Integrating Enterprise Applications: Backgrounder

by S. Radhakrishnan

April 25th, 2005

Introduction

Enterprise Application Integration (EAI) is a serious consideration for many Enterprises. The reach of EAI is quite vast – from partner purchase order exchange to Enterprise Resource Planning (ERP) and supply chain integration, the complexity and criticality of the solutions vary. Specialized EAI products have been around for more than a decade and they have shown tremendous growth and adaptability to changing needs. Nevertheless, integration, so far, has been only a tactical solution – a solution to integrate various applications and channels of communication which, in most of the cases, had been built in silos. This type of integration is needed for quick wins and a faster time to market, for an ever changing business ecosystem. Unfortunately, while integration efforts are on in an enterprise, newer systems are built with the assumption (and hope) that they can be integrated using the "my enterprise standard EAI product".

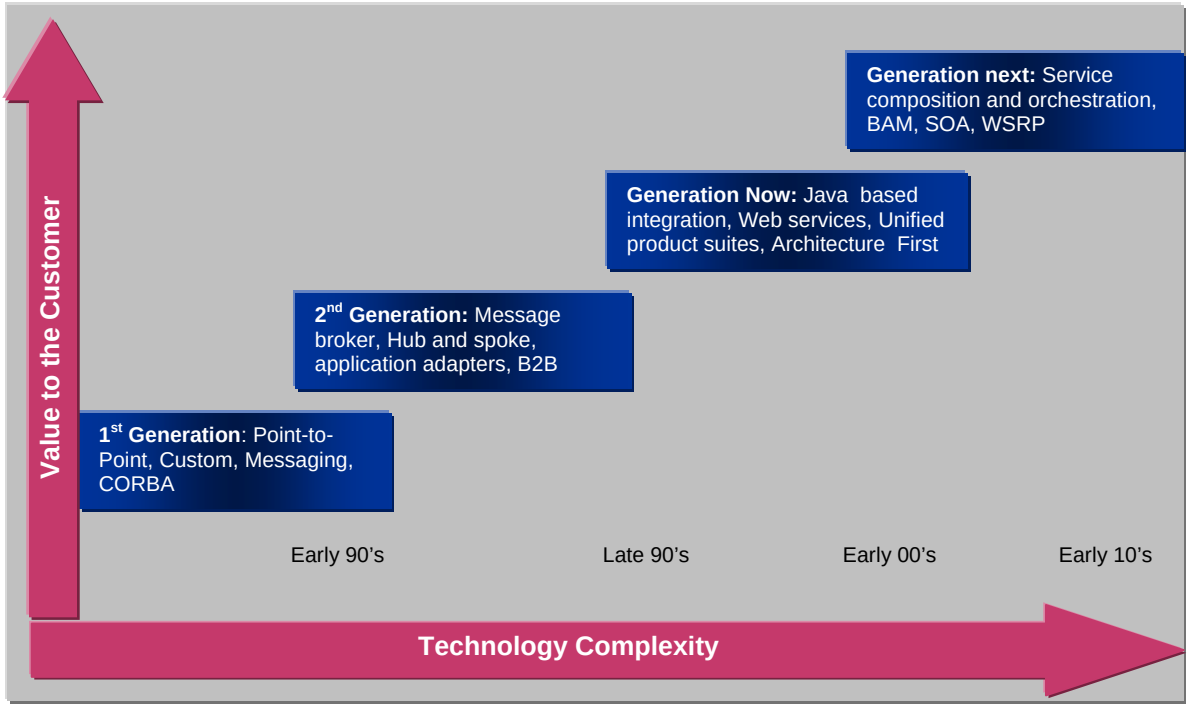
This cycle of moving-and-catching, if based on a clear vision of integration strategy at the enterprise level, creates a well orchestrated IT ecosystem; if not, it creates more integration points for the future: an antithesis of EAI. But to have a well orchestration of systems, integration has to take a wider meaning: Integration of a system across business processes, channels of contact and communications, and data. In a truly complex enterprise this type of integration is non-trivial, and cannot be solved just by adopting a product or technology but can be only addressed by an enterprise-wide strategy and architecture standard.

This paper presents the integration landscape over the last decade and captures the major changes that have occurred so far and also discusses how the changes in technology and service orientation transforms this landscape.

A Brief History of EAI

The necessity of the application integration, of course, is fuelled by the Internet economy, where real time data and application synchronization are of paramount importance. The EAI as it is known today has crossed various stages of evolution. The following diagram captures the various generations of EAI, to be used as a reference point for further discussion.

Figure 1. Integration history



EAI has evolved from simple custom interfaces and point-to-point integration to many-to-many message publication and subscription. Products' focus and differentiators, which were once in specific areas likes Business-to-Business (B2B) integration or simple within-the-enterprise integration (referred to as 'Internal EAI' in this article), offer a wide variety of capabilities and even include application server and data management solutions as part of the product portfolio. This proliferation is both beneficial and detrimental to the enterprise. While they offer a wide variety of choices, without an enterprise level standard the proliferation will create confusion and spaghetti integration – the problem which the EAI is supposed to solve.

The following sections capture the hallmarks of these stages or generations of integration, and elaborate on the maturity of integration technology in addressing specific problems.

Generation 1: Age of Point-to-Point

In this period of early Integration, where the term EAI was barely known, integration products and techniques aimed to provide APIs and interfaces between systems, either by means of custom Remote Procedure Calls (RPC) or by means of distributed computing standards like CORBA*, or by means of messaging products. RPC and CORBA achieve the same goal differently. CORBA being much richer and object oriented when compared to a simple RPC, viz. integrating systems by means of request and response. They do not allow for automatic processing of messages at a later time than when they were issued, though these features can be built. Messaging technologies on the other hand allowed for storing

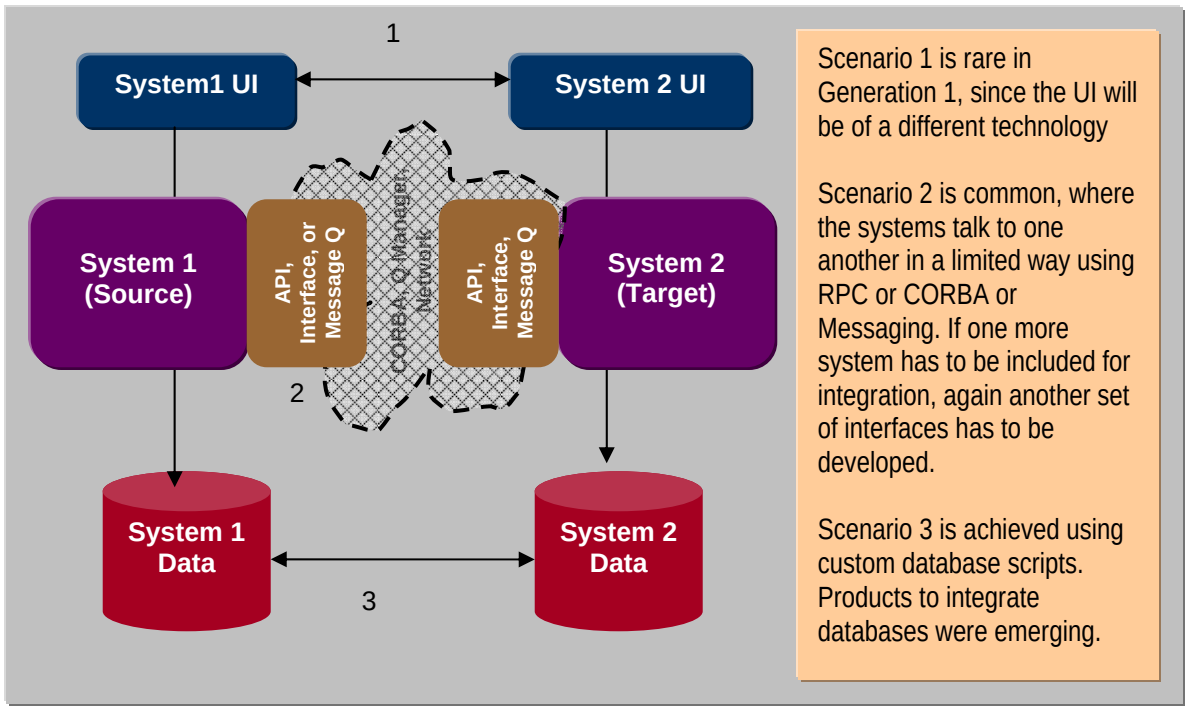
and forwarding the messages that can be processed at a later time. Further, while using messaging it is possible to decouple the source and target systems - for example, the source system can be a custom C++ system and the target a mainframe, both having separate run times. In reality, to achieve integration at the enterprise level both the techniques (messaging and RPC) are used. For inter enterprise integration, Electronic Data Interchange (EDI) was the only means.

The key characteristics of the first generation integration are that most of the integration is custom built using various technologies (the “integration brokers” only started emerging.) Integration is predominantly aimed for point-to-point – this implies that the interfaces are to be developed for each of the participating system pairs (source-target). Integration data flows are to be developed for each of the integration points and there was minimal re-use of code. In this scenario the semantics (format) of the data flow between various systems can proliferate – many variations of the same message structure are quite common. This means that coding, maintaining, and versioning them is a nightmare, particularly for large enterprises. In addition, each of the message types has associated transformation rules with them, which are also to be coded and maintained. The transformation programs (or “sub-routines”) are also proliferated.

For large enterprises CORBA became the standard middleware for the integration projects[1] and for a while CORBA seemed to be the mainstream of integration, specifically mainframe and other legacy systems integrations. However, CORBA’s impact in integration diminished particularly with the advent of proprietary integration brokers and Java*. Nevertheless, many of these integration products and application servers internally used CORBA for communication, across various software components.

The following diagram shows the different levels of integration in the first generation.

Figure 2. First generation integration



In summary, the major achievements of the first generation integration are as follows:

1. Integrating legacy application with other “open” systems” (Unix* or Windows* based)
2. Large scale integration projects were achieved using CORBA

Following are the major setbacks, which are obvious:

1. Too much custom coding
2. Absence of a message broker leading to inflexibility

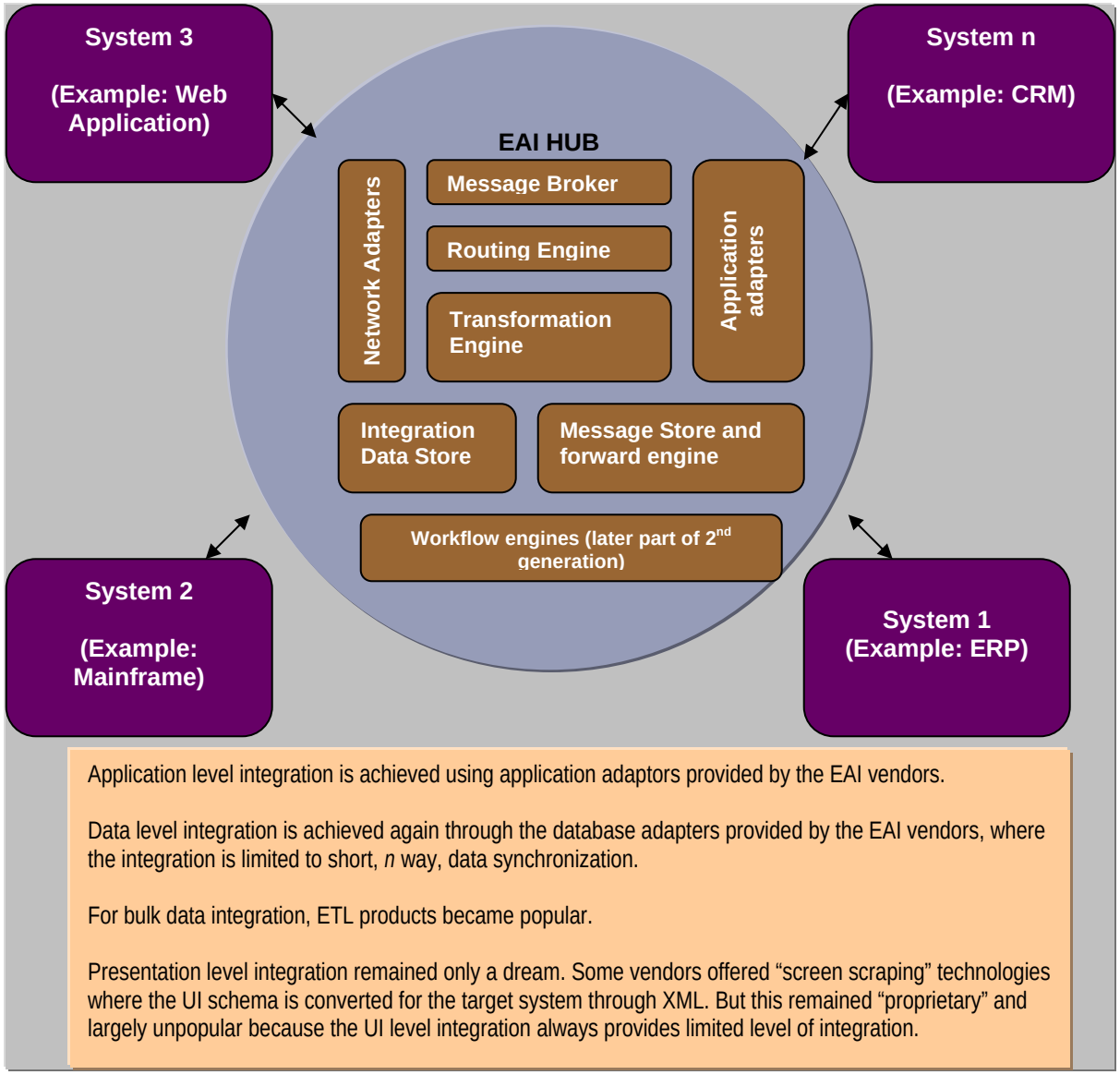
Generation 2: Age of Many-to-Many

In this period, integration products brought in the concept of message brokers and paved a way for *n* X *n* application integration to achieve “Hub and Spoke” integration. Product vendors differed in the way the connectivity to the source and target systems are achieved and used the proprietary way of communication mechanisms. They introduced the concepts of “adapters”, which are the plug-ins for the application or the network, as the case may be. The message broker is responsible for translation of the messages between systems (called “documents” in the EAI terminology) and routing them to the appropriate target applications. The routing rules and mapping rules are defined in the broker, using a visual application editor. This eased a lot of custom development effort and coupled with the need of a faster time to market in the Internet economy, generation 2 products became wide-spread and popular.

The logic processing in the brokers to achieve custom transformation was done using proprietary scripting languages. Each of the product vendors provided their own scripting language. So EAI remained “pure play”, where one has to depend on the vendor’s product for integration, particularly so because the messaging protocols are proprietary as well.

By its nature, bulk data integration, had to be handled separately by the Extract Transform Load (ETL) products or by custom development. However, the second generation products provide good data synchronization capabilities where smaller amount of data changes in one system can be propagated to other systems in real time.

Figure 3. Second generation integration



Following are the major advantages provided by the second generation products:

1. Systems participating in the integration are shielded from one another. Hence, it was possible to develop a largely non-intrusive integration.
2. The integration logic (for example, transformation and mapping rules) can be re-used and shared among multiple projects.
3. The integration logic is modeled by visual editors with drag and drop features, which accelerated the delivery of EAI projects.
4. Real time data synchronization was achieved. In the first generation, synchronous integration at data layer level could not be achieved without touching the business layer or the UI layer.

The major disadvantages of these solutions are as follows:

1. Proprietary language for coding the logic resulting in vendor lock-in – This changed somewhat in the later part of this period as integration software vendors had to adopt Java and XML because of the popularity of these languages.
2. Another less known effect: The EAI product became the center of all applications in an enterprise. The future of an enterprise integration depended on the strength and growth of the EAI vendor.
3. Not all vendors' adopted the organic method of evolving the integration software. In many instances the integration brokers were purchased (because of mergers and acquisitions), and integrated with in-house messaging systems. This led to a period of confusion about the logical placement, duplication of infrastructure, and frequent product upgrades.

The second generation EAI helped to a great extent in launching and consolidating e-Commerce transactions over the Internet. For example, in a booking process in an online store, inventory of the product can be verified with the backend ERP system using EAI, and then the order can be placed by the system. The placed order information can be propagated to a warehouse for shipping and once shipping happens from the warehouse the customer can be informed of the same using EAI with a CRM application. In effect, the second generation products tended to bridge the gap between what is outside the enterprise and what is inside.

Generation Now – Age of Consolidation and Proliferation

Architecture First

The present generation of integration has the following four faces:

1. Technology maturity and unified application and integration: Many of the popular EAI vendors have consolidated the products and offer Portal, Integration Server, and Workflow Management, as a product suite. Thus, it makes it simpler for the enterprise to invest on a single vendor's technologies.
2. Java has entered into the mainstream of integration. The application server vendors have ventured into integration space and offer a unified product suite for server application, integration, and workflow. Relatively, these offerings are new, but they show great potential. For enterprises embarking on investments on integration, this provides one more set of choices, and hence decision making is complicated. More importantly these things got integrated into Java 2 Platform, Enterprise Edition (J2EE)* standard and hence vendor lock-in is reduced to a great extent. With the fear of application server vendors getting into EAI space, proprietary EAI vendors turned J2EE compliant. Further, messaging got standardized through Java Message Service (JMS)*.
3. The emergence of Service Oriented Architecture (SOA) and Web Services has created re-thinking about integration among decision makers like enterprise architects. The usage of Web services versus pure play integration products are seriously debated before an integration project is started, even in cases of enterprises that have invested in pure play integration products. Combined with the two points above, the integration choices become more difficult. This is akin to "too much of a good thing".
4. B2B integration got a fillip from the Web services technology. Inter-enterprise synchronous integration started. This has been traditionally through EDI.
5. The strengthening of partner integration with enterprises: In the extended enterprise paradigm, business partners interact with the enterprise more and more. Depending on the techniques available with both enterprise and business partners, the exchange can happen either through the secure Electronic Data Interchange—Internet Integration (EDIINT), Web services, or B2B extensions offered by integration vendors. In addition, many enterprises have already deployed Partner Portals (extranets). Thus, partner integration becomes technologically proliferated and it is not uncommon to see multiple competing technologies deployed to achieve the same goal.

For enterprises looking for simple and direct solutions, the above choices mean proliferation of technology and confusion. It is less obvious that, the choices have a strong dependency

on architecture style rather than mere technology options. This has to be viewed in the backdrop of the realization of the importance of enterprise architecture strategies and principles. It is clear that enterprises do not want to invest on something that has to be thrown away in a few years time: what is invested on should be justifiable, should provide short and long term returns, and should support future initiatives. Thus, architecture principles and guidelines are increasingly formulated at the enterprise level rather than at the project level. This means that integration, which is within the purview of the architecture, is increasingly seen to comply with the enterprise's architecture principles and standards. This is in contrast with the second generation integration, which was always considered only a "black box" in the enterprise architecture. Projects were executed keeping in mind the short term benefits based on the techniques that the products used and not necessarily based on the "big" picture. This is perfectly well suited for small number of integrations to serve tactical purposes.

Strategically this is a handicap, since re-usability, extensibility, and ease of maintenance are critical in the long run. For example, in many enterprises the business logic is complex and may involve data from multiple systems before arriving at an outcome. Thus, if one codes this logic as part of the proprietary integration solution, it is extremely difficult to change, extend or re-use. Therefore, the impact of placing the logic within proprietary integration (in this case the integration hosts business logic) vs. placing the logic within one of the participating systems (in this case integration is used only to fetch data) vs. placing the logic as a service (either / or the previous two) has to be analyzed for various factors including performance before embarking on a solution. Clearly this is an enterprise architecture issue rather than an integration issue.

The result is that strategists and architects realize that integration issues should be tackled keeping in mind the enterprise issues. Thus, one can see the formulation of Integration Competency Centers (ICC) at the enterprise level to handle cross functional, cross projects, integration issues, and conflict resolution.

Generation Now: Technologies

The following sections highlight some of the various integration technologies, which identify with the present generation of EAI.

EDIINT

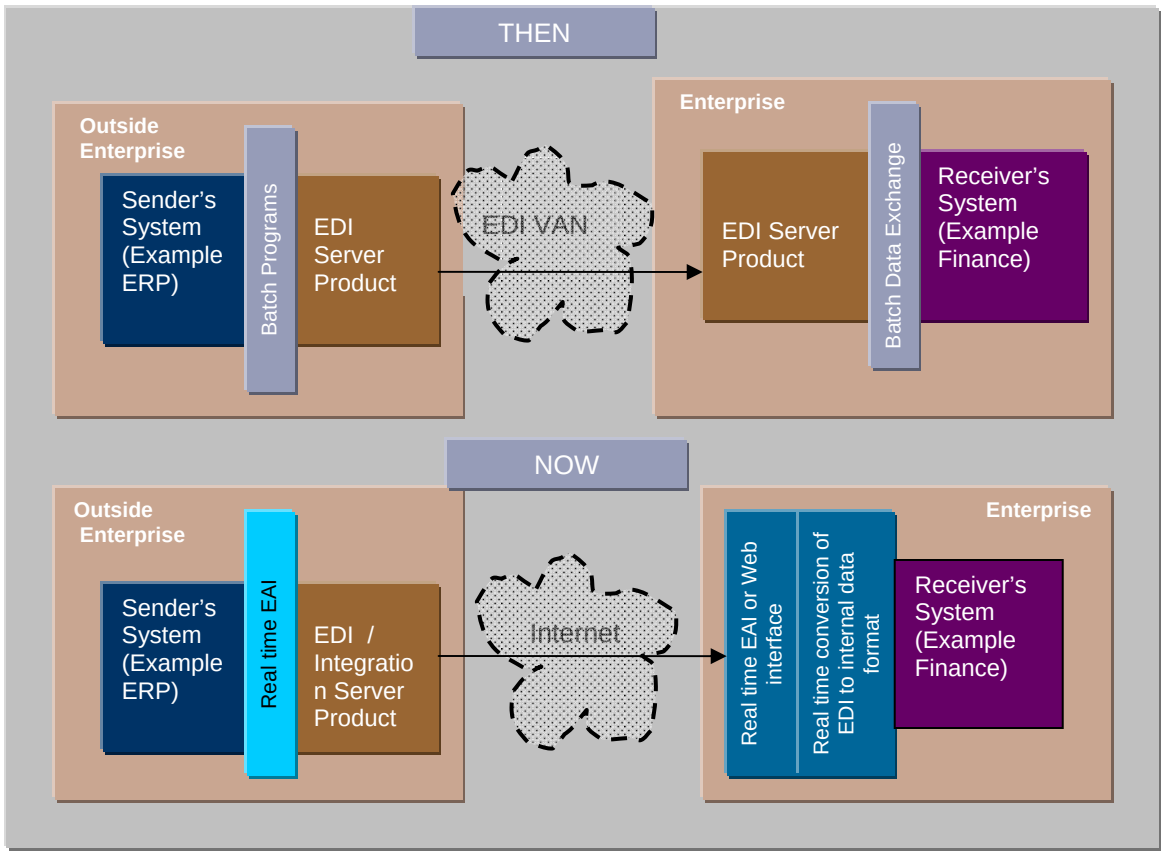
External integration using Internet has become an accepted norm. Proprietary EDI Value Added Networks (VAN) are complemented by Internet protocols of communication. Internet Engineering Task Force (IETF)* has formulated the standards of exchanging electronic documents over the Internet. Many industries who want to retain the investments done on EDI but at the same time want to reduce costs of VAN, have adopted EDI as the message format and Internet as the communication medium. The Applicability Statement (AS) 2* specifications formed by the IETF committee received good industry support and is supported by almost all the leading vendors[2]. This implies that small and medium enterprises could easily take EDI over the Internet because of the low entry cost.

Traditional EDI transactions required EDI products, which handle EDI transmissions and sometimes format conversions, and most of the transactions happen in a batch mode. Since

companies have to pay for the number and size of EDI transactions, it was profitable to combine messages and send them. Further, the expectation was always that it will take some time, sometimes days or weeks, to process these documents and take action. All this changed in the Internet economy: Data processing need not take days or weeks and one doesn't have to pay much to send documents over the Internet.

The following diagram shows the difference: It is clear that the real time data exchange is possible with the combination of EAI, EDI, and Internet.

Figure 4. B2B integration turns real time



Java and Integration

The impact of Java and Web services' on the middleware platform has created a new breed of integration servers. This new breed has evolved from the Java application server: while Java application servers' markets were saturating, application server vendors ventured into the integration arena and developed pure Java-based Application Platform Suite* (APS, a term frequently used by technology analysts). Today, many of the leading Java middleware vendors offer Portal Application Server and Integration Server, including Workflow Management as a package. Of these suites, Application servers follow J2EE standards. Portals, Integration, and Workflow, though in Java, are largely proprietary. Some elements of standardizations, for example using Portlets standards, or Process Definition for Java (PD4J)*, a workflow definition for Java are on the way. However, the core integration (apart

from J2EE Connector Architecture (JCA)*) will remain proprietary — Java's proven usage in an enterprise's mission critical applications, coupled with extension of portal and integration products divide the integration scenario as "Pure Play" and "Application Platform suites". This makes the selection of the integration product for an enterprise, who are beginning the integration journey, difficult as discussed earlier.

On the other hand, Java has emerged as the single scripting language of choice for pure play integration products and all the integration logic can be developed in Java. All other proprietary scripting languages are sidelined and forced into oblivion owing to the developer enthusiasm and popular demand for Java. As a result almost all integration products offer Java as the main scripting language. This means that while one is still using a proprietary integration product, the learning curve is smoothened by the elimination of the necessity of learning a proprietary language.

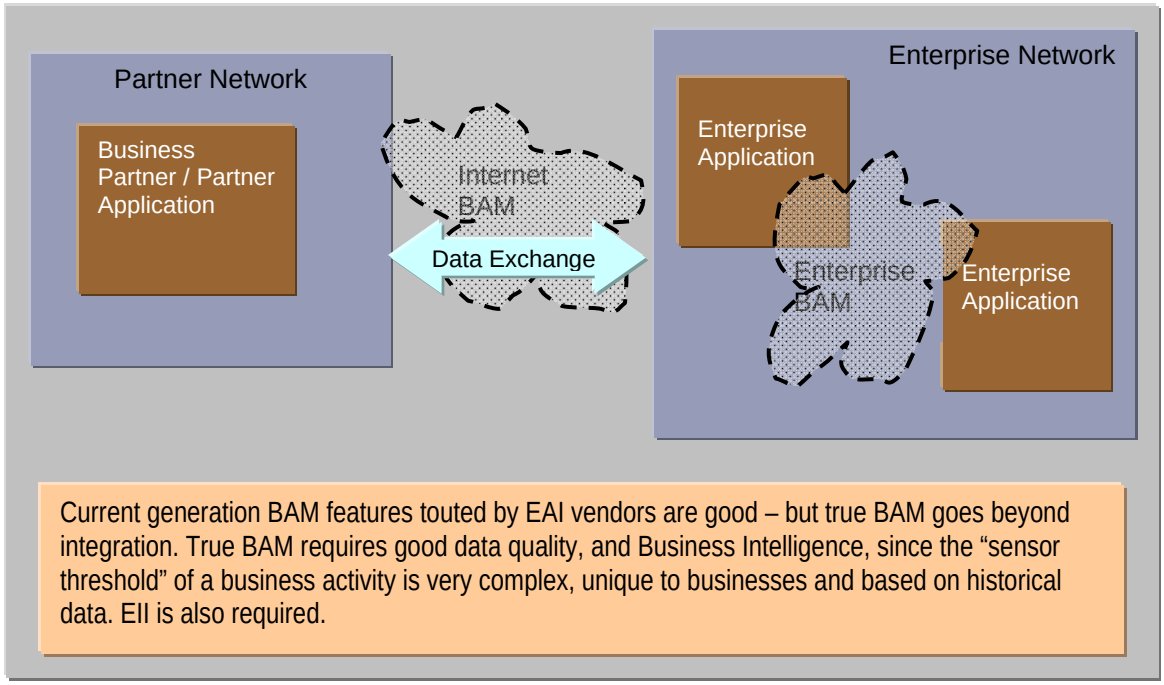
JMS rules the messaging standards in integration and even all pure play software vendors offer JMS as the standard. This gives flexibility for the enterprise to choose the messaging vendors, which is the JMS provider, based on available infrastructure and/or enterprise's standards.

Integration and BAM

Business Activity Monitoring (BAM) is one of the most touted features of the current pure play integration products. This concept is not new: enterprises' always need to have decision making tools. However, the data needed for the decision making had been in silos, data quality was poor, and by the time data reached the decision maker it might have been too late. But, because of the advancements in integration, the executive dashboards can be populated with real time data (for example, based on the inventory levels the seller might want to change the pricing or a red alert is raised when a critical shipment was not made). Though all EAI vendors claim that they offer BAM, in my opinion, a true BAM goes beyond integration and integration and can only be an enabler for BAM. Like all things that last longer, BAM has to be designed for – thus, before BAM is possible, a robust data integration framework has to be put in place, where data is cleaned, rules applied, and what-if analysis done. EAI may be able to provide the "agents" for picking an important deviation or change, but the final decision making goes beyond integration.

All the EAI products are essentially "non intrusive," implying that the existing systems need not be changed extensively for integration. As an example, it may be possible to interrupt the purchase orders exchanged and calculate the total value of purchase orders (by vendor, per line item, and so on.) But linking this data with the number of orders placed in the back end and the value of the invoices raised, requires an integrated process flow rather than integration. Further, in a decision making process where business processes span across systems, the decision making is extremely complicated, which requires complex business intelligence tools and pattern matching. Thus, unless the systems are designed for providing data for BAM, the BAM features of the current generation of integration servers is limited only to providing alerts and notifications of happening or absence of simple pre-defined events. This means that the in the current generation, BAM can happen only within the domain of integration. True BAM is only possible when the underlying architecture provides hooks for BAM within and outside of the business process.

Figure 5. Business Activity Monitoring



Extended Enterprise

The concept of Extended Enterprise, where by the business' internal systems and business partners' systems interact seamlessly has taken root, thanks to the Internet technologies. This type of integration has more to do with the business process rather than the technology. Simply put, the level of integration is determined by the business needs, the participating parties' technology maturity, and the level of mutual trust. Most enterprises realize that this type of integration is needed, but they also realize that business process re-design in most of the cases is a pre-requisite to enable seamless integration with partners and suppliers. While the process design might happen, enterprises are taking three levels of integration in parallel:

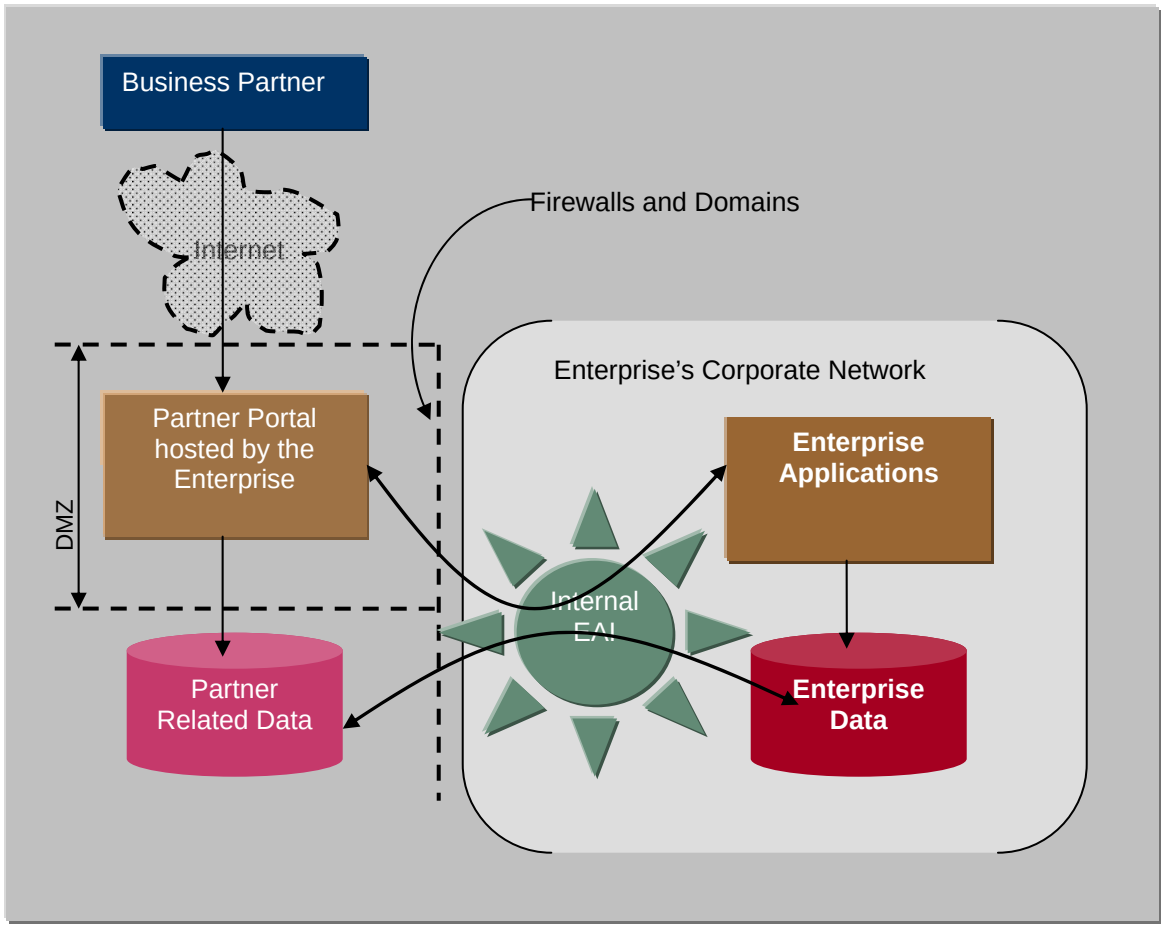
1. Data publication in Portals and Portal to Portal Integration
2. EDI over Internet or XML over Internet (through Portals or Integration Servers)
3. Web Services integration

Partner Integration Through Portals – Information Exchange

Extranet Portals have become significant mechanisms to transact B2B business transactions over the Internet. Business partners access the portals and can do many things: download manuals and historical information, (for example, the types of goods sold in the past month), receive orders, submit latest price list, view the status of the purchase orders, payments, and so on. Event-based alerts and notifications (for example, payment due or shipping delay) are passed on to the partners through the portal. These types of portals are different from mass market portals where customer self service (either online purchase or service request)

is the key thing, whereas in the B2B portals, self service, business transactions, and transaction integrity are very important. This saves a lot of e-mail messages, which would have been sent otherwise and also establishes the business contract more tightly – for example, it is not uncommon for the suppliers of a company to do a mandatory check on the partner portal for notifications or pending items. If the supplier is a large enterprise, it is likely that the supplier's processes are automated. Then information is exchanged using XML or a Web service using the portals, that is, by means of portal integration.

Figure 6. Business partner integration using portals



Since partners and suppliers will access the portal using the public Internet, appropriate security controls including network, application, encryption, authentication, and so on, are to be in place. Fundamentally, the Internet will be separated from the enterprise network, where core business data will be hosted. This minimizes the security vulnerability. Therefore, in many cases the partner data is duplicated (undesirable in data point of view, but recommended for better security.) This is depicted in the above figure. EAI is needed to provide data synchronization and application integration.

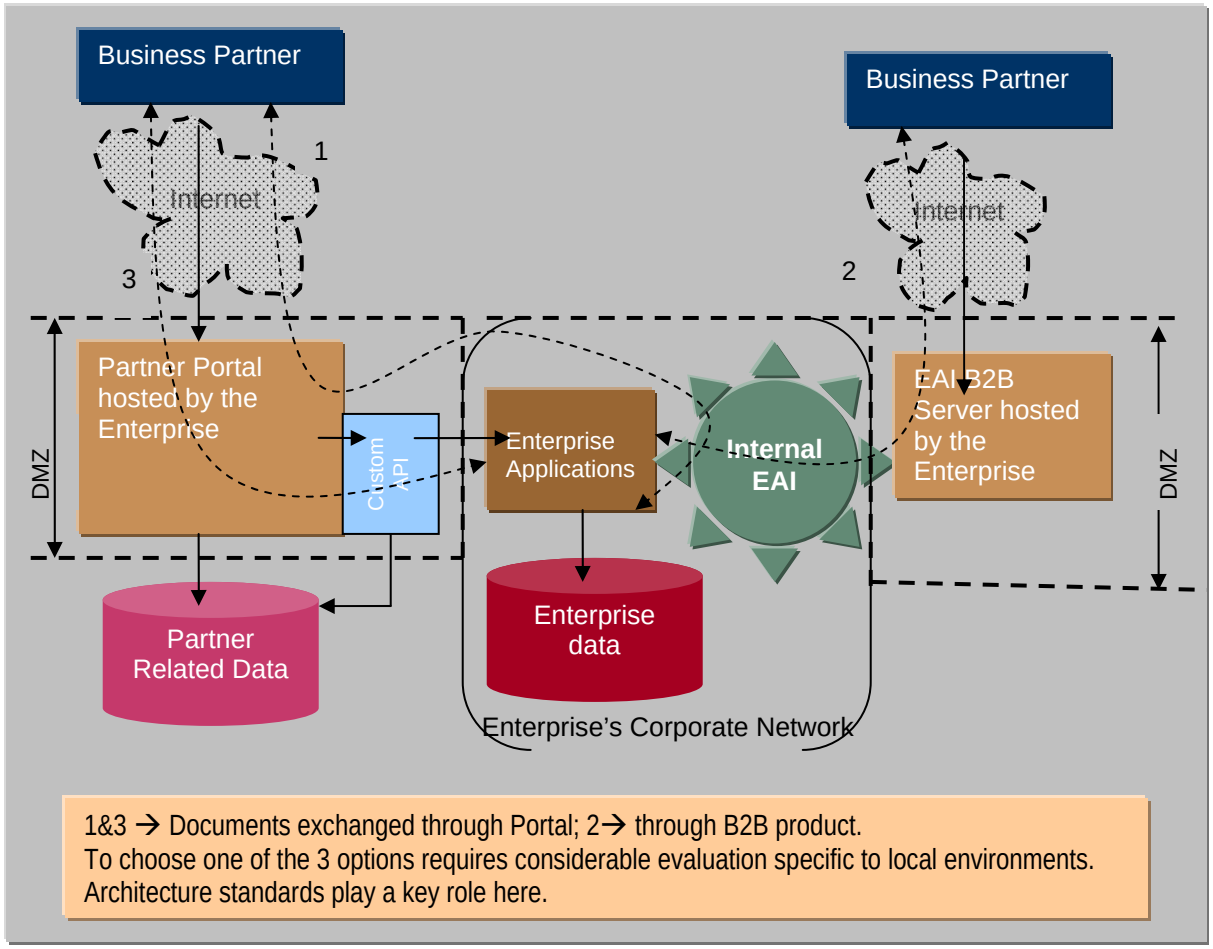
In the case of the business partners having their own portals, the data exchange happens through XML or a Web service. To make the exchange happen, the enterprise and the business partner need to have a technology contract of how the data will be exchanged and whose responsibility it is to disseminate the content down to the hierarchy of the partner's organization. Typically, custom built interfaces are used for this type of integration. With the advent of Java Portlets Specification 168 (JSR)*[3] and Web services for Remote Portals (WSRP)[4], it will become easy to integrate portals. Though promising this is clearly a future trend and is discussed separately in the subsequent sections.

Partner Integration– Document Exchange

The second level of integration is for exchanging business documents, where the event of the document should trigger further business process. This can happen in two ways, either

through the portal and EAI or through EAI. There are arguments for each of these but ultimately the choice will be based on the architecture style.

Figure 7. Partner integration: choosing an option is an enterprise architecture decision



It is clear that the three options need different types of approaches. In option1 and 3 the document exchange happens through the portal, and hence the portal has to be integrated with the enterprise applications using either custom interfaces or EAI. These two (custom or EAI) are still to be considered as options because the choice will depend on what sort of application architecture the portal applications follow and whether any data transformations are needed. EAI of course can be used without considering the custom option; but it may prove to be an overkill in many places where an API level integration might be a simpler solution. For example, if the solution requires data transformation and complex workflows, then EAI solution might be a better fit. In the absence of these two requirements EAI becomes redundant and a custom solution might be a better solution, if it is suitable for future requirements as well.

Web Service Integration

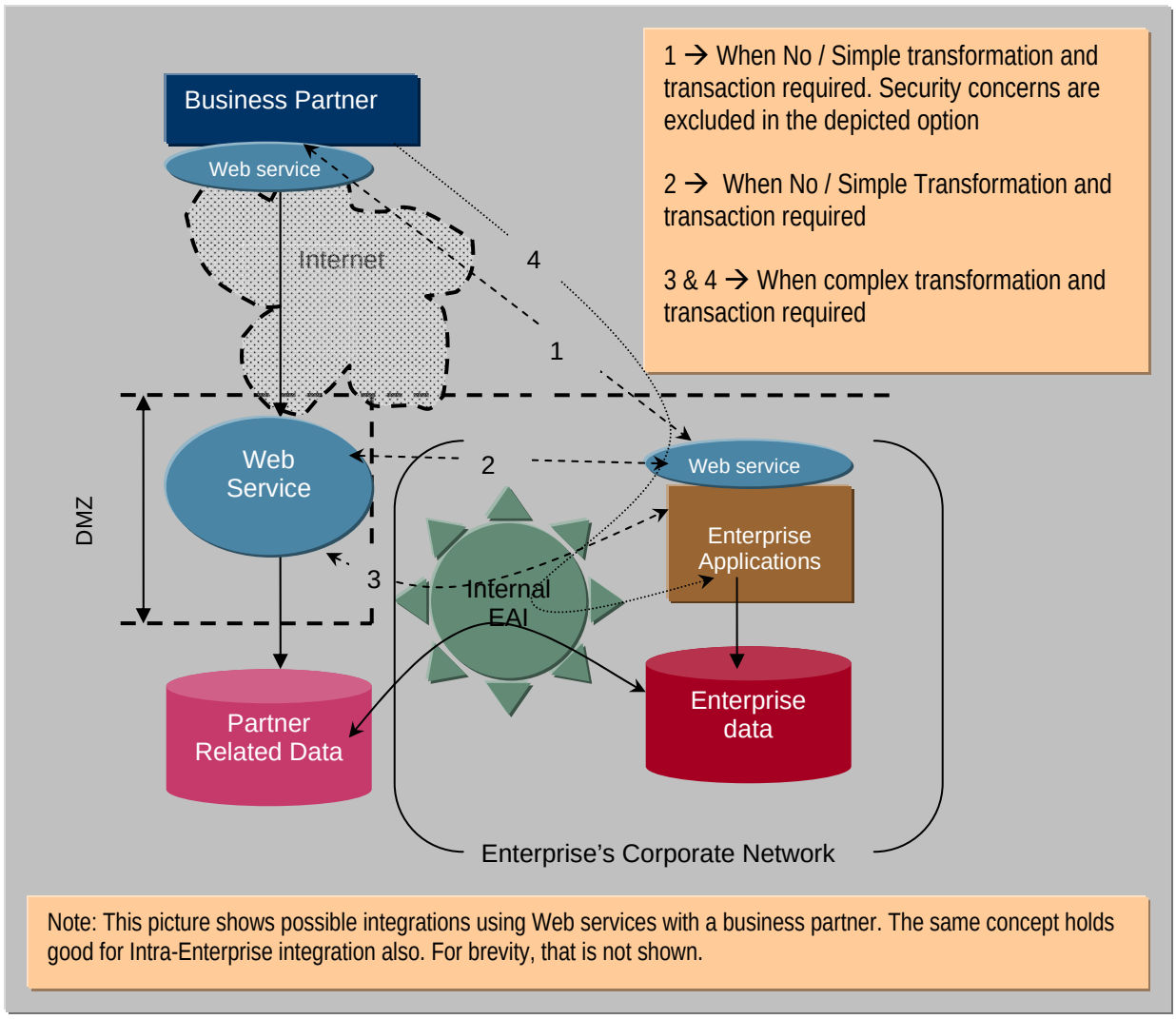
Integration through web services is one of the most popular topics today. As it is well known, web services provide a standard protocol, both for messaging (XML) and communication (HTTP), thereby allowing systems to interact (messaging+communication) in a standard way. This has received a lot of enthusiasm from the technology community and many organizations are involved in refining the standards.

While web services will be a great tool, where custom development is an option, to come to a level of sophistication provided by the “pure play” integration vendor, the web services technology has a long way to go. For example, integration products offer robust security, strong message handling capabilities like error handling, fail over, back up, store and forward, and assured delivery, routing mechanisms like publish and subscribe (groups-based, topics-based,) point–to-point, transformation mechanisms (format and schema conversions of many types and variations of business documents like EDI, XML, SOAP, Files, Proprietary formats and so on,) and communication and application adapters.

Web services can be made do all this but it will cost money to build these features into a custom-based integration solution. Specifically, building the routing and transformation engines will be tedious, where many formats and systems are involved. Therefore, Web services cannot replace the EAI products when many-to-many integration is involved, at least in the near future. But the good news is that all the integration servers offer web services as an option, and hence standards-based interfaces to the EAI product are possible. In this approach, the external integration is done using Web services while the bulk work of transformation and routing are done by the “core” EAI services.

But, for the point-to-point integrations, Web services can be directly used. Thus, a single system can directly interact with another system using Web services or using a common service, which is shared by others. The systems in question could be an internal (to the enterprise) system or a mix of internal and external (to the enterprise) systems.

Figure 8. Integration through Web services



Summary

Generation-Now has the following advantages:

1. Integration space is mature. Many types of integration (pure play, J2EE-based and Web services-based) have emerged, offering good choices for the enterprise.
2. Existing EDI investments need not be thrown away, owing to the EDIINT standard.
3. Owing to the maturity of technology and the development tools it is very easy to assemble any integration solution (when extensive data transformations are not required) even when expensive EAI products are not available with the enterprise.

One can assume that "Integration has arrived." If so, what will be next? The next section touches upon this question.

Generation Next – Service Composition and Orchestration

Impact of SOA and EA

Let's step back and ask a couple of questions – Why is there a need for integration within an enterprise? Why should an enterprise buy expensive integration tools? The answers to these questions can be traced back to the following:

1. Enterprises had multiple products and technologies on which applications were built. Enterprise applications are always a mix of packaged applications (example ERP) and custom applications (example Web applications and client server applications.) They were built in silos, possibly to solve tactical problems, and hence were not built to talk to one another. But changing business models mandate real time information access, for which these systems are to be patched. Custom built integration solutions were tedious and risky, particularly when the participating systems and the data formats are more and varied. Integration products provided improved productivity by means of pre-built transformation templates and adapters, which accelerated the integration project.
2. With the advent of the Internet as a low cost medium, data exchange with business partners became increasingly electronic. This space needs a high level of security, robustness, and assured delivery of information, demanding products of high quality. Integration product vendors offer such quality products, thus making inroads in to the enterprises.

Whatever be the reason, one can observe that integration as of today is only an antidote and quick fix for enterprise's problems. Fundamentally, at best it is a "patchy" solution, and "forced" – not "smooth", however sophisticated the solution is. This is because integration so far is applied after the applications are built in silos.

In Generation Next this trend is likely to change. One of the reasons for this is that integration concerns will be addressed from the beginning and not rather at the end. This is owing to the fact that enterprise architecture and standards are gaining importance within enterprises, which enforces discipline in choosing technologies, products, and architecture styles. However, the biggest impact is created by open standards (like Web services) and SOAs.

Service orientation, though increasingly discussed today, has been known and applied for decades – for example CORBA, Enterprise JavaBeans (EJB)*, and COM are meant to encapsulate services. However, the application of these technologies has limitations in terms of interoperability – they work well in an homogeneous environment, but extensive interfaces have to be developed for interoperability with other technologies, sometimes within the same technology (for example between two CORBA or EJB implementations by two vendors.) Ironically, instead of providing an integration solution, they became one more set of touch points of integration.

The notion of a Web service (whether they act upon SOAP messages or XML messages or proprietary messages) fuelled the revival of SOA. While some people think of the Web service as the solution to everything, in my opinion, they stand a very good chance of providing an interoperable interface. As a side note, I was involved in architecting two large projects using this principle, both involving mainframe applications. At the time I had only a vague idea of the value the architecture would provide. We re-engineered the applications to be service enabled and to work with Web applications. For one application we provided customer information control system (CICS) interfaces to act on XML from the messaging middleware. We converted the other application to provide CICS services acting upon messages from the middleware. These mainframe services can be invoked by sending XML (former) and proprietary (later) messages, which provides transparent integration. While they were designed for interoperability between only the mainframe and the Web, it's a small step for them to interoperate now with any system. They were designed for service orientation in general and for Web services technology in particular.

A successful *basic* SOA needs the following pre-requisites:

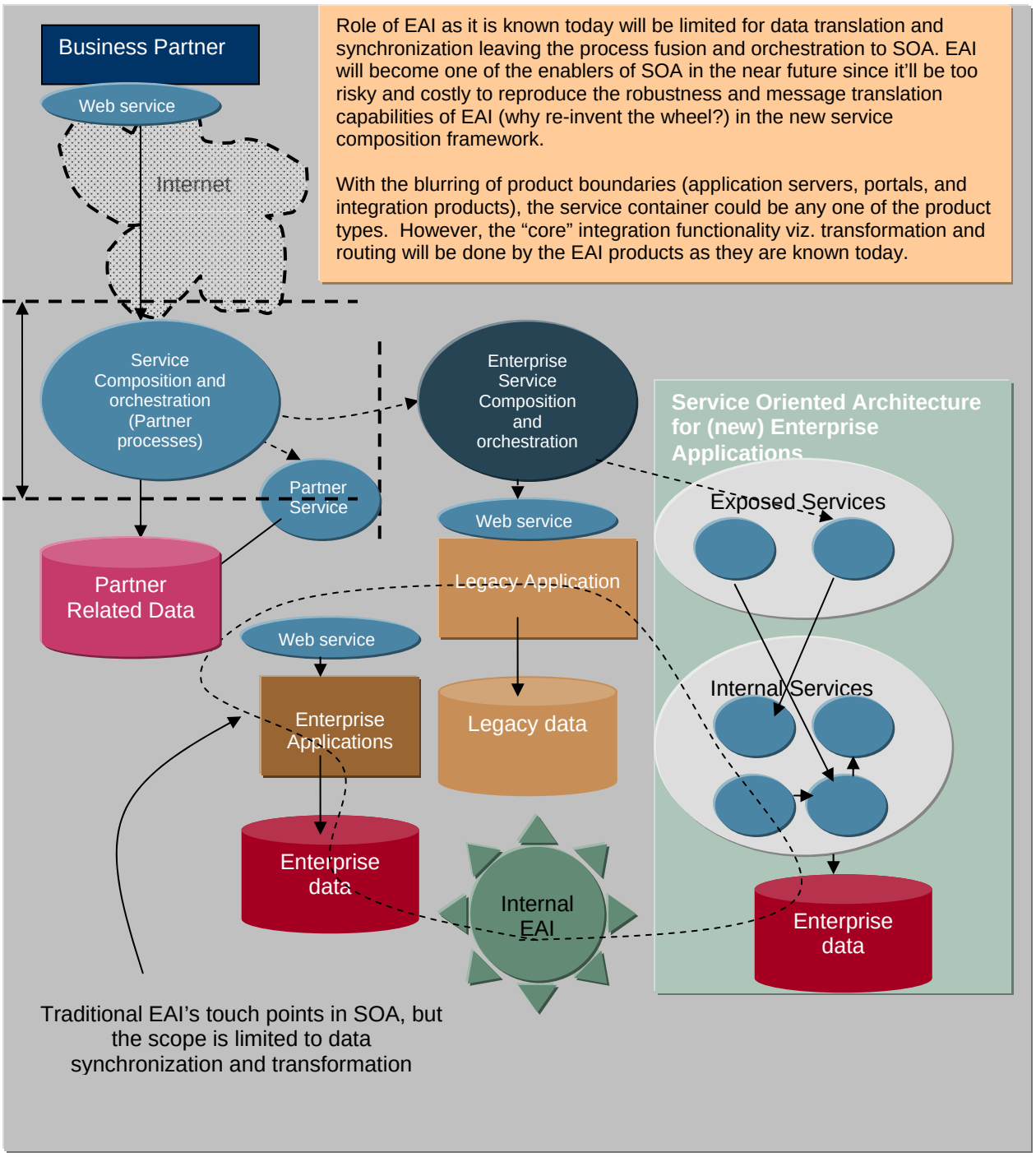
1. Modular, compact, and clean services. This has to be designed or wrapped (as in the above examples.)
2. Standard messaging mechanism and transactional handling mechanism between the services (either proprietary or industry standards-based.)

The Web services technology provides the following pre-requisites:

1. Advanced SOA needs mechanisms for dynamic discovery, binding, and fault tolerance. However, the basic requirements are to take roots in enterprises before dynamic discovery and other advanced service orchestration that can happen.
2. In an SOA-based architecture, integration is the beginning and not at the middle/end of projects. This makes integration a serious topic for enterprise architects.
3. Further, the widespread awareness of SOA makes proprietary package vendors (of ERP, CRM, or SCM) think in terms of services rather than black box products of yesteryears, which were once difficult to integrate. This accentuates the need of integration from the beginning and not at a later stage where monolithic applications were built and they needed to be integrated. However, this doesn’t mean that proprietary applications will become in their entirety “open” and service oriented. It implies that while the core processing will be proprietary, the interfaces will become standards-based, making the smooth integration of proprietary services with other custom/proprietary services.

These changes make the world of integration to be transformed into “composition” where multiple services are composed to provide end goals. The following diagram shows a possible scenario in the new world of composition.

Figure 9. Service oriented architecture for integration

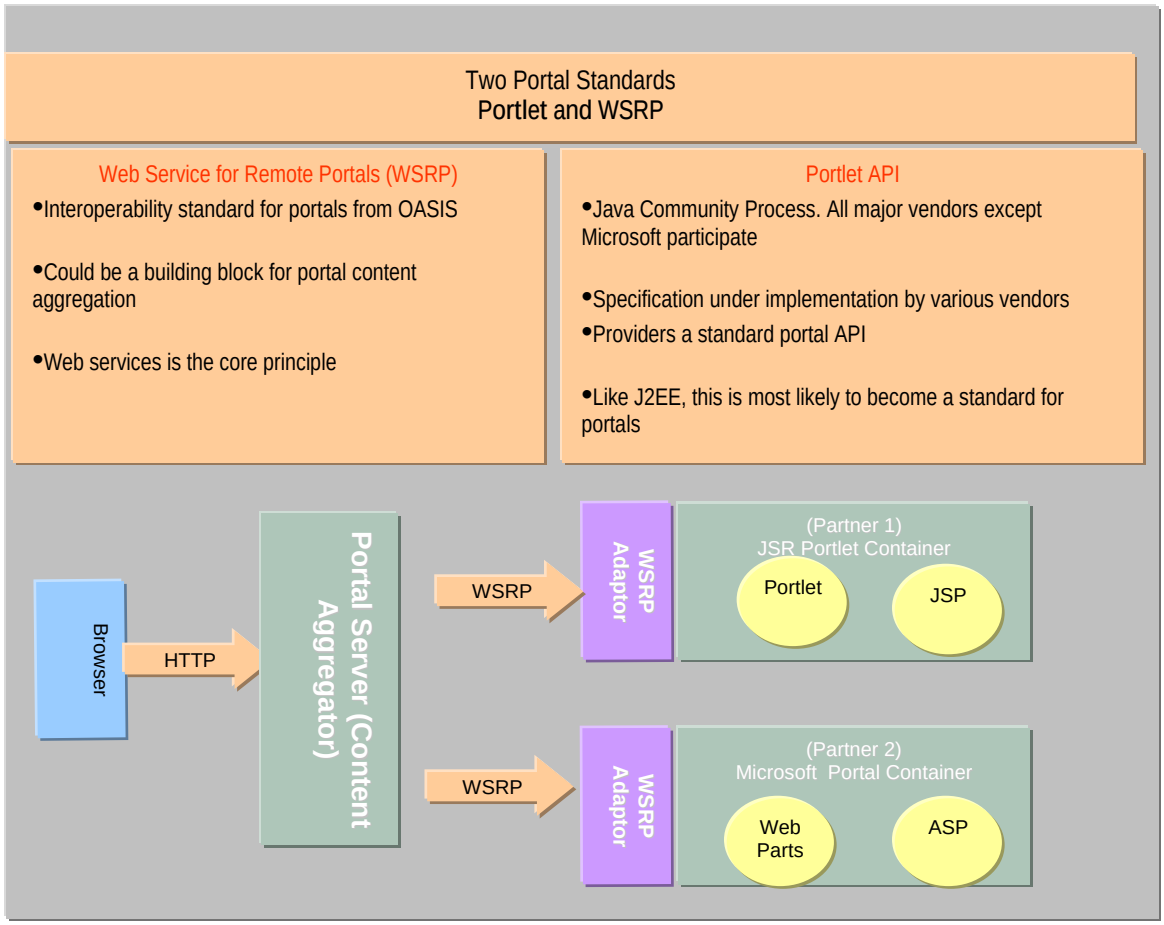


Portal Integration

On the portal front, already portal-to-portal integration is gaining a lot of attention. In the Java portal market, the Java Community Process's Portlet standard has been formulated and various vendors already offer some form of support for this. The Portlet standard provides a mechanism for developing portal pages, and thus this concept can be extended

for integration between two Java portals. Another standard, which needs further industry support before becoming a widely used standard is related to Web services and is known as the WSRP. This allows for interoperability between various portlets.

Figure 10. Seamless UI integration using portal standards

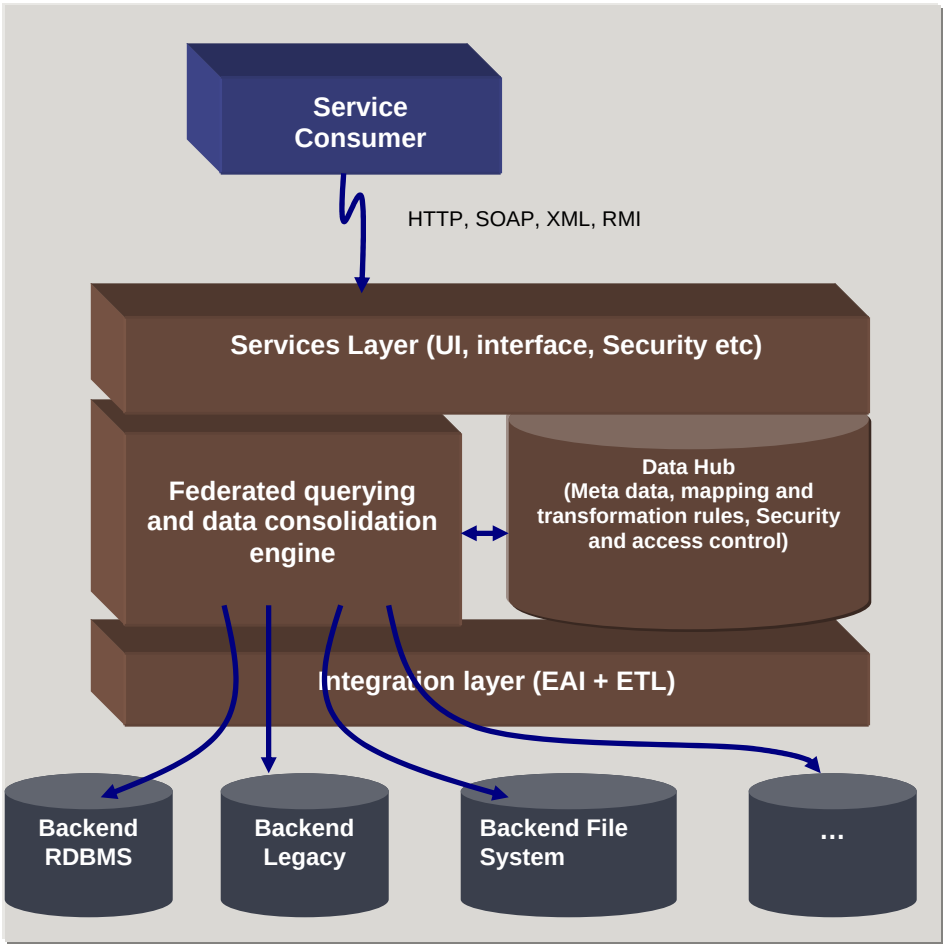


Enterprise Information Integration (EII)

Data Integration, one of the key areas of integration has been handled by the Extract, Transform and Load (ETL) vendors. These tools help to move data from one place to another in bulk form and allow applying format changes to the data. For smaller amount of data, which needs to be synchronized much more frequently, EAI tools are in use. There are many cases in which these two types need to be integrated[5]. Enter EII.

The following diagram shows the components of EII.

Figure 11. EII components: federation is key, so is meta data management



Today, EII is equated with database federation. If EII is synonymous with virtual federation of databases, then many of the products today offer solutions - though they have to be proven in enterprise-class, mission-critical situations. These solutions will result in some quick wins. However, if EII is true to its word, that is "information integration", then it must be part of the enterprise data strategy. Enterprise data integration requires solutions that range from point solutions to custom solutions and should holistically address the enterprise level meta data management, business workflow management, and integration management in a streamlined and cohesive manner.

Summary

The standardization of architecture at the enterprise level will mean that the integration architecture has to follow the enterprise architecture's strategies and standards, and this changes the scenario where the Integration is given a serious consideration from the beginning rather than at the end. This means that the various decisions and strategies of integration like Web services vs. pure play integration, custom interfaces vs. Web services, legacy code re-use vs. re-engineering will be made as part of enterprise standards and

accordingly products and technologies will be chosen. In this scenario, the strategists and architects will not be satisfied with simple integration, unless integration is part of a large picture. The enterprise architects will prefer a unified approach for development. Demands about unification of business process modeling (visual modeling), execution (run time), and orchestration (process integration in heterogeneous environments) shift the focus from simple application integration to process integration. Already application server vendors are likely to offer J2EE Made Easy* (J2EZ), and proprietary modeling (pure play EAI vendors). This trend will further consolidate to more uniform, end-to-end offering using Web Services Flow Language (WSFL) and BPEL4WS, PD4J, and so on, though it will take some time for these technologies to penetrate into mainstream enterprise modeling.

The next generation integration will be centered on strong enterprise's architecture standards. SOA and Web services offer standardization of technology and integration within and outside the enterprise. In this new paradigm, EAI products will still provide a strong integration backbone for messaging, transactions, and transformation, but the focus will shift from tactical application integration to strategic business process integration and orchestration. These strategic needs cannot be fulfilled by a product alone and can be satisfied only with deeper architecture cohesion, which will be brought in by SOA and Web services.

Additional Resources

1. CORBA was the king in the making of Integration- one of the largest CORBA implementations is at Boeing http://www.iona.com/pressroom/archive/boeing2.html.

2. EDIINT Standards are formulated by IETF for SMTP and HTTP. The chapter home page and the AS2 standards are available at http://www.ietf.org/internet-drafts/draft-ietf-ediint-as3-03.txt. The success of EDIINT is captured in this article: http://www.nwfusion.com/news/2004/0223as2.html?page=2

3. OASIS technical committee on WSRP Web site describes the WSRP specification" http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

4. An Open source version of JSR 168 implementation (Pluto* and Jetspeed*) is available with Apache*: http://portals.apache.org/

5. EII is a Strategy and not a technique, S. Radhakrishnan, http://www.dmreview.com/article_sub.cfm?articleId=1014857

About the Author

Dr. S. Radhakrishnan is a senior enterprise class solution architect in TATA Consultancy Services. He is responsible for the architecture and open source initiatives of his organization. His primary focus is to analyze, define, and implement architectures for enterprise applications, and he has been responsible for creating high-volume, mission-critical applications and for providing guidance on architecture issues to a multitude of international customers. He also holds a PhD in computer vision/image processing from Indian Institute of Technology, Chennai, India. His active professional interests include Web and distributed architectures and application integration.

---

[1] CORBA was the king in the making of Integration- one of the largest CORBA implementations is at Boeing http://www.iona.com/pressroom/archive/boeing2.html.

2 EDIINT Standards are formulated by IETF for SMTP and HTTP. The chapter home page and the AS2 standards are available at http://www.ietf.org/internet-drafts/draft-ietf-ediint-as2-15.txt. The success of EDIINT is captured in this article: http://www.nwfusion.com/news/2004/0223as2.html?page=2

4 An Open source version of JSR 168 implementation (Pluto and Jetspeed) is available with Apache: http://portals.apache.org/

5 EII is a Strategy and not a technique, S. Radhakrishnan, http://www.dmreview.com/article_sub.cfm?articleId=1014857