Best Practices for Robotic Process Automation Success

Published: 18 June 2019 **ID:** G00385723

Analyst(s): Gregory Murray

RPA extends automation into use cases that usually require humans. RPA simplifies creation of automated workflows, but doesn't completely democratize an inherently technical system. Infrastructure and operations technical professionals must understand RPA's capabilities and limitations.

Key Findings

- RPA is a discrete set of automation tools and technologies that extends and complements existing automation disciplines.
- While the low- or no-code environment creates opportunities for nontechnical people to integrate through RPA, the entirety of operations, validation, troubleshooting and maintenance will almost certainly require technical professional skill sets.
- Success with RPA depends more on aligning developer persona with the right processes, rather than the right vendor or technology.
- RPA recorders are useful in rudimentary scenarios, some process discovery and mapping, but robust automations will almost always require manipulating the automation workflow modules directly. Expectations may need to be reset.
- A governance framework for selecting and validating RPA use cases, often managed by a COE, is a critical part of any enterprise RPA rollout.

Recommendations

Implement these best practices if you're an infrastructure and operations technical professional managing automation as a component of your efforts to modernize operations:

- Establish the RPA developer personas, and align the form and level of governance to the entire RPA life cycle to those personas, because seasoned developers and citizen developers won't make the same kinds of mistakes.

- Establish a governing body to provide the guidance necessary to ensure that RPA is used only when it is the right tool and that RPA applications comply with all corporate and regulatory requirements.
- Determine correct application of RPA technology by establishing criteria for identifying and prioritizing the use cases you and your team are prepared to tackle.
- Manage process selection criteria as a roadmap, ensuring that the criteria evolve to reflect your level of maturity, skill and investment around RPA.
- Focus on the robot validation framework to ensure robust automation for both seasoned and citizen developers.

Table of Contents

Analysis....3
Document Overview....3
RPA Overview....4
RPA Within the Automation Landscape....6
RPA Platform Overview....7
Script Development Environment....8
Bot Deployment Options....13
Bot Management Platform....16
Best Practices for Adopting and Operating RPA....17
Establish the Developer Persona....18
Establish a Governing Body....18
Use Selection Criteria to Prioritize Well-Suited Cases for RPA....21
Manage Process Selection Criteria as a Roadmap....26
Apply Release Management to RPA Scripts....28
Test and Validate RPA Scripts Before Releasing to Production....28
Ongoing Operations for RPA Scripts....31
Automation and Transaction Throughput....32
Measuring Success With RPA....33
The Role of AI in RPA....34
AI Within RPA....34
AI External to RPA....35
Recommendations....36
Conclusion....36
Gartner Recommended Reading....36

List of Figures

Figure 1. RPA Life Cycle..... 4
Figure 2. Simple RPA Workflow..... 5
Figure 3. RPA in the Automation Landscape..... 6
Figure 4. RPA Platform Components..... 8
Figure 5. Example of Bot Modules..... 9
Figure 6. Changing Window Title Bar..... 11
Figure 7. RPA Versus RDA..... 15
Figure 8. Blue Prism RPA Management Dashboard..... 16
Figure 9. Selection Criteria for Well-Suited Processes..... 21
Figure 10. Layers of Integration..... 23
Figure 11. Manage Selection Criteria as a Roadmap..... 27
Figure 12. RPA Validation Framework..... 30
Figure 13. Layered Testing Approach for RPA Scripts..... 31
Figure 14. Effective RPA KPIs..... 33

Analysis

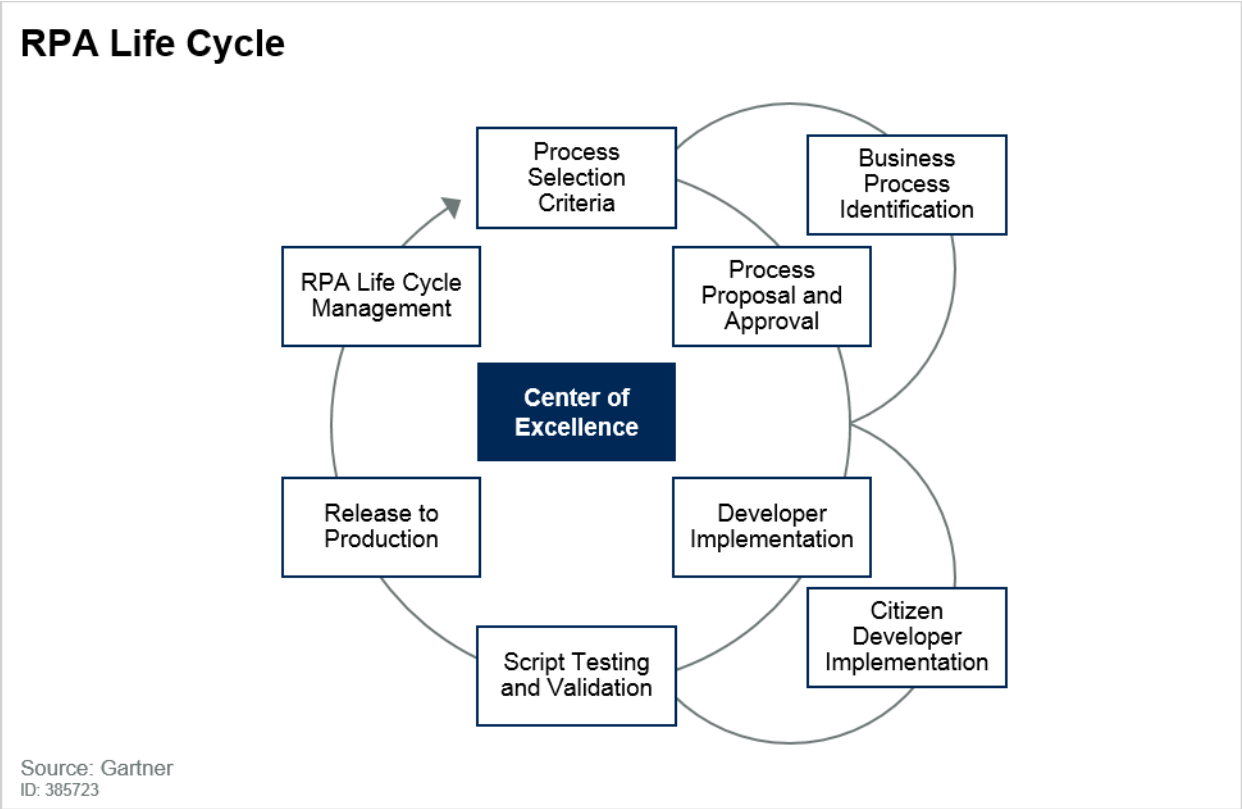
Document Overview

This document has two primary sections.

The first section describes robotic process automation (RPA), the platform and key considerations that will drive RPA development and operational best practices. This first section is helpful for establishing the vocabulary and aligning to the capabilities and architecture of most RPA tools. If you're new to RPA, this section will help dispel some myths and hype around the tool.

The second section of this document begins with the Best Practices for Adopting and Operating RPA section. Readers experienced or familiar with the components and development in an RPA environment may want to skip directly to this section to look more closely at managing the RPA life cycle, as shown in Figure 1.

Figure 1. RPA Life Cycle



RPA Overview

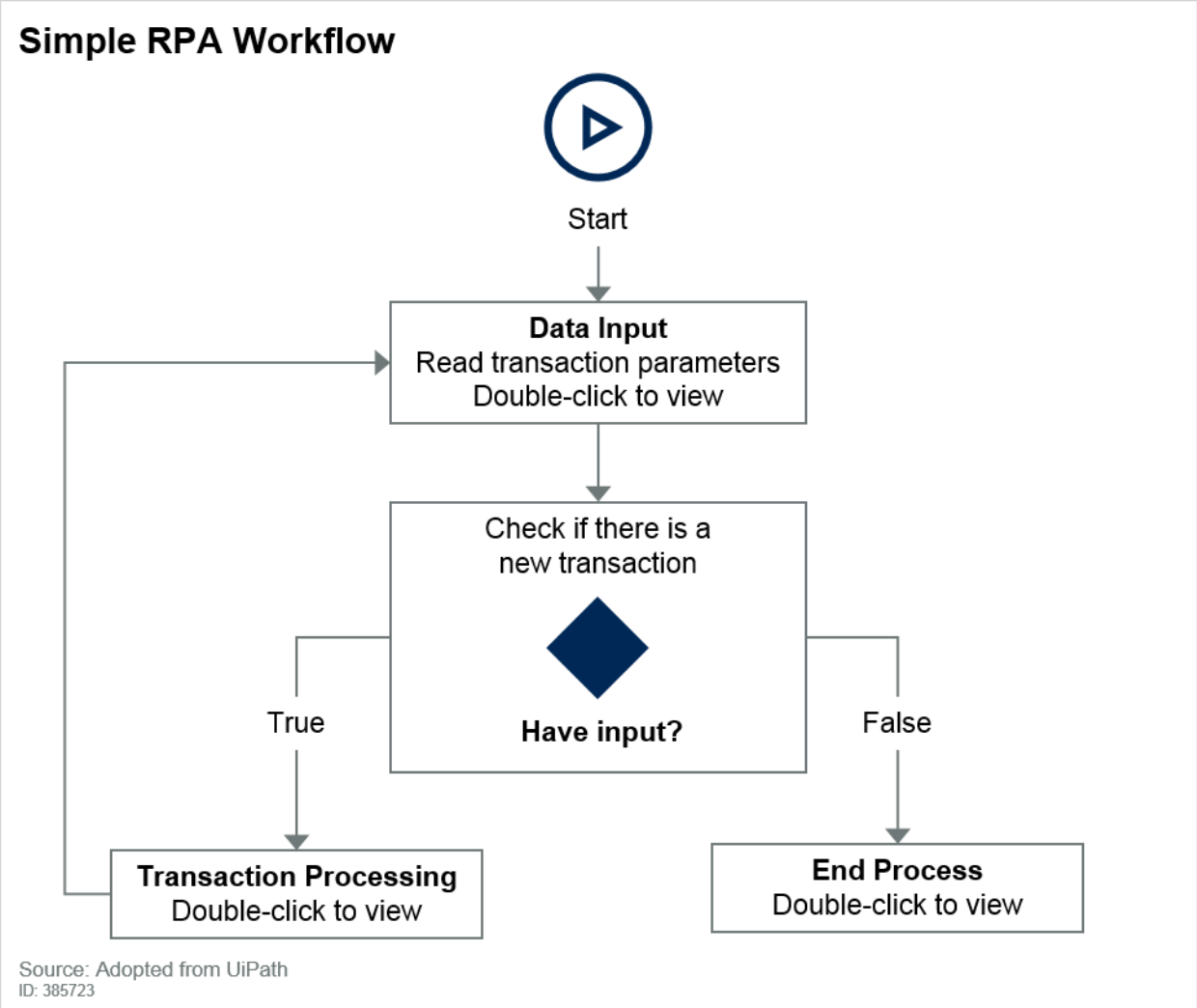
Robotic process automation should be one of many tools and techniques for technical professionals to deliver pervasive automation. RPA solves automation problems that cannot be solved purely with programmatic techniques and does not require modifications to existing processes. When systems or data repositories do not have or support programmatic access, users interact through local application GUIs, and automating repetitive tasks on these applications requires a technique that can automate a GUI. Moreover, this kind of automation will require a technique that can operate an interface that was designed for humans. This is where RPA shines.

RPA emulates human activity — keyboard and mouse input — in addition to purely programmatic operations. This allows RPA to automate systems and operations that have always required humans and, when done well, automate with greater efficiency, availability and accuracy. Because robots don't require breaks, don't get tired, don't get distracted and work around the clock, a robot can significantly increase the amount of processed data, which can be good or bad, depending on the quality of that data. Where data quality is good, there is potential for RPA to maintain that level of data quality. By eliminating transposition or mechanical errors, RPA helps ensure that data input by the robot doesn't introduce errors and preserves upstream data cleansing.

RPA script developers create automation scripts either through a recorder or through a simplified, low-code development interface. The developers will define triggers for these scripts that will invoke

the automations through a variety of system or data center events. An example interface is shown in Figure 2.

Figure 2. Simple RPA Workflow



Source: Adopted from UiPath
ID: 385723

There are limits to how well an RPA robot can replicate human behavior and deal with variance within a process or UI. Therefore, the keys to RPA success lie in:

- Focusing the use of RPA on well-suited processes and data
- Developing and adhering to mature processes for managing the robot life cycle
- Creating diligent alignment between the process owner and the robot owner
- Planning and preparing to manage the complete RPA platform and life cycle

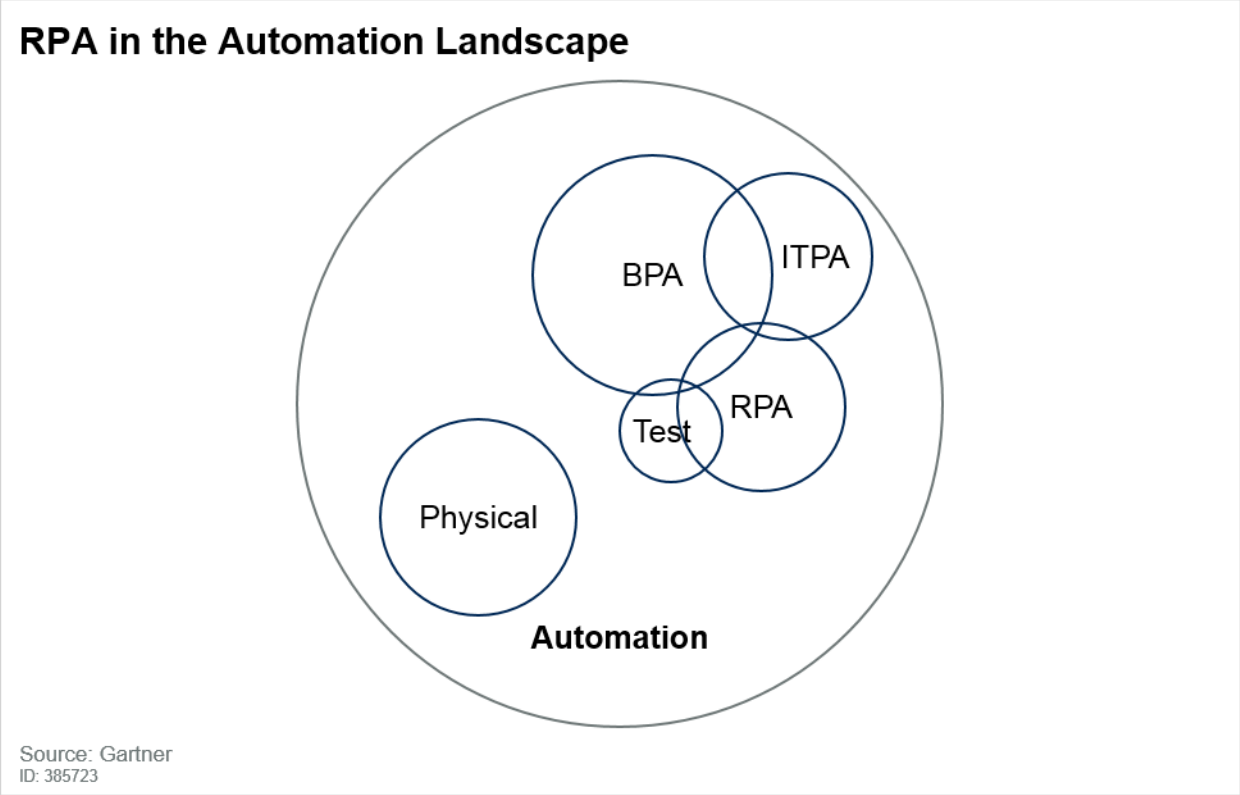
The architecture and technology of the system will reflect vendor selection and the nature of the process being automated.

RPA Within the Automation Landscape

Automation is a disciplined journey, not a casually reached destination. Each application of automation relies on specific tools and techniques to deliver efficient and effective automation. RPA is a unique tool in this continuum that extends the reach of automation beyond programmatic interfaces and allows processes to be automated that were formerly impossible to automate.

Since RPA is but one of many tools and technologies for automation, it is critical to understand and recognize when to use RPA and when to use a different automation tool. A common point of confusion comes from the term "robotic." There are no physical automation capabilities in RPA, like those that we normally associate with robots. As Figure 3 shows, RPA is wholly a software tool that complements, rather than replaces, other automation disciplines, like business process automation (BPA), IT process automation (ITPA) or test automation. There are even opportunities to combine these tools to automate a process at several different levels, creating even more efficiency than any single tool would provide.

Figure 3. RPA in the Automation Landscape



For an in-depth comparison of BPA tools, including RPA, see "Comparing Digital Process Automation Technologies Including RPA, BPM and Low-Code."

BPA seeks to improve efficiency, reliability and accountability by defining, tracking and automating key steps in a business process coordinated across a potentially distributed set of responsibilities. Tools and techniques for BPA are often included with business process management (BPM) packages that help organizations optimize and consolidate business processes like inventory management or requisition approvals. Compared with BPA, RPA tends to be more task-centric, executing a set of manual steps within a task, rather than managing a long-running process.

ITPA tools operate and automate IT services, infrastructure, resources and systems. ITPA automates the provisioning, configuration and monitoring of IT systems and services to improve efficiency and ensure availability. Because of the domain-specific integrations, RPA is rarely an effective replacement for ITPA tools. Also, ITPA tools are almost exclusively used by administrators in data center environments, where RPA can be used by business users to automate business tasks.

Even test automation that can apply similar technologies for operating on-screen elements, in addition to invoking APIs, is very distinct from RPA. The two solutions differ significantly in how the automation is managed and, most importantly, the reporting associated with test automation. Where RPA delivers value primarily through efficient automation of tasks, test automation delivers value in identifying and reporting on functional or architectural issues in a software product. The automation in test automation is a means to that end.

For that reason, RPA will persist as a stand-alone technology, even though some capabilities are found elsewhere. The operational context of these systems becomes overhead to task automation, and they may lack key functionality necessary for managing distributed task robots at scale.

Therefore, the introduction of RPA should be seen as an expansion, rather than replacement, of automation tools and techniques. By leveraging the full scope of these automation technologies, it is possible to extend the reach of your automation efforts. The key is to understand the capabilities and limitations of each automation discipline and apply each technology to the right scope and process.

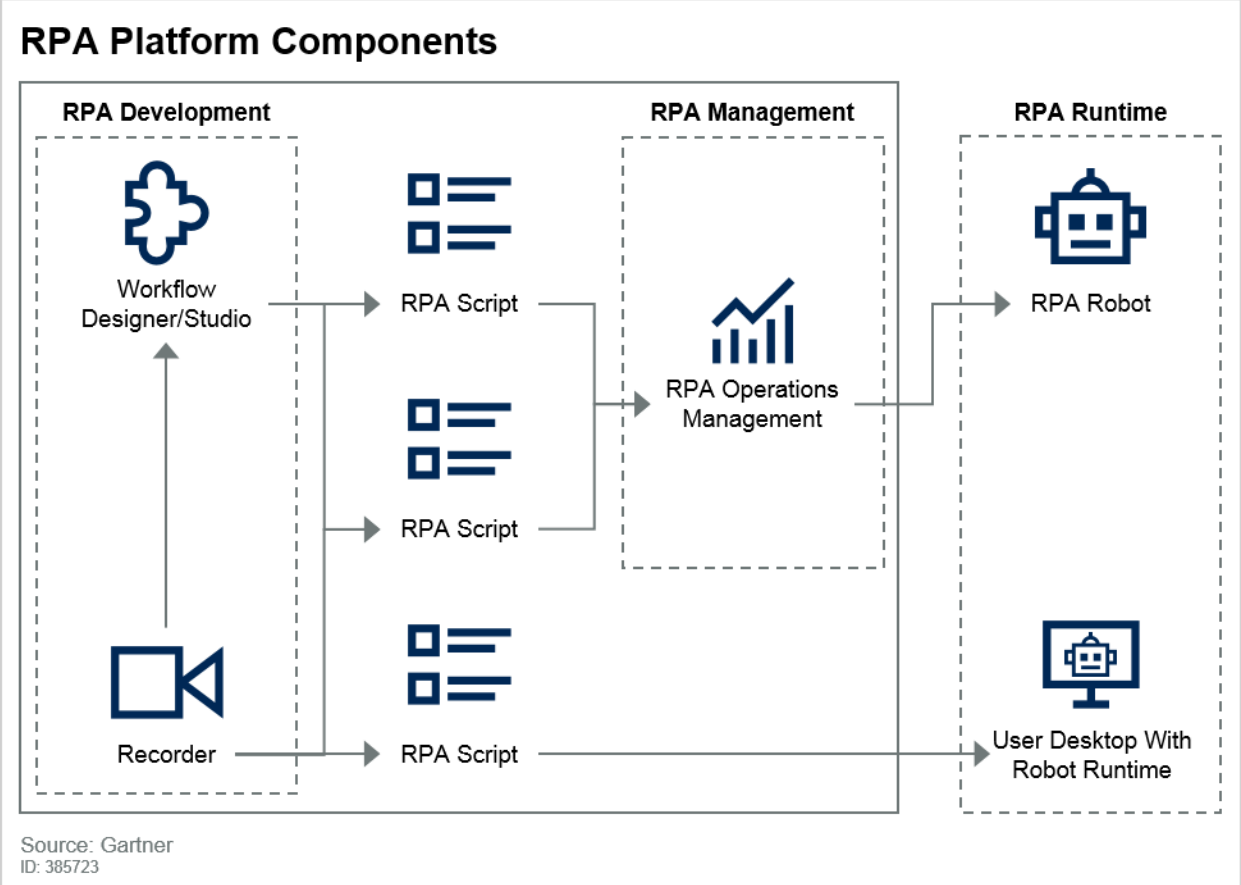
RPA Platform Overview

Before examining these limitations, however, it should be useful to look at the architecture of an RPA platform. While the implementation varies from vendor to vendor, Figure 4 shows the components common to most platforms and environments, which includes a:

- Script development environment to develop the steps and actions within an automation
- Robot execution environment that provides the runtime environment for the scripts
- Management platform to monitor and manage the deployment and operation of scripts and robot runtime environments

The following section will discuss each of these components in detail and highlight critical concepts, as well as advantages of the approaches available from RPA vendors.

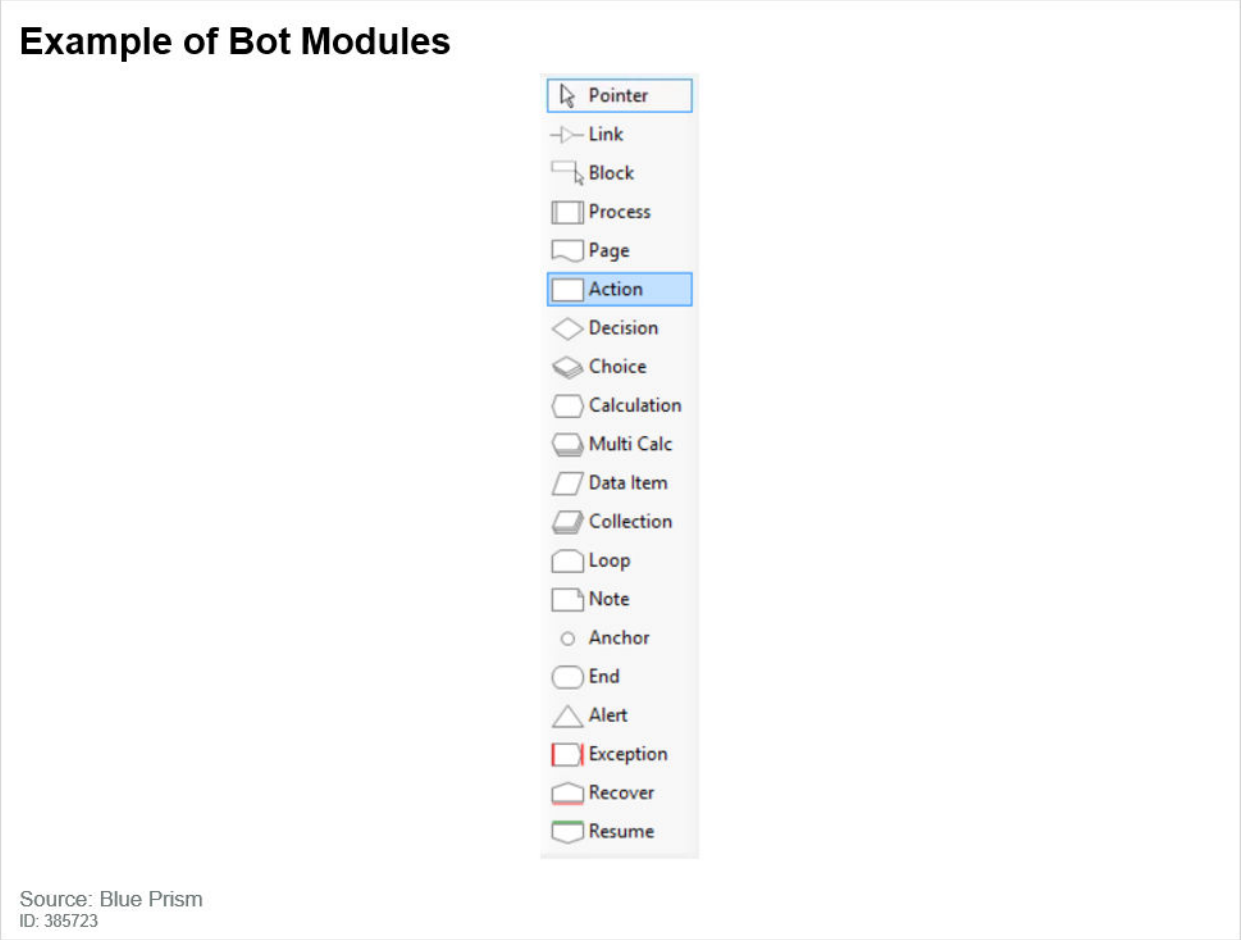
Figure 4. RPA Platform Components



Script Development Environment

The development environment — often called the "studio" — gives architects and developers a workspace to assemble workflows, declare variables and define triggers for the execution of an RPA script. Most RPA development environments are not purely code-based. That is, the development of a bot often involves assembling a set of execution modules in a drag-and-drop workflow designer (see Figure 5). Many modules have the ability to include regular expressions, embedded scripts or even code. However, the primary development for the automation script is in the graphical assembly of modules and connecting them by dragging the output of one module to the input of another.

Figure 5. Example of Bot Modules



The catalog of modules will depend on the vendor platform, but most RPA systems include modules that provide:

- Triggers, a wide range of stimuli from mouse clicks and keystrokes to file system changes, which can invoke a script or a condition within a script
- Application controls
- Mouse and keyboard operations
- Clipboard operations
- Data retrieval
- Data entry
- Conditions and loops

- Notifications (dialogue, sound, mail, etc.)
- File system operations

By chaining and building workflows with these modules, developers can create automation that can manipulate, respond to or extract data from just about any source that can be displayed on screen.

RPA Recorder

One of the most prominent and misunderstood elements of bot development is the concept of a recorder. The recorder allows for the creation of an automation sequence by having the automation platform record the actions that a human worker performs with on-screen elements. An RPA recorder is similar to macro recorders found in Microsoft VB and Microsoft Office. After pressing "record," a worker executes a task or series of tasks. When the recording is stopped, the actions and operations are converted into a set of machine instructions that allows for automated, programmatic repetitions of that sequence.

Windows-based applications and HTML-based forms that are running local to the bot provide a specific identifier for every object on the screen. Every button, every field and every character are uniquely and explicitly identified within either the HTML or the Windows manager.

For instance, the search box on gartner.com is identified in HTML as:

```
<input name="keywords" id="keywords" maxlength="400"
onkeypress="searchboxKeyPress(event,'gSearchForm','');" value=""
placeholder="Search" class="input-large ui-autocomplete-input ui-
autocomplete-loading" autocomplete="off" type="text">
```

Or the "File" menu bar option in Excel is identified within an RPA script as:

```
<wnd app='excel.exe' cls='XLMAIN' title='Book1 - Excel' />
<wnd cls='MsoWorkPane' title='Ribbon' />
<wnd aaname='Ribbon' cls='NetUIHWND' />
<ctrl name='Ribbon' role='property page' />
<ctrl name='File Tab' role='push button' />
```

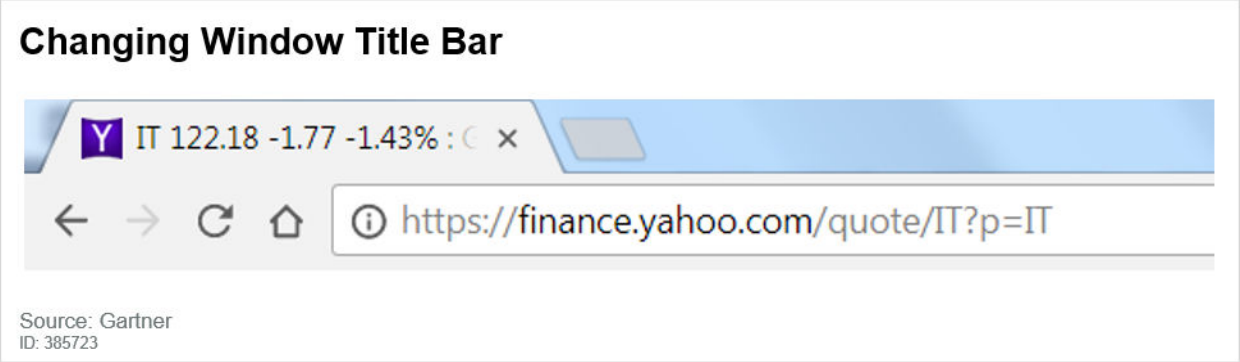
This level of granularity allows the recorder to accurately identify both the object and the action. The RPA recorder can capture a sequence of actions and attempt to repeat those actions against the same objects. Unlike an application-specific macro recorder, however, an RPA recorder will often have techniques that will allow for situations when explicit programmatic references aren't available. For instance, an RPA recorder might specify an action to be taken on an object based on its relative position to another object or based on the appearance (through screen scraping or computer vision) of an object on the screen.

The recorder, however, will not eliminate the need to develop scripts with the drag-and-drop interface. The recorder may work for extraordinarily straightforward and unchanging tasks, but any

task that requires conditionality or iteration based on a condition will require script development beyond the recorder.

An example would be recording a sequence to look up the Gartner stock price on Yahoo Finance. The title of the HTML document, which becomes part of the window title, has both the stock symbol and the instantaneous stock price, as shown in Figure 6.

Figure 6. Changing Window Title Bar



The recorder identifies the text of the stock price as:

```
<html title='IT 122.18 -0.17 -0.14% : Gartner, Inc. - Yahoo Finance' />
<webctrl aaname='122.18' idx='1' parentid='quote-header-info'
tag='SPAN' />
```

As soon as the stock price changes, the title of the HTML document, the window referenced by the recorder and the value of the variable of “aaname” will all change. As a result, this recorded script will fail on subsequent executions because any object reference on this page becomes invalid.

There are many techniques to overcome this changing reference. For instance:

- By using wildcards:

```
<html title='* - Yahoo Finance' />
<webctrl aaname='*' idx='1' parentid='quote-header-info' tag='SPAN' />
```

- By referencing the application and URL rather than the HTML title, as the container for the target object:

```
<html app='firefox.exe' url='https://finance.yahoo.com/quote/IT?=IT'/>
<webctrl aaname='*' idx='1' parentid='quote-header-info' tag='SPAN' />
```

- By any other method available in the bot development environment that can abstract the reference from the changing stock price but still provide absolute determinism in identifying the particular object needed for a workflow

When these scenarios arise, as shown above, they can usually be addressed, but tweaking the recorded workflow is not a function of the recorder. Therefore, any process that has the potential for even small variances will probably require more in-depth development than the recorder can provide. Bot developers will need to recognize references to processes and objects that might shift and either avoid the recorder altogether for these cases, or understand the tools and techniques necessary to modify a recorded workflow to make it repeatable.

> Some of these process changes that break scripts can come from untracked, external sources.

While not effective as a complete development replacement, the recorder can be helpful in discovering and mapping elements and objects within a task. In this way, the recorder can help traverse complex schemas and create element specifications for use in development.

Automating Remote Desktops and Interfaces

As already mentioned, one place where the recorder functionality can shine is when the script is automating an application on the desktop of a remote workstation, through Remote Desktop Protocol (RDP), Citrix, virtual network computing (VNC) or a similar system. In this case, the robot does not have access to either the window manager, Java stack or HTML source to use programmatic and precise object references. Instead, the remote desktop session is rendered as a flat image, with no locally accessible object references. In these cases, the recording function within RPA systems can leverage screen scraping or computer vision technologies to identify the elements being acted on. Without the recorder, automating a remote desktop application is still possible but requires an additional step. Without a recorder, RPA uses surface automation to segment the remote window into rectangular regions, which can then be selected by the script developer and analyzed for text or fields that can be the target of automation.

Similarly, many platforms include capabilities and technologies that allow them to more effectively capture a workflow in remote terminal (TTY) interface. Where the platform allows selection of a recorder type, select the most appropriate recorder type for the interface. Selecting the right type of recorder dictates how the recorder will attempt to identify on-screen elements, either through HTML specification, the window manager object schema, TTY interface or computer vision. While it might be tempting to use computer vision for all recordings, this is not always the best approach because it relies on inference and analysis to identify data elements, rather than explicit object specification.

Not all RPA platforms include a recorder. Because workflows created with a recorder will likely require tweaking for all but the simplest workflows, some vendors forgo the recorder and provide only a module-based studio. This could increase complexity for simple operations, likely excluding business users from creating the script, but could increase productivity for experienced script developers.

API-Based Control

RPA is not limited to automating systems through graphical interfaces. Because most RPA platforms include a module that allows for execution of code (like VB.NET, Python or JavaScript), an RPA system can be used to invoke remote APIs. To simplify this process, some RPA systems include application and data integration modules that allow bot developers to use API-based access without actually having to write code that invokes the APIs.

With the ability to weave both terminal, CLI-, GUI- and API-based operations into a single, automated workflow, RPA has the potential to enable new areas of automation and new automated workflows that haven't been possible.

Note, however, that where all systems involved in a process are modern systems with supported APIs, there are often solutions that are more scalable and more cost-effective than RPA. This is an important attribute of a process to consider when identifying well-suited cases for RPA and is covered in greater detail later in this document.

Bot Deployment Options

There are two deployment options for how and where a robot should run, and both present unique opportunities and challenges. Technical professionals should understand their deployment requirements before determining which vendors to evaluate, since not all vendors support all execution architectures.

There is a range of terminology to describe these approaches, and this document has used only RPA thus far. However, it is necessary to look specifically at the concepts of RPA and robotic desktop automation (RDA). In this section, and this section only, RPA and RDA will refer to the specific architectural and usage patterns found in "IEEE Approved Draft Guide for Terms and Concepts in Intelligent Process Automation" (IEEE P2755/D1). It defines two distinct automation concepts as follows:

- **"Robotic desktop automation (RDA):** Computer application that makes available to a human operator a suite of predefined activity choreography to complete the execution of processes, activities, transactions, and tasks in one or more unrelated software systems to deliver a result or service in the course of human initiated or managed workflow.
- **"Robotic process automation (RPA):** Preconfigured software instance that uses business rules and predefined activity choreography to complete the autonomous execution of a combination of processes, activities, transactions and tasks in one or more unrelated software systems to deliver a result or service with human exception management."

These two approaches are often called "attended RPA" and "unattended RPA," respectively. Elsewhere in the document, RPA will be used to refer to unattended robotic automation, and if relevant, a distinction between unattended and attended/RDA will be made.

At a high level, the development and coding of a script will not determine whether it is run as an RPA robot or an RDA robot. Both use the same operational constructs, conditional operators and

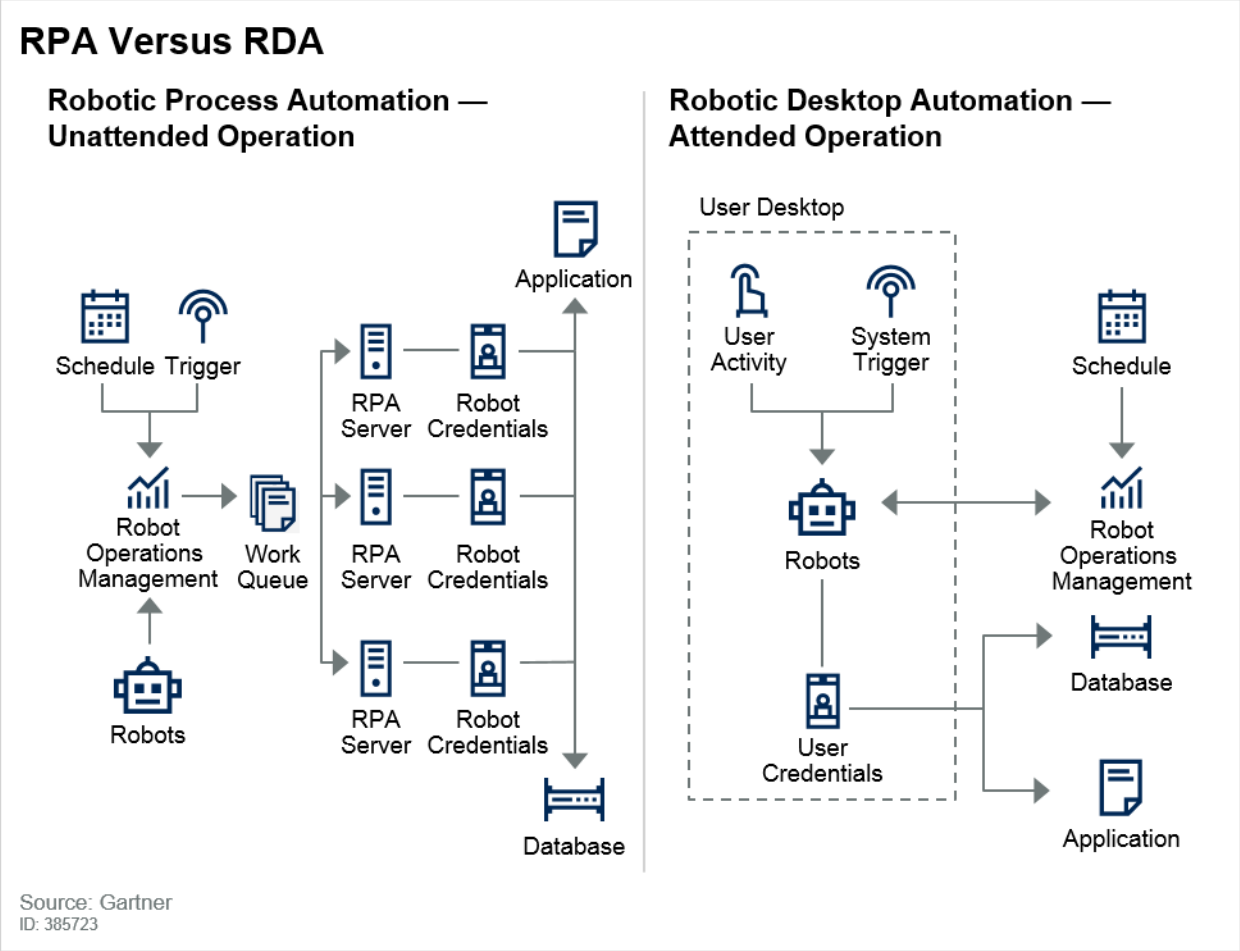
data access connectors. Fundamentally, the difference between these two architectural patterns is whether the script is running on a separate, isolated server (as in unattended RPA) or whether the script is executing using the resources on a local user's desktop (as in attended RDA).

Experienced technical professionals will immediately recognize the implications of these approaches:

- Shared resources versus dedicated resources
- Shared credentials versus isolated credentials
- Distributed management versus centralized management
- Physical security versus desktop access
- Remote versus local invocation and triggering

Figure 7 shows a simplified view of how these two approaches differ architecturally and operationally. RPA, on the left, uses dedicated and isolated server environments, usually accessible only to administrators and authorized operators, to execute scripts as they are triggered. RDA, on the right, has a robot running within the Windows desktop session of a user.

Figure 7. RPA Versus RDA



Both RDA and RPA systems allow for scheduling of script execution and triggering based on any number of detectable changes, such as a file being uploaded into an FTP directory, messages or, in the case of RDA, direct user invocation. Direct user invocation of an RPA script (that is, on a remote system) is possible, if not common, but requires the creation of an interface or messaging system through which a user can trigger the script.

Figure 7 highlights one of the most critical elements of an RPA deployment: the assignment, usage and management of credentials and identity within the script. Technical professionals should take special care to make sure that the introduction of robotic automation, particularly RDA robots, does not interfere with repudiation requirements. This would be the case shown in the RDA of Figure 7, where the robot is using user credentials.

Figure 7 also illustrates the high-level differences between RPA and RDA and doesn't cover all the potential scenarios and implementation patterns. For instance, RDA is often helpful in call center scenarios where each agent has an RPA runtime environment and scripts available to help with

agent activity. Establishing and maintaining distinct credentials for RDA systems is possible and recommended for most environments and use cases. Carefully consider the identity and access management (IAM) within RPA scripts. Vendor systems should be evaluated to ensure that their platform can meet the requirements of securing your environment.

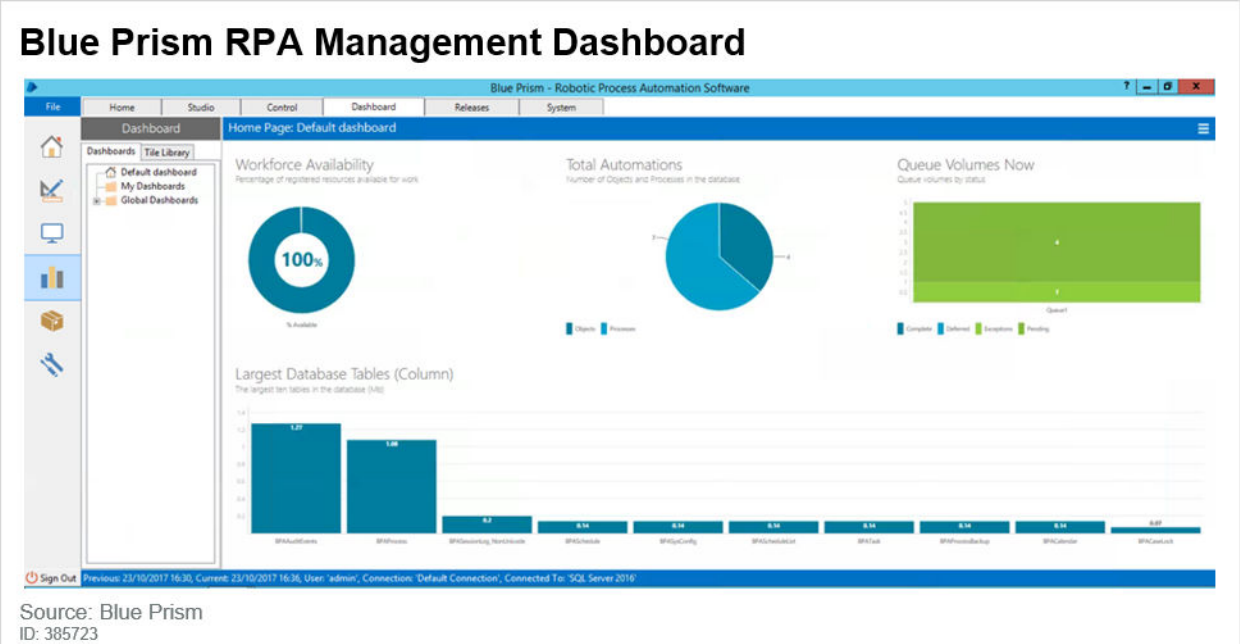
Bot Management Platform

Whether operating in an RDA or RPA environment, robotic automation introduces a distributed management problem that requires mature tools to help manage:

- Scheduling of scripts
- Managing distribution of updated scripts
- Identifying the state of currently running scripts and their runtimes
- Capturing errors and failures in execution
- Access to runtimes and execution of scripts

Since operations are continuous and ongoing, the RPA management platform is as important as the development tools or execution architecture. The RPA management platform gives immediate and actionable insight into the current state of the robotic workforce. As shown in Figure 8, operational attributes, like queue volumes, total number of bots and bot availability, allow an administrator to quickly assess the state and trends of the entire automation platform.

Figure 8. Blue Prism RPA Management Dashboard



Source: Blue Prism
ID: 385723

More Than a Dashboard

In addition to giving operational insight, the RPA management platform will serve several other critical roles required to actively manage the environment. Central version management, for instance, can automate the updating of scripts across a cluster of RPA runtime environments. Without a centrally managed versioning capability, administrators will need to manually identify and update outdated scripts that are incompatible with the updated process.

Some RPA management platforms also provide a centralized repository for the log files generated by the robots, which obviously simplifies the troubleshooting or auditing of a remotely executing robot.

One of the most critical functions of the management platform is credential management. With robots actively accessing and manipulating data, thoughtful consideration and intentional implementation of credential management is a requirement of all RPA deployments. The management platform of an RPA system should allow for the creation, deletion and expiration of credentials for the RPA system, as well as the encryption of any locally stored credentials, if the use case requires them. “IGA, RPA, and Managing Software Robot Identities” is required reading for architects and technical professionals responsible for IAM.

Since the management platform will schedule scripts and manage versions on remote systems, the RPA system management platform provides the interface for adding, monitoring and managing compute resources used for execution. This allows visibility into, and management of, capacity, availability and utilization. At the same time, this interface provides for the assignment and scheduling of scripts onto resources.

Lastly, the RPA system itself requires administration of user access and authorization. The management platform needs to provide the ability to define which users can create, modify or remove scripts and runtimes, as well as perform other administrative functions within the RPA platform. Other common operational actions, like adding or removing resources, accessing logs, or changing configuration of the RPA system, all require authorization management within the RPA platform.

Best Practices for Adopting and Operating RPA

RPA initiatives often begin outside IT, with a business organization driving the technology adoption for its own purposes. In many cases, too, there is a strong perception that this adoption requires only minimal involvement with IT or the broader set of technical organizations. While RPA does simplify the creation of automation scripts, governance of both the usage and operations is necessary to ensure achieving the goals of automating with RPA.

RPA can perform a wide variety of actions, which makes it analogous to an “automation Swiss Army knife.” But, like that tool, the wide variety of capabilities often comes with constraints on the ideal use cases. For instance, a Swiss Army knife might contain a saw, but trying to fell a large tree with that saw will likely end in failure, frustration and, perhaps, a broken tool. If, instead, we constrain the

usage of that saw to saplings with less than a one-inch diameter, the success rate will be much higher, user satisfaction will be much higher and the risk of tool breakage goes down significantly.

Putting constraints around the use cases, aligning developer persona to those use cases and putting in just the right amount of software development life cycle (SDLC) management are the keys to success with RPA.

Establish the Developer Persona

One of the most critical elements in defining the right kind and level of RPA governance is to establish a vision for the persona of the RPA developer within an enterprise. Obviously, there is a range of existing developers who can take on RPA development. But, because of the simplified development experience and some misset expectations around the capabilities of the recorder, many organizations see RPA as a transformative technology that will allow nontechnical users to develop their own automations and alleviate mundane tasks. Achieving this vision requires a much more detailed examination of the end-to-end RPA life cycle, the cultural implications and the governance processes necessary to help steer these newly anointed “citizen developers.”

Citizen developers are not generally familiar with SDLC requirements, operations and security best practices. So things like identity and credential management, source management and versioning, data access associated with an identity, support, and disaster recovery planning will be foreign to most of them. At the same time, the risk of ignoring these disciplines entirely is terrifying. This is why it is critical to establish the vision for the developer persona. Relying exclusively on developers who are steeped in SDLC systems and processes is straightforward and can build on existing software development practices. But enabling citizen developers will require the definition of new processes, procedures and policies to govern and support their efforts in a way that doesn’t foist the entirety of the SDLC requirements on nondevelopers.

> Deciding which developer personas will create RPA scripts is critical to establishing the right level of governance for the application and operation of RPA.

Establish a Governing Body

Nearly every organization that has successfully adopted RPA has relied on some form of governance body — either a center of excellence (COE), a steering committee or, at the inception of the RPA project, just an individual. The governing body will provide the guidance necessary to ensure that RPA is used only when it is the right tool and that RPA applications comply with all corporate and regulatory requirements. In many cases, the scope of the COE is larger than just RPA and might focus on innovation, automation or a higher level than just RPA. Having the right application of governance will help ensure ROI for RPA investments and mitigates the risk of automation error in production environments.

RPA governance should focus on three key milestones:

- Establishment of process selection criteria for RPA
- Approval of RPA application to a process
- Validation of the RPA code before promoting to production

Areas of Governance

The process selection criteria ensure that RPA is used when RPA is the right automation tool and not used when RPA isn't the right automation tool. Without this oversight, citizen developers who have only RPA as an automation tool will see every opportunity to automate as an opportunity for RPA. Seasoned developers, too, might use RPA as a way to get an automation done quickly, even when RPA isn't the ideal choice, or even avoid RPA when it is the right choice for more familiar tools. By proactively establishing the criteria for processes that are well-suited to automation with RPA, enterprises can help align expectations and ensure success with RPA.

The governing body should provide approval and validation that a process truly meets the criteria. A developer may not be able to assess some criteria against a process. For instance, the criteria around data quality suggests that the initial applications of RPA should focus on processes where there is an upstream guarantee of data quality. Citizen developers may not even fully understand this requirement, let alone possess the skill to validate the state of data quality within a process. Even experienced developers may not have sufficient visibility to assess the data quality around a process. Therefore, the COE or governing body should support the RPA developer community by providing the expertise necessary to validate and approve the selection criteria against the proposed processes.

Finally, once a developer has completed their RPA script, there must be a validation phase before code is promoted to production. The governing body or COE must define the validation or testing requirements that must be met for production release. The validation should be the same for all developer personas and should support the development cadence, especially where there are agile or iterative releases. Just as with any other software development, no code should be promoted to production without validation.

The governing body may also establish best practices, development requirements and/or operational requirements. These other areas where governance can help improve reliability or operations include:

- Logging standards
- Identity policies and best practices
- Authentication best practices
- Shared code management policies and processes
- Exception handling standards
- Alerting and notification standards

COE Composition

The composition of the COE should reflect the areas of governance and the vision for RPA adoption. In most cases, start with the two primary stakeholders:

- A business owner to represent the interests of the citizen developers and the business process owner
- A member of the technical team responsible for RPA operations and automation architecture

This combination will need to scale and adapt to additional RPA adoption. But focus on providing the right mix of business and technical expertise to establish the process selection criteria, assess candidate processes for RPA and validate the implementation of a script. As the usage of RPA expands, additional roles within the COE may be needed to support the volume of work or the adoption efforts. As the use of RPA expands, the COE may need additional representation from other teams, including:

- Security — A security representative can provide guidance and assessments for potential applications, a strategy and guidelines for identity and credential management, and validation of the implementation of all security requirements in RPA code.
- HR — Some organizations look to RPA as part of a transformation initiative within the workforce. HR can help realize this vision by creating incentives and recognition for citizen developers who grow their skill set. In some cases, a well-established practice around RPA can change the hiring requirements for a role, and in others, RPA is adopted with the goal to reduce human capital investments in mundane tasks.

> Having HR within the governance body for these aggressive adoption strategies can be critical to realize the transformation vision.

- Data management — As RPA touches more process and more data, having data management representation within the governing body can help identify data quality requirements. Going forward, this role can help establish data quality pipelines to support strategic applications of RPA.

Using a COE to Drive RPA Adoption

Requests to apply RPA can come from across the enterprise, and in many cases, the experts on a process will not be within IT. A COE consisting of technical professionals, business owners and supporting roles can function as a collaborative, impartial governing force to establish and evolve the process selection criteria and review RPA requests and the impact to the workforce. Ultimately, an effective COE will improve success by managing the pipeline and expectations for which processes are ready for RPA and maintaining alignment between business and IT around the priorities and impact of the overall effort.

Use the COE to establish and communicate the process selection criteria that the RPA development team is prepared to handle with existing resources, skills and experience. Then, create a review process where candidate processes can be scored on how well-suited they are for the current set of process selection criteria. Finally, establish a roadmap for process selection criteria, giving potential process owners insight into when the team might be ready to tackle processes that weren't approved with the current set of criteria.

Use Selection Criteria to Prioritize Well-Suited Cases for RPA

One of the most critical determiners of success with RPA is correct application of the technology. This requires establishing criteria for identifying and prioritizing the use cases you and your team are prepared to tackle. For technical professionals who are beginning to apply RPA, there are a few attributes that will simplify the development and support of the script and help ensure success with RPA. Figure 9 captures many characteristics of well-suited candidate processes.

Figure 9. Selection Criteria for Well-Suited Processes

Selection Criteria for Well-Suited Processes

No RPA Alternative	Processes should rely heavily on legacy systems or services that do not have supported APIs.
No Case for Modernization	Processes should have been evaluated and rejected from being reimplemented or modernized with supported APIs.
Stability	Processes should not change very frequently.
Iteration	Processes should repeat often enough to justify RPA investment, but not so often that it introduces risk or inefficiency.
Process Quality	Processes are well-architected, well-implemented and well-documented.
Data Quality	Processes have data sources that are prepared for robotic consumption — that is, they are of good quality and are appropriate for the credentials used within the script.
Process Mechanics	Processes that include conditionality must have decisions that can be expressed and resolved through regular expressions.

Source: Gartner
ID: 385723

Evaluate Alternative Automation Approaches

The first two attributes in Figure 9 are the most important criteria to consider because they protect enterprises from using RPA in ways that create more cost and complexity than RPA resolves. In

2019, modern enterprise applications and services are architected and implemented with supported APIs. If a system has supported, documented APIs, consider developing automation through these APIs (even, perhaps, with RPA), rather than driving automation through robotic manipulation of on-screen elements. These programmatic interfaces allow remote operation and manipulation of data and services in a way that inherently embraces the architectural patterns of automation.

Well-implemented APIs might be a better choice for automation because they are:

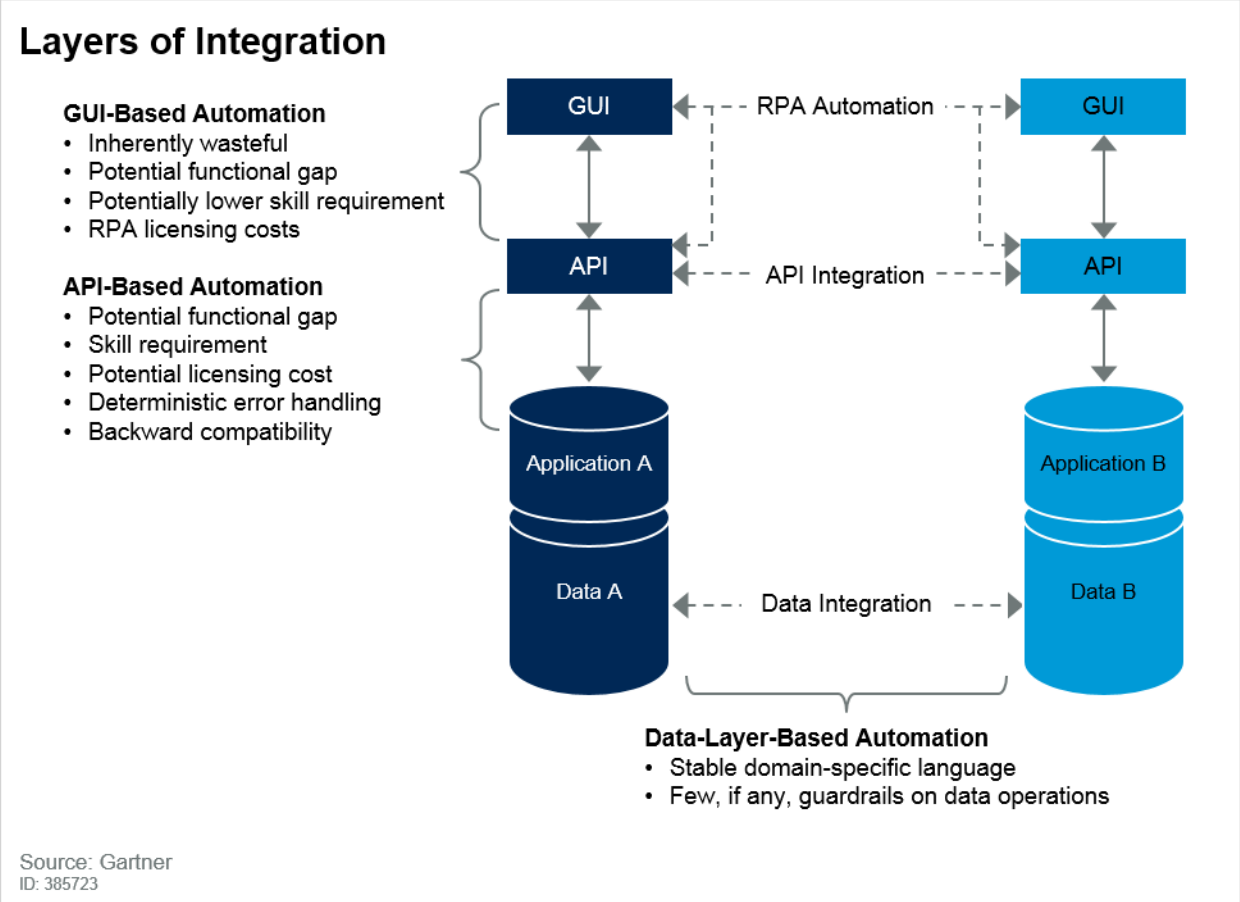
- Documented
- Instrumented for troubleshooting and debug instrumentation
- Versioned with explicit definitions of forward and backward compatibility
- Purpose-built for the kind of machine-level interaction that underpins automation
- Easily integrated with developer and quality engineering (QE) toolchains for validation and testing

> Using RPA to replicate human interaction with a system that has robust APIs wastes the investment in creating those APIs and demands artful translation of human interaction into machine operations.

As Figure 10 shows, integrating through APIs, if they are available, is a shorter and simpler path to automated machine operations.

Where citizen developers are part of the RPA adoption plan, expect for the COE to provide or refer expertise that can help ensure whether a potential system has or doesn't have supported APIs.

Figure 10. Layers of Integration



Modernize Strategic Systems Before Automating

For custom-developed systems that do not have APIs, consider implementing APIs as a potential strategy for enabling more robust automation. In some cases, the long-term cost of modernizing a legacy system with a machine interface (that is, an API) is much lower than automating with RPA. At the same time, adding the APIs might create additional efficiencies or opportunities to integrate.

Always evaluate whether there is a case to justify modernizing a potential process to enable API-based automation. By eliminating processes that don't have a compelling alternative automation approach or modernizing those that merit a more robust automation interface, enterprises can ensure that they're thinking of automation strategically and applying RPA most effectively within that strategy.

Focus on Stable Processes to Avoid RPA Rework

The ideal process would be one that doesn't change frequently and has little to no justification for change in the future. There are lots of these processes and legacy applications in the enterprise

market. Stable processes have the obvious benefit of not requiring frequent changes within the scripted workflow. In addition, stable processes are more likely to be completely understood and, as a result, easier to model. From a cost perspective, if robotic automation is replacing a human worker and the process changes frequency, the burden of reacting to that change might shift from a relatively low-cost administrative worker to the cost of script development and testing.

Express the stability of a process and its associated system as both forward-looking and backward-looking criteria. For example, "Approved processes will have no documented changes in the previous six months and no planned changes in the next 12 months." The numbers are not absolute and will vary between enterprises, based on the maturity of their RPA platform and the level of integration with change management processes. For organizations that are early in RPA adoption, this can help reduce pressure to quickly mature change management around RPA and, potentially, citizen developers.

Maximize ROI and Reduce Risk With Iteration Criteria

Apply criteria for both the maximum and minimum number of times that a process must run within a time frame to be well-suited for RPA. By establishing a minimum value, the criteria will help eliminate processes that do not represent enough work to merit automation. Given that RPA costs are associated with runtime capacity, judicious use of runtime licenses can help ensure ROI. At the same time, processes that run very infrequently can often be subject to unexpected changes that break the automation.

At the other end of the spectrum, establishing a maximum number of iterations helps identify modernization candidates and reduces the risk of using RPA on too ambitious of a process. An absolute maximum is an effective tool for identifying processes that involve enough work and value to justify a modernization effort and automation through an API interface. For example, a process that runs 10,000 times per day should be automated, but that volume of automation is almost certainly more reliable, scalable and cost-effective if done through APIs and not through a GUI. Again, the exact number will vary from enterprise to enterprise, depending on their capacity for reimplementing legacy systems and RPA maturity.

A similar relative maximum that governs the number of cycles per period can help mitigate the risk of a developer being too ambitious with RPA-based automation. It is often helpful to manage these criteria differently for different developer personas or experience levels. For instance, an experienced developer might be limited to 500 cycles per day on an approved process, but the citizen developer's use of RPA is approved only up to 100 cycles per day. In this way, the iteration criteria can help mitigate the risks of a developer error.

Make RPA Easier and More Effective With Quality Processes

One of the attractive features of RPA is that an RPA bot can interact with a system or service using the existing interface as is and without requiring changes to accommodate the automation. Therefore, it should never be necessary to undertake a process optimization effort exclusively to automate through RPA. That said, automating a process with RPA may expose inherent weaknesses in poorly implemented processes. Moreover, if the goal for automation is to improve efficiency, but if

current efficiency is limited by the process, rather than the worker, it is unlikely that RPA will achieve that goal.

Therefore, ideal processes are those whose design is intentional and implementation efficient. Any efforts to improve or optimize a process should precede the implementation of an RPA system and validate whether RPA or rearchitecting around supported APIs is the best path to automation of a system or service.

Measuring process quality can be difficult. As a proxy, consider establishing an initial criterion around process documentation. If this were any other software development initiative, a detailed and complete specification would be a prerequisite. Use this precedent and proven best practice to establish a requirement, at least initially, that all approved RPA use cases must have complete and reliable documentation of the “golden path” for successful execution. Be diligent, too, about subject matter experts overlooking context or preconditions. The documentation should be complete enough that someone with no experience can execute the process repeatedly and successfully.

The documentation can take whatever form best conveys the process workflow. Some clients have found that video documentation with narration can be an effective way to quickly capture high-quality documentation around a process. Whatever the form, quality documentation will pay dividends beyond the development phase as the documentation can facilitate both support and ongoing maintenance.

Require Upstream Data Quality

Data quality, particularly source data quality, plays an important role in identifying the right processes for RPA. The RPA system will automate the manipulation of data and propagate the source data across other systems. Therefore, if there are any potential data quality issues in the source data, those should be addressed before the robot propagates the error and creates a larger data problem down the road. When identifying early candidate use cases for RPA or cases for citizen developers, prioritize processes with a guarantee of upstream data quality.

Over time and depending on the developer persona, the role of RPA in managing data quality can evolve. Once the RPA platform is proven stable and developers have experience in creating reliable automations, the scripts can be enhanced to do rudimentary data validation. For example, when reading a “phone number” field from a record, if the robot encounters alpha characters, the robot can move to the next record and notify the process owner that there is a data issue. In this first step toward integrating RPA into the data quality pipeline, the robot isn’t an active element, but it enhances overall quality by not propagating existing errors. The role of RPA in that data quality pipeline can continue to grow, limited only by developer expertise.

In many cases, however, the remediation of data quality will require policy-driven, robust techniques that are beyond the capabilities within an RPA platform. In these advanced cases, sufficient developer expertise could dispatch a record to a data quality platform where the problematic element could be addressed and fed back into the RPA system.

This evolution is a clear example of a potential ceiling in citizen developer use cases. Use the criteria to constrain citizen developer use cases to processes where RPA will not compound data quality problems.

Another common scenario is to use RPA to extract handwritten text from scanned documents and push that data into a structured object store. There is tremendous potential for such a system to unlock decades of legacy data and bring it into a modern, digital enterprise. Early experiences with these systems suggest that they can be quite effective on typewritten text. However, the quality of handwriting can vary wildly, depending on the quality of the scan, the choice of OCR engine, the readability of the handwriting and the extent to which letters overlap. If necessary, use a secondary data quality criterion to constrain the use of OCR for handwritten text.

Manage Process Selection Criteria as a Roadmap

These process selection criteria are not a static definition. Instead, they should evolve to reflect the level of maturity, skill and investment around RPA. For example, as developer skill increases, criteria can be adjusted to allow more complex and ambitious use cases. Or, as the change management processes around RPA are established and matured, the change windows for well-suited processes can be expanded. By establishing conservative metrics at the beginning of RPA adoption, the roadmap can reduce risk that early failures will have a significant impact and provide visibility for when the organization is ready for more ambitious and challenging processes. Figure 11 provides an example of the types of metrics that can define this roadmap and the expansion of potential RPA use cases over time.

When there are multiple personas, it can be helpful to establish separate, parallel roadmaps for well-suited processes. This would allow for independent governance of both citizen and experienced developers. The governance body would be able to distinguish between the skill sets of the two personas and provide specific criteria, where citizen developers are held to a more risk-averse set of applications, and experienced developers tackle more complex automations.

Figure 11. Manage Selection Criteria as a Roadmap

Manage Selection Criteria as a Roadmap

Attribute	1H19	2H19	1H20	2H20
No Automation Alternative				Design for robot
No Case to Modernize				Design for robot
Stable	**No changes:** • Previous 12 months • Next 12 months	**No changes:** • Previous six months • Next 12 months	**No changes:** • Previous six months • Next six months	**No changes:** • Previous three months • Next six months
Repetitive	**Iterations:** • 20-50 per day	**Iterations:** • 15-100 per day	**Iterations:** • 15-200 per day	**Iterations:** • 10-500 per day
Process Quality	Complete, explicit, specification-level "Golden Path" documentation.		High-level documentation in place. Consulting available for specification-level detail.	
Data Quality	Guarantee of upstream data quality	Data validation within RPA script		ETL integration for quality remediation
Mechanical	Isolated evaluation of local variables with regular expressions	External, programmatic decision input		Binary, schema-defined human interaction

Source: Gartner
ID: 385723

Establishing the selection criteria roadmap can help establish metrics and prioritize investments around maturing the RPA program, too. The impact of investments around data quality, for instance, can be seen in the roadmap as more processes become viable for RPA automation. The roadmap can also reflect the progress of citizen developer skill development and create key milestones for assessing the advance of capabilities.

Apply Release Management to RPA Scripts

Ensure that RPA scripts are documented and versioned. There will, undoubtedly, be cases where a recent change or update to a script will introduce a regression or find a bug in an external system. In these cases, having versioned scripts with even a few bullets of release notes or developer documentation can help to roll back to the previous, working version quickly and minimize interruption.

Examples of critical aspects to document for each process include:

- The goal of the script version (i.e., the reason that this script was created)
- High-level architecture
- The catalog of systems that the robot will access when running the script
- The credentials used by, and authorizations created for, the robot for each disparate system
- The versions, where appropriate, of the remote systems with which the robot will interact, or a compatibility matrix between scripts and their target systems
- The individual, manager and organization that requested and justified the bot

Having even this simple list can save hours trying to troubleshoot an issue when one arises.

Early in the adoption of RPA, with versioning established, enterprises should define a strategy for managing scripts “as code” around their RPA platform. This will not only allow quick rollback in the event of a regression or new bug in a new release of the script, but also simplify the RPA platform and script disaster recovery planning. Where the RPA developers are familiar with source control tools, leverage integration between the RPA platform and source control tool to enable source control. Since citizen developers may not be familiar with these tools, expect to either train them or provide a supporting function that will manage their code check-ins on their behalf.

All these disciplines will become more important as the role of RPA grows and becomes more strategic. Remember, too, to keep the SDLC overhead as low as possible, especially when citizen developers are involved. Communicate clearly with other team members who may not be familiar with this level of discipline or the application of these methodologies.

Test and Validate RPA Scripts Before Releasing to Production

Focus on establishing a robust testing and validation framework for RPA scripts (see Figure 12). The concepts of testing and validation work together to ensure proper and secure operation of RPA

scripts. Validation of the script is a human review by technical members of the COE, and testing is the observation of script execution to confirm that the robot performs as expected.

Complete validation of an RPA robot will require validation in the form of review of key architectural and implementation features, such as:

- Data access
- Identity and access management
- Execution strategy and resource allocation
- Upstream data quality
- Behavioral monitoring strategy
- Overall architecture and workflow to process mapping

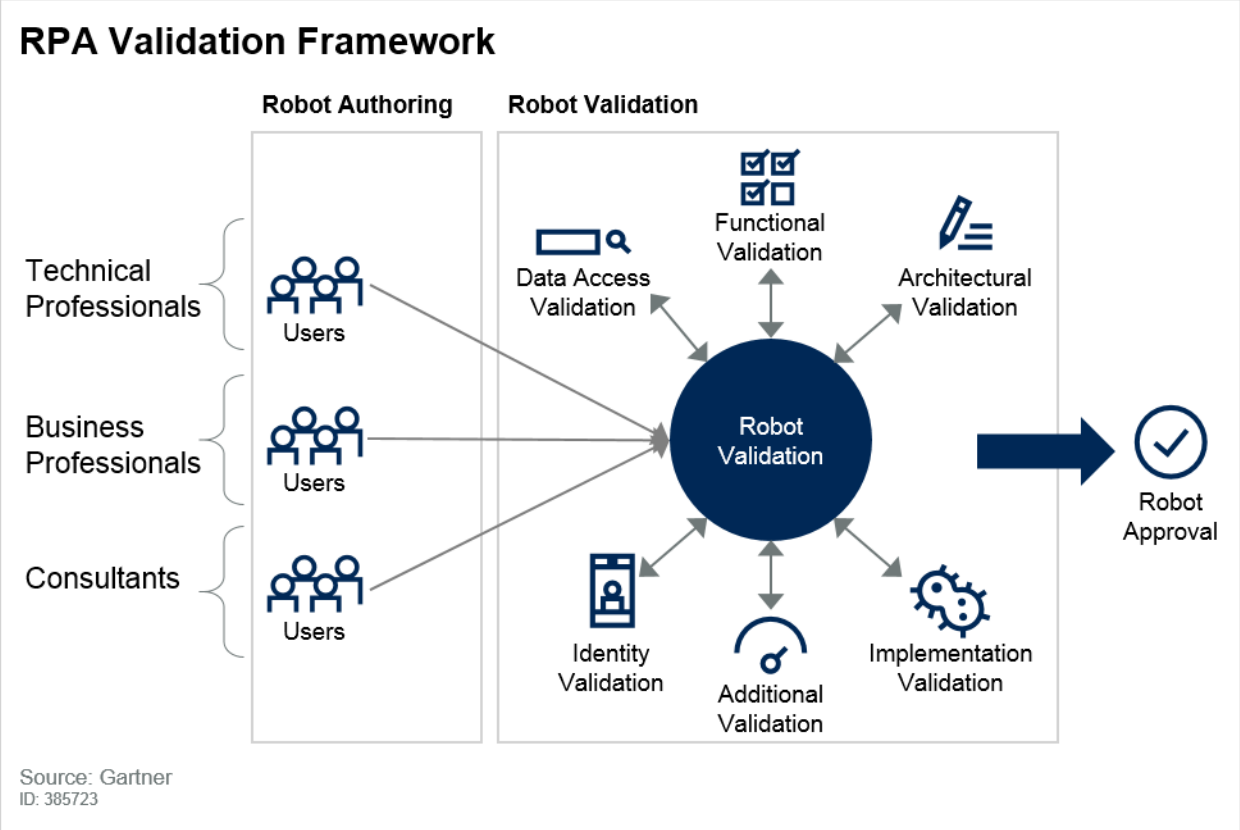
Identity management and access management are the two most critical elements of validation. The RPA operations team or security should ensure that the credentials and identity used for the script are appropriate for the operation. One goal of this validation is to ensure that nonrepudiation is not violated and that robot activity is always discernable from human activity. With citizen developers, in particular, user logon credentials are often used out of convenience. In this scenario, it becomes impossible to discern whether humans performed an action or a robot (potentially developed by someone else) took the action using their credentials.

The other critical dimension of validation is data access. Robots should not execute scripts with privileges far beyond what is required for their intended task. At extreme scale, it may not be reasonable to have unique identities and authorizations for every individual script, and group policy might help scale the identity management. In this case, strike a balance between restricting access and managing robot identities. Ensure that robots do not have extraneous access to unnecessary data repositories, and ensure that any RPA application that accesses sensitive or regulated data adheres to all requirements.

For a more detailed examination of managing credentials and identity for RPA, see “IGA, RPA, and Managing Software Robot Identities.”

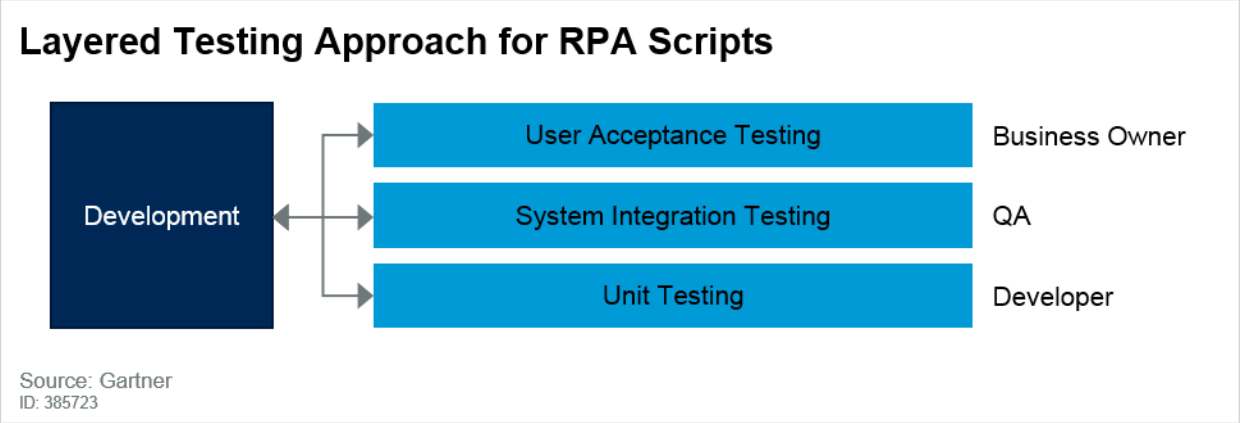
With the right validation review framework, many pitfalls (often encountered by citizen developers) around RPA identity and data access can be eliminated. This becomes a critical milestone in getting to production and should be applied regardless of who developed the RPA script.

Figure 12. RPA Validation Framework



Successful testing of the script should also be required to have a script promoted to production. Testing in RPA is relatively straightforward and usually consists of a series of signoffs that the script does, indeed, execute the process successfully. Figure 13 shows a three-layer approach common in many software development environments. The developer is responsible for unit testing and ensures that the components of the script work individually. Most RPA tools, too, will have some rudimentary code validation tools to ensure variables are declared properly and that there are no infinite loops, divide by zeros or other errors in the script.

Figure 13. Layered Testing Approach for RPA Scripts



Once unit testing is complete, a test group will look again at the operation of the script in the context of the full process automation. Where possible, this testing should not take place on production systems. However, many clients report that, because RPA is often used on systems that are no longer in active development, there is no development environment. Even in these cases, you should take any and all available measures to isolate production data from RPA testing. Once the system integration testing has confirmed that the RPA script is accessing and interoperating with systems properly, the script can be validated by the process owner through user acceptance testing.

RPA code should be promoted to production only after completing both a validation review of implementation and the battery of functional tests.

Applying the appropriate SDLC discipline to RPA scripts will reduce errors, improve reliability and availability, and protect your RPA efforts from falling victim to lessons already learned in software development. RPA scripts, just like every other system implemented in the data center, should be managed through the full life cycle of investigation, analysis, design, build, test, deploy, and maintenance and support. Whether applying waterfall or agile development processes, it is critical is to ensure that each script is well-architected, sufficiently documented, thoroughly validated and properly supported over time.

Work within the COE to find the right level of governance, which balances between the burden and benefit of applying these disciplines.

Ongoing Operations for RPA Scripts

Postrelease, RPA scripts will require some measure of ongoing maintenance and operation. To varying degrees, all aspects of application support and operation are required. Support, disaster recovery, change management and monitoring seem to be the most common.

Support for and of RPA scripts is almost always handled by an existing support interface. Attempts to extend the citizen developer model to the support space will face significant challenges in tools, skills, technology and culture. Instead, most organizations find that existing IT support processes and organizations are effective at managing the first-level support and ticket management. The RPA developer, whether a citizen developer or technical developer, is a point of escalation. In either case, the documentation that was gathered in the process proposal and approval phase is often a key enabler for the transfer of support responsibility.

Without question, IT and the RPA operations team will need to manage the DR responsibilities for both the RPA platform and scripts, regardless of developer persona.

Change management, on the other hand, will often require the participation of the business process owner. Quite often, the manual, legacy processes that are well-suited for RPA are not under any formal change management, which leads to increased failures due to unforeseen changes. Where there is a change management process familiar to your RPA developers, applying that to RPA can mitigate the impact of these changes. Simply putting RPA processes under change management may not be an option, however. Where citizen developers are creating the automation, they're usually not familiar with change management tools or processes. In this case, the best way to engage citizen developers in change management might be a simple ad hoc process to communicate any impactful changes. Given that formal change management isn't an option, even ad hoc oversight is better than nothing.

A similar challenge exists around monitoring and management of ongoing RPA operations. That is, "citizen developers" do not translate to "citizen operators." Given, too, the additional administrative rights of using the management console, there's additional risk of extending access to nontechnical users. Therefore, most enterprises reserve the management and monitoring for members of the technical or RPA operations team. However, because many RPA applications have a strong interest from the business owners, you should establish a reporting cadence and content without requiring access to the RPA management console.

Automation and Transaction Throughput

Before closing with the impact of artificial intelligence (AI) on RPA, consider one last aspect for which technical professionals might need to plan or prepare for RPA — the impact of automation on underlying and ancillary systems.

In many cases, augmenting workers or offloading tasks to robots entirely will have no significant impact on underlying data repositories, architectures or infrastructure. There are, however, cases where the robot can, even if only intermittently, increase the volume or pace of transactions to the point that existing application servers or data access systems may not be able to handle the increased load. Although it is less likely, the opposite could be true, too: A poorly implemented script or overloaded runtime environments could result in a *decrease* in overall transaction rate. Understanding the expected and observed transaction rates will help measure impact and help troubleshoot performance or resource problems.

This is another reason why the COE and a modicum of SDLC discipline are important. Having early insight into which processes will be automated and which data repositories might be affected gives

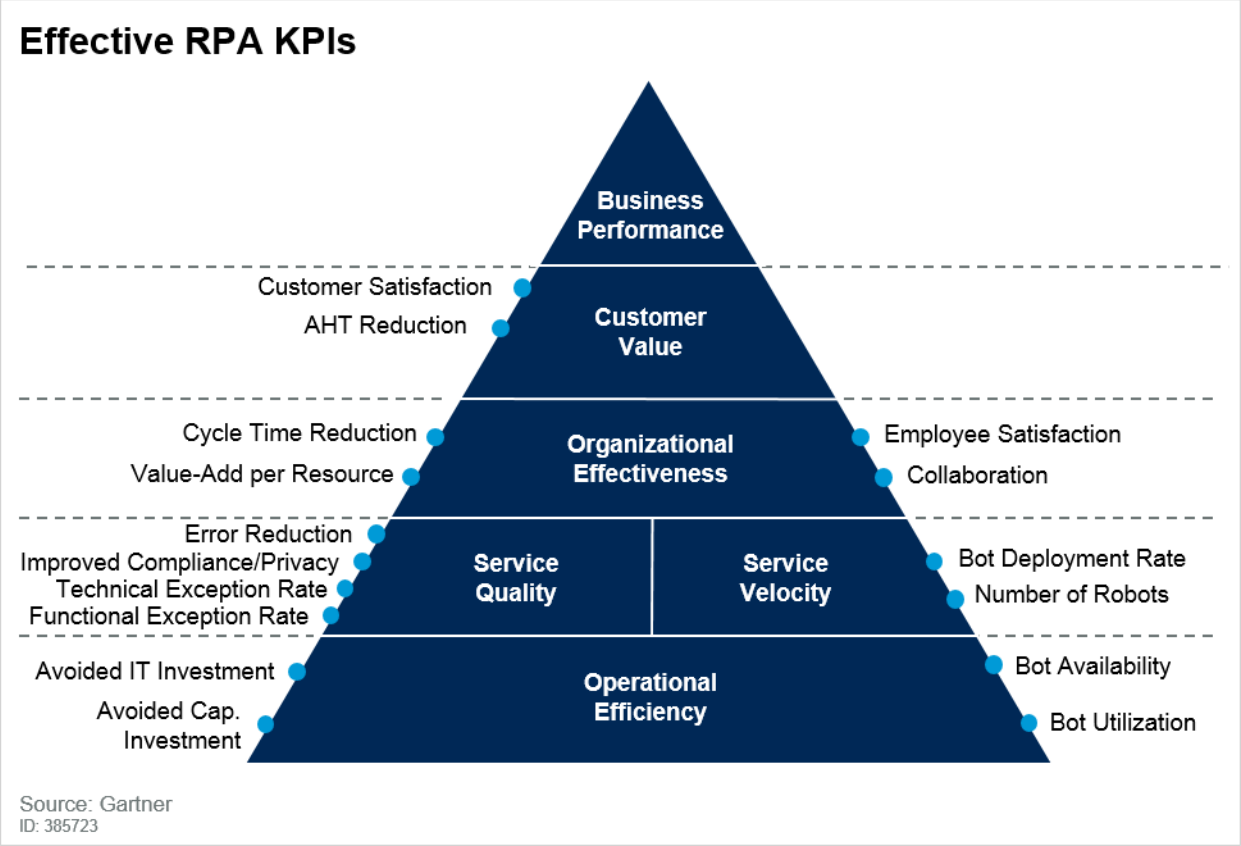
technical professionals the opportunity to check for dependencies that might be at or near capacity. This could even be considered a process selection criterion itself. An organization might decide that the automation of a process must not require additional capital investments in any underlying system.

Measuring Success With RPA

Measuring and reporting success with RPA is critical in advancing the overall automation and modernization agenda. However, many organizations report that RPA falls short of expectations because of unrealistic goals. With RPA being most effective at repetitive, mechanical tasks, the impacts are rarely reflected in top-level business goals, like overall revenue or sales growth. Additionally, goals that are outside the domain of the RPA platform — such as head count reduction — often disappoint because they involve many other factors outside offloading mechanical tasks.

Instead, set expectations, as shown in Figure 14, around operational efficiency, service delivery and organization effectiveness.

Figure 14. Effective RPA KPIs



Avoid metrics for RPA success that go beyond the direct impact of RPA. Metrics like head count reduction or productivity depend on external factors that are outside the governance of the RPA COE.

The Role of AI in RPA

AI promises to catapult RPA into new areas of automation and alleviate some restrictive application of robots caused by the mechanical, rigid nature of workflows. IEEE describes the emergence of these platforms as follows:

> *"In the last several years, an all-new family of process automation technologies has emerged with a great deal of attention. These new products are quite different than the simple click recorders of yesteryear. Complex execution engines with fully developed management platforms are changing how business is done. Often coupled with increasingly sophisticated rules engines, analytics, machine learning, and cognitive computing, these new tools can perform tasks previously requiring human operators. More advanced tooling is now starting to involve context assessment, reasoning, decision-making, and automated orchestration of service process fulfillment. Collectively, this capability is known as Software-Based Intelligent Process Automation (SBIPA) or Intelligent Process Automation (IPA) for short."*

Exploring the implications and understanding how technical professionals will need to respond to this trend is simplified by segmenting the discussion into:

- AI within the RPA platform
- AI external to the RPA platform

AI Within RPA

With machine learning (ML) and ML-powered AI (MLpAI) defining new ways for machines to analyze and interpret data, there's a natural fit to a system that is trying to replicate human activity. In fact, some vendors have already incorporated MLpAI functionality to improve on-screen element specification resilience. In this case, a computer vision AI component, rather than a static, schema-driven specification, can "see" and identify an element, even if it has moved or changed, making the script more resilient to process changes.

Without a doubt, the trend to augment the full spectrum of pseudohuman activity and RPA capabilities with greater "intelligence" is becoming a core strategic component for all RPA vendors. Potential capabilities and enhancements of RPA through incorporating AI hint at future robots that can:

- Use ML classifier or clustering algorithms to make interpretive, rather than expression-based, decisions on how to handle data or workflow decisions. This allows enhanced complexity and conditionality capabilities.
- Enable iterative, reinforced learning that allows for multiple passes or multiple users to teach, rather than record, the system how to execute a workflow. Such a system could identify

combinations of efficiencies or decision logic across iterations to refine an automated workflow or learn which hierarchical element specifications are fixed and which change.

- Expand and enhance computer vision capabilities to do even more advanced object detection or higher-confidence OCR.

One of the best strategies to address this trend is the incremental approach managed through the process selection criteria. As these new technologies become more pervasive, use a limited number of processes to both gain proficiency and build trust in how these systems operate and how to manage these cognitive actors.

AI External to RPA

At present, there are nearly innumerable opportunities to combine RPA with ML, MLpAI and cognitive systems to create cutting-edge systems that can bring sophisticated automation to an already powerful automation framework. Some of these cases require embracing the belief that these advanced AI systems would sit beside legacy applications that do not have supported APIs, pointing to RPA as the best automation strategy. For some readers, that will be difficult to imagine. For other readers, this will be commonplace. This document won't try to catalog all the ways that AI can work with RPA, but instead will highlight a few key areas to stir creative thoughts within your own systems and applications.

Decision making in an enterprise is always complex, even for humans. Deciding how to respond to a customer service inquiry, for example, requires several layers of understanding: Is the customer satisfied or unsatisfied? How do we resolve the issue? What are the business rules and policies that govern this issue? This would be an exceedingly difficult scenario to automate with RPA on its own.

If such requests are processed by a system that has a far-richer understanding of the business context and rules, the results could be published as a variable accessible to the RPA robot. In this way, we can gain additional value from the existing expert system, avoid the need to replicate complex insights and rules, and augment existing RPA systems to automate systems that require decisions impossible to express as a regular expression.

Computer vision systems are already present in some RPA platforms as a way to specify elements or extract text. But many more opportunities exist to use the data that a computer vision system extracts from images and video as input to an RPA process. A computer vision system could detect, for instance, that a video feed contains a damaged box on a conveyor belt, as the asset tag is scanned into inventory. While the workers or machines on the line address the physical asset, an RPA system could automatically capture and log the date, time and location of the damaged asset in a loss reporting system. Even though the robot is still performing a straightforward process of copying the information from the inventory system to the loss reporting system, the workflow is triggered by an MLpAI system.

Chatbots, or "conversational platforms," are often envisioned next to RPA robots, giving a whole new way for a worker to invoke a robot. Simply sending an SMS message or speaking a command to a desktop virtual assistant could initiate any manner of automated workflow.

Recommendations

- **Embrace RPA as a targeted automation technology for well-suited scenarios.** Use RPA within and alongside other automation platforms to get compounding effects of complementary automation technologies.
- **Focus on a robust platform and process of robot validation, rather than organizational ownership.** Software doesn't go from developer desktop to production. Robots shouldn't be any different. Validate identity, security, access and architecture through COE review, and validate functionality, implementation and resilience using automated software test harnesses.
- **Plan for the implementation of RPA as an operational platform.** The "Day 2" aspects of RPA are just as important as the robots themselves. Evaluate vendors on management capabilities, and plan for infrastructure and operations to manage this system.
- **Establish business partners to jointly manage the RPA roadmap through a COE.** Business and IT should work together to establish and evolve the set of criteria that mitigates risk by prioritizing processes that are well-suited for the teams' ability to create and validate a robot.
- **Apply established, proven software development life cycle techniques to robots.** Managing robots as software releases takes advantage of the lessons learned through decades of producing reliable software. Processes used for other software projects will make robots easier to support and troubleshoot and will help ensure uninterrupted operation.
- **Pursue opportunities to expand potential applications for robots by taking advantage of both embedded and external AI capabilities.** Automation is a natural fit for AI. Expect RPA vendors to incorporate capabilities that can address greater complexity with greater resilience. At the same time, integrating RPA with existing and emerging AI systems can create very capable systems that can deliver immediate results.

Conclusion

RPA augments and extends the reach of automation within the enterprise. Establishing an RPA platform requires a holistic approach that addresses operational issues well beyond authoring the robot. Once the platform is established, process selection plays a critical role in delivering repeatable success with the RPA platform. Technical professionals should prepare to play a critical, permanent role in the validation, management and operation of an RPA platform. The evolution of AI technologies, both internal and external to the RPA platform, will continue to expand the set of use cases that can be automated by RPA.

Gartner Recommended Reading

Some documents may not be available as part of your current Gartner subscription.

"Market Guide for Robotic Process Automation Software"

“Robotic Process Automation: 8 Guidelines for Effective Results”

“Peer Lessons Learned: Implementing Robotic Process Automation”

“IGA, RPA, and Managing Software Robot Identities”

GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp