David Chappell

Introducing Blue Prism

Robotic Process Automation for the Enterprise

Sponsored by Blue Prism

Contents

Robotic Process Automation with Blue Prism

Doing more with software is the essence of digital transformation. An important part of this transformation is automating business processes, using software rather than people wherever possible. *Robotic process automation (RPA)* is an increasingly popular approach to doing this.

RPA can have a transformative impact on organizations, bringing lower costs, increased reliability, and faster process execution. Done well, it can also let IT and business people work together to implement and change automated processes quickly. This paper introduces RPA, then describes how it's provided by Blue Prism, a leading vendor in this market. The goal is to explain what RPA is and how Blue Prism supports it.

What is Robotic Process Automation?

Even in our digital era, many business processes are still carried out by people. Yet these processes commonly rely on one or more applications, with human beings providing the driving intelligence. For example, think about a back office that handles customer orders. Each order might require getting a customer's name, then looking up and validating the customer's shipping address. Once this information is available, the next steps might be to calculate the shipping cost and place the order. In a typical back office environment, people execute this process, often relying on multiple applications to complete it.

But why? Just as more and more manufacturing processes are now done by robots, why can't more and more business processes be done by software robots? The answer is that with RPA, they can. Put simply, RPA means using software rather than people to carry out business processes that rely on applications. Figure 1 illustrates this idea.

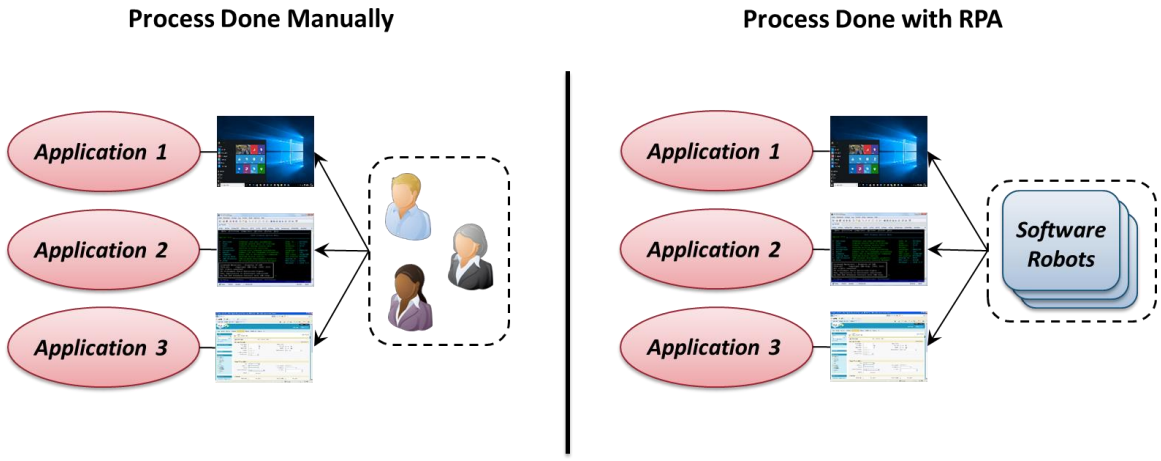

Figure 1: Robotic process automation allows using software robots rather than people to drive business processes.

As the figure shows, RPA allows replacing people with software robots. While this kind of automation can be useful when a process uses just one application, it's especially valuable when a process relies on multiple applications, as in the example shown here. Rather than integrating the applications through whatever application programming

interfaces (APIs) they might expose, RPA instead accesses each one through its user interface, just as a person would.

For the right kinds of processes, the economics of a robotic workforce are appealing. In fact, RPA can sometimes bring enormous benefits, especially for large enterprises. Think about it: Software robots can work every day around the clock, and they aren't tempted to take shortcuts. If your workload increases, you don't have to hire, train, and deploy new people; you just run more robots. If your workload decreases, you don't need to lay anybody off; you just shut down some robots.

Blue Prism, a UK-based technology company, provides software for RPA. (In fact, they coined this now industry-standard term.) This paper describes Blue Prism's offering, which is also called Blue Prism. Before diving into the technology, though, it's worth taking a closer look at why organizations choose to use RPA.

The Value of Robotic Process Automation

Automating processes using RPA has a lot going for it. Among its attractions are the following:

- Automating a process using RPA is typically simpler than using API-based integration. It also requires less technical knowledge and can be done faster.
- Automating a process using RPA is usually less expensive than using API-based automation. Because of this, you can automate more business processes for less money. And while API-based automation is typically used only for very high-value processes, RPA can provide a return on investment even for processes with lower business value. For example, maybe an application's transaction volumes don't justify the effort and expense of API-based automation. Using a lower-cost RPA approach can make this automation viable.
- Once a structured development process is in place, RPA can let business people create their own software robots. This frees the business from total reliance on IT for process automation. Just as important, RPA can let business people change their automated business processes without making every change request go through the IT bottleneck.
- Software robots make processes more accurate. Unlike people, they don't get tired and make mistakes. They just execute the defined process precisely over and over and over. RPA can also improve data quality, since input errors made by people go away.
- Applications accessed via RPA needn't expose any APIs. Instead, software robots rely on the same user interfaces that people use. This also lowers the risk of automating a process, since connecting with applications through APIs can provide more direct—and potentially more damaging—access to their functionality.
- In many applications, including modern web applications and older client/server solutions, important parts of the business logic are implemented in the user interface. On the web, for example, data validation is frequently done in JavaScript running in the browser. RPA takes advantage of this, something that's harder to do with API-based process automation.
- RPA allows automatically logging what would otherwise be manual operations in a business process. These logs are then available for audit, providing a written record of the process. This can be useful for things such as determining why a process raised an exception and meeting regulatory requirements.

As this list suggests, RPA is often a good choice for automating business processes. It's worth noting, though, that API-based automation can sometimes yield better performance. For example, suppose the information required for a single operation is spread across several different web pages or screens. While API-based automation might be able to access that information in one chunk, using RPA might mean stepping through all of the required pages or screens (although RPA technologies can also potentially access these APIs).

Like every technology, RPA has pros and cons. It's the right choice for some scenarios, but not for everything. Here are some of the characteristics of a process automation project for which this approach is a good fit:

- Getting a solution running quickly and inexpensively is important.
- The business process doesn't need to handle extremely high transaction volumes.
- The business process changes frequently, and business people need to be able to make those changes themselves.
- The business value of automating a process isn't high enough to justify the investment required for API-based integration.

For example, think about automating a business process that spans a range of different applications from different eras, including recently built web apps, client/server software, and 25-year-old mainframe software. If the goal is to create an automated process that handles tens of thousands of transactions a day, if the process doesn't change often, if all of these applications expose the right APIs, and if the time and money are available, API-based process automation is probably the best choice. But if the automated process is needed quickly at lower cost, if it might change frequently, if it need handle only a few thousand transactions a day, or if the right APIs aren't available, using RPA is probably a better approach.

An Overview of Blue Prism

Blue Prism is a set of tools, libraries, and runtime environments for RPA. Figure 2 illustrates the product's main components.

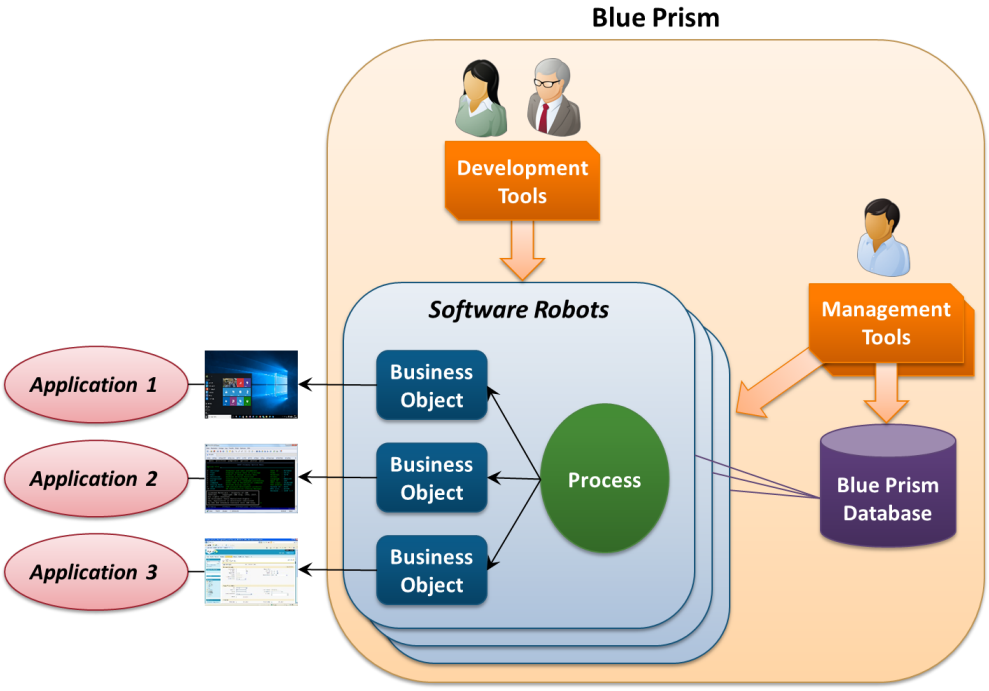

Figure 2: Blue Prism lets business analysts and developers create software robots using business objects and processes, then manage those robots.

Each software robot has two main parts: one or more *business objects* that interact with the user interfaces of the applications this robot uses, and a *process* containing the logic that drives the robot.

Blue Prism has built-in support for connecting business objects to various kinds of application user interfaces, including browser-based HTML interfaces, Windows interfaces, mainframe applications accessed via terminals, and interfaces built using Java. Each business object implements a particular set of actions against an application's user interface. For example, a single business object might be capable of logging in to an application, entering a customer name into a particular screen, retrieving a result, then logging off.

A developer or business analyst uses a Blue Prism development tool called *Object Studio* to create these objects graphically—writing code isn't required. The creator of this robot also uses another Blue Prism development tool, *Process Studio,* to graphically define the steps in the robot's process. Each step in a process invokes actions in business objects to interact with an application, and once again, writing code isn't required. In a very real sense, a process acts like a human user accessing each application to carry out the business process.

To store business objects, processes, and other information, the product provides the SQL Server-based *Blue Prism database*. IT and business people can use Blue Prism's management tools to schedule when robots run, view information about running robots, and more. These tools also allow configuring users, viewing audit logs, and performing other management tasks. And to make life simpler for the people using the product, all of Blue Prism's client tools are provided as tabs in a single application called the *Interactive Client (IC).*

The Blue Prism technology is intended to be used by IT people and business people working together. IT must be involved originally to set things up and configure deployments. Without this, robots can't run in a scalable, reliable,

and secure way. Once this is done, business people can create and modify automated processes largely on their own. This approach, involving both IT and business, lets an organization be sure that its software robots meet compliance and regulatory requirements while still letting the robots be built and updated quickly. To distinguish it from simpler script-based technologies, this style of process automation is sometimes called *enterprise RPA*.

Automating business processes through RPA isn't hard to understand. Still, to see how these pieces work together, it's useful to walk through a typical scenario. The next section does this, showing how the various parts of Blue Prism are used.

A Blue Prism Scenario

Imagine a simple business process carried out by a call center worker to determine a customer's shipping cost and place an order. This process requires the worker to interact with three different applications, entering information into each one and getting information in return. Figure 3 shows how this looks.

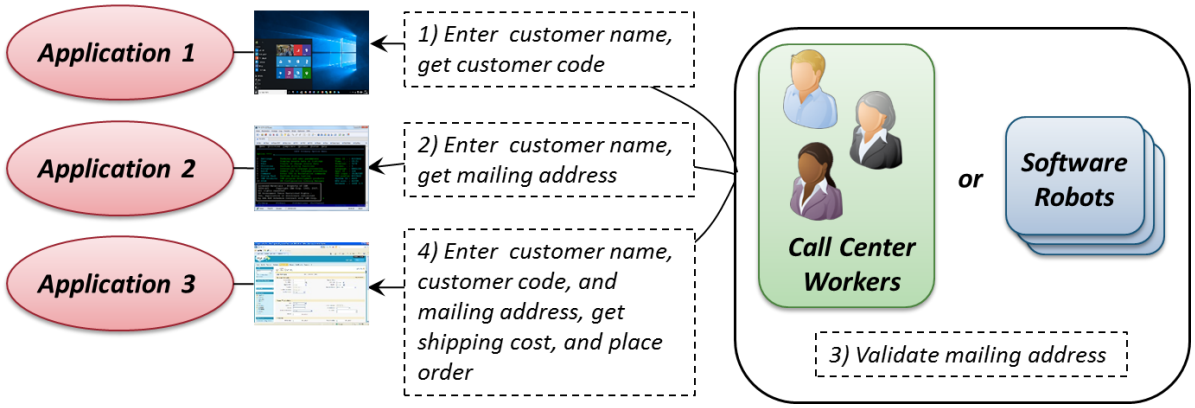

Figure 3: In this example process, call center workers or software robots interact with three different applications to place an order.

As the figure shows, the process first requires the worker to enter the customer's name into a Windows application, then get back a code indicating how much business the firm gets from this customer (step 1). The process next requires the worker to enter the customer's name into a mainframe application, retrieving the customer's mailing address (step 2). Next, the process requires the worker to validate this address, making sure that it contains all required fields (step 3). Finally, the worker enters all of this—the customer name, code, and mailing address—into a Web-based application that determines the shipping cost and places the customer's order (step 4).

To automate this process with Blue Prism, a developer or business analyst uses Object Studio to create a business object for each of the three applications. Each business object implements the same actions that a call center worker performs on this application. For example, the business object that interacts with the Windows application might contain a login action, an action that enters the customer's name, another action that retrieves the customer code, and a logout action.

Once these objects have been created, a developer or business analyst uses Process Studio to graphically define the steps in the process. Each step can invoke actions in one or more business object. In this example, the process

has four main parts: entering data into the Windows application and getting a result, entering data into the mainframe application and getting that result, validating the mailing address (which can be done by a business object), and entering all of this data into the Web-based application.

Once the business objects and the process have been created, the robot can be executed. (Note that with Blue Prism, robots run on servers in the organization's datacenter rather than on user desktops.) An IT or business person can use Blue Prism's management tool to schedule when robots run, then monitor their execution. And to help create applications safely, Blue Prism defines a methodology for designing, building, and deploying a new automated process. This methodology includes typical software development practices such as using distinct environments for development and production, with a business sign-off to move a new robot into production.

If the business process needs to change, a business analyst or developer uses Process Studio to modify it. If the user interface to one of the applications changes significantly, or if supporting the changed business process requires implementing a new action on one of the applications, the analyst or developer uses Object Studio to change the application's business object as needed.

The main thing to understand is that a robot built using objects and processes can execute the same steps as a person. This includes not only the interactions with applications, but also steps the person might perform manually, such as validating the address. This is what makes it possible for Blue Prism robots to take the place of people in automating a business process.

Why Not Use a Record Button?

If all you want to do is replicate a user's interactions with applications, why bother creating business objects and processes? Why not provide a simple Record button that lets you record an example user's behavior in a script, then run that script over and over? Creating a script can be fast and easy, and some automation products do this.

Blue Prism isn't one of them, however, for several reasons. For one thing, making a script run in a reliable, scalable, secure fashion can be challenging. Scripts are commonly designed to be run by a user on a desktop machine, while scalable software robots must run unattended on servers. Blue Prism's goal is to create a managed workforce of robots, not to help individuals run scripts on their own.

Also, once a library of Blue Prism business objects exists, those objects can be reused across multiple processes. And because only business objects talk directly to applications, changes in application user interfaces typically don't require changes in processes; at most, only the affected business object needs to be updated. This is usually simpler than working out which scripts must change, then updating each one.

Scripting with a record button can be simple. But it's not the best approach for enterprise RPA.

The Technology of Blue Prism

Having a big-picture view of how Blue Prism works is a good place to start. But really understanding the product requires more. This section takes a deeper look at the Blue Prism technology, describing each of its main components.

Creating Blue Prism Applications: Business Objects

As just described, a Blue Prism robot relies on business objects to interact with applications. Understanding what a business object is first requires understanding the components each one contains, then seeing how the Blue Prism development tools are used to create those components. This section looks at both.

What is a Business Object?

A business object acts as an adapter to the user interface of a specific application. To accomplish this, each object has two main parts:

- An *application model* that exposes the elements of this application's user interface.
- One or more *actions*, each of which implements all or part of an operation that the business object can perform.

Figure 4 shows how these two parts relate to one other and to the world outside the business object.

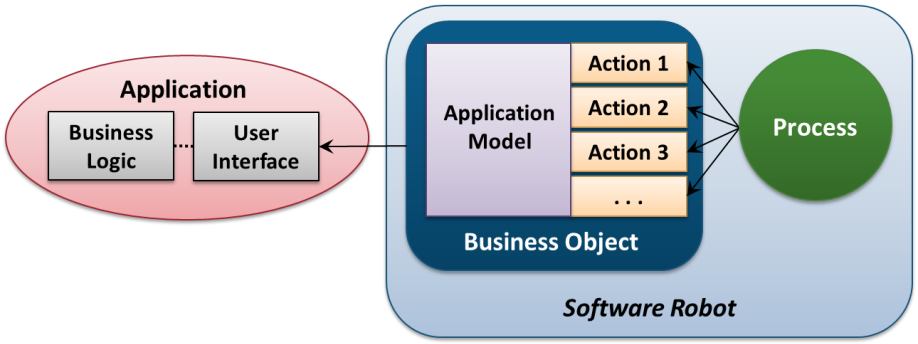

Figure 4: A business object includes an application model that interacts with a particular application's user interface, together with actions that implement operations on that application.

Understanding business objects requires looking at both of these components. What follows walks through each one.

Creating an Application Model

To interact with an application, a robot needs to know about that application's user interface. An application model represents this knowledge, describing how the business object connects to the application. Since application interfaces use a variety of technologies, Blue Prism provides standard libraries that support various approaches, called *modes*, for doing this. The options include:

- An HTML mode, which allows working with HTML, JavaScript, and other common components of a browser interface.
- A Windows mode, allowing access to Windows applications built using the .NET Framework, the Microsoft Foundation Class (MFC) library, Visual Basic 6, PowerBuilder, and other Windows-based user interface technologies.
- A Java mode for working with user interfaces created using Swing, Java applets, Oracle Forms, and other technologies based on the Java Virtual Machine.
- A mainframe mode, which allows access to mainframe applications through terminal emulators from various vendors, including Attachmate, Micro Focus, IBM, and others.
- A Citrix mode, allowing access to applications that provide their user interfaces via technologies such as Citrix XenApp and Microsoft Remote Desktop Services.

To select a particular mode for a newly created business object, a developer or business analyst uses a wizard that's part of Object Studio. It's also possible to use more than one mode in a single business object. If an HTML page contains a Java applet, for example, both the HTML mode and the Java mode can be used.

Whatever modes a business object uses, Blue Prism models each screen in an application's user interface as a set of *elements*. Each element represents some aspect of what's on this screen. For example, the elements in a screen accessed via the HTML interface might include a field for entering data, a button to click, and a hyperlink. The application model defines the elements of each screen for a particular application, then makes them available to the actions in this business object. Just as a traditional API can expose an application's business logic to other code, an application model exposes an application's user interface to the logic contained in actions.

To create application models, a business analyst or a developer uses the Application Modeler, part of Object Studio. The screen shot in Figure 5 gives a sense of how this tool is used.

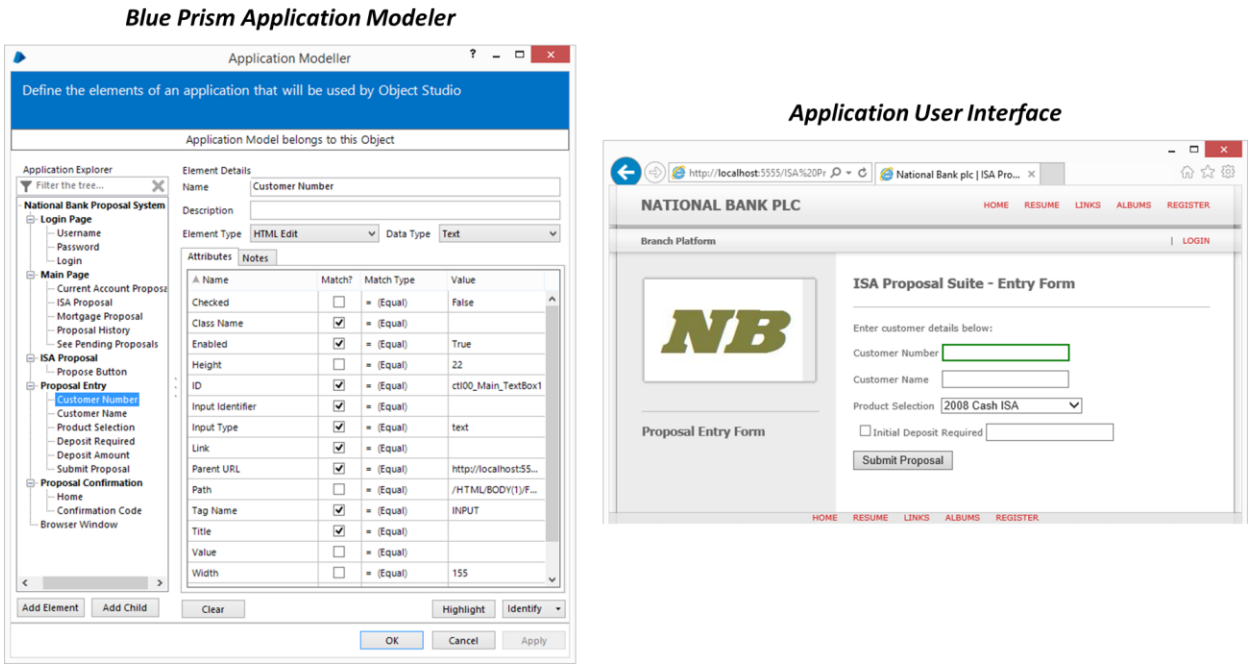

Figure 5: The Application Modeler in Object Studio helps its user create an application model.

To create an application model, a business analyst or developer must determine which screens in an application's user interface this business object will use. Once she's done this, she then uses the Application Modeler to step through each screen, clicking on the elements that should be made available to the business object's actions. In Figure 5, for example, which is for an HTML interface, the Application Explorer pane in the Application Modeler window on the left contains a list of screens and elements for the application shown on the right. As this pane illustrates, the Login screen contains elements such as fields for Username and Password, together with a button for Login. Similarly, the Proposal Entry screen has elements such as Customer Number, Customer Name, and Product Selection. This model of the application's user interface was created by stepping through those screens with the Application Modeler.

The window on the right in Figure 5 shows the actual user interface for the application's Proposal Entry screen. Notice that the Customer Number element is highlighted. Doing this causes the Application Modeler to bring up information about that element, such as its type and attributes. By checking one or more boxes in the Match column, the developer indicates which of these attributes the application model should use to identify this element at run time. Because Blue Prism uses attributes to identify elements, the application model can sometimes remain unchanged even if parts of the application's user interface are modified. For example, the ID attribute of an HTML element remains the same even if a web page's layout changes, so an application model might be able to handle this change without modification.

Once the application model is complete, the required elements of each screen in this application are now available to be used. Those elements are accessed by actions, and so the next thing to look at is what actions are and how they're created.

Creating Actions

As their name suggests, actions implement the logic of a business object. As Figure 6 shows, each action contains a set of *stages* along with *data items* holding information used by that action.

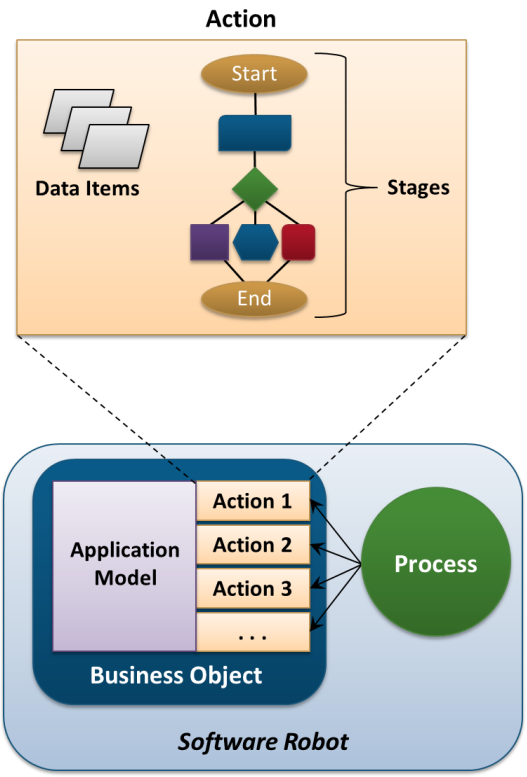

Figure 6: Each action in a business object can contain stages and data items.

An action can be published, which lets it be called by processes and potentially other software. An action can also remain private, making it available only to other actions in this business object. Whichever option is chosen, the action begins with a Start stage and finishes with an End stage, as Figure 6 shows. In between appear whatever stages are required to carry out the tasks this action performs. Some of the most commonly used stages in business objects include the following:

- Read: gets data from a user interface element and stores it in a data item.
- Write: puts data from a data item to a user interface element.
- Navigate: opens menus, clicks buttons, and carries out other tasks required to navigate through an application's screens.
- Wait: pauses execution until a condition is met in the application. For example, a business object might need to wait until a particular window has appeared before proceeding or wait for the entire application to finish loading.
- Link: creates a link between stages in an action.

- Decision: acts like an If statement, branching an action's control flow based on the value of one or more data items.
- Choice: acts like a switch statement, selecting one of several control flow options based on the value of one or more data items.
- Calculation: contains an expression that performs a calculation on data items, such as adding two numbers, extracting a sub-string, or converting a date from one format to another.
- Loop: iterates through a collection of data.
- Code: contains arbitrary code written in languages such as C# and Visual Basic. This might be used to carry out complex data manipulation, for example, or to access an application-specific interface, such as a COM object in a Windows interface. Code stages can also be used to access database interfaces, giving a page direct access to application data.
- Exception: raises an exception in the execution of an action.
- Recovery: begins a block for handling exceptions.
- Resume: ends a block for handling exceptions.
- Alert: sends a notification to one or more subscribers via email indicating that something has occurred. A business object might send an alert whenever an exception is raised, for example.

Creating an action—combining a set of stages—is like creating a method or subroutine. Rather than use code, however, the person creating this action does it graphically using Object Studio. He can also define properties for each stage, such as a name. As mentioned earlier, the goal is to allow business analysts to define this logic, not just developers. Figure 7 shows a simple example of creating a login page using Object Studio.

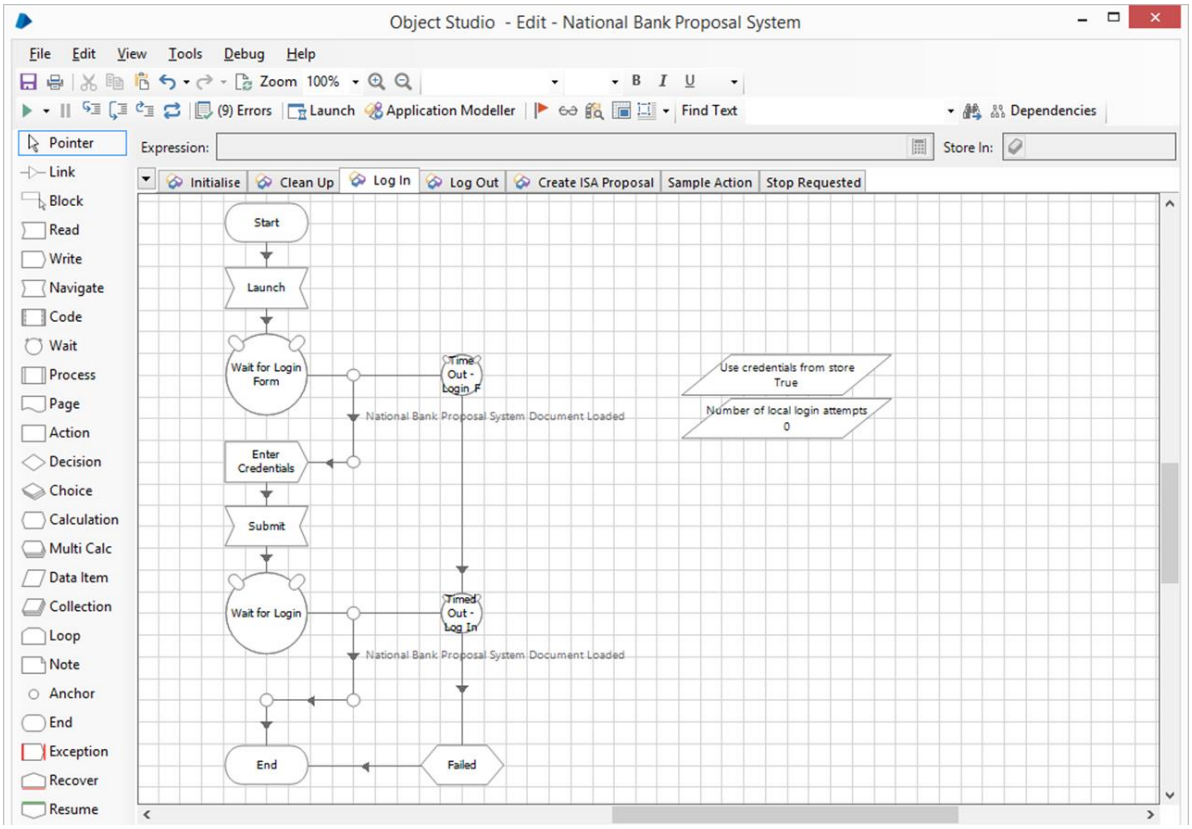

Figure 7: A business analyst or developer uses Object Studio to define the stages in an action.

As the figure shows, each action in this business object appears as a tab in the Object Studio design surface. Here, the analyst has selected the Log In action, and so the tool shows the data items and stages in this action. The data items, shown on the right, include a Boolean indicating that this action should use login credentials contained in the Blue Prism database and a counter to keep track of how many login attempts this action makes.

The action has a Start and End stage, with the stages required to perform the login in between. Immediately after the Start stage appears a Navigate stage named “Launch” that starts the application this business object works with. The Wait stage, shown as a circle, waits for the login screen to load, with a time-out-based error handler in case something goes wrong. Next comes a Write stage, named “Enter Credentials”, that enters the user name and password. Another Navigate stage appears next, this time named “Submit”, that clicks the screen’s Login button. The action then executes another Wait stage, waiting for a screen that indicates whether the login has succeeded or failed. If it fails, an error handler is executed. If it succeeds, the action ends normally—the login was a success.

Once business objects are created, they can be accessed directly by processes, as described in the next section. They can also be accessed by other software, however. Each business object can expose a WSDL interface, with its actions invoked via SOAP. This can be useful for an organization pursuing a SOA strategy, for example, or in other situations.

The way business objects work makes basic testing straightforward. An analyst creating an action can open an Object Studio window for the business object he’s creating and another window for the application this business

object works against. When he clicks on a stage in the business object, the result of that stage, such as writing a value into a field, shows up immediately in the application's user interface screen. This allows validating the basic functionality of stages without waiting for recompilation. Object Studio also contains a built-in debugger, allowing things such as stepping through an action's logic one stage at a time, setting breakpoints, and examining the contents of data items.

Creating Blue Prism Applications: Processes

The purpose of a Blue Prism application is to automate all or part of a business process. To allow this, Blue Prism provides Process Studio, a tool for creating processes. Before looking at this tool, however, we need to look more closely at Blue Prism processes.

What is a Process?

Since Blue Prism accesses applications through their user interfaces, a process acts much like a human user—it implements a software robot's logic. And just as a person might interact with several applications to carry out a series of steps, a process can invoke actions in several business objects to carry out those same steps.

In some ways, a process is similar to a business object. Every process is defined using one or more *pages*, each of which is similar to a business object action, and each page contains some number of stages and data items. There are important differences between business object actions and process pages, however. For example, in a business object, any published action can be called at any time; the object exposes a set of actions that can be invoked in any order. A process, however, always begins at its Main page, and the pages it contains always execute in a defined order. (This shouldn't be surprising—it's implementing a business process, not a set of independent actions.) Figure 8 illustrates this idea.

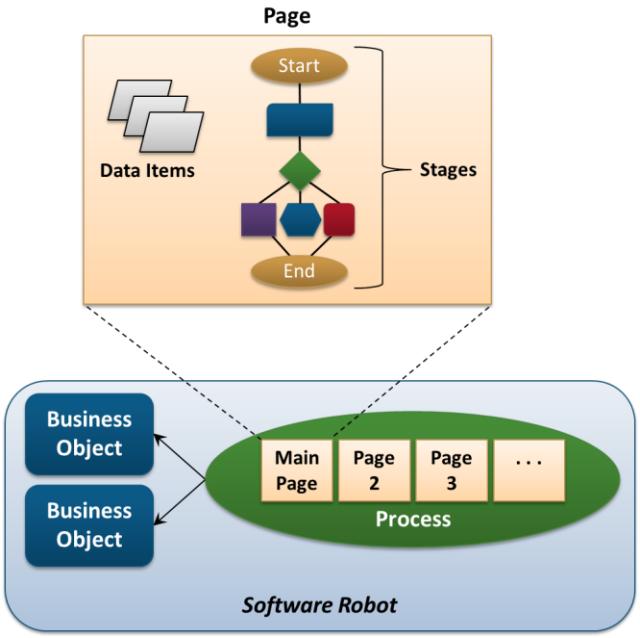

Figure 8: A process contains one or more pages and always begins execution at its main page.

As the figure suggests, execution of a process always begins with the Start stage of its Main page. A simple process might be defined entirely in this Main page, but more likely, the process will be spread across several pages. For example, a group of stages that are used frequently might be defined on their own page, letting them be invoked much like a subroutine. Throughout its execution, stages in the process invoke actions in business objects as needed.

The stages allowed in a process are similar, but not identical, to those used in a business object. The control flow stages, such as Link, Decision, Choice, and Loop are all available, as is the Calculation stage. Similarly, the stages for raising and handling exceptions—Exception, Recovery, and Resume—are available.

Some stages are more specific to processes, however. The most important of these is Action, which allows calling a published action in a business object. This stage can also invoke a SOAP-based service or call a COM object directly. If necessary, a process can also call another process through the Process stage.

Just as some stages appear only in processes, some business object stages aren't allowed in processes. A process can't contain Read, Write, or Navigate, for example, since all of these are intended for working directly with an application's user interface. Processes rely on business objects to do this—they don't do it themselves. A process also can't contain Code stages, which means that low-level customizations are relegated to business objects.

As Figure 8 shows, a page in a process can contain data items. One way to use these is to hold parameter values that are passed into a process when it's started. For example, a particular process instance might be assigned 100 accounts to work with, with that number passed in as a parameter.

This raises a more general question: How do processes get the input they need to function? Suppose a process must work its way through a list of customer accounts, for example, using the right information for each customer. Where does this information come from? One option is to pass it in as parameters when a process instance is started. Another is to use a Blue Prism *work queue*, which can be read by multiple process instances (and are described in more detail later). It's also possible for processes to get their input from external software, such as an existing customer workflow system.

Creating a Process

To create a process, a business analyst or developer uses Process Studio. Just as processes are similar to business objects, Process Studio is similar to Object Studio, as Figure 9 shows.

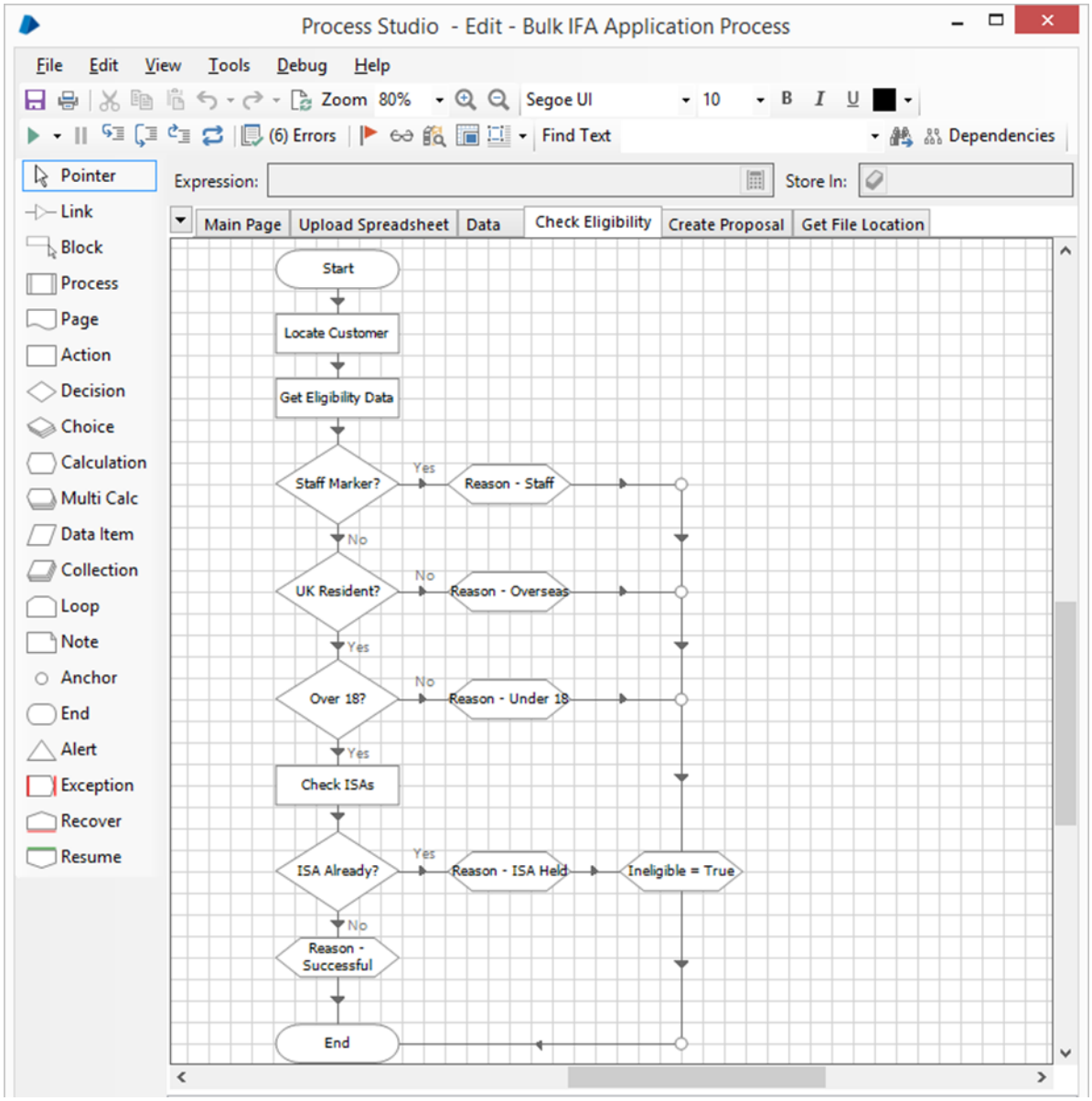

Figure 9: Using Process Studio, a business analyst or developer can arrange stages to define the logic of a process.

A process defined in Process Studio looks much like a traditional flowchart. As in Object Studio, each page in a process has its own tab, letting the analyst divide the process into logical groupings. Also like Object Studio, Process Studio allows its user to create logic graphically by assembling stages on a design surface. The tool includes a built-in debugger that allows stepping through a process, setting breakpoints, examining data items, and more.

One of the challenges in working with business processes is changing them safely. To help with this, Process Studio keeps track of the previous version of a process being edited. The user of the tool is then able to switch to a view that shows both the old and new versions side-by-side, letting her see exactly what's different. Blue Prism also provides a process history tool that allows tracking revisions to processes over time.

Another challenge in working with business processes is keeping track of what actually happens when they execute. For example, think of a process that takes different paths based on whether, say, a hotel room is available. Using Process Studio, an analyst can get a real-time display of what percentage of requests take the success path and of how many fail for a particular kind of process. Information about the number and type of exceptions a process generates is also accessible. This immediate feedback can be helpful in making an automated business process run more effectively.

Deploying Blue Prism Applications

A process acts like a human user who thinks and types very fast. Today, each human user typically has his own desktop Windows machine. For a process to replace that user, each process must also have its own Windows desktop. Fortunately, this doesn't imply that using Blue Prism requires a room full of dedicated PCs. Instead, each process instance typically runs in its own Windows virtual machine (VM).

For example, recall the scenario described earlier in which a business process required accessing a Windows application, an IBM mainframe, and a Web application. The software robot used to automate this business process—a Blue Prism process and the three business objects it uses—runs inside its own VM. To each of the applications, this robot will look like just another user. Figure 10 shows how this looks.

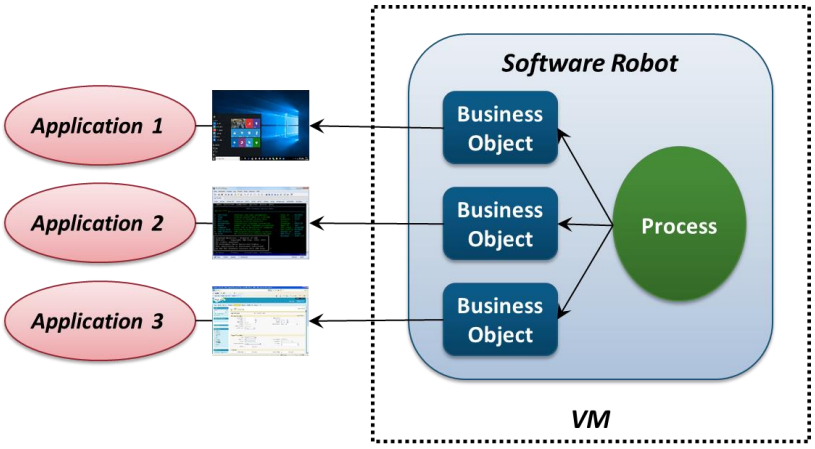

Figure 10: Each instance of a robot typically executes inside its own VM.

Enterprise business applications usually have many simultaneous users. To accomplish this with Blue Prism means deploying many VMs, each running an instance of a robot. A single physical server will typically run a number of these VMs, as Figure 11 illustrates.

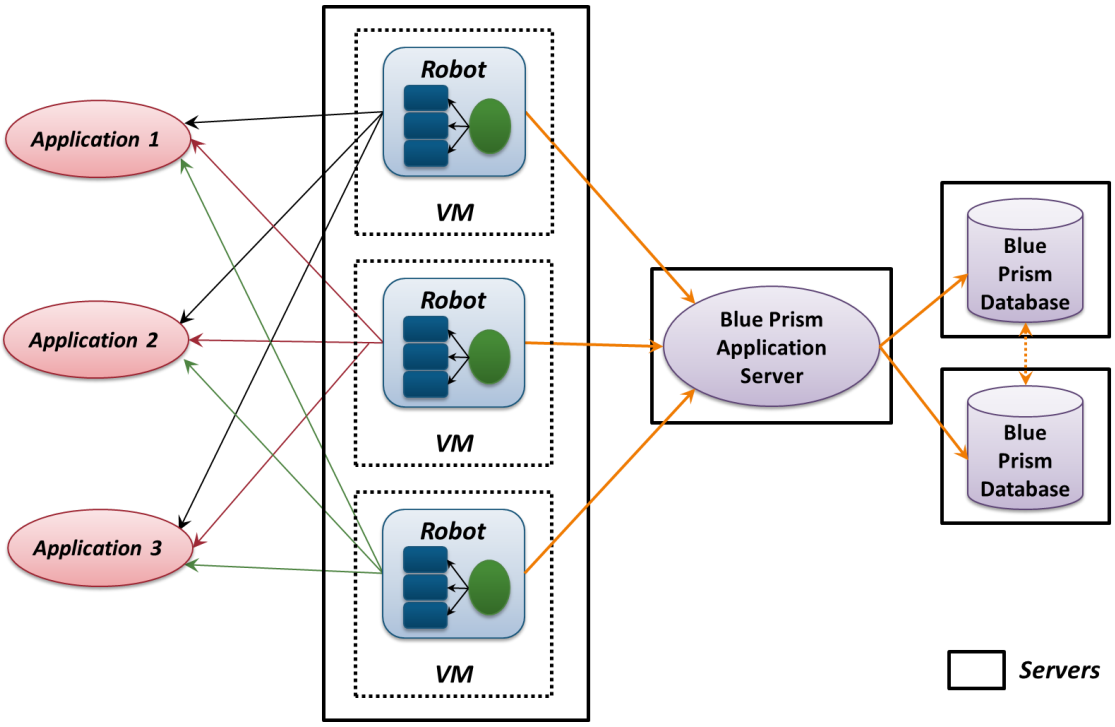

Figure 11: Multiple VMs, each running a single Blue Prism robot and sharing a Blue Prism database, can run on a single physical machine.

As the figure shows, a Blue Prism application server acts as a common control point for all of these robots. All of them also share a Blue Prism database. This database provides a shared repository for the environment, storing the code for each robot, the work queues those robots use, audit logs, and more. It's implemented using SQL Server with Always On Availability Groups, providing a redundant, reliable store for this enterprise RPA environment.

Work queues commonly play an important role in Blue Prism environments. Items to be processed, such as customer orders, can be put into a queue, then processed simultaneously by many different robots. Each robot executes the same Blue Prism process, but works on different data, e.g., the customer order it's pulled from a work queue. This makes scaling up or down simple; the person controlling the robotic workforce can just increase or decrease the number of robots that read from a particular work queue. It also helps keep robots busy, since each one can work at its own pace, pulling a new item from the queue whenever it's ready.

Managing Blue Prism Applications

Just like manual processes, automated processes—software robots—need to be controlled and managed. To allow this, Blue Prism provides two tools, both implemented as tabs in the Interactive Client: Control and System.

The purpose of the Control tab is to let business analysts and IT staff work with robots. The tool lets them perform tasks such as:

- Starting and stopping robot instances. (It's also possible to do this from the command line or with external software—using the Control tab isn't required.)

- Viewing the Session Log produced by each robot, either while the robot is running or after it's completed. This log can record when each stage in a process is executed, what data was used, and more.
- Creating process schedules. For example, a business analyst might define a schedule that runs process *X* at 9 am every Friday morning. If *X* completes successfully, the schedule then runs process *Y*. If *X* doesn't complete successfully, the schedule runs process *Z*.
- Viewing work queues, examining things such as how far robots have gotten in a particular queue and what exceptions have occurred.

While the Control tab allows working with running robots, the System tab focuses on configuring the Blue Prism environment. For example, an administrator can use it to add a new Blue Prism user, then define the roles that user can operate in. Based on these roles, a user might be allowed to create new business objects and processes or perhaps just run existing robots. The System tab can also be used to install and manage robots and to carry out a variety of other management functions. As with the Control tab, the information the System tab uses to do all of this is stored in the Blue Prism database.

Blue Prism also provides the ability to create and use dashboards. Each dashboard is a collection of tiles, and exactly which tiles appear on a dashboard is configurable. The kinds of information a dashboard can display can be grouped into three categories:

- Information that's useful for controllers, the people who are looking after a robotic workforce, such as data about running robots and the work queues they're using.
- Information aimed at the managers of those controllers, including aggregates of the more detailed information used by controllers themselves.
- Information intended for the business owner of these robots, such as tracking the percentage of work that's being handled by robots rather than people.

Different people can create different dashboards, based on their needs. Figure 12 shows a simple example.

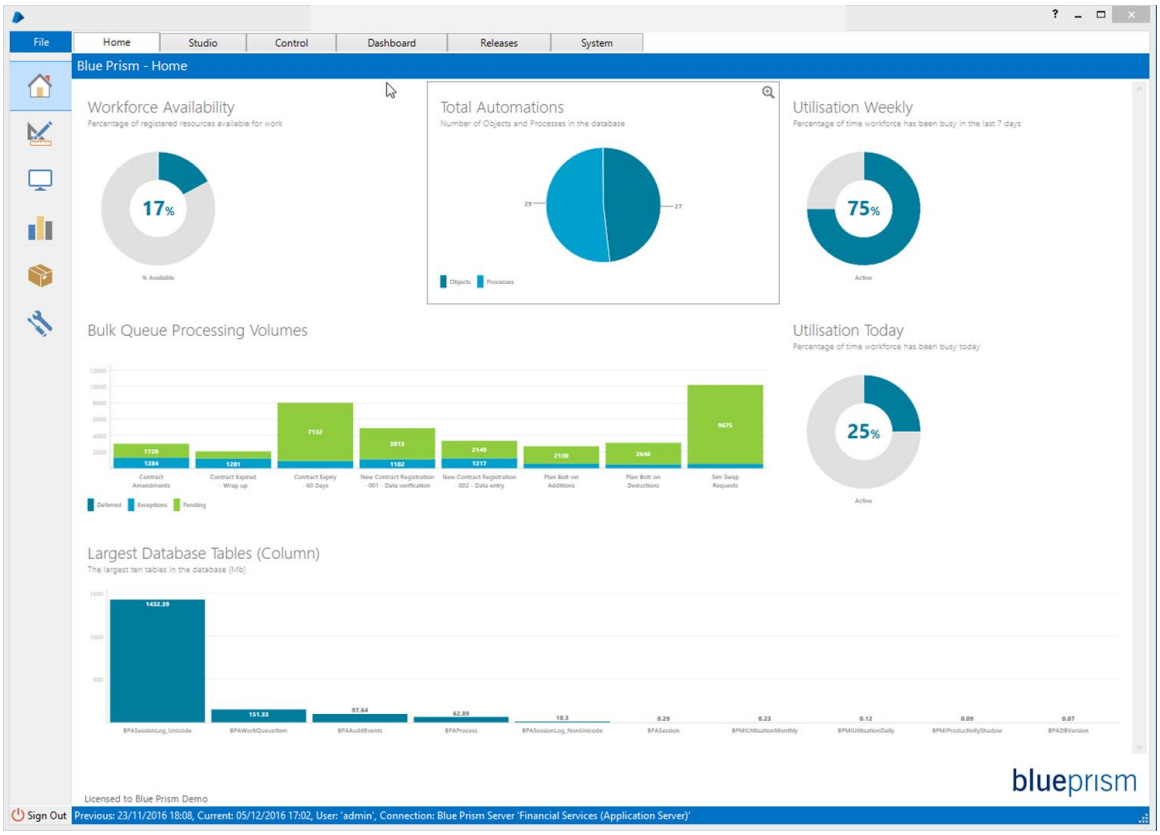

Figure 12: A Blue Prism dashboard provides a convenient way to display information about a robotic workforce.

This example includes a mix of these different information types, including robot utilization, work queue processing volumes, and details about the Blue Prism database. The goal is to give different people easy access to the information they need.

Securing Blue Prism Applications

Any application intended for enterprise deployment must squarely address security, and Blue Prism is no exception. One important foundation for security in the product is role-based access control. Once a Blue Prism administrator has defined what roles a user can act in, the system will allow that user to perform only the actions allowed by these roles.

Blue Prism also keeps track of significant changes to its environment. Using the System tab, an administrator can examine records of things such as when a user logs into Blue Prism, when that user starts or deletes a robot, when that user changes her password, and more.

Another issue, especially for an RPA technology, is safely managing the user names and passwords used to log in to applications. To do this, Blue Prism supports Active Directory, which is how most Blue Prism customers handle identity. The product also provides its own encrypted credentials database, however, controlled by a Blue Prism administrator, that can be used if desired.

Conclusions

Automating business processes with RPA can have substantial business value. In situations where this automation must be done quickly and at low cost, where changes are frequent, or where business people must be able to control the work themselves, relying on robotic process automation can be a good solution.

Blue Prism is designed for these situations. The product's primary goals include the following:

- Helping organizations automate business processes more quickly and for less money. An important corollary of this is the ability to apply process automation to lower-value scenarios that might not be worth automating using more traditional approaches.
- Providing tools that let business analysts play a significant role in creating and changing automated processes. The intent is both to help organizations do these things in less time and to reduce the need for IT involvement in the development process.
- Supporting enterprise RPA, letting organizations create a robotic workforce—running in datacenters, not desktops—that's scalable, manageable, and secure.

No technology is right for every situation, and robotic process automation is no exception. Yet applied to the right problems, this approach can be a good choice, as Blue Prism shows. Why not let technology do everything it can?

About the Author

David Chappell is Principal of Chappell & Associates (http://www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.