Enterprise Application Integration (EAI) Techniques

The development of technology over the years has led to most systems within an organisation existing in heterogeneous environments. That is to say, different applications were developed with varying languages, operate on different hardware and available on numerous platforms. The problems lay in the fact that when implementing systems, decisions on the technology employed differed from department to department and also had some dependence on the latest trends. What emerges is that these systems serve only the departmental needs. Information and process sharing across an organisation is not accommodated for. These types of systems are known as 'stovepipes'.

Each of these stovepipe systems held independent data; it was recognised that customer information and the sharing of this information across departments was extremely valuable to an enterprise. Allowing the disparate systems to interoperate became increasingly important and necessary. As organisations grew, so too did the desire to integrate key systems with clients and vendors.

Research has shown that during software development, a third of the time is dedicated to problem of creating interfaces and points of integration for existing applications and data-stores. Clearly, the idea and pursuit of application integration is not something new. What *is* new are the approach and the ideas that Enterprise Application Integration (EAI) encompasses and the techniques it uses. In order for it to be a success and a realistic solution, applying EAI requires involvement of the entire enterprise: business processes, applications, data, standards and platforms.

Business Process
The focus here is on combining tasks, procedures, required input and output information and the tools needed at each stage of a process. It is imperative that an enterprise identifies all processes that contribute to the exchange of data within an organisation. This allows organizations to streamline operations, reduce costs and improve responsiveness to customer demands[3].

Application
The aim here is on taking one application's data and/or functionality and merging them with that of another application. This can be realised in a number of ways. For example, business-to-business integration, web integration, or building websites that are capable of interacting with numerous systems within the business.

Data and Standards
This addresses the need to have a global standard by which data can be shared and distributed across an enterprise's network of systems. Without this format, the two aforementioned integrations would not be viable. To achieve this, all data and its location must be specified, recorded, and a metadata model built.

Platform
This provides a secure and reliable means for a corporation's heterogeneous systems to communicate and transfer data from one application to another without running into problems.

There are two types of logical integration architecture that EAI employs: *Direct Point-to-point* and *middleware-based* integration.

Point-to-point Integration
When dealing with very few applications, this method is certainly adequate. Point-to-point integration is usually pursued because of its ease and speed of implementation. It must be stressed though, that the efficiency of this method deteriorates as you try and integrate more systems. So, although to begin with you only have a few systems, consideration must go into the future; scalability is a huge concern. You may begin with integrating two systems, but integrating more could lead you to something that resembles Figure 1.

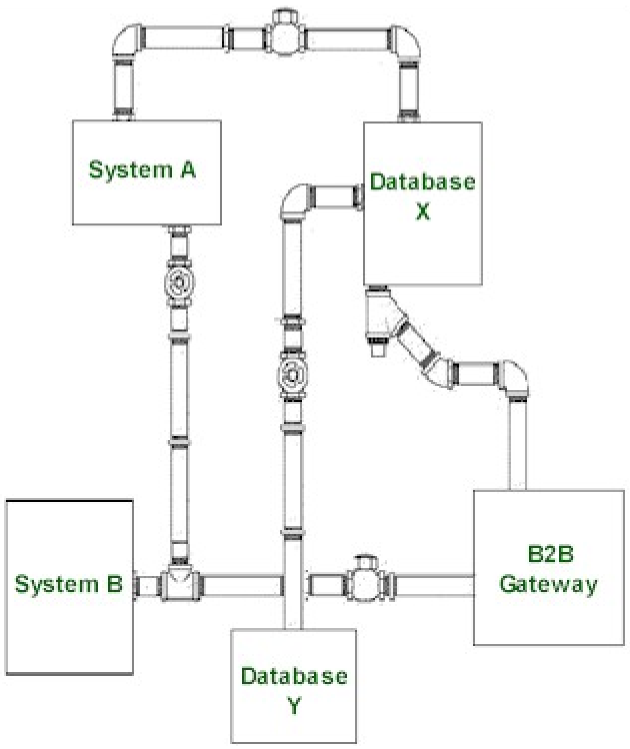

Figure 1. The later stages of integration[1]

In theory, you could end up in a situation like that of Figure 2. The number of integration points is double the number of systems. This will be problematic because of the tight coupling between the systems. Alterations in one system could have adverse effects on another. Each additional application thus becomes more difficult to maintain and integrate.

To alleviate the issue of high amounts of integration points and thus relieving the coupling problem, the use of middleware has been introduced whereby the number of integration points will be equal to the number of systems.

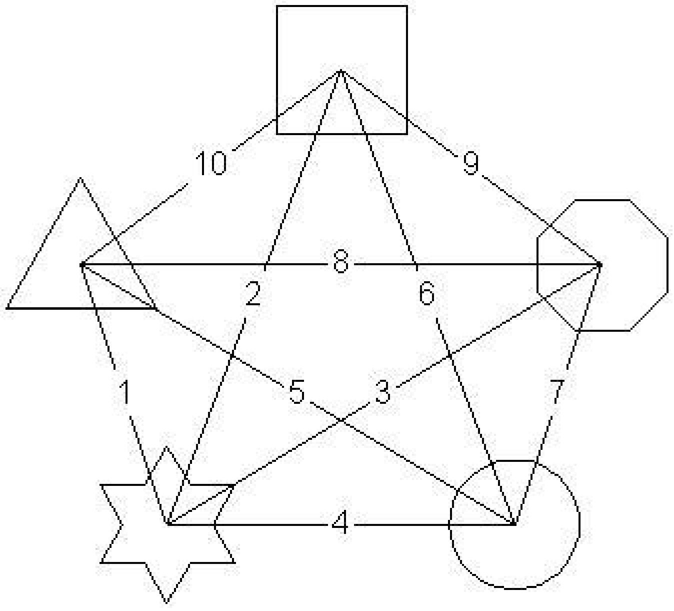

Figure 2. Number of point-to-point connections [1]

Middleware
An intermediate layer (as can be seen in Figure 3) provides generic interfaces through which the integrated systems are able to communicate. Middleware performs tasks such as routing and passing data. Each of the interfaces define a business process provided by an application[1]. Adding and replacing applications will not affect another application.

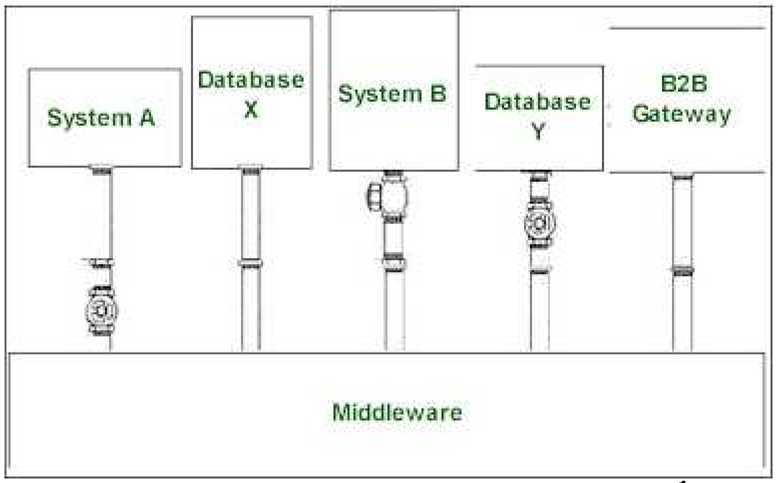

Figure 3. Middleware-based Integration[1]

In comparison to the point-to-point approach, middleware-based integration can easily support a larger amount of applications and does not require as much maintenance. Despite these benefits, it must be noted that there is an added initial complexity of setting up the middleware and converting existing applications to use the middleware APIs [1].

Having selected the integration architecture, a decision must be made regarding the method of integration. Organisations must appreciate both business processes and data. They must then select which of those require integration. This can take on several dimensions; in EAI there are four common types of integration:

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

Data-level
At this level, backend data stores are integrated to enable the movement of data between them. Put simply, information can be extracted from one database, processed as needed, and then updating it in another database. In an EAI enterprise, this could mean drawing data from as many as hundreds of databases and thousands of tables. For this reason, keeping the integrated application's data intact is a problem. For example, one table might have dependencies to others, and the integrated application may be the sole enforcer of those dependencies.

Data-level integration can be push- or pull-based. Push-based integration is when one application makes SQL queries on another application's database; data is pushed into another application's database. In contrast, pull-based integration is used *when an application requires passive notification of changes within another application's data*[1].

Cost benefits of data-level integration give it its advantage over other approaches. This is because on the whole, the application is not altered; code is not changed and so the expense of changing, testing, and deploying the application is not incurred[2].

Data-level integration should be used when the application up for integration does not provide any APIs or client interfaces. This is typically represented as the only option with custom applications lacking application APIs.

Application-level
This refers to making use of interfaces contained within custom or packaged applications such as SAP, Peoplesoft or Baan. These interfaces are leveraged to provide access to business processes and information. This approach is probably the best way to integrate applications as it allows you to invoke business logic to preserve data integrity. Developers are able to bundle many applications together so that business logic and information can be shared. This approach is more widely used and is preferred since it is transparent to the integrated application and the application's data integrity is preserved.

Method-level
In effect, this is a more complicated form of application-level integration and is used less frequently. Common operations on multiple applications are aggregated into a single front application. For example, the method for updating a customer record can be accessed from numerous applications without having to rewrite each method within the respective application. Since all applications that interact with the integrated applications do so via

this front application, method-level integration requires the integrated applications to support a RPC (remote procedure call) or distributed component technology.

The disadvantage lies in the fact that changing the integrated application API will break the front application components and the applications that rely on them. Given this then, it is usually more appropriate to opt for application-level integration using middleware.

User interface (UI)-level integration
Although it is more primitive, this approach is necessary and useful. Applications can be bundled together and their user interfaces used as common point of integration – this approach is known as proxy-based user-interface level integration. A second type which is scripting-based exists.

There are those who regard this approach as unstable, and although not preferred, user interface-level integration should be used on occasions when you cannot easily or directly access the database, or when your business logic is embedded in the user interface. For example, mainframes that do not provide data stores or public APIs. Many client/server applications embed the business logic in the client. Accessing and maintaining data integrity in instances like these can only be achieved by user interface-level integration.

The Right Method?
The task of choosing the right method of integration is predominantly an exercise in constraint-based modelling[1]. It will vary depending on the business, and its current technological situation. The general guideline is to analyse each system and identify all potential interfaces into that application. Should the application not have any API; the backend data store represents the only option. In other cases, APIs and a CORBA infrastructure may exist so application-level integration can be employed. Having chosen the integration method, the next step is to "identify a common integration XML Schema in order to encompass all integration objects and their associated attributes"[1].

The EAI Process
The following steps builds upon the integration methods I have just mentioned. It is a high-level process which can be applied to an EAI project to help steer its course and ensure that right, informed decisions are made during the course of the project and hence secure its success.

Step 1: Understanding the Enterprise and Problem Domain
This in-effect is like requirements gathering. It involves speaking to numerous people within the business, specifically heads of departments, in order to gain an understanding of what is and is not important. Obtaining quality information at this stage is imperative as it will lead to and impacts steps 2 and 3.

Step 2: Making Sense of the Data
Even though most EAI projects integrate at the data-level, but even if it was at a different level, you still need have an understanding of the database. This is done by identifying data, cataloguing the data and then creating enterprise metadata model.

Step 3: Making Sense of the Processes
In order to determine how to approach the enterprise business model, a view of enterprise at process/method-level is undertaken. Business processes need to be understood and documented. This involves looking at how they relate to each other AND to the metadata model built in step 2.

Step 4: Identifying Application Interfaces
In addition to seeking common methods and data to integrate, interfaces also need to be addressed. The reason for this is that interfaces will differ from application to application so you should validate all assumptions you have about them and build a repository of information about what is available.

Step 5: Identifying the Business Events
This looks at WHAT invokes an event, WHAT takes place following this event and any other events that may be invoked. So for example, a customer signing up for credit on an online store represents event. You can capture this event and make something else happen e.g. automatically running a credit check.

Step 6: Identifying Schema & Content Transformation
This stage addresses how the schema and content is transformed. The need for this stems from the fact that data in one system will not make sense to another system so it needs to be reformatted accordingly. Achieving this assures maintenance of consistent application semantics across all systems within an enterprise.

Step 7: Mapping Information Movement
This involves looking at what data element or interface information is moving from. So for example, the customer id from the sales database needs to move to the credit-reporting database. The movement of this information needs to be mapped so at all times, we know where it is physically located and security present.

Step 8: Applying Technology
It is very unlikely that the final solution will come from a single vendor. Many technologies exist and so you will most probably have a mixture of products. It is important to understand available solutions and then match these to criteria. This is a difficult process and requires a pilot project to prove the technology will work. Thus the the time taken to select the technology could be as long as the EAI project itself.

Step 9: Testing
Though it is expensive and time consuming testing is essential. It will ensure that the final solution will scale and can handle the rigors of day to day usage. For proper testing planning is a must. This is because most EAI projects will be implemented in an enterprise that have business-critical systems so very rarely can these systems be taken offline for testing.

Step 10: Considering Performance
In order to build performance into a solution, it must be designed and tested for before going live. This is because once a solution has been deployed, you cannot proceed with fixing performance issues. An example of the kind of test you would do is testing your

solution under a different amount of users, say 100, 500, and then 10,000 users. This will help you evaluate if your solution is capable of coping under these different conditions.

Step 11: Defining the Value
Addressed here, is the question of what the business value of integrating the systems is and the overall value of the EAI solution. The general method employed to determine this value is by evaluating *dollars* saved. There are two types. *Hard dollar* looks at things like reduction of error rates or if orders are able to be processed more quickly. *Soft dollar* is less tangible for example, customer satisfaction or whether there has been increased productivity over time. This is in general terms and is likely to differ from business to business.

Step 12: Creating Maintenance Procedures
Finally, once you have reached a solution, it is not just a matter of deploying it and leaving it be. Maintenance issues overtime need to be addressed. Who will solve problems or monitor performance? A good idea is to document activities that need to occur. It is important to remember that the EAI solution is the heart of enterprise. Responsible for moving information between business critical systems, it is because of this very nature that makes it a vulnerable point of failure which could be the demise of an enterprise. So at this stage disaster recovery issues should also be introduced and resolved.

Conclusion
Five years ago, it was expected that the EAI services market will become the most important and fastest- growing IT sector. In accordance with IDC research, "worldwide revenues in this market will jump from $5 billion in 2000 to nearly $21 billion in 2005" [5]. Now this was only a prediction and taken into consideration were issues that may inhibit the growth of EAI. These include, "cost of services, human issues regarding EAI engagements, and business-to-business integration challenges."

Despite its techniques and specified integration techniques, without any support from key players in the industry, the popularization and realization EAI would not materialize. Luckily, market leaders include like BEA Systems which support EAI's development, and large system integration firms include IBM Global Services exist.

One example of the success of EAI is TIGRA[6], which integrated different financial front-office trading systems with middle- and back-office applications. This was done through the use of middleware and integrating at the data-level.

We have looked at where the need for EAI stems from, and the issues that need to be addressed. Techniques have been discussed as how to achieve this, and there are examples of when it has been applied successfully. There is strong support from the IT industry and I believe the importance of EAI will continue its growth not only in the IT industry but more significantly, its impact and necessity will be realized by that of the business sectors which employ the use of IT.

1. Abraham Kang, *EAI Using J2EE*, 2002
2. David S.Linthicum, Enterprise Application Integration, Addison-Wesley, 2000
3. Andre Yee, "Demystifying Business Process Integration." EaiQ.
4. EAI Overview, IT Toolbox, 2002
5. IDC, "The Enterprise Application Integration Market Simmers with Robust Growth Expectations." February 28, 2001.
6. TIGRA: An Architectural Style for Enterprise Application Integration" W. Emmerich, E. Ellmer and H. Fieglein. Proc. of 23rd Int. Conference on Software Engineering