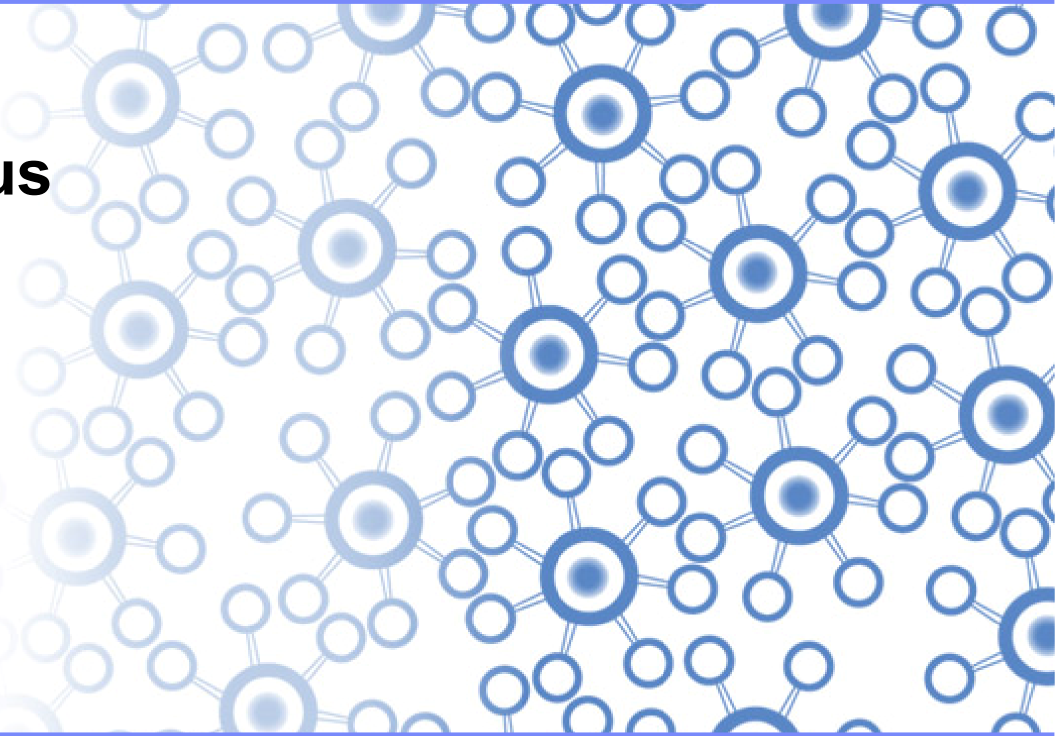
The Enterprise Service Bus

WebSphere ESB



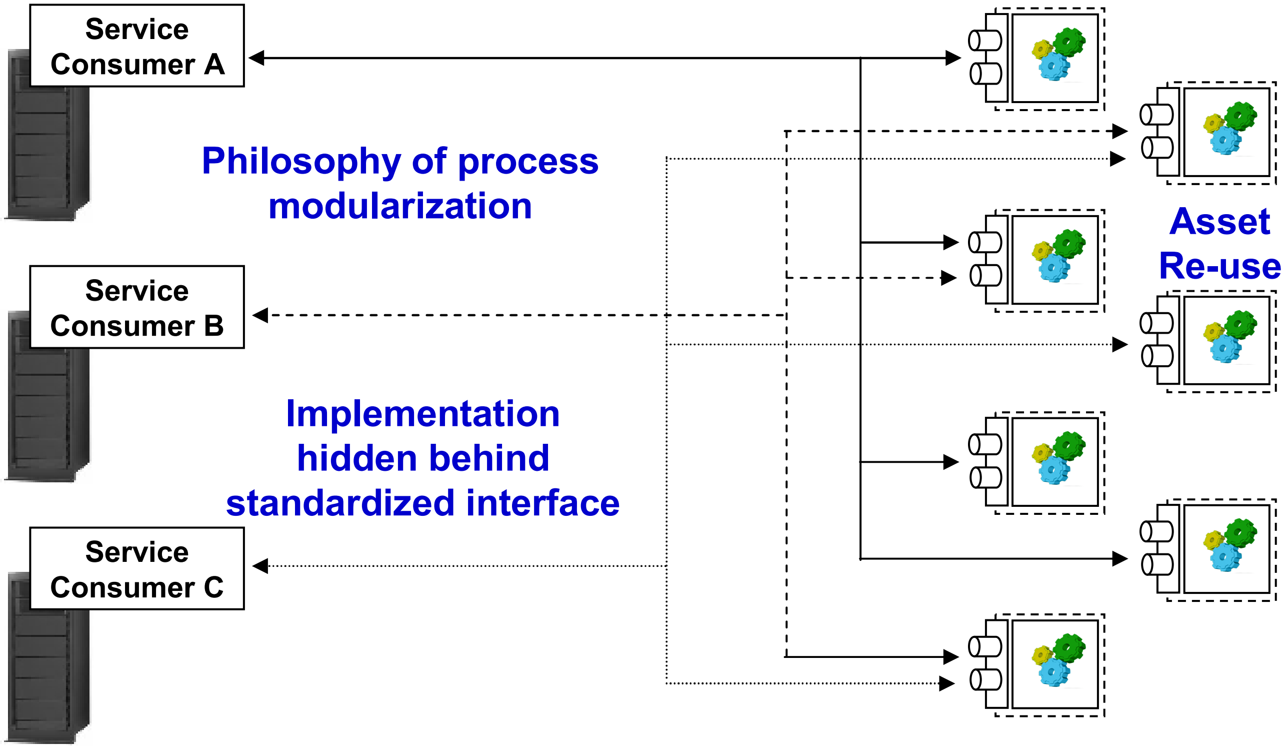
IMPACT 2009

Agenda

- WebSphere Enterprise Service Bus Overview
- Ball State University Case Study
- Q & A

The Value of Loosely-Coupled Services

It allows the flexible re-use of assets. Construct business processes by using these services in the order needed:



Service Consumer A

Service Consumer B

Service Consumer C

Philosophy of process modularization

Implementation hidden behind standardized interface

Asset Re-use

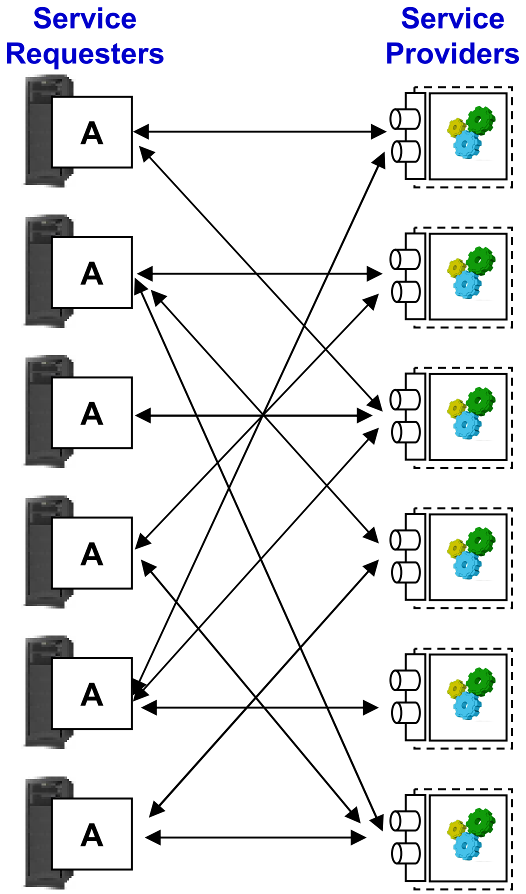
This is good -- it achieves at least a few key things:

- Establishment of a mindset towards reusable, service-oriented design
- The creation of an inventory of reusable service assets
- The hiding of complex implementation details behind a standardized interface

But this *by itself* does not solve the complexity issue …

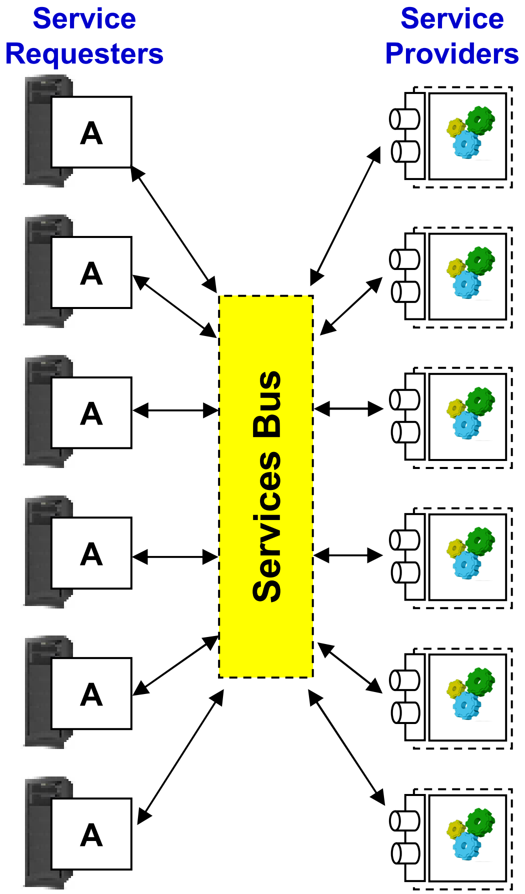
Still Point-to-Point

Services are only the first step. If only a few, then easy to manage. But when the number increases, the complexity increases as well. We need to address the point-to-point nature.



This picture is not unlike the picture we painted earlier.

Our goal is to get to a picture like this, where the connections are handled by a kind of "any-to-any" system

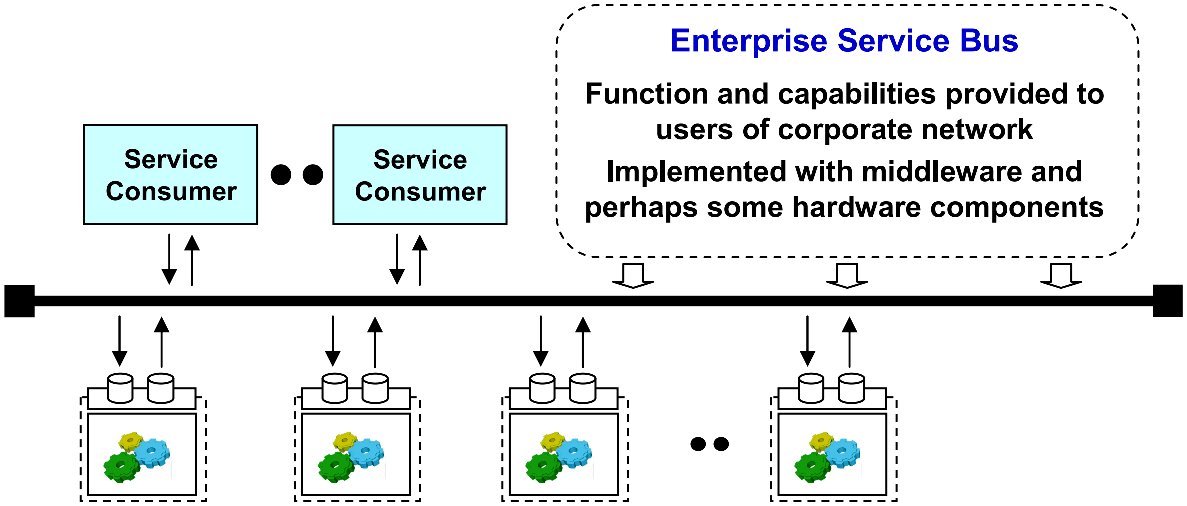


What is it? And what does it provide? …

The ESB -- What It Is and What it Provides

Here's the picture from our Introduction presentation. Key point is that the ESB is function mapped on top your existing network infrastructure.

- **Messaging services**
  Support different message types; content-based routing; guarantee message delivery.
- **Management services**
  Monitor performance; enforce SLA
- **Interface services**
  Support web services standards and provider "adapters" for non-standard interfaces
- **Mediation services**
  Transform messages between formats.
- **Security services**
  Encryption, authentication, authorization



Service Consumer

Service Consumer

Enterprise Service Bus

Function and capabilities provided to users of corporate network

Implemented with middleware and perhaps some hardware components

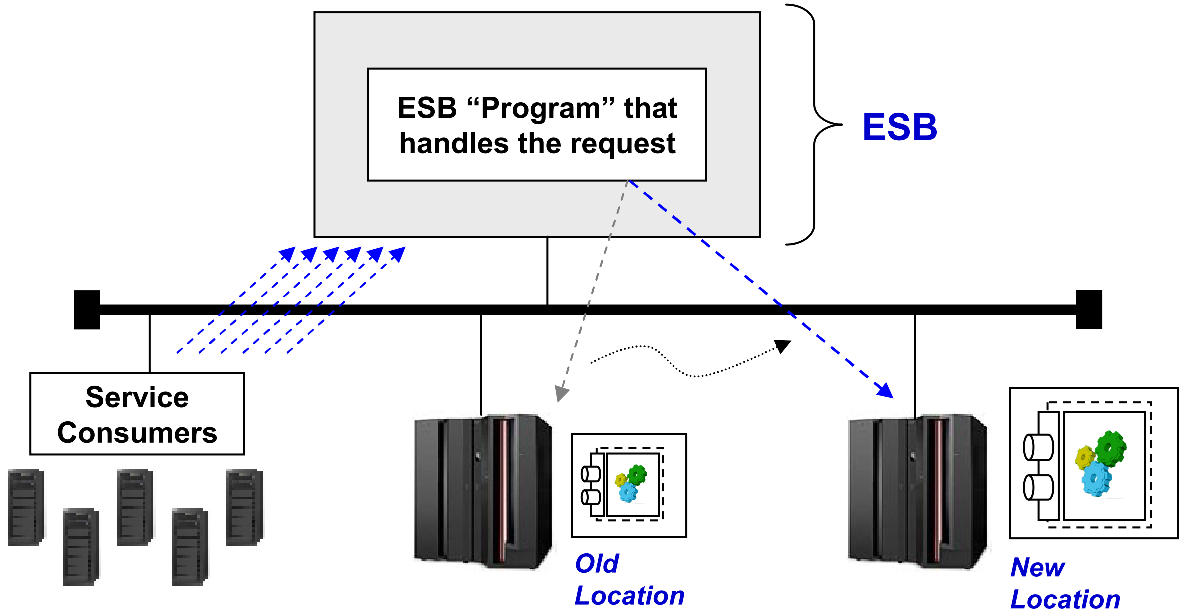
An IP network built on Ethernet does not provide any of this.
This is function *above and beyond* basic networking

Even with this it's a bit of a slippery concept. Let's take a look at one more "concept picture" then introduce IBM's ESB products.

A common initial use of the ESB …

Ability to "Alias" the Service -- Improved Flexibility

Even if we don't do fancy message transformation, simple routing through ESB provides the benefit of improving flexibility:



ESB "Program" that handles the request

ESB

Service Consumers

Old Location

New Location

- Service users continue to go to same ESB location
- Definition inside ESB modified to point to new service location
- Change hidden from service users

Admittedly a simple example. And dynamic retrieval of an updated WSDL in a Web Services world achieves the same result. But what about application connections that don't use Web Services? Or, what if client has cached copy of WSDL and it's not updated?

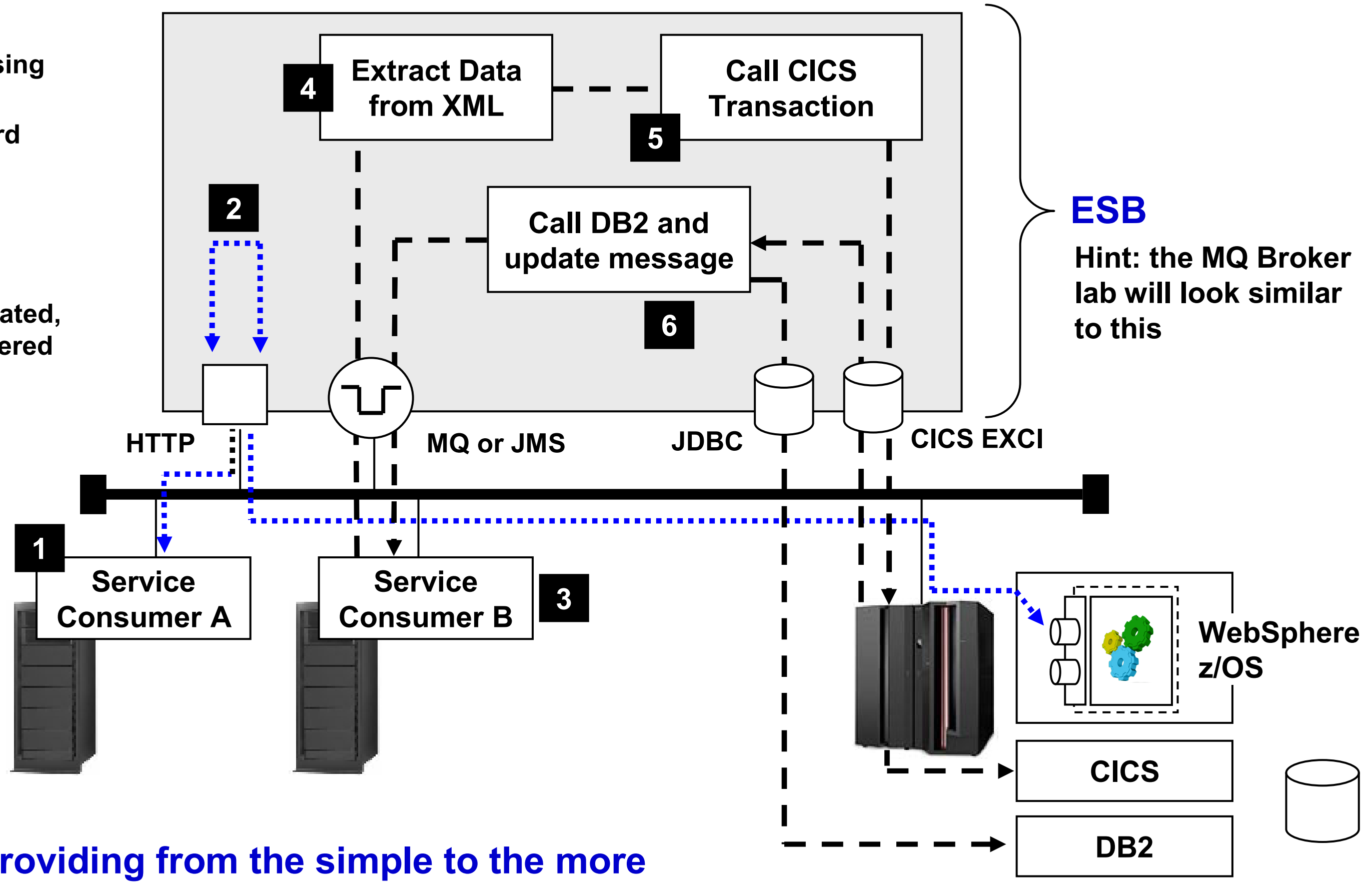
Even simple implementations of ESB can serve important role as intermediary that hides service location details behind common entry point. Then it can be expanded as needed to do additional protocols, protocol remapping, message transformation, etc.

More complex examples …

The ESB as a “Black Box” for Service Consumers and Producers

Here’s a conceptual diagram showing ESB handling two different request flows. One is simple pass-through; other requires some format transformation.

1. Consumer A has standard SOAP/HTTP request
2. ESB provides simple addressing and routing for A’s request
3. Consumer B has non-standard XML over MQ or JMS
4. Data extracted from XML, COMMAREA formatted
5. CICS tran called
6. DB2 called and message updated, formatted into XML and delivered back to client



Hint: the MQ Broker lab will look similar to this

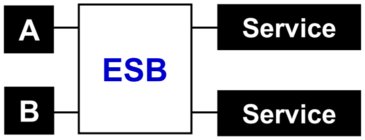
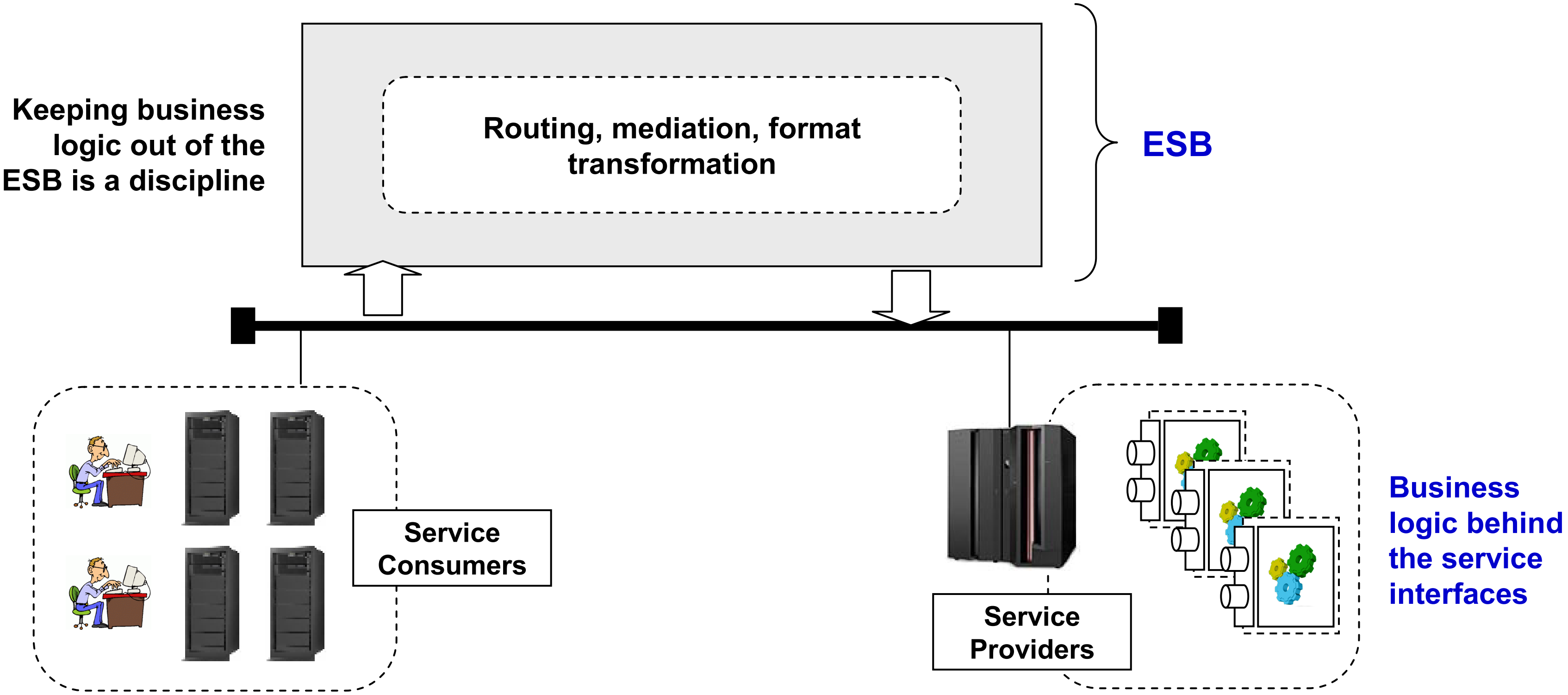


Illustration shows ESB providing from the simple to the more complex. Communication flows across existing network. Whole thing is “transparent” to consumers or providers of services

The Golden Rule …

No Business Logic in the ESB!

The ESB has within it a computational environment. It would be *possible* to code business logic there, but it is strongly recommended you do not:

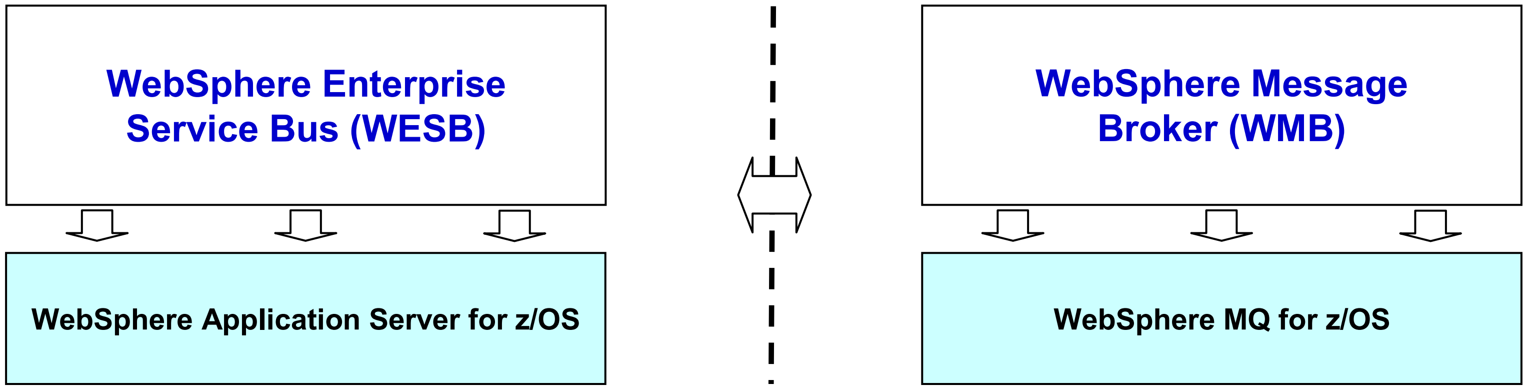

Keeping business logic out of the ESB is a discipline
Routing, mediation, format transformation
ESB
Service Consumers
Service Providers
Business logic behind the service interfaces

Why? Because business logic in the ESB starts to break down the "service oriented" approach. It's a trend back in the direction of tightly integrated and inflexible.

IBM Product Implementations …

IBM’s Implementation of ESB into Product

Ultimately we need to get to the point where we can point to something and say “There, that’s IBM’s ESB on z/OS.” Here they are:



WebSphere Enterprise Service Bus (WESB)

WebSphere Application Server for z/OS

WebSphere Message Broker (WMB)

WebSphere MQ for z/OS

They *can* interoperate

- Built on the proven J2EE platform of WebSphere Application Server
- Focus is standards-based access and J2EE connector support to backend systems

- Built on the proven messaging platform of MQ
- Standards-based access *and* non-standard through a host of connectivity options

Key Points:

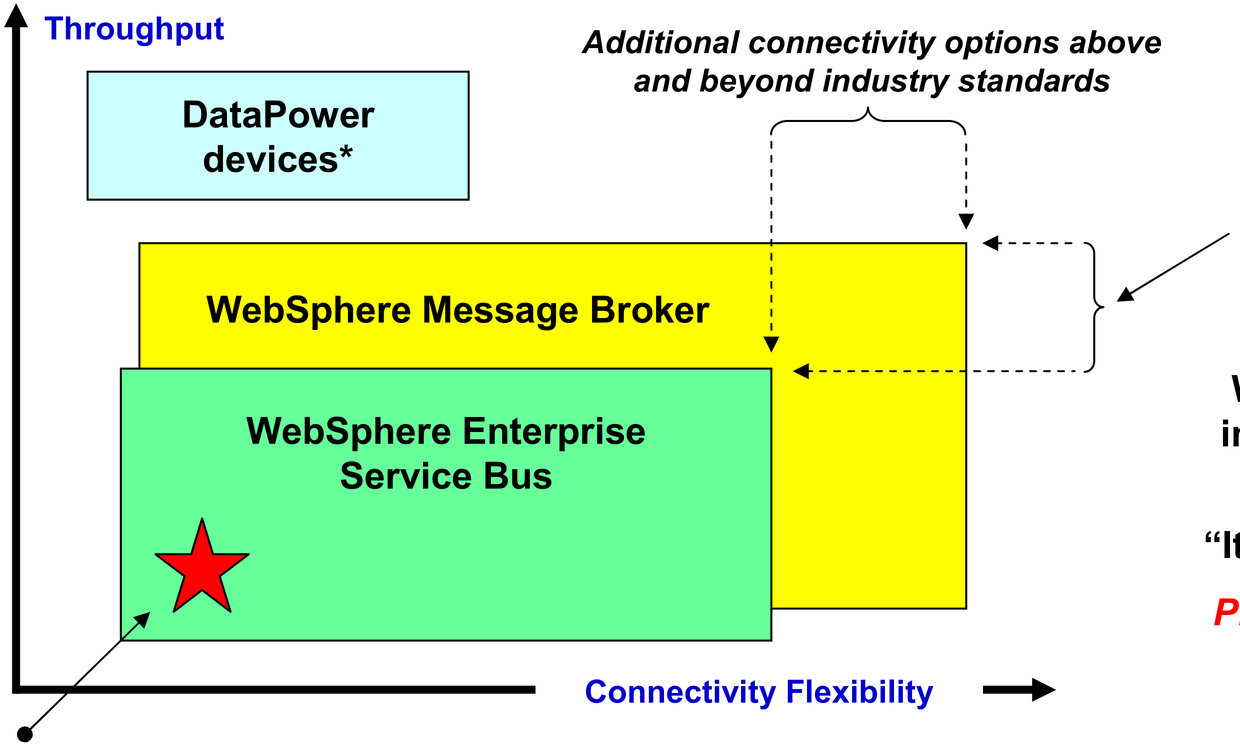
- Shows how ESB function is implemented in middleware (makes “real” the concept of ESB)
- There is the question of “Which Should I Consider?” which we’ll cover next

  It’s going to come down to function needed and degree of non-standards access. WESB focuses on industry standard only.

Product Positioning …

Approximate Product Positioning

Rough picture of how WESB and WMB (and DataPower) relate to one another:


Throughput
DataPower devices*
Additional connectivity options above and beyond industry standards
WebSphere Message Broker
WebSphere Enterprise Service Bus
Connectivity Flexibility

Non-Java (native) programming implementation

Which you should use involves understanding your needs.

"It depends" applies here

Please do not read more into this chart than it intended!

It's intended to be a way of understanding the *approximate* positioning of the products

What's not represented here is the degree of industry standard adoption by WESB: SCA, SDO, J2EE, JCA, etc. If that's important in your decision model, then factor that separately from what's shown here.

DataPower = rackmount hardware devices
`www.ibm.com/software/integration/datapower/`

- Integration Appliance XI50 -- wirespeed message transformation
- XML Security Gateway XS40 -- WS-Security, XML encryption, XML validation
- XML Accelerator XA35 -- processing XML, XSD, XPath and XSLT at wirespeed

WESB …

Ball State University: Case Studies

Ball State University

- **20,000 Students with a 660-acre campus.**

- **Accredited by the Higher Learning Commission of the North Central Associa of Colleges and Schools.**

- **170 bachelor's, 88 master's, 16 doctoral programs. Specialist programs include:**
  - Applied Sciences and Technology
  - Teachers College
  - Communication, Information, and Media

- **$25 million in external grants for faculty research and education projects. About 70 percent of the submitted proposals are funded**

- **Ranked as the nation's top wireless campus by Intel in 2005.**

- **Alumni of note… David Letterman, Jim Davis (Garfield creator), John Schnatter (Papa John's Pizza)**

Ball State University - Case Study 1

- Lost revenue due to incorrect address in billing system

Industry: *Education*
URL: *http://cms.bsu.edu/*

"SOA has been such a gift to us. It enables us to embrace a new technology that provides services at a level that we couldn't even imagine before."
–Dr. O'Neal Smitherman

Ball State University SOA & ESB Summary

Ball State University bridges disparate systems and solves key administrative issue with IBM SOA solution.

CHALLENGES

- Coordinate 40 name and address systems to streamline administrative processes and ensure information integrity for users

SOLUTION

- SOA with Enterprise Service Bus to connect siloed applications without hand-coding individual API calls

BENEFITS

- Ability to develop and implement services in an SOA environment for resolving name and address discrepancies in 10 months, as opposed to several years for hand-coding individual application connections
- Confidence that IBM solution can lead to wider use of SOA to further streamline administrative business processes
- Services created for this project can be reused in later SOA efforts

The Mailing Address Pain Point

- **The BSU President's Office received a heated complaint from a recent graduate who was experiencing difficulty getting a particular BSU department to mail him his bill to pay (so he could then get his diploma), yet he received a Alumni Association request to donate money**

- **The billing department did not have the student's current address (post graduation) to send the bill**

- **The Alumni Association did have the current address (purchased from a 3rd party marketing company)**

The IBM System z Accelerated SOA Workshop

- Introduces the client to IBM technology and solidifies SOA terminology with all parties
- Brings together cross brand specialists to discuss how to accurately architect the solution
- Deliverable is a pilot SOW
- The 20+ IBMers and Clients signed off on the POC
- The IBM team utilized existing CICS code on the mainframe as the central repository and created services that would call into the system.
- The BSU team would use existing skill in ‘dot net’ to create a new Web Front End

The Bigger Problem & Findings

- **16 departments within BSU that utilize a student's 'Name & Address'**
  - For the most part they did not share 'updated' student information when produced by a student

- **No master central repository for a student's 'Name & Address'**
  - addresses varied among departments

- **The existing architecture/platforms determined the business process and the operating rules for more than just managing a a student's address**

- **Distributed architecture created overlap, duplication, confusion, and lack of integration**

- **Didn't know how to successfully deploy *a system* that tries to connect systems without cutting off other existing systems**

Why SOA and ESB at BSU Made Sense

- Existing architecture / platforms that determined the business process and the operating rules can still be used and reused
- Vintage and legacy software applications in production are kept and reused
- Existing business process are either maintained or improved based upon by BSU and not an outside SW vendor
- Costs are reduced
- Technical skill set already exists with in University Computing Services
- BSU has an opportunity to be an SOA leader in Higher Education
- ESB connects siloed applications without hand-coding individual API calls

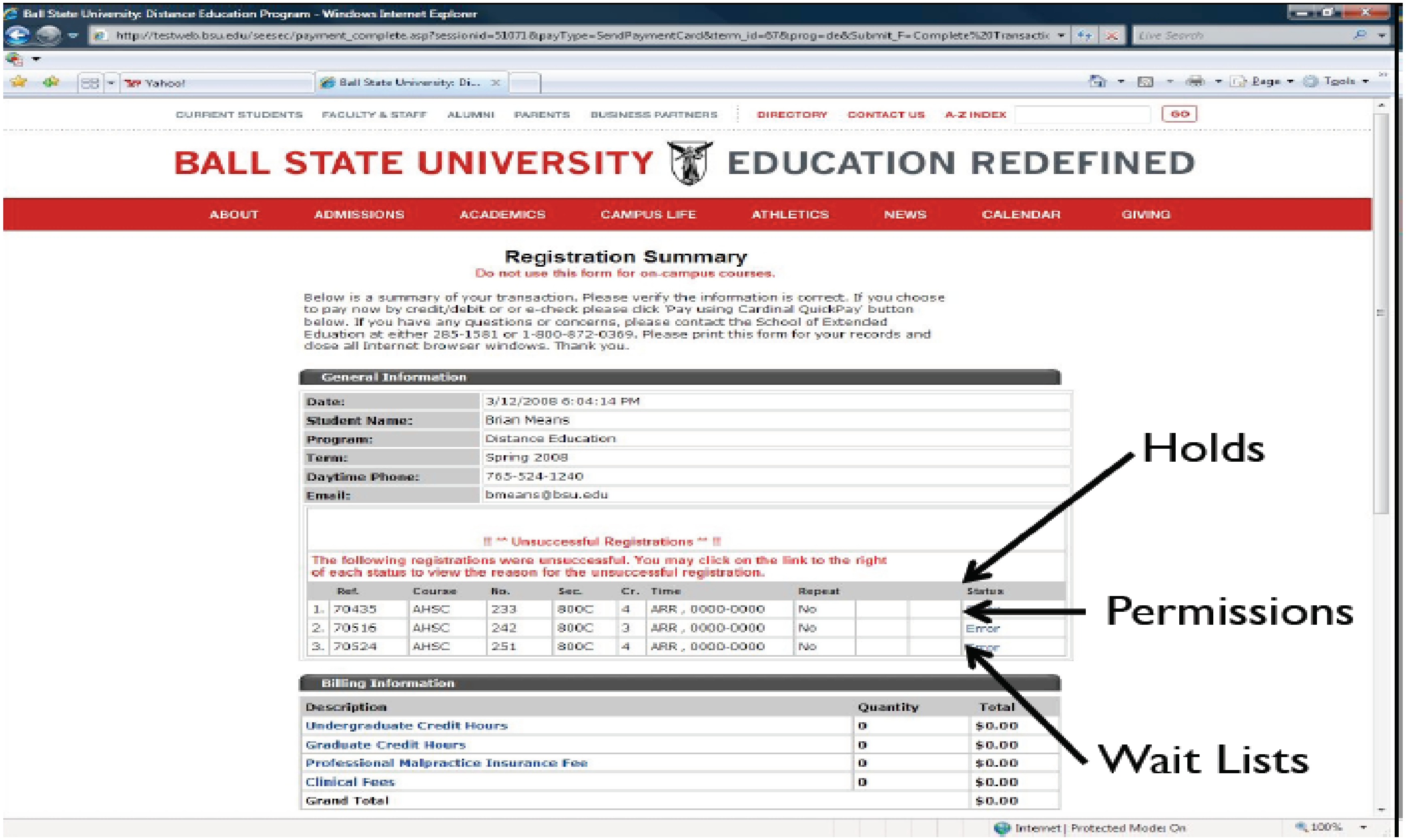
BSU's benefits from SOA and teaming with IBM

- **Any BSU application that utilizes a Student's Name & Address now updates all of the others applications in REAL TIME**
- **Information on the student is now standardized across all of the university's departments**
- **Leveraged existing Microsoft 'dot net' skills with University Computing Services dept.**
- **IBM was willing to "invest" in BSU's success**
  - Workshop is Free
  - Utilize highly skilled Pre Sales and Lab Services
  - Mentoring was a key element of the IBM SOW
  - Delivered POC, cost was low (relative to Microsoft)
  - BSU becomes IBM's cornerstone SOA account in Higher Ed
- **Generate recognition within the Higher Education Industry**

Ball State University - Case Study 2

- **Lost revenue due to student inability to register for classes**

BSU Extended Education Integration Project


Ball State University: Distance Education Program - Windows Internet Explorer
http://testweb.bsu.edu/seesec/payment_complete.asp?sessionid=51071&payType=SendPaymentCard&term_id=67&prog=de&Submit_F=Complete%20Transactic
Live Search
Yahoo!
Ball State University: Di...
Page
Tools
CURRENT STUDENTS
FACULTY & STAFF
ALUMNI
PARENTS
BUSINESS PARTNERS
DIRECTORY
CONTACT US
A-Z INDEX
GO
BALL STATE UNIVERSITY
EDUCATION REDEFINED
ABOUT
ADMISSIONS
ACADEMICS
CAMPUS LIFE
ATHLETICS
NEWS
CALENDAR
GIVING
Internet | Protected Mode: On
100%

Registration Summary

Do not use this form for on-campus courses.

Below is a summary of your transaction. Please verify the information is correct. If you choose to pay now by credit/debit or or e-check please click 'Pay using Cardinal QuickPay' button below. If you have any questions or concerns, please contact the School of Extended Eduation at either 285-1581 or 1-800-872-0369. Please print this form for your records and close all Internet browser windows. Thank you.

General Information

Date:	3/12/2008 6:04:14 PM
Student Name:	Brian Means
Program:	Distance Education
Term:	Spring 2008
Daytime Phone:	765-524-1240
Email:	bmeans@bsu.edu

!! ** Unsuccessful Registrations ** !!

The following registrations were unsuccessful. You may click on the link to the right of each status to view the reason for the unsuccessful registration.

	Ref.	Course	No.	Sec.	Cr.	Time	Repeat			Status
1.	70435	AHSC	233	800C	4	ARR , 0000-0000	No			[illegible]
2.	70516	AHSC	242	800C	3	ARR , 0000-0000	No			Error
3.	70524	AHSC	251	800C	4	ARR , 0000-0000	No			Error

Billing Information

Description	Quantity	Total
Undergraduate Credit Hours	0	$0.00
Graduate Credit Hours	0	$0.00
Professional Malpractice Insurance Fee	0	$0.00
Clinical Fees	0	$0.00
Grand Total		$0.00

Holds

Permissions

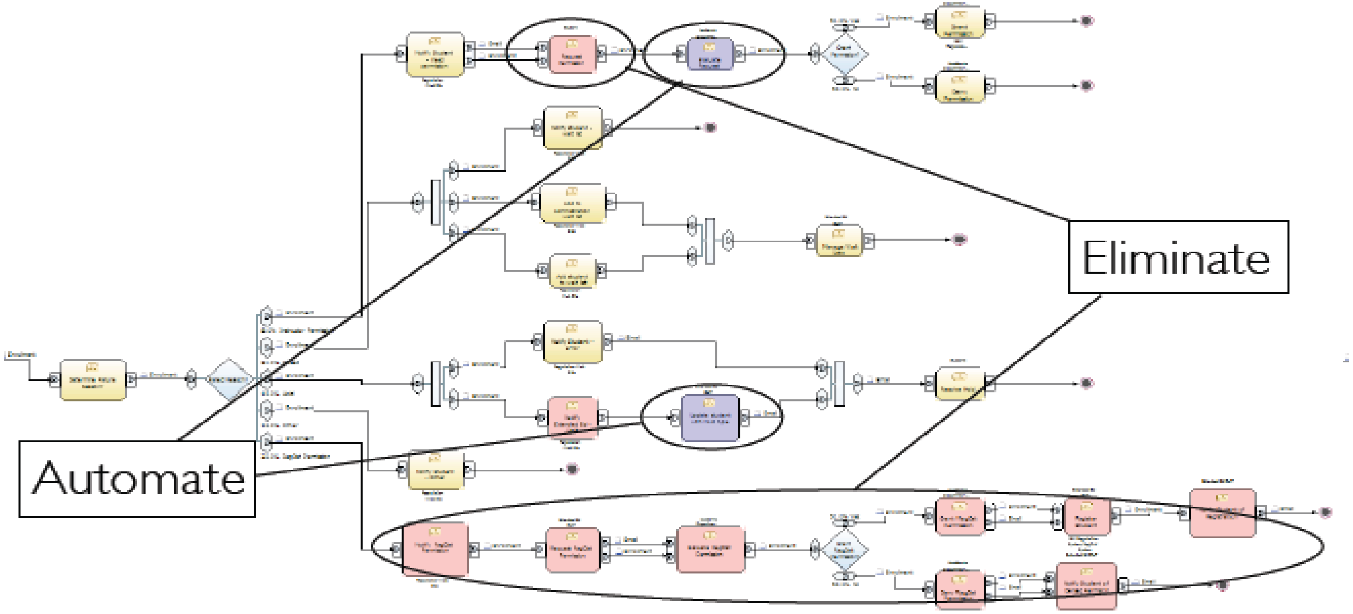
Wait Lists

Holds Permissions Wait Lists

Extended Education Modeling: Current-State Model

The current process includes many manual steps performed by the student body and extended education staff. Opportunities for automation and elimination steps were identified

Holds
Permissions
Wait Lists

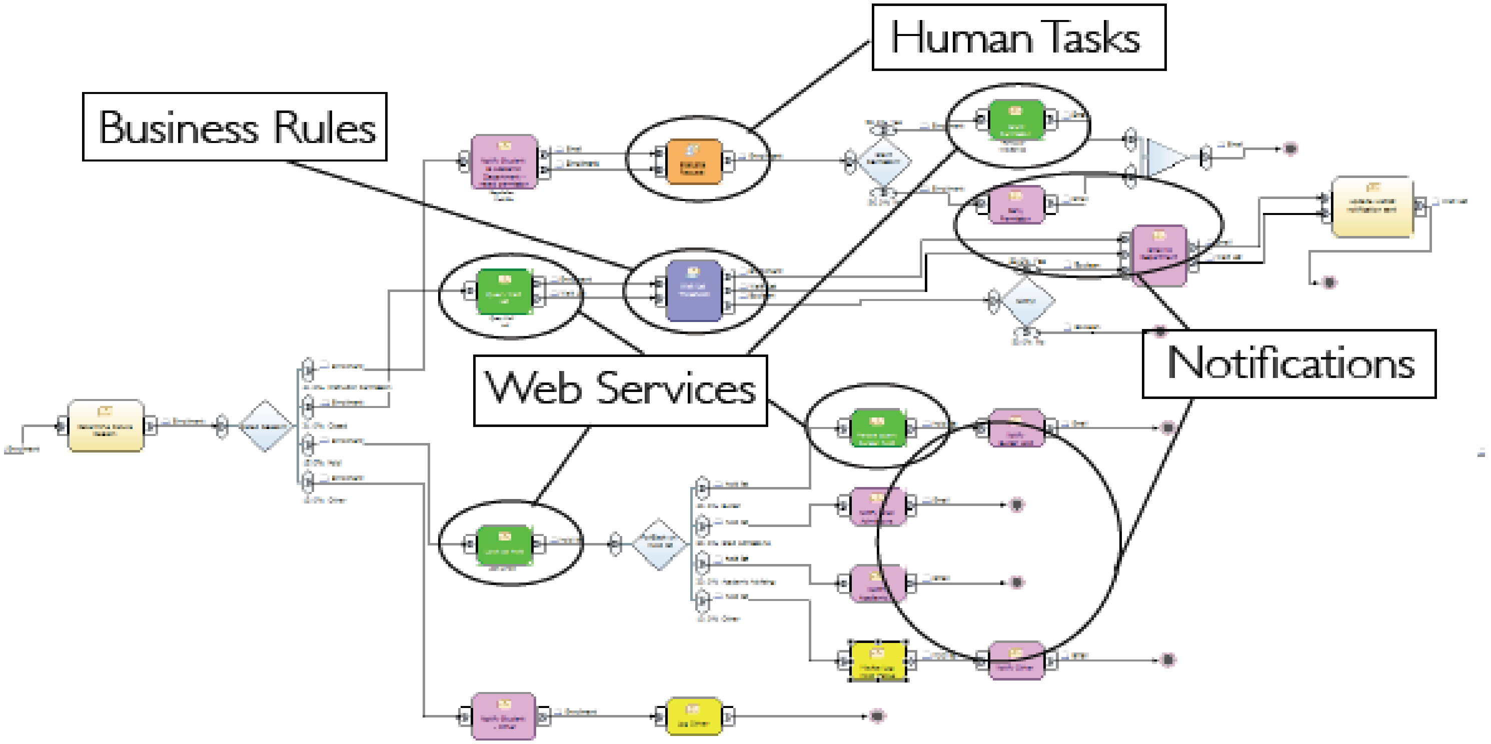


Current-state Issues

- Manual processes can benefit from automation
- Need to decrease work of extended education staff
- Need to increase service to students
- Need to reduce denied enrollments and/or reduce time required to resolve the reason why enrollment cannot be completed

Extended Education Modeling: Future-state Model

In the to-be process business rules, human tasks, notifications and web services enhanced the model



To-be State Benefits

- Student involvement reduced
- Automation reduces staff work and increases information sent to students
- Human Task simplifies work for academic departments
- Business Rule makes wait list management flexible

Extended Education Integration: Project Timeline

I Week Each:

WebSphere Business Modeler

- As Is model
- To Be model
- Export to BPEL (business process execution language)

2 Weeks:

WebSphere Integration Developer and Testing

- Import BPEL from Business Modeler
- 3 Process components (BPEL)
- 1 Human task component – built-in escalation feature
- 3 HATS web services
- 2 Stored Procedures – MS-SQL 2005 - built-in JDBC adapter
- 1 Email Component – built-in email adapter within WID
- 1 Business rule

2 Weeks:

Deploy to Process Server on z/OS

- Performance and stress testing
- Final end user testing
- Deploy

BSU Project Observations:

Selecting the right process

- Keep focus on the business issues
- Find the SME that can define business needs
- Be patient

Fast payoff – from problem definition to initial beta testing – 3 weeks

Business value

- Reduce time to enrollment into courses
- Moved the gatekeeper issues to the gatekeeper creators (Bursar, Departments, etc…)

Entire process are self documenting

- 300+ page PDF document consumer ready
  - ***So complete that BSU wanted to use it as a project acceptance document***

Industry: *Education*
URL: *http://cms.bsu.edu/*

"If you tried to have each system be aware of 40 other systems where addresses are kept, you'd be constantly in maintenance mode trying to accommodate how another system maintains addresses and how do you communicate to it. So, the advent of SOA and Web services is not just an evolutionary step, to me it's a revolutionary step."
Fred Nay, IT Director, Ball State University

Ball State University Student Registration Summary

Ball State University solves key student registration issue with an IBM BPM solution running on z/OS.

CHALLENGES

Minor unpaid student fees and fines prevented student from registering and paying tuition for upcoming semester

Effective Customer Service

High involvement with Registration Staff

Business flexibility

SOLUTION

WebSphere Business Modeler to document their existing processes and then to make changes to resolve the problems

WebSohere Integration Developer to assemble and wire the services required by the process model

WebSphere Process Server on z./OS to orchestrate and streamline business processes and perform workflow automation

System z runtime environment

BENEFITS

Streamlining and automation of business processes in student registration

Less involvement from Registration Staff

Increased number of successful registrations

Improved, meaningful messages & next steps when registration fails

Services created for this project can be reused in later SOA efforts

Self-Service Pattern

The BSU Projects:
How was all this accomplished?

IBM provided:

- Guidance, training and experience
- SOA and PIW workshops

The Solutions included:

- An SOA infrastructure
- Reuse of existing assets (CICS applications, other apps)
  - Existing production environment not upset
  - Existing skill set – 30 years of CICS and mainframe

The Product set included:

- IBM WebSphere Process Server
  - Provides the engine for runtime execution of the business processes
- IBM WebSphere Business Modeler – to model the business processes
- IBM WebSphere Integration Developer – to assemble the SOA services specified in the model
- IBM WebSphere Application Server
- IBM WebSphere ESB
  - Connects siloed applications without hand-coding individual API calls
  - With WebSphere Application Server, communicates between decoupled back and front ends, choosing the destination for a message and transforming it into the correct format
- DB2, the SOA database
- IBM CICS Transaction Server, Version 3.1
  - Extends CICS applications to an SOA
  - Strengthens the application development capabilities of CICS
- IBM WebSphere Host Access Transformation Server (HATS) - to generate the Web Services Definition Language (WSDL) that is callable by BSU's .NET front end to publish information to the user interface.

धन्यवाद
Hindi

多謝
Traditional Chinese

Grazie
Italian

ขอบคุณ
Thai

Gracias
Spanish

多谢
Simplified Chinese

Спасибо
Russian

Thank You

Obrigado
Brazilian Portuguese

شكراً
Arabic

Danke
German

Merci
French

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다

Backup Slides

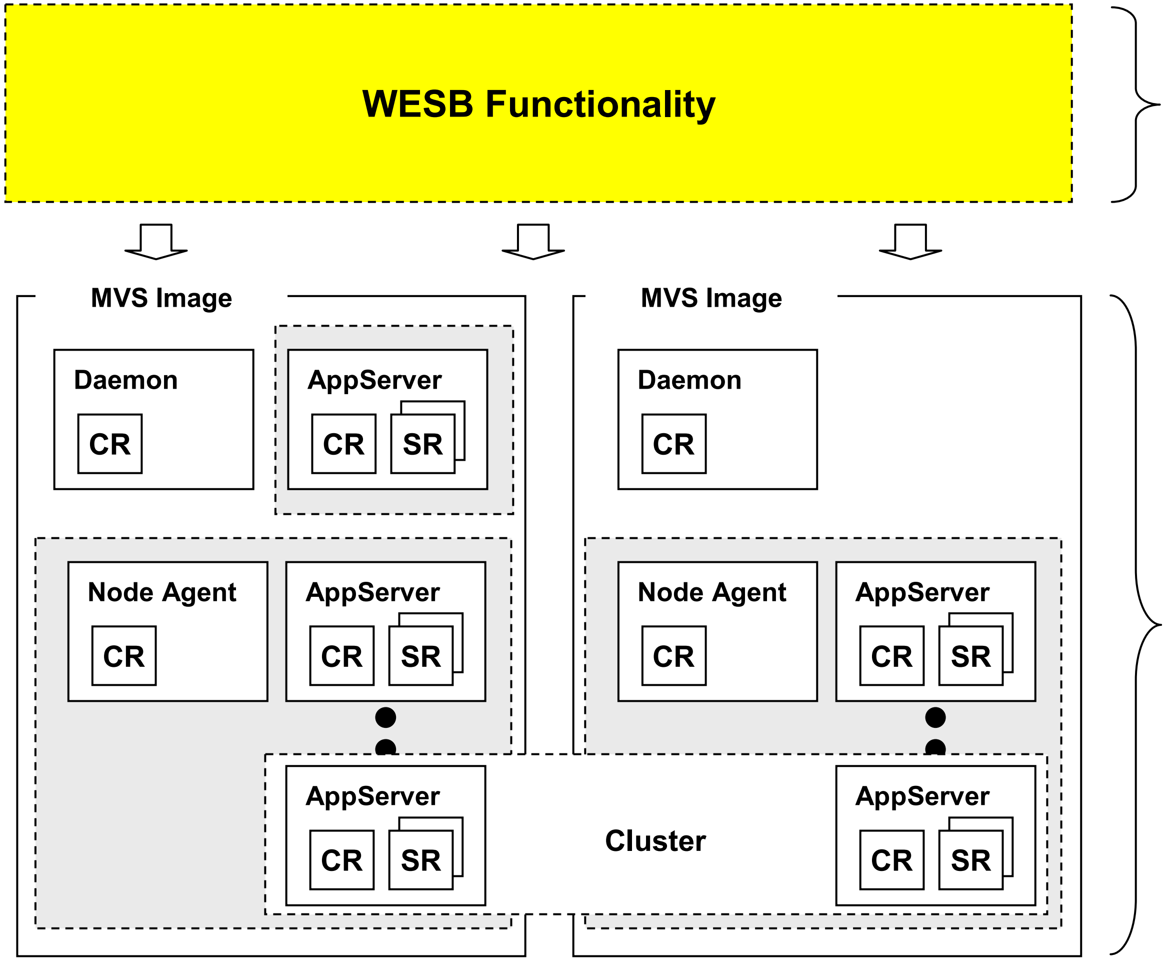
WMB and WSRR

An introduction to

WebSphere Enterprise Service Bus (WESB)

Start With A Recap of WebSphere Application Server

We do this because WESB is built on top. WAS provides the communication infrastructure and J2EE runtime environment. WESB extends that.



This builds on it by providing:

- Mediation framework
- SCA, SDO support (more in a bit)

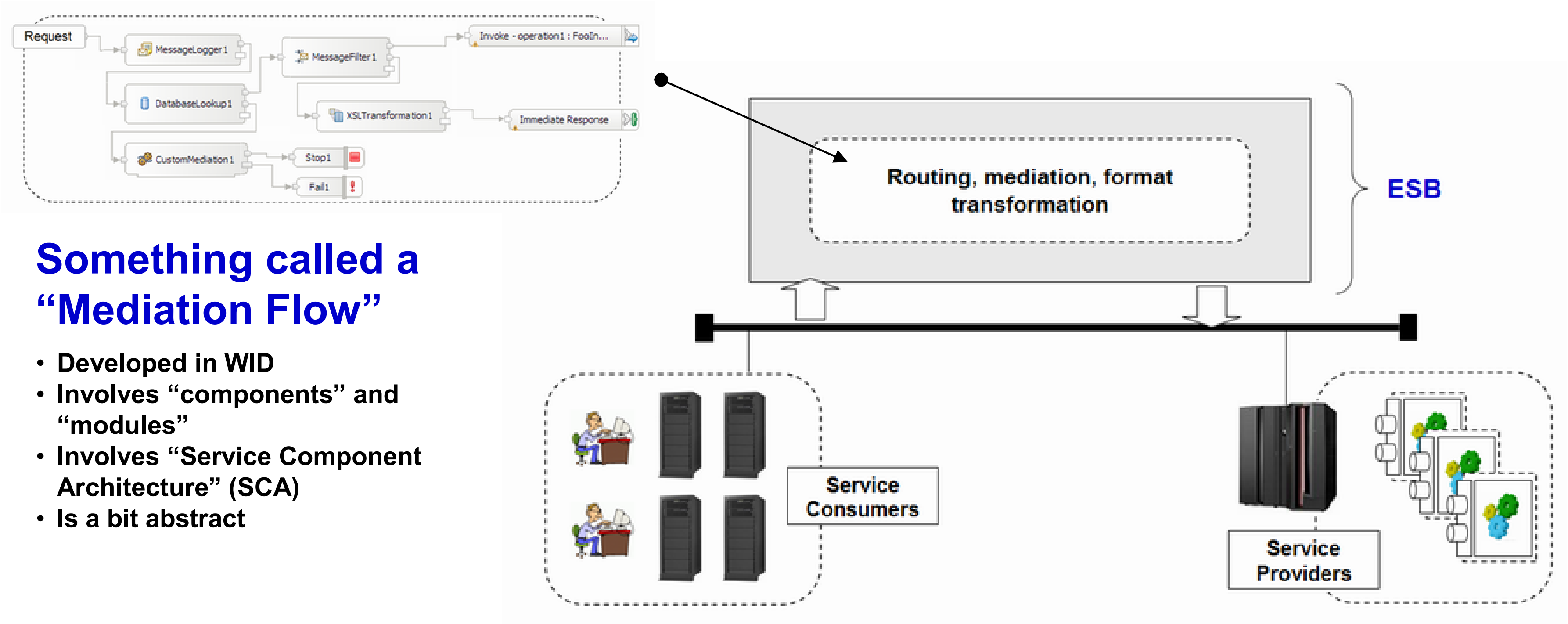
This provides the necessary core infrastructure

- Physical runtime environment
- J2EE application framework
- Communication access
- Essential protocol support
- Web Services intrastructure
- Messaging infrastructure
- Backend data adapter support
- Management and administrative framework

Programming in the ESB …

Message Handling Intelligence in the ESB

To discuss WESB it quickly becomes necessary to talk about the programming capabilities WESB supports. That will be our focus over next several charts.



Something called a "Mediation Flow"

- Developed in WID
- Involves "components" and "modules"
- Involves "Service Component Architecture" (SCA)
- Is a bit abstract

It's a key part of the WESB story. Talking about it requires us to explore how the mediation flow is built and what it can do. It's a more abstract discussion.

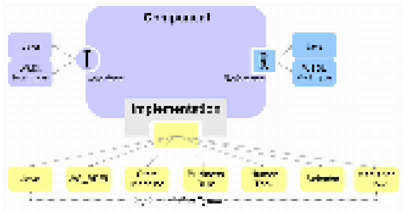
Intro to SCA …

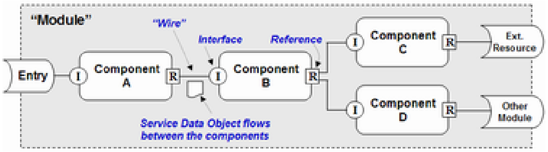
Background and History of Service Component Architecture

Launched around 2005, this is intended to standardize development of applications that conform to SOA principles.

Key Vendors: BEA Systems, Cape Clear Software, IBM, Interface21, IONA Technologies PLC, Oracle, Primeton Technologies Ltd, Progress Software, Red Hat Inc., Rogue Wave Software, SAP AG, Siebel Systems, Software AG, Sun Microsystems, Sybase, TIBCO Software Inc.

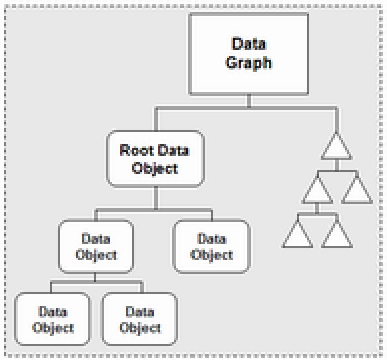
Abstract Program Representation





- Orients programming model around concept of services with interfaces and references
- Graphical development tools will assist in drawing out and generating the code

Abstract Data Representation



- A standardized way to represent data
- Provides for easier data interchange
  - Within a mediation flow
  - Between service implementations

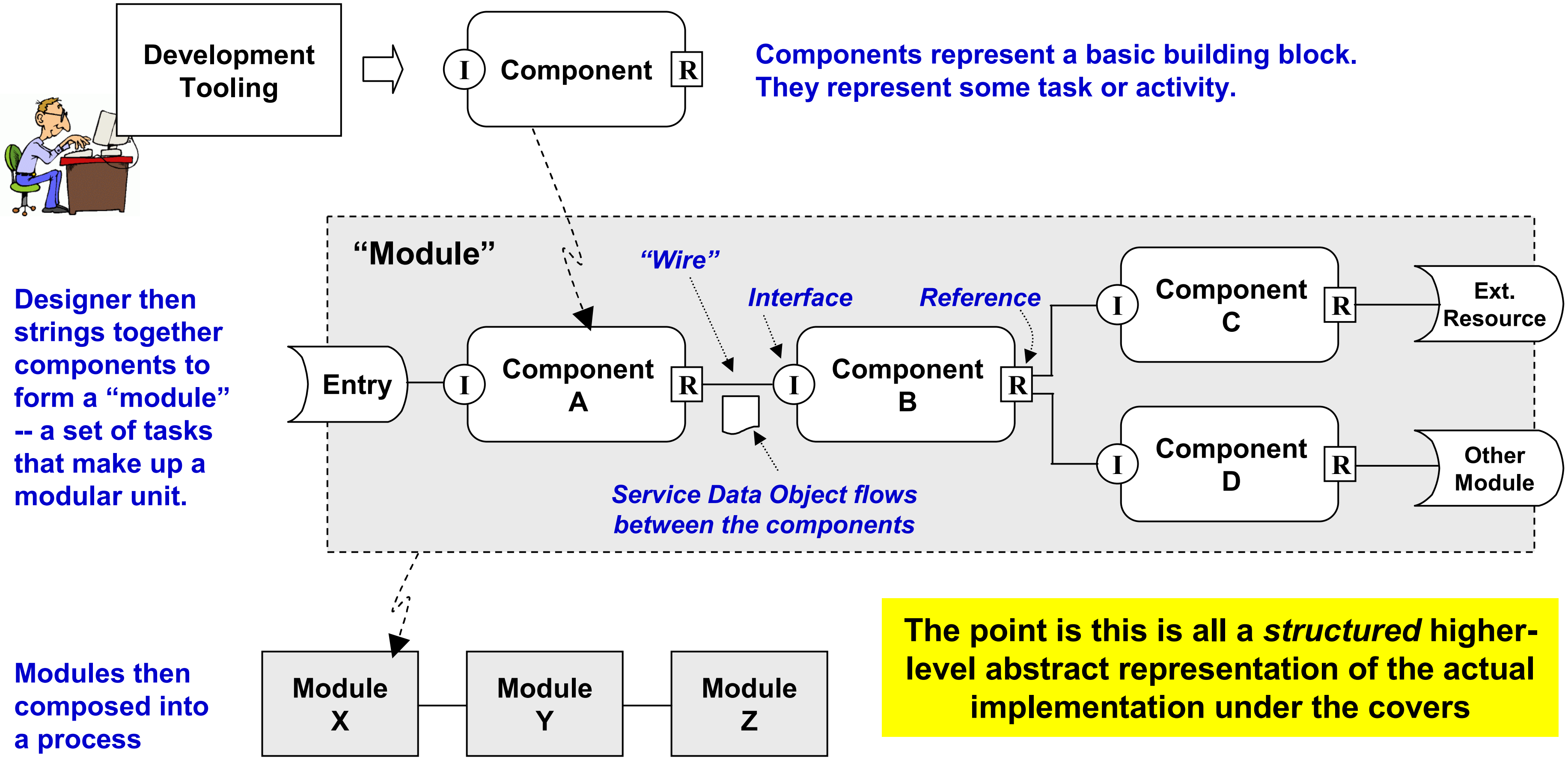
WESB's programming model is based on SCA, so discussing it requires we touch on SCA.

High Level of SCA …

High Level of Service Component Architecture (SCA)

SCA is an *architectural model* that seeks to *abstract* the actual implementation and provide a consistent higher level model for developers *and tooling*

This can be a very slippery concept to some



Components represent a basic building block. They represent some task or activity.

Designer then strings together components to form a "module" -- a set of tasks that make up a modular unit.

Service Data Object flows between the components

Modules then composed into a process

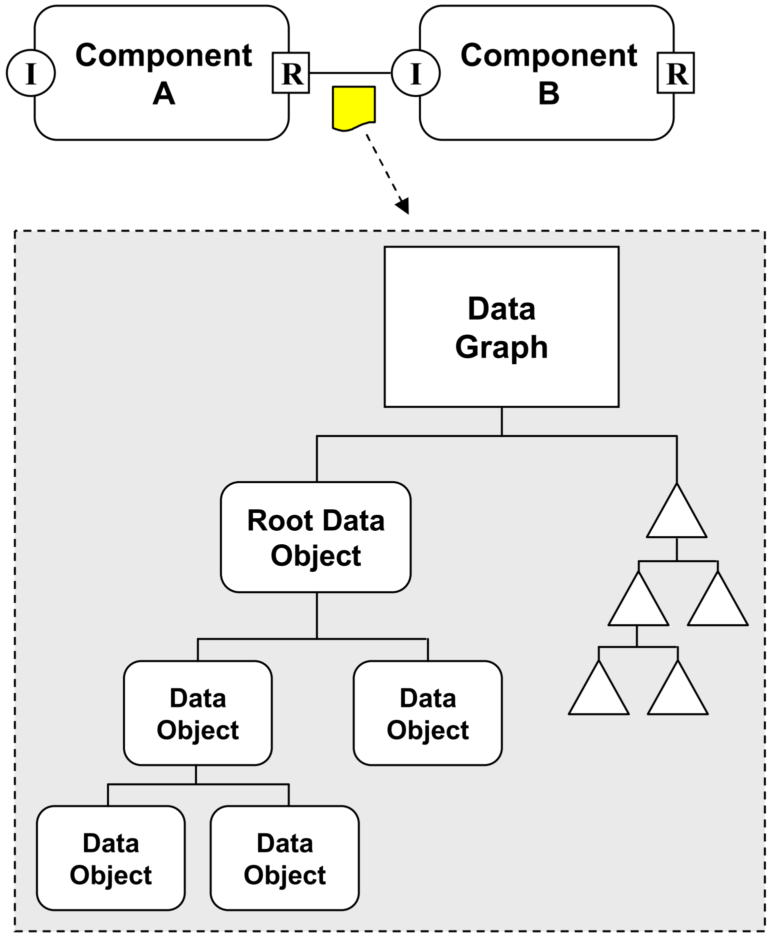
The point is this is all a *structured* higher-level abstract representation of the actual implementation under the covers

Service Data Objects …

Service Data Objects

A structured way of representing data handled by components *and* data that flows between components. When flowing, SDO's are in XML format.

This is often represented in a "tree" format:



Data Graph

A "container" (outer wrapper) for the data objects that are held within. A message flowing will typically have more than one data object.

Data Objects

A data object represents a piece of data. A person's name, an invoice number, a price, whatever. Multiple data objects are typically part of a larger message. They too are arranged in a tree format which represents their relationship to one another

Each data object has:

- **Name of data object**
- **Type of data object (simple/complex; scalar/array)**
- **Value (as well as the default value)**

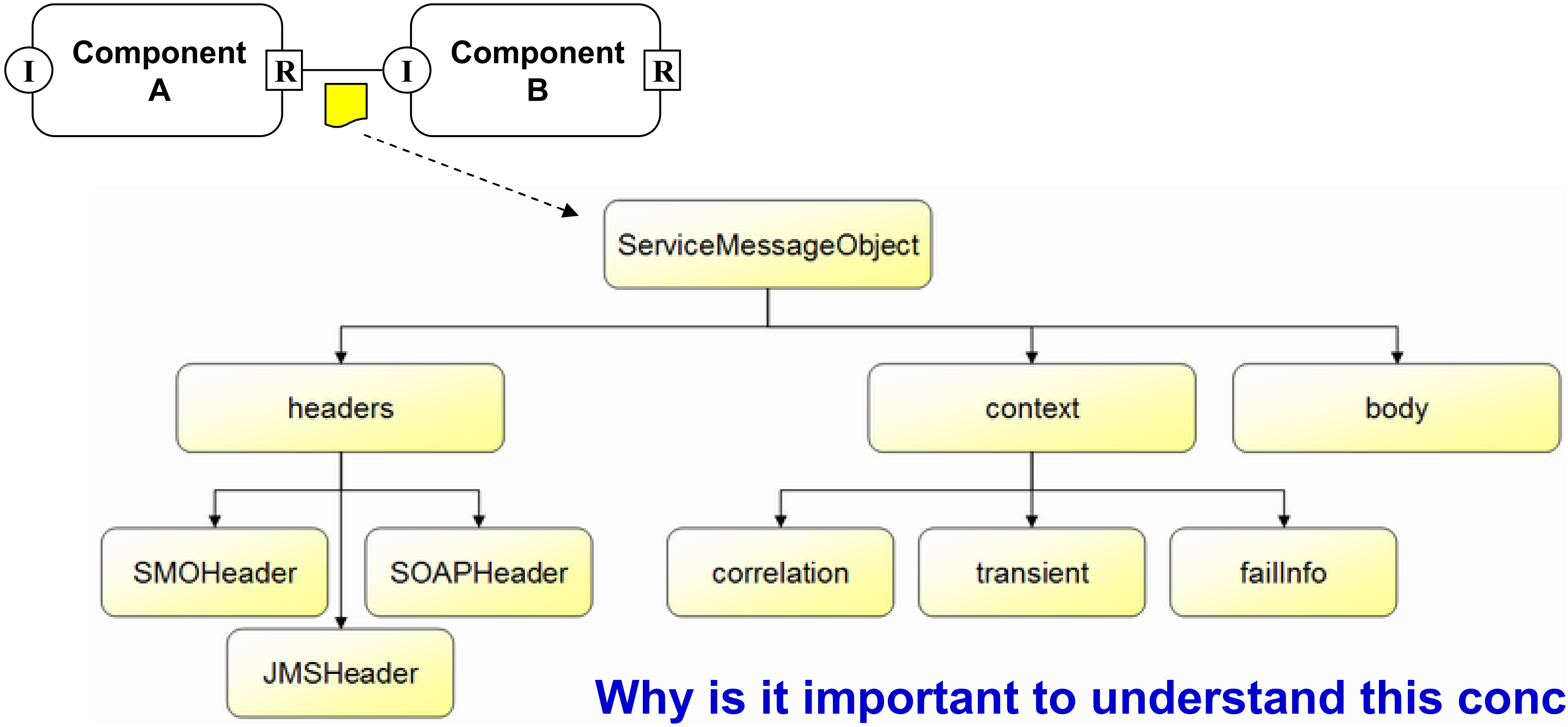
Change Summary

Represents the incremental changes made to the data objects as the SDO moves between components.

Messages are a type of Data Object …

Service Message Objects

Are a form of Service Data Object. The contents of the message is repesented in a tree format:



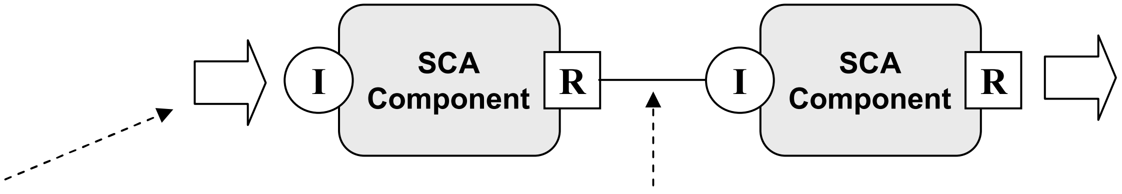
Why is it important to understand this concept?

Because any transformation of the message is going to made against this structured format. Querying specific data elements will be aided by such a structured format. All with structured change data.

Bindings …

The “Bindings” of an SCA Component

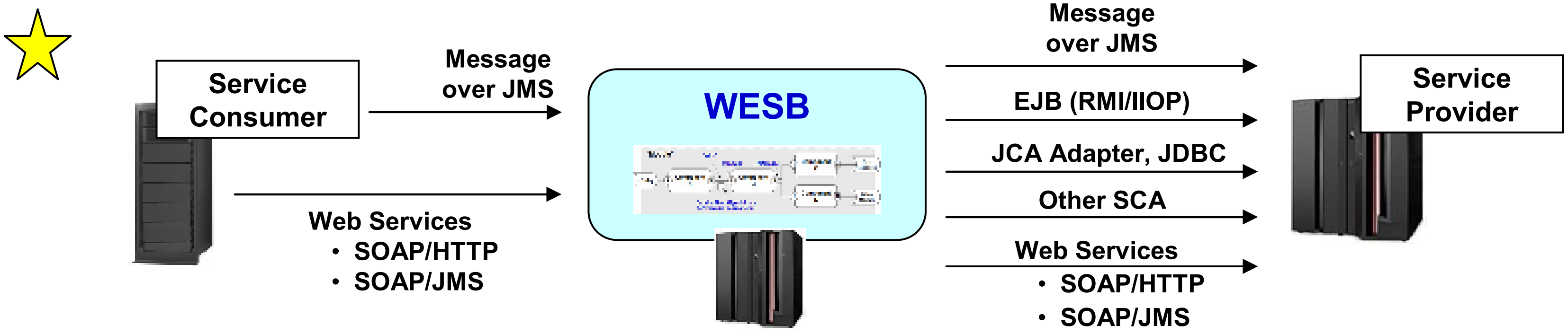
These define what can invoke a component, and what a component can invoke. It’s not “anything” -- there’s a defined set of things:



- JMS (non-SOAP)
- Web services (SOAP/HTTP and SOAP/JMS)
- JCA compliant adapters
- RMI/IIOP (EJBs)

- JMS (non-SOAP messages)
- Web services (SOAP/HTTP and SOAP/JMS)

- SCA bindings (default for component-to-component)



These are all open standard protocols

Development tool …

The Tooling to Develop Mediation Flows

From our first presentation:

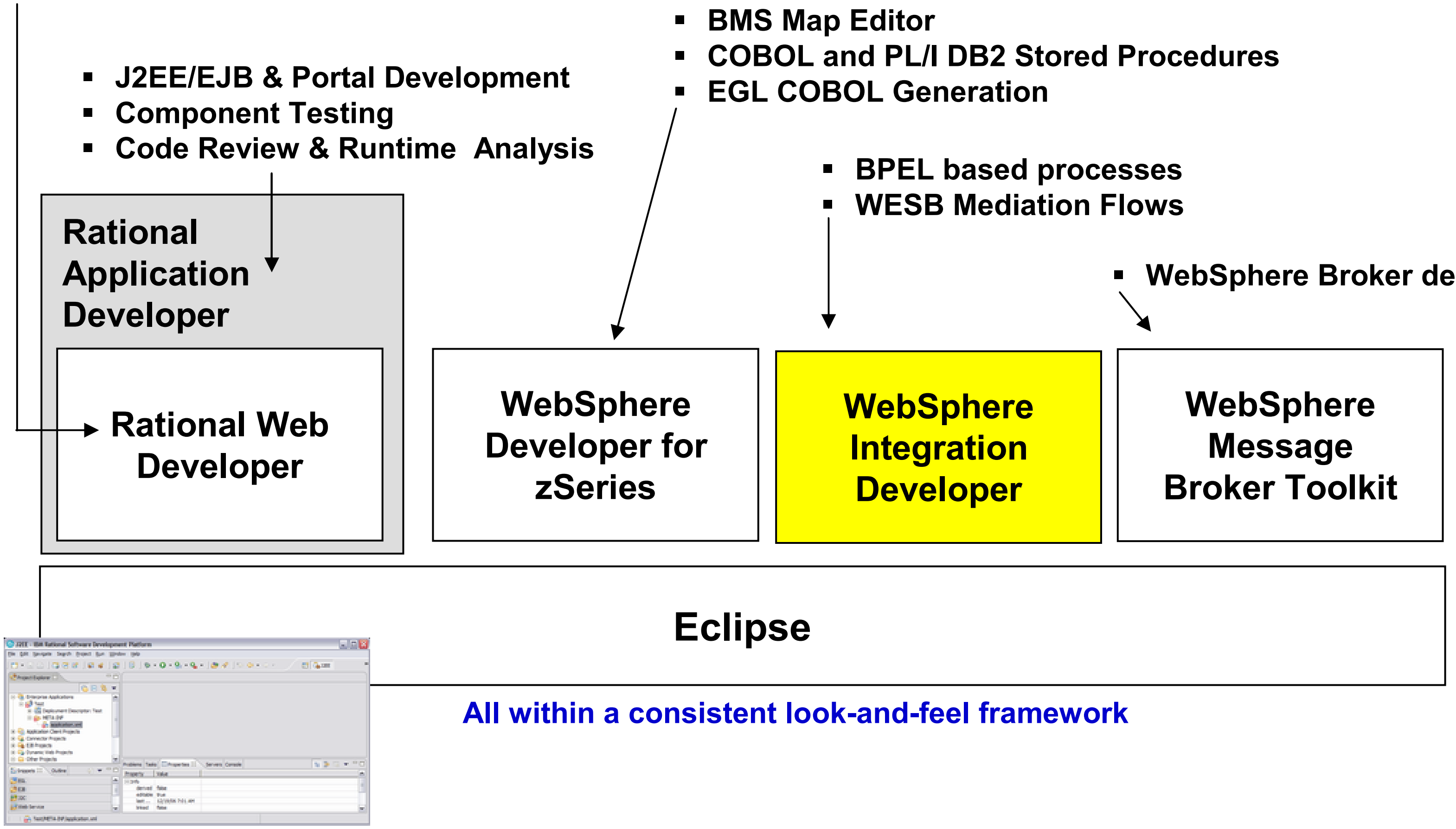
- Web development (servlets, JSPs)
- Web services development
- XML and DB access tools

- z/OS Application Development
- XML Services
- BMS Map Editor
- COBOL and PL/I DB2 Stored Procedures
- EGL COBOL Generation

- J2EE/EJB & Portal Development
- Component Testing
- Code Review & Runtime Analysis

- BPEL based processes
- WESB Mediation Flows

- WebSphere Broker development



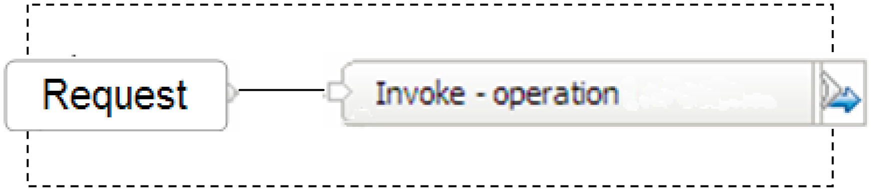
Rational Application Developer
Rational Web Developer
WebSphere Developer for zSeries
WebSphere Integration Developer
WebSphere Message Broker Toolkit
Eclipse

All within a consistent look-and-feel framework

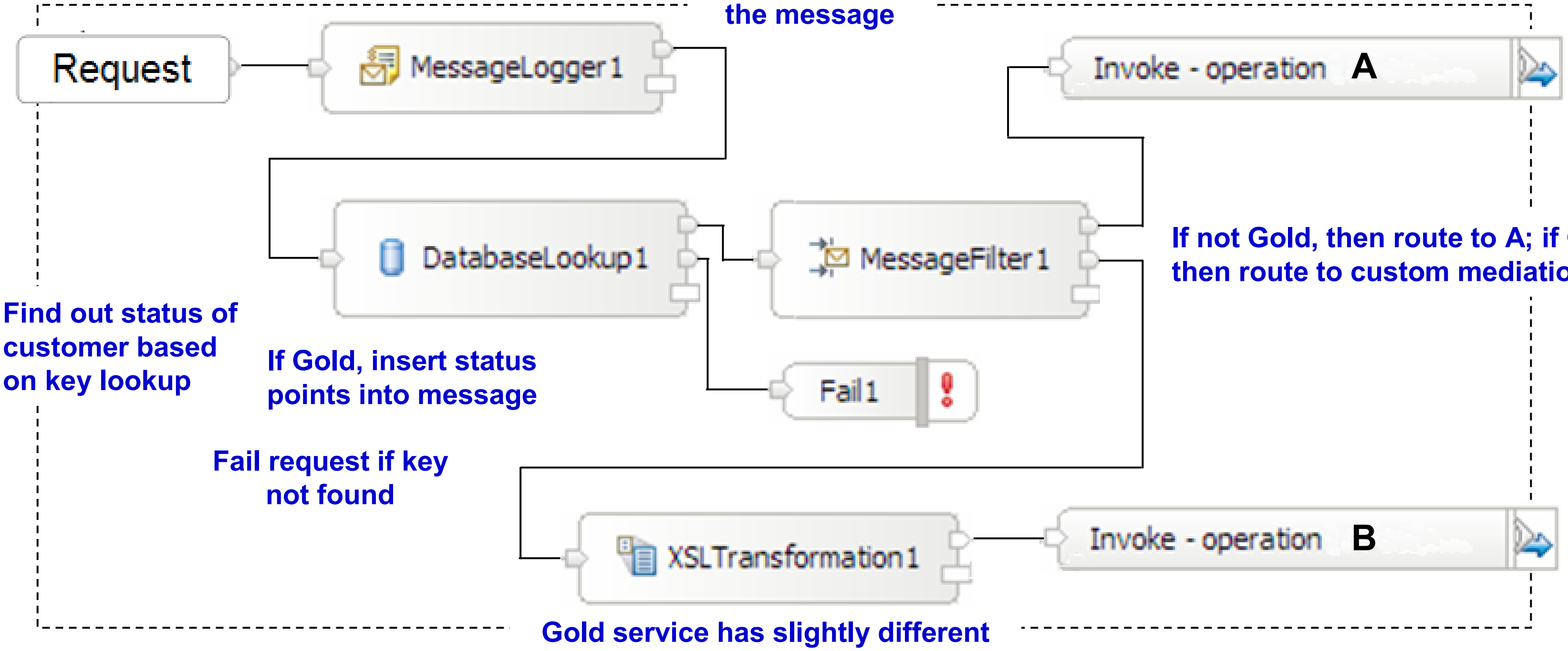
Mediation “primitives” …

Some Examples of WESB Mediation Flows

From simplest to a little more complex



The simplest -- the request is immediately routed to the invocation of the external service
(Not *required* to do mediation … only if you need to)



Log a portion of the message

Non Gold customer

If not Gold, then route to A; if Gold then route to custom mediation and B

Find out status of customer based on key lookup

If Gold, insert status points into message

Fail request if key not found

Gold customer

Gold service has slightly different XML input format, so transform XML

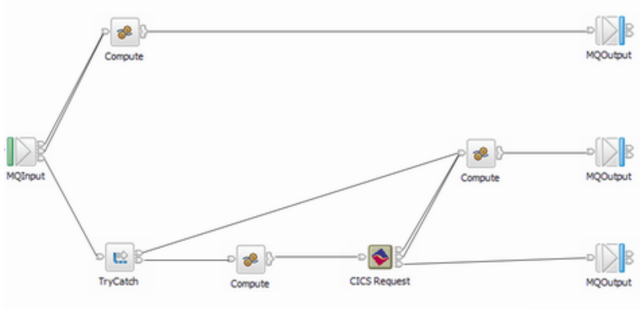
When multiple mediation flows in WESB …

An introduction to

WebSphere Message Broker (WMB)

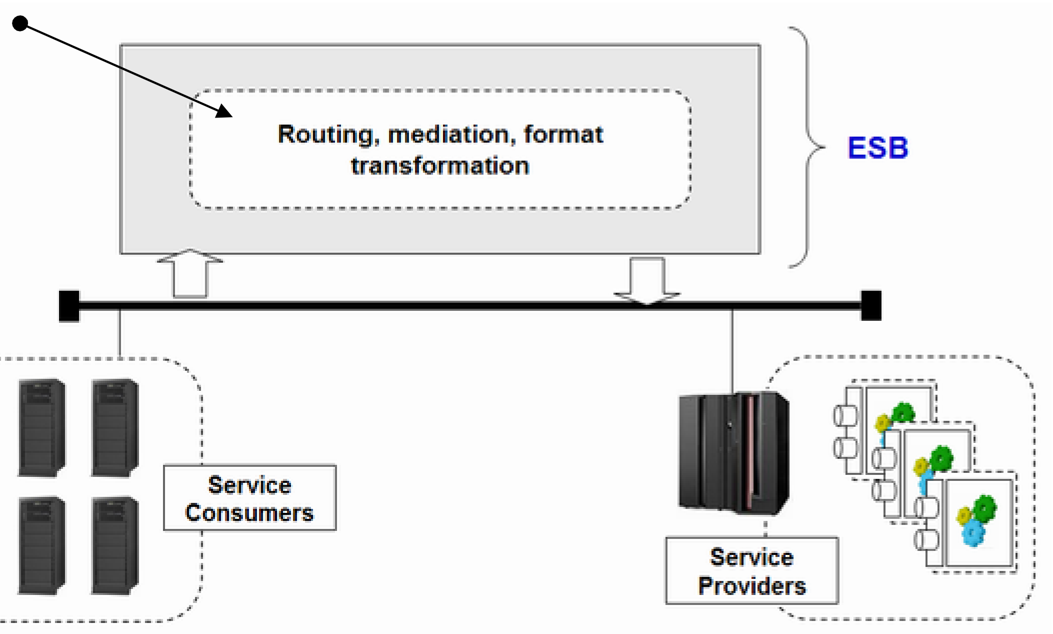
Message Handling Intelligence in the ESB

Again, we need to talk about the programming capabilities of the ESB



Something called a "Message Flow"

Similar to WESB's mediation flow *in concept*.

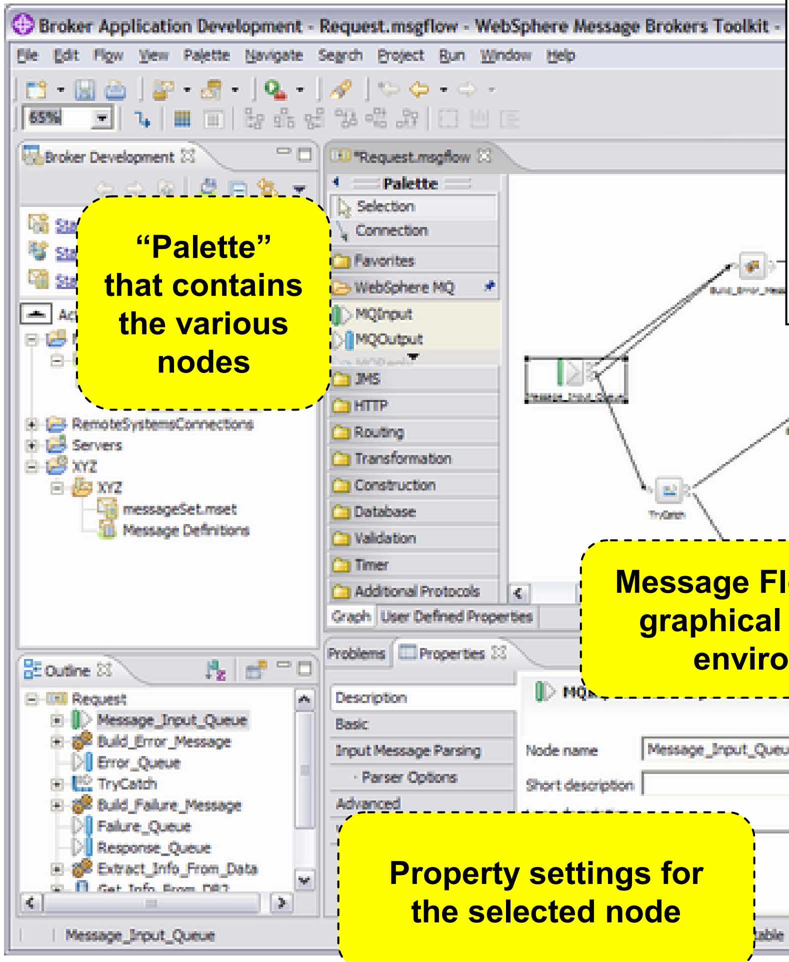


WMB's built in capabilities are far more extensive than WESB's. Much of this story is going to be told by reviewing these built in capabilities.

Broker Toolkit …

Broker Toolkit

Yet another "Eclipse-based" tool used to create WMB message flows:



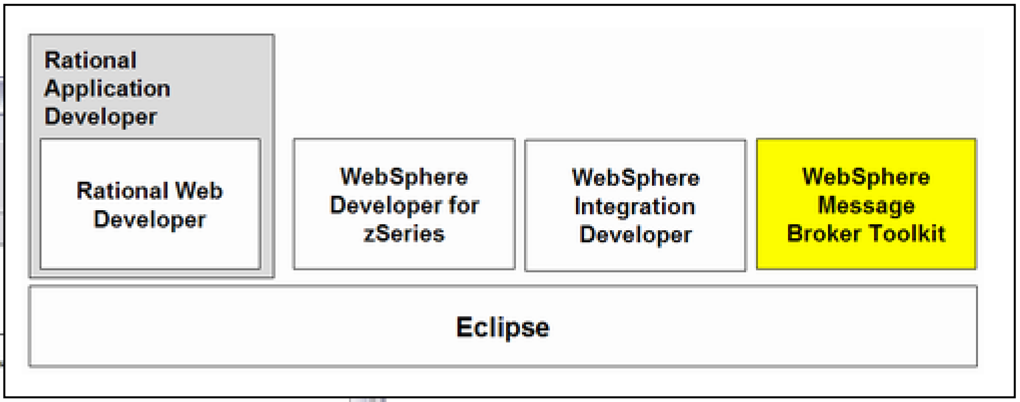
Node name: Message_Input_Queue

Short description:

"Palette" that contains the various nodes

Message Flow Editor -- graphical drag/drop environment

Property settings for the selected node



Rational Application Developer

Rational Web Developer

WebSphere Developer for zSeries

WebSphere Integration Developer

WebSphere Message Broker Toolkit

Eclipse

This is like any graphical environment -- takes some time to get proficient.

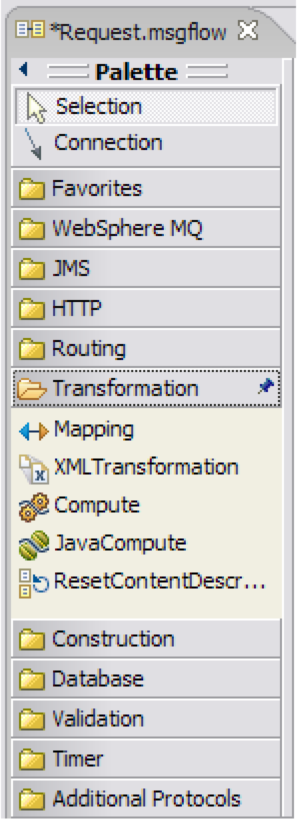
You'll see all of this in lab

The deployment artifact is the BAR file … Broker ARchive

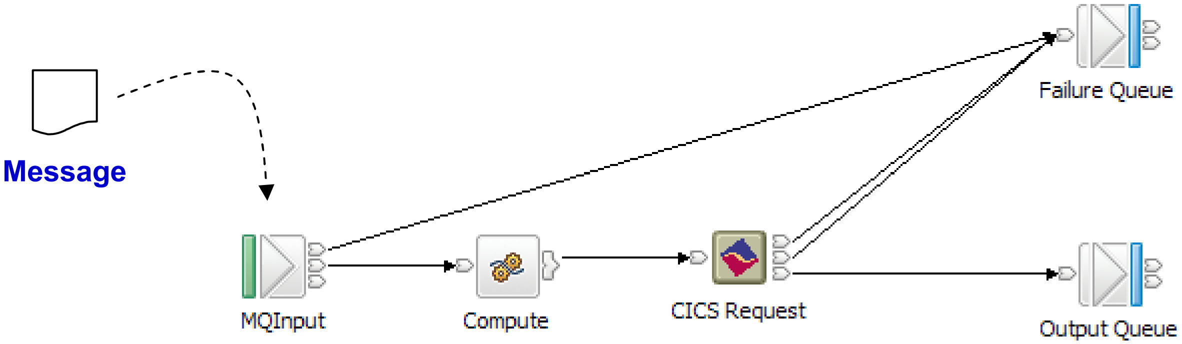
Nodes …

Nodes

Nodes are the basic building blocks of a message flow. They represent functional routines that encapsulate the flow logic. Nodes are used to create a flow, which represents the "reusable integration application" inside the ESB:



Broker Toolkit offers a "Palette" with nodes available for use



Key Points:

- **Each node has a set of properties, such as the MQ queue name, or the URL of the requested web service, etc.**
- **Compute nodes contain your integration logic**
  Compute = ESQL (WMB Language)
  Java Compute = Java
- **Many nodes exist that are not shown here**
  Nodes that come with WMB
  Nodes you can add to WMB

Built-in Nodes …

Built-in Nodes: Input/Output Related

The Broker Toolkit comes with a set of built-in nodes you can use to start building message flows right away. Additional nodes are downloadable.

The downloadable nodes come in the form of SupportPacs. More in a bit.

MQ

MQInput — **Read message off queue**

MQOutput — **Write message to queue**

MQGet — **Read message off queue**

MQReply — **Send response to queue indicated by requester**

MQOptimizedFlow — **Related to pub/sub**

HTTP

HTTP Input — **Receive HTTP request**

HTTP Request — **Send web service request to URL**

HTTP Reply — **Reply to original requester**

JMS

JMSInput — **Receive from JMS destination**

JMSOutput — **Reply to JMS destination**

JMSMQTransform — **JMS to MQ JMS transform**

MQJMSTransform — **MQ JMS to JMS transform**

Telemetry and Real Time

SCADAInput — **Receive from telemetry device**

SCADAOutput — **Receive to telemetry device**

Real-timeInput — **Receive over MQ JMS real time transport**

Real-timeOptimizedFlow — **Real time pub/sub**

The WMB InfoCenter has an excellent description of what each of these nodes do. Just search on node name. URL for InfoCenter:

`http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp`

More built-in nodes …

Build-in Nodes: Routing, Transformation and Database

Routing

Filter	Route based on message content
Label	Used with RouteToLabel
Publication	Route to topic subscribers (pub/sub)
RouteToLabel	Route to label node
AggregateControl	Beginning of message fan-out
AggregateReply	End of a fan-in
AggregateRequest	Related to fan-in/fan-out messages

Transformation

Compute	Construct new output message with ESQL
JavaCompute	Construct new output message with Java
XMLTransformation	Transform XML using XSL
Mapping	Populate message with new content
ResetContentDescriptor	Forces reparsing of the message

Database

Database	Interact with database using ESQL
DataInsert	Insert data into table
DataDelete	Delete data from table
DataUpdate	Update data in table
Warehouse	Store entire message (or part) in database

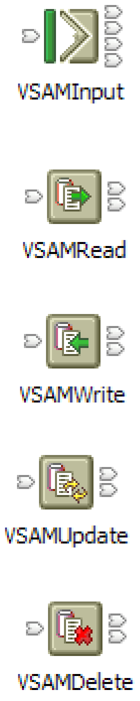
Those were the "built-in" nodes. More nodes are possible. They can be added to your Toolkit. They come packaged as "SupportPacs"

z/OS nodes …

z/OS Specific Nodes

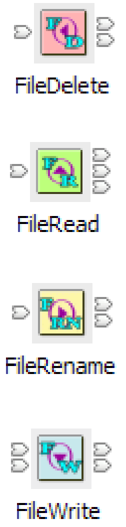
These nodes are designed to interact with specific z/OS resources.

IA13
VSAM



Has a z/OS-side component that needs to be installed

IA11
File Adapter for z/OS sequential files



File format:

- •QSAM
- •F,FB,V or VB only

Has a z/OS-side component that needs to be installed

IA12
CICS

Provides local access to CICS

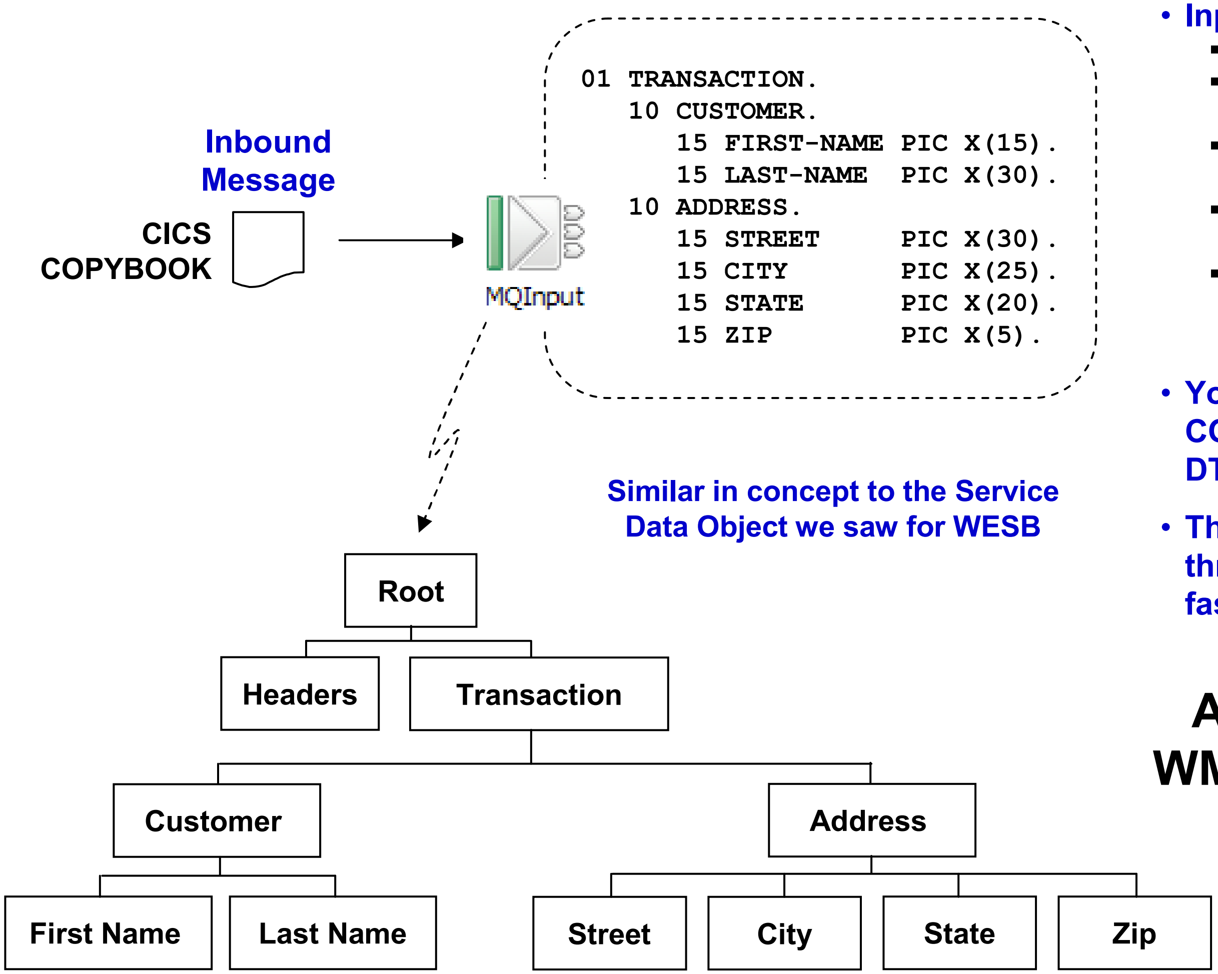
(Note: there are other ways to access CICS without this node. More on that later.)

These SupportPacs extend the function of WMB to do z/OS-specific things

Parsing the received message …

Parsing the Message … and the Logical Message Model

The Input node of a message flow parses the message and constructs a "logical message model" -- a structured representation of the data:



```
01 TRANSACTION.
   10 CUSTOMER.
      15 FIRST-NAME PIC X(15).
      15 LAST-NAME  PIC X(30).
   10 ADDRESS.
      15 STREET     PIC X(30).
      15 CITY       PIC X(25).
      15 STATE      PIC X(20).
      15 ZIP        PIC X(5).
```

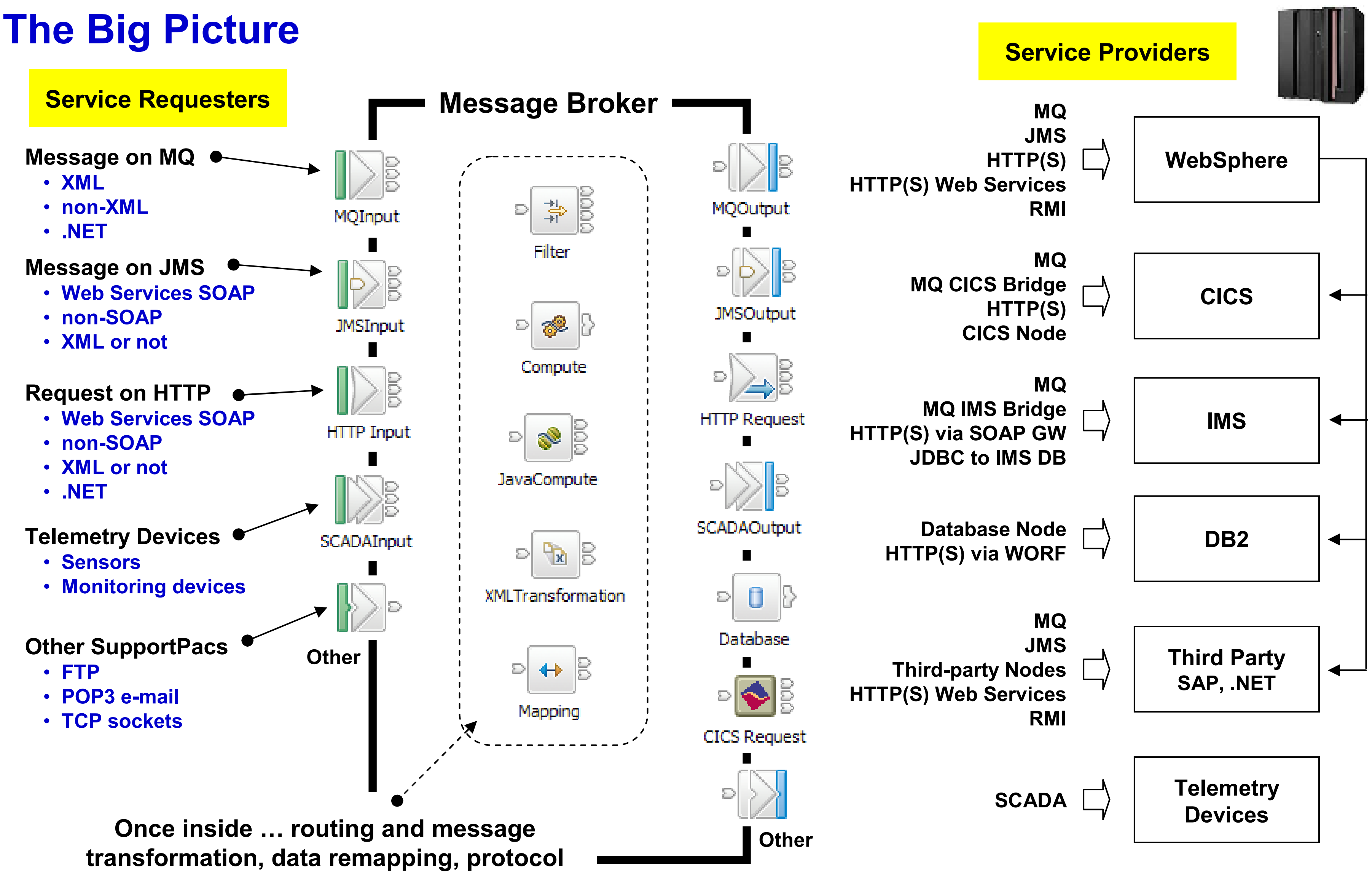
Similar in concept to the Service Data Object we saw for WESB

- **Input node has built in parsers:**
  - BLOB – No structure, just a sequence of bits
  - XML, XMLNS, XMLNSC – Self defining, generic XML message
  - MRM – Fixed record structures, tagged/delimited or XML
  - JMS – Standard folder structure for a JMS message
  - These parsers can be supplemented with user written custom parsers or those purchased from a third party
- **You can define modeling by importing COPYBOOK, C Header Files or XML DTDs or schema**
- **This provides a consistent model throughout WMB, and provides for very fast data access.**

A lot of flexibility here … WMB06 workshop goes into far more detail.

Big Picture …

The Big Picture



Service Requesters

Message on MQ
- XML
- non-XML
- .NET

Message on JMS
- Web Services SOAP
- non-SOAP
- XML or not

Request on HTTP
- Web Services SOAP
- non-SOAP
- XML or not
- .NET

Telemetry Devices
- Sensors
- Monitoring devices

Other SupportPacs
- FTP
- POP3 e-mail
- TCP sockets

Message Broker

Service Providers

MQ
JMS
HTTP(S)
HTTP(S) Web Services
RMI → WebSphere

MQ
MQ CICS Bridge
HTTP(S)
CICS Node → CICS

MQ
MQ IMS Bridge
HTTP(S) via SOAP GW
JDBC to IMS DB → IMS

Database Node
HTTP(S) via WORF → DB2

MQ
JMS
Third-party Nodes
HTTP(S) Web Services
RMI → Third Party SAP, .NET

SCADA → Telemetry Devices

Once inside … routing and message transformation, data remapping, protocol conversion, pub/sub, transaction management, logging, monitoring

Small Detour

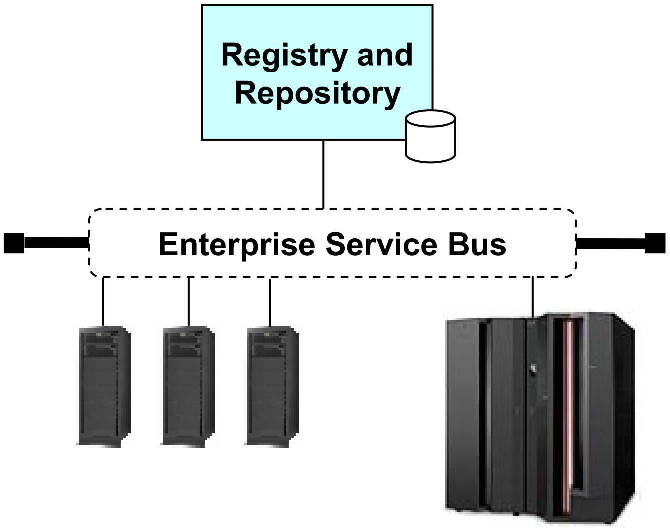
WebSphere Services Registry and Repository

(WSRR)

Setting the Stage for Discussion of “Registry and Repository”

There are two fundamental things a registry and repository is going to provide:

As a runtime source of information, *retrieved programmatically*

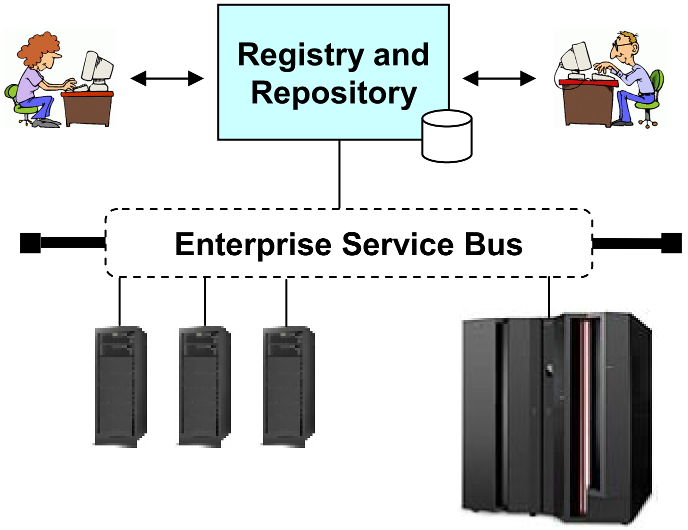


Registry and Repository

Enterprise Service Bus

This is perhaps the more common use of a registry people think about …

As a organized and centralized source of information for planning, developing and managing the SOA environment



Registry and Repository

Enterprise Service Bus

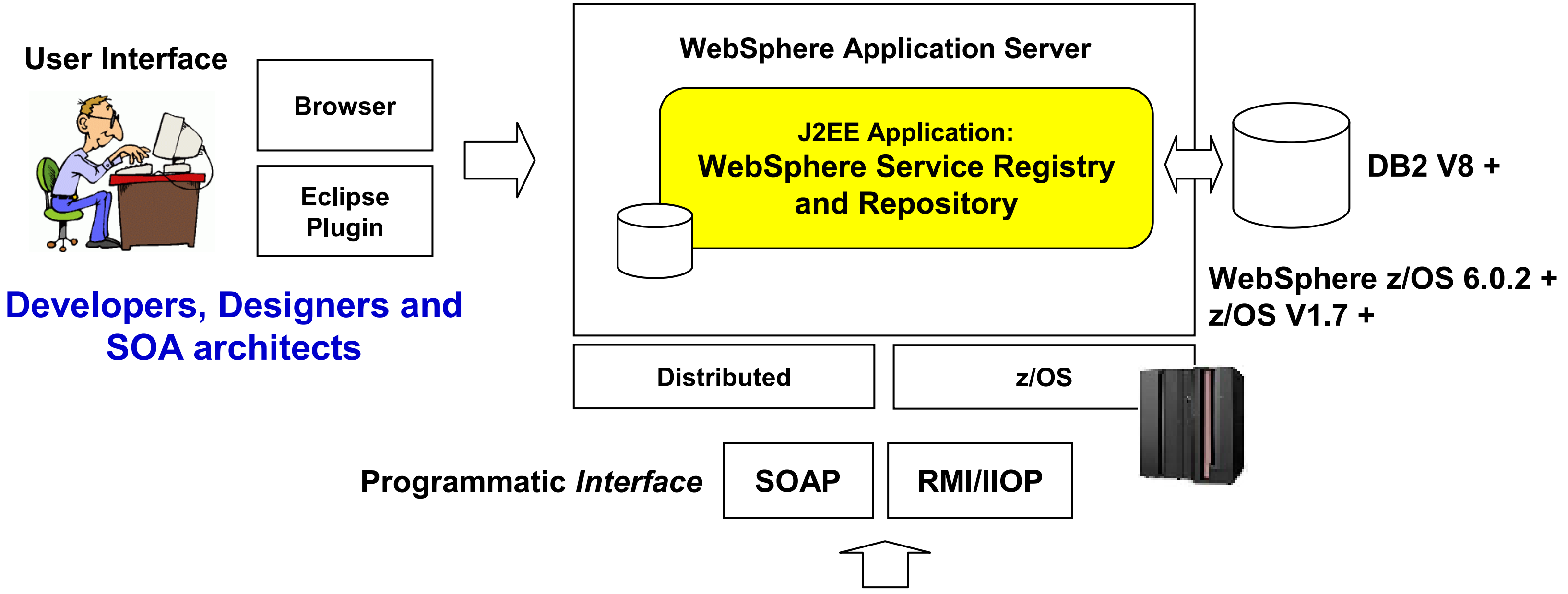
… but overlooking this means you’ll miss what IBM is planning for this function.

Imagine trying to control this if the information was scattered across different notebooks, spreadsheets, yellow sticky notes on the wall, in their heads, etc.

WebSphere Service Registry and Repository …

WSRR … What It Is Physically

It's a sophisticated WebSphere J2EE application with a relational data store:



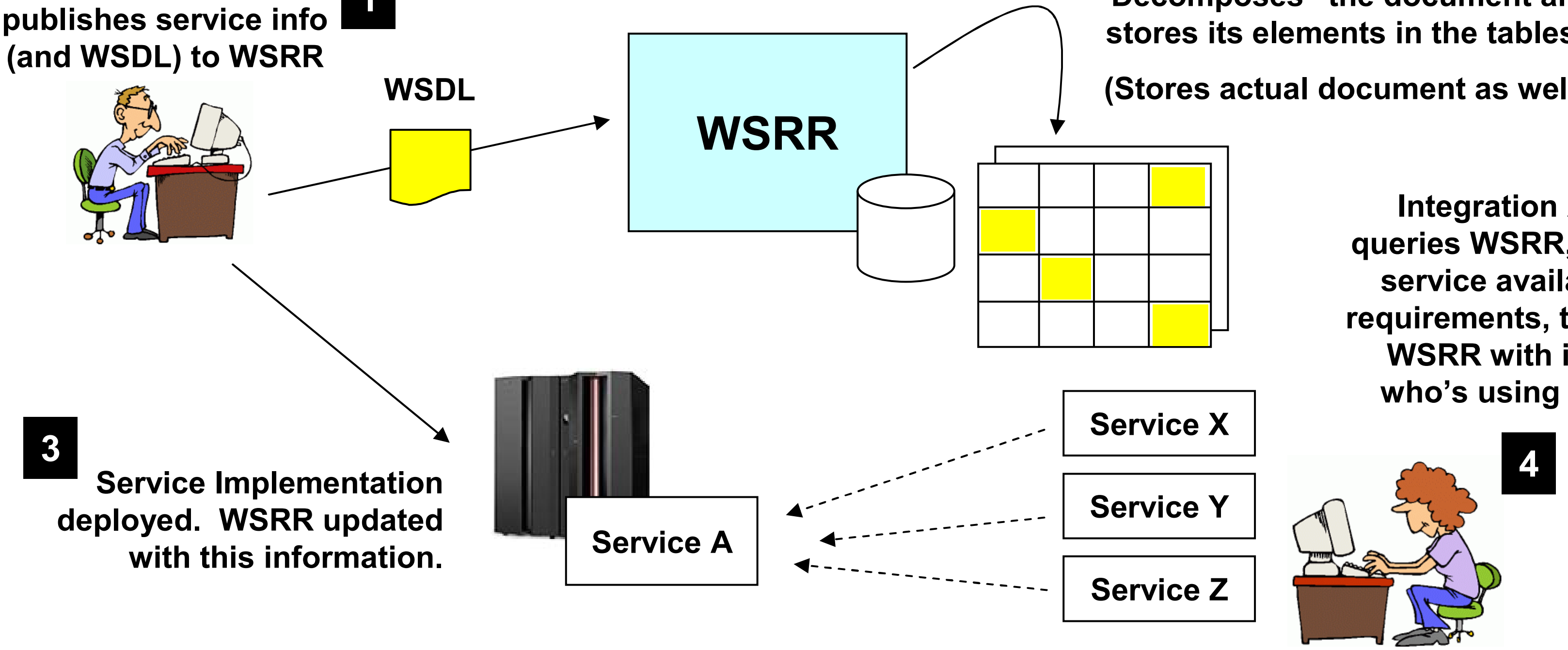
SOA runtime components - ESB, requesters, process manager
Tools to monitor and manage the environment

This is designed to be more than just a place where service endpoints are registered, or WSDL files stored.

An example …

Fine Grained Queries and Associations

To give you a sense of this, consider the following:



1 Service Developer develops service and publishes service info (and WSDL) to WSRR

2 “Decomposes” the document and stores its elements in the tables.

(Stores actual document as well)

3 Service Implementation deployed. WSRR updated with this information.

4 Integration Architect queries WSRR, determines service availability and requirements, then updates WSRR with info about who’s using Service A.

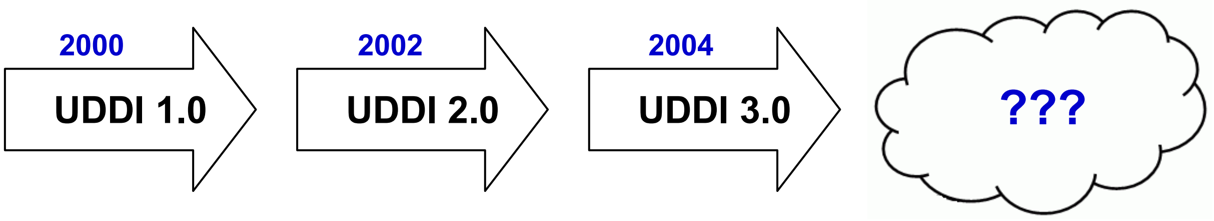
WSRR now has information about the service, provides for fine-grained queries against individual elements of service artifact, has information on status of service, and has information about what other things are using the service.

This is more than just a simple registry for service endpoint information

Evolving standards …

Evolving Standards in this Space

The UDDI standard appears to be stabilized. It has limitations. Work is under-way to define a broader industry standard. IBM is involved in that effort.



2000 UDDI 1.0

2002 UDDI 2.0

2004 UDDI 3.0

???

UDDI V3 has some meta-data capabilities, but its focus is primarily WSDL. Its limitations are the reason why there are proprietary extensions to UDDI in the market.

The UDDI Task Committee has done no additional development on UDDI in several years.

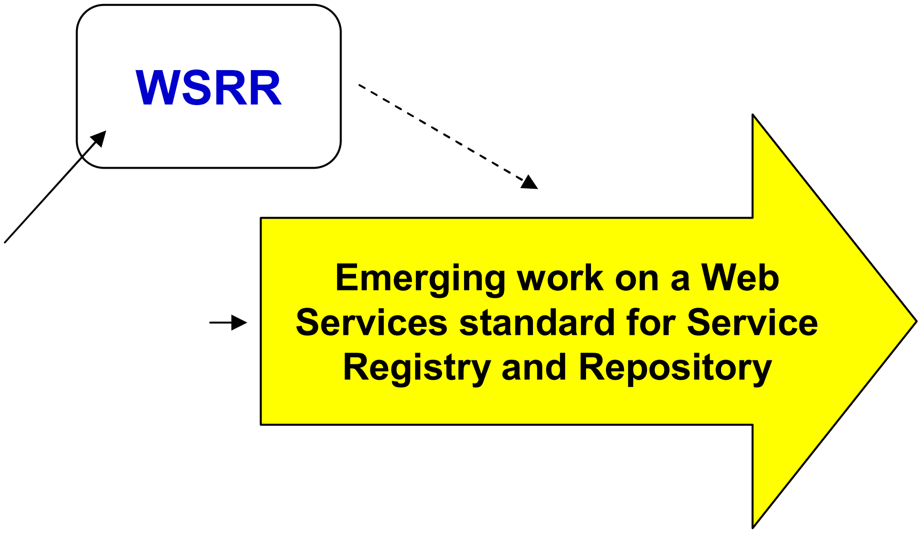
(Recent work has been technical notes explaining how UDDI works and how to use it)

A decision to close down the task committee is pending.

IBM was heavily involved in UDDI from the beginning.

IBM is heavily involved in the development of the emerging registry and repository standards

Over time WSRR will merge up with those standards



WSRR

Emerging work on a Web Services standard for Service Registry and Repository