META Delta

Delta 2344 9 July 2003

Increasing the Value of EAI Implementations

Adaptive Infrastructure Strategies, CRM Infusion, Integration & Development Strategies

Janelle Hill

Despite widespread implementation of enterprise application integration (EAI) initiatives and use of integration servers, few organizations realize the full benefits of their investment. The major obstacles to achieving a strong return on the investment in EAI initiatives have everything to do with organization, culture, and application design issues and little to do with product choices.

The three most common inhibitors to achieving a strong ROI on EAI initiatives are:
1. Not having a shared-services model
2. Not having application-neutral adapters
3. Not fostering a development culture of reuse

Most integration server products are purchased under an application-specific project budget, such as conversion to an ERP or CRM system, with large interfacing requirements. The purchase is justified on the basis of programmer productivity benefits over the hand coding of so many interfaces. As a result, the implementation reflects the information integration needs of the specific application and does not facilitate EAI, where "E" means enterprisewide sharing of consistent information across application boundaries. Nearly 80% of EAI implementations have fallen into this trap, reducing their benefits by approximately 30%.

Through 2004, declining prices and standardization of EAI functionality will continue to exacerbate this tendency by making it easier to justify the investment. Avoiding this requires moving interface development out of the realm of application-specific project teams and into a shared-services model. However, the difficulty in making this change in the IT culture (and also in the corporate culture for funding application projects over infrastructure projects) means that by, 2005, 70% of Global 2000 enterprises will still have multiple islands of integration. Organizations with strong CIO and CxO leaders driving for information consistency and a holistic, integrated view of business operations will remain in the minority through 2008, at which time this trend will begin to reverse as more of the application portfolio is Web services-enabled and designed for integration.

A Shared-Services Model for Integration

Many organizations fall into the first EAI pitfall by not recognizing that applications get integrated because something needs to be shared across application domains. Whenever things need to be shared — whether they are data, documents, logic, or a user interface — they should be developed and managed independently of the applications that share them. Therefore, IT organizations must move the responsibility for application integration out of the application-specific project team and into a shared-services model, just as was the case with database development.

META Trend: Initially deployed (2002/03) as little more than an Internet-based set of integration and interoperability standards (XML, WSDL, UDDI), Web services will emerge by 2005/06 as a new services-oriented architecture displacing component-oriented paradigms (J2EE, CORBA, COM+). Web services will provide a metadata-driven agile infrastructure of composable service protocols based on a common composition language (XML), common protocol binding strategy (WSDL/UDDI), and active network protocols (SOAP).

With the current strong focus on service-oriented architecture (SOA) and Web services, the term "shared services" may be confusing. By "shared services," we refer to the shared human resources, the work that they perform, and the components they code. In a shared-services model for integration, integration specialists are a shared resource, performing work (i.e., "services") on behalf of multiple application teams. This team should be responsible for coding integration components (using SOA design principles) to be shared across many interfaces. Often, this team is also responsible for the operations management of the shared integration services they create.

A catalog of integration services created by the integration specialists should include both technical/infrastructural services (e.g., transport services, routing services, an interface auditing service) and business functional services (e.g., a customer master data synchronization service, a trading partner setup service, a purchase order acknowledgment service). Figure 1 provides further characterization of technical/infrastructure services and business functional services. Ideally, the design principles of SOA and Web services (i.e., loose coupling and late binding) are used to develop these services (see ADS Delta 1246).

Integration servers naturally support this model by providing a set of capabilities commonly required for integrating applications. One of the first responsibilities of the integration team is to install and configure these capabilities to create a foundation of shared technical services. For example, routing must be set up to use direct addressing, content-based addressing, or both (in which case there might be two routing services offered to application teams).

Application-Neutral Adapters

Most application portfolios currently include applications that reflect various architectures, languages, technology generations, and style, with each typically having very proprietary interfaces. In applying SOA principles, a best practice to facilitate migration to Web services-based interfacing is to implement application-neutral adapters in XML. For example, to better share customer information across a diverse set of applications, the integration specialists should collaborate with customer data stewards to design an XML-based Intermediate Document Format (IDF — see SIS Delta 832) to represent the key data elements to be shared across the enterprise. The application-specific teams will still have to develop code to extract their application's particular data elements and map them into the IDF. Although some IDFs will stabilize over time as use cases encompass overlapping sets of functionality, some will go through radical changes. For example, a business that has catered to the B2B market and then starts up a consumer line will suddenly find that the stable customer master is now in need of a radical overhaul. Thus, the shared integration team should develop a process to continuously refine the reusable IDFs. The approach of building application-neutral adapters also delivers on the SOA principle of loose coupling, so that both the producing and consuming applications can change independently without completely breaking the entire interface.

Fostering Reuse

The natural outcome of using a shared-services model and application-neutral adapters is reuse. For example, if multiple applications reuse the set of technical services and business functional services created by the integration team, the cost of the shared infrastructure is lowered due improved manageability. Although reuse is an inherently attractive objective, in reality, reuse levels are rarely higher than 10% of the total developed code (see ADS Delta 1246). The costs of reuse (particularly related to the additional effort to make something reusable) present an argument against a comprehensive strategy. Reuse initiatives should instead be concentrated on those interaction points where reusability will be the highest. Application integration is particularly fruitful, especially legacy systems integration, B2B connectivity, and master data synchronization.

Avoiding the EAI Pitfalls

Regardless of whether a project is the first EAI project or not, the project at hand should be the catalyst to identify shared informational needs. Accordingly, someone must be designated to be the integration specialist. This individual must become familiar with the specific tools and development processes used for interfacing. Ideally, the individual should have some middleware implementation experience. Lacking that, strong data architectural experience, SOA experience, and good interpersonal skills for working across multiple teams are beneficial. This person will be responsible for developing IDFs for any entities that are identified as having high reuse potential (i.e., to be highly shared). Also, at least one application outside of the current project scope should be included for its requirements for shared data and technical services it could exploit.

Bottom Line

Application project teams should refuse to be responsible for integration and push the IT organization to create a shared-services model for integration.

Business Impact: Avoiding EAI implementation pitfalls will increase the ROI of integration tool investment by as much as 30% and is a first step toward building a more adaptive infrastructure to improve business agility.

Addendum

Figure 1 — Characteristics of Integration Services

Technical/Infrastructural Services
Technical/infrastructural services typically map to infrastructural technologies (e.g., middleware, database, security, communications). Within any area of technology, there may be multiple choices to deliver multiple service levels (i.e., different security levels, different performance levels, different availability levels, etc.).

Examples of technical services for integration include:

- ***Transport services:*** Three different transport services might be message-based, file transfer, and mailbox (store and forward). Alternatively, there might be one transport service whose WSDL defines alternative degrees of latency and possible envelope size. Depending on the subscriber's requirements, different technology components are used.
- ***Routing or addressing services:*** Options include direct addressing, content-based, distribution list-based, and role-based service.
- ***Data quality service:*** This technical service might invoke some data cleansing and validation routines.
- ***Authentication service:*** This might be a consistent method for authenticating partners.
- ***Audit service:*** This service might standardize how long messages are kept for historical auditing needs.
- ***Scheduling service:*** Options include clock (scheduled) or event-triggered service.

Business Functional Services
In the context of integration, business functional services should focus on relationships (i.e., entities, their attributes, and their relationships) and not on process logic. (Process logic should remain in the domain of the application owner.) Thus, these components are coarse-grained services that reflect standard mappings and translations.

Some examples of business functional integration services include:

- ***Master data synchronization:*** This initiative should reflect the "go forward" semantics for the master data element (e.g., customer, product, property). Logic would include the mapping from the IDF to this service and any required transformation logic. It would also likely include the rules for coordinating the change to subscribing applications and might invoke a data quality service. Lastly, this service should include logic to determine whether another copy of the service must be provisioned to meet performance criteria.

Source: META Group