

The Resource for e-Business and Application Integration

October 2001  
www.eaijournal.com



Modulant Solutions Presents:  
**The Big Issue:**  
Interoperability vs. Integration  
By Jeffrey T. Pollock

The **Age**  
of the **XML**  
**Database**

# The Big Issue: Interoperability vs. Integration

By Jeffrey T. Pollock

Despite the hype around the newest generation of Enterprise Application Integration (EAI) and process automation products, many IT managers remain cautious about adopting them. There's also growing press coverage about a Standish Group report that 88 percent of integration projects fail. Sure, there are many project-related causes for these failures (implementation process, poorly managed risks and requirements, etc.), but there's also something fundamental missing from most available technology solutions that would greatly improve the odds for integration success — complete application interoperability.

Interoperability-based approaches focus on the exchange of meaningful, context-driven data between autonomous systems. Integration approaches, in contrast, typically attempt to build a monolithic view of the enterprise. They integrate processes and applications at the event and message levels so multiple systems become one logical unit. The two approaches can be complementary, but an interoperability solution would usually focus on how to exchange the minimal amount of information (not just data) to make two or more systems interoperable. Traditional EAI focuses on making two or more systems integrate by sharing Application Program Interfaces (APIs), messages, or tightly

coupled workflow.

What do interoperability solutions require? As you can see in Figure 1, interoperability is comprised of both application integration and information integration. Application integration is the technology solution, where most of the product development effort has focused. Information integration is the linguistic, social, and philosophical solution, where technology is only beginning to catch up with academia. For any moderately complex integration project, you'll require both types of solutions. One core problem is that most EAI vendors have an anemic, if any, information integration strategy.

## Semantic Interoperability

Application integration is only half the solution because getting two or more computer systems to share bits and bytes is no trivial task. Only recently has the technology become widely available to say for certain that any computer system (with at least one interface, human or otherwise) can share data with any other computer system. Often, the integration choices are so abundant as to be confusing.

Information integration is only half the solution because each computer system in an interoperability scenario possesses its own view of reality that includes domain representations,

semantic particularities, knowledge structures, and programmatic constraints that limit how it communicates. Academics in various research laboratories (including MIT, Stanford, Berkeley, Georgia Tech, and others) have been working for more than a decade on creating tools that enable the interchange of information in its entire multitude of forms and intricacies.

Implementers of EAI and custom one-off integration solutions have also faced this problem. Highly complex integration projects can include a whole range of semantic conflicts that few (if any) traditional EAI tools will accommodate without creating an extremely brittle integration framework.

Consider the financial analyst at

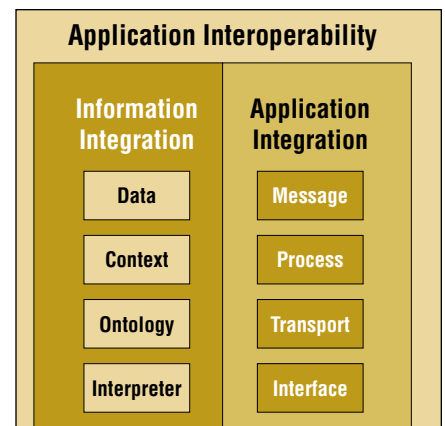


Figure 1

Daimler-Chrysler, Jayne, who needs to find a cumulative roll-up of net income, net sales, and total assets for fiscal year 2000. She normally uses the Worldscope database, but recalls a co-worker telling her that two other applications also contain relevant information. Jayne starts out as she usually would, but encounters problems when querying the two new applications. After some trial and error with the queries and reports, a few calls to system administrators, and asking her co-worker some questions, Jayne determines that the crux of her problems are that the information stored in the various systems don't conform to the same standards. In all three applications, the company name is different (DaimlerChrysler, Daimler Benz AG, and Daimler-Chrysler Corp.) and in one of the applications, the financial scale factor was different (scaled in 1,000s). The third application was using a different currency, so historical exchange rates would be required to reconcile the statements.

These examples are representative of a few of the types of semantic conflicts that can arise in the typical integration environment. Most EAI tools will only resolve these types of conflicts in a case-by-case basis with custom-scripted rules, custom coding, and in a point-to-point manner whereby the source is programmatically "aware" of the target.

In the remainder of this article, we'll look at the basic components of an application integration solution and information integration solution. We'll see that application integration solutions focus on the technology problems involved with connecting two systems, whereas information integration solutions focus on resolving information conflicts by creating a semantic interoperability framework.

### Application Integration

Application integration is the process of linking disparate software systems to become part of a larger system. You can link systems at several levels: data, application, transaction, process, and human. These are commonly represented by a hierarchical pyramid structure as seen in Figure 2. The higher you go on the pyramid, the more complex you can expect your integration scheme to get. At the top, the solution to a given integration problem becomes human-centered. This is by far the least desirable because of the

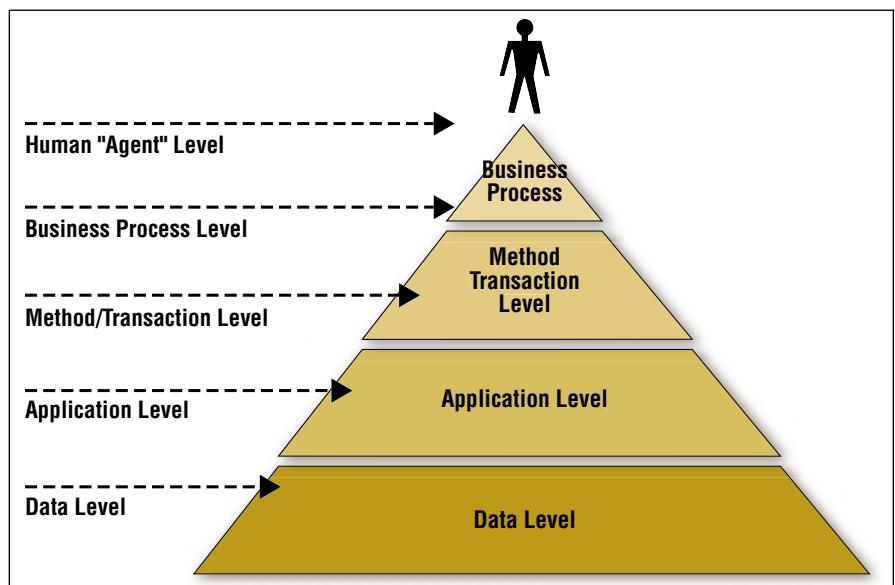


Figure 2

manpower required to complete it and the typically high error rate.

The trap most system integration companies fall into is focusing on the various technical aspects of this pyramid instead of the quality of the integration itself. Most vendors will say that their given product will provide the highest quality integration, but look a little closer. You can see that most solutions fail to tackle the full scope required by complex integration problems — transformation of information, not just data.

### Interface Technologies

The interface technologies that include adapters, connectors, and agents have proved to be challenging. The interface is responsible for providing a view into and out of the source and target systems in a cross-system exchange. The sheer number and variety of platforms and systems, combined with the variety of EAI integrators, have created an environment where thousands of proprietary interfaces exist — making it all confusing for the typical enterprise integration project.

The distinction between a basic proxy, a connector, an agent, or an adapter is sometimes purely semantic, but generally relates to how complex the interface is. For instance, a proxy is typically a simple gateway that emulates application behavior and maps to the integration environment, whereas a connector can be a fairly "fat" piece of code that performs basic translations, provides security mechanisms, exposes

process APIs, and knows how to use different transports.

Most integration and interoperability systems require some kind of interface, even if it's only to establish a transport mechanism such as Remote Method Invocation (RMI) or Hypertext Transfer Protocol (HTTP).

### The Process Conductor

The process conductor is simply the control element that orchestrates the many messages that are being transmitted at a given time. In other words, it's the instruction set that tells different source systems, target systems, and integration servers what order to do what steps in. For most EAI tool vendors, this is a programmatic, scripted area where you focus your business process tool. By providing a graphical interface to this instruction set, a tools vendor can provide business analysts a way to control the delivery steps within the context of a business process that has already been defined.

As you can imagine, this is a highly volatile place because the business process typically changes quite frequently. Shifting customer demands, new operational efficiencies, technology adoption, competition, and mergers or acquisitions cause fundamental shifts in a business process. It's the process controller that provides a message-level view into this exchange and lets end users alter a process at runtime.

### The Transport Medium

When discussing interoperability, integration, and communication, there must always be a discussion of medium. For human communication, our mediums may be as diverse as the air transmitting sound waves, light to reflect body position and movement, or even e-mail to transmit words on a page. For machine communication, the medium is usually far more concrete. Typical mediums for EAI communication include Systems Network Architecture (SNA), HTTP, and proprietary message transport protocols. Other mediums for composite EAI approaches include RMI, Internet Inter-ORB Protocol (IIOP) and other component/object transport technologies.

At a core level, the medium is like choosing between FedEx, the U.S. Postal Service (USPS), or UPS. It's a delivery mechanism. The message itself is independent of the medium and the medium doesn't care if you are shipping boxes or letters.

### The Message Content

The message content comes in two parts: the wrapper and the data. The message wrapper is what specifies a multi-part Electronic Data Interchange (EDI) exchange, an eXtensible Markup Language (XML) formatted exchange, or an object-based exchange. A wrapper may also be considered as a native database exchange via Open Database Connectivity (ODBC) or Net8.

The EAI tool market is sometimes sliced into categories based on what kinds of messages are used. Figure 3 shows how messages are commonly exchanged. The various approaches to message communication also lend themselves to the classification of core EAI categories:

- Composite integration (application object and component interfaces)
- Multi-step integration (message-centric queuing systems)
- Data consistency integration (native database interfaces and shared schemas).

Each of these methods is concerned with the type of messaging that's involved and not with the transformation itself. As long as we're talking about message technology or object technology, we're really talking about integration via the message style. We've not yet begun to discuss the integration

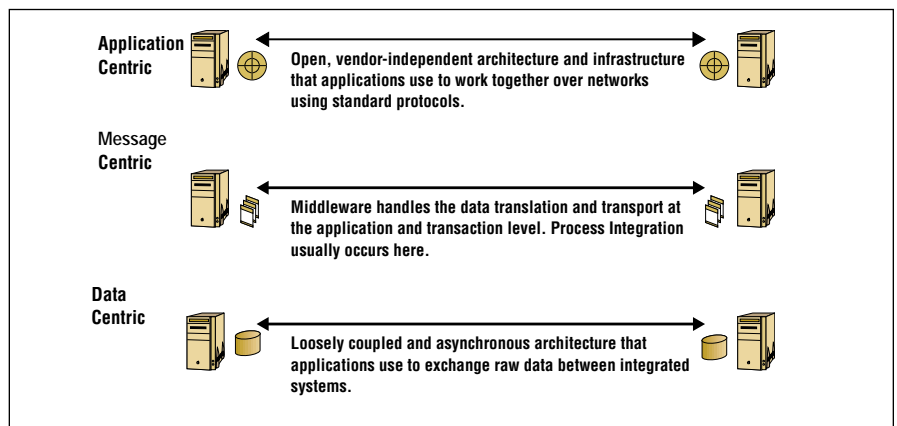


Figure 3

of information.

### Information Integration

When you move into complex environments that require more than basic translation capabilities, most application integration tools fail. Dozens of projects come to mind where integration-only tools have failed to meet project objectives. It's in these complex environments where companies require interoperability

**When you move into  
complex environments  
... most application  
integration tools fail.**

tools that offer a robust set of information integration techniques.

Fundamentally, an information integration technique will focus on a complete picture that delivers more than data, objects, or messages. It will focus on conveying meaning to create true fluency. Meaning, in a practical sense, is about using metadata, business rules, and user-supplied application context to facilitate a robust information transformation. You actually give the software information about the environment, domain, and metadata that a programmatic algorithm can use to manipulate information from one system so another can fully understand it.

Most application integration software focuses on the transportation of data/objects/messages between heterogeneous systems. They'll perform basic translation tasks such as string manipu-

lation, mathematical operations, conversion routines, and filtering. But they won't facilitate transformation of an entire schema domain representation to another partially compatible schema domain representation. The central tenet of a solid information integration approach is to maximize the exchange of meaning in terms the target can understand. Another way to say it is that information integration is focused on preserving the semantics while transforming the context. Interoperability platforms that use information integration techniques should do that.

### From Data to Wisdom

The easiest part of the information integration equation to understand is the data itself. Data is the fundamental building block upon which relevance is built. By itself, data has no inherent meaning. For example, what does the symbol "21" mean to you? Clearly, this symbol means different things if you are in Las Vegas, if you are 20 years old, or if you are at a military funeral. The numeral is a data point, but in a domain library of gaming rules for blackjack, 21 is information.

The experience strategist Nathan Shedroff presents, in his recent book, *Experience Design*, a revised view on the classic information hierarchy that's the most common way of describing the progression from data to wisdom. Data is primary. Data with meaning is information. See Figure 4 for a graphical representation of this progression.

Today's interoperability solutions must "get it." The difference between data and information is critical to the effective, efficient exchange of information between heterogeneous systems. You can do that by embedding meaning

in your data representations.

## Ontology

Ontology is defined as a particular theory about the nature of being or the kinds of existents (things that can exist). For the purposes of defining information in computer systems, most ontological groups work with this definition: an explicit specification of a conceptualization. In either case, the ontology provides meaning to data because it puts raw structured or unstructured data in the form of a structured conceptual specification. Practically speaking, this is valuable because the specification itself is about a conceptual understanding of a broad information domain. In contrast, a typical application schema is a structured concrete representation of data points that actually exist within an application context.

Ontology is being used to describe information and also as a tool to resolve semantic heterogeneity conflicts. Some conflicts are purely structural: XML to Unified Modeling Language (UML), for example. Even more common is a structural conflict called impedance mismatch — the relational to object conflict found in most three-tier application architectures. Semantic conflict, on the other hand, is about the conflict between the content and intended meaning of heterogeneous information systems. This comes in three forms:

- Confounding conflicts, where information appears to have the same meaning, but doesn't
- Scaling conflicts, where different reference systems measure the same value
- Naming conflicts, where naming schemes differ significantly (e.g., synonyms and homonyms).

However, the representation of data in ontology isn't a complete solution. So many forms of ontology exist that, in and of themselves, they're not valuable. The ontology has to be connected to your real application with metadata and context descriptions.

## Metadata and Context

Metadata is data about data, but metadata comes in many shades of gray. There's metadata that DBAs handle in the database and then there's the data about the database, the DBAs themselves, the company, the business domain, and so on.

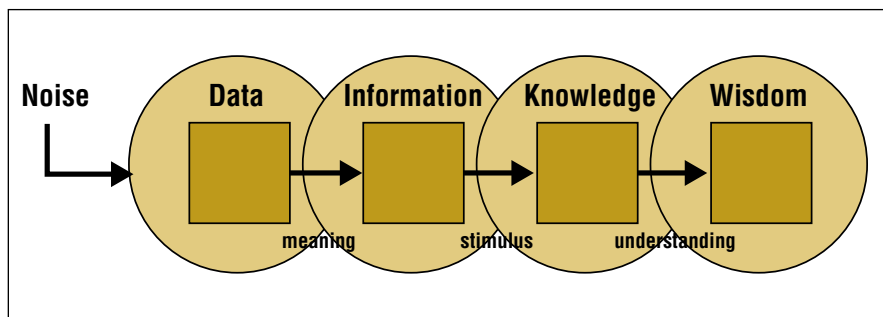


Figure 4

Typical metadata will include data types, data relationships, table space information, column names, precision, length, primary/foreign key constraints, and other database-centric metadata. Such metadata is fine for most business intelligence applications to enable people to perform better queries. This kind of metadata is much better than most tools that only give you minimal metadata capability.

To fully enable rich communication between systems and more intelligent forms of analytic applications, you need to extend the metadata you care about to include human-defined context, busi-

## More companies are beginning to take advantage of metadata.

ness rules, and metadata. It's this kind of metadata — such as problem domain, company name, database name, database location, application name, and information constraints — that can provide meaning to a given integration transaction. This meta-metadata, sometimes referred to as context, needs to be a flexible element of the integration that can be defined by the database administrator, mainframe operator, or problem domain expert.

So what do you do with the data, its meaning, and its context now that you have it? You transform it.

## Transformation

Most vendors will market their trans-

formation capabilities as a core component of their integration frameworks. Dig deeper, however, and it's easy to see that they're really selling a set of translation tools that perform string manipulation, mathematical functions, conversion routines, filtering, simple mapping, and more. This style of translation is almost always on the instance data itself. But if your product does not use the metadata, preferably the context, of the transaction and an ontological representation of the information, then it's probably not transforming the information. This is fine for some scenarios. Not everybody requires a baseball bat if they only have to swat a fly. In other scenarios, though, translating the data isn't enough. You have to alter the meaning of a whole information set to transform it. You cannot transform just one data element. Transformation techniques must ripple through the data via ontology, relationships, and information trees to alter or preserve the meaning across various forms.

Ultimately, this is a practical application graph theory and requires ontological mapping constructs. It solves the business problem of interoperability instead of the technical problem of integration. In other words, the academics have given us implementers the basic mathematics to perform structured manipulation of ontological trees of meaning in data.

The recent proliferation of transformation and context-sensitive tools show that customer feedback is forcing vendors to re-examine their integration strategies. People are getting tired of the same old EAI technology solutions, repackaged in a business-to-business (B2B) wrapper. More companies are beginning to take advantage of metadata and, more important, business context during the integration process. This approach relies on transformation, not translation, to enable full-scale interoperability.

## Summary of Basic Criteria

A full-featured interoperability solution should include:

- **Transport service** — A mechanism or protocol to move messages from one place to another.
- **Message container** — A codified wrapper and structure for the data and metadata during the exchange. This could be achieved with proprietary containers (EDI), XML, Common Object Request Broker Architecture (CORBA), Enterprise JavaBeans (EJB), Distributed Component Object Model (DCOM), and others.
- **Integration interface (transport or message)** — A way to get your data in the container and send it over your protocol.
- **Process controller** — A conductor that defines and orchestrates multi-step, batch, publish/subscribe, and request/reply transportation of messages within the scope of analyst-defined business processes.
- **Data encapsulation** — The encapsulation of the raw elements of an information exchange.
- **Ontological information representation** — A specification of a conceptual model that your data sources are represented by. This representation may be accomplished by a direct mapping or a third conceptual specification of a higher order than source and target specification.
- **Context and metadata** — Roughly speaking, the data about the data

from the application's perspective is the metadata, while the data about the data from a business perspective is the context.

- **Information transformation** — A capability to manipulate entire sets of data, relationships and all, based on information contained in metadata and user-supplied context information. Typically, this requires a conceptual model using ontological mapping techniques.
- **Data translation** — A set of algorithms and user-modifiable scripts that enable your integration tool to translate based on pre-defined rules (e.g., string manipulation, mathematical functions, conversion routines, and filtering).

## Conclusion

The distinction between integration and interoperability is really a case study in architecture. The principle that any structural or software architect considers when building a framework is the separation of concerns into layers of responsibility. To keep autonomous systems discrete, the interoperability architect will focus on making the information and systems interoperable while avoiding a monolithic, tightly coupled integration effort.

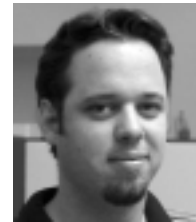
Consider a Taiwanese investor who needs to send an urgent message to a Texas oilman about his plans to invest in a Chinese drilling expedition. The Taiwanese investor writes the letter in Mandarin, packages it in a FedEx letter container, and sends it to Texas. The oilman's secretary recognizes the source of

the letter and opens it. She gives the letter to an interpreter in a manila envelope. The interpreter then speedily issues the message to the oilman, who then asks the interpreter to draft a response in Mandarin. The secretary sends the reply back via UPS next-day air.

There's a clear separation of concerns here. The letter carrier (FedEx and UPS) is the transport. The envelope and FedEx package are message containers. The secretary is a connector for the message (the letter). True interoperability here depends wholly on the interpreter. Ultimate interoperability requires metadata about the message and rules and context about the source and target systems.

The market is starting to open up to provide this next generation of capabilities. As we focus more on application interoperability, semantic transformation, and ontological mapping, business will benefit. ☺

### About the Author



*Jeffrey T. Pollock is a technology evangelist and product architect at San Francisco-based Modulant Solutions, a next-generation enterprise interoperability software tools provider. He has successfully designed and built application server-based middleware solutions since 1996. e-Mail: jeff\_pollock@yahoo.com; Website: modulant.com.*