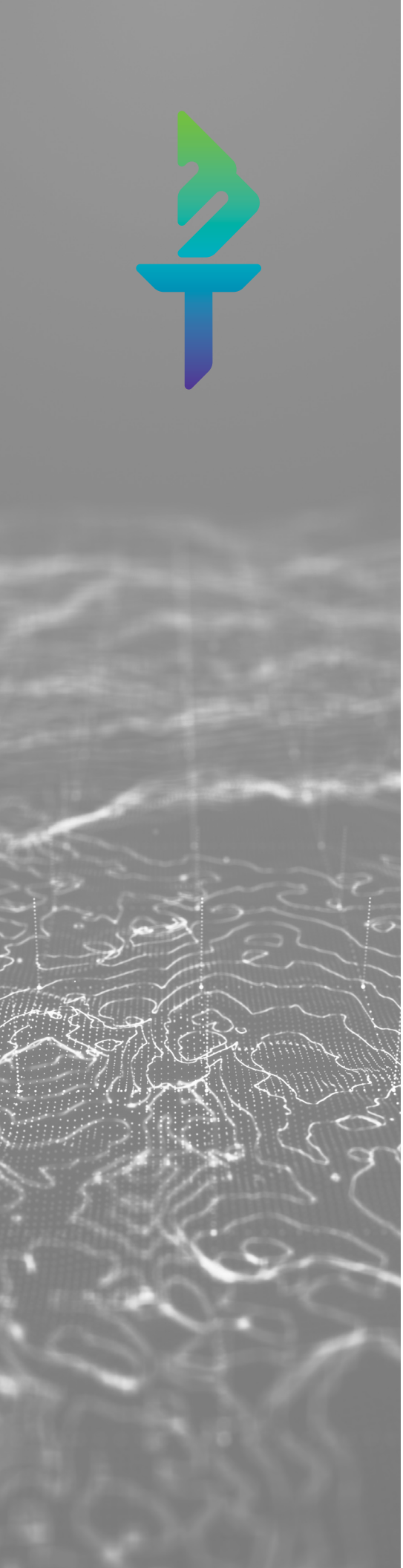
Designing full automations involving UI interactions is often a quicker, more streamlined, and unified experience in UiPath. Development can take place entirely within a single editor. In many cases, Appian RPA will be paired with the base Appian platform, which entails designing in at least two editors – more if custom Java methods and business rules are involved.

## User-Driven Processes

Some workflows are inherently user-driven, in which case Appian is the platform of choice. Appian is well suited to record-centric processing, and can tailor user experiences based on roles and assigning tasks to different groups or individuals. Users work with modern, dynamic interfaces that can pull data context from disparate systems. The users' portions of workflows are part of larger process models in which other automation steps (data manipulation, database operations, integrations, etc.) take place. Here, RPA would most often extract data from an external system's front end and/or complete predictable workflow steps within said system. For example, a recruiter might begin the onboarding process for a new employee. At one of the steps in the process, an RPA bot will use info entered by the new hire to set up direct deposit in a legacy payroll system. Appian's high degree of modularity translates to greater confidence when building complex applications, and its reporting capabilities offer visually appealing business insights.

UiPath App Studio is a cloud development platform that enables developers to create user-driven applications that can easily start processes built in the main platform. It features standard input and data display options and components that can be conditionally hidden or disabled. Styling options allow color, font, size, and other aspects to be modified. Most importantly, interacting with components can initiate various events. This includes message display, value updates, and starting UiPath RPA processes, whose execution results can be further used within the app. Persistent storage takes place via saving data to a UiPath cloud data store, enabling such functionality as a simple display of transaction history.

App Studio has significant shortcomings when compared to Appian. For one, the developer experience is much more drag-and-drop oriented. With no



means to view the underlying code for components, it is more cumbersome for developers to make precise edits or quickly ascertain what an app does from a glance. Although components can start processes, actually viewing the process requires opening that project in the separate UiPath Studio editor. While there is a persistent storage option, data display is relegated to lists and tables. There's no out-of-the-box integration with an enterprise database solution for more valuable reporting via views. Charts and graphs need to be generated through Excel in an RPA process and then loaded into the UiPath app. A large barrier to building complex apps is that App Studio does not emphasize modularity and maintainability. There's no concept of constants or reusable custom components. This increases the effort of building similar applications and making changes like value or label updates. Lastly, not having a concept of tasks or traceable workflows makes it difficult to create collaborative systems for business teams. As a result, apps are more suited for isolated users.

As it currently stands, UiPath's App Studio is only a viable option for simplistic processes and siloed work. A proper BPM solution ties together different steps in a larger workflow and helps to provide organizational insights and transparency. In this sense, Appian's user-driven workflows are valuable for both the individual and the team.

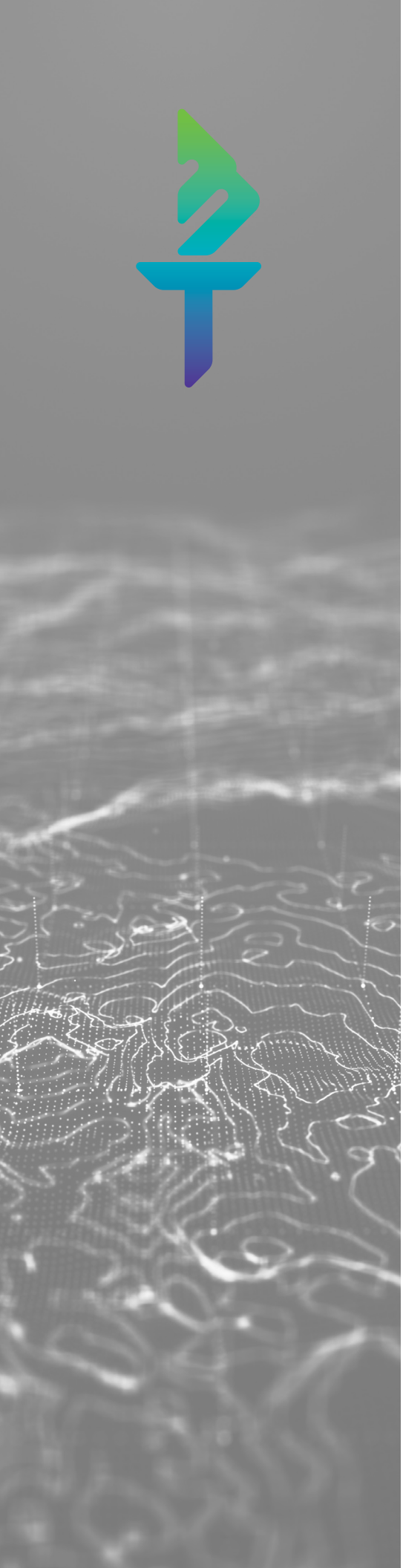
## Implementation Analysis

Below are high-level and in-depth comparisons of how implementing various pieces of RPA functionality differs between Appian and UiPath. As is apparent, there are areas where Appian lags behind UiPath for the time being.

### Web App Automation

Configuring actions by which a robot will interact with user interfaces.

| Appian | UiPath |
|---|---|
| **Effort: Low**<br>• Recorder tool auto-generates actions<br>• Ad hoc action configuration involves some tedium or manual work | **Effort: Very Low**<br>• Recorder tool auto-generates actions<br>• Simple ad hoc action configuration<br>• UI Explorer tool for fine-tuning |
| Appian's recorder tool is the most accurate way to configure web app actions. It identifies portions of the web page's source code corresponding to elements of interest. Outside of using the recorder, a developer would have to use browser dev tools to inspect the source code and extract the pertinent piece. | UiPath's recorder tool auto-generates actions, identifying portions of the web page's source code corresponding to elements of interest. It's also simple for developers to add actions manually. The actions enable the dev to highlight elements and obtain the source code portion ad hoc. A dedicated UI Explorer tool similarly allows for highlighting elements but also extracts much more of the source code for fine-tuning. |



## Image-Based Automation

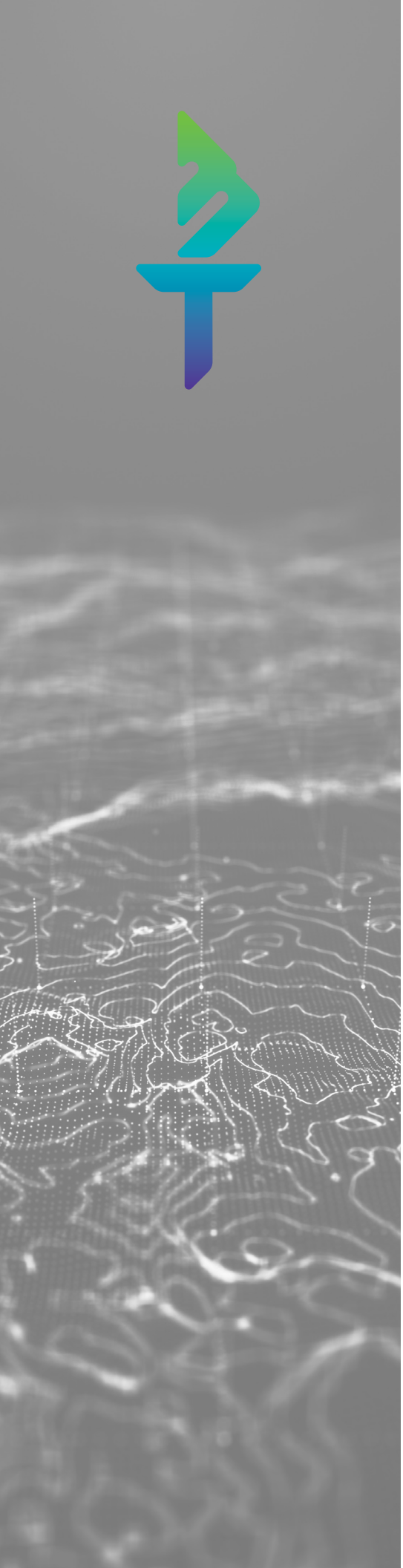
Configuring actions by which a robot will interact with on-screen images.

| Appian | UiPath |
|---|---|
| **Effort: Medium**<br>• Separately launched support image tool to create images of interest<br>• Image-based action must load an image of interest. Any offsets are configured manually | **Effort: Very Low**<br>• Image-based action has innate screenshot tool to capture the image of interest<br>• Recorder tool can auto-configure an offset from an image |
| The support image tool must be launched through an Appian RPA application installed on the developer's machine. Screenshots of the entire screen are cropped to just the image of interest, with the resulting image saved to a specific location accessible to the robot. Within the process editor, this image must be located and loaded into the image-based action. Configuring offsets from the image involves manual entry of pixel counts. | Image-based actions enable developers to create a screenshot of just the image of interest. The image is saved to the project and is immediately available for use. When working with offsets from images, the recorder tool removes any guesswork and generates accurate actions. |

## Workload Distribution

Configuring multiple robots to work over a common queue of tasks to increase throughput.

| Appian | UiPath |
|---|---|
| **Effort: High**<br>• Write Java methods to specify a queue, reserve queue items, release queue items, and update the queue | **Effort: Very Low**<br>• Out-of-the-box actions for picking up queue items and updating their status<br>• Reusable queue |
| Developers must write custom Java methods that in turn call methods from various queue interfaces. Queue item processing involves reserving the item to ensure concurrent updates to that item are not made and then releasing the item when done, at which point the item's status is updated. For cleanliness, a new queue is created and closed per batch of items, similarly using custom Java methods for a bot to reserve the queue and close it once there are no more items to process. | Low-code actions for interacting with queues and queue items simplify development. Innate locking mechanisms with queues and out-of-the-box actions save developers the effort of having to ensure the same queue item cannot be modified by multiple process executions. The same queue can be reused for subsequent runs, as items can be filtered on status. |



## Appian RPA Advancements

In the span of about 2 years since its RPA rollout, Appian has made significant strides in key areas to increase the appeal of this platform offering. Below we examine some of these advancements.

### RPA Capability

- Image recognition
- Credential support
- File system operations
- Looping workflows

The ability to have a robot perform actions on an image added much-needed versatility to automations, while credential support improved security and maintainability. File system operations and for-each loops expanded what was possible to configure with just low-code actions.

### Base Appian Platform Synergy

- Parameter support
- Process invocation
- Reusable components support

Robotic processes were enhanced to accept and return parameters to the calling process, enabling them to be more dynamic. An option was also added to execute a Process Model from within a robotic process instead of exclusively the reverse – useful for when a small portion of the workflow needs to take place within the base platform before proceeding with the remainder of a transaction. Support for reusable business logic meant developers could independently develop and test complex rules before plugging them into the process.

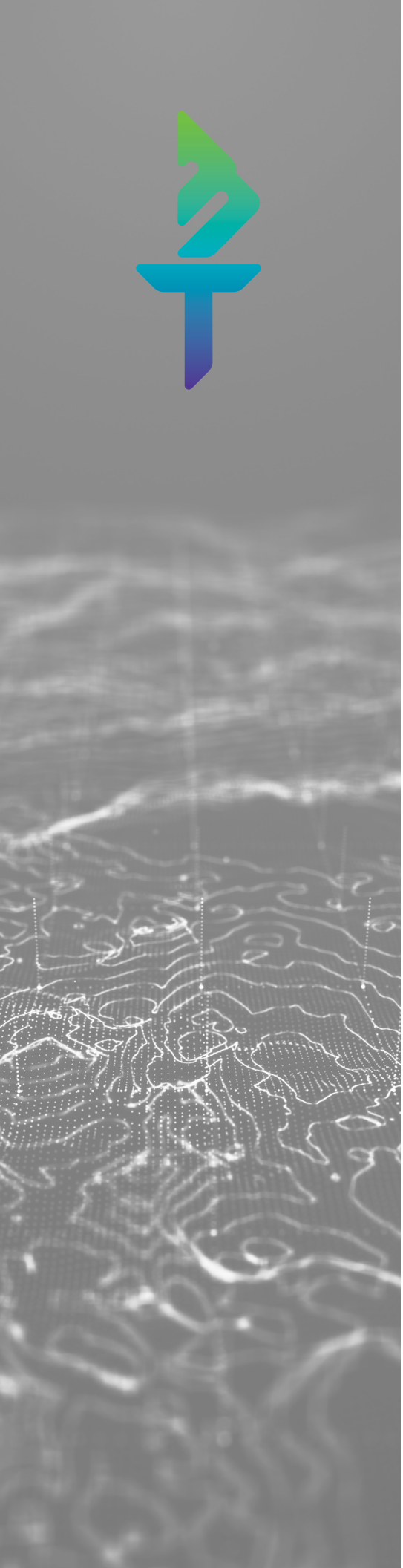
### Developer Experience

- Low-code actions
- Editor redesign
- Synchronous process execution
- Streamlined recording tool

Upon initial release, Appian RPA development entailed Java coding in an IDE. Methods written in this way were then imported into Appian. Low-code actions for common robot actions made development more streamlined and accessible. The major redesign of the editor made it easier to ascertain what a process does at a glance.
Making RPA workflows execute synchronously within the Process Modeler meant developers no longer had to build in polling mechanisms to advance the process. Lastly, the current iteration of the task recording tool automatically imports generated actions into the workflow, whereas previously a recording file had to be created and manually imported.

## Conclusion

Selecting UiPath or Appian as the RPA platform of choice would depend on the types of automations an organization can or plans to build. If the workflows to be automated have no user involvement, UiPath is an excellent choice. Easily configured UI and image-based automation actions, combined with a seamless developer experience, help to provide rapid delivery.



Since the prerequisite for Appian RPA is the base Appian platform, an organization considering RPA, in general, might contemplate opportunities for a BPM solution to tie together and optimize its processes. If such opportunities exist or may come along in the near future, Appian is the better investment. For the portions requiring user interaction, Appian will offer more options to build rich and well-contextualized interfaces. Apart from this, the platform's record-centric applications help drive its powerful reporting capabilities. Support for modularity is also a major point in Appian's favor. Reusable components help accelerate development of similar features and applications, on top of aiding in testing and debugging. This practice must be introduced to UiPath Apps to facilitate building complex applications. Although more effort is required to implement some functionality in Appian RPA compared to UiPath, given the platform's track record in just the last few years, it's safe to say Appian will continue to build upon its brand of RPA to make it more appealing to businesses and developers.

## About the Author

Victor Le is a Principal Consultant with Macedon. He initially focused on Appian development, building several financial applications in areas such as consumer lending, KYC, and PPP. Later he specialized in UiPath, delivering and supporting many healthcare automations utilized in telemedicine, patient admissions, and staffing services.

## About Macedon

Macedon is a recognized leader in intelligent automation and cloud data solutions. We have deep expertise with industry-leading technologies that we leverage to solve our clients' unique challenges. Our hybrid roles achieve better solutions faster than traditional development teams.

**Contact: (571) 526-4281**
**info@macedontechnologies.com**