EAI: concepts and trends

N. Aloia, C. Concordia, G. Weinbach*
CNUCE – CNR Via G. Moruzzi, 1 56100 Pisa (Italy)
* Objet Direct 38 Rue Copernic, 75116 PARIS

E-mail: cesare.concordia@cnuce.cnr.it

Abstract

The ever-increasing growth and penetration of Application Integration technologies and market is bringing about significant changes in how Enterprise Information Systems are organized and managed. Moreover, the increasing reliability of tools and techniques stemming from the "Internet revolution" is rendering it ever more convenient to base the development of new applications on this technology, especially because of the great advantages it affords in terms of scalability and distribution.
These two principles lead to development of computer systems highly scalable and whose different components should be integrated allowing information exchanges between them. The main technique adopted to achieve this aim is the so-called Enterprise Application Integration (EAI). The present paper deals with the fundamental characteristics and current trends of Application Integration and related technologies and illustrates the problems that need to be tackled in development and management of an integrated and flexible enterprise information system.

Introduction: Enterprise Information Systems and urbanization

An information system (IS) is a set of organized procedures, manual and automated, providing the means to run an organization. According with Kenneth C. and Jane P. Laudon [LA] an information system is a set of interrelated components that collect (or retrieve), process, stores, and distribute information to support decision-making, co-ordination, and control, in an organization. The computer system is the component of the information system, which handles the automatic processing of information. Over years information systems have evolved mainly because of the new sense enterprises give to "information" per se. Enterprises (and in general organizations) want to use all information collected to add value to their product and services especially to be competitive with Web revolution.
Traditional information systems don't explicitly accomplished this task, they are used for facilitating the realization, communication and processing of the information, and, often, are constrained by the organizational structure, as well as the hardware and software architectures adopted. To respond to new exigencies, more flexible systems are needed. The computer systems cannot be monolithic; the different components should interact in a clever way to accomplish each function needed to reach the requested equi-finality [SC].
The main solutions emerged over last years to align enterprises information needs and computer systems are the openness and the "urbanization" of the information systems. These two principles lead to development of computer systems highly scalable and whose different components are integrated allowing information exchanges between them. The main technique adopted to achieve this aim is setting "middleware" between different

applications, the scope of this software is to intelligently translate and route data between applications. This is called Enterprise Application Integration (EAI).

It is rarely the case that a commercially available application offering will address totally organization's requirements, and no cross-function application suite can address all of the requirements found in large structures. Therefore is true that almost all systems have some integration needs.

Common sense argues that different kinds of integration problems call for different integration approaches.

There is no lack of information on EAI. A simple search on the Web will bring many documents on different technologies and tools. Not many documents, however, deal with general concepts on design and implementation. In this paper are collected and ordered information on these topics.

Application Integration Patterns

Introducing patterns in integration technology allow analysts and designers to move from mere concepts to concrete design and architecture fields. Moreover patterns can provide a basis to address the current landscape of Enterprise Application Integration. In practical terms, patterns in software development could be considered as recurring actions, techniques or logical organization. Dealing with the domain of application integration, there are several categories of patterns that should be assessed as part of the analysis and design phase of the project. IBM offers a set of reusable assets for help developers in building applications with integration features [IB], Grosh introduced a taxonomy based on "behavioral patterns for EAI" [GR], Yee's patterns are classified according to their role in the architecture of the system (User, Integration, Adapter)[YE] etc.

To achieve our purposes however we'll use a more pragmatic approach [RU]. In this taxonomy, the word "pattern" is not intended to mean "patterns of design or code", or "patterns of operational processes for an integration project", instead, a "pattern" defines a type of integration problem, a solution, as well as parameters applied for e-Business Integration. Structurally, these patterns are built considering user requirements, enterprise organization, legacy systems and data. Integration patterns identify patterns of how integration solutions are designed. The following taxonomy covers most of the common integration scenarios implemented today:

- Monolithic applications: self-contained systems, usually built using traditional programming methods. In these systems different application are tightly bound each other.
- Data consistency: applications and systems that coordinate their behavior by exchanging data via shared data stores. That means: there is a common data meta-model and different applications must translate data in their native model.
- Multi-step: applications that coordinate their behavior through the asynchronous exchange of transactions exemplify the multi-step integration pattern. An intermediate broker (for example a Message Oriented Middleware, MOM) transforms and routes the right data to the right applications, according to the input requirements of each target. Some authors distinguish between Single-Step Application Integration (SSAI) and Multi-Step Application Integration (MSAI)

the latter addresses many-to-many integration, which SSAI cannot, by providing the so-called “sequential logical processing”.

- Composite applications: systems that implement their behavior through method calls to external application services adhere to the composite application pattern. The idea behind the composite application pattern is to create “meta-applications”, whose capabilities are a composite of encapsulated services and integration logic. The services themselves may be implemented with independent components, wrapped applications, or even combinations of several applications or components.
- Autonomous distributed: the autonomous distributed integration pattern is characterized by the combination of application services to produce a composite behavior, as in the composite application pattern, but it goes a step further by exploiting frameworks for service discovery and dynamic method binding, enabling integration to occur at runtime (e.g. web services).

Currently data consistency is yielding to multi-step as the most widely used integration pattern, but probably there will always be problems for which data consistency is the best approach. And despite the current hype around service-oriented architectures and web services, the composite application and autonomous distributed patterns are not replacements for the data consistency and multi-step approaches.

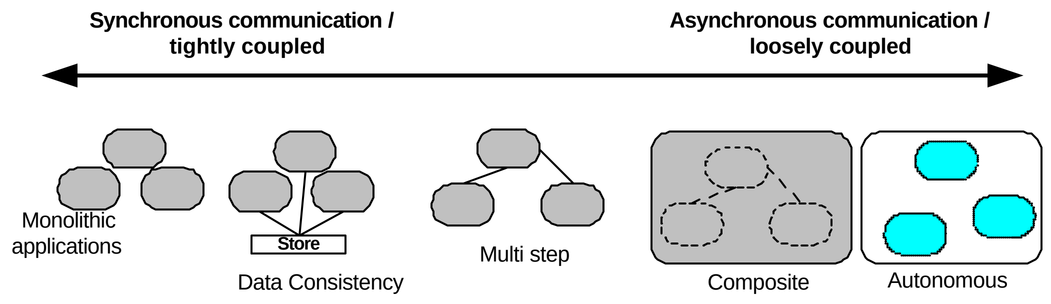
Synchronous communication / tightly coupled
Asynchronous communication / loosely coupled
Monolithic applications
Store
Data Consistency
Multi step
Composite
Autonomous

Figure 1 Integration patterns

Integrating applications

There are two main views on integration: from inside the enterprise (A2A integration or Enterprise Application Integration (EAI)), and view from the outside (B2B integration (B2Bi)). For most organizations the main issue is to integrate consistently intra-organization and inter-organization aspects of their activities. In both views (Fig. 2): integration end-points are applications services and data, end-points are accessed either directly or via middleware using resource-specific adapters. The core of both integration views are the services that determine how native interfaces combine resource, inputs and integration logic to produce the desired behavior for a given step, that are: routing, transformation, messaging and transaction services. How core services are implemented strongly depends from the integration patterns adopted. Most organizations have integration problems that fit more than one pattern, including some with hybrid requirements; there is an increasing need for integration solutions that offer a complete functional "stack".

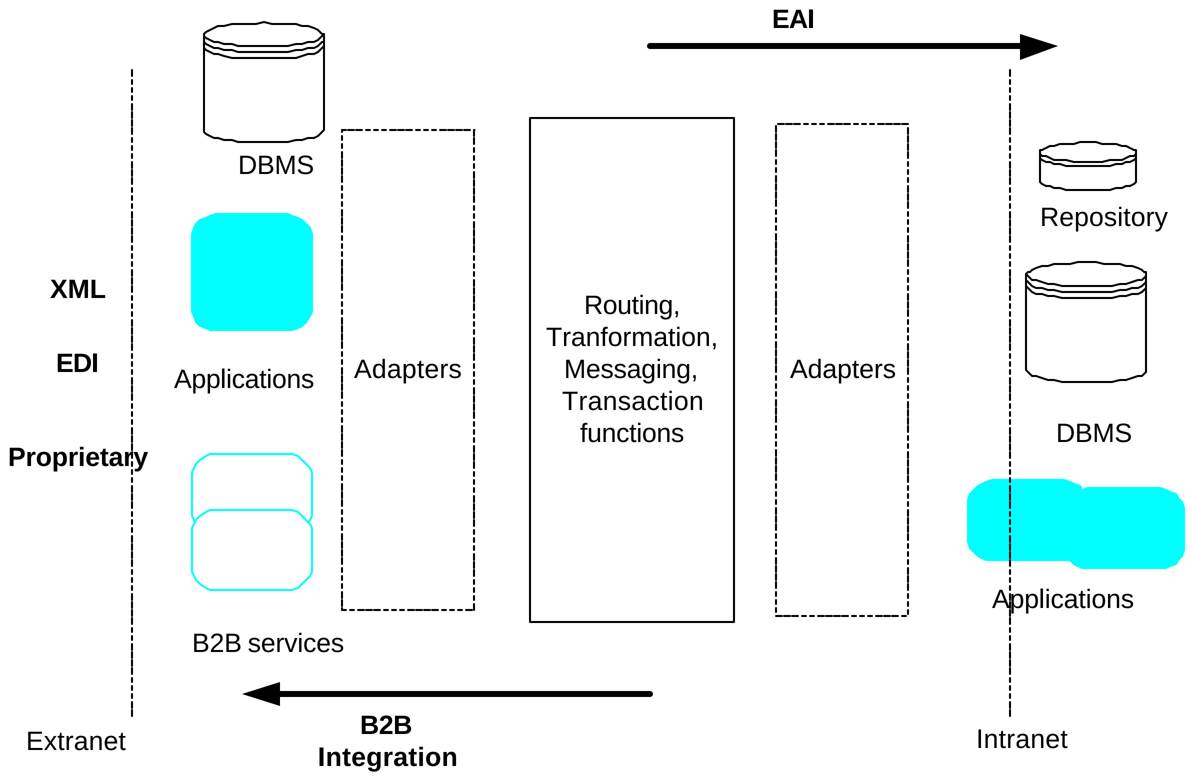

Figure 2 B2Bi and EAI

Integration Technology: state of art

As previously stated, data consistency and multi-step integration patterns are currently dominating integration practice. The "messaging capabilities" are now a commodity, and products like WebSphere MQ (formerly MQ Series) and Java Message Service (JMS) are today considered as de facto standards. Therefore innovation is pursued elsewhere, application integration solution providers are more and more offering tools whose aim is to merge B2B integration with A2A integration. As an example, products like WebSphere Process Manager (IBM) or TIB/InConcert (TIBCO) allowing users to define and manage business processes and workflow, are completely integrated into IBM's and TIBCO's EAI suite offers. That means intra-enterprise and extra-enterprise integration strategy could be implemented using a unique approach, therefore avoiding duplication of adapter, transformation, management, and other elements. Another element that strongly influenced integration's evolution is the Internet. The opportunity to use this environment for implementing information systems offers considerable benefits, while at the same time posing new problems. Internet-based, hosted integration services are emerging as a cost-effective way for small and medium companies to participate in integration. Smaller players can participate in value chains with much larger partners, for instance iPlanet (Sun) or WebLogic Integration (BEA) offers many facilities to implement an Internet scale application integration approach, but also other solution providers furnish native software modules to include Internet advantages to their products.

Integration strategy

Designing an integration technology's strategy is not easy. Aside from knowing the specific organization's needs the designer should have a clear point of view about how integration should work. In our opinion the main point here is to enable users to deal separately with functional (business) and implementation (computer system) requirements, without sacrificing processes completeness or services performance. The aim is to build an information systems flexible and scalable that easily adapts to the eventually changed organization exigencies, granting at the same time coherence and consistency of services and functionality. To achieve this aim the complexity of integration problems must be tackled at three levels:

- Process view: organization activities must be analyzed and all so called strategic and organizational invariant [JE] processes must be singled out. Invariant processes in an organization are processes that represent the main organization's tasks and that are supposed to be stable "enough" over time.
- Information system view: this view includes all tasks that can be automated. Enterprise's semantic of each task must be clearly described using a set of rules or standards and tasks' relationships must be represented in explicit form. The main goal in this view is to remove ambiguity and redundancy defining responsibilities and roles. An accurate settlement of this view offers two important advantages: allows developer and system architects to refers to structured and coherent information in building computer system and, above all, make easy to maintain information and data compatibility in case of system or interfaces evolution.
- Computer system views: the implementation of the different tasks singled out in the information system views. The aim is to build a system where each task, defined in information system view, could be implemented using an appropriate tool independently from its compatibility with others products. Considering this, the correct EAI solution must be chosen to lay the foundation of the computer system.

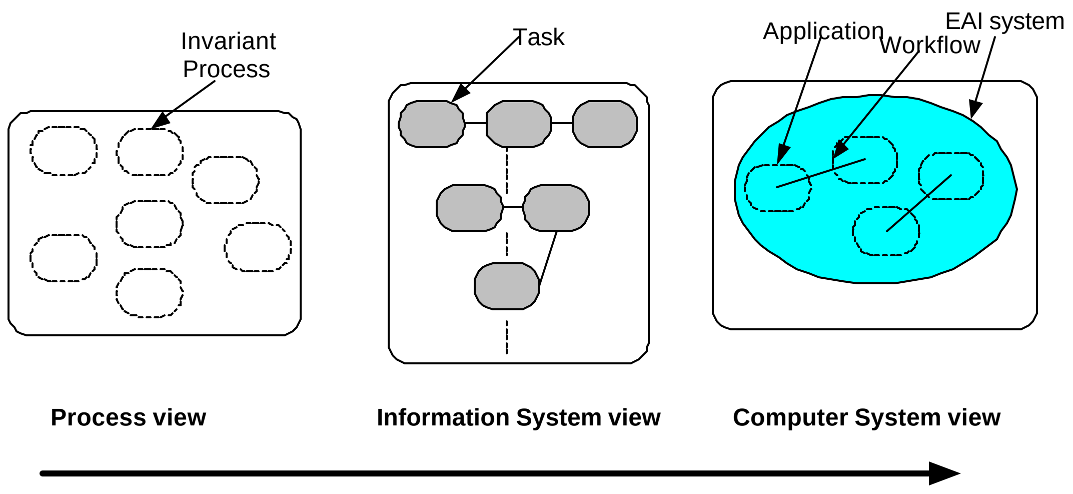

Figure 3 Strategies

Separating IS specifications and system implementation enables integration deliverables reusability. This principle allows reducing the effort of building and maintaining integration solutions in multiple runtime environments. The quality (completeness and non ambiguity) of the IS process modeling, combined with the use of integration tools (such as EAI) might allow automatic generation of some part of the computer system view (i.e. transformation, routing and workflow information), providing a high level of maintainability and global coherence.

There are some principles that lead each phase of view's definition, choosing an integration solution:

- Maintain environment independence: customers should be able to change completely or partly the information systems choices he made to adequate IS to organization evolving requirements. Open technical solution allowing use of de facto standard messaging services, Internet protocols, multi platform programming languages (e.g. Java) and standard interface to storage mechanisms (DBMSs and file systems) are preferred to achieve this goal. This principle ensures continued system viability in the face of change and enables maximum advantage to be realized from existing IT infrastructure.
- Manage functionality through abstraction: abstraction is used to allow users to deal with different application and resources at functional level. Dealing with resources through abstraction simplifies maintenance and make it easy to specify systems that can be implemented in different environments with minimum changes.
- Model the system completely and deploy from the model: that is Model-Driven Integration, in which systems are specified and maintained using diagrams, abstractions, and rules, has received great attention in the market[OM]. Building solution conformed to this principle simplifies system configuration, maintenance and management.

- Use a single set of tools and services to support EAI and B2Bi: in theory a common set of runtime services and design-time tools can be used to solve several integration problem. The goal is to build an infrastructure that supports both EAI and B2Bi needs. This principle can be implemented only if services are combined with the right set of abstractions and architectural layering.

Integration trends

Both integration technology and the integration market are evolving rapidly, therefore in this paragraph we try to do a snapshot of current trends.

Integration is today *pervasive*. As commercial offerings mature and organizations gain experience with start up projects, integration technology is being widely applied. All the patterns defined above are present in the current integration scene, reflecting the increasing scope and impact of integration practices. Integration features are being embedded in application server platforms, messaging systems, portal servers, and other infrastructure technologies, as well as commercial application products. However most of these offerings provide limited integration services, offering little value beyond their intended problem spaces. Organizations in specific industry sectors have particular integration needs that are not supported by most off-the-shelf integration solutions. Purpose-built solutions may address specific needs, but they offer limited support for broader enterprise integration requirements. In larger organizations, this can lead to a *multi-vendor integration problem*. However today, thanks mainly to cross platform technology, a middle road is emerging, in which divergent industry-specific offerings are implemented on common architectural foundations. In early implementations, the integration emphasis was on applications interaction. The missing components were support for manual process steps and a common process view that could provide a basis for modeling requirements at a business level, as well as for tracking and managing process status. New solutions are emerging that support the ability to capture and model process integration requirements at the business level, including long-duration processes that span organizational and system boundaries, like collaborative product design. But they also realize faster implementation of interfaces to back office applications, Web applications, enterprise data stores, trading partner systems, and other IT resources, and the ability to monitor and manage the operational result using business-level process models.

Web services

The "web services" framework is emerging as an important "dynamical" integration model. This vision of dynamic integration is enabled by a standards-based framework that supports web services discovery (Universal Description, Discovery, and Integration - UDDI), service formulation and instantiation (Web Services Description Language - WSDL), and service invocation (Simple Object Access Protocol — SOAP). These standards can be used together, or separately, to support different levels of integration automation. Each service provider implements the web services themselves as purpose-built applications, agents, or wrappers around existing applications. Web services are potentially important in a number of areas, including enterprise portals and online exchanges. Enterprise portals could exploit web services not only for publishing and implementing services for internal and external applications, but also for personalization of services based on user identity, history, and runtime circumstances. Private and public exchanges might use web services to exploit these same opportunities, but could also use them to dynamically bind buyers and sellers, according to exchange-specific heuristics. The foundation for implementing the web services model is mostly in place, but few real-world implementations exist today. Like the other application patterns we've examined, the autonomous distributed pattern, based on the web services framework, will seek its own level beside other integration patterns, as business models evolve to take advantage of this new approach.

Integration technology and Information Technology

As more organizations discover the advantages of implementing new systems by integrating existing applications and services, integration technology is becoming a fixture of modern IT architecture. As the scope of integration projects increases from few connected systems to hundreds or thousands, integration is becoming mission-critical. This trend imposes new requirements on integration technology, aimed at ensuring the availability, reliability, and performance of integrated systems. The main architectural challenges posed by these requirements are to avoid single points of failure, ensuring the integrity of transactional processes, and managing dynamic workloads. During the past several years, integration brokers have moved beyond simple client/server architectures to support a "federated static" approach. In the federated static model, two or more servers are distributed in a network, and applications are bound to specific server instances. Federated static implementations generally provide static workload management and some transaction integrity. Because they still present a single point of failure for some portion of the integration workload, they don't adequately address the availability requirement. The next evolutionary step in integration broker architectures is the "federated dynamic" approach [LI]. In the federated dynamic model, two or more servers implement a distributed integration service in which applications are not bound to a specific server instance. No single point of failure exists, because application requests can be redirected in the case of individual server failures. The integrity of distributed transactions is maintained across server instances through transparent sharing of transaction and state information. Dynamic workloads are allocated to server pools according to availability, capacity, and resource affinity. An initial step toward federated dynamic architecture occurs in the form of cluster-based distribution. Clusters provide a convenient basis for implementing dynamic fail-over and load balancing on a single platform. Ultimately, however, achieving federated dynamic behavior across multiple platforms and locations will require that integration brokers implement the federated dynamic model natively.

Conclusions

There is no lack of information on EAI. A simple search on the Web will bring many documents on different technologies and tools. Few documents, however, deal with general concepts on design and implementation. In this paper, we indicate what we believe to be the key concepts, the strategies and trends of application integration technologies; keeping in mind that the dynamic nature of integration technology (and market) can make some of the technologies and solution described above quickly outdated.

References

[IB] http://www.ibm.com/developerworks/patterns/

[JE] "Urbanisation du business et des SI", G. Jean, Hermes 2000

[LA] "Management Information Systems" Kenneth C. and Jane P. Laudon, 4th ed., Prentice Hall, 1996

[LI] For complete definition and details see the interesting article: " Making EAI Scale" by David S. Linthicum http://www.intelligenteai.com/feature/010416/linthicum.shtml

[OM] The Model Driven Architecture (MDA) is the new strategy of the OMG: http://www.omg.org/mda/executive_overview.htm

[RU] William Rue has established this taxonomy. Other taxonomy can be found in literature. An interesting book on this topic seems to be "Patterns for e-business: A Strategy for Reuse" IBM Press 55 US$, but I only saw a sample chapter of this book

[SC]."Management Systems", P. Schoderbek, Wiley 1977

[YE] "Making EAI Work in the Real World: EAI Patterns" A. Yee, http://eai.ebizq.net/enterprise_integration/yee_3.html

[GR] "Data Imperative: patterns in EAI behavior" eAI Journal, September 2001 pp22-28