

Infogain Perspective

SOA versus traditional integration options by Krish Khambadkone

Prior to the advent of SOA, the true contender for Enterprise Integration was EAI. A plethora of tools and technologies emerged in this space quickly to fill the void. However, due to a lack of standards in the initial stages several issues arose, including:

- Proprietary vendor toolsets leading to lengthy learning curves
- Complex and lengthy implementation cycles
- Department- or Business Unit-Specific containering for EAI implementations
- Limited application shelf life
- Technology-driven implementations without the requisite business goal analysis.

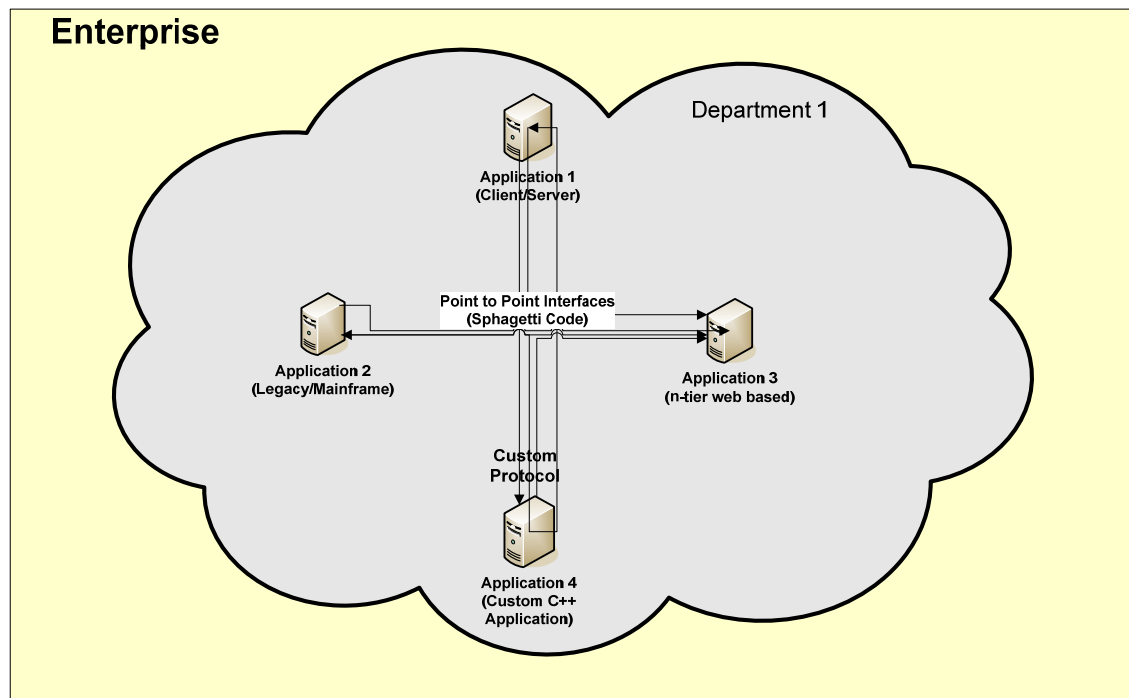
With each vendor defining its own approach to EAI, the selection of the most strategic toolset was often a daunting task in and of itself. The following table highlights some of the differences between the two approaches.

EAI	SOA
Technology Driven	Business Driven
Project Based, confined to Department or Business Group	Enterprise Driven, Company-wide effort
Generally a Bottom Up approach, driven by product and technology	Starts with Top Down approach, followed by bottom up and then settles with Hybrid
Partially supported by standards	Almost wholly standards based at all levels with XSD, WSDL, JAX-WS, BPEL etc.
Extension of client server and Point to Point (Adapter) integration paradigms	New Paradigm that requires a new train of thought
Can work successfully with traditional software development methodologies such as SDLC, SCRUM, Agile	Needs new types of control mechanisms such as Governance and Competency Centers in addition to traditional methodologies

Integration pattern generally resembles Hub and Spoke	Several patterns of integration possible, of which Hub and Spoke can be a component
Does not lend itself to enterprise-wide integration	Complements your existing investment in middleware by adding an Enterprise Integration Layer on top

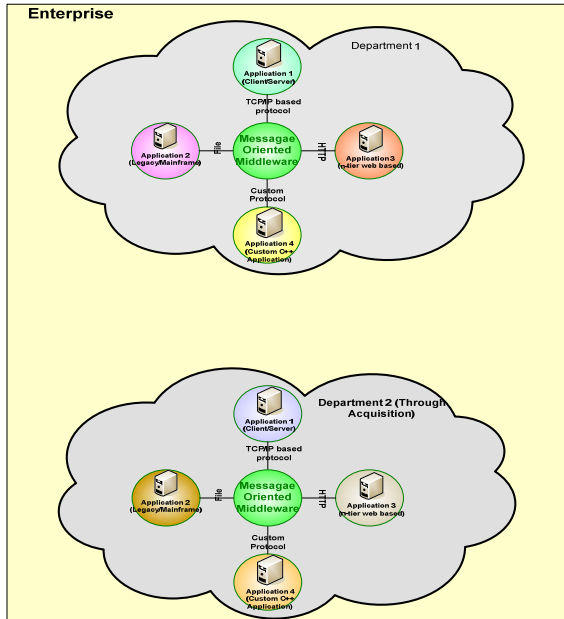
The following sections highlight the pitfalls and difficulties associated with the various approaches to integration and how the maturity of technology over time addresses these issues. As the types and number of applications within the enterprise grow, the need to have a seamless integration mechanism arises. There is a typical integration pattern that has formed and matured over the years:

1. **Point to Point Integration:** Ad hoc Integration involving direct custom coded interfaces between Application 1 and Application 2 with no mediation in the form of middleware.



EAI-oriented Mechanisms and their Pitfalls

2. **EAI with Message Oriented Middleware:** This was the first attempt to introduce a mediating agent to avoid and mitigate the risks and pitfalls of point to point interfaces.



Pros:

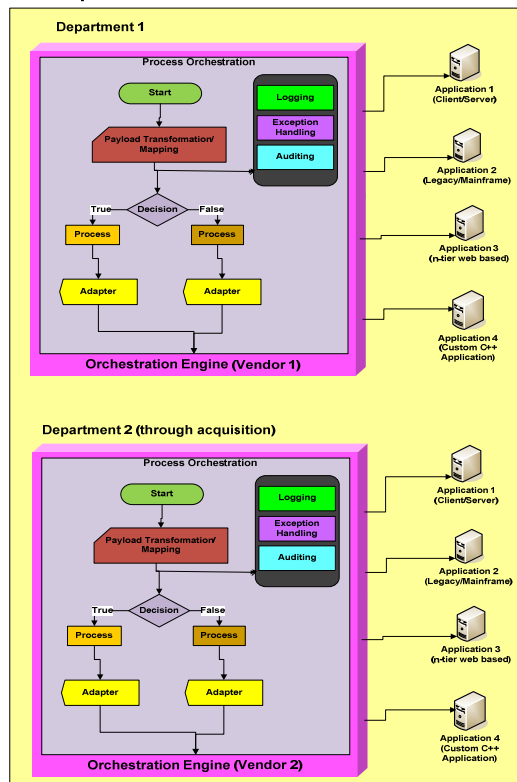
- Centralized processing of information through message normalization
- Provided for synchronous and asynchronous operations
- Operated on a hub and spoke paradigm that was easy to understand and adopt.

Cons:

- Lack of standards, wherein each vendor had proprietary APIs, scripting and tools
- High barriers for entry in the form of acquisition costs and learning curve
- Low shelf life as next generation toolsets made them obsolete very quickly
- No enterprise-wide visibility leading to proliferation of several integration efforts within the organization.

3. **EAI with Business Process Management (low level process orchestration):** These are the orchestration engines that were the earliest precursors of today's advanced BPEL engines. They brought together various elements of the enterprise such as Databases, EIS systems like SAP and Oracle, and J2EE elements like JDBC, JMS and EJB. They then orchestrated them as processes or single atomic units of work while taking care of transaction management and session resilience.

Enterprise



Pros:

- Mitigated a lot of the issues associated with MoM-oriented integration
- Users could design processes using graphical drag and drop palettes
- Typical application development issues like Transaction Management and Session Resilience was taken care by the BPM Tool
- Allowed for Straight Through Processing with focus on SLAs and QoS.

Cons:

- Initial offerings lacked standards though the later ones that followed were Java-based by generating code artifacts and allowing the user to implement custom code elements
- Cost was a barrier for entry as these tools were expensive to acquire
- B2B interactions were possible only through protocols like RosettaNet, cXML etc.
- No enterprise-wide visibility; confined to Department and Business Unit.

Service Oriented Architecture

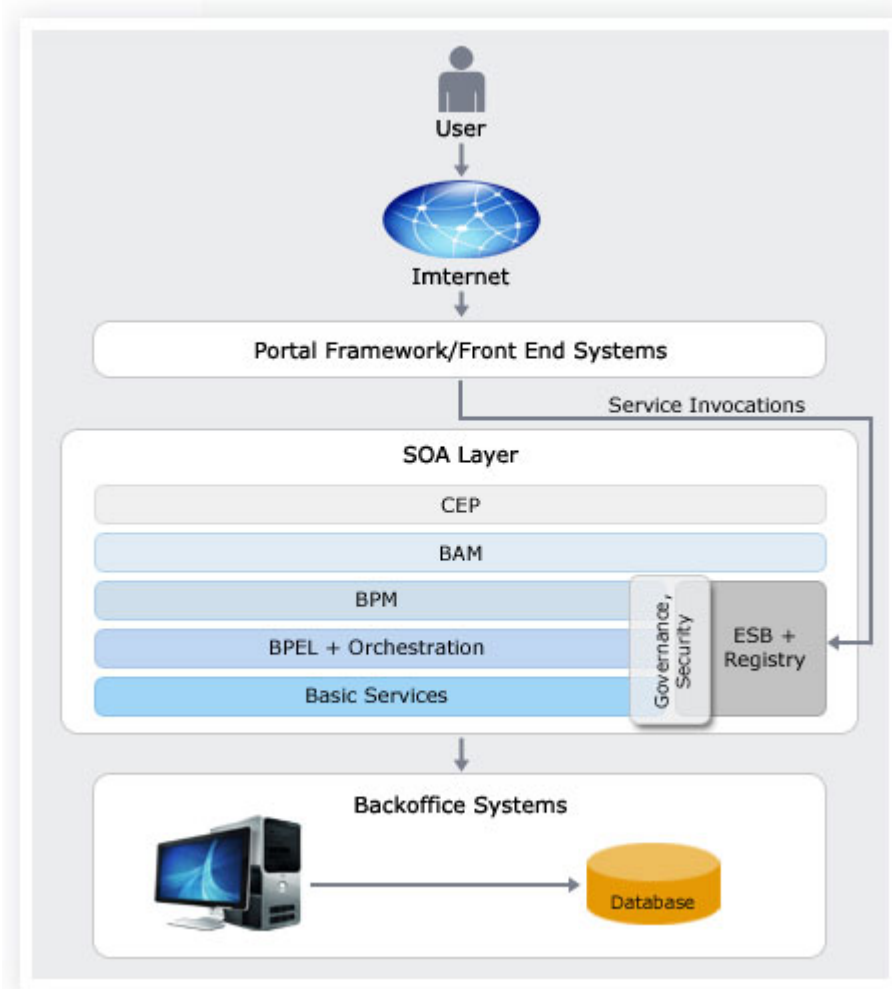
4. Service Oriented Architecture (SOA):

This is the current state-of-the-art in integration technology. SOA is a whole new paradigm that follows a maturity pattern and requires a prescribed set of practices, principles and technologies to support each maturity level.

SOA with Business Process Management (Process Orchestration and Service Orchestration using BPEL and BPMN)

Implementation of SOA at the ground level starts with the implementation of very simple services that typically involve a single step—most likely a CRUD (Create, Read, Update, Delete) operations to a database. More complex multi-step orchestrations are implemented and driven by standards like BPEL and BPMN.

Even after the tools have advanced this far and greatly eased and standardized application integration, this is an ongoing process. BPEL tools will be used to build services by pulling together various elements of the enterprise and making them available as enterprise-wide shared services. As the number of services increase, additional toolsets will have to be adopted to support this ongoing maturity model, as illustrated in the following diagram:



SOA brings together another set of essentially web technologies like HTTP, SOAP, XML, XSD and WSDL to help democratize your applications and business processes. Data representation standards like XML, XSD and SOAP make it highly declarative and transform SOA implementations into business-driven solutions with technology becoming an afterthought—a bridge you could cross later. In a nutshell, with SOA your application functionality can be delivered over the web either in a collective or a discrete fashion.

Summary

This need for enterprise integration arose with the proliferation of enterprise applications such as SAP and Oracle Apps built around databases and the increasing need for data exchange between these systems.

The earliest attempts at integration often used ad hoc interfaces that resulted in the term "spaghetti code". The next attempt was Message Oriented Middleware. Development and maintenance of such systems required deep technical skills due to the lack of standards, and implementations were essentially very tactical in nature. Then entered LLPO or low level BAM, which gave a human face to integration. Multi-step processes could be orchestrated through IDEs that provided drag and drop widgets for design time construction of complex processes.

However, none of these methods solved the deeper issues around true *Enterprise Integration*. It wasn't until the advent of web services and SOA that we realized a truly game changing strategy for integration. With SOA, these processes can be fronted by a service which is built on a set of industry standards. The whole complex process can also be centralized and viewed as a function or API that can be accessed using a familiar URL.

SOA has not only streamlined application integration within and between enterprises but has also made possible new paradigms such as SaaS and Cloud computing. EAI itself is a stepping stone for achieving the goals of SOA in the enterprise. Likewise, SOA is a prerequisite to full-on execution of SaaS and Cloud strategies. These additional computing paradigms are out of the scope of this article, but are excellent options for future topics of discussion.