

INTRODUCING BLUE PRISM

AUTOMATING BUSINESS PROCESSES WITH PRESENTATION INTEGRATION

DAVID CHAPPELL

MARCH 2010

SPONSORED BY BLUE PRISM

CONTENTS

Business Process Automation with Blue Prism
The Role of Presentation Integration3
An Overview of Blue Prism
A Blue Prism Scenario
The Technology of Blue Prism9
Creating Blue Prism Applications: Visual Business Objects9
What is a Visual Business Object?
Choosing a Connector
Creating an Application Control Interface11
Creating Pages
Creating Blue Prism Applications: Processes
What is a Blue Prism Process?
Creating a Process
Deploying Blue Prism Applications
Managing Blue Prism Applications 20
Securing Blue Prism Applications 21
The Blue Prism Methodology
Conclusions
About the Author

BUSINESS PROCESS AUTOMATION WITH BLUE PRISM

Every organization wants to lower its costs and keep up with change. Using technology effectively is an important part of doing both. Yet business needs can sometimes outpace an IT group's ability to satisfy those needs.

For example, automating a business process can make it faster, more reliable, and less expensive. But doing this typically requires integrating multiple applications, which can be time consuming and expensive. And once the integration is done, the process is often hard to modify. What's needed is a way to automate a business process quickly while still letting that process adapt to change.

One way to do this is through *presentation integration*. Once derisively known as "screen scraping", this technology has matured into an effective approach to interacting with applications. Because it's simpler than more traditional integration technologies, using presentation integration to automate a business process can be faster and less expensive. And because presentation integration can be used today on a wider scale than was possible with screen scraping, it can support enterprise processes. This makes it useful in a variety of situations, including cases where automating a process in the traditional way would be too expensive.

Blue Prism provides process automation based on presentation integration. By providing tools for business analysts, the product can also help people who aren't technology specialists create and modify automated processes. And by exposing application user interfaces through reusable services, it can help those applications fit into modern service-oriented environments. This paper describes Blue Prism, explaining how the technology works and how it attempts to reach these goals.

THE ROLE OF PRESENTATION INTEGRATION

Automating business processes requires connecting to the applications that support those processes. This can be done using either presentation integration or an alternative that's often called *business logic integration*. Figure 1 shows how each approach looks for a process that relies on three applications.

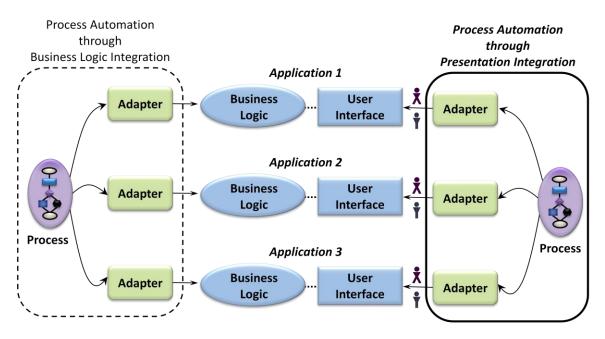


Figure 1: Process automation can be done by connecting to application business logic or by interacting with application user interfaces.

As the left side of the figure shows, business logic integration uses adapters that connect directly to each application's business logic. The automated process then interacts with these adapters to carry out the steps required. The right side of the figure shows the alternative, presentation integration. Here, adapters connect to the user interface of each application (how the application "presents" itself to users) rather than to its business logic. The automated process plays a role much like a human user, accessing each application as a person would.

Both approaches have value. Today, though, it's fair to say that business logic integration is more popular. Most mainstream integration products use this approach, as do typical business process management systems (BPMSs). In fact, the idea of service-oriented architecture (SOA) can be seen as a generalization of business logic integration.

Yet automating processes through presentation integration has a lot going for it. Among its attractions are the following:

- It can be fast and relatively easy to implement compared to business logic integration. No changes to the applications are required, and with the right tools, creating adapters for existing user interfaces can be straightforward.
- It can cost less than automating processes through business logic integration. To some degree, this is a corollary of the relative simplicity of presentation integration. It also means that presentation integration can be used with lower-value business processes. For example, perhaps an application's transaction volumes don't justify the effort and expense of business logic integration, or maybe the technical complexities of business logic integration exceed the business value of automating this part of the process. Because presentation integration is quicker and less expensive, it can make sense in situations like these.

- Because the implementation effort isn't huge, automated processes that use presentation integration can be relatively easy to change. This is important, as many business processes are quite dynamic. Automated processes that are difficult to change can be worse than manual processes, since they make a firm's operations more inflexible. With appropriate tools, it's even possible for business analysts to make the necessary modifications, freeing IT staff from dealing with many small change requests.
- Presentation integration can be used to access all kinds of applications, from modern Web solutions to legacy mainframe software. An application doesn't need to expose special integration interfaces. Think about a mainframe application, for example, that doesn't provide any interfaces for business logic integration. Presentation integration can still be used to access it, since this approach requires just a terminal interface. Or perhaps an application is accessible only via a remote access technology such as Citrix XenApp or Microsoft Remote Desktop Services. With the right support, presentation integration can also be used to access this kind of software. Even processes that depend on client/server applications with extensive client logic can be automated using this approach.
- 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. Presentation integration takes advantage of this, something that's harder to do with business logic integration.
- Presentation integration 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.
- Because process automation through presentation integration is relatively inexpensive and fast, it can be used to verify the value of automating a particular business process. If greater performance is needed, an organization might then decide to invest in a more expensive business logic integration project.

Presentation integration is often a good choice. Still, there are situations in which it's not the right solution. Among the challenges of this approach are the following:

- Performance isn't always great. For example, suppose the information required for a single operation is spread across several different Web pages or screens. While data or business logic integration might be able to access that information in one chunk, using presentation integration requires stepping through all of the required pages or screens.
- Only information exposed through an application's user interface is available. If the process being automated requires access to more than this, presentation integration isn't a good option.

Its limitations have led a number of people to dismiss presentation integration. In fact, the original notion of screen scraping truly was problematic. Like many technologies, this first version wasn't robust enough to meet enterprise requirements. Today, though, more modern approaches have appeared, along with better tools, improved management capabilities, and more effective methodologies. For example, a more modern object-oriented approach can avoid the security and maintenance challenges of customized per-

application scripts. Similarly, rather than requiring multiple desktop machines to scale a process that uses presentation integration, many virtual machines can now run on a single physical server.

The truth is that, like every technology, presentation integration has pros and cons. It's the right choice for some scenarios, but not for everything. As the lists above suggest, 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.
- The business value of automating a process isn't high enough to justify the investment required for business logic integration.

For example, think about automating a 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 business process that handles tens of thousands of transactions a day and doesn't change often, and if the time and money are available, business logic integration is probably the best choice. But if the automated process is needed quickly for as little cost as possible, if it will change frequently, and if it need handle only a few thousand transactions a day, presentation integration may well be a better approach.

Don't be confused, however. While there are situations that can be solved using either business logic integration or presentation integration, it's more accurate to think of these technologies as complements than as competitors. Depending on the business requirements, both have value. In spite of this, there aren't a lot of products available today that support this approach. One that does is Blue Prism, which provides a modern take on presentation integration. The next section gives an overview of the Blue Prism technology.

AN OVERVIEW OF BLUE PRISM

Blue Prism is a set of tools, libraries, and runtime environments for automating business processes through presentation integration. Figure 2 shows the product's main components.

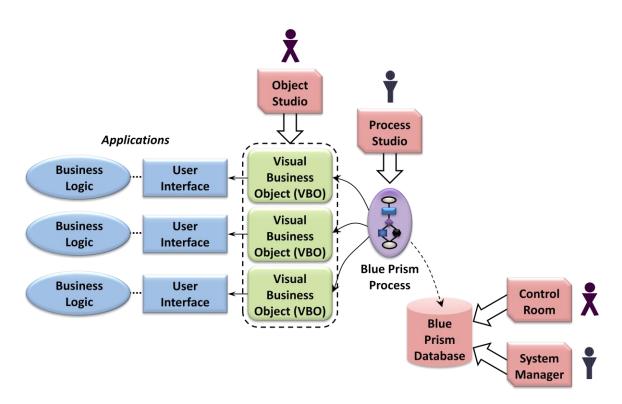


Figure 2: Blue Prism lets business analysts and developers create visual business objects, define processes that use those objects, track the execution of those processes, and more.

Blue Prism has built-in support for connecting to various kinds of application user interfaces, including browser-based HTML interfaces, Windows interfaces, mainframe applications accessed via terminals, and interfaces built using Java. Whatever the interface technology, the adapter used to connect to an application is called a *visual business object (VBO)*. Each VBO implements a particular set of operations against an application's user interface. For example, a VBO 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 Blue Prism's *Object Studio* to create these objects graphically—writing code isn't required.

Each VBO exposes its operations to a *Blue Prism process*. To create a Blue Prism process, a developer or business analyst uses Blue Prism's *Process Studio*. This tool lets its user graphically define the steps in the process, each of which invokes operations in VBOs to interact with an application. Once again, writing code isn't required. In a very real sense, a Blue Prism process acts like a human user accessing each application to carry out the business process.

To store VBOs, Blue Prism processes, and information about them, the product provides a SQL Serverbased database. IT and business people can use a tool called *Control Room to* start Blue Prism processes, view information about running processes, and more. Another tool, *System Manager*, allows configuring users, viewing audit logs, and performing other management tasks. Automating business processes through presentation integration 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 operator to determine a customer's shipping cost and place an order. This process requires the operator to interact with three different applications, entering information into each one and getting information in return. Figure 3 illustrates this scenario.

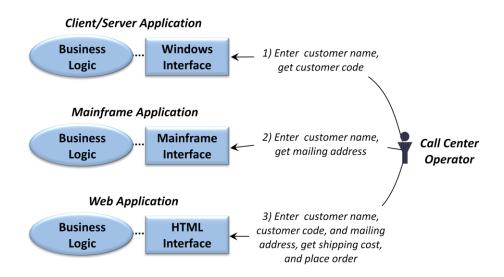


Figure 3: In this example process, a call center operator interacts with three different applications to place an order.

As the figure shows, the process first requires the operator to enter the customer's name into a client/server application with a Windows interface, then get back a code indicating how much business the firm gets from this customer (step 1). The process next requires the operator to enter the customer's name into an mainframe application, retrieving the customer's mailing address (step 2). Finally, the operator 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 3).

To automate this process with Blue Prism, a developer or business analyst uses Object Studio to create a visual business object for each of the three applications. To interact with the application, each VBO uses an appropriate interface: the Windows interface, the mainframe interface, and the HTML interface, respectively. Each VBO implements the same operations that a call center operator performs on this application. For example, the VBO that interacts with the client/server application might contain a login operation, an operation that enters the customer's name, another operation that retrieves the customer code, and a logout operation.

Once these objects have been created, a developer or business analyst uses Process Studio to define graphically the steps in the Blue Prism process. Each step can invoke operations in one or more VBOs. In this example, the process has three main parts: entering data into the client/server application and

getting a result, entering data into the mainframe application and getting that result, and entering these results into the Web-based application.

Once the VBOs and the Blue Prism process have been created, the complete solution can be executed. An IT or business person can use Control Room to start and stop instances of the process, monitor its execution, or perform other management tasks. To determine who's allowed to run the process, audit how people use the system, and more, a Blue Prism administrator uses System Manager. And to help create this application safely, Blue Prism defines a specific methodology for designing, building, and deploying a new automated process.

When the 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 operation on one of the applications, the analyst or developer uses Object Studio to change the application's VBO as needed.

It's worth noting what the clear separation between VBOs and Blue Prism processes implies. For one thing, the same VBO can potentially be reused by multiple processes. Rather than forcing every process to create its own adapter for each application, all of them can share the same VBO. This object-oriented approach also means that changes to the application's user interface require just its VBO to change. As long as the change doesn't require modifying the VBO's external interface, all of the processes that use it can remain as they are. This is quite different from traditional screen-scraping technologies, where each process had its own link to the user interface, and it's an important aspect of Blue Prism's design.

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, focusing on each of its main components.

CREATING BLUE PRISM APPLICATIONS: VISUAL BUSINESS OBJECTS

Blue Prism implements presentation integration with VBOs. Understanding what a VBO is first requires understanding what components each one contains, then seeing how Object Studio is used to create those components. This section looks at both.

What is a Visual Business Object?

The role of a VBO is to act as an adapter to the user interface of a specific application. To accomplish this, each VBO has three main parts:

- A *connector*, which is a standard library provided by Blue Prism for communicating with a particular kind of application user interface.
- An *application control interface (ACI)* that uses the VBO's connector to expose the elements of a specific application's user interface.
- One or more *pages*, each of which implements all or part of an operation that the VBO can perform.

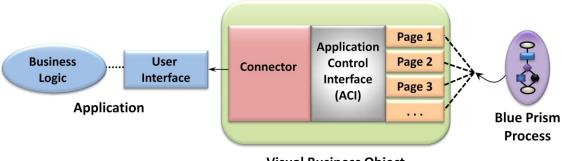


Figure 4 shows how these three relate to one other and to the world outside the VBO.



Figure 4: A VBO communicates with an application's user interface via a connector, uses an ACI to expose the elements of that interface, and implements operations in pages.

Understanding VBOs requires looking at these three components. What follows walks through each one, showing what it is and how it's created.

Choosing a Connector

The connector's job is to handle low-level interaction with an application's user interface. Since application interfaces use a variety of technologies, Blue Prism provides a variety of connector choices. The options are:

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

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

Creating an Application Control Interface

Whatever connector a VBO 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 job of the application control interface is to define the elements of each screen for a particular application, then make them available to the pages in this VBO. Just as a traditional application programming interface (API) can expose an application's business logic to other code, an ACI exposes an application's user interface to the logic contained in pages.

To create ACIs, a business analyst or (more likely) 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.

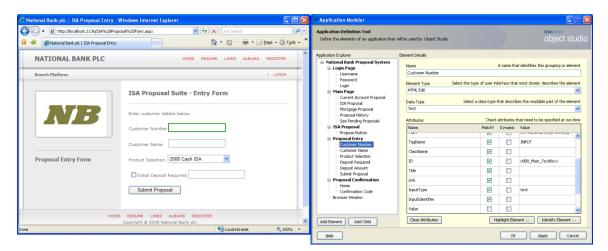


Figure 5: The Application Modeler in Object Studio helps its user create an ACI.

To create an ACI, a developer must determine which screens in an application's user interface this VBO 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 VBO's pages. In Figure 5, for example, which is for an HTML interface, the Application Explorer pane in the Application Modeler window contains a list of screens and elements for an example application. As this pane shows, 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 left in Figure 5 shows the actual user interface for the 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 ACI should use to identity this element at run time. Because Blue Prism uses attributes to identify elements, the ACI can sometimes remain unchanged even if parts of the application's interface are modified. For example, the ID attribute of an HTML element remains the same even if the page's layout changes, so an ACI might be able to handle this change without modification.

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

Creating Pages

The logic of a VBO is implemented in or more pages. As Figure 6 shows, each page contains an operation defined as a set of *stages* along with *data items* holding information used by that operation.

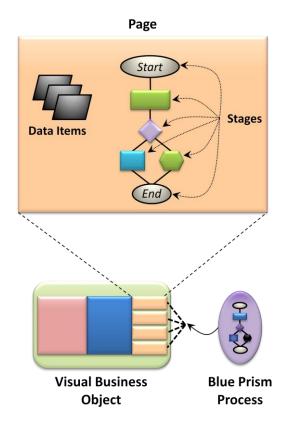


Figure 6: Each page in a VBO can contain stages and data items.

A page can be *published*, which lets it be called by Blue Prism processes and potentially other software. A page can also remain private, making it available only to other pages in this VBO. Whichever option is chosen, the page 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 operation this page performs. Some of the most commonly used stages in VBOs 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 VBO 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 a page's operation.
- Decision: acts like an If statement, branching an operation'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.
- Page: invokes another page in this VBO.
- Exception: explicitly raises an exception in the execution of an operation.
- 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 VBO might send an alert whenever an exception is raised, for example.

Creating an operation on a page—combining a set of stages—is like creating a method or subroutine. Rather than use code, however, the person creating this operation does it graphically using Object Studio's *Page Designer*. 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 integration logic, not just developers. Figure 7 shows a simple example of a login page created using Page Designer.

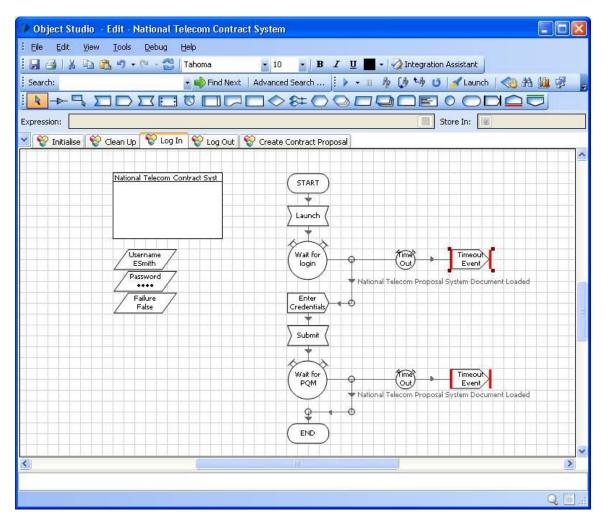


Figure 7: A business analyst or developer uses Object Studio's Page Designer to create a page's operation by arranging stages on that page.

As the figure shows, each page in this VBO appears as a tab in the Page Designer design surface. Here, the analyst has selected the Log In page, and so the tool shows the data items and stages on this page. The data items include what you'd expect a login page to need: a user name and a password. The operation has a Start and End stage, like all operations, 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 VBO works with. The Wait stage, shown as a circle, waits for the login screen to load, with a time-outbased error handler in case something goes wrong. Next comes a Write stage, named "Enter Credentials", that enters the user name and password from this page's data items. Another Navigate stage appears next, this time named "Submit", that clicks the screen's Login button. The operation 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, however, the operation ends normally—the login was a success.

It's worth pointing out that while this simple example uses data items to hold the username and password, a VBO would more likely read these from Blue Prism's credentials database, described later. This is significantly more secure, since this sensitive information need not be kept in the VBO itself. It's

also another example of how modern presentation integration technologies have addressed some of the traditional weaknesses found in earlier versions of this approach.

Once they're created, VBOs can be accessed directly by Blue Prism processes, as described in the next section. They can also be accessed by other software, however. Each VBO can expose a WSDL interface, with its operations invoked via SOAP. This can be useful for an organization pursuing a Web services-based SOA strategy, for example, or in other situations.

The way VBOs work makes basic testing straightforward. An analyst creating a page can open an Object Studio Page Designer window for the VBO he's creating and another window for the application this VBO works against. When he clicks on a stage in the VBO, 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 and operations without waiting for recompilation. Object Studio also contains a built-in debugger, allowing things such as stepping through an operation's logic one stage at a time, setting breakpoints, and examining the contents of data items.

Each VBO implements a set of operations against a particular application's user interface. Once an organization has created VBOs for its applications, those objects can be used (and potentially reused) by Blue Prism processes. How those processes are created is described next.

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 Blue Prism processes. Before looking at this tool, however, it's important to understand exactly what a Blue Prism process is.

What is a Blue Prism Process?

Since Blue Prism accesses applications through their user interfaces, a Blue Prism process acts much like a human user. Just as a person might interact with several applications to carry out a series of business steps, a Blue Prism process can invoke operations in several VBOs to carry out those same steps.

In some ways, a Blue Prism process is quite similar to a VBO. Like a VBO, a process is defined in one or more pages, and each page contains some number of stages and data items. There are important differences, however. The biggest one is the way in which pages can be invoked. In a VBO, any published page can be called at any time; the object exposes a set of operations that can be invoked in any order. A Blue Prism 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 operations.) Figure 8 illustrates this idea.

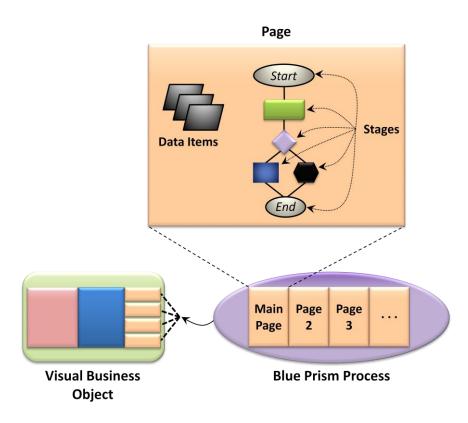


Figure 8: A Blue Prism process contains one or more pages and always begins execution at its main page.

As the figure suggests, execution of a Blue Prism 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 operations in VBOs as needed.

The stages allowed in a Blue Prism process are similar, but not identical, to those used in a VBO. 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. Processes can also use the Page stage to call another page in the same process.

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

Just as some stages appear only in processes, some VBO 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 VBOs 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 VBOs.

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 a set of 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 *workqueue*, which allows multiple process instances to read from a spreadsheet or other file. The workqueue feeds information to the running instances while managing access to the shared file they're using. It's also possible for Blue Prism processes to get their input from external software, such as an existing customer workflow system.

Creating a Process

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

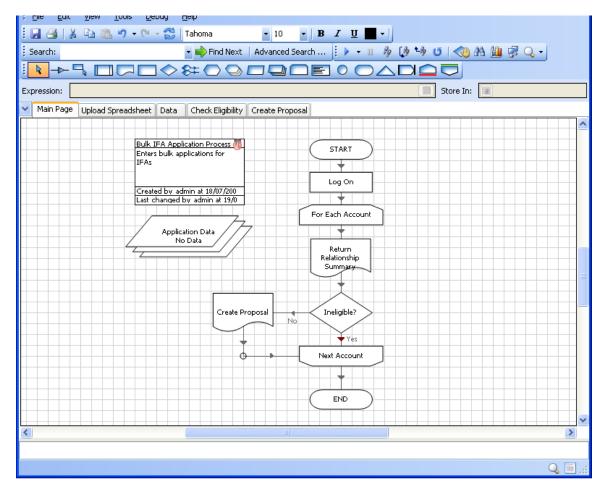


Figure 9: Using Process Studio, a business analyst or developer can arrange stages to define the logic of a Blue Prism process.

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 as well 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. The number and type of exceptions a Blue Prism process generates is also accessible, providing another kind of useful information. This immediate feedback can be helpful in making an automated business process run more effectively.

DEPLOYING BLUE PRISM APPLICATIONS

A Blue Prism 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 Blue Prism 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 Blue Prism 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 the Windows-based client of a client/server application, an IBM mainframe terminal (via a terminal emulator), and a Web application through a browser. Figure 10 shows how a Blue Prism implementation of this process would typically be deployed.

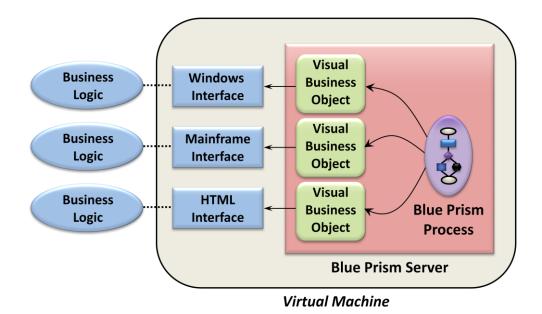
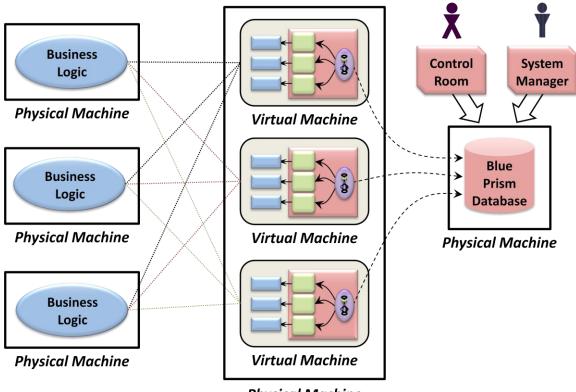


Figure 10: Each instance of a Blue Prism process, together with the VBOs it uses, typically runs in its own virtual machine.

The Blue Prism process and the three VBOs it uses all run in a Blue Prism server. Each VBO accesses the user interface of the application it's built for: the Windows client, the mainframe interface, and the Web browser. All of these things run in a single VM, just as the user interfaces of all three applications would run on a human user's desktop Windows machine. The business logic of each application doesn't execute in this VM, of course—it almost certainly runs on some other physical machine. Still, to each of these applications, the Blue Prism process will look like just another user.

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



Physical Machine

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

As the figure shows, a commom Blue Prism database can be shared by multiple VMs. In fact, even VMs spread across multiple physical machines can share a single instance of a Blue Prism database. This provides a common repository for the information used by Control Room and System Manager to control, track, and manage a group of running Blue Prism processes. The next section gives a brief overview of these tools.

MANAGING BLUE PRISM APPLICATIONS

Just like manual processes, automated processes need to be controlled and managed. To allow this, Blue Prism provides two tools: Control Room and System Manager.

The purpose of Control Room is to let both business analysts and IT staff work with Blue Prism processes. The tool lets them perform tasks such as:

- Starting and stopping instances of Blue Prism processes. (It's also possible to do this from the command line or with external software—using Control Room isn't required.)
- Viewing the log produced by each Blue Prism process, either while the process is running or after it's completed. Among other things, this log records when each stage in a process is executed.
- Creating process schedules. For example, a business analyst might define a schedule that runs
 Blue Prism process X at 9 am every Friday morning. If X completes successfully, the schedule then

runs Blue Prism process Y. If X doesn't complete successfully, the schedule runs Blue Prism process Z.

Viewing workqueues, examining things such as how far Blue Prism processes have gotten in a particular input file and what exceptions have occurred.

While Control Room allows working with running processes, System Manager 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 VBOs and processes or perhaps just run existing processes. System Manager can also be used to install and manage Blue Prism processes and to carry out a variety of other management functions. As with Control Room, the information System Manager uses to do all of this is stored in the Blue Prism database.

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 Audit Log Viewer, part of System Manager, an administrator can examine records of things such as when a user logs into Blue Prism, when that user starts or deletes a process, when that user changes her password, and more.

Another issue, especially for a technology based on presentation integration, is safely storing and managing the user names and passwords used to log in to applications. To do this, Blue Prism provides an encrypted credentials database controlled by a Blue Prism administrator. For example, a set of credentials (i.e., a user name and password) might be usable only by specific Blue Prism processes started by users in specific roles and running on specific servers. It's also possible to use Active Directory in place of some parts of Blue Prism's user management mechanisms. While it's not required, larger organizations in particular can find this useful.

THE BLUE PRISM METHODOLOGY

When an organization creates a Blue Prism process, it commonly builds directly on live, running applications. This is a scary thought to many IT people: build and test a new application on top of inproduction software? Yet remember that the main benefits of using presentation integration to automate business processes are speed and low cost. To achieve these things, it's not feasible to construct dummy copies of the applications (and the data) that underlie the process. Working on live systems is all but required.

Viewed from another perspective, though, this approach isn't so shocking. Recall that deploying a new Blue Prism process instance is like adding a new human user. This is something that organizations have plenty of practice with, since new people learn to use enterprise applications regularly. Just as a new user must go through a supervised training process, so must a new Blue Prism process be created using a defined methodology. This methodology has four phases: Design, Read-Only, Update, and User Acceptance.

- Design: Business users who understand the process being automated create a process design document. This document specifies in detail what scenarios should be implemented for this business process. Next, business people and IT people work together to create a solution design document that describes the Blue Prism solution. Finally, the team writes a test plan that defines how these scenarios will be tested once the automated process is available.
- Read-Only: Business analysts and/or developers work to create the VBOs and Blue Prism process required for a solution. They focus, however, only on read operations—no updates are created. For a typical process, this covers the majority of the interactions between Blue Prism software and the applications it uses. By limiting their efforts to read-only operations, the development team can go a long way toward building a complete solution with no risk to running applications. They can also discover a great deal about how accurate the process design document is, learn about challenges such as undocumented pop-up boxes, and more. And because there's no danger of damaging an organization's systems, this phase doesn't require continuous participation from an expert in the business process being automated.
- Update: The team creating the solution is joined by an experienced user, someone who understands this business process in detail. The team walks through the work done in the Read-Only phase, verifying that it's correct and making changes where it's not. With the participation of the experienced user, the team then begins to add operations that update the applications used by this process. Much like training a new user, the people creating the automated process are guided by an old hand who checks their work at each step. Once an initial scenario has been implemented correctly, the team moves on to implement the next scenario in the process design document. As before, this new scenario is checked step-by-step until it's correct. This process continues until all of the scenarios have been implemented correctly. Once this milestone is reached, the Blue Prism process is deployed in a real configuration—in VMs running on one or more physical servers—and proven to work in this environment.
- User Acceptance: This final phase consists of a series of iterations. In each one, the development team asks the business users to choose, say, ten distinct scenarios for this process. The Blue Prism process executes these scenarios, and the results are checked. Once again, it's much like training a new human user to execute these scenarios. One important difference is that Blue Prism automatically maintains an audit trail for each execution of the process. This gives everyone involved a clear sense of what exceptions were encountered. Also, because the Blue Prism process stops if an exception occurs, the possibility of corrupting one of the applications the process uses is limited. As before, any problems are fixed, and the next iteration, with a different set of scenarios, begins. This process continues until the number of exceptions falls to an acceptable number. Once this happens, the Blue Prism process can go into production.

Notice the role that business people play in this methodology: They're deeply involved in making sure that the automated process is correct. In fact, business analysts might even be the prime movers on the team developing the process. To some IT people, this kind of business-led computing is terrifying. Even with the Blue Prism methodology just described, won't the product's business-friendly tools tempt non-expert people to take too much into their own hands? And if something goes wrong, won't they blame IT rather than themselves?

These are legitimate concerns, especially in organizations where there's not much trust between the business and IT. Still, when the overarching need is for rapid process development and frequent process change—and it often is—making these efforts contingent on IT often isn't feasible. Helping the business side of the organization do as much as it can, then gently working with them to fix any mistakes, is likely to be a better course. After all, they know their own processes intimately, and they certainly want to automate them correctly. With effective change control and solid communication, IT and business people can work together toward a common goal.

CONCLUSIONS

Automating business processes can bring substantial value. In situations where this automation must be done quickly and at low cost, and where changes are frequent, relying on presentation integration can be a good solution, especially when the performance requirements aren't too high.

Blue Prism is designed for these situations. Combining tools for presentation integration, process automation, and management, it aims at providing an enterprise-class solution for this category of problems. The product's primary goals are 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.
- Allowing the functions of application user interfaces to be exposed through reusable services. By encapsulating all direct application interaction in objects, Blue Prism can avoid many problems of older presentation integration technologies and fit better into a service-oriented world.

No technology is right for every situation, and presentation integration is no exception. Yet applied to the right problems, this approach can be a good choice. As Blue Prism shows, presentation integration surely has its place in the modern world.

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (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 technology.