Enterprise Application Integration becomes relevant when state changes of one application have an effect on the operation of another application. The degree of dependence can vary from a simple data lookup to a data change or the initiation of a complex business process involving multiple systems.

Enterprise Application Integration

The degree of complexity in integration requirements influences the complexity of your integration architecture. This paper focuses on comparing:

- Integration mechanisms (data manipulations, Application Programming Interface manipulations, business process changes)
- Orchestration approaches (batch processes, orchestration engines, service bus implementations)
- Integration propagation times (real time, scheduled, on demand)

Integration Mechanisms

Application integration mechanisms include a variety of techniques: application data level integration, application API level integration, and business process level integration.

1 **Data Level Integration** occurs through data access of an application's data storage. In today's data storage environment this often means accessing relational databases. This integration approach has the potential benefit of greater performance. Direct access to the data means that there is no application API adding overhead and an industry of associated tools and techniques has grown to support this approach. These tools are in the business intelligence ETL space, but integration vendors (such as TIBCO) also offer tools that monitor changes to database tables and raise events when these changes occur.

"Integration mechanisms can satisfy an array of business requirements ranging from basic hardware services to highly scalable development platforms suitable for implementation of new applications."

While data level integration may seem convenient, it has associated risks. Directly accessing application data bypasses any business logic implemented in the application code and thus can cause issues in data consistency when writing directly to a database. Direct reads to an application's data can be broken if that application's data schema is changed. This results in broken integration points that may be difficult to find if they are not regularly used. The data schema itself may be difficult to understand and there is the risk that a data query may not return the desired outcome. Therefore, integration via direct data access is not the preferred approach, although data volumes, performance issues, or the lack of other mechanism may require this technique.

2 **API Level Integration** makes use of a published API to access and modify application data via method calls. The API is offered in a form that allows for invocation from a variety of environments and languages. The most common current approaches will involve some form of web-based transport interface with a portable data payload such as XML or JSON.

API level integration is preferable to data level integration. API provides a level of abstraction that isolates the data storage schema, making it easier to modify that schema without concern for integration side effects. It also ensures that appropriate business logic is applied to any data changes in the application.

The use of API level integration assumes that the application developer has provided an API, and that the API exposes the required functionality, which is not guaranteed. The robustness of an application API can depend on the maturity of the application and the vendor's commitment to supporting integration. An early-release application may not have had sufficient time to produce a complete API, and some vendors do not see supporting integration points as a high priority.

In general, API level integration is preferable to data level integration, despite the additional overhead, as it reduces the degree of coupling between the integrated applications.

3 **Business Process Integration** considers that integration and applications often exist to automate business processes. Business processes can have a substantial impact on the complexity of integration architectures.

Complex or inconsistent business processes can be poor candidates for automation and integration. For example, if the sales process is performed inconsistently across an organization it is likely that different tools will be used to support the process, and that data will be scattered across these tools.

"An enterprise's integration architecture must be capable of dealing with this variety of integration challenges, while ongoing enterprise architecture activities work towards alleviating the problem points in business processes and the supporting applications."

The lack of data consistency and having no single, defined source of trusted data means that automation and integration will be difficult to implement.

Business processes must be defined to be as consistent and straightforward as possible before considering the integration of the supporting software systems. Analysis and design of the business processes and supporting software systems may reveal ways to leverage applications in ways that reduce the number of required integration points.

These three integration mechanisms can satisfy an array of business requirements ranging from basic hardware services to highly scalable development platforms suitable for implementation of new applications.

Integration Orchestration

Both data level and API level integration provide mechanisms for manipulating an application's persistent data store. Orchestration provides the mechanism for invoking and sequencing integration interactions.

Batch processing

The simplest approach to orchestration is through the use of scheduled batch processes. This invokes small applications or scripts to perform integration through the data or API mechanisms discussed above.

In the simpler cases, this approach can be effective; most Business Intelligence ETL processes fit these scenarios. However, this approach may not scale effectively as the number of batch processes grows and the complexity of the integration logic becomes more difficult to track. Batch processes can also have limitations with respect to how quickly changes propagate throughout the various applications. Complex interactions between batch processes may put limits on how often the processes can be run.

Integration Orchestration Engines

A number of vendors sell integration tools that specifically address integration orchestration; examples include TIBCO and Microsoft BizTalk. These toolsets supply powerful orchestration capabilities. This includes graphic design tools, clustered execution engines, adapters for applications and API technologies, monitoring capabilities, failure detection, and error resolution capabilities.

A vendor-supplied orchestration engine supplies a broad range of capabilities but is costly in initial acquisition and in operational complexity.

About Quadrus

Quadrus is a recognized leader in IT professional services and solutions. Headquartered in Calgary, Alberta, Quadrus has delivered hundreds of successful projects across Western Canada since 1993. We are committed to providing the highest quality service to our valued clients.

Contact us:

info@quadrus.com

www.quadrus.com

+1 (403) 257 0850

Integration challenges must be complex and tool adoption must be far-reaching to achieve benefits that are commensurate with the costs.

Service Bus Integration Strategies

Many leading edge architectures for highly scalable systems, particularly in the web space, make extensive use of queuing products as an application communication solution.

The use of queues across applications makes it possible to achieve event-based application integration. The applications are aware of the events that precipitate application state changes.

Operational Considerations

Integration architecture must consider the temporal aspects of the integration strategy. For example, what degree of latency is acceptable in propagating changes across applications? There is a wide range of possibilities from changing the overnight batch processing of sales data into a data mart, to changing the near real-time updates from a work force management system into a customer relationship system. Timeliness of the integration requirements must be factored into the integration architecture and tool selection.

Large numbers of application integrations can add complex updates and upgrades. Changes to an application can break existing integrations, so it is important to consider existing integrations when performing the impact analysis of an application change. It is also critical to maintain an inventory of integration points to facilitate change impact analysis and for general support.

Integration Strategy Summary

The selection of an appropriate integration architecture and strategy must balance the considerations discussed above. The ideal scenario involves carefully designed business processes implemented on service bus–based applications that update data state in near real time response to business events.

In practice, it is likely that the application mix will include a variety of application architectures that lend themselves to integration activities with varying degrees of success. An enterprise's integration architecture must be capable of dealing with this variety of integration challenges, while ongoing enterprise architecture activities work towards alleviating the problem points in business processes and the supporting applications.