David Chappell
& Associates

Understanding BPM Servers

David Chappell

October 2004

Sponsored by Microsoft Corporation

Contents

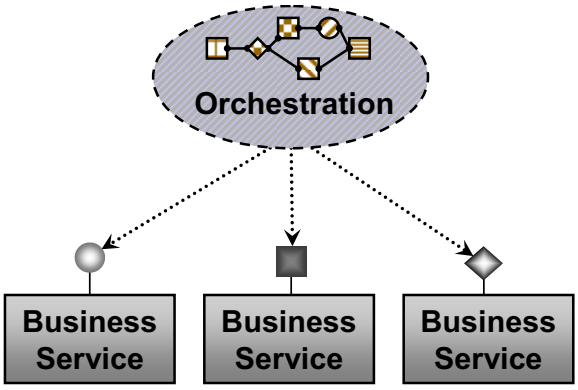
Composite Applications in a Service-Oriented World

The next generation of application architecture is upon us. Just as mainframes gave way to client/server applications, which in turn were supplanted by multi-tier applications, the next major architectural style has appeared. The new norm is *service-oriented applications*, where business logic is exposed to other software through standard services. Made practical by the universal agreement on web services, applications built in this style will dominate over the next few years.

Building service-oriented applications raises an obvious question: what will call those services? One important category of clients will be user interfaces such as web portals, desktop applications, and software running on mobile devices. Services themselves will be another type of client, since some services will surely rely on others to carry out their functions. But there's a third category of client that may turn out to be the most important of all: process logic, the software that knits independent services together into cohesive solutions.

For example, think about what's required to add a newly-hired employee to various software systems within a company. Carrying out this task might require interacting with one application that handles payroll, another that maintains information about health insurance, a third that tracks retirement benefits, and perhaps others. Each of these applications exposes services (in at least a loose sense of the word) that can be accessed by the logic driving this process. Taken together, this collection of services and process logic makes up a *composite application*.

The figure below gives an abstract view of a composite application. Business services, such as those supporting the addition of a new employee, are made available by various applications. The process logic that drives these services, most commonly referred to as an *orchestration*, is implemented as a separate piece of software distinct from the business services. Typically, as in the case of adding a new employee, a composite application implements all or a substantial part of some business process.



Supporting Orchestrations: BPM Servers

Business services can be provided by legacy applications, by packaged applications such as ERP systems, or by new software built on an application server such as the .NET

Framework or a J2EE-based product. Today, every major software vendor has embraced service-oriented development, and so the foundations for creating service-oriented applications are in place. But what is the right foundation for orchestrations? What does a platform designed to support the process logic of a service-oriented composite application look like?

The answer is now clear. Just as application servers were created to support the multi-tier applications that appeared in the last great change in architectural styles, a new style of server is appearing to support the orchestrations that the shift to services implies. Multiple vendors are working to fill this gap, and as is typical for new infrastructure technologies, there's still some diversity in their approaches. It took several years for the industry to reach a consensus on exactly what technologies should be included in an application server, and it's likely to take some time for a similar consensus to emerge on what a process-oriented platform should provide. This consensus is coming, however, and a new kind of server is making its way into the world. Focused on supporting process-driven composite applications in a service-oriented world, this server fits squarely within the concepts of *business process management (BPM)*, and so can be thought of as a *BPM server*. Examples of BPM servers today include Microsoft's BizTalk Server 2004, IBM's WebSphere Business Integration Server Foundation, BEA's WebLogic Process Edition, and others.

The Genesis of BPM Servers

From a service-oriented perspective, the goal of a BPM server is plain: it should provide the right foundation for orchestrations that drive composite applications. To achieve this goal, several different streams of technology have come together. They include the following:

Service-oriented applications allow creating a service-oriented architecture (SOA), and the move to SOA is perhaps the strongest force in application development today. Yet this approach effectively implies the creation of a platform supporting process logic—orchestrations—to drive the services an SOA exposes. This platform is a BPM server.

Vendors of integration products realized that both enterprise application integration (EAI) and business-to-business (B2B) integration were just subsets of the larger problem of automating business processes. Connecting diverse applications, perhaps across organizational boundaries, is important, but it's not enough. While EAI and B2B provided a good lens for viewing this problem in the 1990s, it's now become apparent that the real goal is BPM. As a result, integration products have morphed into BPM servers.

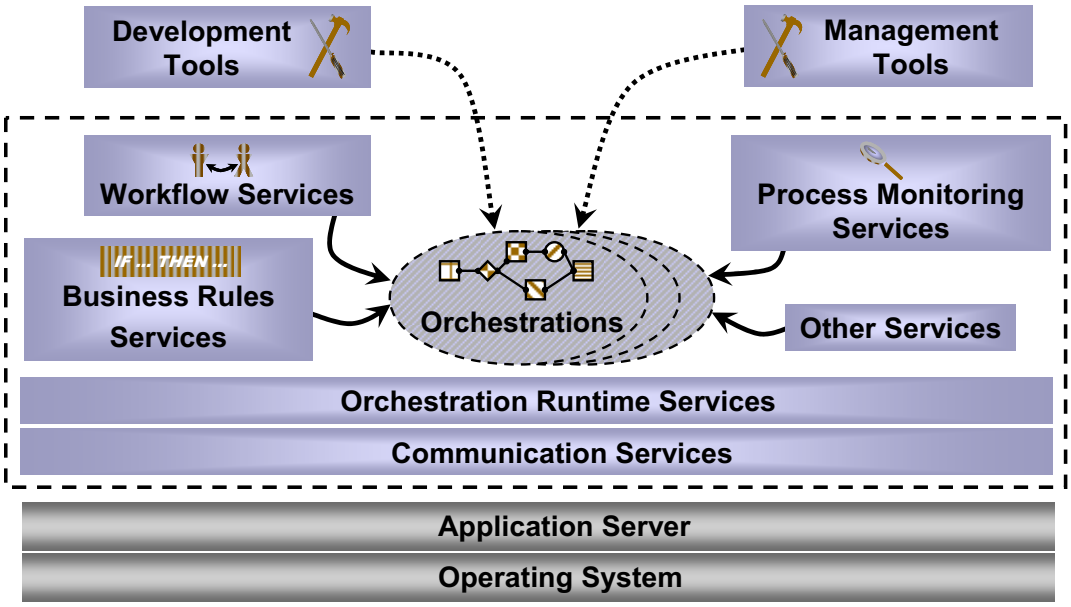
Vendors of traditional workflow products, focused on person-to-person processing, also realized that workflow was just another subset of business processes. Like vendors of integration products, they saw that the future lay not in specialized products for this particular area, but rather in the more general approach defined by BPM. Like integration products, traditional workflow products have also moved to become BPM servers.

It's important to realize that while BPM server-based composite applications are a natural evolution in software development, they aren't a panacea. This kind of application can be challenging to maintain, given the diverse technologies it relies on and its inherently distributed nature. Performance might also be a concern, especially for web services

interactions in very high volume scenarios. As with other technologies, using this approach makes sense only when it's the best solution for the business problem at hand.

Examining BPM Servers

Real business processes are seldom simple. The orchestration that drives a service-based process might access several applications, run for hours, day, or weeks, implement complex business rules, interact with many different people, and more. To support all of this, a BPM server must provide a diverse set of technologies and tools. The figure below illustrates what a typical BPM server provides and how its pieces fit together.



As the diagram shows, BPM server vendors commonly build their offerings on an application server, such as the .NET Framework or a J2EE-based product, which in turn runs on some operating system. On this foundation, a BPM server can provide a range of support for orchestrations, including the following:

Communication services that allow an orchestration to interact with the various business services a composite application uses;

Orchestration runtime services that directly support an orchestration during execution, such as state management, transaction support, and more;

Development tools for defining the orchestration logic that drives a composite application, specifying mappings between data used by various business services, and other purposes;

Management tools for managing the orchestration, its communications, and other aspects of the BPM server;

Business rules services, allowing the business rules an orchestration uses to be created and managed separately from the process flow;

Workflow services, allowing people to participate in the business process this composite application implements;

Process monitoring services, including both monitoring at a technical level and business activity monitoring (BAM) that provides real-time information about a running process;

Other services, which vary depending on exactly which areas a particular vendor chooses to emphasize.

BPM servers include a diverse set of technologies. To get a better sense of what these products typically offer, it's worth examining each of the areas listed above in more detail.

Communication Services

The most fundamental problem that a BPM server needs to solve is communicating with the business services an orchestration uses. Eventually, the dominant way to do this will be via SOAP-based web services, and so the communication services built into a BPM server today must include support for web services. Yet while SOAP may one day be all that's needed, that day is years away. Realistically, other communication mechanisms are also necessary, at least for the next few years. Native connections to common communications products such as IBM's WebSphere MQ are required, for example, as are links to popular applications such as SAP R/3. The typical approach to addressing this problem is to provide *adapters* that support these connections to diverse technologies.

It's worth pointing out how different these requirements are from the communication services provided by a typical application server. The .NET Framework, for example, provides only SOAP, .NET Remoting, and access to Microsoft Message Queuing (MSMQ), while a typical J2EE application server provides SOAP, Java Remote Method Invocation (RMI), and access to the Java Message Service (JMS). Neither offers the diverse communication choices required to create functional composite applications in today's enterprise environments.

Orchestration Runtime Services

BPM servers provide a broad range of runtime services for orchestrations. This section describes the most important of these.

Executing an Orchestration

As described in the following section on development tools, orchestrations are typically defined graphically rather than through code. Diagrams aren't directly executable, however, so a BPM server must provide some way to translate an orchestration's graphical depiction into an executable form, then actually execute it. One approach is to translate an orchestration diagram into a low-level form, such as Java bytecode or the .NET Framework's Intermediate Language (IL), then execute it like any other program. Another possibility is to transform the logic represented in the diagram into a process-oriented language, then execute the resulting program on a specialized process engine.

Because different BPM servers implement orchestrations in different ways, trying to run an orchestration created for one vendor's product on another BPM server is unlikely to be successful. Yet there are situations where it can be useful to define web services-based interactions between orchestrations that can be executed on any vendor's BPM server. To address this need, a group of vendors including Microsoft, IBM, and others have defined the Business Process Execution Language (BPEL). Now being standardized under the auspices of OASIS, BPEL is an XML-based language for defining interactions via web

services. While most BPM servers today support BPEL in some fashion, how they do it varies across products.

To get a sense of how BPEL might be used, suppose a large manufacturer wishes to define a standard purchasing process, then get all of its suppliers to implement this process. To accomplish this, the firm could describe the process in BPEL, then distribute the resulting description to its suppliers. Assuming the BPM servers those suppliers were using supported BPEL and that all of the process's interactions used web services, implementing this cross-enterprise process would become significantly easier.

Managing an Orchestration's State

One of the biggest differences between an orchestration and the business services it uses is the time each takes to execute. A request to a typical service generates a reply within a few seconds. Because it commonly drives all or part of a business process, however, an orchestration may run for hours, days, or weeks, depending on how long the process takes to finish. What if human approval is required at some point in the process, for instance, and the person who must give her approval is on vacation? Because business processes can take a long time to complete, the orchestrations that control them can also run for a long time.

This long-running nature affects how an orchestration manages the in-memory information it maintains—the state—about a running process. If the orchestration is blocked for a significant period of time, keeping this state in memory doesn't make much sense. Instead, a BPM server should provide a way for an orchestration's state to be automatically written to disk, then restored again when the business process resumes, even if it's days or weeks later.

State management illustrates another notable difference between BPM servers and application servers. Since supporting long-running business processes isn't their primary purpose, application servers haven't traditionally addressed this kind of state management. Because they are explicitly intended to support long-running orchestrations, however, BPM servers do provide this service.

Handling Transactions

Many business processes require the all-or-nothing behavior characterized by a transaction. For example, an orchestration driving a business process might need to invoke two business services and ensure that either both requests succeed or both fail. This kind of atomic transaction can be accomplished using a standard two-phase commit protocol, and it's something that BPM servers typically support. In fact, application servers include this feature, so a BPM server built on an application server can offer this quite easily.

The nature of many business processes raises another issue, however. What if a particular process requires all-or-nothing behavior, but a traditional atomic transaction isn't possible? Atomic transactions require locking data for the life of the transaction, something that isn't a problem when the transaction is short. But suppose the services that must be bundled into an all-or-nothing group include one that requires human approval. Even if the required approver isn't on vacation, the time it takes for a person to respond is likely far too long for data to remain locked. Or what if one service that must be in this transactional group doesn't participate in atomic transactions? This isn't a far-fetched worry, since many applications won't let arbitrary clients lock their data.

To handle situations like these, a BPM server supports *long-running* transactions. Also called *business activities* and other names, long-running transactions handle errors not by rolling back all updates, but rather by executing some kind of compensating logic when an error occurs. For example, suppose a particular long-running transaction includes an atomic transaction that transfers money from one bank to another, followed by an operation that executes another application once the transfer has succeeded. If this final operation fails, the logic of the business process requires that the money transfer be undone. Yet the atomic transaction that performed this transfer has already committed—how can it be reversed? The answer is that compensating logic must run if a failure occurs, logic that might execute another atomic transaction to undo the effects of the transfer. A BPM server provides built-in facilities that allow the creator of an orchestration to define this compensating action, then have it automatically execute when a long-running transaction fails.

While compensation is useful when atomic transactions aren't possible, it's not without problems. Suppose an orchestration modifies some data in the early part of a long-running transaction, for instance, then runs a compensating operation later to change this data back to its original state. What happens if some other application accesses that data in between these two events? This second application may well use data that's ultimately deemed to be incorrect in making business decisions, such as computing credit risk. Or think about operations for which there is no obvious compensation. If an orchestration causes a missile to be launched, there's no way for compensating code in that orchestration to reverse this. Yet while compensation isn't a perfect solution, it is nevertheless the right approach for an important category of problems faced by business processes.

Correlating Requests and Responses

Suppose two instances of the same type of orchestration each invoke the same business service at around the same time. Each orchestration has sent some information into the service, such as a purchase order, and each is waiting for the same response, such as an invoice. How can the correct response, the matching invoice, be delivered to the right orchestration? In other words, how does *correlation* happen between requests and responses?

On the face of it, this problem seems simple: just make the request a remote procedure call (RPC), forcing the orchestration to block until the response comes back. Since each RPC typically happens over a single logical connection, the response can be easily routed to the correct requester. But this simple solution falls apart when the requester is an orchestration driving what might be a long-running business process. What happens if the response takes a week to come back? Keeping a logical connection alive that long isn't practical. Another possible solution is for the BPM server to insert a unique identifier into the request message, then require that the response contain the same identifier. This also sounds appealing, but it implies modifying the software sending that response to include this identifier, an unattractive prospect.

A better solution takes advantage of the strong likelihood that the request and response messages themselves contain information that can be used to correlate them. A purchase order and its matching invoice, for example, probably contain identical purchase order numbers and other values that can be used to uniquely identify this interaction. A BPM server can allow the creator of an orchestration to indicate which values in the request and response should be used to correlate requests and responses, then automatically route responses to the orchestration instance that made the request. Once again, this is another

example of how the long-running nature of business processes forces BPM servers to provide a function that's missing from traditional application servers.

Exposing an Orchestration as a Web Service

As discussed so far, an orchestration acts as a client of services. By providing the central logic for a business process, it drives the operations that make the process go. There are many cases, however, where it's useful for an orchestration to be a service itself. Suppose, for example, that an orchestration for some complex business process wishes to make use of other existing orchestrations, or that a business process needs a user interface (not an unlikely occurrence). To address situations like these, a BPM server can allow an orchestration to expose itself as a web service. Doing this makes clear how blurry the line between services and orchestrations can be, but it's nonetheless an important feature for a BPM server to provide.

Development Tools

An orchestration implements the logic that drives a business process. Accordingly, orchestration logic tends to be process-oriented, more concerned with issues like the order in which services should be invoked than with detailed problems like constructing a SQL query. This focus on process rather than service-level detail makes it possible to describe an orchestration graphically rather than with a traditional programming language. While typical services will still mostly be created in Java, C#, or some other conventional language, virtually all of today's BPM platforms provide graphical tools for creating orchestrations.

Doing this has some substantial advantages. Graphically-defined logic can be quicker to understand, making both creation and maintenance easier and cheaper. And unlike logic defined in, say, C#, graphically-defined orchestrations can potentially be developed at least in part by business-oriented analysts rather than by purely technical developers. This increases the chances of getting the process right, since the people who truly understand the process can be much more involved in defining its logic.

The ability for business-oriented analysts to participate in defining orchestrations has led some BPM enthusiasts to assert that developers are no longer required to automate business processes. With BPM, they claim, business-oriented analysts can do it all. Even a cursory look at today's BPM server products, however, and at the technical complexity of combining business services provided by diverse software into composite applications, makes clear that these assertions are wishful thinking. While BPM is certainly a step forward, developers will still play an important role in creating applications. To make it easier for developers and business people to work together, however, some BPM servers provide two levels of orchestration tooling. One tool targets business analysts, and so focuses purely on process-oriented definition. The other tool targets developers, exposing more of a composite application's technical detail. Business analysts and developers can then share definitions across these tools to more effectively create orchestrations.

While defining process logic graphically is a useful approach, it's not without its weaknesses. Graphical tools might provide only 90% of the solution, for example, requiring a developer to write code to handle the rest. And even though the idea of business analysts working with software developers to create graphical processes is appealing, many organizations just don't work this way. In fact, many organizations don't even have business analysts as such, which can make it harder to reap the advantages of a service-oriented approach.

Another kind of tool is also important for creating orchestration-driven composite applications. To see why, think about how an orchestration might use the data it exchanges with business services. One common scenario is that information obtained from one service must be transformed in some way, then sent to another service. For example, suppose an order received from one application involved in a business process is rejected for some reason. The message sent to indicate this rejection might contain some of the same information that was in the original order message, such as a unique identifier and the quantity requested. Or perhaps information used by different services, such as a part number, is maintained in different formats, e.g., character and numeric.

For situations like these, data mapping tools can provide a way both to control what data is copied between services and to modify that data as it's copied. Because many BPM servers handle all data as XML internally, it's common to use XSLT as a standard approach for modifying information when required. Although not every application requires them, data mapping tools are nonetheless an important part of a typical BPM server.

Management Tools

Managing a BPM server and the orchestrations it supports is an essential aspect of creating effective composite applications. As a result, management tools are critically important for any BPM server. Management is a broad area, and different BPM servers take different approaches. Among the common themes are support for configuring orchestrations and the server itself, the ability to monitor message flows and set thresholds, and tools for creating reports that describe the behavior of orchestrations and their interactions.

Business Rules Services

Imagine a lender that processes mortgage loan applications under $300,000 differently than applications for larger amounts. This policy of handling different-sized loans in different ways is an example of a *business rule*. Business processes depend on business rules, and so does the software that implements those processes. In a composite application supporting this loan application process, for instance, the rule just described might cause an orchestration to invoke one business service for applications less than $300,000 and another for applications above this amount.

It can sometimes make sense to implement business rules directly in an orchestration, and typical BPM servers allow creating orchestrations that do this. It can also be useful, however, to separate business rules from the process logic that depends on them. This is commonly done using a *business rules engine (BRE)*, software designed explicitly for storing and executing business rules. Rather than embedding rules directly, an orchestration can instead invoke the BRE when necessary to make a decision based on one or more rules.

Using this approach has several advantages, including the following:

- Business rules can be stored in a single place. Rather than being scattered throughout the process logic of an orchestration, rules can be organized into well-defined groups. This makes it easier to understand what rules are being used, and it also makes reuse of those rules easier.

- Less technical people, such as business analysts, can create and modify rules. While working with rules in a BRE still requires some technical bent, it's much simpler than working with business rules implemented in a programming language. Because

developers aren't required, rules can be changed more quickly to reflect changing business requirements.

Changes to rules can be deployed more easily. In many business processes, the rules change more frequently than the process itself. In the example just described, for instance, suppose the limit at which mortgage loans are processed differently increases from $300,000 to $350,000. Separating the rules from an orchestration's logic can allow this change to be made more easily, since the orchestration itself need not be directly updated.

But BREs aren't a free lunch. They can also present challenges, including the following:

Maintaining substantial sets of rules can be difficult. What impact will adding a new rule or changing an existing one have on how a process operates? Without an effective way to answer this question before the addition or change is made, BRE-based processes can produce surprising results.

While letting non-IT people in an organization directly change rules can lessen the burden on the IT department, it can also create new problems. People outside IT often don't have good change control procedures, for example, and so rules can be added or modified in uncontrolled ways. Since the problems this causes show up in software, IT professionals are still likely to be held accountable.

Despite these concerns, many applications can benefit from using a BRE. This includes both composite applications and other more traditional software. In fact, there's no strong technical reason why BREs should be thought of as part of BPM servers, and several firms today sell them as distinct products. Nonetheless, it's common today to view BREs as part of the broad landscape of BPM and thus as a component of a BPM server.

Workflow Services

On its own, an orchestration typically implements the controlling logic for a business process that executes without any human intervention. Sometimes called *straight-through processing*, this approach is a perfect match for many types of business processes. Yet it's not enough for many others. Think once again about loan origination, for example. Different people acting in different roles may need to be involved at various stages in this process, with tasks passed from one person to another. Each of these individuals might need to make a decision, approve a decision made by someone else, or provide other input to the process. Human-oriented business processes like this are critical in many organizations, and so viewing processes solely in terms of orchestrations and the business services they invoke ignores an important part of reality. Given this, a complete BPM server must support *workflow*, a term that today is generally used to mean human involvement in a business process.

Workflow technologies frequently allow creating a work list of specific tasks for each of the people involved, perhaps a way to define various roles that can be filled by different individuals, and other services. Workflow products predate the advent of BPM, and while some BPM servers have roots in this area, those that grew from integration-oriented products frequently add workflow support on top of their existing technologies. However it's accomplished, allowing people to interact with orchestration-driven composite applications is a central requirement for a BPM server.

It's also one of the most difficult problems to solve. Depending on which side of the fence a BPM server originated on—human-based workflow or software-based straight-through

processing—it's likely to be much better at one of these two approaches than the other. Implementing orchestrations that also support workflow in an effective, straightforward way is a challenging problem. It's fair to say that no BPM server today has fully solved it.

Process Monitoring Services

As with anything that automates a business process, keeping track of what's happening in a composite application is important. As described earlier, a BPM server's management tools commonly provide a way to monitor a process at a technical level. But while providing information aimed at technical staff is important, it's not enough. Business people can also benefit from services that monitor a running business process.

The reason for this is obvious: making good business decisions requires up-to-date information. If an organization's business processes are built on a BPM server, the orchestrations running on that server must make this information available in a way that's useful to business people, not just technicians. How many orders have been received in the last hour, and for which products? What products are selling best at each location today? What percentage of loan applications was approved this morning? Providing answers to questions like these is the province of *business activity monitoring (BAM).*

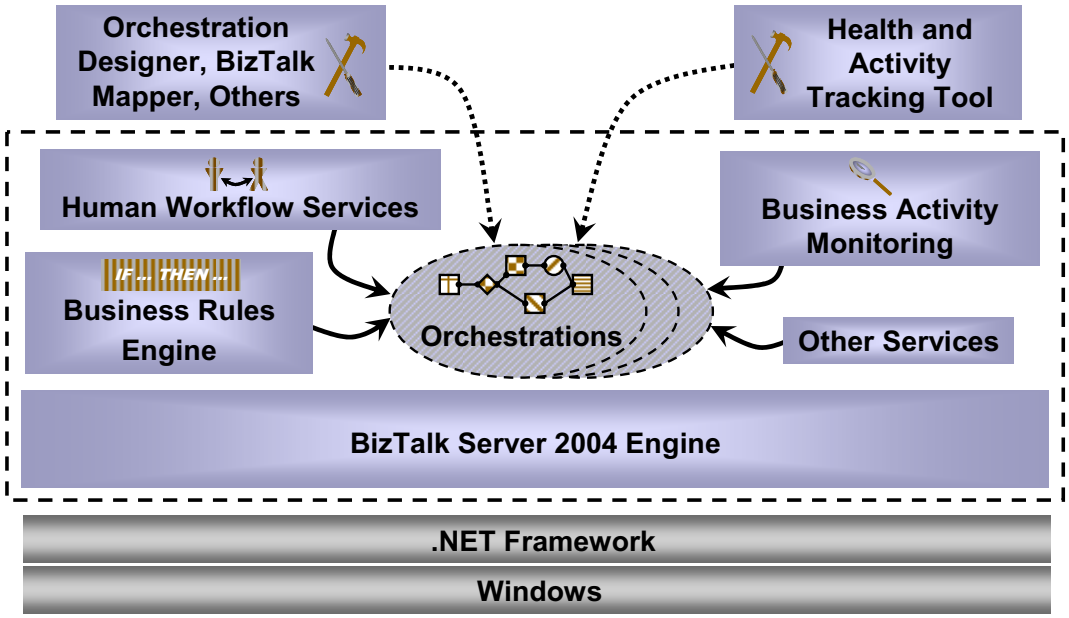
Originally defined by Gartner, BAM is a broad concept, encompassing more than just information from BPM servers. Because it runs the heart of an orchestrated business process, however, a BPM server must make available the information needed to provide BAM. And unlike traditional business intelligence technology, which focuses on historical data, BAM provides up-to-the-minute information about currently executing business processes. The goal is to give business people the best possible inputs for making decisions.

Other Services

While the services described so far reflect typical BPM server products, there is a good deal of variation, too. Some products add B2B-oriented services, for example, such as management of trading partner profiles and support for industry-specific standards such as RosettaNet and SWIFT. Others provide powerful tools for business process modeling and simulation, allowing organizations to make better decisions about how those processes should look. Given that different vendors emphasize different aspects of the problem, this variation isn't surprising. Yet despite the availability of extra features in specific products, the commonality across BPM servers is large. Their similarities are much bigger than their differences,

An Example BPM Server: BizTalk Server 2004

It's useful to understand BPM servers in an abstract way. It's also useful to illustrate these ideas with a concrete example. This section describes Microsoft's BizTalk Server 2004 in terms of the conceptual framework just described. The figure below shows how the major technologies of BizTalk Server 2004 correspond to the services in the BPM server diagram shown earlier.



As the diagram indicates, BizTalk Server 2004 is built on the .NET Framework and runs on Windows. The core of the product, called the BizTalk Server Engine, is used by every orchestration. Other components may also be used depending on what's required for a particular application. The following sections describe these components and the engine that acts as the product's foundation.

Communication Services

In BizTalk Server 2004, communication services are provided by the messaging aspects of the BizTalk Server Engine. Adapters provided by Microsoft and others support various communication mechanisms, including HTTP, FTP, SOAP, and WebSphere MQ, along with connections to popular applications such as SAP R/3. BizTalk orchestrations also rely on *pipelines*, components that perform various actions on incoming and outgoing data. These actions can include creating or verifying digital signatures, converting the data into and out of XML, and others.

Orchestration Runtime Services

Along with communication services, the BizTalk Server 2004 engine also provides orchestration runtime services. Those services include the following:

Execution: In BizTalk Server 2004, the graphical definition of an orchestration is transformed into a standard .NET assembly, then executed using the Common Language Runtime (CLR). Versioning and other aspects of an orchestration's life cycle use the mechanisms assemblies provide, as in any .NET Framework-based application. BizTalk Server 2004 also allows importing a BPEL definition to create an orchestration and exporting an existing orchestration to BPEL[1].

State management: An orchestration that receives no inputs for a period of time will have its state automatically written to disk using Microsoft SQL Server as the

[1] For more on Microsoft's views on BPEL, see http://www.gotdotnet.com/team/wsservers/bts2004/bpel_v10.zip.

underlying storage mechanism. When an input arrives for this orchestration, the BizTalk Server Engine causes the orchestration to resume and reloads its state.

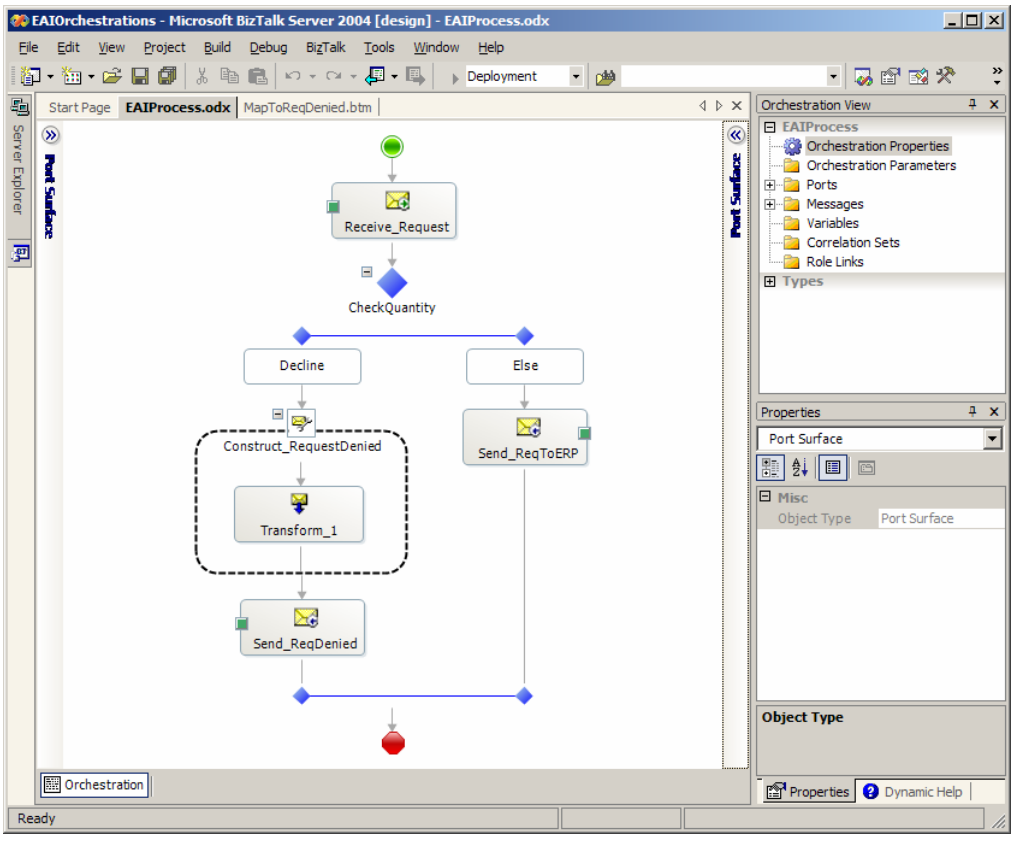
Transaction support: The creator of an orchestration can group a set of operations into a scope, then mark that scope as requiring either an atomic transaction or a long-running transaction. The operations in a scope that uses an atomic transaction will rely on traditional two-phase commit technology to ensure all-or-nothing behavior. Scopes marked as long-running have associated user-defined compensation logic that executes when a failure occurs.

Correlation: An orchestration's creator can define correlation sets that inform the BizTalk Server 2004 Engine which message fields should be used to associate requests and responses. The Engine then routes those responses to the correct orchestration instances when they arrive.

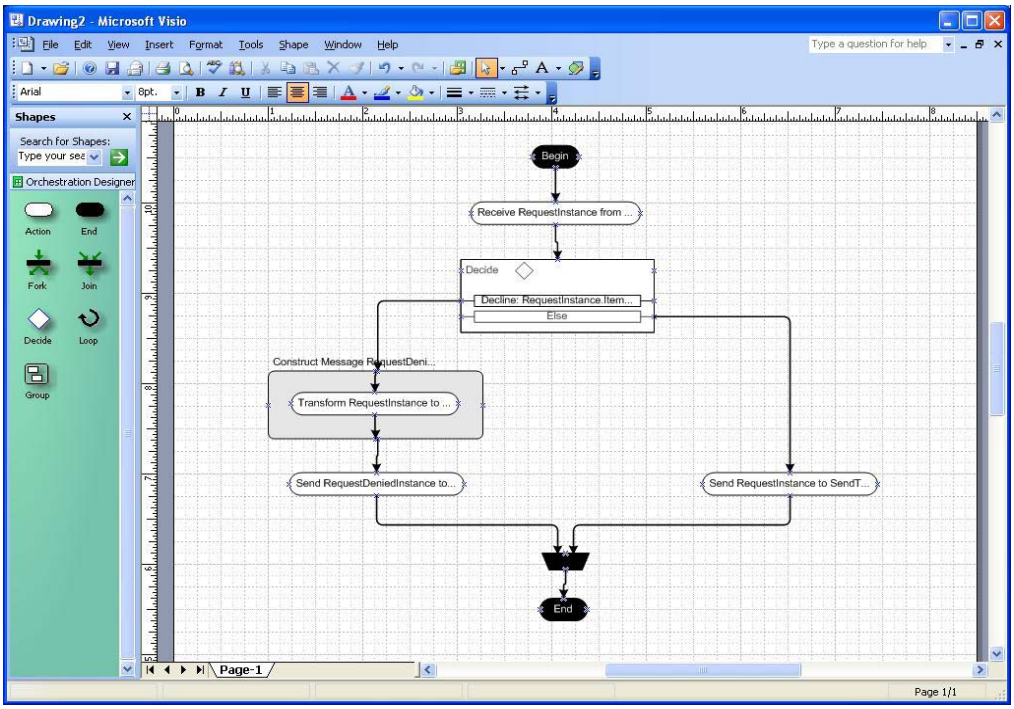
Exposing an orchestration as a web service: An orchestration can be exposed as a web service using the .NET Framework's ASP.NET technology.

Development Tools

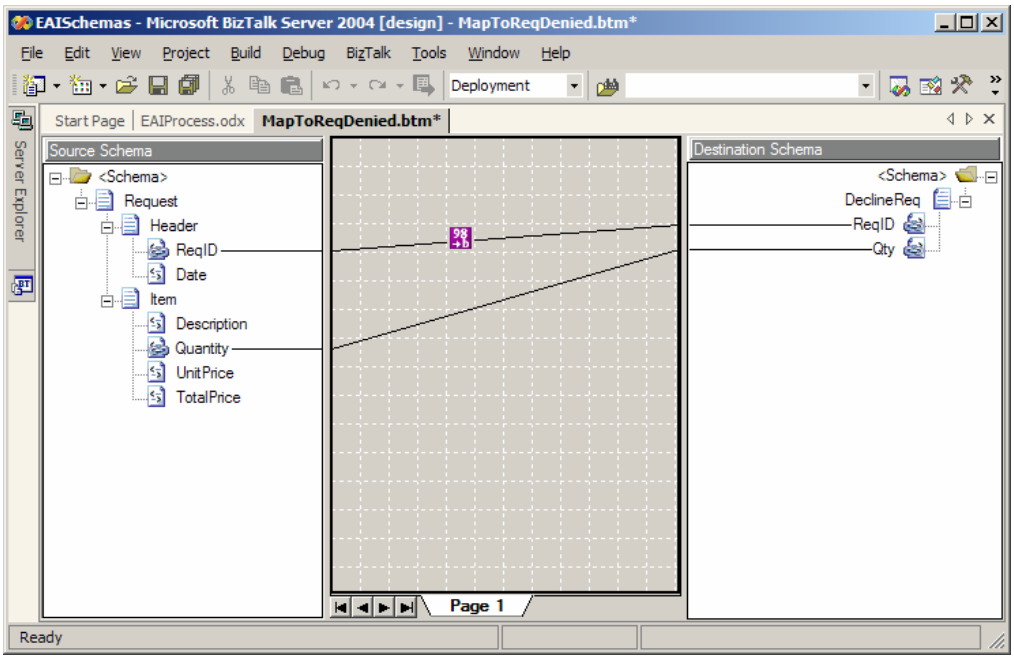
The primary tool for creating an orchestration in BizTalk Server 2004 is the Orchestration Designer. Running inside Microsoft's Visual Studio, this tool allows a developer to specify an orchestration's behavior using various shapes. In the simple orchestration shown below, for instance, the diamond shape represents a decision (i.e., an if-then-else statement), while envelopes represent sending and receiving messages (i.e., interacting with business services). Other shapes are used to create loops, execute actions in parallel, group actions into scopes, and carry out other functions.



Orchestration Designer is meant to be used by developers. To allow business analysts to participate more easily in orchestration design, BizTalk Server 2004 also includes a tool called the Orchestration Designer for Business Analysts. Rather than running inside Visual Studio, this tool runs inside Microsoft Visio, as shown below. Orchestrations created with this tool can be imported into the developer-oriented Orchestration Designer, modified, then copied back to their original home for more work. The intent is to allow developers and business analysts to work together, providing an appropriate tool for each.

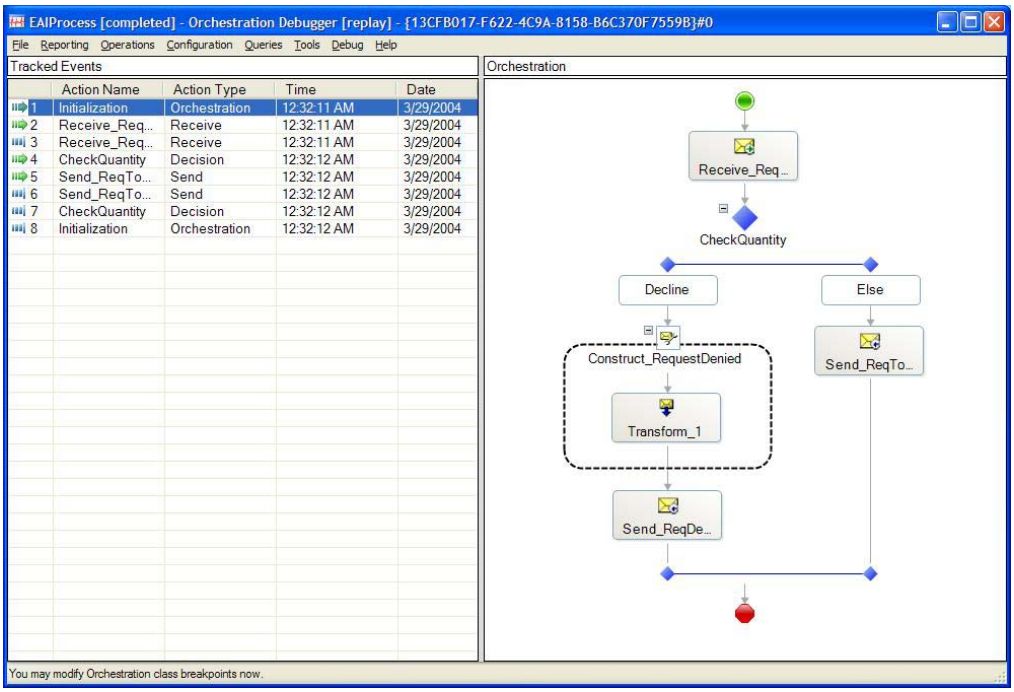


BizTalk Server 2004 also includes tools for data mapping. The BizTalk Editor provides a graphical approach for creating XML schemas, while the BizTalk Mapper, shown below, allows defining mappings and transformations between fields in messages defined by those schemas. This example shows two message schemas, each defined using the BizTalk Editor. Two values from the order request message are being copied into the message sent when an order is denied. One of those values, containing a unique identifier for this order, is transformed from character to numeric form when it is transferred using one of the BizTalk Mapper's built-in transformations.



Management Tools

In BizTalk Server 2004, management of the server itself and of the orchestrations it supports is done using the Health and Activity Tracking (HAT) tool. The HAT tool provides a range of management functions, including the ability to display current and historical information about executing orchestrations. The screen below, for example, illustrates tracing an orchestration as it executes.



Business Rules Services

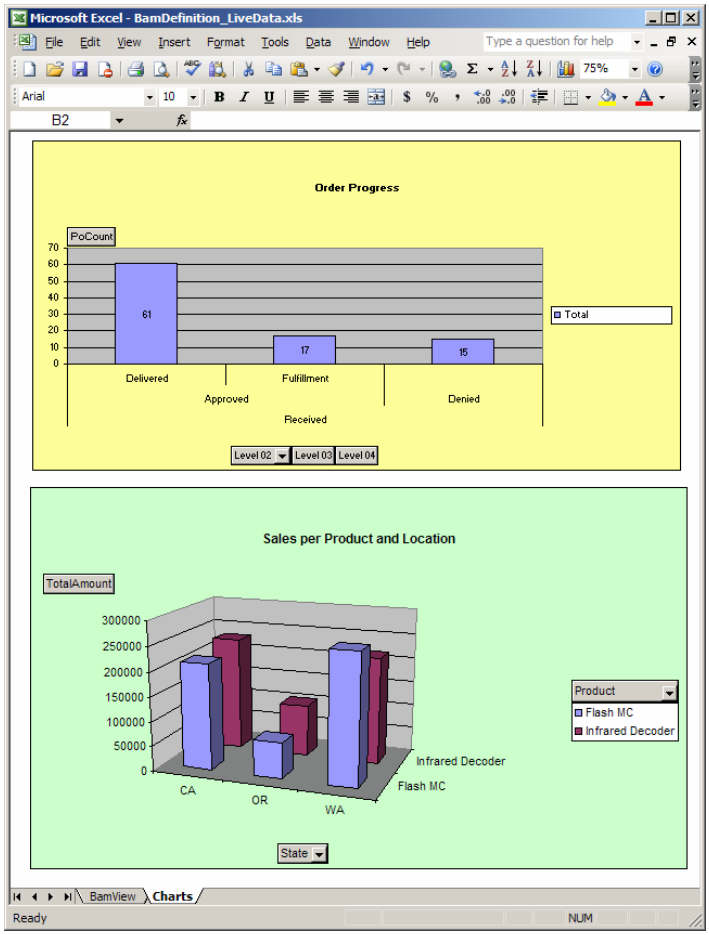
BizTalk Server 2004 includes a Business Rules Engine as a standard part of the product. An orchestration can optionally invoke this engine to evaluate a business rule, then use the result of this evaluation to control decisions in the process it implements. Rules are created using the Business Rule Composer, a graphical tool that allows defining business-oriented vocabularies, then expressing rules in those terms. Assuming an appropriate vocabulary was defined for loan origination, for example, a rule might be expressed as IF LoanAmount > 300000 THEN MortgageType = Jumbo.

Workflow Services

BizTalk Server 2004 includes a workflow framework called Human Workflow Services (HWS). Using this framework, it's possible to create orchestrations that support human involvement in business processes. HWS is accessible via web services, and so processes that use it can interact with people via any client that can make SOAP calls. HWS does not include a graphical tool for defining workflows and specifying work lists, however, nor does it have direct support for building interactive clients. Third-party products provide these services, and developers can also create clients directly.

Process Monitoring Services

For technical monitoring services, BizTalk Server 2004 provides the HAT tool, as described earlier. For business-oriented monitoring, the product also includes a component called Business Activity Monitoring. Using a tool known as the Tracking Profile Editor, a developer can configure an orchestration to make specific information available to the BAM component. Exposed via a web service, this component can be accessed by Excel or other clients to display information about a running process in a way that's meaningful to business users. Using BizTalk-provided Excel add-ins, for instance, a business user can create BAM views that display and update information. The figure below, showing order progress and sales tracking information, illustrates a simple BAM view provided via Excel.



Other Services

Other technologies included in BizTalk Server 2004 include the following:

Business Activity Services, including tools for working with trading partners and managing those relationships.

Enterprise Single Sign-On, which provides a way to map credentials across different authentication systems.

Various accelerators providing support for industry standards such as HL7, RosettaNet, and SWIFT.

Conclusions

The goal of a BPM server in a service-oriented world is to make it easier to create, run, and manage the process logic—the orchestration—that drives a composite application. The technologies these products support today certainly aren't the last word, and more developments are certain to come. One possibility, for example, is that the distinction between BPM servers and application servers may eventually disappear, as BPM and application server capabilities are merged. Whatever happens, the basics of this platform style are in place. As service-oriented composite applications become more widespread, the importance of the technology that supports them is sure to increase.

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, David helps information technology professionals around the world understand, use, and make better decisions about enterprise software technologies. He has presented keynotes and seminars in 35 countries, his books have been published in ten languages, and his consulting clients have included Hewlett-Packard, IBM, Microsoft, Stanford University, Target, and others.