INTRODUCTION TO MESSAGE PATTERNS

Interesting applications rarely live in isolation. Whether your sales application must interface with your inventory application, your procurement application must connect to an auction site, or your PDA's PIM must synchronize with the corporate calendar server, it seems like any application can be made better by integrating it with other applications.

All integration solutions have to deal with a few fundamental challenges:

- *Networks are unreliable.* Integration solutions have to transport data from one computer to another across networks. Compared to a process running on a single computer, distributed computing has to be prepared to deal with a much larger set of possible problems. Often times, two systems to be integrated are separated by continents and data between them has to travel through phone-lines, LAN segments, routers, switches, public networks, and satellite links. Each of these steps can cause delays or interruptions.
- *Networks are slow.* Sending data across a network is multiple orders of magnitude slower than making a local method call. Designing a widely

distributed solution the same way you would approach a single application could have disastrous performance implications.

- *Any two applications are different.* Integration solutions need to transmit information between systems that use different programming languages, operating platforms, and data formats. An integration solution needs to be able to interface with all these different technologies.
- *Change is inevitable.* Applications change over time. An integration solution has to keep pace with changes in the applications it connects. Integration solutions can easily get caught in an avalanche effect of changes – if one system changes, all other systems may be affected. An integration solution needs to minimize the dependencies from one system to another by using *loose coupling* between applications.

Over time, developers have overcome these challenges with four main approaches:

1. *File Transfer* — One application writes a file that another later reads. The applications need to agree on the filename and location, the format of the file, the timing of when it will be written and read, and who will delete the file.

2. *Shared Database* — Multiple applications share the same database schema, located in a single physical database. Because there is no duplicate data storage, no data has to be transferred from one application to the other.
3. *Remote Procedure Invocation* — One application exposes some of its functionality so that it can be accessed remotely by other applications as a remote procedure. The communication occurs real-time and synchronously.
4. *Messaging* — One applications publishes a message to a common message channel. Other applications can read the message from the channel at a later time. The applications must agree on a channel as well as the format of the message. The communication is asynchronous.

While all four approaches solve essentially the same problem, each style has its distinct advantages and disadvantages. In fact, applications may integrate using multiple styles such that each point of integration takes advantage of the style that suits it best.

Source:

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