
Financial Systems Integration

-

Frameworks for Success

By John Clarke

Code Red Consultancy Limited

60 Lombard Street

London

EC3V 9EA

Tel: +44 (0) 171 464 8450

Fax: +44 (0) 171 464 8672

Financial Systems Integration - Frameworks for Success

Integrating often disparate IT systems can be a time consuming and expensive business. This paper draws on the experience of Code Red consultants operating in the finance industry, to offer advice on avoiding the potential pitfalls and provides some pointers to selecting the appropriate architecture for an organisation.

1. The Context.

In today's business world, IT systems have achieved widespread acceptance as one of the primary means to allow an organisation to improve efficiency and productivity, to become more responsive to the changing markets in which it operates by developing, producing and selling new products and to provide better levels of service to its customers. The exponential growth in the use of the Internet as a business medium, over the last couple of years, has added yet more fuel to the fires powering the engines of change.

For any commercial organisation the need to achieve and maintain competitive advantage is paramount. This need can manifest itself in the IT systems world in a number of ways.

Existing systems need to be adapted or enhanced to cope with new requirements – new products, new services, new means of delivery, new workflow processes. Existing information repositories need to be merged or accessed in new ways (“data warehousing/data mining”). Human intervention needs to be minimised. Doing business over the Internet means that existing systems have to be accessible either directly or indirectly

via a Web interface. The potential volume of business possible over the Internet should not be underestimated. Scalability of the IT infrastructure to cope with these increased volumes is therefore essential.

New systems being developed need to be able to drop seamlessly into the existing infrastructure and be able to access information and functionality within that infrastructure. Organisations merge or get taken over, initiating a whole new process of change for the IT systems of the merging organisations. An additional factor is the constant demand to reduce the development time for all these new features since the time to market for a new product may directly depend on the ability of the IT infrastructure of the organisation to support it.

To a great extent, the success or failure of an organisation in meeting these IT criteria depends on its ability to successfully integrate its IT systems, both by making its existing systems work effectively together and by reaching a situation where new systems can be incorporated into an existing IT infrastructure with a minimum degree of pain.

2. The Problem.

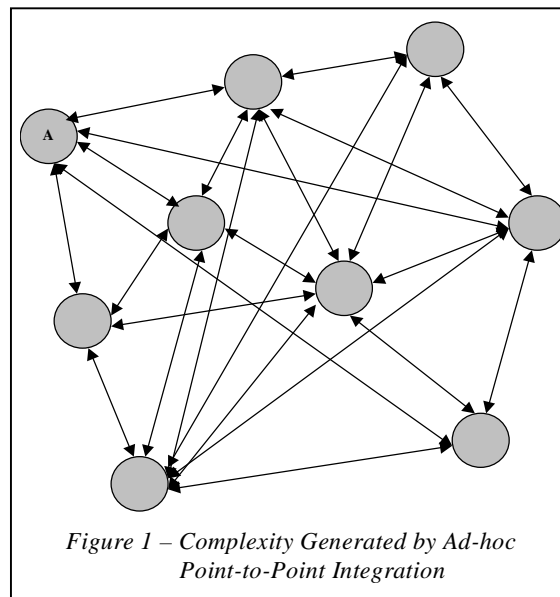
The difficulties of integrating systems are manifold. Different hardware and software platforms, different databases, different implementation languages, different design philosophies, disparate API's, different communications infrastructures and existing ad-hoc "stovepipe" point-to-point integrations all contribute to make the task of achieving a clean, uniform approach to systems integration within an organisation into a potential snake pit. In short, the very IT systems legacy which organisations must seek to utilise can itself prove to be the major obstacle to be overcome.

A common situation in many organisations is an almost unmanageable "cats cradle" of systems. This is expensive to keep running, vulnerable to failure and difficult to change – all symptoms that absolutely negate the business needs outlined in the previous section.

Every time a new system is introduced, there is an exponential increase in the number of point to point integrations that may be required for it to interact with already installed systems. Because the picture is so complex, the potential failure modes are many and poorly understood. Often, knowledge essential to keep the whole edifice standing is contained in the heads of a very few key individuals. People are afraid to introduce change into such a model because of the unpredictable effects this may have on the whole. When change must be introduced, it often proves expensive to do so because of all the custom integrations required and the

amount of testing required to ensure that pre-existing systems still behave correctly. On-going support and maintenance costs are also high because of the sheer volume of software to be kept running.

The situation is summarised in figure 1.



Here we see some of the potential for complexity. The blobs represent systems; the lines represent custom interfaces between the systems. In this case there are only nine systems, and the set of interfaces shown is not exhaustive – there is the potential for many more.

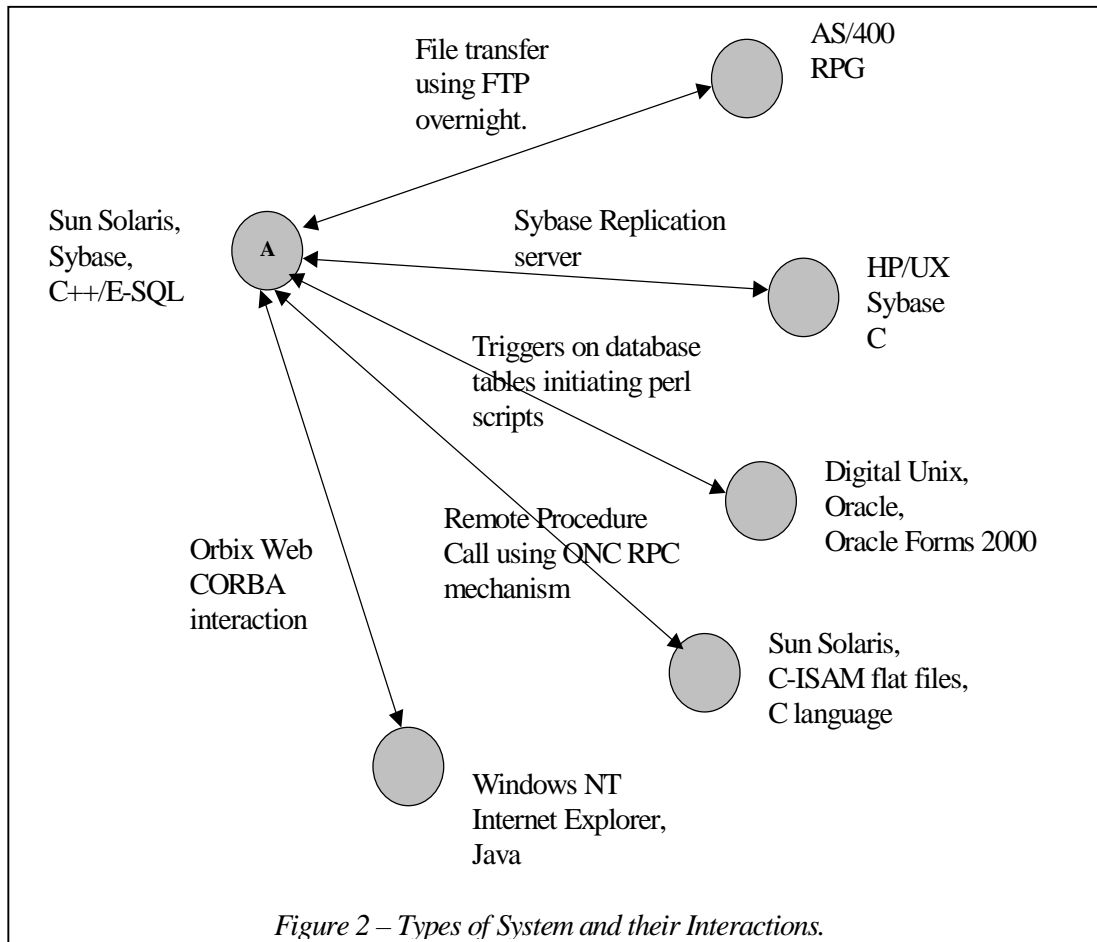
Now consider the impact of removing or adding a system to this picture. Clearly, as the number of systems increases the degree of complexity increases and the potential cost of development and maintenance increases.

When we take into account the nature of each of these systems and their

associated set of interfaces the picture is even worse. Consider node A in the above diagram in some more detail.

Here we see the additional level of complexity caused by hardware, software, implementation languages, design decisions and so forth. The effects of replication and distribution within individual systems haven't been considered here but it can be assumed that yet another layer of complexity would be added.

Typically the reason for this situation having arisen in the first place is historical. The evolution of an IT infrastructure over time together with the constant pressure to “get something out there that works” if allowed to continue unchecked will inevitably lead to such a situation. The natural reaction is to “leave well alone” if at all possible but from an organisations business perspective this is simply not acceptable.



3. The Solutions.

It is clear that “something” must be done to move away from the (admittedly worst case) scenario outlined in the previous section. The question is what is this “something”? A good place to begin is to look at the criteria for a successful systems integration strategy. Here are the most obvious ones:

- Widely supported, preferably standards based.
- Covers full range of hardware and software platforms.
- Offers a range of “qualities of service” in terms of performance, robustness, bandwidth etc.
- Supports different types of interaction (e.g. RPC, message, remote object invocation).
- Hides distribution issues such as location, routing, request broking from the systems being integrated.
- Offers the ability to support a common interchange format for information flowing between systems.
- Is able to scale to cover the entire organisation if necessary i.e. can be global in scope.
- Provides a uniform framework for new systems being introduced
- Allows legacy systems to be encapsulated to comply with the framework.
- Should allow for change in the future (e.g. different kinds of interaction, increased volume of information flowing)

There are a number of offerings on the market today which claim to help solve the systems integration problem. All of

these products take their own particular approach to providing a solution which meets to a greater or lesser extent the criteria outlined above. The majority of products available right now fall into two broad categories – Distributed Object Middleware (e.g. CORBA, DCOM) and Message Oriented Middleware (e.g. IBM MQSeries, New Era Of Networks, STC Datagate).

The Message Oriented Middleware purveyors like to position their products as “Enterprise Application Integration (EAI)” platforms which provide the enterprise wide backbone to allow all the systems in an organisation to exchange information and co-operate together. Basically all of these provide the ability to hook systems running on a wide variety of hardware and operating system platforms into a message queue based infrastructure which provides routing, guaranteed delivery, ordering of messages and redundancy in the case of failures in a manner largely transparent to the systems in question. In addition, some of these products offer a common “canonical” message format and also incorporate a rules engine to allow the infrastructure to make decisions about what to do with a message in terms of routing, data transformation etc based on the content of the message and rules specified by users. These kinds of product are currently the market leaders in middleware generally.

The rapid advent of Extended Markup Language (XML) has a potentially interesting impact here. XML has been designed with the thought of representing semi-structured data in

mind. As such, it is perfectly possible to use it as the common format for defining message content. An integration strategy using XML as the message format would have to make provision for the definition of the XML elements to be used and the parsing of these at receiving entities. However this is unlikely to be any more effort than using any other user-defined canonical format and given the availability off the shelf of XML parsers could in fact be a lot less effort. XML is on the road to standardisation by the W3C consortium and is likely to become a de-facto standard for information interchange before the formal standardisation process is complete.

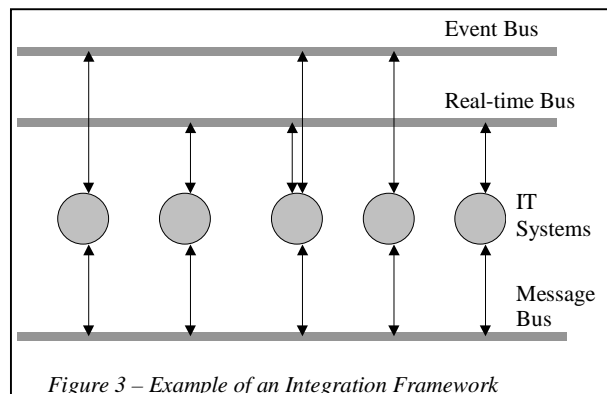
The Distributed Object Middleware products tend to be viewed more as a tactical solution within the scope of an individual system although the Interface Definition Language forming part of the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) has the potential to form the basis of a specification for an enterprise wide integration infrastructure. The problem with CORBA is that it is often perceived as complex, difficult to use and expensive – precisely the issues we are trying to address in the existing “spaghetti” systems integration model. It does however offer a powerful, scalable and standards based approach to systems integration and as such is worthy of consideration.

CORBA's main competitor in the Distributed Object Middleware arena is Microsoft's Distributed Common Object Model (DCOM). The major disadvantage with this offering is that it is intended to work in a Microsoft environment only, although

interoperability with CORBA is now a possibility.

The reality for an organisation is that it needs to support several different kinds of information interchange concurrently with one another. Large volume, low frequency batch data exchanges must co-exist with real-time or close to real-time smaller volume data exchanges. The increasing use of event-driven systems requires the ability to perform a “publish/subscribe” type of interaction. The various product offerings that provide this functionality include TIB, Talarian, OpenTrade and these must complement the more mainstream types of middleware already described.

It is unlikely that any single product offering or integration approach will offer the range of functionality, performance and scalability required to satisfy all these needs. What is required is an architecture which does meet these requirements wherever necessary (bear in mind that not every system in an organisation will need to use every kind of service provided). Such an architecture may well comprise several different vendor products and might look something like the picture in figure 3.



In this framework there is an organisation wide Message Bus to carry

messages formatted in some canonical format (possibly XML) transparently between systems. This can be constructed using one of the available message oriented middleware products.

The Event Bus component provides a means for systems to receive notification of some change in some other place in the organisation. This may be by means of a publish/subscribe mechanism like TIBCo ETX or via some alternative means like the CORBA event service. On receipt of a notification of interest, a system may then take some action. It may propagate some new events, it may request some information from elsewhere or it may post some data onto its output message queue, for example.

The Real-time Bus component is intended for use by interactions where a degree of timeliness must be assured – a request for the current market price of some instrument, or the entry of a trade via some sort of transaction. Again, this could be implemented using an off-the-shelf package. OpenTrade or Talarian are examples of this type of packaged middleware.

The individual systems in figure 3 may have a finer detail of structure than is shown on the diagram. For example one of the systems might be a real-time dealing room system supporting 300 trading stations via a network of replicated servers communicating using CORBA. This system would use the integration framework to transfer

information to other systems in the organisation, perhaps a Risk management system doing intraday Value at Risk calculations using traders current positions in the market as inputs. Another example is an E-commerce application server interacting with customers over the Internet and exchanging information internally with order management, ERP and accounting systems.

Of course, each organisation has its own set of priorities and factors that will influence the development of an integration architecture. These factors should be carefully analysed before a decision is arrived at. It may be that there is no requirement for a publish/subscribe infrastructure, or that most information flows need to be real-time. In these cases the architecture should reflect this and could potentially be much simpler.

The bottom line is that a successful integration framework must take into account many factors, some organisation specific and some generic. The resulting framework is unlikely to be satisfied by a single product vendor package. The objective is to maintain a clear view of the requirements of the organisation and the factors in play and make objective choices of integration components based on this view. The integration architecture should then ensure that the components selected are melded together to provide the comprehensive framework that the organisation requires.