Enterprise Application Integration Techniques

James Fenner

": the combination of processes, software, standards, and hardware resulting in the seamless integration of two or more enterprise systems allowing them to operate as one.[1]"

The introduction of new technology is always met by resistance from the exponents of current technology. However, "technology at this point is simply a means to an end. What is really strategic is the use of the information and how we exploit and maximize it.[2]" This means that before we can introduce Enterprise Application Integration, we must justify its use. To do this, we will initially look at why it is needed, starting with a brief history of the development of computer systems. Once we have provided the justification for its use, we can then look at the specific requirements for successful implementation. Then we examine the two different integration architectures, point-to-point and middleware, and look at their respective advantages and disadvantages. The next step is to look at the four common integration methods, and what criteria are useful in choosing the appropriate one. Then we look at examples of who actually uses this, and in what context. Finally, we can examine the likely future for Enterprise Application Integration techniques.

A brief history

For many thousands of years, man has used tools to help him carry out tasks. Cavemen attached sharpened flints to wooden poles, enabling them to attack prey from a distance. Modern man uses computers to help him carry out many tasks, from shopping and banking to space exploration and medicine. The first computers were primarily used to automate formerly manual tasks. These tasks were typically split into small sections, much as you would find businesses split into departments. As the system had been developed by or for a specific department, the system would often do exactly the same as the manual steps used to, giving a very narrow scope. The resultant system would generally be independent of others; there would be no integration whatsoever: "There was no thought whatsoever given to the integration of corporate data. The entire objective was to replicate manual procedures on the computer.[3]"

These were known as "stovepipe" systems, giving us something like figure 1.[4]

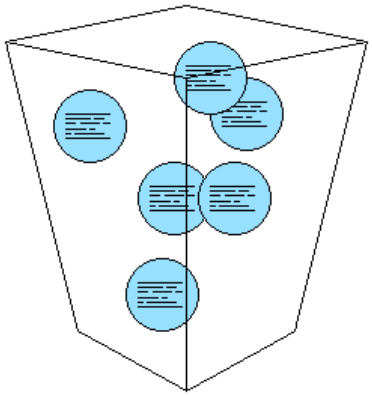

Figure 1. Independent islands of computing[4]

This "islands of automation" model was abandoned gradually for three reasons: [3]

1. It became increasingly necessary to allow the systems to interact. This meant the existing systems had to be integrated to provide interoperability – the scale meant that you couldn't simply start again!
2. The realisation that customer information within the stovepipe system had massive value, particularly when viewed as a whole. For example, a software vendor may have separate systems for home, business and government client, and no way to obtaining a global view of the information.
3. The desire to integrate key systems with vendors and customers.

As integration became more and more important, systems would tend to look more like figure 2. [3]

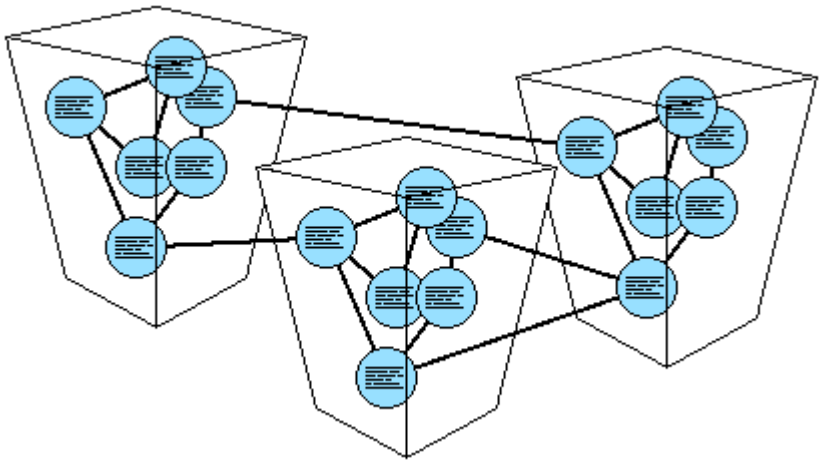

Figure 2. The integrated enterprise[3]

This brings us (nearly) up to the present. In the last decade, packaged software solutions became very popular. Software such as SAP, Oracle ERP, PeopleSoft, JDEdwards, Siebel and Clarify would tend to work well individually, but again, would create "information islands" as before, albeit on a larger scale. When common data had to change, it had to be updated manually. For example, in a German bank, clerical staff would read from one screen and type data into a user interface of another application. This was slow, and generated many errors[5]. Also, "the further proliferation of packaged applications, applications that addressed the potential problems of the Year 2000, supply chain management/business-to-business (B2B) integration, streamlined business processes, web application integration, and overall technology advances[6]" meant that long term scalable solutions were required. These solutions are called "Enterprise Application Integration" (EAI) techniques.

EAI Requirements

Now that the need for EAI has been recognised, we must continue by explaining what EAI must involve in order to be a realistic solution. It must cover every part of an enterprise system, including architecture, hardware, software and processes.[1]

- ***Business Process Integration (BPI):*** It is fundamentally important for a corporation to specify the processes involved in the exchange of enterprise information. "This allows organizations to streamline operations, reduce costs and improve responsiveness to customer demands[7]." This can include process management, process modelling, and workflow. Here, we involve the combination of tasks, procedures, organizations, required input and output information, and tools needed for each step in a business process. [1]

- ***Application Integration:*** Here, the goal is to "bring data or a function from one application together with that of another application that together provide near real-time integration[8]." This can include, business-to-business integration, customer relationship management (CRM) systems which can be integrated with a company's backend applications, web integration, and building web sites that interact with multiple business systems.[1]

- ***Data Integration:*** If we want the above two integrations to succeed, we must also integrate the data involved. Its location must be identified, recorded, and a metadata model must be built (a master guide for various data stores). Now, data can be shared or distributed across database systems, providing it is in a standard format such as COM+/DCOM, CORBA, EDI, JavaRMI, and XML.[1]

- ***Platform Integration:*** Finally, the separate needs of the heterogeneous network must be integrated. Platform Integration deals with the processes and tools that are required to allow these systems to communicate, both optimally and securely, so data can be passed through different applications without difficulty. For example, finding how an Apple can pass data to a wireless palmtop is part of the entire corporate system integration.[1]

These activities, as we have said, are essential if EAI is to replace the unsatisfactory integration techniques that we have arrived at through time.

EAI Integration Architectures[9]

Within EAI, there exist two types of integration architecture:

- Direct point-to-point (PTP)
- Middleware-based

Point-to-Point

This is the basic, more traditional approach. It is used because it is easy and quick, certainly viable for situations where we have few systems to integrate. For example, a new web site may need to interface with an existing sales order system and point-to-point integration may appear suitable. However, as you integrate additional applications, you get a situation like that shown in Figure 3:[10]

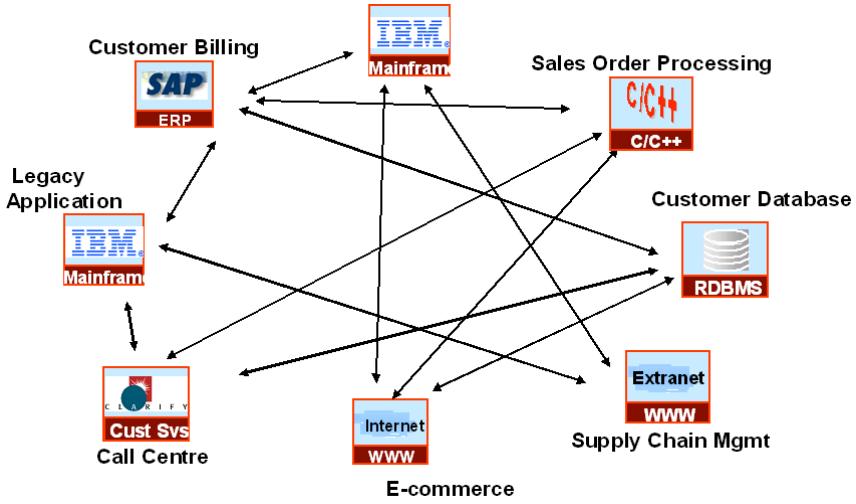

Figure 3. The later stages of a point-to-point integration[9]

As this solution scales up, the infrastructure proves brittle. The tight coupling, dependence, and number of integration points are all major disadvantages. The eight applications shown in figure 3 are using a total of 12 integration points, all of which need support. In theory, you could need up to double the number of integration points compared to the number of applications, as shown in figure 4, where the five applications need ten points of integration:[8]

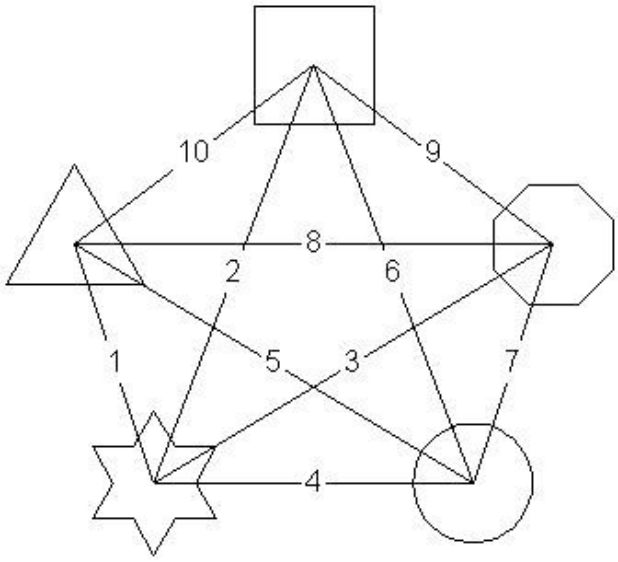

Figure 4. Number of point-to-point connections[8]

If we want to avoid this, we need to provide an intermediate layer that can isolate the changes between applications, effectively reducing the coupling. To do this, we use middleware.

Middleware

As we have said, we need to provide something to mediate between applications. By using middleware, we can provide generic interfaces, which allow applications to pass messages to each other. Each of these interfaces defines a process that the application provides. In figure 5, we can see a logical depiction of this principle: [8]

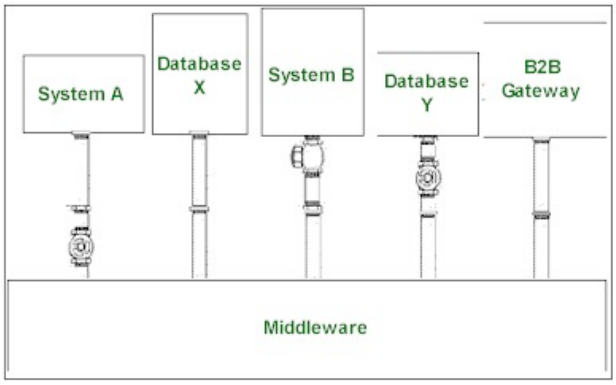

Figure 3. Middleware-based integration[8]

Now, our five applications only have five integration points, potentially halving this cost. We can also add and replace applications in a manner that does not affect other ones. The middleware itself can perform operations such as routing, transforming, aggregating, separating, and converting on the data that is passed. Note however, that there is additional complexity in terms of setting up the middleware, and converting the applications to use the middleware APIs.

The Four Integration methods[8]

Once the logical EAI architecture has been selected, we can move onto the actual integration method that is suitable to use. There are four common integration methods:

- Data-level integration
- User interface (UI)-level integration
- Application-level integration
- Method-level integration

Data-level integration

Here, the backend data stores of the relevant application are integrated, and can be either push or pull based. When using push based, one application makes SQL calls on another application's database tables. This is through database links or stored procedures, and data is pushed into another application's database. However, pull based integration uses triggers and polling. The triggers capture changes to data and write the identifying information to interface tables. It is then possible for adaptors to poll the application's interface tables and retrieve the pertinent data. This pull based integration is used when an application requires passive notification of changes within another application's data.

When the application that needs to be integrated does not provide any APIs or client interfaces, you would use data-level integration. You must also have a good understanding of the business operations that may affect the application's data model. It is typically the only option with most custom applications that lack APIs.

User Interface-level integration

This ties integration logic to user interface code, and can be either scripting or proxy based. When using scripting based, the integration code is embedded into the user

interface component events, common with client/server applications such as PowerBuilder or Vantive.

In cases where direct access to the database is not easy or possible, or when the business logic is embedded in the user interface, this is the correct integration method to use. Mainframe and client/server applications are often good candidates for this. Mainframes do not tend to have access to friendly data stores, and do not provide public APIs. However, user interface level integration is generally used as a last resort. If you add scripting logic to catch events with client/server applications they become very difficult to maintain, as integration levels increase and more changes occur. User interface changes can break integration triggers and logic anyway. This tight coupling creates a permanent link between the maintenance of the user interface and the integration code.

Application-level integration

This is considered the best way forward for application integration, and it uses the integrated application's integration frameworks and APIs. It is good to use, since it is transparent to the integrated application and it preserves the application's data integrity. The application interface allows you to invoke business logic to preserve data integrity. Integration API examples include Siebel's Java DataBeans and SAP's JCA (J2EE Connector Architecture).

Method-level integration

This is less frequently used specialisation of the application level integration method shown above. Here, we aggregate common operations on multiple applications into a single application that fronts the integrated applications. It is generally used when each integrated application has a similar set of API or functional methods. The integrated applications must support a Remote Procedure Call (RPC) or distributed component technology. The main disadvantage to this approach is again the tight application coupling in front components. They will break when changes are made to the integrated application API, and these problems will propagate down to the other applications that rely on them. This is used when we have distributed component or CORBA technology.

How to choose an integration method? [8]

This is really an exercise in constraint-based modelling. You must look at each system and define the possible interfaces into that application. In some cases, the application does not have any API; therefore the backend data store represents the only option. In other cases, APIs and a CORBA infrastructure may exist; so employ application-level integration. [8]

But who uses this?

It's all very well having a well-defined integration system, but without the backing of major industry, these techniques will soon disappear. Fortunately there exists a massive backbone in the large multi-national companies to support development of EAI. Market leaders include BEA Systems, CrossWorlds Software, IONA Technologies, Level 8 Systems, Mercator Software, NEON (purchased in 2001 by Sybase), SeeBeyond, Software AG, TIBCO, Vitria Technology, and webMethods. Market-leading large system integration firms include IBM Global Services,

Accenture, PricewaterhouseCoopers, CSC, and EDS[1]. So, we really do have large current support for the use of EAI.

For an example of the type of work a typical EAI company might do, see table 1 below[11]:

Project Name	Description and Technologies
Business Support System for Russian Government Agency April 2001 - December 2002 (planned)	The project is covered by a NDA so only a general description can be provided. The main contractor is one of the world's largest IT companies. Digital Design is developing the software part of the solution worth almost 1 million USD. The system is integrated with a standard industrial document management system, a search engine, and an Oracle database.
International Paper August 2001 - December 2001	Order-tracking system for client support personnel and company management. The system is integrated with Scala ERP, Prodis (production control system) and TPC (Transportation Process Control system). Main functionality is to provide information about order placement, production, stock availability, credit control, transportation and customer claims. The system is being used in the production environment serving a large number of International Paper's clients' requests.
Electronic Mail System for Russian Ministry of Railways August 2001 - June 2002	The client used to have independent IT policies in its 22 subsidiary companies. Digital Design developed a number of applications providing connectivity between different messaging platforms and catalogues. The system is being used by 50,000 users in 11 time zones. On top of this messaging system, Digital Design rolled out HP Open View Operations to manage both the client's corporate network and the Electronic Mail System from a single centralized location. This was done using SOAP, a leading edge technology which provides many advantages, such as seamless integration of various applications over the Internet.

Table 1. Use of EAI in Industry[11]

What for the future?

EAI is still a maturing technology, but seems to have a brilliant future. Why? To put it simply "It's the economy, stupid.[12]" The EAI market is set to become the most important and fastest growing IT sector in the next three to five years. Apparently, "worldwide revenues in this market will jump from $5 billion in 2000 to nearly $21 billion in 2005. This increase represents a strong compound annual growth rate (CAGR) of over 30%. By comparison, the corresponding opportunity of the overall IT services industry will increase at a CAGR of 11% during the same period.[13]" This growth will also be primarily felt in this part of the world "North America and Western Europe will generate more than 90% of the demand for global EAI services through 2005, with Japan and Latin America driving the remainder of this service demand.[12]" It is not therefore surprising that so many major companies are involved as we have previously seen. However, there are factors that may slow the predicted growth, which are the "cost of services, human issues regarding EAI engagements, and business-to-business integration challenges.[12]"

But what does this all mean for a modern company? Vendors will emerge to lead the market, vendors who have accumulated additional layers of application content knowledge. Rather than just using a propriety tool, they will be able to choose from "a toolbox of possible technologies[1]", each of which will be particularly suited to whatever the current project may be. They will be able to offer economical integration services, more economical than those available to us now. This will be partly because they can avoid costly trial and error testing, due to their familiarity with the capabilities and limitations of the products, and as they can leverage their investment in application knowledge and process development. They will also understand the most important questions to ask customers about their software product, and this enhanced relationship will allow them to provide more economical, reliable maintenance and support of evolving customer integration requirements[1]." It is therefore clear that although there are massive financial possibilities to be exploited, there is much work required to achieve the previously stated goals.

Conclusion

So, we have seen that EAI is an important tool for any company in the IT industry. We have looked at its roots, and why it has been developed, and this has led us to define criteria for successful integration. We then moved onto the EAI integration architectures, and then examined the integration methods. Then we looked at the use of EAI in industry and the future it holds. It certainly seems that the importance can only grow.

---

[1] EAI.ITtoolbox.com
[2] CIO magazine 1st July 2002, former Wal-Mart CIO Kevin Turner
[3] William Inmon, "A Brief History of Integration." EAI Journal.
[4] XML and Java, Todd Sundsted
[5] Emmerich, Tigra
[6] John P. Desmond and Ed Acly, "Beating the Integration Blues." Software Magazine. September 1999.
[7] Andre Yee, "Demystifying Business Process Integration." EaiQ.
[8] www.gartner.com
[9] EAI using J2EE, Abraham Kang
[10] Anonymous CITL Consultant
[11] Digital Designs UK Ltd
[12] 1992 US Democratic campaign slogan
[13] IDC, "The EAI Market Simmers with Robust Growth Expectations." February 28, 2001.