# SWENG568: Enterprise Integration

## *Lesson 5: Connectivity and Enterprise Service Bus (ESB)*

## Learning Objectives

- Understand Different Perspectives of Connectivity in Enterprises
- Master the Conceptual Model of Enterprise Service Bus: A Pillar of Enterprise Integrations
- Get Familiar with One of ESB Products

By the end of this week, make sure you have completed the readings and activities found in the Lesson 5 Course Schedule.

## Different Perspectives of Connectivity in Enterprises

Users are requiring more relevant, more customizable access to data and information to improve their productivity and responsiveness. IT organizations are thus under increasing pressure to find ways to simplify software connectivity to a wide variety of data and information sources so as to consistently deliver the needed data and information on demand.

The integration architecture evolves as the computing technologies advances. As we discussed earlier, compared to point-to-point architecture, the improvement in terms of ease of integration, reliability, and scalability using peer-to-peer architecture with support of asynchronous messaging is substantial (Figure 5.1). Technological heterogeneity can be substantially ameliorated as more Web Services based applications becomes available. However, there will always exist a variety of legacy, technically improved existing systems, and new systems in a large enterprise. To ensure different applications can be integrated in the enterprise or in an inter-enterprise setting, a universal integration bus becomes necessary.

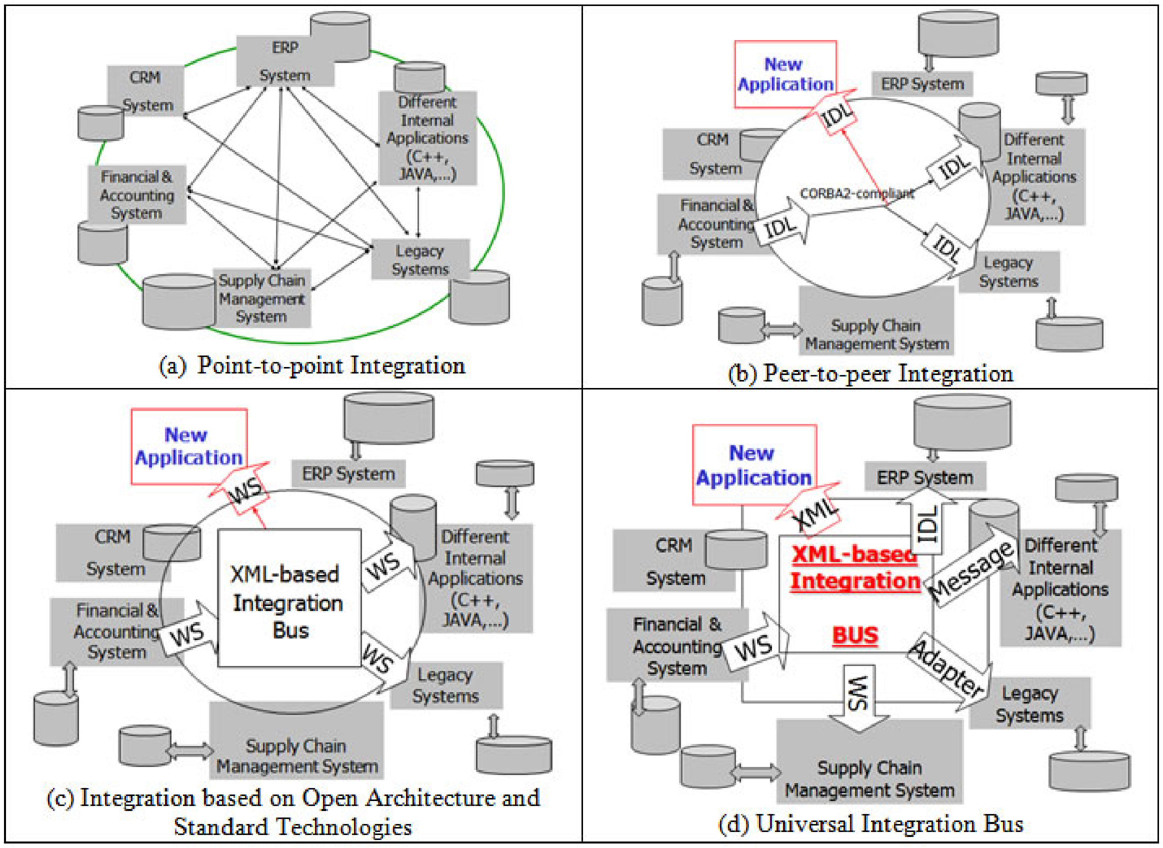
ERP System
CRM System
Different Internal Applications (C++, JAVA,...)
Financial & Accounting System
Legacy Systems
Supply Chain Management System
(a) Point-to-point Integration
New Application
IDL
CORBA2-compliant
(b) Peer-to-peer Integration
WS
XML-based Integration Bus
(c) Integration based on Open Architecture and Standard Technologies
XML
XML-based Integration BUS
Message
Adapter
(d) Universal Integration Bus

Figure 5.1 Evolution of Solutions to Integrating Heterogeneous Systems

General speaking, connectivity in the context of computer networking technology, refers to the use of computer networks to have computer systems connected to each other. Networking connectivity is the foundation of enterprise integration, enabling the potential for information resources to be shared between these computer systems and their final users. From the application integration perspective, connectivity should facilitate the obliteration of the gaps among applications to entail efficient and cost-effective cross-platform, high-speed, real-time data/functionality sharing. In Lesson 1, we overviewed the evolution of integration patterns by focusing on communications, indicating that it is the socket as the rudimentary software connectivity that forms the fundamental concept for all other types of inter-application integration patterns (Figure 5.2).

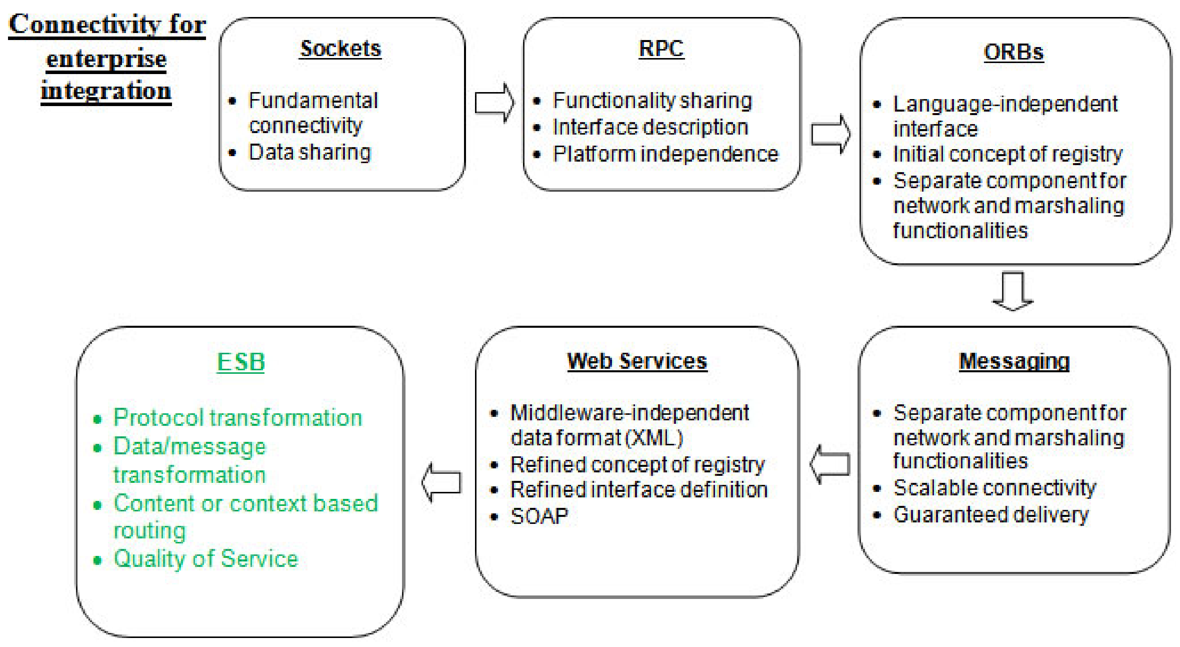
Connectivity for enterprise integration
Sockets
Fundamental connectivity
Data sharing
RPC
Functionality sharing
Interface description
Platform independence
ORBs
Language-independent interface
Initial concept of registry
Separate component for network and marshaling functionalities
Messaging
Separate component for network and marshaling functionalities
Scalable connectivity
Guaranteed delivery
Web Services
Middleware-independent data format (XML)
Refined concept of registry
Refined interface definition
SOAP
ESB
Protocol transformation
Data/message transformation
Content or context based routing
Quality of Service

Figure 5.2 Connectivity: the Enterprise Integration Perspective

Figure 5.2 illustrates the gradually increased supports and enhanced software connectivity focusing on ease of integration between applications [4].

- ***Socket and related raw communication protocols*** *(e.g., **TCP/IP, SOAP and FTP**) lack basic features (such as location transparency) that are necessary for developing real-world componentized distributed systems; the required development effort is only justified when performance is a primary concern.*
- **RPC** (Remote Procedure Call) and **ORBs** (Object Request Brokers) are suitable primarily for request/reply applications and have little or no support for event-driven or asynchronous communication; in addition, besides the basic request/reply pattern, there is little standardization of programming models for RPCs and ORBs.
- ***MOM*** *(Message Oriented Middleware) provides added flexibility over RPC/ORB with in-built support for event-driven as well as request/reply interaction between programs; however, MOM's require communicating modules to agree on topic or queue names, creating an underlying coupling that is not easy to remove without providing explicit support for message metadata, which MOMs lack. (It is possible to deploy complex service-oriented and event-driven applications over a MOM, but the lack of support for message metadata and a programming model for business components significantly complicates the implementation.)*
- ***Web Services*** *provide a standardized protocol and programming model for the development of business components; however, without a communications backbone and message metadata, Web Services degenerate into a relatively inflexible request/reply point-to-point infrastructure for service-oriented applications with no support for event-driven applications.*

Based on what we have discussed so far, the following concepts and features should be included and supported in an efficient and cost-effective connectivity-based middleware:

- Interface (i.e., IDL or the like)

- Peer-to-peer relationship between applications
- Stub/proxy/skeleton or adaptor in general, which shields the programmer from system and network calls
- Marshalling/unmarshalling of arguments for transmission over the network
- External data representation (XDR) which encodes data in a machine-independent format
- Synchronous and asynchronous interaction
- Language- and platform-independent programming

ESBs combine the strengths of all of the above discussed middleware protocols, with support for synchronous request/reply and asynchronous, event-driven communication and event-notification, business-component interfaces, industry standards such as SOAP/HTTP and Web Services, together with partial support for the management of business components [Table 5.1].

| **Technology** | **Socket** | **RPC** | **CORBA, DOCM** | **MOM** | **Web Services** | **ESB** |
|---|---|---|---|---|---|---|
| **Industrial support (popularity)** | Full support | Partial support | Partial support | Partial support | Full support | Full support |
| **Interface definition** | | Partial support | Full support | | Full support | Full support |
| **Service registration and discovery** | | Partial support | Partial support | | Full support | Full support |
| **Messaging and event notification** | | | | Full support | | Full support |
| **Quality of service** | | Partial support | Partial support | Full support | | Full support |
| **Content- and context-based routing** | | | | | | Full support |
| **Service interactions** | | | Partial support | Partial support | Full support | Full support |

Table 5.1 ESB Combines the Strengths of Pervious Middleware Technologies [4]

Apparently, ESBs or service buses emerge as our universal integration bus in enterprise integration. According to [2], an ESB is software middleware that provides fundamental services for more complex integrations. In essence, ESB enables flexible, adaptable, and scalable software connectivity to obliterate the heterogeneity among distributed back end services and applications and distributed heterogeneous front-end users and information consumers what universal middleware is really supposed to do: hide complexity, simplify access, allow developers to use generic, canonical forms of query, access and interaction, handling the complex details in the background. The key to ESB's appeal, and possibly also its future success, lies in its ability to support incremental service and application integration as driven by business requirements, not as governed by available technology. "ESB is not a new software product – it's a new way of looking at how to integrate applications, coordinate resources, and manipulate information." [3]

## The Conceptual Model of Enterprise Service Bus

As discussed above, by focusing on providing a solution to the various heterogeneity problems found in a large intra- and inter-enterprise setting, ESBs combine the strengths of all other middleware, in support of (Figure 5.3)

- Interface (i.e., IDL or the like)
- Peer-to-peer relationship between applications
- Stub/proxy/skeleton or adaptor in general, which shields the programmer from system and network calls
- Marshalling/unmarshalling of arguments for transmission over the network
- External data representation (XDR) which encodes data in a machine-independent format
- Synchronous and asynchronous interaction
- Language- and platform-independent
- Transformation
- Content- and context-based routing

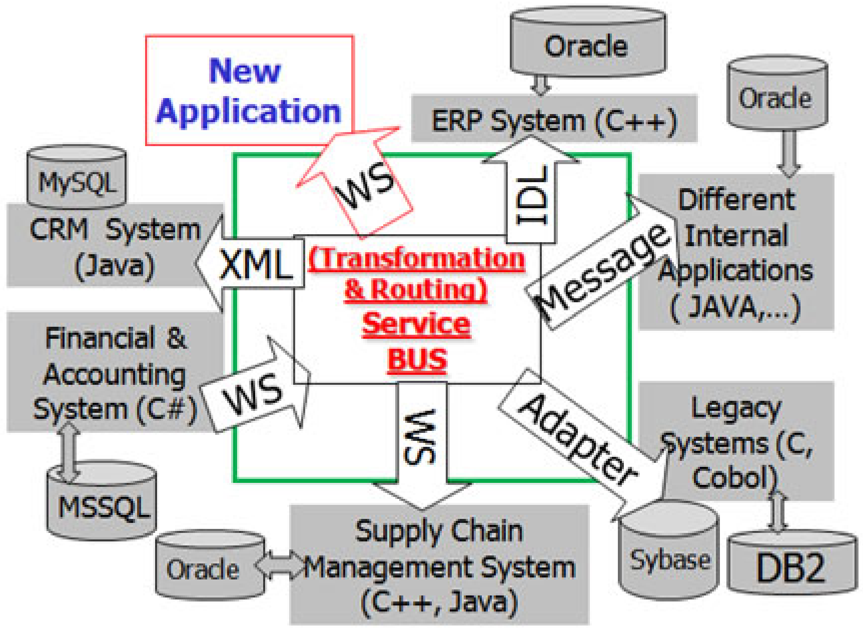
New Application
Oracle
ERP System (C++)
Oracle
MySQL
CRM System (Java)
WS
IDL
Different Internal Applications ( JAVA,...)
XML
(Transformation & Routing) Service BUS
Message
Financial & Accounting System (C#)
WS
WS
Adapter
Legacy Systems (C, Cobol)
MSSQL
Oracle
Supply Chain Management System (C++, Java)
Sybase
DB2

Figure 5.3 Service Bus: the Enterprise Integration Perspective

A more generic view of the services of an ESB is given in Figure 5.4, where necessary supports of integrating heterogeneous applications are radically and conceptually illustrated as a variety of adapters. Adapters are historically and widespread used in industry whenever applications requires connectivity components for communications.

- SOAP/HTTP: this is an essential support for Web Services compliant applications.
- XML/HTTP: this provides a necessary support for Web based applications when SOAP is not needed or supported.
- File Directory Service: this enables the traditional communication for certain legacy applications.
- SOAP/Messaging: when asynchronous messaging protocol makes more sense for large scale integration project, this meets the radical need.
- RMI/IIOP: given that there are many CORBA-based integration projects, this interface continues to fully leverage the existing investment.
- XML/SMTP/FTP: under certain circumstance, this communication supports data conveyed over mail service or file transfer protocols.
- XML/SQL: by providing this support, data sources of a typical application can be directly integrated with other applications if necessary.
- Proprietary adapters: many ESB solution providers also provide some proprietary adapters to allow more effective integrations with their proprietary technologies.

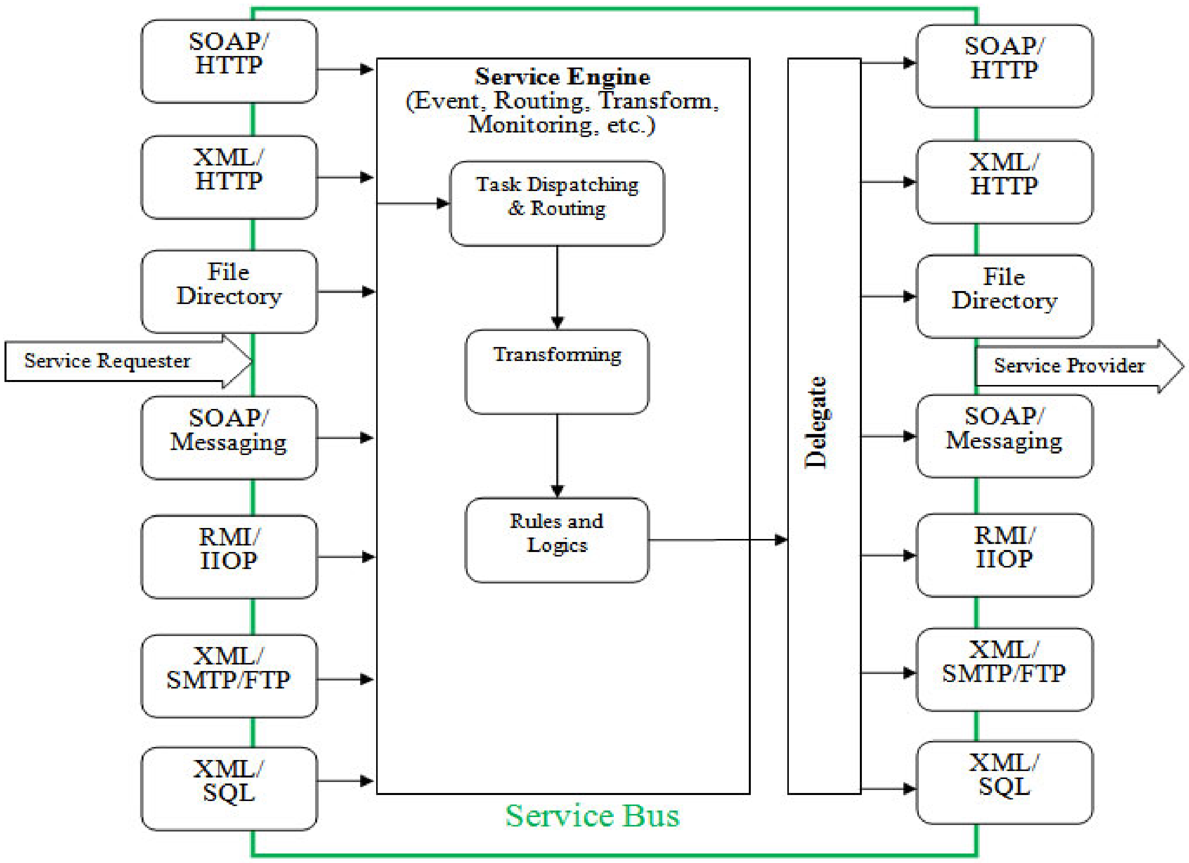

Figure 5.4 Conceptual Model of the services of an ESB

We now focus on three core features beyond what we have so far seen in other middleware.

Protocol Transformation (4 of 9)

## Protocol Transformation

It is very typical that different communication protocols are used in different applications. When they want to communicate with each other, protocol mismatch can arise, which we call a protocol heterogeneity issue in enterprise integration in our last lesson. Figure 5.5 clearly shows the complexity in terms of resolving the mismatch if they all have to communicate with each other from time to time to fulfill the business needs.

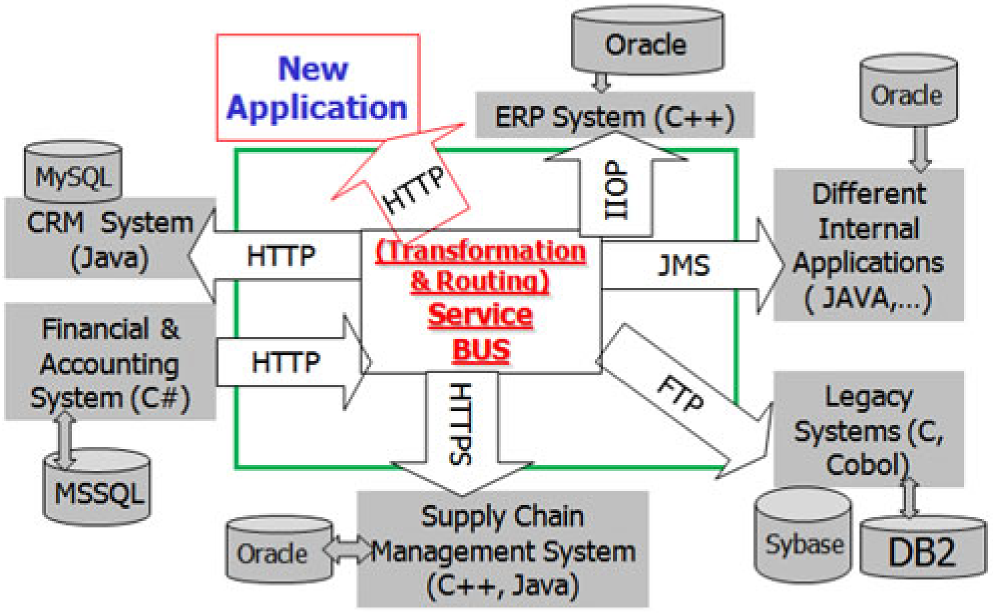
New Application
Oracle
ERP System (C++)
Oracle
MySQL
CRM System (Java)
HTTP
IIOP
Different Internal Applications ( JAVA,...)
HTTP
(Transformation & Routing) Service BUS
JMS
Financial & Accounting System (C#)
HTTP
FTP
HTTPS
Legacy Systems (C, Cobol)
MSSQL
Oracle
Supply Chain Management System (C++, Java)
Sybase
DB2

Figure 5.5 Protocol Transformation Support in ESB

To ensure this issue can be effectively resolved, an ESB includes a protocol transformation facility that essentially allows mismatch protocols application to share their data/functionality.

Data Transformation (5 of 9)

## Data Transformation

Similar to the variety of database systems, very few data transmitted between applications in a given enterprise would have the same schemas. Data used in each application might have been designed to meet different business domain needs. Very few enterprises would have an enterprise data model as businesses typically continue to evolve and change. Their information systems were also typically invested along with the growth of business and advancement of technologies. Therefore data take different formats to best fit in the circumstances as time goes. Figure 5.5 shows possible data and messages over different protocols would be used by different application.

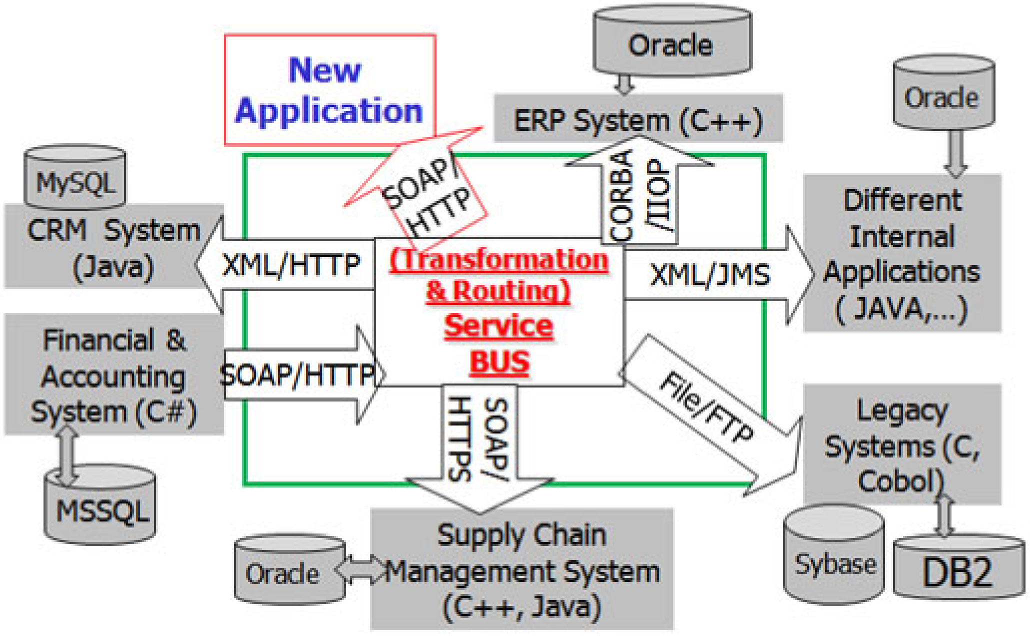
New Application
Oracle
ERP System (C++)
Oracle
MySQL
CRM System (Java)
SOAP/HTTP
CORBA /IIOP
Different Internal Applications ( JAVA,...)
XML/HTTP
(Transformation & Routing) Service BUS
XML/JMS
Financial & Accounting System (C#)
SOAP/HTTP
SOAP/ HTTPS
File/FTP
Legacy Systems (C, Cobol)
MSSQL
Oracle
Supply Chain Management System (C++, Java)
Sybase
DB2

Figure 5.6 Data Transformation Support in ESB

To ensure this data mismatch issue can be effectively resolved, an ESB includes a data transformation facility that essentially relays mismatched data between different applications through data transformation (Figure 5.6). This support is typically implemented using XLST discussed in Lesson 4.

Listing 5.1 shows a simple example. Data in a delimited format can be easily transformed (or reformatted) as an XML data. Listing 5.2 shows how the XML data based on one schema can be further transformed into another XML file based on another schema.

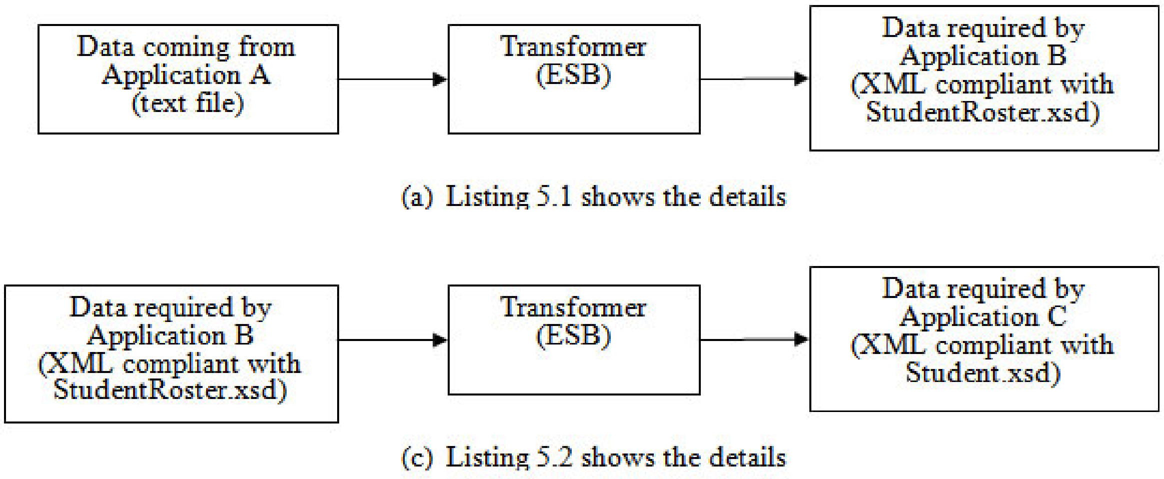
Data coming from Application A (text file)
Transformer (ESB)
Data required by Application B (XML compliant with StudentRoster.xsd)
(a) Listing 5.1 shows the details
Data required by Application B (XML compliant with StudentRoster.xsd)
Transformer (ESB)
Data required by Application C (XML compliant with Student.xsd)
(c) Listing 5.2 shows the details

Figure 5.7 Data Transformation Support Examples in ESB

## Listing 5.1 Sample of Data Transformation (Delimited file to XML)

Input: Delimiter based text file (semicolon as the delimiter)

1111;Bob Smith; 111-222-3333;bsmith@psu.edu;215-222-2356;123 Tulip Road, Ambler, PA 19002;321 College Ave., University Park,  PA 16802;John Smith;206;1111-206;1

Schema for Transformation: StudentRoster.xsd file

```
<?xml version="1.0" encoding="UTF-8"?>
  <!--W3C Schema Generated by BIE on Wed Mar 17 08:23:21 EST 2010-->
  <!--Document Root: root-->
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
       <xs:element name="root">
            <xs:complexType>
                 <xs:sequence>
                      <xs:element minOccurs="0" name="row" type="rowType"/>
                 </xs:sequence>
            </xs:complexType>
       </xs:element>
       <xs:complexType name="rowType">
            <xs:sequence>
                 <xs:element ref="StudID"/>
                 <xs:element ref="Name"/>
                 <xs:element ref="SSN"/>
                 <xs:element ref="EmailAddress"/>
                 <xs:element ref="HomePhone"/>
                 <xs:element ref="HomeAddress"/>
                 <xs:element ref="LocalAddress"/>
                 <xs:element ref="EmergencyContact"/>
                 <xs:element ref="ProgramID"/>
                 <xs:element ref="PaymentID"/>
                 <xs:element ref="AcademicStatus"/>
            </xs:sequence>
       </xs:complexType>
       <xs:element name="StudID" type="xs:string"/>
       <xs:element name="Name" type="xs:string"/>
       <xs:element name="SSN" type="xs:string"/>
       <xs:element name="EmailAddress" type="xs:string"/>
       <xs:element name="HomePhone" type="xs:string"/>
       <xs:element name="HomeAddress" type="xs:string"/>
       <xs:element name="LocalAddress" type="xs:string"/>
       <xs:element name="EmergencyContact" type="xs:string"/>
       <xs:element name="ProgramID" type="xs:string"/>
       <xs:element name="PaymentID" type="xs:string"/>
       <xs:element name="AcademicStatus" type="xs:string"/>
  </xs:schema>

Output: XML file

<?xml version="1.0" encoding="UTF-8"?>
  <root>
       <row>
            <StudID>1111</StudID>
            <Name>Bob Smith</Name>
            <SSN> 111-222-3333</SSN>
            <EmailAddress> bsmith@psu.edu</EmailAddress>
            <HomePhone>215-222-2356</HomePhone>
            <HomeAddress>123 Tulip Road, Ambler, PA 19002</HomeAddress>
            <LocalAddress> 321 College Ave., University Park, PA 16802</LocalAddress>
            <EmergencyContact>John Smith</EmergencyContact>
            <ProgramID>206</ProgramID>
            <PaymentID>1111-206</PaymentID>
            <AcademicStatus>1</ AcademicStatus >
        </row>
    </root>
```

## Listing 5.2 Sample of Data Transformation (XLST: StudentRoster.xsd to Another.xsd)

Output: XML Another.xsd file

```
<?xml version="1.0" encoding="UTF-8"?>
  <!--W3C Schema Generated by BIE on Sun Mar 13 10:29:11 EST 2005-->
  <!--Document Root: root-->
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
       <xs:element name="root">
            <xs:complexType>
                 <xs:sequence>
                      <xs:element minOccurs="0" name="row" type="rowType"/>
                 </xs:sequence>
            </xs:complexType>
       </xs:element>
       <xs:complexType name="rowType">
            <xs:sequence>
                 <xs:element ref="Name"/>
                 <xs:element ref="ID"/>
                 <xs:element ref="Email"/>
                 <xs:element ref="Phone"/>
            </xs:sequence>
       </xs:complexType>
       <xs:element name="Name" type="xs:string"/>
       <xs:element name="ID" type="xs:string"/>
       <xs:element name="Email" type="xs:string"/>
       <xs:element name="Phone" type="xs:string"/>
  </xs:schema>

  Transformation Stylesheet: xsd file

<?xml version="1.0" encoding="UTF-8"?>
  <xsl:stylesheet exclude-result-prefixes="" version="1.0" xmlns:lxslt="http://xml.apache.org/xs
  <xsl:output indent="yes" method="xml"/>
  <xsl:template match="/">
  <xsl:apply-templates select="root"/>
  </xsl:template>
  <xsl:template match="text( )"/>
  <xsl:template match="root">
  <xsl:element name="root">
  <xsl:apply-templates select="*"/>
  </xsl:element>
  </xsl:template>
  <xsl:template match="root/row">
  <xsl:element name="row">
  <xsl:value-of select="child::text()"/>
  <xsl:element name="Name">
  <xsl:value-of select="Name"/>
  </xsl:element>
  <xsl:element name="ID">
  <xsl:value-of select="StudID"/>
  </xsl:element>
  <xsl:element name="Road"/>
  <xsl:element name="City"/>
  <xsl:element name="State"/>
  <xsl:element name="Zip"/>
  <xsl:element name="Email">
```

```
<xsl:value-of select="EmailAddress"/>
</xsl:element>
<xsl:element name="Phone">
<xsl:value-of select="HomePhone"/>
</xsl:element>
</xsl:element>
</xsl:template>
</xsl:stylesheet>

Output: Student XML file

<?xml version="1.0" encoding="UTF-8"?>
  <root>
       <row>
            <Name>Bob Smith</Name>
            <ID> 1111</ID>
            <Email> bsmith@psu.edu</Email>
            <Phone>215-222-2356</Phone>
       </row>
  </root>
```

## Routing and Scalable Connectivity

As we discussed in Lesson 1, in addition to the complexity and heterogeneity issues, the integration practitioners commonly and frequently confront with scalability and agility issues as business grows:

- **Scalability:** An integrated system has to be scalable in terms of continuously delivering the expected information services when the system and supported business grow with time.
- **Agility:** An integrated system has to be flexible, responsive, and adaptable as market demands fluctuate and technologies advance.

Assume there is an order fulfillment service. When the order is ready for delivery, it might be executed differently. The selection can be based on the type of order, for example in terms of size. If it is a bulky order, the delivery service will be UPS (assume it is “a” in Figure 5.8); otherwise, a USPS service (assume it is “b” in Figure 5.7) will be requested.

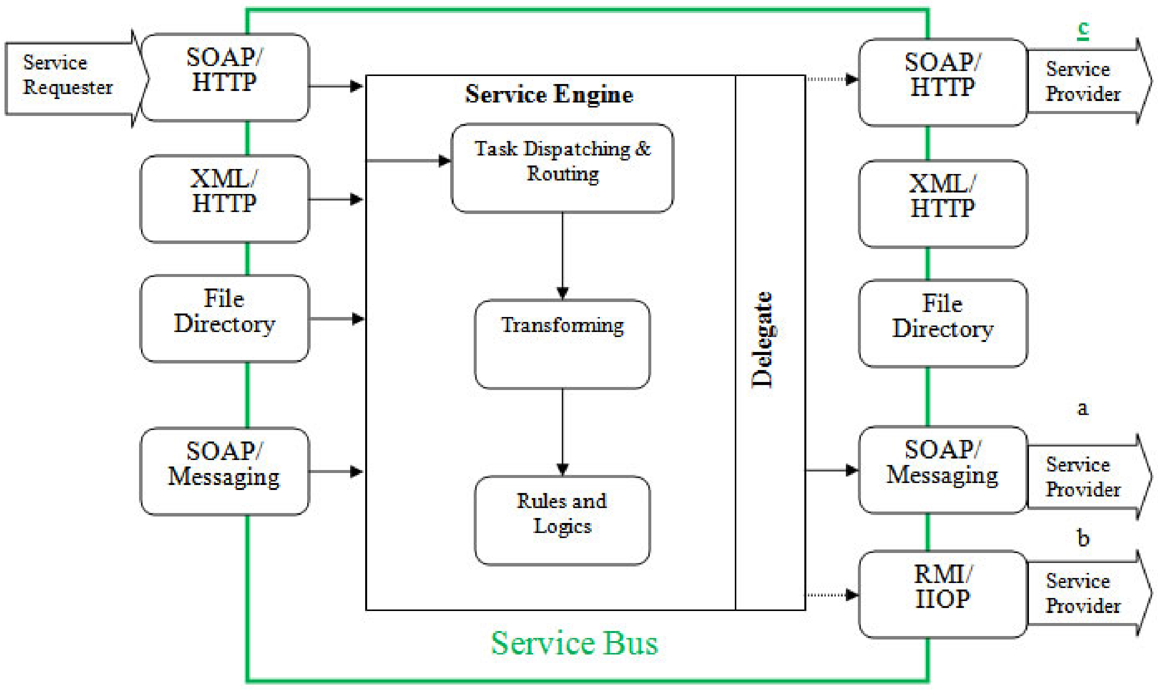

Figure 5.8 Scalable and Agile Service Examples in ESB

Assume a year later, the company signs up another delivery service provider – FedEx. When an ESB is used, this kind of expanded support can be done quickly and easily. A minor change in the routing logic allows the third service provider to be fully integrated (assume it is “c” in Figure 5.7).

## Types of ESB

The first type of ESB essentially relies on ORBs. Typically, this type of ESB is easy to use and relatively inexpensive. As it might come with application servers or be evolved from conventional ORBs, the enabled supports and services are limited.

The second type of ESB is based on asynchronous messaging service systems. Three main benefits are provided in this type of ESB:

- It is highly scalable
- It can be used to integrate a more diverse set of applications
- It guarantees the delivery of messages among integrated applications

## ESB Products

There are many ESB solutions in the market [2]. IBM as one of the major ESB vendors promotes ESB as an effective way to meet the challenges of integrating large-scale and heterogeneous applications. IBM’s ESB provides a single and unified architecture built around IBM WebSphere [3], which can:

- *Distribute information across an enterprise quickly and easily*
- *Mask differences among underlying platforms, software architectures, and network protocols*
- *Ensure information delivery even when some systems or networks may go off-line from time to time.*
- *Re-route, log, and enrich information without requiring applications to be rewritten*
- *Provide incremental solution implementations so all enterprise services and applications need not change immediately or all at once*

| **Vendor** | **Name** |
|---|---|
| **Sun Microsystems** | GlassFish ESB [4] |
| **TIBCO** | Tibco Enterprise Messaging Service (EMS) [5] |
| **IBM** | Websphere Enterprise Service Bus [3] |
| **Oracle** | Oracle ESB [6] |
| **Fiorano Software** | Fiorano ESB [7] |
| **Microsoft** | Biztalk Server [8] |
| **Neudesic** | Neuron ESB [9] |
| **Progress Software** | Progress Sonic ESB [10] |
| **AdroitLogic** | UltraESB [11] |
| **JBoss** | JBoss ESB [12] |
| **Software AG** | Webmethods ESB Platform [13] |
| **Mule Soft** | Mule ESB [14] |
| **Apache** | ServiceMix [15] |
| **Redberri** | Redberri Enterprise [16] |

Table 5.2 List of ESB Products[2]

## Examples of ESBs

TIBCO Enterprise Message Service (the product information fully copied from [5])

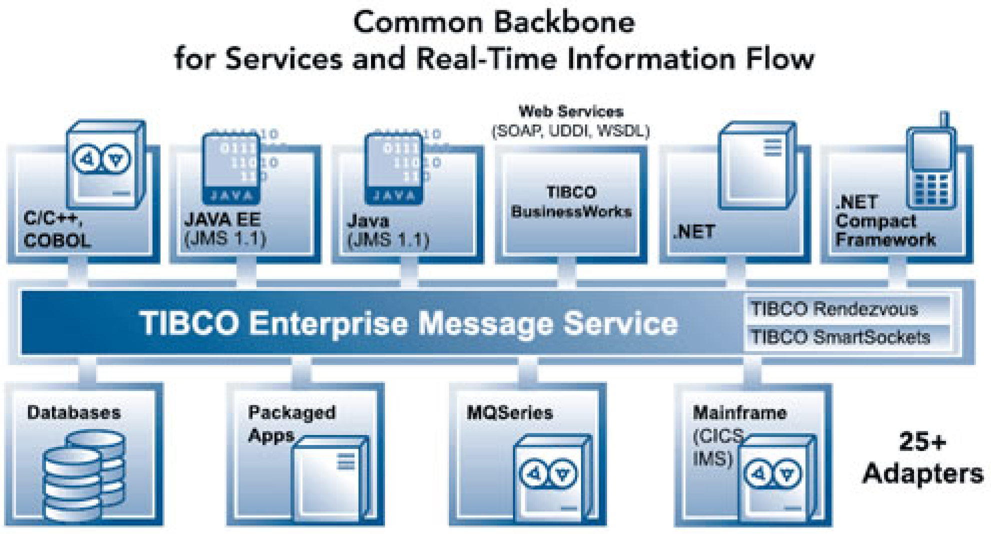

Figure 5.9 TIBCO Message Service Bus
(Copyright http://www.tibco.com/assets/bltc99371625c88f3d5/ds-ems.pdf (http://www.tibco.com/assets/bltc99371625c88f3d5/ds-ems.pdf) )

Key Features

- *A distributed and reliable architecture, with support for load-balancing, routing, and fault tolerant configurations that together remove single points of failure. By using TIBCO's messaging solution, companies have been able to reliably support over 50,000 messages per second and achieve 99.999% uptime.*
- *One of the broadest sets of messaging semantics; supports request/reply and publish/subscribe interactions, synchronous and asynchronous messaging, multicast deployments and different levels of reliable messaging, including support for externally managed XA-compliant transactions. These capabilities enable developers and administrators to support different types of service protocols on the same platform and fine-tune qualities of service for even the most demanding applications.*
- *A distributed message bus with multi-protocol support for Java Message Service (JMS), TIBCO Rendezvous®, and TIBCO SmartSockets™. TIBCO supports many of the leading open standards, including web services, and provides adapters for third-party applications and infrastructure including IBM MQSeries. This makes it possible for companies to reuse their existing investments and focus on delivering new functionality.*
- *Native support for a broad range of development technologies and platforms including Java EE and Microsoft .NET on servers, desktops, and mobile devices, and C and COBOL on the mainframe.*
- *Support for security standards with the administrative control needed to deliver high performance and secure messaging solutions. It has full SSL support for client-to-server and server-to-server connectivity and plug-in security capability for custom-built authentication (JAAS) and authorization.*
- *Operational flexibility through integration with third-party relational databases.*
- *Built-in monitoring and management capabilities that provide detailed administrative functions and statistics and support automation through an administrative API or command-line shell. Companies can also leverage TIBCO's common monitoring and management framework for top-down, end-to-end distributed monitoring and management of TIBCO and non-TIBCO products.*

- *Custom transport channel for enabling TIBCO Enterprise Message Service™ as the message transport for Microsoft .NET applications on the Microsoft Windows Communications Framework.*

Apache ServiceMix (the product information fully copied from [15])
Apache ServiceMix is an open source ESB (Enterprise Service Bus) that combines the functionality of a Service Oriented Architecture (SOA) and an Event Driven Architecture (EDA) to create an agile, enterprise ESB.

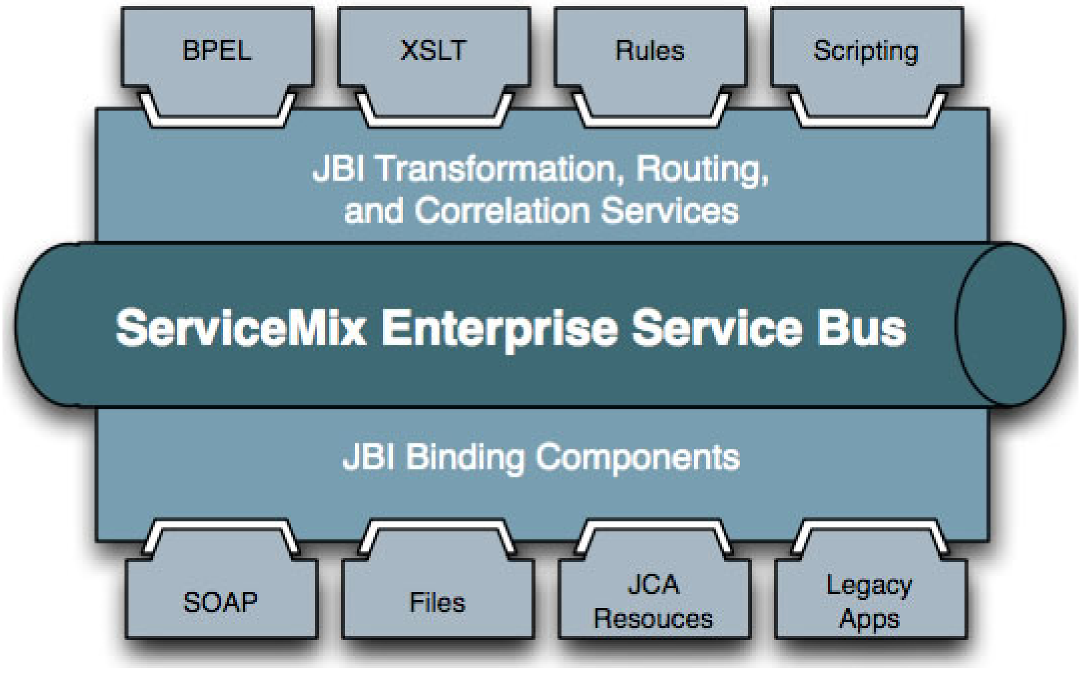
BPEL
XSLT
Rules
Scripting
JBI Transformation, Routing, and Correlation Services
ServiceMix Enterprise Service Bus
JBI Binding Components
SOAP
Files
JCA Resouces
Legacy Apps

Figure 5.9 Apache's ServiceMix (Copyright © http://servicemix.apache.org/home.html (http://servicemix.apache.org/home.html) )

References (9 of 9)

## References

[1] Roshen, W. 2009. SOA-based Enterprise Integration: A Step-by-Step Guide to Services-based Application Integration. McGraw-Hill, New York, USA.

[2] Wikipedia:

- ESB: http://en.wikipedia.org/wiki/Enterprise_service_bus,
- UDDI: http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration.

[3] IBM: http://www.ibm.com

[4] GlassFish Enterprise Service Bus (ESB):
http://www.oracle.com/technetwork/documentation/legacy-glassfish-esb-193461.html

[5] TIBCO Enterprise Message Service : http://www.tibco.com/products/automation/application-integration/activematrix-businessworks/enterprise-service-bus

[6] Oracle ESB: http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html

[7]Fiorano: www.fiorano.com

[8] BizTalk Server: http://www.microsoft.com/biztalk/en/us/default.aspx

[9] Neuron ESB: http://www.neuronesb.com/

[10] SONIC ESB: http://www.aurea.com/products/enterprise-service-bus

[11] UltraESB: http://adroitlogic.org/

[12] Joss ESB: http://www.jboss.org/jbossesb

[13] webMethods ESB

[14] Mule ESB: http://www.mulesoft.org/

[15] Apache ServiceMix: http://servicemix.apache.org/home.html

[16] Redberri Enterprise: http://www.redberri.com/products_redberri_enterprise.htm