Chapter 2

Information Infrastructure Issues and Problems

KEY TAKEAWAYS

- Traditional middleware solutions are built on outdated technology.
- Today's businesses are wising up to the importance of the information contained in the messages—as opposed to the technology surrounding the messages.
- Plumbing, or integration, is necessary but not sufficient. Without it, enterprise software communication would not be possible. With it alone, communication is inefficient, expensive, and lacking focus on what matters most—information.
- The real challenge for enterprises today is to build an infrastructure that enables information, in all formats, to be utilized freely at the right time and place.
- Dynamic, adaptive middleware architectures will rely on semantic interoperability.

Defining holistic middleware architecture may not be the first order of business for most CEOs, but cutting costs and improving business relationships surely are. So, although the question of semantic interoperability may be technical geek speak for most, software architects and CIOs should take heed of this emerging vision. It could make you a corporate hero.

In this chapter we will examine the roots of application integration solutions and discover the reasons why they so frequently become money pits. Key technical components of typical EAI solution sets will be discussed, highlighting specific areas that lead to high maintenance costs and problematic configuration management. It will be made clear why enterprise information, in the form of structured, semi-structured, and unstructured content, should take primacy over all other technical middleware concerns. Finally, semantic interoperability will be introduced as a necessary complement to typical integration solutions because it can alleviate the cost of deploying existing integration solutions.

Adaptive Information, by Jeffrey T. Pollock and Ralph Hodgson
ISBN 0-471-48854-2

LIMITATIONS OF CURRENT TECHNOLOGY APPROACHES

Failed integration projects are common today. The sad fact is that the vast majority of enterprise efforts to link trading partners, unify corporate data, extend business processes, and enable IT system collaboration end with unsatisfactory results. Analysts report that over 80% of all data integration projects have failed or significantly overrun budgets.[1] These numbers must surely be taken with a grain of salt, but they do indicate that something is amiss.

Off-the-Shelf Integration Products

Enterprise application integration (EAI) and business process management (BPM) companies offer integration platforms that provide adapters, transport, work flow, and transformation capabilities to companies seeking to integrate disparate applications.

EAI approaches have been successful because they create a centralized management paradigm for controlling information flows throughout the enterprise. This is vastly improved from the era before EAI, when no alternatives to spaghetti integration existed at all. Additionally, the clear separation of architecture layers in most EAI tools enables these vendors to consistently create new value-added services such as B2B, I2I, A2A, supplier networks, analytics, etc.

However, EAI-type approaches are struggling. IT managers have found problems that become impediments to widespread adoption. The typical EAI solution requires a very tightly coupled environment, which severely restricts the flexibility, agility, and adaptability of the integration framework. This disadvantage negatively impacts the portability of the business and process data contained within the framework.

In addition, EAI solutions include adapters that require significant amounts of custom code to facilitate any integration that is not supported by prepackaged templates—which turns out to be most integrations. For EAI or BPM solutions to work properly, custom code must be written to connect every application within the system, at the data layer, the adapter layer, or both. This custom code is extremely expensive to implement and maintain.

Furthermore, when a company selects an EAI application it is virtually forced to remain with that one vendor to continue integrating new applications—which usually means buying more services, tools, and adapters. Total cost of ownership (TCO) is very high for most EAI solutions because the solutions are tied so heavily to a business's processes. Every time an application or business process is modified, changes must be made to the integration framework.

After years of buying EAI solutions, Fortune 1000 corporations and government agencies have created an entrenched industry that cannot be easily displaced. Alhough EAI technologies are entrenched, they are not the final word for linking

[1] The Big Issue, Pollock, *EAI Journal*.

incompatible business systems. The ongoing focus of most EAI vendors on proprietary process-based solutions detracts from their overall utility, and future promise, precisely because TCO customer concerns often derive from other causes.

Web Services and Service-Oriented Architectures

The Web Services framework has recently taken the computing industry by storm. Sun, Microsoft, IBM, and most other large software vendors have embraced the concepts and languages that underlie the Web Services model. The combination of UDDI, WSDL, and SOAP forms a triad of technologies that will shift the entire market toward service-oriented architectures (SOA). Together, these technologies provide directory, component lookup, and exchange protocol services on top of an HTTP or SMPT network protocol. This capability translates into a loosely coupled physical communications channel for moving messages around.

However, Web Services are not without shortcomings. From a business perspective service-oriented architectures do not yet solve several crucial aspects of the integration problem such as robust transactional support, adequate security features, improved directory services, and sufficient architectural soundness to provide mission-critical performance.

Perhaps the most significant improvement opportunity for Web Services is in the area of information management and schema transformation. Fundamentally, Web Services technologies handle messages in a loosely coupled manner, but they will not enable the recipient to understand the message that has been sent. With Web Services this part of the exchange relies on custom-coded solutions or widespread community agreement (rarely achieved) on some kind of document exchange standard.

Therefore, the recent and rapid adoption of Web Services continues to highlight the pressing need for semantic interoperability technologies.

Data Warehouses and Metadata Management

Performing systems integration with a data warehouse involves creating a central database that is the primary owner of enterprise data—and the unified data source for client applications. As a way of enabling applications to share information, the data warehouse appears conceptually sound, because it puts a strong emphasis on collecting and unifying enterprise data.

However, implementing this approach leads to a series of challenges that undermine this solution. For one, creating a unified data model for the central repository is extremely time-consuming and generally not adaptable to change. For another, homogenized data is frequently not as valuable because context and relationships are typically lost. Finally, updating and maintaining the data in a central repository on a real-time basis is extremely difficult because of the diversity and complexity of layering multiple business process rules and the cleanliness issues of the data. Because of these shortcomings, data warehouses are predominantly used for archiv-

ing historical data, mining data, and performing trend analysis, and not as an application integration solution for operational data.

Some companies are beginning to provide metadata management capabilities that involve moving, viewing, and extracting data from databases. Typically, such software solutions offer some query capabilities and the ability to mediate requests from a number of different data stores. In addition, many of these companies focus directly on analytics and decision support systems.

From a technology perspective, these kinds of solutions are not ideal for a generalized integration platform in the enterprise for a number of reasons. Metadata management tools tend to be database-centric: built on relational database systems, usually ignoring the need for API-based integration and sometimes even ignoring robust XML support. Their vision of metadata is frequently limited, two-dimensional, and fails to include the much richer form of environmental metadata that a robust integration solution must accommodate.

Portals

A portal is sometimes used within an organization, or across organizations, as a single entry point for a number of systems behind the scenes. Leading edge companies that provide portal-based federated search and retrieval capabilities typically have serious limitations with regard to data variety and deployment options. Technologies that do exceptionally well at searching, pattern recognition, and taxonomy generation on unstructured data don't work very well with structured sources. Generally speaking, these tools lack sophisticated ways to account for embedded contexts (by way of the inherent structure of RDBMS, object, or hierarchical data sets), native metadata, or precision in structured query formatting. Aside from technical limitations, concerns with vendor lock-in and handling machine-to-machine interoperability (as with traditional EAI) still go undressed with the portal-based approach.

Systems Integrators—Custom Solutions

Systems integrators get nearly any IT job done. Major consultancies like BearingPoint, CGEY, AMS, IBM Solutions, and Lockheed Martin Consulting are the masters of brute force approaches to solving technical challenges. In fact, they have a vested interest in solutions like this because they often help create one-off point-to-point integrations—which they might be contracted to maintain for years to come. This is possibly the least efficient approach for a technically advanced, elegant, and easily maintainable technology solution, but, sadly, it is probably the most common IT integration solution.

Standard Data Vocabularies

As an approach to application integration, standards-based interchange involves agreeing on a specific standard data format for recording and storing information. One of the earliest and best known attempts at standards-based interchange is EDI, used to exchange information among trading partners. More recently, many existing and newly developed standards have shifted into the more modern technology of XML. Whereas EDI was used almost exclusively for intercompany data exchange, XML standards are evolving as a way to exchange information both internally and across multiple enterprises.

Using a standards-based approach means that each standard develops based on input from a wide community and typically has cross-organization collaboration and support. Such a standard represents advance agreements on the syntax and meaning of a given set of exchanges. However, standards have a difficult time responding to business and environmental changes that occur rapidly in most industries. Furthermore, the politics surrounding standards creation often undermine their ubiquity and effectiveness. Perhaps the largest problem with many standards is that local business context may be lost because each system that uses the standard has to convert its information to a common, specialized "view of the universe." Historically, the standards-based formats either have proliferated almost out of control (XML-based vocabulary standards) or have been very narrow in focus (X12, EDIFACT), creating a problem for IT managers attempting to decide on one or only a few.

Many people mistakenly believe that XML will solve all application integration problems. The reality is that XML is simply a mark-up language that enables information to be "carried" from one incompatible system to another. The use of XML alone will not resolve differences in how information is processed by different systems (i.e., how the "semantics of the information" are interpreted). XML is simply one piece of the puzzle, not a complete solution. Standards have their place in the IT toolkit, but standard vocabularies will not be the primary vehicles of information between systems that need a flexible and robust method to communicate with each other.

TRADITIONAL APPLICATION INTEGRATION

In the early 1990s MRP/ERP vendors were making a killing by selling the promise of integration-proof monolithic enterprise systems. The idea was that if the enterprise would just buy all its software from one vendor, there would be no need to integrate anything. But a threat to their plan for world domination was looming on the horizon. Message-oriented middleware (MOM) arose out of the telecommunications industry with the kickoff of the Message Oriented Middleware Association (MOMA) in 1993. Even as early as 1985 commercial work was beginning on DACNOS (IBM and the University of Karlsruhe), software that utilized an asynchronous message-driven communication model.

As initially conceived, the MOM was intended to solve the messaging problem with a simpler approach, document-like messages, than CORBA—another key mid-

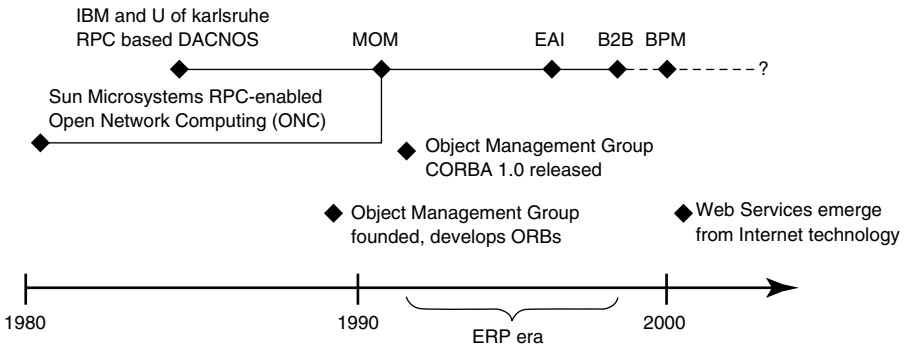

Figure 2.1 Synopsis of message-oriented middleware history

dleware technology of the day. Early MOM technologies eventually adopted many extract transform and load (ETL) capabilities and spawned a new service offering that, in 1997, was renamed enterprise application integration (EAI) software. EAI companies began to offer more comprehensive end-to-end platforms with adapters, transport, and transformation capabilities to companies that needed to extend the reach of their existing systems. More recently, EAI companies started to layer even more management tools, like partner, process and standards management, on top of their transport systems in an attempt to rebrand as B2Bi or process integration vendors.

Modern EAI approaches have been successful because they have adapted to the needs of exchanging data within business processes and created a centralized way to control information flow from a single point—instead of through distributed code and granular interface, as with CORBA. As mentioned above, the clear separation of concerns in the typical EAI architecture allows for the easy layering of new value-added services such as B2B, I2I, A2A, supplier networks, and analytics.

Key Components

Most integration infrastructures are built around two key concepts: events and messages. This is the logical equivalent of a letter and the action of dropping it in a mailbox—with dozens of variations on that simple theme. Users can define messages that are broadcast to everyone or messages that go to specific individuals. They can route all messages via a central hub, like a post office, or they can bypass the hub and go straight to the recipient. Services such as translation routines can rewrite the messages' content into a different "language" for the recipient or convert values to different scales and formats . . . and so on and so forth.

EAI tools have already been well written about in many books, and the authors have no intention of providing deep technical insight into common EAI architectures here. However, because so many concepts are important to grasp, it will be useful to briefly cover the following major components of the typical EAI tool set:

- **Adapter**—small or large units of code that reside physically close to the participating source application to convert its native protocols, vocabulary, and languages to formats that are known by the information bus or central hub
- **Transport**—the over-the-wire protocol that is responsible for the physical movement of data, message and content, from one system to another
- **Message**—instructions, which accompany the data from one physical location to another, that can contain work flow rules, metadata, and queries
- **Process Controller**—sometimes known as an event manager, depending on the topology that is employed, that listens for messages and can determine the appropriate actions to take with them—such as sending them along to other systems or launching other subroutines to work on the data

Many other components can be included in integration packages, but most of them include some variation on these components. Many integration solutions also include a transformation controller. Most EAI tools use application code to perform data transformations. Usually an integrated development environment (IDE) is provided to augment a simple graphical user interface for building these data transformations. Code is then deployed in the adapters, the hub, or both. In practice, some tool vendors opt not to provide sophisticated transformation capabilities because they deliver highly specialized software targeted to specific industry verticals and deem sophisticated transformation capabilities unnecessary.

Disadvantages and Concerns

However, EAI-type applications have struggled in their own right because IT managers are wary of several key disadvantages that impede adoption. EAI solutions create tightly coupled integration environments that eventually restrict the flexibil-

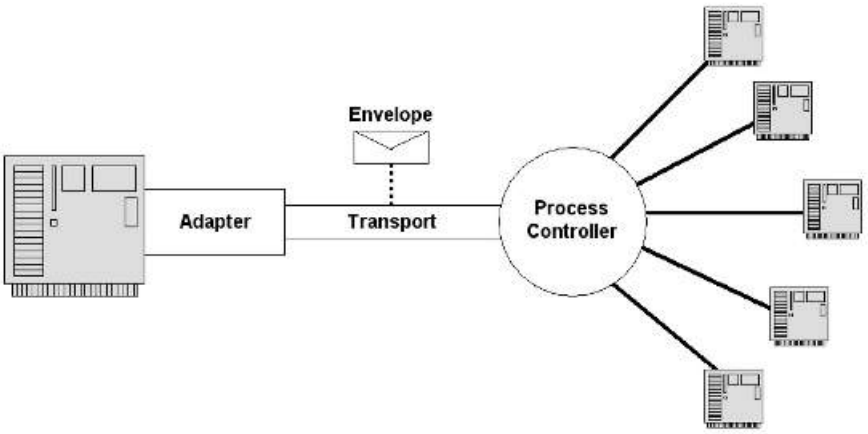

Figure 2.2 Simple EAI architecture view

ity, agility, and adaptability of a enterprise. Tight coupling impacts the bottom line in both lost opportunity cost and high maintenance costs. Integration vendors would have IT managers view the enterprise as a single monolithic system of subsystems. In this view the middleware itself is the enterprise operating system—managing processes and messages to subprocesses. But like a computer operating system, this EAI paradigm only works well if every subsystem implements the same vendor's software.

Other significant concerns and disadvantages include the amount of custom code required for the application adapters used to facilitate integration. To summarize, the two most significant concerns are not technical; instead, they are business issues:

- Vendor lock-in
- Total cost of ownership (TCO)

However, these concerns stem widely known EAI technical limitations such as:

- Proprietary interfaces for adapters, process control, and sometimes even messaging and event protocols
- Tightly coupled architectures resulting from hard-coded interfaces between adapters, hubs, and message buses
- Clumsy interfaces for customization of process and data transformations
- Static, precompiled, or prescripted, routines for event management, data transformation, and resource management

These are technical legacies of the EAI's roots grounded in the remote procedure call (RPC) style of distributed computing. RPC epitomized the tightly coupled interface, and the tightly coupled interface is what makes responding quickly to business changes so difficult.

Application Integration Trend: Target the Vertical Markets

A visible trend in the EAI and B2B industry is the specialization of solutions to specific verticals. For example, a single vendor may repackage the same tool set to target health care, automotive, or customer relationship management market segments. This gives an illusion of a tight match between customer requirements and the vendor's tool set. For better or worse, the behind-the-scenes rationale for this kind of specialization has less to do with actual technology innovation or benefits than with the sales team's quarterly numbers.

However, the vertical specialization of these integration solutions also indicates a growing trend with integration customers—they are beginning to care more about their data and the processes in which the data participate. Therefore, companies are beginning to demand that middleware vendors offer solutions that are specifically targeted toward their industry verticals.

These customer demands stem from highly publicized failure rates of EAI and the strong desire to manage risk, under the natural assumption that if a solution is specialized for a particular kind of data and process it will have better odds of success. However, the technical causes of failures in the past—tight coupling, poor data management, and proprietary interfaces—are still present in the vertically targeted systems. Eventually, the high total cost of ownership and vendor lock-in issues will resurface and further disrupt the EAI industry.

Integration's Core Issue

In reality, and from a technical perspective, when it comes to middleware—the black box with no face running between lots of gray boxes—data is just data and process is just more data. Middleware is truly a horizontal infrastructure solution that requires few, if any, capabilities that are specific to a given industry. If some middleware solution works really well in one industry it has really strong odds at working well in another.

This is true because most middleware components are not aware of details about specific industries, such as data and process labels—they're busy looking at messages and putting stuff on queues. The rest of the middleware components that do operate on industry-specific data or processes can't inherently discern the difference between XML tag that says `<customer_number>` and `<rfid_part_num>`—the software engineers who wrote the programs or configured the middleware are the ones who really know the difference!

This sort of reliance on human factors is what leads to integration's inefficiencies. The "verticalization" of the EAI industry is sure to sell more products, but it will only add minimal value to the end users. Thankfully, those who have become aware of these facts are beginning to refocus on the importance of the data, concepts, and information that actually move through integration's pipes and fittings. It has been the unfortunate lack of focus in these areas that has caused the further popularization of the idea that integration middleware is just a commodity.

APPLICATION INTEGRATION AS PLUMBING

Plumbing is a good thing. Plumbing provides the infrastructure to connect key services in a house. But plumbing alone does not make a house. Plumbing is a commodity. However, plumbing is also a necessary requirement in a home.

Although many integration, B2B, and process integration vendors will tell you that their software is much more than plumbing, if you look closely at what value they actually provide, chances are it will probably primarily involve the movement (and management) of message, documents, or data between different physical instances of enterprise software. In other words, plumbing.

Why Connecting Systems is Necessary

Physical movement of messages and transactions among software systems has been accomplished in a wide variety of ways. File transfer protocols like gopher, FTP, and HTTP along with network and transport protocols like DECnet, SNMP, TCP, and others are among the more widely deployed connection protocols. Each of these technologies' role in the enterprise highlights the importance of the software plumbing. Without the core layers of the standard communications stack[2] and the management services provided by integration tools, the volume and scale of information transfer today would simply not be possible.

However, as with household plumbing, not all pipes and fittings are created equal.

The Coupling Question

Software coupling is the style and characteristics of how two different software components are connected. The easiest way to think about the kinds of connection techniques for integrating systems is to divide technologies into two categories:

- Tightly coupled plumbing
- Loosely coupled plumbing

This breakdown makes it clear that—despite the complexity and rhetoric in the application integration industry—there are not really many significant differences among EAI alternatives.

Tightly coupled integration systems use rigid control mechanisms that require each system to know specific details about other systems, or about the middleware itself, for the system to effectively communicate. To use the integrated interfaces, control flows, process and work flow languages, vocabulary transformations, management and monitoring schemes, and other elements of a network community each participant must have a tight coupling with some aspect of the network.

Tight coupling is not inherently bad. However, some consequences of tight coupling drive up total cost of ownership and drive down return on investment inside large enterprises. These negative consequences usually only become apparent over time and at a large scale. Because technology must always keep up with changing business environments, tight coupling causes a myriad of expensive maintenance nightmares.

On the other hand, loosely coupled integration techniques, which have only recently emerged, form the future of most integration styles. Loose coupling is characterized by indirection—always using an intermediary step to get to a final destination—between networked systems. This indirection establishes connections without predefined or explicit knowledge of the details behind participant nodes or middleware. Loose coupling insulates the nodes on the network from change,

[2] For additional information about the OSI network model, see Chapter 6.

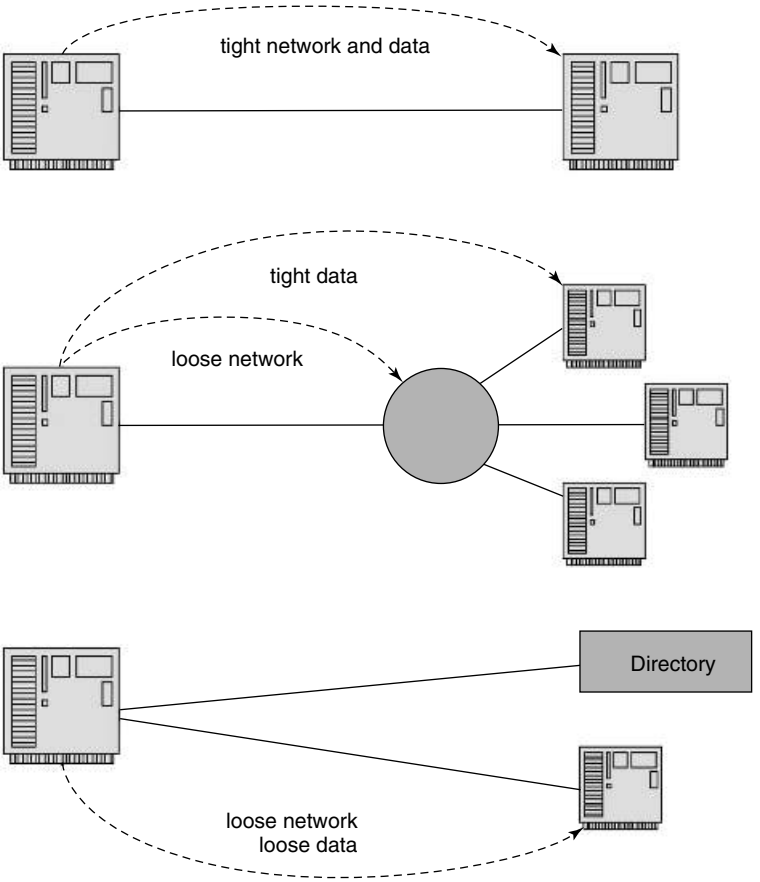

Figure 2.3 Tight-coupling and loose-coupling components

while delegating more of the "knowledge" about the system interfaces to the network itself.

Loose coupling is not inherently good. Early implementations of loose coupling suffered from poor performance, lax security, and inadequate error recovery. As these concerns are addressed, loosely coupled enterprise systems will overcome many of the persistent barriers presented by tight coupling.

Although a significant proportion of current integration systems could benefit from loosely coupled technology, let us not forget that a number of specialized application areas will continue to be best served by tightly coupled integration frameworks.

Business Process Plumbing

Advocates of business process management (BPM)—which is process-centric integration—believe that the critical challenge in enterprise integration can be solved

by better management of the transactional work flow among participating nodes on the network. On the surface, this is a compelling argument. The single greatest factor of change in the enterprise is business process, which cuts across organizational and system information silos. However, these advocates are missing one key point with far-reaching implications: To computers, business process models are just data.

Business process consists of instructions, or business rules, that define the timing of specific events, the treatment of messages, and what to do with messages during error flows. Execution of business process is delegated to software components that can reside in different physical locations on the network. For business process execution to take place, these units of code must interpret the messages, proprietary formats that can only be understood by the software that created the message to begin with.

Today's business process management solutions have had mixed results at best. Their tight coupling of interfaces and process rules create brittle environments that require constant attention to run effectively.

Because business process vendors build their tools around process data, rather than process semantics, the connections between enterprise systems are just as static as they always were.

Ironically, attempts to eke more value out of BPM approaches have led to the adoption of a loosely coupled middleware approach: service-oriented architectures (SOA). Even with an embedded SOA underneath the BPM solution, BPM will still be the plumbing. Fundamentally, the business of managing messages, work flow, and transactions in a message centric middleware solution are plumbing concerns. Because the focus of BPM is on the process, as opposed to the information contained in the message, it exists at a technology layer several abstractions removed from the content of the message itself.

Service-Oriented Plumbing

Service-oriented architectures (SOA) have emerged as the next great alternative to legacy middleware technologies and a key enabler for loose coupling. The SOA framework is the first widespread, viable solution for loosely coupled integration. SOA builds on the architectural patterns of component-based development and enables IT groups to build services and connect them on demand as they are needed. Unlike traditional plumbing, SOA gives users the ability to establish dynamic "pipes" that are initiated on demand with little overhead or prior knowledge about existence of the service. An infrastructure built on this approach can support integrations without a confusing collection of different protocols, interfaces, and conduits into various applications.

Nevertheless, the SOA paradigm is just plumbing. Because it is concerned with the physical nature of connections, interfaces, protocols, transport, and control mechanisms for the movement of messages, it is just plumbing.

Why Plumbing is Insufficient

Plumbing, or integration, is necessary but not sufficient. Without it, enterprise software communication would not be possible. With it alone, communication is inefficient, expensive, and lacking focus on what matters to the business—information.

In many cases, technologists responsible for specifying and deploying integration-based solutions do not fully understand how important the contents of the messages being transmitted is. If they did, they would spend more time on the underlying information architecture. Instead, today's most common solution is to just create a new XML schema that is adequate for describing the data that needs to go over the wire.

Unfortunately, this haphazard care for enterprise information has created an environment in which business's decision makers rarely have a complete view of the business and they pay dearly, in money and chaos, for each operational change they make.

ENTERPRISE CONTENT, NOT PLUMBING, IS KING

Okay, so it might have been 1999 when you last heard that "content is king," but here it is in reference to the content of enterprise messages—not Internet web pages. The invoices, repair schedules, financial information, patient data, sales figures, and customer data inside those messages are what comprise enterprise content. Isn't the whole point of integration to move content around so it can be used in new ways?

Crack open a Web Services SOAP envelope or an EAI message and you can learn something about the business. The XML documents and text files that fly around at the speed of light contain the records of importance for process flows and business-to-business exchanges. Even for process-centric applications, the representation of process in the middleware is the information that enables business logic to control the process flow events. The rules and structure that define a process are content that look and behave just like data.

Unfortunately, the amount of attention given to enterprise content, information, and data inside technology circles is minuscule compared to sexier subjects like Web Services and programming frameworks like. NET and J2EE. This is because it is difficult to understand and articulate the enterprise data that represents the foundation of every enterprise system.

Enterprise Information Data Structures

Present understanding of enterprise data structures is quite sophisticated. Not only are enterprise data different in internal binary formats, such as the difference between a text file and an object, but the information is also organized within a particular structure and representation. Basically, the continuum of data structure formats

ranges from very unstructured to very structured. This structural continuum is commonly broken into at least three parts:

- **Structured Data**—This is the most organized of this continuum. Typically, it will have definitions stored as metadata, such as type, length, table name, and constraints. Examples of structured data include relational databases, object models, and XML documents.
- **Semistructured Data**—This type of data is less clear; no specific delineations exist on where to draw its boundaries. For the most part, it is acceptable to think of semistructured data as data that are organized but not explicitly defined in a self-contained way. Traditional examples of semistructured data include positional text messages, such as EDI, tokenized, and comma-separated value (CSV) data files.
- **Unstructured Data**—This is simpler to understand: data that possesses no inherent organization intended to communicate its own meaning. Essentially it is like free-form text that is completely arbitrary in presentation and contains no metadata or structure that can be useful in describing its concept structures.

Fundamentally, the organization or structure of the data does not inherently make it more meaningful, only easier to process. The work of understanding what the data means, its semantics, is still a task that has to be undertaken with lots of effort in understanding the meaning behind the data.

Continuum of Knowledge

Understanding data content still requires human interaction. Without a programmer to encode the rules inside the software, the software is incapable of discerning meaning. One way to understand the reasons behind this is to consider the relationship between data and knowledge inside the continuum of knowledge.

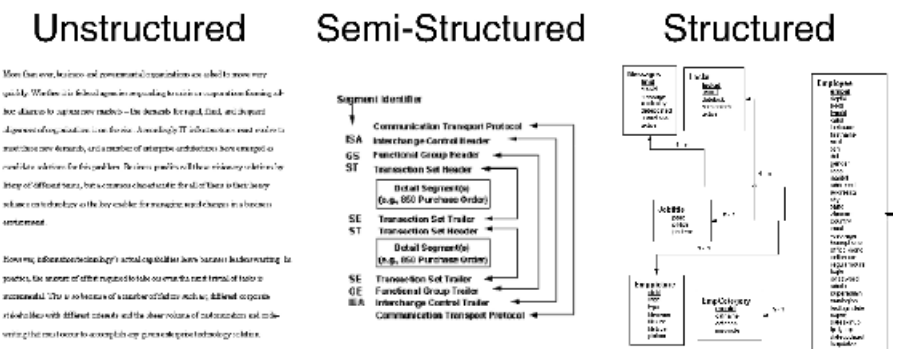

Figure 2.4 Structured data continuum

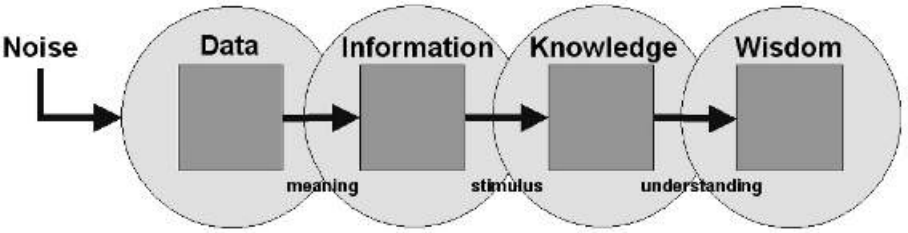

Figure 2.5 Hierarchy of knowledge

Although philosophers, scientists, and other academics do not have a comprehensive understanding of what knowledge actually constitutes, the above representation is commonly accepted as a basis for understanding the relationship between data, information, knowledge, and wisdom.[3] Consider the following example:

Data: "1234567.89"

Information: "Your bank balance has jumped 8087% to $1,234,567.89."

Knowledge: "Nobody owes me that much money."

Wisdom: "I'd better not spend it because I will surely get jailed."

Basically, information is data plus meaning. The discussion of what constitutes meaning is covered in Chapter 4, Foundations in Data Semantics, and is too broad to cover here.

The real challenge for enterprises today is to begin to build an infrastructure that enables enterprise information, of all formats, to be utilized freely at the right time and place. These capabilities will transition the industry away from commodity enterprise plumbing solutions and uncover new enterprise value enabled by seamless information interoperability.

FINAL THOUGHTS ON OVERCOMING INFORMATION INFRASTRUCTURE PROBLEMS

All problems of information interoperability can be solved, but brute force methods are still the most popular ways to effectively deal with these problems. However, brute force efforts result in inefficiencies and lost opportunities and have likely cost integration customers trillions. Results like these have soiled the reputation of the EAI category and prompted pundits like Nicholas Carr, as mentioned in Chapter 1, to write *Harvard Business Review* essays entitled "IT Doesn't Matter."

Carr's essential point is that IT doesn't enable companies to distinguish themselves in a meaningful way from their competitors. He concedes that IT is essential for competing but successfully argues that it is inconsequential for strategic advantage.

[3] *Experience Design*, Nathan Shedroff.

Strategic advantage is won when companies can build unique differentiators valuable to their customers. Building unique differentiators is precisely what happens when enterprise *data* evolve to enterprise *information*, because information interoperability enables businesses to better adopt and adjust their core capabilities. This kind of information capability transcends the argument that technology is a commodity precisely because information interoperability results are unique to an enterprise, thus enabling it to strengthen its existing differentiators—or pursue new ones.

Semantic Information Interoperability

The next great step forward in the computing industry will be its shift toward capabilities that can discern meaning inside digital systems. These new capabilities will enable enterprise content visibility and interoperability of unprecedented scales, which will in turn drive businesses' ability to establish strategic competitive advantages in both established and new ways.

Semantic interoperability emphasizes the importance of information inside enterprise networks and focuses on enabling content, data, and information to interoperate with software systems outside of their origin. Information's meaning is the crucial enabler that allows software to interpret the appropriate context, structure, and format in which the information should reside at any given moment and inside any given system. This information ubiquity is the beginning phase of a truly information-driven organization.

Integration	*Interoperability*
• Participant systems are assimilated into a larger whole	• Participant systems remain autonomous and independent
• Systems must conform to a specific way of doing things	• Systems may share information without strict standards conformance
• Connections (physical and logical) are brittle	• Connections (physical and logical) are loosely coupled
• Rules are programmed in custom code, functions, or scripts	• Rules are modeled in schemas, domain models, and mappings
• Standard data vocabularies are encouraged	• Local data vocabularies are encouraged

The technology components to enable interoperability of this kind already exist. Universities, companies, and standards bodies throughout the world have been working on semantic technologies for many decades. Widespread adoption of these technologies is now possible because of the rising maturity of these approaches, increased R&D funding, and poor performance of more popular traditional techniques.