# Interoperability Report: Seamless Integration vs. Heterogeneity

**HASITH D. YAGGAHAVITA**
TECHNICAL ARCHITECT
EUROCENTER DDC PVT LTD
24th DECEMBER 2007

# Table of Content

## *Introduction*

Most of the organizations today live by the slogan *"Buy the best, build the rest"*. Cost and risk of building from ground-up has made most organizations to consider buying commercial-off-the-shelf (COTS) products. Effective integration of these purchased IT systems has become the main responsibility of IT managers today.

Unfortunately, it is not very common for an organization to find all the required systems to be homogeneous. Often they are from different vendors having diverse architectures and operating on different platforms. The schema of the information model can widely differ from one system to another. At all these complexities, a seamless integration and smooth business process flow is what expected from the IT infrastructure.

According to Pollock (2001), robust integration architecture should support both *'Application Integration'* as well as *'Information Integration'* against heterogeneity. 'Application Integration' is the process of linking different software systems to become a part of a larger system. This is the technical solution that decides the level of integration (data level, application level, transaction level, process level, or human level) and technology of communication. Therefore 'Application Integration' mainly deals with the transportation of data/objects/messages between heterogeneous systems.

On the other hand, 'Information Integration' deals with the meaning and semantics of the communication. The meta-data, business rules and domain schema of one party should be understood by the other party for the integration to be successful. Maximum exchange of meanings by transformation of one entire domain representation schema to another partially compatible domain representation schema is the challenge of 'Information Integration'.

'Application Integration' aspects are primary requirements that need to be satisfied by any integration architecture. But that is still only half of the total picture. Most integration architectures fall in to the trap of focusing on much of these technical aspects but forget the quality aspects of 'Information Integration'. Simple integration requirements may be full filled by architectures biased to one arena, but complex integrations definitely require lot of attention and balance of both these aspects.

Despite the number of integration technologies and patterns exists, the extent to which the above goal is realized is debatable. This report looks in to and evaluates two such integration architectures published in order to solve the integration puzzle. The selected two architectures present two dissimilar approaches towards enterprise integration. Main focus of the evaluation is to study the two architectural patterns to assess their support for 'Application Integration' and 'Information Integration' aspects as set by Pollock (2001) in his white paper.

## *Information Brokering*

### Architecture Summary

One of the common integration patterns is to employ an information broker to mediate the integrating partners. The paper *'An architecture for efficient, flexible enterprise system integration'* published by Grundy et al (2003), discuss the implementation details of such a brokering architecture. The authors claim that the brokering architecture presented covers wide range of technology integrations over most of the other architecture patterns like 'file and document exchange, 'database integration and federation', 'message and document translation', 'distributed object interaction' or 'workflow and business process management'.

According to the authors, the key goals of this architecture are as follows:

1. Minimal (ideally no) modification should be necessary to the various enterprise systems
2. The performance of the customer purchasing portal must be very good, without delays caused by accessing remote applications and databases across wide area networks
3. Integration should be done in a consistent manner with no bespoke solutions for different technologies
4. Systems should not have direct knowledge of each other and integration support is isolated
5. Data and business process integration both need to be supported in the integration framework
6. The integration approach should be scalable, reliable and provide secure control over business data.

According to the point 1, 3, 4 and 5 above, goals of this architecture focuses on solving issues that can arise in seamless integrations across heterogeneous systems, This means that the architecture should have clear support for 'Application Integration' as well as for 'Information Integration' aspects.

Special characteristic seen in this architecture is that, building on the concept of data replication between the systems. In summary, each system keeps a dataset copy of its dependents and updates itself by employing optimistic access models. Then the conflicts and inconsistencies are resolved if any. The each communicating system accesses the replicated data from the other systems just as its local data. The synchronization of data is taken cared of by the brokering framework.

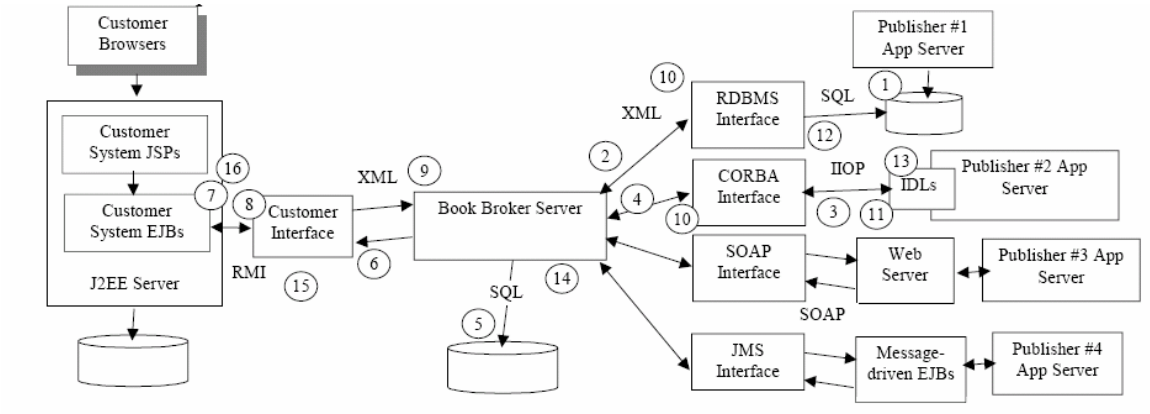


Figure 1: Brokering Architecture (Grundy et al, 2003 p.3),

'Application Integration' is performed by a set of interface components as depicted in the Figure 1. Several communication technologies can be supported by this approach. Synchronous remote procedure calls are handled by employing relevant interface agents such as CORBA, RMI or DCOM. SOAP or JMS interfaces can provide asynchronous messaging. SOAP interface operates on XML documents whereas JMS interface supports both XML documents and binary encoded data. RDBMS interface provide synchronous insert, update and delete operations on external partner databases. According to this architecture the 'Application Integration' can be performed at various different levels. Also different styles such as synchronous and asynchronous communication can be supported. These integration interfaces should be hand coded and can be differ from one technology to another.

'Information Integration' happens as a two step process. Specific data format of first communicating party is converted to the broker intermediate neutral format. Then the message in neutral format is converted back to the specific format of the second communicating party. The neutral format contains the meta-data such as entity key correspondence, changes made to document elements, and document status (e.g. committed, pending commit, rejected) which are encoded as XML documents. These conversions happen in the interface components and can be much bespoke and hand coded.

## Evaluation

The architecture under discussion has paid a considerable attention to the 'Application Integration' concerns. It supports wide variety of technologies and wide variety of communication styles through the interface components. High degree of freedom in implementing the interface components allows diverse technologies to be used in integrations. Still the architecture lacks presentation of a conceptual framework that unifies these technical integrations. This can result a variety of inconsistent hand coded implementations which are specific for each integrating party. This in fact conflict with the third architectural goal stated.

The architectural discussion has paid vary little attention to 'Information Integration' aspects. All the semantic conversions are bespoke and no guideline is placed on the same. 'Information Integration' happens at the interface components which are also responsible of 'Application Integration' aspects. Performing the two important interoperability aspects 'Application' and 'Information' by a monolithic entity can be

seen as a weak point in the architecture. Lack of clear separation of concerns in this architecture can lead to confusions in large scale implementations.

On the other hand, the broker implementation of this architecture is bound to the business process under discussion. For different business processes, specific broker implementations would be required if no reusable broker framework is placed over the discussed architecture. Therefore the extensibility of this architecture should be critically analyzed before employing for complex integration scenarios.

In overall, due to the complexities introduced by data replication and synchronization, authors are forced pay much of the attention to resolving the same. Even though the achievement of performance goals is justified by the paper, there is no clear justification how the other stated goals are realized.

## *Publisher/Subscriber Notifications*

### Architecture Summary

Another popular integration paradigm is the event driven interactions with publishers and subscribers. The paper 'Implementing a High Level Pub/Sub Layer for Enterprise Information Systems' published by Antollini et al. (2006) discusses the implementation details of such pub/sub architecture. The paper claims that the discussed architecture supports heterogeneous event produces and consumers where as most of other pub/sub services assume homogeneous information namespaces.

As common to any pub/sub notification service, the depicted architecture proposes to have a set of clients who asynchronously communicate by using a notification service. Notification service decouples the clients and act as the medium for message delivery. Clients can be producers or consumers. The producers can publish messages under a particular topic where consumers can subscribe to topics to receive the messages. Notification service is responsible of delivering all messages published after the subscription to the subscriber.

The architecture proposes to have a high level abstraction later which operates on any event notification service such as JMS or Rebeca. This extension approach provided is named as *'Concept Based Approach'*. This approach mainly focuses on resolving data interpretation problems. The architecture also provides an abstraction layer where the notification services specific details are well decoupled for the communicating parties. This makes the underline notification service pluggable by the design allowing developers to use their desired notification service technology as the backend.

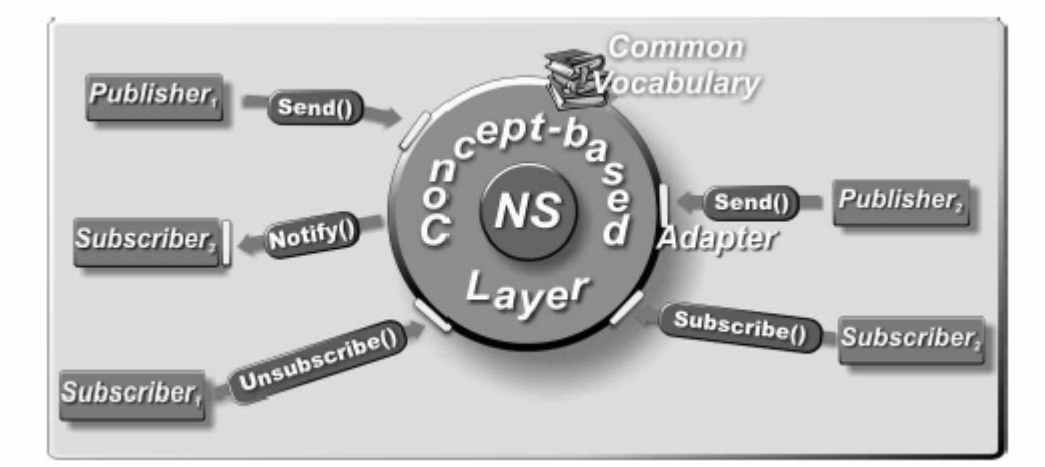


Figure 2: Concept based Approach (Antollini et al., 2006 p.4)

As depicted in the Figure 2, the notification service (NS) is wrapped by the 'concept based layer' introduced by the architecture. Clients do not interact with the underline notification service, but uses higher level API provided by the 'concept based layer' in communication. 'Common Vocabulary' provides domain specific vocabularies used by the participants. 'Adaptors' in each side of the communicating parties do the necessary transformations by resolving any vocabulary issues.

MIX (Metadata based Integration model for data X-change) data modeling process is used to achieve 'Information Integration' under this architecture. Data objects should be associated with a concept label that describes the data. These labels should be picked from the 'Common Vocabulary'. This technique ensures the meaning of the data objects are well understood by the participating applications. MIX architecture is basically composed of four sub-components namely conversion and matching operations, events, expressions and semantic context. Applications and adaptors can use the functionality provided by MIX as appropriate to do the translations. Events created by publishers can be decorated with available semantic information. Subscribers can use MIX to create expressions to indicate the type of events they are interested. Adaptors are responsible of performing the conversions and matching operations required for data interoperability.

'Application Integration' is performed through the exposed API of the framework. This means that the integration happens at the code level of each communicating party. Producer applications can obtain a publisher class from an 'abstract factory' that makes the underline notification service transparent to the application. Consumers can subscribe to the API by providing its listener instance.

### Evaluation

The ‘concept based layer’ proposed by this architecture well focuses on resolving the ‘Information Integration’ issues involved. The adaptors have the clear responsibility of solving the semantic conflicts of information when crossing boundaries. Therefore this architecture can be effectively used when integrating systems with heterogeneous domain schemas. Tagging based MIX approach can be seen as an extensible ‘Information Integration’ strategy that can help in complex multi-channel integrations.

The architecture suggests the ‘Application Integration’ to happen through an API at the code level. Therefore the presented implementation in the paper only supports integration of systems with same programming language (currently Java). Different APIs have to be developed for wide variety of other languages if the system to be practically useful. Anyhow, the development of these APIs can cause their own interoperability complexities. Solving this interoperability issue can be very challenging in practice, but the paper has no discussion of solving the same.

On the other hand, event notification systems are asynchronous by nature. But there can be systems that only support synchronous communications. In such scenarios, it may be required to employ an additional abstraction layer that mimics synchronous interaction on top of this asynchronous channel.

As per the discussion, it is evident that this architecture has focused ‘Information Integration’ aspects over ‘Application Integration’ aspects. Even though the integration of heterogeneous domain schemas can be well supported, this architecture may struggle when integrating systems that are operating on different technical platforms.

## *Conclusion*

Defining an effective architecture for seamless integration is non trivial, because the interoperability ground to be covered is extensive. There are lots of aspects that need to be considered before selecting a particular architecture for enterprise integrations. Through out the report we have looked in to two such primary integration aspects namely ‘Application Integration’ and ‘Information Integration’. As discussed, the technology heterogeneity complexities as well as information heterogeneity complexities should be effectively handled by any integration architecture to be successful.

The ‘Information Brokering’ and ‘Pub/Sub Notification’ architectures discussed above have own strengths and weaknesses. These architectures are evaluated against the balanced realization of ‘Application Integration’ and ‘Information Integration’ aspects. The ‘Information Brokering’ architecture has paid most of its attention to the technical characteristics of the integration. It has paid little attention to enhancing the quality of the information exchanged. On the other hand ‘Pub/Sub Notification’ architecture was built with a quite inflexible technical view but has a good support for

'Information Integration' aspects. According to the overall discussions, it is evident that each of the discussed architecture is biased towards one of the ends.

Even though the focus of this report was to evaluate against 'Application Integration' and 'Information Integration' aspects, there are dozens other important concerns for successful integrations (e.g. security, scalability, coupling, performance, etc.).

Both the discussed architectures are modeled by the authors with a single architectural pattern in mind. It is clear that these architectural patterns have even failed to deliver the requirements under our discussion. Therefore it is evident that simple single view architecture would not be capable of catering all the demands of integration seamlessness. Commercial grade integration architecture would require multiple architectural styles and patterns to be employed to support these complexities. For that reason it is vital to pay a careful attention to all these aspects before deciding on a particular architecture for your next integration project.

## *References*

Antollini, M., Cilia, M., Buchmann, A.., 2006, Implementing a High Level Pub/Sub Layer for Enterprise Information Systems. [Online]. Available At: http://www.dvs.tu-darmstadt.de/publications/pdf/AHighLevelPubSubLayer.pdf [Accessed 20th December 2007]

Grundy, J. et al., 2003. An architecture for efficient, flexible enterprise system integration. [Online]. Available At: http://www.cs.auckland.ac.nz/~trebor/papers/GRUN03A.pdf [Accessed 20th December 2007]

Pollock, J.T., 2001. The Big Issue: Interoperability vs. Integration. [Online]. Available At: http://isotc.iso.org/livelink/livelink/fetch/-1287650/2426503/XML_semantic_interoperability-Integration_Pollock_1.pdf?nodeid=2425856&vernum=0 [Accessed 21st December 2007]