Redbooks Paper

Mark Endrei
Martin Keen
Carla Sadtler

Application Integration patterns

The Application Integration patterns capture commonly observed solution alternatives in the domain of Enterprise Application Integration (EAI). They capture best practices around back-end integration of applications and data, process automation, and workflow implementations involving human interactions. It is important to note that front-end integration such as the composition of a portal or single sign-on across multiple applications is captured by the Access Integration pattern.

The Application Integration pattern can be implemented using any one of the four Process-focused application patterns and the Data-focused application patterns. These various designs provide solution flexibility to address the specific needs of the business process being automated.

Using the pattern

Within the context of Patterns for e-business, Application Integration patterns are used at two different levels:

- To design complex solutions by combining more than one Business pattern. An example might be creating an e-commerce site by combining Self-Service and Information Aggregation patterns.
- To support the implementation of a given Business pattern. For example, it is hard to imagine the Self-Service::Decomposition application pattern without leveraging the Application Integration best practices. Similarly, all the Application patterns for Extended Enterprise are in one sense the implementation of Application Integration across organizational boundaries.

Typically, the requirements that drive application integration call for the seamless execution of multiple applications and access to their respective data in order to automate a complex, new business function. Reliable integration of applications - be they legacy stovepipe applications, packaged software applications, or custom applications - requires the use of proven, repeatable patterns. At its highest level, application integration can be divided into two essentially different approaches:

- **Process-focused integration:** The integration of the functional flow of processing between the applications.
- **Data-focused Integration:** The logical integration of the information used by applications.

Neither approach is necessarily better than the other. Rather, specific integration requirements dictate which approach best solves a given business problem. For example, the integration of an e-commerce application with an Enterprise Resource Planning (ERP) system for a newly created sales order would most definitely be a Process-focused integration activity. However, in the same solution, the master data synchronization of the product catalog between the ERP system and the e-commerce system would be a Data-focused integration activity.

Note: Certain types of integration between applications can be accomplished at the user interface level as well, as covered in the Access Integration pattern.

What's next

Enterprise Application Integration is a complicated undertaking. It requires, first, a thorough understanding of the individual applications being integrated, and also the possible methods that can be used to interconnect them.

Defining the Application Integration patterns

This section defines the Application Integration patterns by documenting the Business and IT drivers that lead to the selection of this pattern, the context within which it can be used, the proposed solution, and examples of its usage.

It also discusses typically observed application integration requirements that can help determine which of the two Application Integration categories (Process-focused or Data-focused) you should use in designing your e-business solution.

Business and IT drivers

Typical Business and IT drivers that result in the selection of this Integration pattern are:

- The business processes need to be integrated with existing business systems and information.
- The business activity needs to aggregate, organize and present information from various sources within the organization.

Context

Application Integration patterns can be observed in solutions that call for close integration with systems and databases that exist within the organization. It serves as a back-end integration pattern, and is critical for the successful implementation of certain Business patterns. For example, solutions that use the Self-Service business pattern or Extended Enterprise business pattern often rely on these same application integration techniques. Similarly, many Custom designs and Composite patterns use Application Integration application patterns.

For example, take the case of a company that wants to integrate their retail and wholesale departments. Currently, both departments have proven IT infrastructures but have no inter connectivity. The Process-focused Application Integration patterns address this problem. These patterns can be applied in a case where the business process needs to be integrated between existing business systems within the organization. The Process-focused Application Integration patterns can be used to integrate the retail ordering and wholesale inventory systems, eliminating ordering lag and providing an up-to-date inventory.

Solution

The Application Integration pattern typically consists of the following elements:

- Business applications and data that need to communicate, interact, and integrate with other business applications and data within the organization
- A network which:
  - Is based on TCP/IP and other Internet technologies, or on proprietary protocols
  - Can be a dedicated LAN or WAN connection
- Other business applications and data which can be:
  - Custom developed systems (old and new)
  - Enterprise Resource Planning systems and other packaged applications, such as SAP, BAAN and PeopleSoft
  - Databases
- Application integration services, which include:
  - Protocol Adapters
  - Message handlers
  - Data transformation
  - Decomposition/recomposition
  - Routing/navigation
  - State management
  - Security

Putting the pattern to use

This is probably one of the most commonly used patterns and it can be observed in any solution where an application needs to integrate with other applications, legacy systems and databases. Examples include:

- An electronics retailer/wholesaler, ABC Electronics from our sample scenario, needs to integrate their retail ordering process with their inventory management system.
- A telecommunications company needs to integrate their online sales systems and their core provisioning systems to improve efficiency and customer service.

Application Integration considerations

Choosing the right Application Integration pattern can only be done in the context of specific solution requirements. These requirements encompass not only the specific needs of application integration to be deployed, but also the constraints posed by the enterprise's existing IT infrastructure and technology investments. This section details considerations to be made and questions to ask in determining the best integration technique for a solution under consideration.

Request for information versus request for processing

Is the integrated solution for informational access only or is it intended to integrate requests for processing? The Process-focused Application Integration patterns are concerned with integration of the functional flow of processing between applications. The Data-focused Application Integration patterns are concerned with integration of the information used by applications.

Foreground versus background integration

Is there a user awaiting the outcome of the operation or is this operation running behind the scenes? An example of a foreground (or real-time) process may be a user retrieving a price quote for the purchase of product whereas a background (or batch) process would be the synchronization of pricing information from the central office out to all of the local stores.

Scope of integration

Does the integration project involve only a single Business pattern, multiple Business patterns, or the creation of an entire e-infrastructure for multiple e-business solutions?

Operation latency (applications and/or data queries)

How long will it take the operation to complete in the application? Operations that can not complete in less than a few seconds typically dictates the need for asynchronous methods of integration. A query on product inventory may be a quick operation whereas the computation of the production plan for the manufacturing of that inventory could take minutes to hours to complete.

Geographic proximity

How close do the applications being integrated reside to one another? Similar to the idea of operation latency, an often overlooked element of the EAI design is the proximity of the participating applications in relation to each other. Integration of applications residing in the same data center has a much smaller integration latency than integration of applications spread around the world.

Process re-engineering

Is there a need to re-engineer business processes or extend an existing business process? Most legacy business processes are embedded within the applications themselves. Business Process Management (BPM) is performed by the existing application(s). Sometimes the EAI effort is merely trying to better integrate functional operations of a disconnected, narrow (or "stovepipe") business process. Other endeavors are more ambitious with the desire to improve business processes through integration.

There are varying degrees of process extensions for application-based BPM:

- Extending the reach of the business process by integration with other applications.
- Joining together two separate application-based business processes into one unified process.
- Separating BPM from application logic by implementing the process in a Process Manager. This option extends the domain of the process by allowing it to encompass any participating application under any specific sequencing and process flow control.

Application portfolio

What is contained in the mix of applications? This might include pre-packaged software, legacy applications, or newly developed applications. One of the most important elements of an EAI project is to survey the application landscape. Some environments are heavily based on pre-packaged software. Other environments are completely homegrown custom applications. Other environments may be a mixture of pre-packaged applications working with legacy homegrown applications.

This survey will detail several key points about the enterprise environment:

- Can the application interfaces be extended as part of the integration activity? Homegrown applications may have standardized interfaces or can be extended to implement standards. Interfaces into pre-packaged applications typically can only be standardized through implementation of sophisticated adapters.
- Is there a central cornerstone application in the enterprise environment or a portfolio of peer applications? Is the business processing focused around one key application (perhaps an ERP system) with all other applications being subservient to that application?
- How many applications are being integrated? For instance, a typical Self-Service application may be integrating the Web application server with one back-end system. At the other end of the spectrum would be a project creating a centralized customer information system that may require feeds from 100+ different applications. There is a significant difference in the selected solution for integrating two applications versus integrating 100+s of applications.

Key characteristics of the application portfolio and enterprise architecture that affect the EAI approach include these:

- Number of applications
- Degree of centralization of the data repositories
- Completeness of the application interfaces
- Conformity of the participating applications to the EAI data and interface model.

Tight coupling versus loose coupling

What is the level of independence between the application implementation and the EAI interface? How likely is it that changes to the application will require changes to the interface or changes to the integration approach? The degree of invasiveness not only affects the application adapter. It can also affect the integration hub processing and even require changes to the partner application. The further across the application integration topology a change ripples, the more expensive this change will be. The degree of invasiveness is often described in terms of coupling (loose coupling versus tight coupling) or black box versus white box approach.

Ideally, the less invasive the integration, the more successful the integration will be in the long-term. This is the primary reason for the use of messaging based integration to isolate as much of the integration processing from any application-specific dependencies. EAI best practices should be employed to ensure that the integration is as non-invasive as possible.

However, EAI projects will vary in the level of independence available based on the completeness of the participating applications functionality and interfaces. For environments with heavy application-specific processing required, it is best to implement these using a sophisticated integration broker component supported by easy to use application development tools. This ensures that future extensions to the integration can be implemented quickly and easily.

What's next

If you have determined that the Application Integration pattern is appropriate for use in your solution, the next step is to select an associated Application pattern based on your specific Business & IT drivers.

If the Application Integration pattern is not appropriate for your development efforts, review the Business patterns to determine which pattern best addresses your e-business needs.

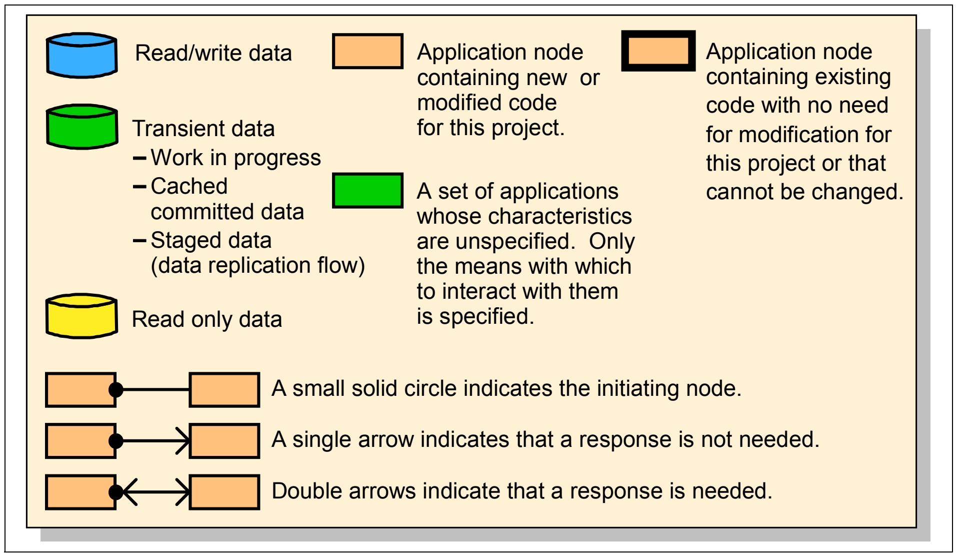
Application patterns

Application patterns for Application Integration can be broadly categorized as either Process-focused or Data-focused. These two categories enable different types of integration functionality.

The focus of this redbook is Application patterns for Process-focused Application Integration. In particular the Serial Process and Parallel Process application patterns and their Workflow variants are explored. A brief overview of the other Application Integration patterns is provided. For full details on the other Application Integration patterns, please see the IBM Patterns for e-business Web site:

http://www.ibm.com/developerWorks/patterns

The diagram conventions shown in Figure 1 are used in the Application patterns that follow.



Read/write data
Transient data
– Work in progress
– Cached committed data
– Staged data (data replication flow)
Read only data
Application node containing new or modified code for this project.
A set of applications whose characteristics are unspecified. Only the means with which to interact with them is specified.
Application node containing existing code with no need for modification for this project or that cannot be changed.
A small solid circle indicates the initiating node.
A single arrow indicates that a response is not needed.
Double arrows indicate that a response is needed.

Figure 1 Application pattern diagram conventions

Process-focused Application patterns

Process-focused Application Integration patterns are observed where multiple automated business processes are combined to yield a new business offering or to provide a consolidated view of some business entity by integrating multiple corporate business systems. An often quoted example is the consolidated view of the state of all relationships of the business with a particular customer.

This mode of integration is highly flexible. In its more sophisticated form it enables “late binding” of the targets of integration and is particularly useful in tying together different platforms and technologies. However, it represents a more difficult design and development task compared to data-focused integration and often requires complex middleware.

The Process-focused Application Integration patterns are presented here in order of increasing flexibility and sophistication. As the Application patterns build on each other, their capabilities and reliance on middleware increase, and they require less application development effort. From the following Application patterns, select the one that best fits your requirements:

- **Direct Connection application pattern:** Message/Call Connection variations
- **Broker application pattern:** Router variation
- **Serial Process application pattern:** Serial Workflow variation
- **Parallel Process application pattern:** Parallel Workflow variation

Business and IT drivers

Table 1 and Table 2 summarize the business and IT drivers for the Process-focused Application Integration patterns and their variations.

Table 1 Business drivers

Business Drivers	**Direct Connection Message variation**	**Direct Connection Call variation**	**Broker Router variation**	**Broker**	**Serial Process**	**Serial Workflow variation**	**Parallel Process**	**Parallel Workflow variation**
Improve the organizational efficiency	✓	✓	✓	✓	✓	✓	✓	✓
Reduce the latency of business events	✓	✓	✓	✓	✓	✓	✓	✓
Support a structured exchange within the organization	✓	✓	✓	✓	✓	✓	✓	✓
Support real-time one-way ’message’ flows	✓		✓	✓	✓	✓	✓	✓
Support real-time request/reply ’message’ flows		✓	✓	✓	✓	✓	✓	✓
Support dynamic routing of ’messages’ to one of many target applications			✓	✓	✓	✓	✓	✓
Support dynamic distribution of ’messages’ to multiple target applications				✓	✓	✓	✓	✓
Support automated coordination of business process flow					✓	✓	✓	✓
Reduce cycle time through parallel execution of portions of a process flow							✓	✓
Support human interaction and intervention within the process flow						✓		✓

Table 2 IT drivers

IT Drivers	**Direct Connection Message variation**	**Direct Connection Call variation**	**Broker Router variation**	**Broker**	**Serial Process**	**Serial Workflow variation**	**Parallel Process**	**Parallel Workflow**
Minimize total cost of ownership (TCO)			✓	✓	✓	✓	✓	✓
Leverage existing skills	✓	✓	✓	✓	✓	✓	✓	✓
Leverage the legacy investment	✓	✓	✓	✓	✓	✓	✓	✓

IT Drivers	Direct Connection Message variation	Direct Connection Call variation	Broker Router variation	Broker	Serial Process	Serial Workflow variation	Parallel Process	Parallel Workflow
Enable back-end application integration	✓	✓	✓	✓	✓	✓	✓	✓
Minimize application complexity	✓	✓	✓	✓	✓	✓	✓	✓
Minimize enterprise complexity			✓	✓	✓	✓	✓	✓
Improve maintainability			✓	✓	✓	✓	✓	✓
Improve flexibility by externalizing process logic from application logic					✓	✓	✓	✓
Support for long running transactions						✓		✓

Quality of Service concerns

This section highlights Quality of Service (QoS) capabilities that are of particular concern in the Process-focused Application Integration domain. A QoS capabilities framework for business integration is based on the following general concerns:

- Operability
- Availability
- Federation
- Performance
- Security
- Standards compliance
- Transactionality

Important: This profile is intended as a rough first guide to QoS concerns which differentiate this domain, suitable for high level architectural design. They are not a substitute for thorough analysis at a later design stage.

The following QoS concerns are of particular importance when working in the Process-focused Application Integration domain.

Operability

The complexity of IT infrastructure is increasing so systems management capabilities are needed to ensure that Application Integration solutions can be managed effectively. For example, clustering solutions may be a consideration for availability management and reducing operational costs.

Performance

High volume workloads are often experienced in the intra-enterprise integration domain, so there is a generally a need to carefully assess the expected workload and to plan for future growth in workload.

Standards compliance

Rather than using different approaches for each application integration exercise that an organization performs, standards need to be identified and applied in order to control development and integration costs.

Private standards are also acceptable when beneficial. Adopting WebSphere MQ, for example, as intra-enterprise message-oriented middleware provides assured, once-only delivery messaging that can be widely used across the organization.

Transactionality

Transaction services are often important in intra-enterprise application integration scenarios in order to preserve data integrity and to avoid data loss. Consider using transaction management products that work with XA compliant resource managers to provide a commit and rollback facility, ensuring that either all resource updates are completed or all updates are rolled back.

Direct Connection Application pattern

The Direct Connection application pattern represents the simplest interaction type and is based on a 1-to-1 topology. It allows a pair of applications within the organization to directly communicate with each other. Interactions between a source and a target application can be arbitrarily complex. Generally, complexity can be addressed by breaking down interactions into more elementary interactions.

More complex point to point connections will have modeled connection rules such as business rules associated with them, as shown in Figure 2. Connection rules are generally used to control the mode of operation of a connector depending on external factors. Examples of connection rules are:

- Business data mapping rules (for adapter connectors)
- Autonomic rules (such as priority in a shared environment)
- Security rules
- Capacity and availability rules

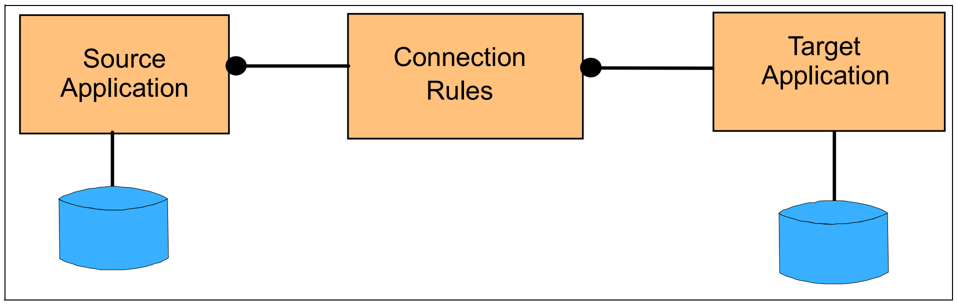


Figure 2 Direct Connection application pattern

> **Note:** The Connection Rules component is not needed when there are no modeled rules associated with the connection.

The Direct Connection application pattern has two variations:

- Message Connection variation
- Call Connection variation

All applications of the Direct Connection application pattern will be one variation or the other. The variation required depends on whether the initiating source application needs an immediate response from the target application in order to continue with execution.

Both variations may be used either with synchronous or asynchronous communication protocols. However, there are preferences for a specific protocol type depending on the variation. For example, the Call Connection variation has a more natural fit with synchronous protocols while the Message Connection variation favors asynchronous protocols.

We examine these two variations in more detail later in this section.

Business and IT drivers

The business and IT drivers for choosing the Direct Connection application pattern are to:

- Improve the organizational efficiency
- Reduce the latency of business events
- Support a structured exchange within the organization
- Support real-time one-way message flows
- Support real-time request/reply message flows
- Leverage existing skills
- Leverage the legacy investment
- Enable back-end application integration
- Minimize application complexity

The primary goal is to allow one application to gain direct and real-time access to another in order to reduce the latency of business events.

Solution

This Application pattern, as shown in Figure 2 on page 10, is divided into a number of logical components:

- The Source Application tier represents one or more applications that are interested in initiating an interaction with the target application.
- The Connection is the line between the source application and the target application representing a point-to-point connection between the two applications.
- The Connection Rules tier represents any business rules associated with the connection, such as data mapping rules and security rules.
- The Target Application tier represents a new application, a modified existing application, or an unmodified existing application. This application is responsible for implementing the necessary business services.

Guidelines for use

Direct integration between applications can be inflexible, in that any changes to one application may have knock-on effects on other applications. Changes to the target application may also require changes to the source application. Such changes can become both expensive and time consuming, especially when the target application is being accessed by a number of different source applications.

Different IT departments may also be responsible for developing and maintaining the source and target applications. Under such a scenario, development might be difficult to coordinate, especially if the interfaces between the applications being integrated are not properly defined and documented. Because of this, it is important to clearly define such interfaces in advance.

Benefits

The Direct Connection application pattern offers the following benefits:

- It works with applications that have simple integration requirements with only a few back-end applications.
- It increases the organizational efficiency and reduces the latency of business events by providing real-time access to business data and business logic, and avoiding manual synchronization of data between applications.
- Direct access to back-end applications reduces the duplication of business logic across multiple tiers. As a result, changes to business logic can be made in one tier rather than in multiple applications.
- It can enable re-use of investments already made with the organization.

Limitations

Although this is a reasonable starting Application pattern for integrating applications in a one to one relationship with one another, this pattern will result in a many to many “spaghetti” configuration with point to point integration mappings for each application pair. Also, the expansion of this implementation into a multi-point configuration will require additional application logic to handle the coordination.

This pattern cannot be used for intelligent routing of requests, decomposition and re-composition of requests, and for invoking complex business process workflow as a result of a request from another application. Under such circumstances, you should consider a more advanced Application pattern, such as Broker or Serial/Parallel Process.

Putting the Application pattern to use

ABC Electronics, an electronics retailer/wholesaler, wants to integrate their retail and wholesale departments. Currently, both organizations have proven IT infrastructures but have no interconnectivity. The first process ABC Electronics wants to focus on is the inventory and order replenishment process. Currently, the items sold are tallied at the end of the month by the retail ordering process and delivered to the wholesale organization by internal mail. This creates a lag in the inventory replenishment process and causes many out of stock situations. A primary business goal is to minimize the loss of sales due to out of stock situations. To meet these requirements ABC Electronics chooses the Direct Connection application pattern.

Direct Connection: Message Connection variation

The Message Connection variation, shown in Figure 3, applies to solutions where the business process does not require a response from the target application within the scope of the interaction.

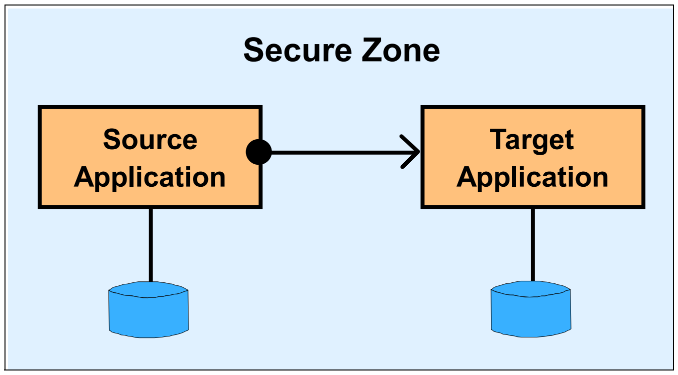


Figure 3 Message Connection variation

> **Note:** We chose not to show the connection rules box in Figure 3 because we want to focus on the connection itself.

Business and IT drivers

The business and IT driver for choosing the Message Connection variation of the Direct Connection application pattern is to:

- Support real-time one-way message flows

The main driver for selecting this variation is when the business process has no interest in the result of the operation. This variation also has the most natural fit when message-oriented middleware is used, such as IBM WebSphere MQ.

Putting the Application pattern to use

In our scenario the retail department of the ABC Electronics organization needs to notify the wholesale department to update their inventory records when
a part needs to be ordered. The retail department does not require any acknowledgement of this request. To meet these requirements, ABC Electronics chooses the Message Connection variation of the Direct Connection application pattern.

Direct Connection: Call Connection variation

The Call Connection variation, shown in Figure 4, applies to solutions where the business process depends on the target application to process a request and return a response within the scope of the interaction.

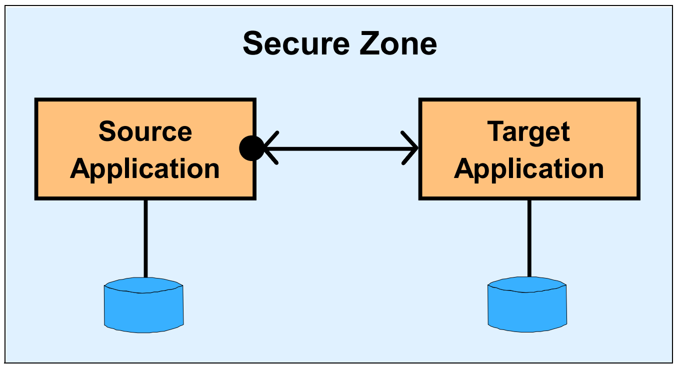


Figure 4 Call Connection variation

Note: We chose not to show the connection rules box in Figure 4 because we want to focus on the connection itself.

Business and IT drivers

The business and IT driver for choosing the Call Connection variation of the Direct Connection application pattern is to:

- Support real-time request/reply message flows

The main driver for selecting this variation is when the business process does require a result message from the interaction.

Putting the Application pattern to use

In our scenario the retail department of the ABC Electronics organization needs to be advised by the wholesale department of the expected delivery date of a part on order that is out of stock with the retail department. To meet these requirements, ABC Electronics chooses the Call Connection variation of the Direct Connection application pattern.

Broker Application pattern

The Broker application pattern, shown in Figure 5, is based on a 1-to-N topology that separates distribution rules from the applications. It allows a single interaction from the source application to be distributed to multiple target applications concurrently. This application pattern reduces the proliferation of point-to-point connections.

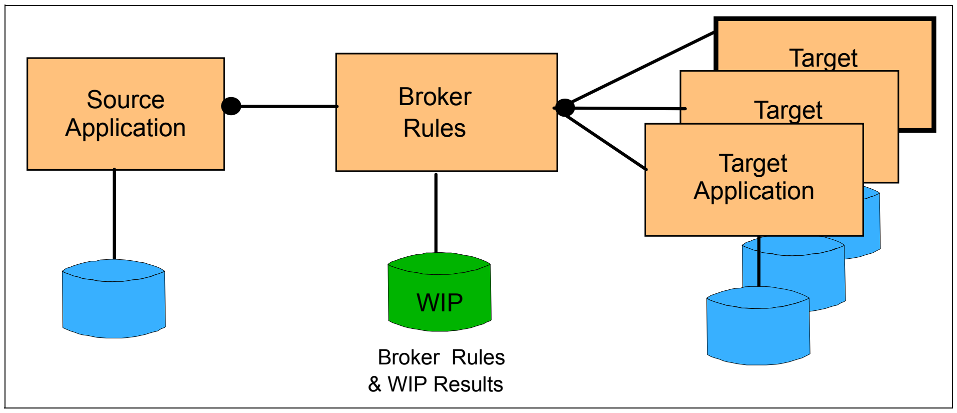


Figure 5 Broker application pattern

The Broker application pattern applies to solutions where the source application starts an interaction that is distributed to multiple target applications that are within the organization. It separates the application logic from the distribution logic based on broker rules. The decomposition/ recomposition of the interaction is managed by the broker rules tier.

The Broker pattern reuses the Direct Connection pattern to provide connectivity between the tiers. The Broker Rules may support Message variation or Call variation (or both variations) of the Direct Connection pattern.

The Broker application pattern was previously known as the Aggregator application pattern for read intent calls and the Broker application pattern for Messages and update intent calls. However, this distinction was found to be of insufficient value to warrant a separate pattern — and so it has been dropped from the revised PI patterns.

Business and IT drivers

The primary business driver for selecting this Application pattern is to allow one application to interact with one or more of multiple target applications. Using a hub-and-spoke architecture instead of a point-to-point architecture allows for the seamless integration of applications while minimizing the complexity. A request for information can be routed to one of many targets or simultaneously to multiple targets. The resulting request message can be decomposed into multiple request messages, and the reply messages then recomposed into a single reply message using appropriate recomposition rules.

This externalization of routing, decomposition, and recomposition rules from individual source and target applications increases the maintainability and flexibility and reduces the enterprise wide integration complexity.

This Application pattern is particularly important when a processing request requires execution of multiple interactions concurrently, or where the source application should be relieved of the need to know anything about its targets.

The primary IT driver for selecting this Application pattern is to allow loose coupling of clients and services with minimum modification to each. The solution should allow for multiple transmission protocols to be used and for transformation of protocols between client and service.

Solution

This Application pattern, as shown in Figure 6 on page 16, is divided into a number of logical components:

- The Source Application tier represents one or more applications that are interested in interacting with the target applications.
- The Broker Rules tier reduces the proliferation of direct connections. In addition, it supports message routing, decomposition and recomposition, message enhancement and transformation. These rules are often captured as business rules that govern the behavior of the broker tier. This tier also uses a work-in-progress data store to retain the intermediate results from the responses coming back from target applications until all the necessary responses are received.
- The Target Application tier represents new, modified existing, or unmodified existing applications. These applications are responsible for implementing the necessary business services.

Guidelines for use

To increase the flexibility of the solution and responsiveness to changing business requirements, it is recommended that particular attention is paid to definition of reusable messages/services that pass through the Broker tier.

Robust transaction processing systems should be used to implement the back-end applications to ensure availability, scalability, and performance.

A decomposition implementation (one source call to multiple target calls) requires state persistence and re-composition of the response messages.

Standards should be used where possible to minimize future changes required to the source and target applications.

Benefits

The benefits of this Application pattern are:

- It allows the integration of multiple, diverse applications.
- It minimizes the impact to existing applications.
- The Broker tier provides routing services, relieving the source application from being aware of the target application.
- The Broker tier can provide transformation services that allow the source and target to use different communication protocols.
- The Broker tier can provide decomposition/recomposition of messages, allowing one request from the source to be satisfied using multiple target applications. The fact that the response is a composite of multiple requests and responses is hidden from the source application.
- The Broker tier minimizes the impact of changes in location of the target application.

Limitations

Logic must be implemented at the broker for routing and decomposition/recomposition tasks.

Putting the Application pattern to use

ABC electronics consists of multiple Retail stores and Wholesale departments. The Retail stores get their supplies from the Wholesale departments and have a need to request the delivery dates of those supplies before ordering. Currently there is no integration of the Retail and Wholesale applications. All interaction between the two are done over the phone or by mail. A solution must be found to allow Retail stores to request delivery dates from the Wholesale departments.

To eliminate the need for the Retail departments to know which Wholesale department carries which supplies, a Broker is needed to take incoming requests and direct them based on part numbers to the Wholesale department that carries them. In the event that a part is carried by multiple Wholesale departments, the broker must get delivery dates from each and return the best date and the Wholesale department that can supply it to the Retail department.

Broker: Router variation

The Router variation of the Broker application pattern, shown in Figure 6, applies to solutions where the source application initiates an interaction that is forwarded to at most one of multiple target applications.

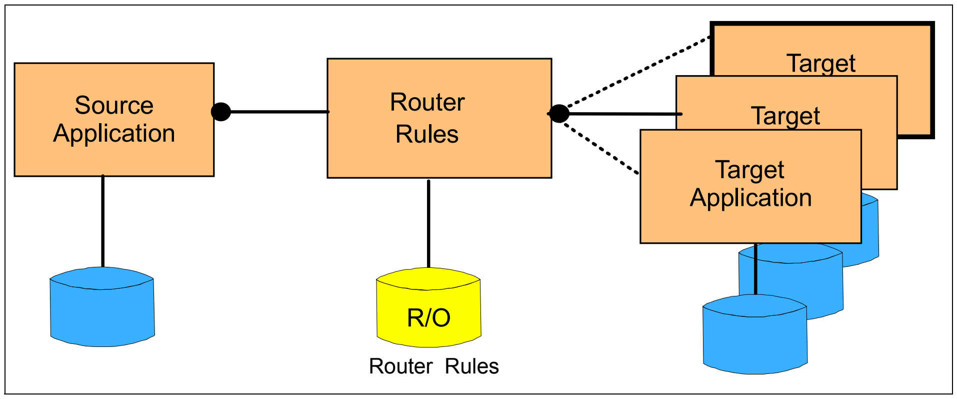


Figure 6 Router variation

Where the Broker application pattern enables 1:N connectivity, the Router application pattern enables 1:1 connectivity where the Router Rules tier selects the target.

The Router variation of the Broker application pattern was previously known as the Router variation of the *Aggregator* application pattern.

Business and IT drivers

The primary business driver for selecting this Application pattern is similar to that of the Broker application pattern. The difference lies in the fact that the Router tier routes the request to only one of multiple target applications. The requirement for transformation of message and interface format still applies. Externalizing the routing from individual source and target applications increases the maintainability and flexibility and reduces the enterprise wide integration complexity.

This Application pattern is particularly important when a processing request requires the source application to be relieved of the need to know anything about its targets.

The primary IT driver for selecting this Application pattern is to allow loose coupling of clients and services with minimum modification to each. The solution should allow for multiple transmission protocols to be used and for transformation of protocols between client and service.

Solution

This Application pattern provides a routing function to allow any attached (initiating) application using a single router link to connect to one of multiple target applications. While access to multiple applications is supported, at any given time an application is connected to only one other application.

This Application pattern, as shown in Figure 6 on page 16, is divided into a number of logical components:

- The Source Application tier represents one or more applications that are interested in interacting with the target applications, one target at a time.
- The Router Rules tier represents any business rules associated with the message handling, such as routing and transformation. It receives requests from multiple source applications and routes them intelligently to the appropriate target applications. The resulting integration is essentially a point-to-point connection between source and target. This tier implements minimal business logic.
- The Target Application tier represents new, modified existing, or unmodified existing applications. These applications are responsible for implementing the necessary business services.

Guidelines for use

The guidelines for this application pattern are the same as those for the Broker application pattern.

Benefits

The benefits of this Application pattern are:

- It allows the integration of multiple, diverse applications
- It minimizes the impact to existing applications
- It provides routing services, relieving the source application from being aware of the target application.
- It provides transformation services that allow the source and target to use different communication protocols.
- The use of a router minimizes the impact of changes in location of the target application.

Limitations

With the Router variation, there is limited ability in the router to manipulate the requests. It performs intelligent routing and protocol transformation, but does not have the ability to send simultaneous requests to the target applications based on one incoming request, nor does decomposition / recomposition ability.

Putting the application pattern to use

ABC electronics consists of multiple Retail stores and Wholesale departments. The Retail stores get their supplies from the Wholesale departments and have a need to request the delivery dates of those supplies before ordering. Currently there is no integration of the Retail and Wholesale applications. All interaction between the two are done over the phone or by mail. A solution must be found to allow the Retail stores to request delivery dates from the Wholesale departments. To eliminate the need for the Retail departments to know which Wholesale department carries which supplies, a Router is needed to take incoming requests and direct them based on part numbers to the Wholesale department that carries them. This differs from the example outlined in the Broker pattern in that only one Wholesale department will carry a part. There is no need to distribute one request to multiple Wholesale departments simultaneously to see who can supply the part at the earliest date.

Serial Process Application pattern

The Serial Process application pattern, shown in Figure 7, extends the 1-to-N topology provided by the Broker application pattern by facilitating the sequential execution of business services hosted by a number of target applications. Thus it enables the orchestration of a serial business process in response to an interaction initiated by the source application.

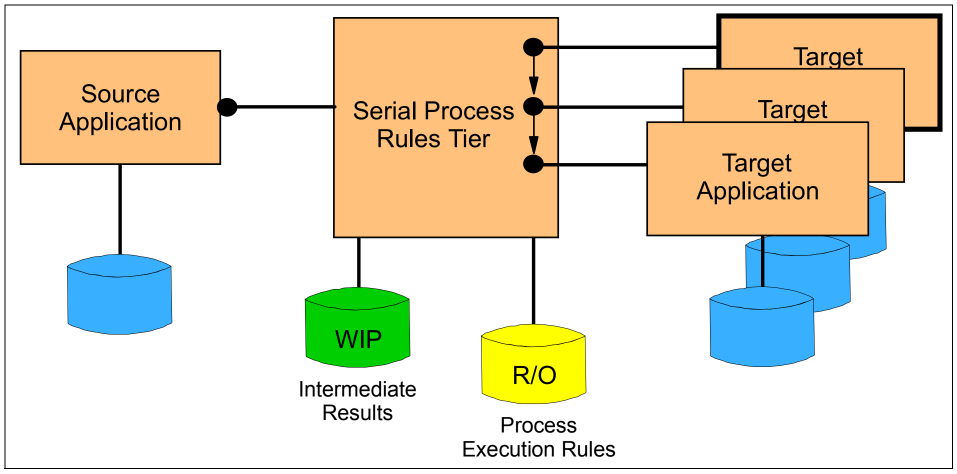


Figure 7 Serial Process application pattern

Business and IT drivers

The primary business driver for selecting this Application pattern is to support the composition of end-to-end business process flows by leveraging business services implemented by a number of target applications.

From an IT perspective the key driver for selecting this Application pattern is improving the flexibility and responsiveness of IT by externalizing the process flow logic from individual applications.

Solution

The Serial Process application pattern is broken down into three logical tiers:

- The Source Application tier is the same as for the Broker application pattern.
- The Serial Process Rules tier supports most of the services provided by the broker tier in the broker application pattern, including routing of requests, protocol conversion, message broadcasting, and message decomposition / recomposition. In addition, it supports the separation of business process flow logic from individual application logic. The process logic is governed by serial process rules that define execution rules for each target application, together with control flow and data flow rules. It may also include any necessary adapter rules. The combination of these process execution rules are stored in read-only databases. This externalization of process flow logic is essential for the implementation of a flexible and responsive IT environment that can respond quickly to changing business needs. It also makes it possible to compose new end-to-end processes by combining different business services provided by different applications. Finally, this tier utilizes a work-in-progress (WIP) database to store the intermediate results from the execution of different process steps.
- The Target Application tier is the same as for the Broker application pattern.

Guidelines for use

The flexibility and responsiveness provided by this Application pattern is heavily dependent on the externalization of process execution logic from individual applications. Applications designed based on a Service Oriented Architecture (SOA) approach, with well defined coarse-grained business services that represent a unit of work, are better suited for participation in this Application pattern. One must be able to compose these business services into an end-to-end process flow. A given service may need to participate in more than one end-to-end process.

Typically, legacy applications are not designed with this thinking in mind. Similarly, many of the legacy applications have significant amounts of process logic embedded within them. These constraints in existing environments may pose challenges to fully implementing the vision promised by this Application pattern. Careful refactoring of legacy and packaged applications by wrappering them into business services is a good starting point for the eventual wide-spread implementation of this Application pattern within an enterprise.

Composition of process flows by tieing together different applications may introduce the need for compensating transaction support. This is especially the case when certain participating target applications do not leverage XA compliant transaction processing engines. In such cases, it may be necessary to design compensating transaction pairs for every affected transaction and execute them if there is a need to reverse a particular portion of the process flow. Participating legacy and packaged target applications may need to be modified to introduce compensating transactions if they do not already implement such mechanisms.

Finally, in selecting middleware products that facilitate automation of business processes pay particular attention to the Business Process Management capabilities supported by the business process design tools and the process execution engines. The eventual goal is to enable business users to compose business processes and make necessary changes with minimal involvement from IT professionals. The business processes thus defined must be easily exported into a process execution engine. More sophisticated business process management tools allow for the definition of metrics during the process design to measure the effectiveness of process implementation and support monitoring of the metrics in the process execution engine.

Benefits

The Serial Process application pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows and by externalizing process logic from individual applications.

In addition, it provides a foundation for automated support for Business Process Management that enables the monitoring and measurement of the effectiveness of business processes.

Limitations

This Application pattern is ideally suited for straight through processing where human interactions are not necessary to complete an end-to-end process. If support for human interactions are needed to complete certain process steps, consider the Workflow variation of this Application pattern.

Similarly it does not support the parallel execution of multiple tasks. Under such circumstances, consider the more advanced Parallel Process application pattern discussed later in this chapter.

Putting the Application pattern to use

ABC Electronics, an electronics retailer/wholesaler, wants to integrate their retail department with their two inventory wholesale departments, namely Wholesale A and Wholesale B. Currently, these three departments have proven IT infrastructures but have no interconnectivity. ABC Electronics wants to focus on automating the inventory replenishment process. Typically, the retail department places orders with Wholesale A. However, when the Wholesale A is unable to guarantee delivery within 7 days, Wholesale B is contacted to check the anticipated delivery date and the order is placed with departments that guarantees the shortest delivery date.

To meet these business process automation requirements, ABC Electronics chooses the Serial Process application pattern. The primary driver for this selection is the need for externalization of process logic from individual application thus promoting flexibility and responsiveness to changing business needs.

Serial Process: Workflow variation

The Serial Process Workflow variation of the Serial Process application pattern, shown in Figure 8, extends the basic serial process orchestration capability by supporting human interaction for completing certain process steps.

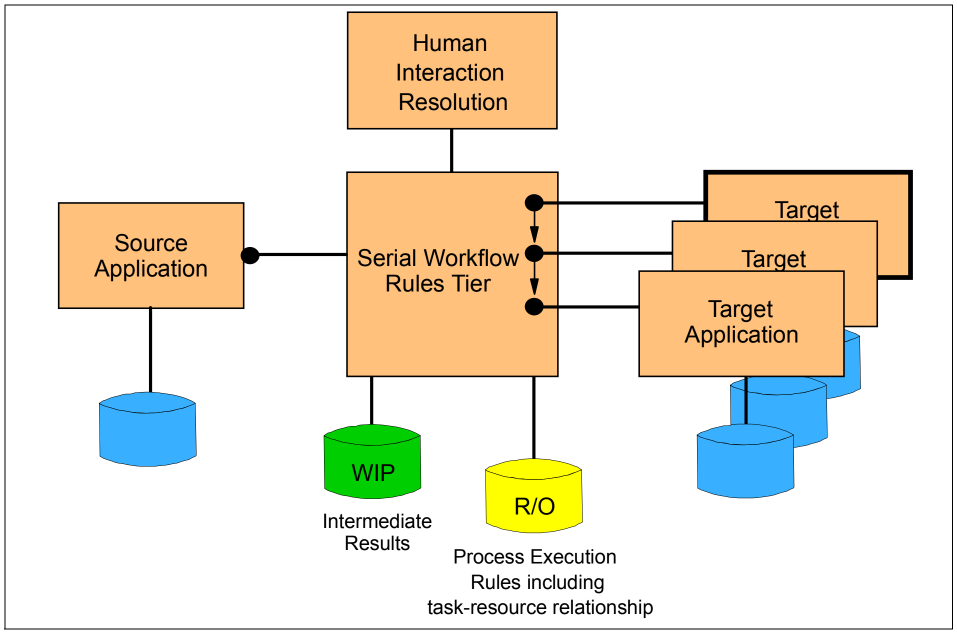

Human Interaction Resolution
Source Application
Serial Workflow Rules Tier
Target
Target
Target Application
WIP
Intermediate Results
R/O
Process Execution Rules including task-resource relationship

Figure 8 Serial Workflow variation

Business and IT drivers

All the business and IT drivers listed under the Serial Process application pattern apply to this variation as well. The additional business driver for selecting this variation is the need to support human interaction and intervention within the process flow. Support for long running transactions is another IT driver, which is often a pre-requisite for the automation of complex process flows involving human interaction.

Solution

The Serial Workflow variation is broken down into three logical tiers:

- The Source Application tier is the same as for the Serial Process application pattern.
- The Serial Workflow Rules tier supports all the services provided by the serial process rules tier within the Serial Process application pattern. In addition, it supports certain tasks within the process to be routed to human actors for completion. To accomplish this the process execution rules are augmented with task-resource relationships that define which resources are capable of performing which tasks. In this context:
  - A task is a portion of the end-to-end process.
  - Resources are capable of executing these tasks.
  - People, departments, and target applications can all be resources capable of executing a particular task.

  This tier resolves the task-resource relationship during the execution of a process. If the need for human interaction is identified, the task is added to a worklist associated with an individual or a department as a work item to be completed by a human and the process is typically suspended until the completion of the task.

  Finally, this tier provides support for long running transactions and utilizes a work-in-progress (WIP) database to store the intermediate results from the execution of different process steps until the complete execution of the end-to-end process.
- The Target Application tier is the same as for the Serial Process application pattern.

Guidelines for use

The following guidelines apply to this variation in addition to the guidelines that are documented under the Serial Process application pattern.

It is recommended that people-based exception handling be implemented for majority of the automated tasks within the process. In other words, if an automated task reaches certain error conditions, human actors must be able to intervene and handle the exceptions.

Benefits

The Serial Workflow application pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows that externalize process logic from individual applications. Further flexibility is introduced by the externalization of task-resource resolution rules.

In addition, it provides a foundation for automated support for Business Process Management that enables monitoring and measurements of the effectiveness of business processes.

Limitations

It does not support the parallel execution of multiple tasks. Under such circumstances, consider the more advanced Parallel Process application pattern and Parallel Workflow variation discussed later in this chapter.

Putting the Application pattern to use

ABC Electronics, an electronics retailer/wholesaler, wants to integrate their retail department with their two wholesale departments, namely Wholesale A and Wholesale B. Currently, these three departments have proven IT infrastructures but have no interconnectivity. ABC Electronics wants to focus on automating the inventory replenishment process. Typically, the retail department places orders with Wholesale A. However, when the Wholesale A is unable to guarantee delivery within 7 days, Wholesale B is contacted to check the anticipated delivery date.

The main change from the scenario used in Serial Process application pattern section is documented below. If both Wholesale A and Wholesale B cannot offer delivery within 7 days a retail dept. manager must review the shortest anticipated delivery date proposed by the wholesale department systems and approve the order before placing the same. The intent of this review is to determine if other sourcing options must be considered.

To meet these business process automation requirements, ABC Electronics chooses the Serial Workflow variation of Serial Process application pattern. The primary drivers for this selection include the need for externalization of process logic from individual application thus promoting flexibility and responsiveness to changing business needs and need to support human interaction.

Parallel Process Application pattern

The Parallel Process application pattern, shown in Figure 9, extends the basic serial process orchestration capability provided by the Serial Process application pattern by supporting parallel (concurrent) execution of the sub-processes.

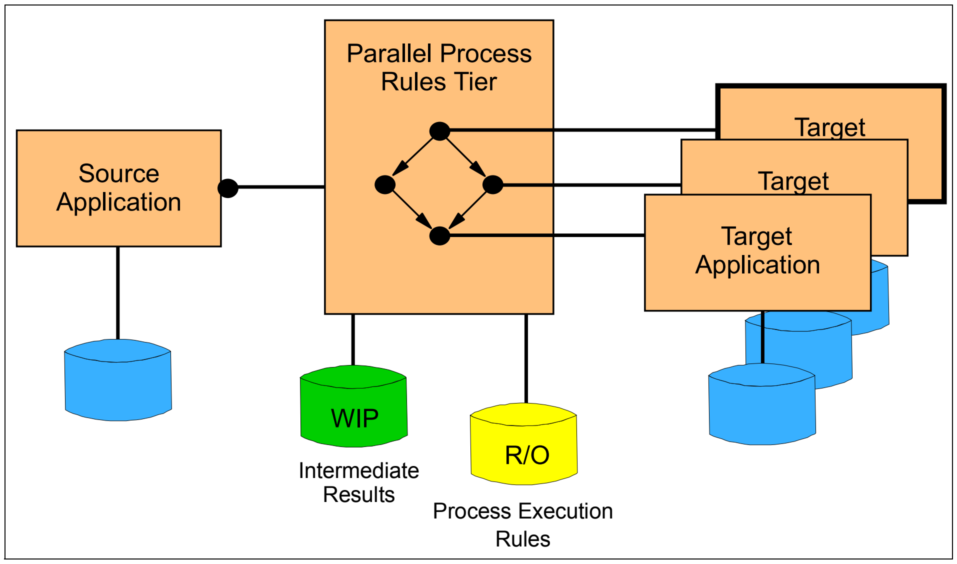


Figure 9 Parallel Process application pattern

Business and IT drivers

All the business and IT drivers listed under the Serial Process application pattern apply to this Application pattern as well. The additional business driver for selecting this pattern is the need to reduce cycle time through the parallel execution of certain portions of the process flow.

Solution

The Parallel Process application pattern is broken down into three logical tiers:

- The Source Application tier is the same as for the Serial Process application pattern.
- The Parallel Process Rules tier supports all the services provided by the serial process rules tier within the Serial Process application pattern. In addition, the interaction initiated by the source application may control parallel (concurrent) sub-processes on multiple target applications. Each sub-process may consist of a sequence of operations executed in succession on a target application. This parallelism requires that additional start and join conditions be defined for sub-processes executing in parallel. This requires sophisticated runtime engines that can initiate parallel threads of control, and ensure these threads join upon completion, and manage them as a unit (for example to allow cancellation of the process, or to report its status).
- The Target Application tier is the same as for the Serial Process application pattern.

Guidelines for use

The following guidelines apply to this variation in addition to the guidelines that are documented under the Serial Process application pattern.

The implementation of parallel processes without sufficient support from the selected runtime engine would require the development of excessive custom code. The need for parallel process execution must be analyzed before middleware selection decisions are finalized.

Judicious use of parallelism is a powerful tool for reducing the cycle time of a process in the right circumstances. However, in practice, it is critical to ensure that all the error scenarios are carefully analyzed, and that the impact of these scenarios upon the end user experience is thoroughly understood. The number of error scenarios and processing complexity increases exponentially with the degree of parallelism. Hence best practice is to start serial, and introduce limited parallelism only where there is a clear and worthwhile benefit.

Benefits

In addition to providing all the benefits provided by the Serial Process application pattern, this pattern provides a foundation for the reduction of cycle times by implementing parallel processes.

Limitations

Parallel processes are more complex to design, test, and operate than serial processes.

In addition, this Application pattern is ideally suited for straight through processing where human interactions are not necessary to complete an end-to-end process. If support for human interactions are needed to complete certain process steps, consider the Workflow variation of this Application pattern.

Putting the Application pattern to use

ABC Electronics, an electronics retailer/wholesaler, wants to integrate their retail department with their two wholesale departments, namely Wholesale A and Wholesale B. Currently, these three departments have proven IT infrastructures but have no interconnectivity. ABC Electronics wants to focus on automating the inventory replenishment process.

The main difference from the scenario used in Serial Process and Serial Workflow application patterns sections is that here both Wholesalers are queried in parallel to find who offers the shortest delivery time. In other words here Wholesale Dept. A is not considered as the defacto supplier of parts. The order is then automatically placed with the wholesale department that offers the shortest delivery date.

To meet these business process automation requirements, ABC Electronics chooses the Parallel Process application pattern. The primary drivers for this selection include the need for externalization of process logic from individual application thus promoting flexibility and responsiveness to changing business needs and the need for reducing cycle time of queries by simultaneously enquiring the two departments for the best delivery date.

Parallel Process: Workflow variation

The Parallel Process Workflow variation of the Parallel Process application pattern, shown in Figure 10, extends the basic parallel process orchestration capability by supporting human interaction for completing certain process steps. This is the most sophisticated Process-focused Application pattern in the domain of Application Integration patterns.

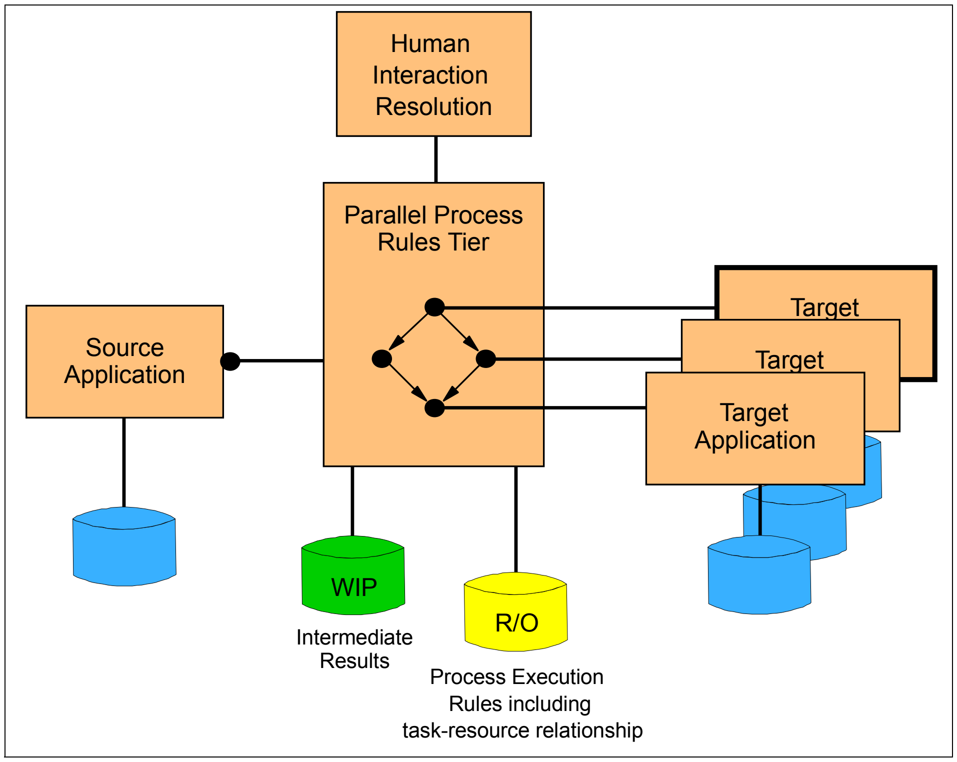

Human Interaction Resolution
Parallel Process Rules Tier
Source Application
Target
Target
Target Application
WIP
Intermediate Results
R/O
Process Execution Rules including task-resource relationship

Figure 10 Parallel Workflow variation

Business and IT drivers

All the business and IT drivers listed under the Parallel Process application pattern apply to this variation as well. The additional business driver for selecting this variation is the need to support human interaction and intervention within the process flow. Support for long running transactions is another IT driver, which is often a pre-requisite for the automation of complex process flows involving human interaction.

Solution

The Parallel Workflow variation is broken down into three logical tiers:

- The Source Application tier is the same as for the Parallel Process application pattern.
- The Parallel Workflow Rules tier supports all the services provided by the parallel process rules tier within the Parallel Process application pattern. In addition, it supports certain tasks within the process to be routed to human actors for completion. To accomplish this the process execution rules are augmented with task-resource relationships that define which resources are capable of performing which tasks. In this context:
  - A task is a portion of the end-to-end process.
  - Resources are capable of executing these tasks.
  - People, departments, and target applications can all be resources capable of executing a particular task.

  This tier resolves the task-resource relationship during the execution of a process. If the need for human interaction is identified, the task is added to a worklist associated with an individual or a department as a work item to be completed by a human and the process is typically suspended until the completion of the task.

Finally, this tier provides support for long running transactions and utilizes a work-in-progress (WIP) database to store the intermediate results from the execution of different process steps until the complete execution of the end-to-end process.

- The Target Application tier is the same as for the Parallel Process application pattern.

Guidelines for use

The following guidelines apply to this variation in addition to the guidelines that are documented under the Parallel Process application pattern.

It is recommended that people-based exception handling be implemented for all automated tasks within the process. In other words, if an automated task reaches certain error conditions, human actors must be able to intervene and handle the exceptions.

Benefits

The Parallel Workflow application pattern improves the flexibility and responsiveness of an organization by implementing end-to-end process flows that externalize process logic from individual applications. Further flexibility is introduced by the externalization of task-resource resolution rules.

It supports the reduction of cycle time by supporting parallel execution of portions of a process flow.

In addition, it provides a foundation for automated support for Business Process Management that enables monitoring and measurement of the effectiveness of business processes.

Limitations

Only a few middleware products are capable of supporting all the capabilities needed to realize this Application pattern. If this Application pattern is implemented using middleware products that don't support the necessary capabilities, the implementation could be very complex.

Putting the Application pattern to use

ABC Electronics, an electronics retailer/wholesaler, wants to integrate their retail department with their two wholesale departments, namely Wholesale A and Wholesale B. Currently, these three departments have proven IT infrastructures but have no interconnectivity. ABC Electronics wants to focus on automating the inventory replenishment process.

The main difference from the scenario used in Parallel Process application patterns sections is document here. In this scenario both Wholesalers are queried in parallel to find who offers the shortest delivery time. The order is then automatically placed with the wholesale department that offers the shortest delivery date, unless the shortest delivery time received from the wholesale departments exceeds 10 business days. In that case, a human intervention is required by the Retail Department Manager to review the anticipated delivery date to determine other sourcing options which must be considered.

To meet these business process automation requirements, ABC Electronics chooses the Parallel Workflow variation of the Parallel Process application pattern. The primary drivers for this selection include the need for externalization of process logic from individual application thus promoting flexibility and responsiveness to changing business requirements, the need for reducing cycle time of queries by simultaneously enquiring the two departments for the best delivery date, and the need for supporting human interaction during the execution of the process flow.

Data-focused Application patterns

When applications need to share information rather than coordinate processing, data-focused application integration is more appropriate than a process-focused approach. Note, however, that when the frequency of data update is extremely high (for example, when integrating an order entry system with a back-end ERP system), process integration is the best solution. When this is not the case, however, integration of (application) data repositories is handled outside of any specific application request.

In delineating Data-focused Application Integration patterns, two key environmental questions should be asked:

- Is the enterprise data topology centralized or decentralized?
  - **Centralized:** This integration effort will bring about centralized access to all or a subset of the enterprise data model.
  - **Decentralized:** Applications will retain their isolated repositories but now with cohesion based on data integration.
- What is the database affinity type?
  - **Homogeneous:** All repositories are of the same type.
  - **Multi-vender Relational:** All repositories are relational with ODBC/JDBC support for interoperability but are from different vendors.
  - **Heterogeneous Structured:** repositories are not all relational, but all have a structured layout.
  - **Structured/Non-Structured:** The need to integrate non-structured (for example, free-form text) with structured data sources.

Refer to the IBM Patterns for e-business Web site for further details:

http://www.ibm.com/developerWorks/patterns

Previous Application Integration patterns

Table 3 provides an overview of the relationship between the previous Process-focused Application Integration patterns and the revised Process-focused Application Integration patterns presented in this chapter:

- Direct Connection is retained for application coordinated requests.
- Transactional is now a quality of service. Transactionality may apply to all of these patterns, so it is applied as a quality of service rather than being a separate pattern.
- Aggregator/Broker are combined into Broker for broker coordinated requests.
- Managed Process is split into Serial Process and Parallel Process for process managed coordinated requests.
- The read-only versus read/write classification used with old patterns is not used with the new patterns, since:
  - For Transactional and Managed Process, read-only is not applicable.
  - For Direct Connection, the same pattern applies in both cases.
  - For Aggregator/Broker, the observed patterns are identical.

Table 3 Relationship to old Process-focused Application Integration patterns

	Old Pattern		**New Pattern**
	Information Request (R/O)	**Processing Request (R/W)**	
Application Coordinated	Direct Connection	Direct Connection	Same
Transactional Coordinated	Not applicable	Transactional	Now a Quality of Service
Broker Coordinated	Aggregator	Broker	Broker
Process Managed Coordinated	Not applicable	Managed Process	Split into: ► Serial Process ► Parallel Process

Notices

Trademarks