The Enterprise Service Bus: Making Service-Oriented Architecture Real

M.T. Schmidt et al.

Presented by: Mikael Fernandus Simalango

**Introduction**

**SOA in Early Days**

- simple publish-find-bind triangle
- plain vanilla interaction:
  - request-response between service requester and provider
  - how about other interaction patterns eg. asynchronous invocation, publish-subscribe, complex events?
  - we need more capabilities and flexibility

**HOW?**
Enabling Enterprise Service Bus

Formal definition:
- The infrastructure which strengthen a fully integrated and flexible end-to-end SOA by providing connectivity layer between services
- Does not include business logic of service providers, requestors, or containers that host the services

Services Revisited
- A service: a software component that is described by meta-data, which can be understood by a program
- Distinguishing feature of a service: meta-data descriptions are published to enable reuse of the service in loosely coupled system across networks

Hence, ESB basically deals with the meta-data
**ESB and meta-data**

- Meta-data contains description of service requestors and providers, what they require and capable of providing, respectively
- The meta-data is independent of implementation specifics
- This meta-data is stored in ESB registry to assist the process of mediating and matching requestors and providers (link matching)
- All meta-data can be discovered, used, and modified at runtime

**Service capability and requirements declaration for meta-data**

![Diagram of service capability and requirements declaration for meta-data]
Core ESB Components

- Service Registry
- Link
- Mediation pattern

Service registry manages meta-data about service interaction endpoints and also information about domain model.

Domain model can be:

- A standard message sets representing general knowledge about a topic space
- Complex ontology describing concepts and their relation in a particular topic space
**ESB service registry content**

Endpoints need to register with the ESB.

Registered service requestors are represented as **bus service requestors** (BSRs) and registered service providers are represented as **bus service providers** (BSPs).

Service providers that are not registered as BSPs are invisible to the ESB.

ESB also holds details of **links** and **mediations**.
ESB Links and Mediations

ESB supports two concepts to facilitate interactions between endpoints:

- **Links**
  - Between service requestors and providers (interaction endpoints)
  - Enable basic connectivity between interaction endpoints with a configurable QoS

- **Mediations**
  - Between interaction endpoints
  - Connectivity by dynamic alterations to routing and QoS
  - Allow interaction endpoints to modify their behaviours

Both realize the contract between interaction partner that is implicit in the declaration of the capabilities and requirements

ESB Links

- Has two endpoints
  - One for attachment to BSPs
  - The other for attachment to BSRs

- A link defines “ideal counterpart” for service requestors and providers
  - Can be configured manually
  - Can be created dynamically based on requirements and capabilities of the endpoints
Problem: existing applications were seldom designed to be linked together
  – Protocol mismatch
  – Format mismatch
  – QoS mismatch

Addressing the problem: ESB mediation
  – Interposing mediations between service requestors and providers
  – It can reconfigure the links between requestors and providers

Hence, the role is to satisfy integration and operational requirements within the infrastructure
**Mediation in ESB integration model**

![Diagram of ESB Mediation Model]

**Mediation point**
- At the requestor -> mediation will be performed regardless of provider for the requestor
- At the provider -> mediation will be performed whenever provider receives a request, regardless of the requestor

**Interface mediation**
- Operate on the message payload, can change its content and structure
- Message payload: information required by service provider

**Policy mediation**
- Operate on message context
- Message context: available in message header, containing additional QoS and routing information about the link and mediations required between service requestor and provider
Basic patterns for mediation:

- **Monitor pattern**
  - Used to observe messages as they pass through the ESB without updating them

- **Transcoder pattern**
  - Changes the format of the message payload without changing its logical content

- **Modifier pattern**
  - Updates the payload of the message without any change to the context information

- **Validator pattern**
  - Determines whether a message should be delivered to its intended destination or not

- **Cache pattern**
  - Returns a valid response to the requestor without necessarily passing the request to a service provider

Basic patterns for mediation (cont’d):

- **Router pattern**
  - Changes the intended route of a message, selecting between the service providers associated with the mediation

- **Discovery pattern**
  - Queries ESB registry to discover the set of service providers that match the requirements of the requestor, selects one of them, and routes the message to it

- **Clone pattern**
  - Makes a copy of message and modifies its route

- **Aggregator pattern**
  - Monitors messages from one or more sources over a time period and generates a new message or event, based on the input it considers
ESB Usage Patterns

- Brings abstract patterns into real-world implementations
- Provide a means for describing and defining interactions and component topologies at the system or solution level
- Fundamental concept: broker application pattern
  - Distribution rules are separated from applications
  - Enabling flexibility in the distributions of requests and events
  - Reducing the growth of point-to-point connection
  - Simplifying management of network and system

ESB Usage Patterns (cont’d)

- Variations of broker application pattern:
  - Service and event-routing pattern
  - Protocol switch pattern
  - Proxy or gateway pattern
  - Event distribution pattern
  - Service transformation pattern
  - Matchmaking pattern
Service and Event-routing Pattern

** A request or event is distributed to at most one of multiple target providers
** Target selection can be made based on availability, workload, or detection of error situation after looking up appropriate service providers in the service registry

Protocol Switch Pattern

** A routing pattern
** Requestors and providers use different network protocols
** From the example:
  – SOAP/HTTP requests are mapped into SOAP/JMS infrastructure
Proxy or Gateway Pattern

- Another routing pattern
- It maps service interface or endpoints, usually to provide security functions or auditing capabilities
- A single point of contact is provided for multiple services and the details of inner services can be hidden from the service requestors

Event Distribution Pattern

- Events can be distributed to one or more target provider
- Service requestors may subscribe themselves to get notification about certain events of interest
Service Transformation Pattern

- Requestors and providers use different service interfaces or providers of same business function provide different interfaces
- ESB provides necessary translation for the differing interfaces

Matchmaking Pattern

- Another routing pattern
- Suitable target services are discovered dynamically based on a set of policy definitions
- Used in dynamic environments with hundreds or thousands services attached to the ESB
**ESB leverages an integrated and flexible SOA**

**Service meta-data managed through a service registry is the key component of ESB**

**Clear definition of the interfaces, capabilities and requirements of the service will enable mediations to handle differences between service requestors and providers**

**Several ESB usage patterns exist to articulate abstract ESB concept into enterprise implementation**

THANK YOU