Enterprise Service Bus (ESB)

CIS 8090 Fall 2016
ESB

- A frequent architectural element
- Structural architectural backbone
- Essential part of Service Oriented Architectures (SOA)
- Applicable for local and web/cloud configurations
SOA principles
Figure 2: Centralized control over distributed ESB infrastructure

Diagram showing the relationship between clients, interface or adaptors, runtime engines, and providers within a distributed infrastructure.
Figure 3: The role of the ESB in an SOA

- External Service Requesters
- Internal Service Requesters
- Business Service Choreographer
- B2B Gateway
- Enterprise Service Bus
  - Routing, transformation, mediations, security, and so forth
- External Service Providers
- Internal Service Providers
- Service Routing Directory
- Business Service Directory

Infrastructure components for service-oriented architecture
ESP defined (Wikipedia)

- refers to a **software architectural** construct,
  - implemented by technologies in a category of **middleware infrastructure** products,
  - based on **Web services** standards,
  - provides foundational services for more complex **service-oriented architectures** via an event-driven and XML-based messaging engine (the bus).

- The ESB generally provides an abstraction layer
  - **on top of** an **Enterprise Messaging System**
  - allows integration architects to exploit the value of messaging without writing code.

- Contrary to the more classical **EAI** approach of a monolithic stack in a hub and spoke architecture, the foundation of an **enterprise service bus** is built of base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary.
alternatively

- for high performance enterprise service buses,
- [other] "standard" message formats should flow across the bus, not just XML.
  - Generating XML and parsing it can be costly in terms of processing and memory, and high volume scenarios may not be viable.
According to HOME DEPOT’S MANAGER OF ENTERPRISE ARCHITECTURE, CHRIS HUFF

- “An enterprise service bus (ESB) enables a business to make use of a comprehensive, flexible and consistent approach to integration.”

- “Due to the complex and varying nature of business needs, an ESB is essentially an **architectural pattern** which unifies message oriented, event driven and service oriented approaches to integration.”

*IBM’s Definition (to me is most succinct and from the leader in the space)*
• an ESB is basically an EAI hub that supports the standards required for SOA and supplies the tools required to provide real-time service enablement.

• The tools/capabilities of an ESB should include:
  • a message transport mechanism,
  • message transformation and brokering,
  • support for all open protocols (SOAP, etc.),
  • message workflow,
  • a wide array of resource adapters,
  • a service repository,
  • audit capability and global event and transaction management.
Rally Projects around the ‘vision’
ESB - a core component in a SOA
(from Loosley Coupled
http://looselycoupled.com/glossary/ESB)

- [The ESB is a] Universal integration backbone.
- ... acts as a shared messaging layer for connecting applications and other services throughout an enterprise computing infrastructure.
- It supplements its core asynchronous messaging backbone with intelligent transformation and routing to ensure messages are passed reliably.
- Services participate in the ESB using either web services messaging standards or the Java Message System (JMS).
- Originally defined by analysts at Gartner, ESB is increasingly seen as a core component in a service-oriented infrastructure.
Key Benefits

- faster and cheaper accommodation of existing systems
- increased flexibility: easier to change as requirements change
- standards-based
- scales from point solutions to enterprise wide deployment (distributed bus)
- more configuration rather than integration coding
- no central rules engine, no central broker
ESB’s Salient characteristics

- it is an implementation of Service Oriented Architecture
  - it supports Web services standards

- Ideally it is operating system and programming language agnostic; it should work between Java and .Net applications, for example

- supports messaging (synchronous, asynchronous, point-to-point, publish-subscribe)
  - often uses XML (eXtensible Markup Language) as the standard communication language.

- it includes standards-based adapters (such as J2C/JCA) for supporting integration with legacy systems
Salient characteristics - continued

- it includes support for service orchestration & choreography
- it includes intelligent, content-based routing services (itinerary routing)
- it includes a standardized security model to authorize, authenticate, and audit use of the ESB
- it includes transformation services (often via XSLT) between the format of the sending application and the receiving application, to facilitate the transformation of data formats and values
- it includes validation against schemas for sending and receiving messages
- it can uniformly apply business rules, enrichment of the message from other sources, splitting and combining of multiple messages, and the handling of exceptions
Salient characteristics—continued

- it can conditionally route or transform messages based on a non-centralized policy - meaning that no central rules engine needs to be present

- it is monitored for various SLA (Service-Level Agreement) thresholds message latency and other characteristics described in a [Service Level Agreement](#)

- it (often) facilitates "service classes," responding appropriately to higher and lower priority users

- it supports queuing, holding messages if applications are temporarily unavailable

- it is comprised of selectively deployed application adapters in a (geographically) distributed environment
Key Disadvantages

- Enterprise Message Model is usually mandatory
  - Value of the ESB requires many disparate systems to collaborate on Message Standards
  - Versioning of messages between systems, if not planned for, can cause tight coupling instead of the intended loose coupling
  - Extra translation layer when compared to regular messaging solutions

- Requires more hardware to run

- Requires learning new skillsets

- Rarely realizes ROI (Return On Investment) on first few projects
  - Next few projects generally refine messages and generalizes services better.
  - The fifth project may begin to realize ROI.

- Requires mature IT governance model and a well defined enterprise strategy already in place to implement an ESB effectively.