Chapter 1.9
Enterprise Application Integration (EAI)

Christoph Bussler
Merced Systems, Inc., USA

ENTERPRISE APPLICATION INTEGRATION (EAI) TECHNOLOGY

As long as businesses only have one enterprise application or back end application system there is no need to share data with any other system in the company. All data that has to be managed is contained within one back end application system and its database. However, as businesses grow, more back end application systems find their way into their information technology infrastructure managing different specialized business data, mainly introduced due to the growth. These back end application systems are not independent of each other; in general they contain similar or overlapping business data or are part of business processes. Keeping the data in the various application systems consistent with each other requires their integration so that data can be exchanged or synchronized. The technology that supports the integration of various application systems and their databases is called Enterprise Application Integration (EAI) technology. EAI technology is able to connect to back end application systems in order to retrieve and to insert data. Once connected, EAI technology supports the definition of how extracted data is propagated to back end application systems solving the general integration problem.

BACKGROUND

Typical examples of back end application systems that are deployed as part of a company's information technology (IT) infrastructure are an Enterprise Resource Planning (ERP) system or a Manufacturing Resource Planning (MRP) system.

DOI: 10.4018/978-1-60566-242-8.ch088

In the general case, different back end application systems store potentially different data about the same objects like customers or machine parts. For example, a part might be described in an ERP system as well as in a MRP system. The reason for the part being described in two different back end application systems is that different aspects of the same part are described and managed. In fact, this means that the not necessarily equal representation of the object exists twice, once in every system. If there are more than two systems, then it might be very well the case that the same object is represented several times. Any changes to the object have to be applied to the representation of the object in all systems that contain the object. And, since this distributed update cannot happen simultaneously (in the general case), during the period of applying the change the same object will be represented differently until the changes have been applied to all representations in all back end application systems. It therefore can very well be the case that during an address update of a customer object the customer has two addresses. Some objects representing the customer have already the new address while others still have the old address. This situation exists until the distributed update is complete. Furthermore, in most cases there is no record of how many systems represent the same object. It might be the case and actually often it is the case that a change is not applied to all objects because it is not known which back end application system has a representation of the object in the first place. Only over time these cases will be detected and rectified, mainly through the resolution of error situations.

In summary, the same object can be represented in different back end application systems, the updates to an object can cause delays and inconsistencies, and locations of object representations can be unknown due to missing object registries.

A second use case is that precisely the same object is replicated in different back end application systems. In this case the update of the object in one system has to be applied to all the other systems that store the same object. The objects are replica of each other since all have to be updated in the same way so their content is exactly the same. Only when all the objects are updated they are consistent again and the overall status across the back end application systems is consistent again. In the replicated case it must not be possible that the same object exposes different properties like in the address example above.

A third use case is that applications participate in common business processes. For example, first a part is being purchased through the ERP system and upon delivery it is entered and marked as available in the MRP system. The business process behind this is consisting of several steps, namely purchase a part, receive the part, make the part available, and so on. In this case the back end application systems do not share common data, but their data state depends on the progress of a business process and it has to update the back end application systems accordingly. The data will change their state according to the progress of the business process. In this sense they share a common business process, each managing the data involved in it.

All these three use cases, while looking quite different from each other, have to be implemented by companies in order to keep their business data consistent. EAI technology (Bussler 2003) (Hohpe and Woolf 2003) allows to accomplish this at it provides the necessary functionality as described next.

ENTERPRISE APPLICATION INTEGRATION TECHNOLOGY

Enterprise application integration technology addresses the various scenarios that have been introduced above by providing the required functionality. In the following the different functionalities will be introduced step by step. First, EAI technology provides a reliable communication