CHALLENGES OF ASYNCHRONOUS MESSAGING

Asynchronous messaging is not the panacea of integration. It resolves many of the challenges of integrating disparate systems in an elegant way but it also introduces new challenges. Some of these challenges are inherent in the asynchronous model while other challenges vary with the specific implementation of a messaging system.

- *Complex programming model.* Asynchronous messaging requires developers to work with an event-driven programming model. Application logic can no longer be coded in a single method that invokes other methods, but the logic is not split up into a number of event handlers that respond to incoming messages. Such a system is more complex and harder to develop and debug. For example, the equivalent of a simple method call can require a request message and a request channel, a reply message and a reply channel, a correlation identifier and an invalid message queue (as described in *Request-Reply*).
- *Sequence issues.* Message channels guarantee message delivery, but they do not guarantee when the message will be delivered. This can cause messages

that are sent in sequence to get out of sequence. In situations where messages depend on each other special care has to be taken to re-establish the message sequence.

- *Synchronous scenarios.* Not all applications can operate in a send and forget mode. If a user is looking for airline tickets, he or she is going to want to see the ticket price right away, not after some undetermined time. Therefore, many messaging systems need to bridge the gap between synchronous and asynchronous solutions. (See the *Request-Reply* pattern.)
- *Performance.* Messaging systems do add some overhead to communication. It takes effort to make data into a message and send it, and to receive a message and process it. If you have to transport a huge chunk of data, dividing it into a gazillion small pieces may not be a smart idea. For example, if an integration solution needs to synchronize information between two exiting systems, the first step is usually to replicate all relevant information from one system to the other. For such a bulk data replication step, ETL (extract, transform, and load) tools are much more efficient than messaging. Messaging is best suited to keeping the systems in sync after the initial data replication.

- *Limited platform support.* Many proprietary messaging systems are not available on all platforms. Often times it is easier to FTP a file to another platform than accessing it via a messaging system.
- *Vendor lock-in.* Many messaging system implementations rely on proprietary protocols. Even common messaging specifications such as JMS do not control the physical implementation of the solution. As a result, different messaging systems usually do not connect to one another. This can leave you with a whole new integration challenge: integrating multiple integration solutions! (See the *Messaging Bridge* pattern.)

So asynchronous messaging does not solve all problems, and can even create some new ones. Keep these consequences in mind when deciding which problems to solve using messaging.

Source: http://www.enterpriseintegrationpatterns.com/patterns/messaging/Introduction.html