WebSphere

Enterprise Integration

A Survey of Programming Models, Tools and Runtimes

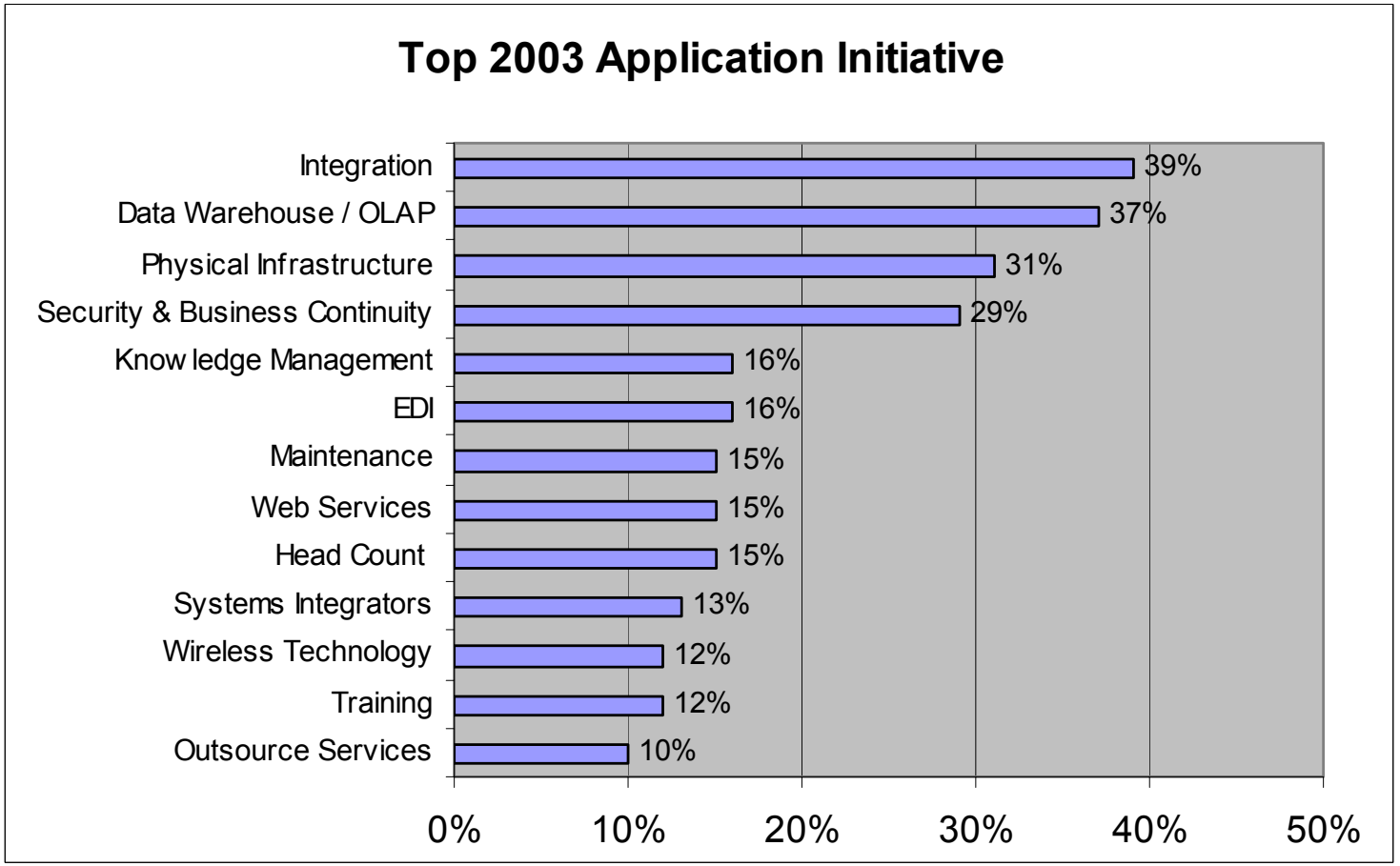
Rob High, Jr.

DE, Chief Architect, WebSphere Application Server

With credit due to Don Ferguson, Kristof Kloeckner, Mike Beisiegel

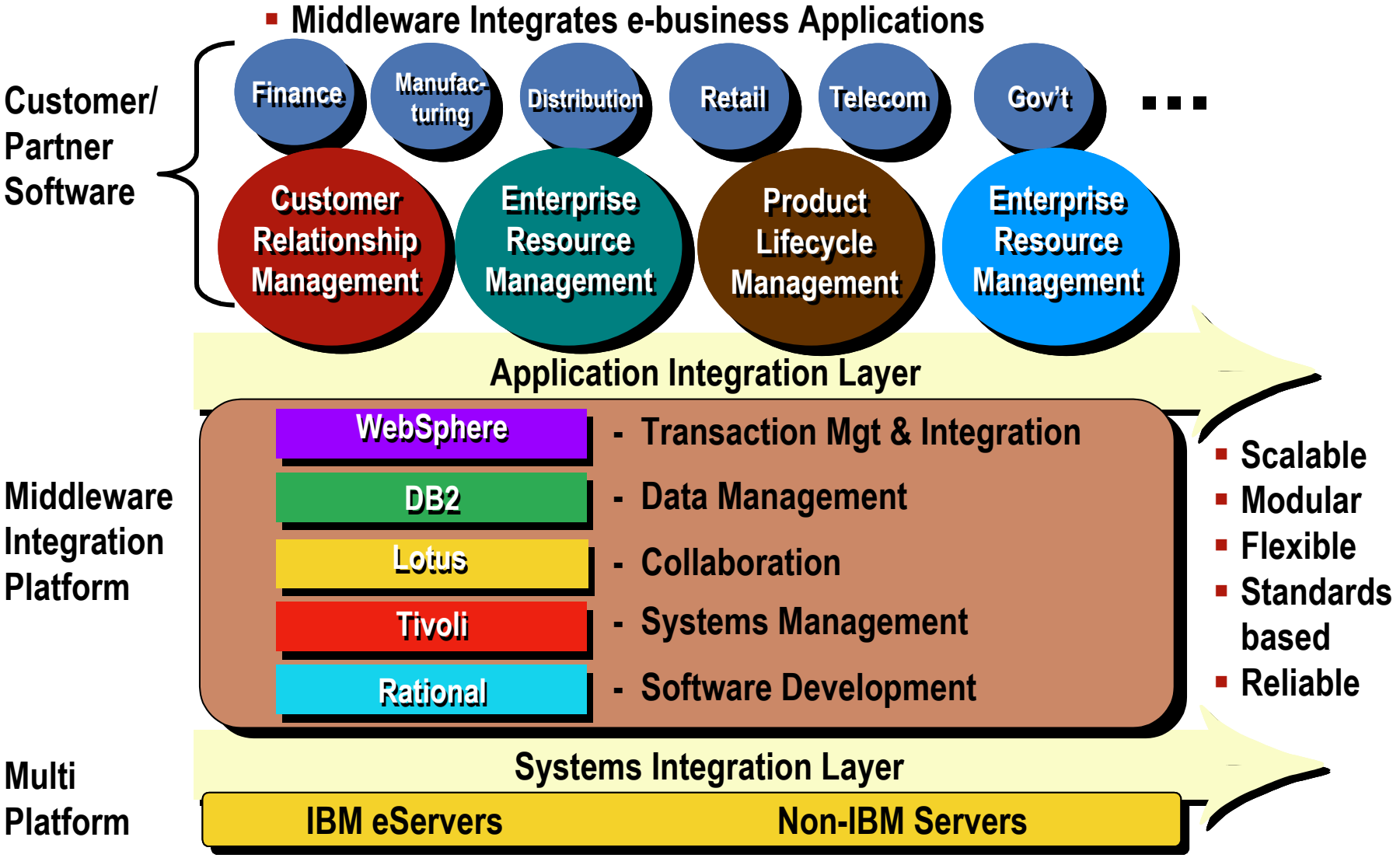
Enterprise Integration and the WebSphere Programming Model

Integration is the strategically most important area to overall business in 2003.

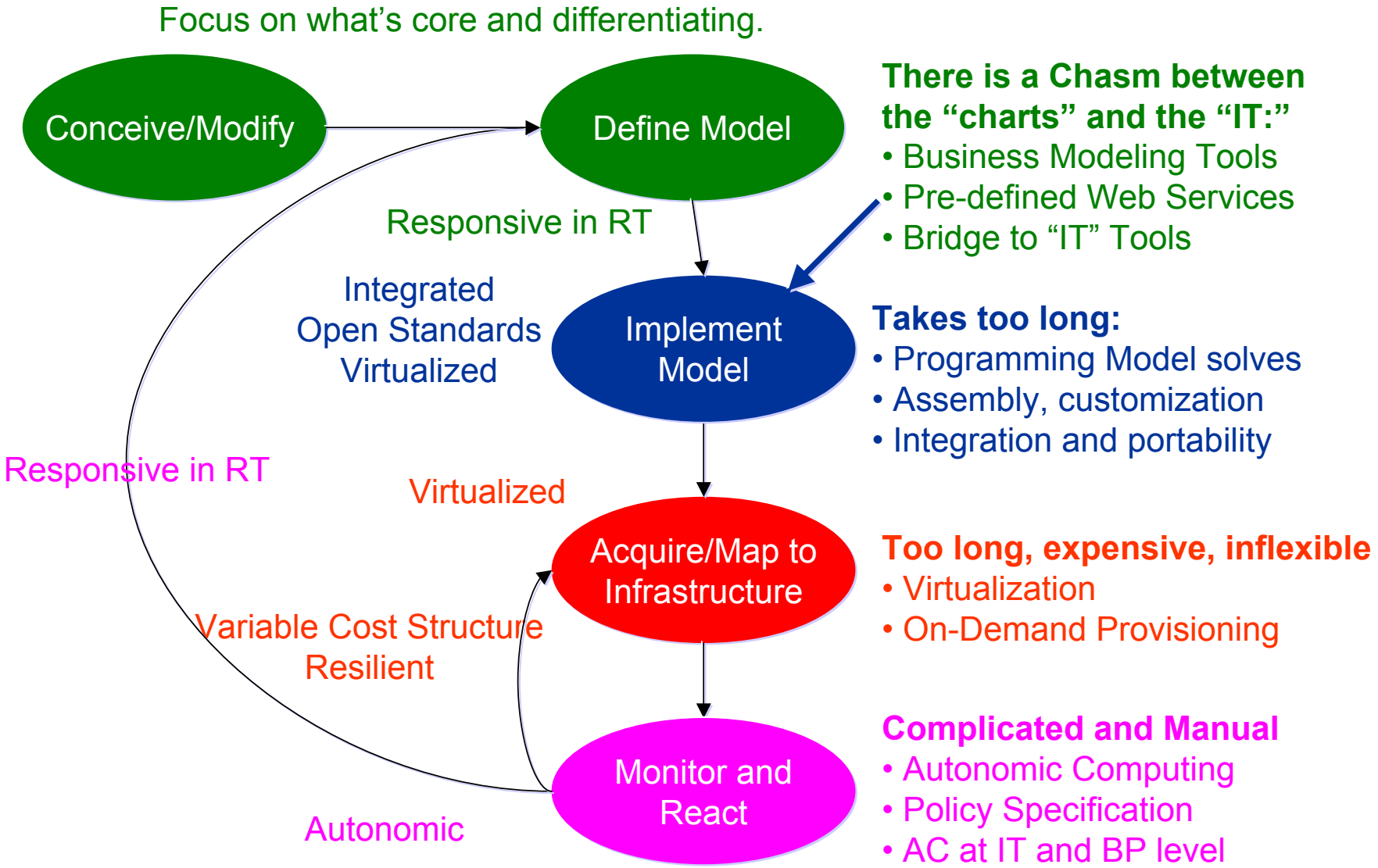


Source: AMR Conference Presentation (11/02)

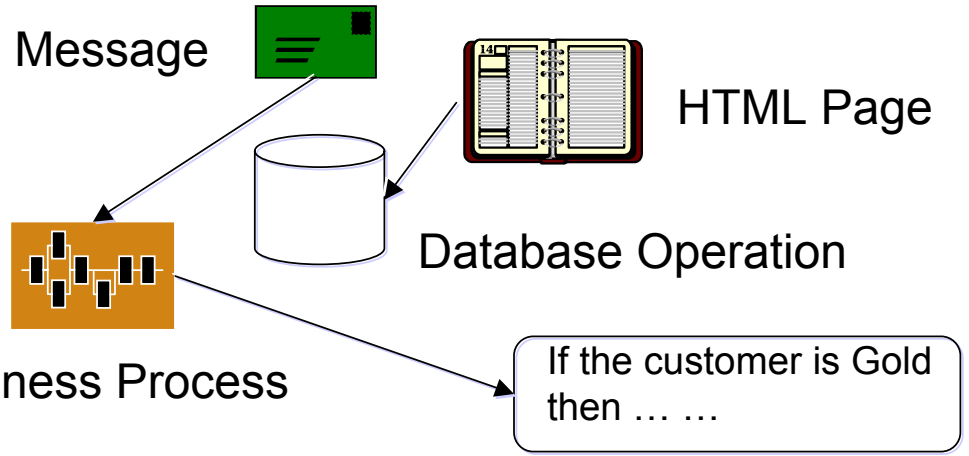
IBM Software Strategy



Application Environment: What do Companies do?



What is a Programming Model?



1a. Parts that people build that define their business (Table – Filing Cabinet)

1b. How parts use parts

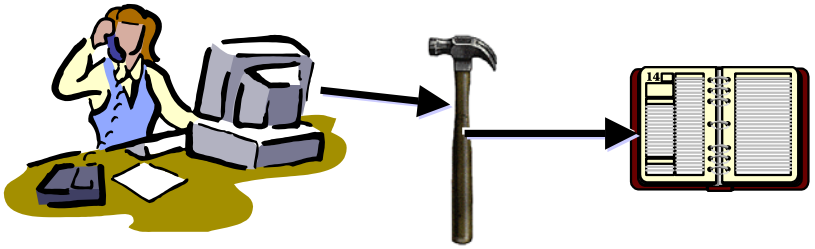
2. A set of roles

Web designer

Business Analyst

3. A set of skills for each role

4. Tools that help



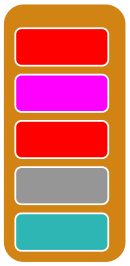
Why have a programming model?

- Simplicity and improved ease of use
  - Limits what any one person needs to know.
  - Allows us to tailor the abstraction and tools to their skills.
- Supports a "programming value chain"
  - Parts use each other in well defined ways.
  - Business define their "core parts" and "out source" others
- Isolates applications from the "infrastructure" details
  - I think in abstractions, e.g. "A table of documents."
  - Middleware maps to physical mechanisms
- How does this help integration?
  - I cannot produce a "host-able application" without a contract between my parts and the "hoster"
  - I do not have to "implement everything from scratch" because I can use and find pre-built "parts" that I need in my implementation
  - Bridges the gap between "core to business" and "running code."

Our programming model enables portability (J2EE, BPEL4WS) and interoperability (Web services)

What's this "Component/Service thing?"

WSDL



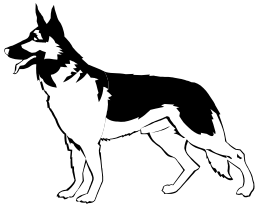
Message M1, M2, … …
Op1, inMsg1, outMsg1, faultMsg1
Op2, inMsg2, outMsg2, faultMsg2
… … …

"Dude, that's an EJB"

"*You whipper snapper, we Invented that in IMS in 1923.*"

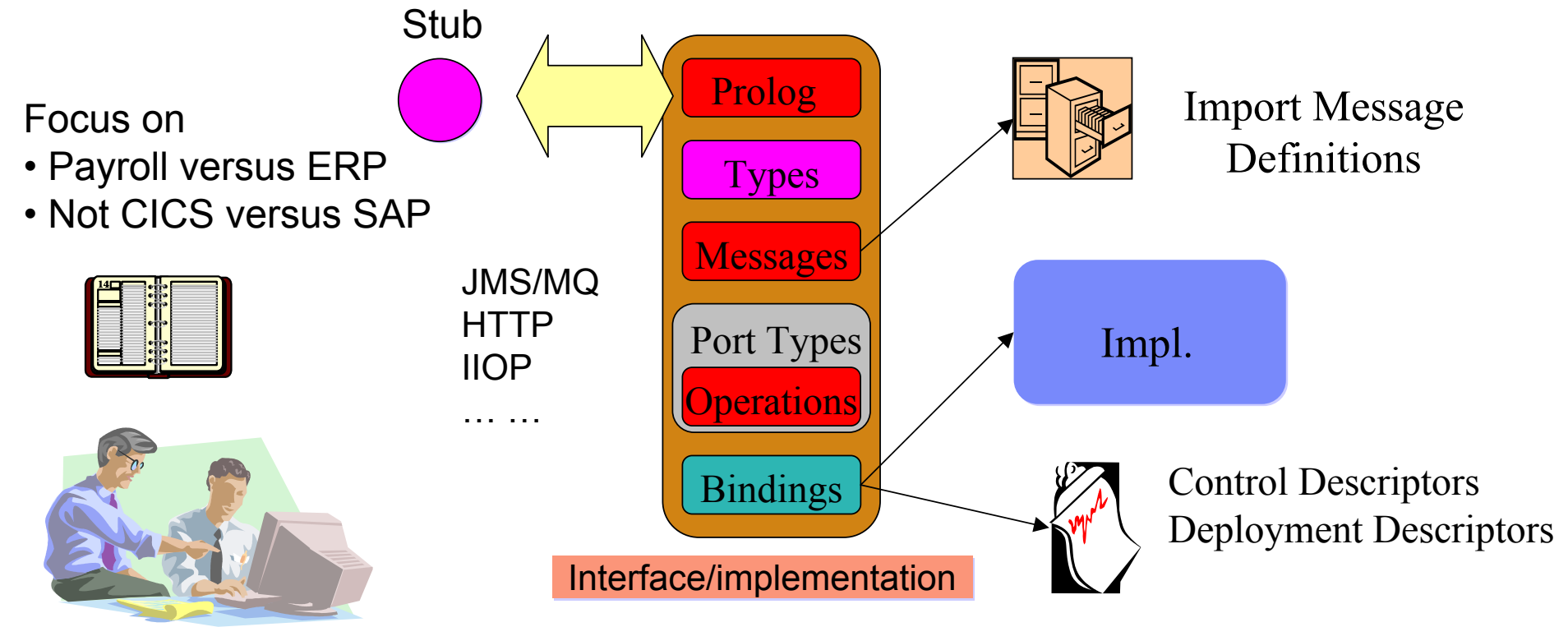
".NET. I like it."

Narr! Das ist eine IDOC



"Those lying IMS swine. That's CICS."

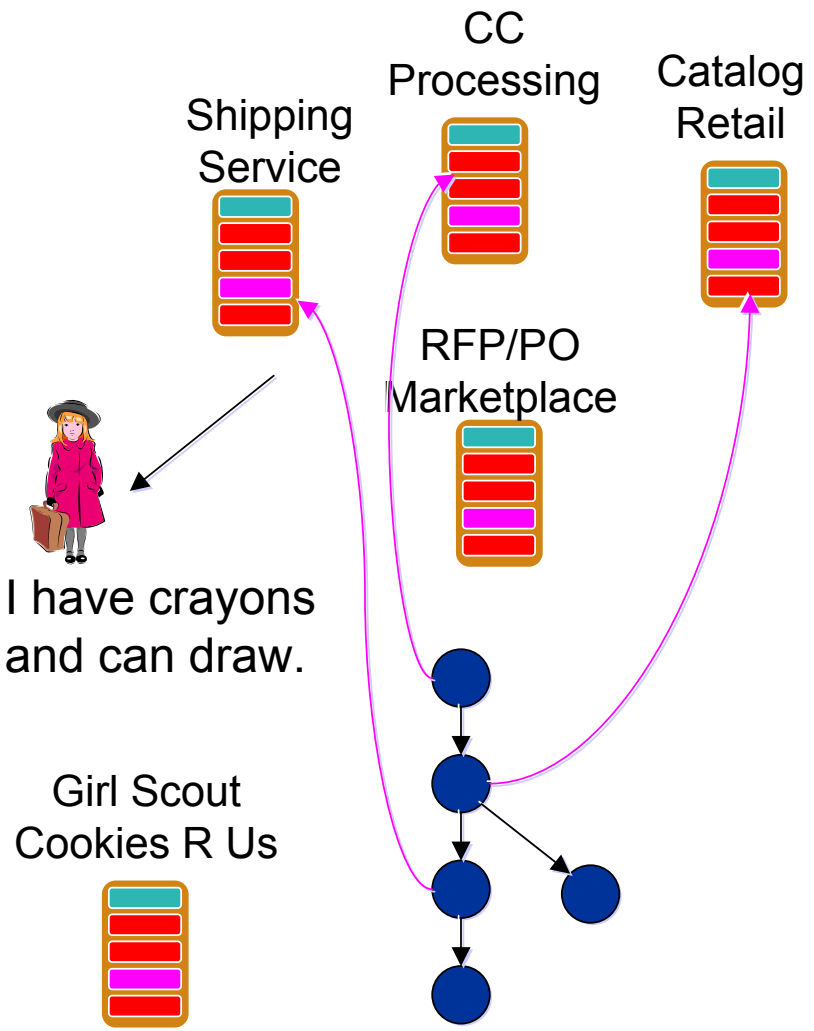
Component/Service Model

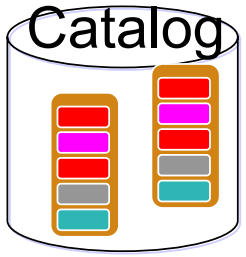


- Simplifies through a common, open, interoperable model.
- Programmers focus on business aspects, not infrastructure aspects
- Supports interoperability between “tools” and “middleware”
- Enables a “computing supply chain”

Building Components is Easy

Making interesting components is child's play.



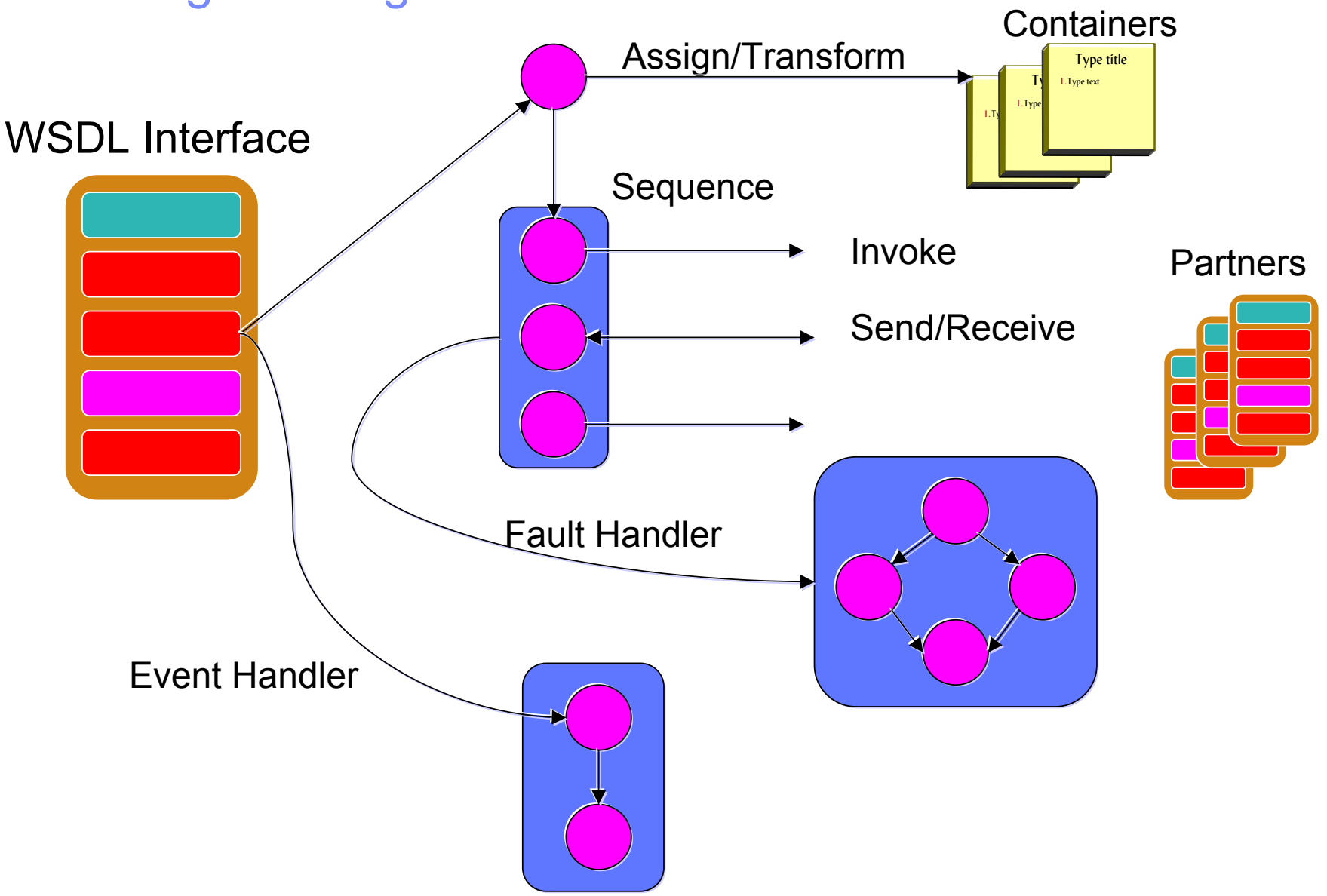


UDDI

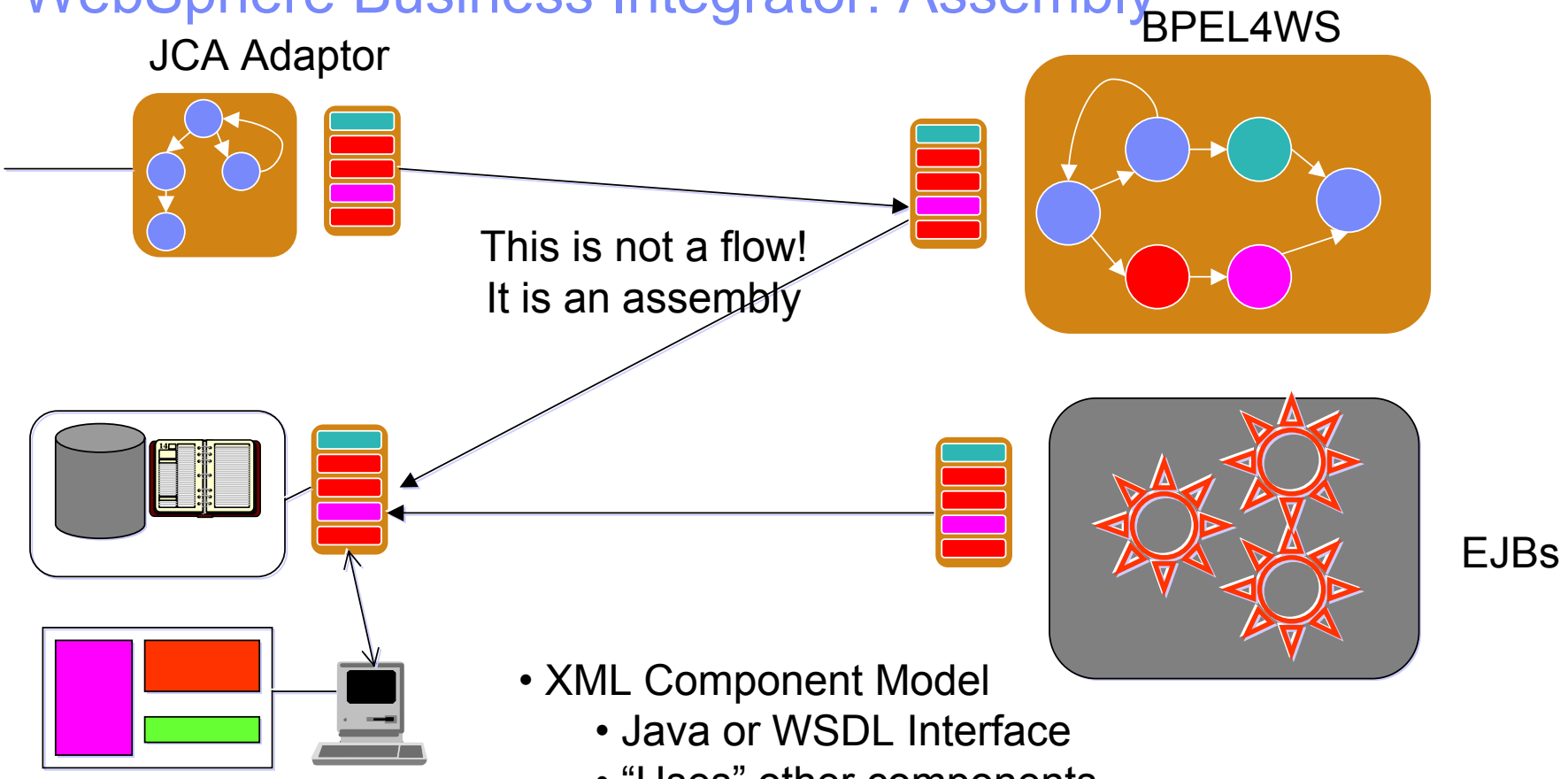
Tools

- Pre-built components
  - Intra-enterprise
  - From partners
- Simplify programming to
  - "Flow charts"
  - Really enable "BPM Tools" and BCS

The Programming Model: BPEL4WS



WebSphere Business Integrator: Assembly



- XML Component Model
  - Java or WSDL Interface
  - "Uses" other components
  - Logically wire together to build a "solution"
- Several Component "Types"
  - EJBs, BPEL flows, Application Adaptors
  - People Facing Components
  - Mediations in the bus

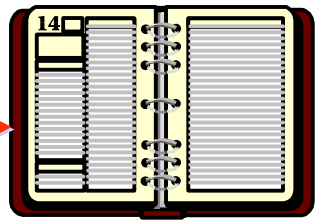
Application Development Tools

Business Analyst: Model “Things” and “Processes”

HoloSofx

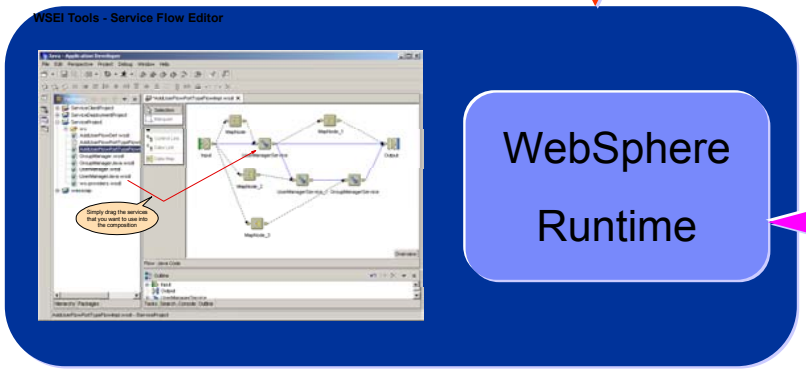
UML (Rational)

“UDDI”



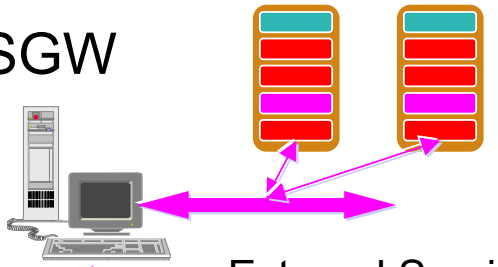
IT Professional

WebSphere Studio



Unit Test

WSGW

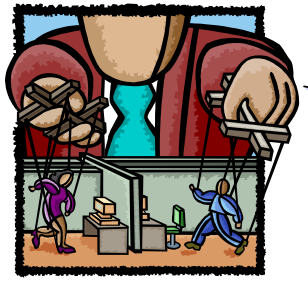


External Services

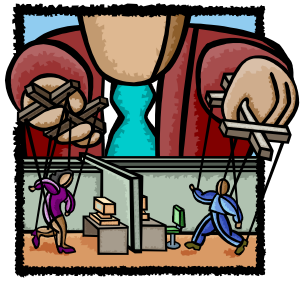
Internal Services

Customization

I am an ISV.
I want to sell my solution



That's cool,
but not quite right.



Neat, I externalized
my policies.

Applications have "things"
and "verbs."

Commerce

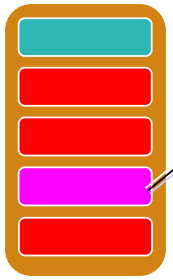
CRM

I have a pencil.
I can fill out forms.

Programming Model: Business Rules

O.K., you can visually assemble new services from existing ones.
How do you build the basic ones?
How does an ISV build a service/component that the customer can customize?
What if my application is hosted? Each enterprise is different

Well, sometimes someone writes code



```
public boolean addLineItem(PO p, Item i) {
        if (p.noElements() > 50) ||
          (p.value() > 10000) {
                throw …
        }

        p.add(i);
}
```

Could be COBOL, C, EJBs, … …

This is really dumb!

Programming Model: Business Rules

```
public boolean addLineItem(PO p, Item i) {
        if (self.getHelper().isValid(p,i) {
                p.add(i);
        }
        else {
        }
}
```

The "Strategy Pattern."

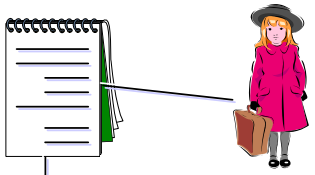
```
public boolean addLineItem(PO p, Item i) {
        helper = WSLookUp(/…/…/…);
        if helper.isValid(p,i) {
                …       …
        }
```

Now, you're golden,
Or well violet.

Programming Model: Business Rules

Bind using
- Date/Time
- Application Version
- Message Predicate
- … …

Required "Services" document POVs



Call

Simple rule templates
Java, JavaScript
Rule Engines
… …

Prolog

Types

Messages

Port Types

Operations

Bindings

Mediate

Customize
Tailor in a hosted site

Impl

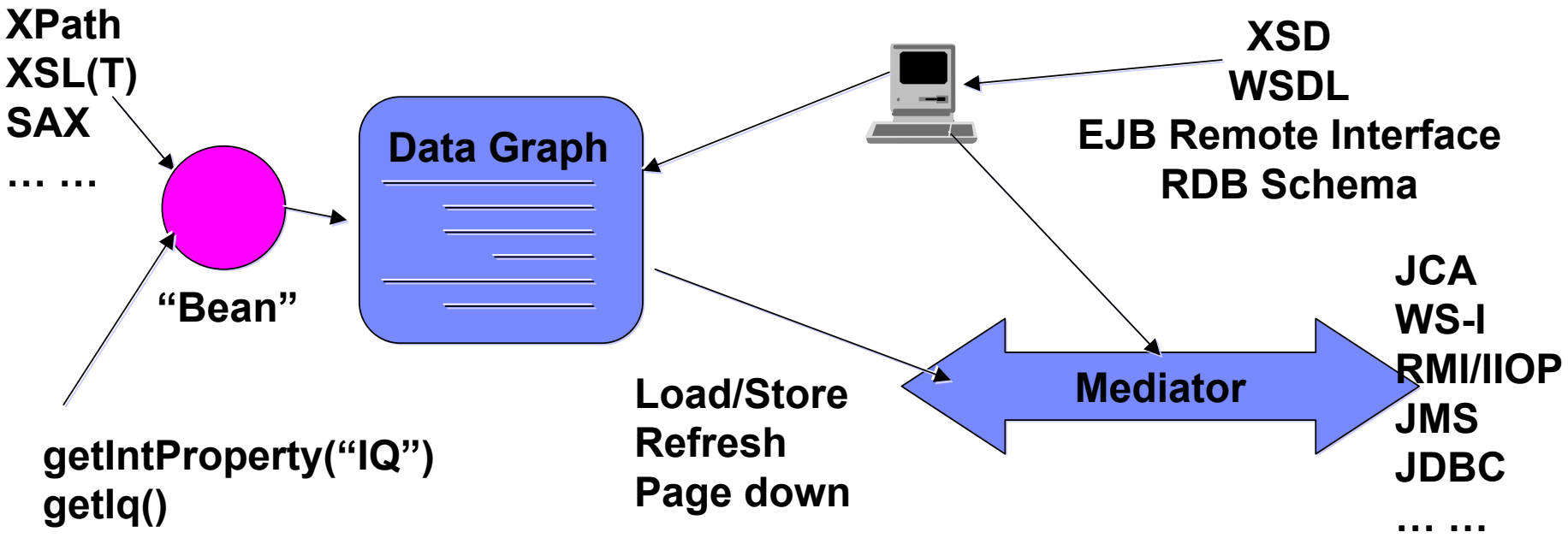
Web Service Binding (Find)

POV = RPC Stub

Control Descriptors
Deployment Descriptors

Programming Model: WebSphere Data Objects

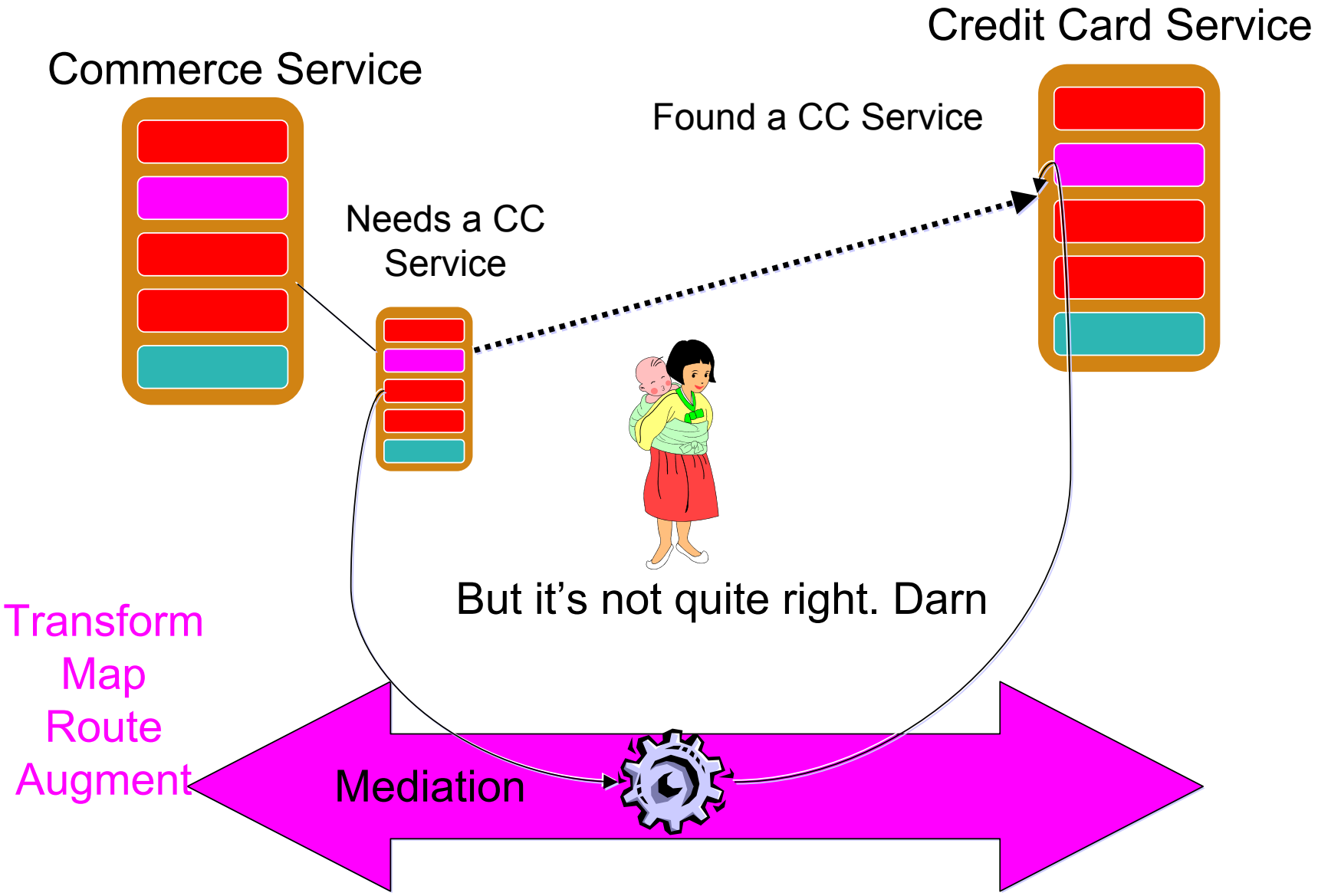
- Once upon a time we had “commands.”
- We also had data access beans.
- Then we had three or four types of EJB Access Beans
- Then we had JROM, JAXB, etc.
- We also have XSL(T), Java mapping code for transformations
- This seems rather cruel to our customers.



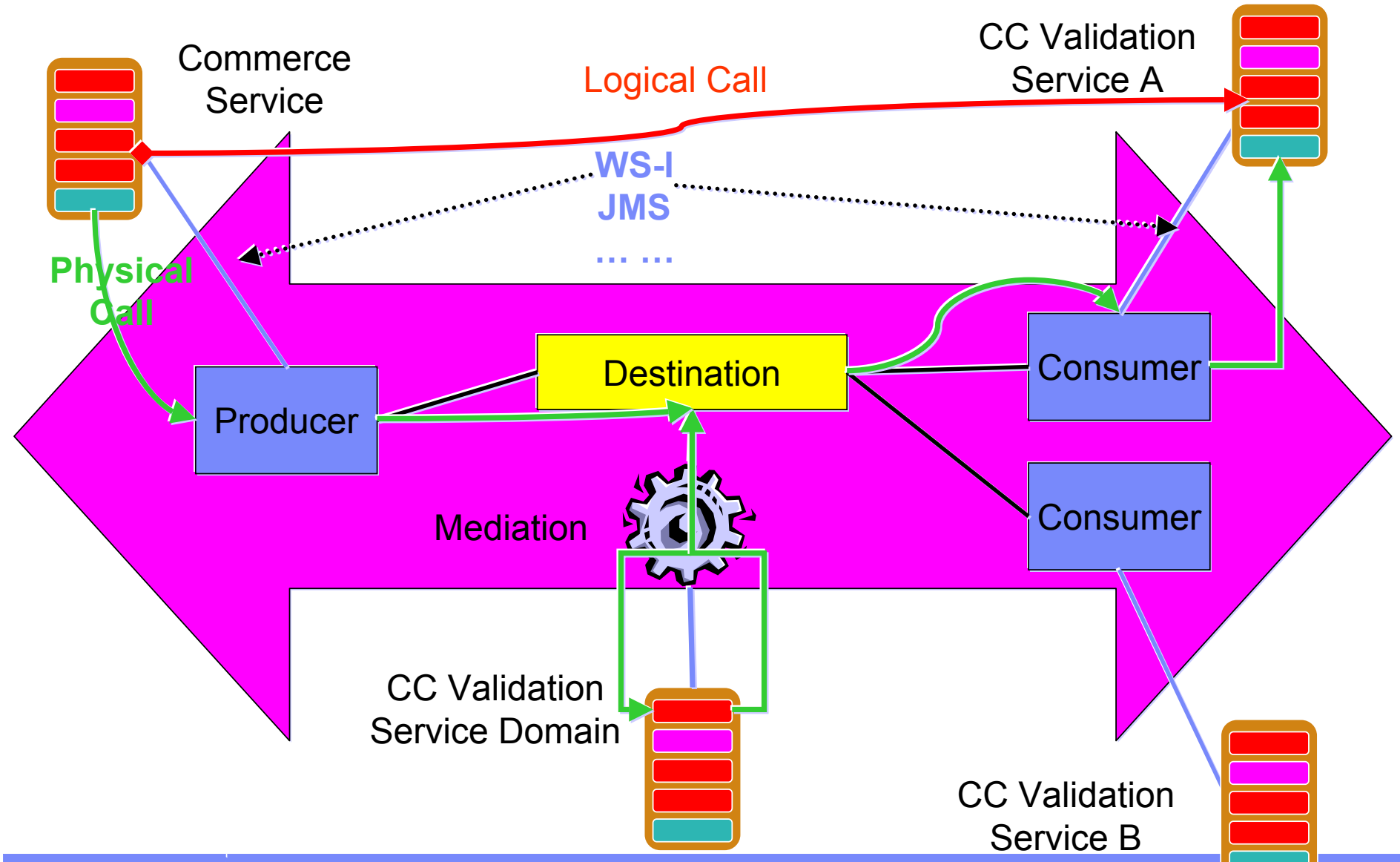
Programming Model: WebSphere Data Objects

- **Uniform abstraction and realization for "sources of data"**
  - WSDL operations
  - Java Objects (EJB results)
  - Rowsets
  - XML documents; "Business Objects"
  - … …
- **Provides**
  - Optimized in memory data structure
  - Both XML access and Java access
- **Some advanced functions**
  - Caching outside of "connection" and collision detection (with PM)
  - On demand fetch of large data in chunks
  - Interoperate with Microsoft .NET Web services using ADO.NET
  - Validation and metadata association with Data Graph
- **Uniform**
  - Model for "data objects" for WBI, BPEL4WS, Message Broker, Web services, Command Cache, etc.
  - Model for "transformations" and "maps."

Services Oriented Architecture and Mediations



Service Bus is …



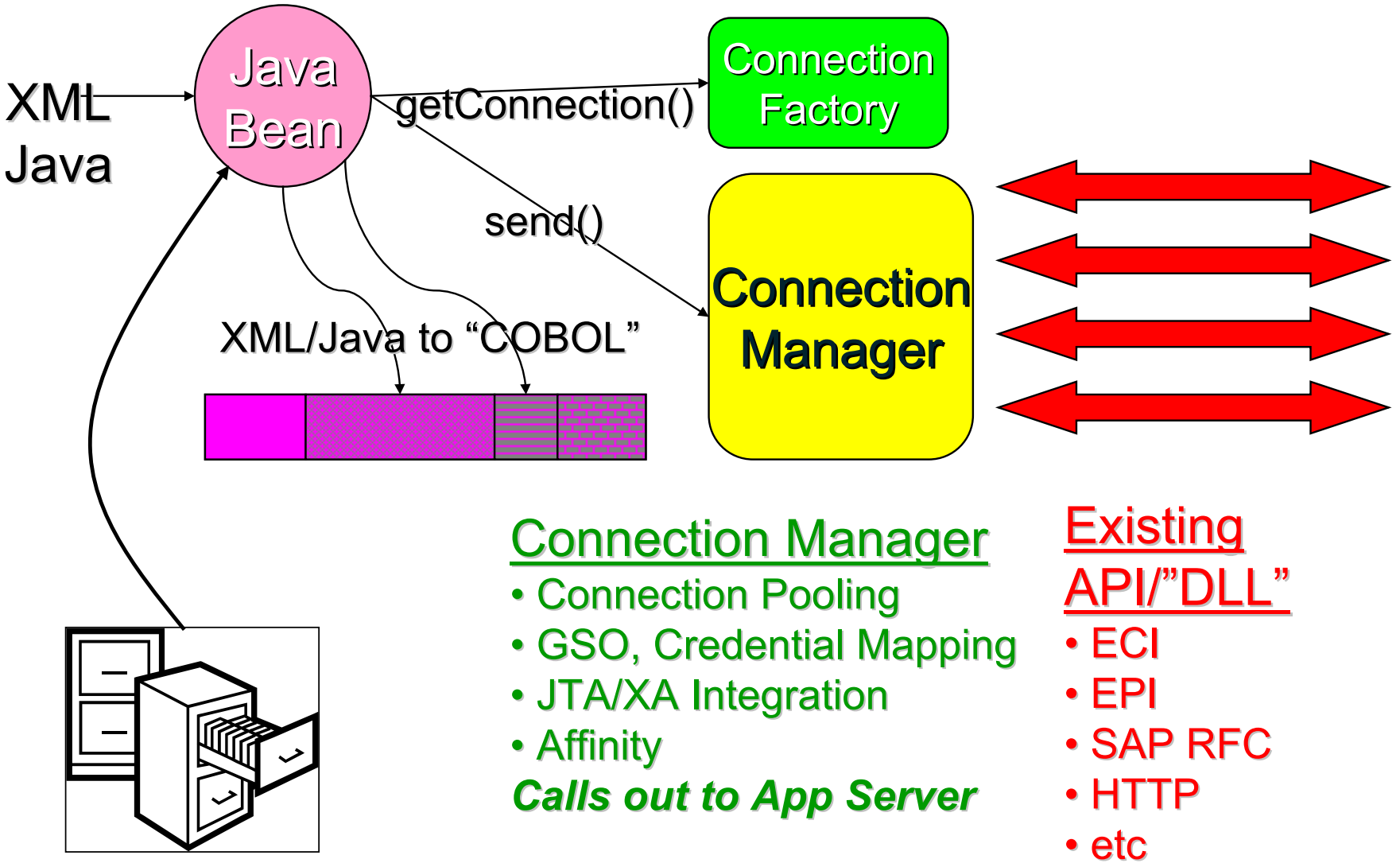
Well, that was as clear as mud (Try animating it)

- **JetStream is**
  - JMS for WebSphere (Development, Single System, Cluster, Networked)
  - Integration with MQ Networks
- **Service Bus adds**
  - Support for multiple protocols (WS-I, IIOP, … …)
  - Mediators (Observers)
- **O.K., your still babbling. What's this mediation thing**
  - A ***Destination*** is like a Queue
  - Has message producers and consumers.
- **I asked what a mediation is? You're not cooperating**
- **A mediation transparently interposes between the producers and consumers, and "calls a third web service."**
  - Transformations, Routing decisions
  - Logging, Auditing, …
  - Anything that can be a Web service
- **Oh, and this stuff will all run in the bus.**

Service Oriented Architecture and Mediators

- What is a Service Oriented Architecture?
  - In the old days, we “linked” by address.
  - Then we linked by “name.”
  - Now we link by “capability”
    - What is your business function?
    - How well do you deliver it?
- What is a mediation?
  - Helps map clients to suppliers.
  - Impedance mismatch and added value transparent to the callers
- What are the values to “customers”
  - “On demand” tailoring of actions when messages arrive. Added after the solutions deploy.
  - More robust, and can find the best supplier *at the time I need it.*
  - Enables a new type of “in network” provider that matches requestors to supplier, “On Demand.”

Application Integration



Connection Manager

- Connection Pooling
- GSO, Credential Mapping
- JTA/XA Integration
- Affinity

Calls out to App Server

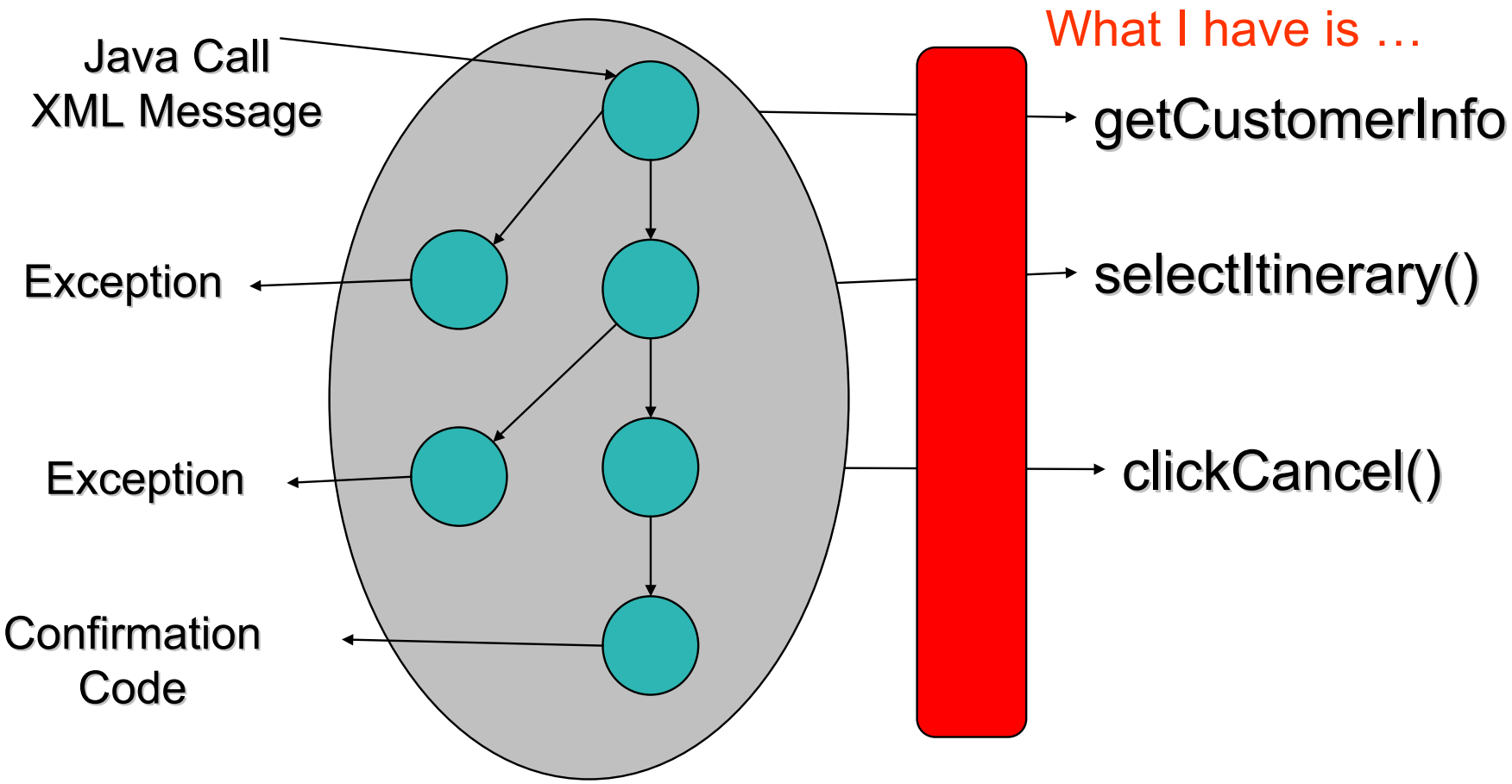
Existing API/"DLL"

- ECI
- EPI
- SAP RFC
- HTTP
- etc

Adaptors

What I want is …

public String cancelTrip(String who, Date when) throws … …



Connectors and Adaptors

- Some things are just not based on all the J2EE, WSDL, BPEL4WS, telepathic, self-describing, … … technology
- Connectors and Adaptors solve the problem
  - Preserve existing investment
  - Automate the process of integrating the existing with the "new"
    - Security
    - Transactions
    - etc.
- Allows existing systems to
  - Appear like a Web service
  - Participate in mediation, SOA, flow charts, etc.
- Publish "intra-enterprise" systems into value chain hubs and business partners, and vice-versa.

Container Management for Services

```
public double deposit(double funds) {

        balance = balance + funds;
        return balance;

}
```

What you want to do

Container Management for Services

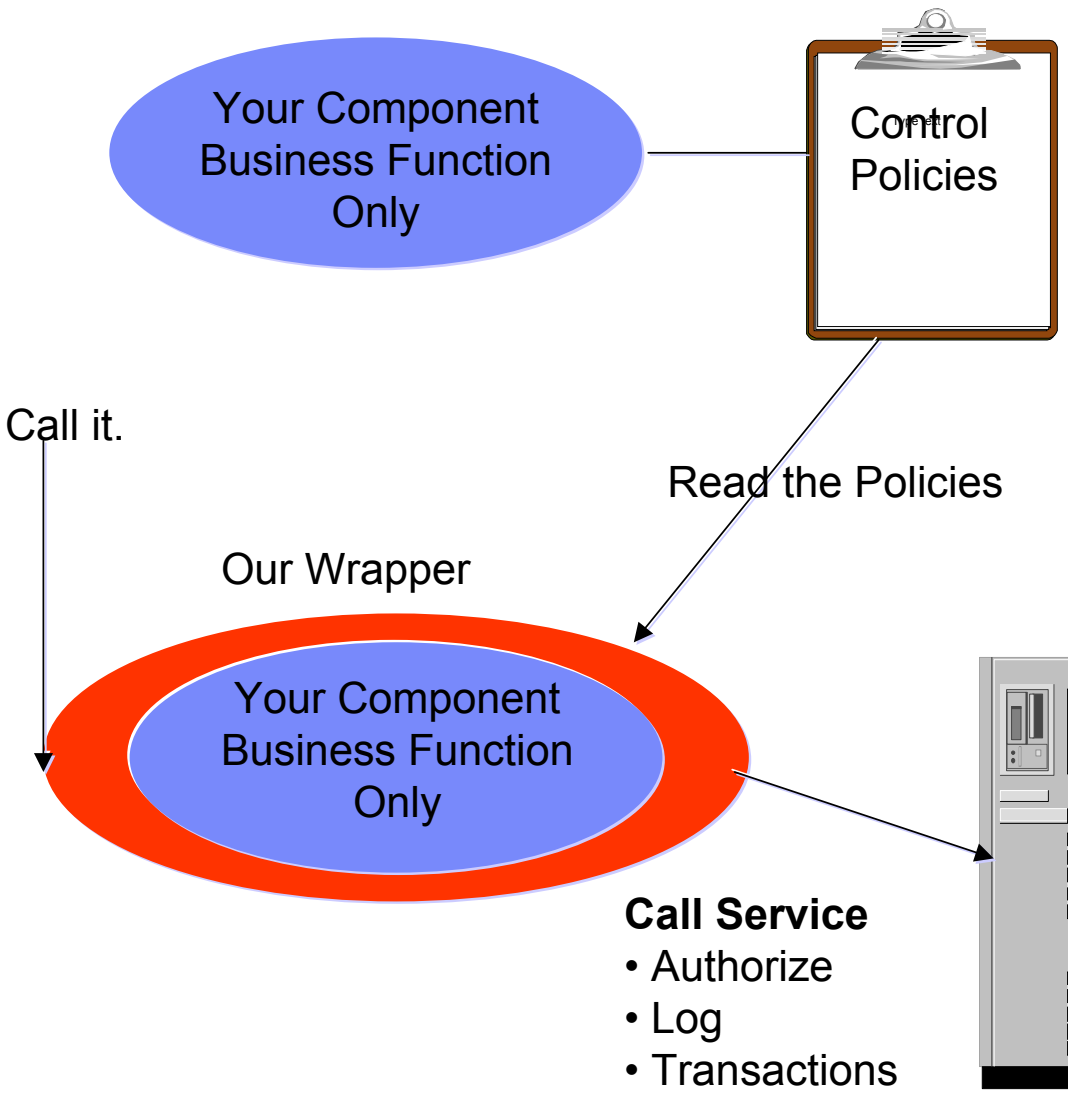
```
public double deposit(double funds) {
        java.jdbc.Connection c = getConnection();
        tran = current.getTransaction();
        if (tran == null) {
                tran = current.begin();
        }
        Principal who = context.getCallerID();
        if (authorized(who, self, fund) == false)
        {
                // Oh no … …
        }
        balance = balance + funds;
        tran = current.commit();
        return balance;

        // Now undo all of that stuff if it fails
}
```

What you must do.

What you want to do

Container Management



"Programmer"

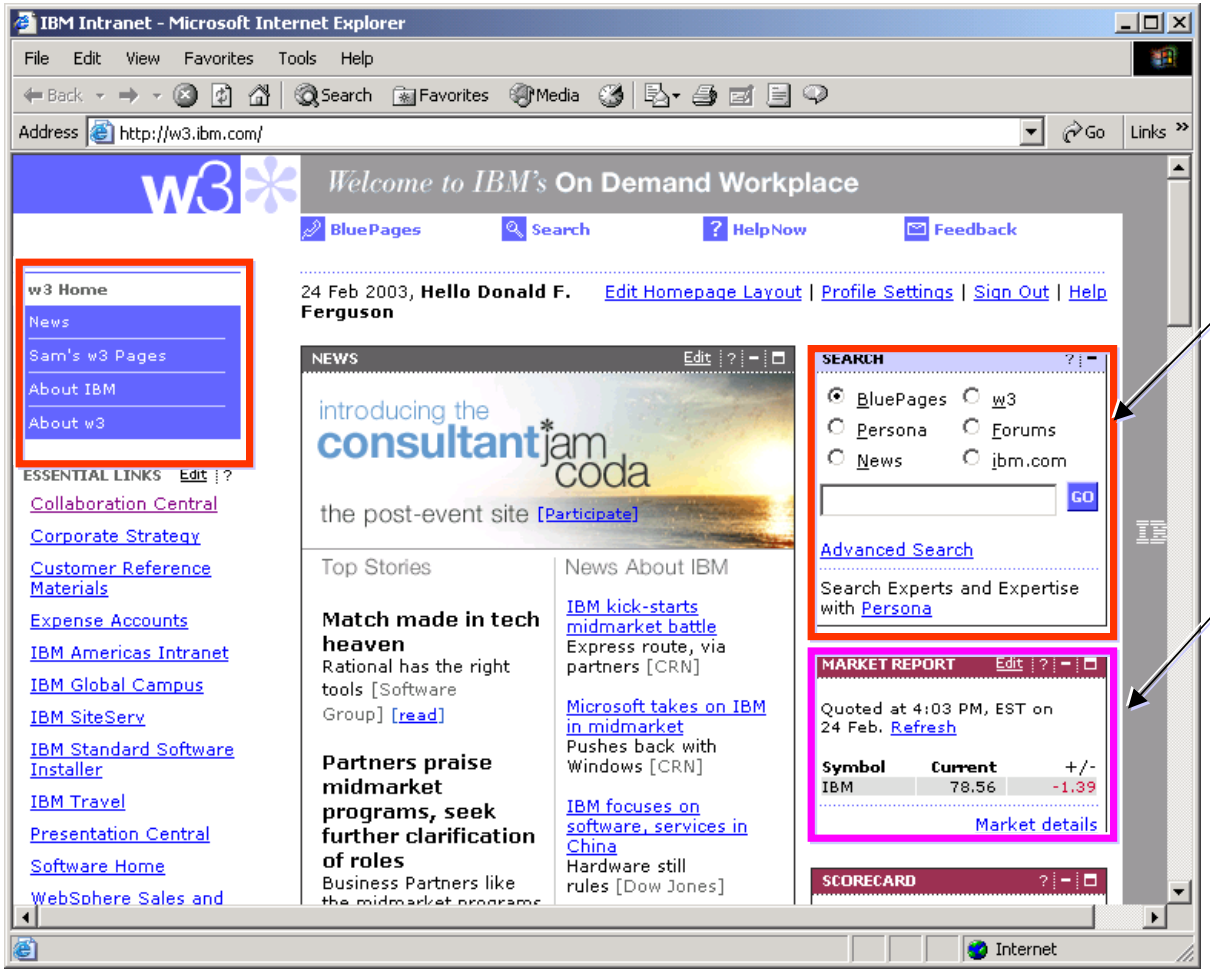
- Writes only business logic
- Declares what he wants

Middleware and tools get
It right.

Benefits:

- Vastly simpler, quicker, robust
- App. runs anywhere

Sometimes you gotta talk to people



- Personalize
  - The site tells you
  - Who you are?
  - What they have?
  - Marketing rules?
- Customize
  - You tell the site
  - What you want

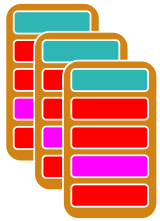
Portal and Personalization

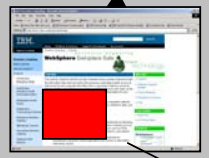
Remote Portlets

Controller

Registration
State
Rules

The Bus





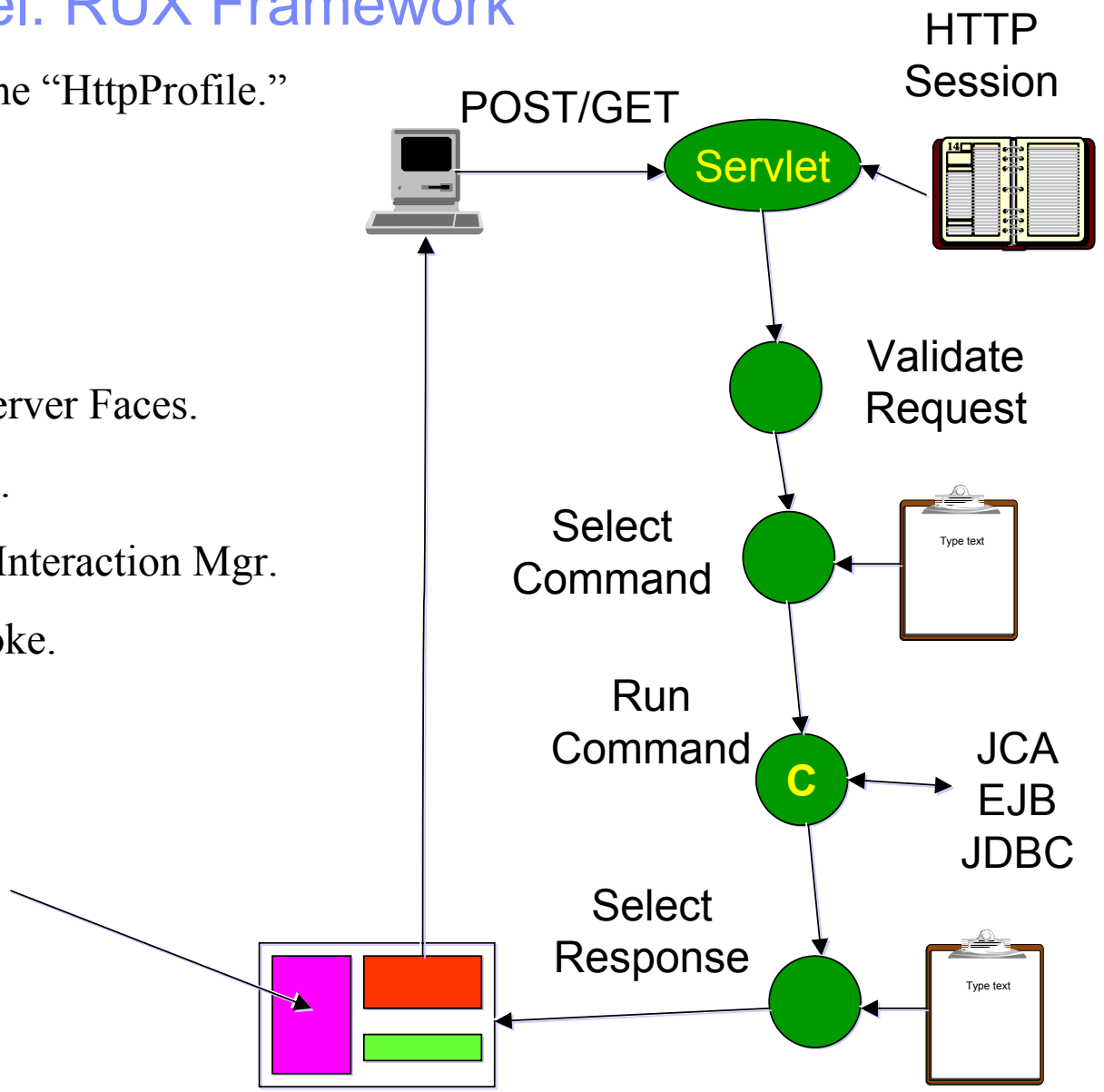
Actions

Select
Template

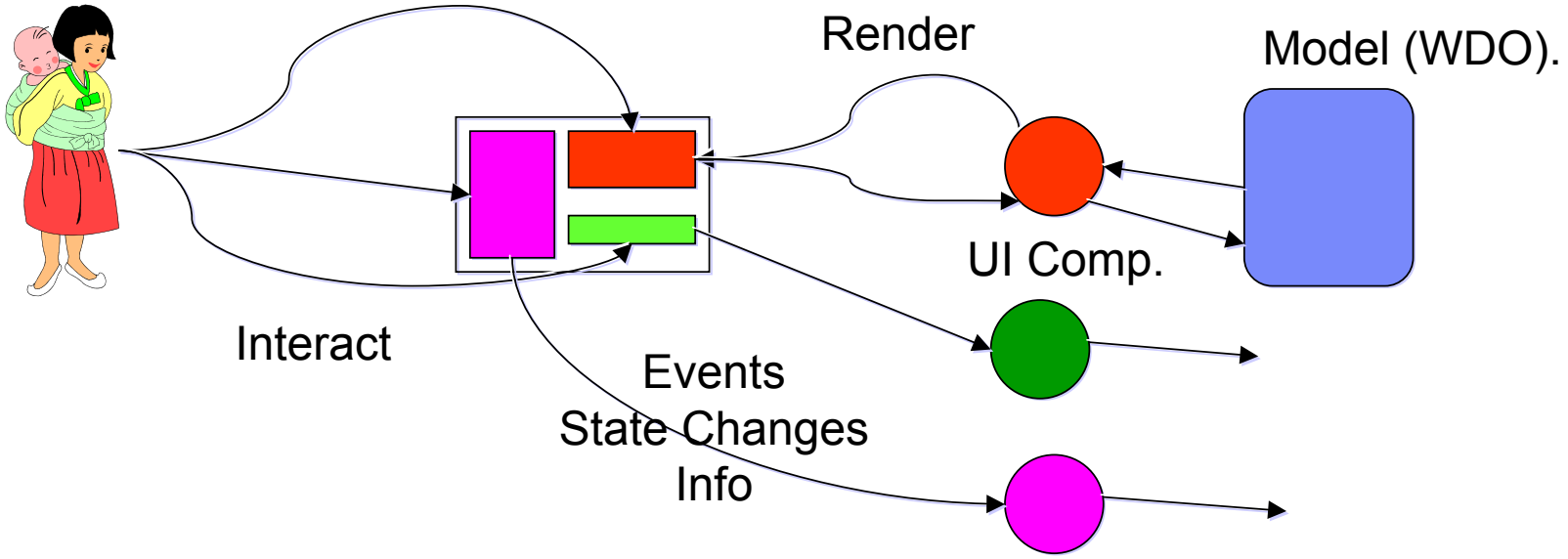
People Props
Data props.
Rules

Programming Model: RUX Framework

- Once upon a time we had the “HttpProfile.”
- Then came Struts.
- Then came Jade (EAD4J).
- Then came Portlets.
- Now we are getting Java Server Faces.
- This time we really mean it.
- Oh yeah, I forgot the WBI Interaction Mgr.
- Some people can’t take a joke.



Programming Model: Reach and User Experience



- Java Server Faces: Think *Visual Basic™ meets HTML on LSD*
- A page contains a set of UI controls that interact with models
  - Render UI properties into output format in "right place"
  - Handle updates when event happens on page
- Mechanism for routing information from the "changed page" to the right UI control
- A set of predefined UI controls (Notebook, Tree, etc.)

Programming Model: Reach and User Experience

- O.K., this is getting freaky.
- How does this relate to Struts, Portlets and JSPs?

1. Portlets are “independent applications” on a page
   - Internally they are a spooky mix of JSPs, Servlets, UI controls, etc.
   - They understand how to resize, events between Portlets, WSRP, etc.
2. The JSF UI components are JSP tag libraries. A JSP defines the list of “components” and their placement.
3. Struts provides an extensible, replaceable controller pattern for page sequencing.
4. Trying to move existing “frameworks” to provide “content” within the overall model.
5. We are doing PM, runtime and tool integration to bring this together.

Oh, we are going to run this on the client also. And in the network.

Programming Model: Reach and User Experience

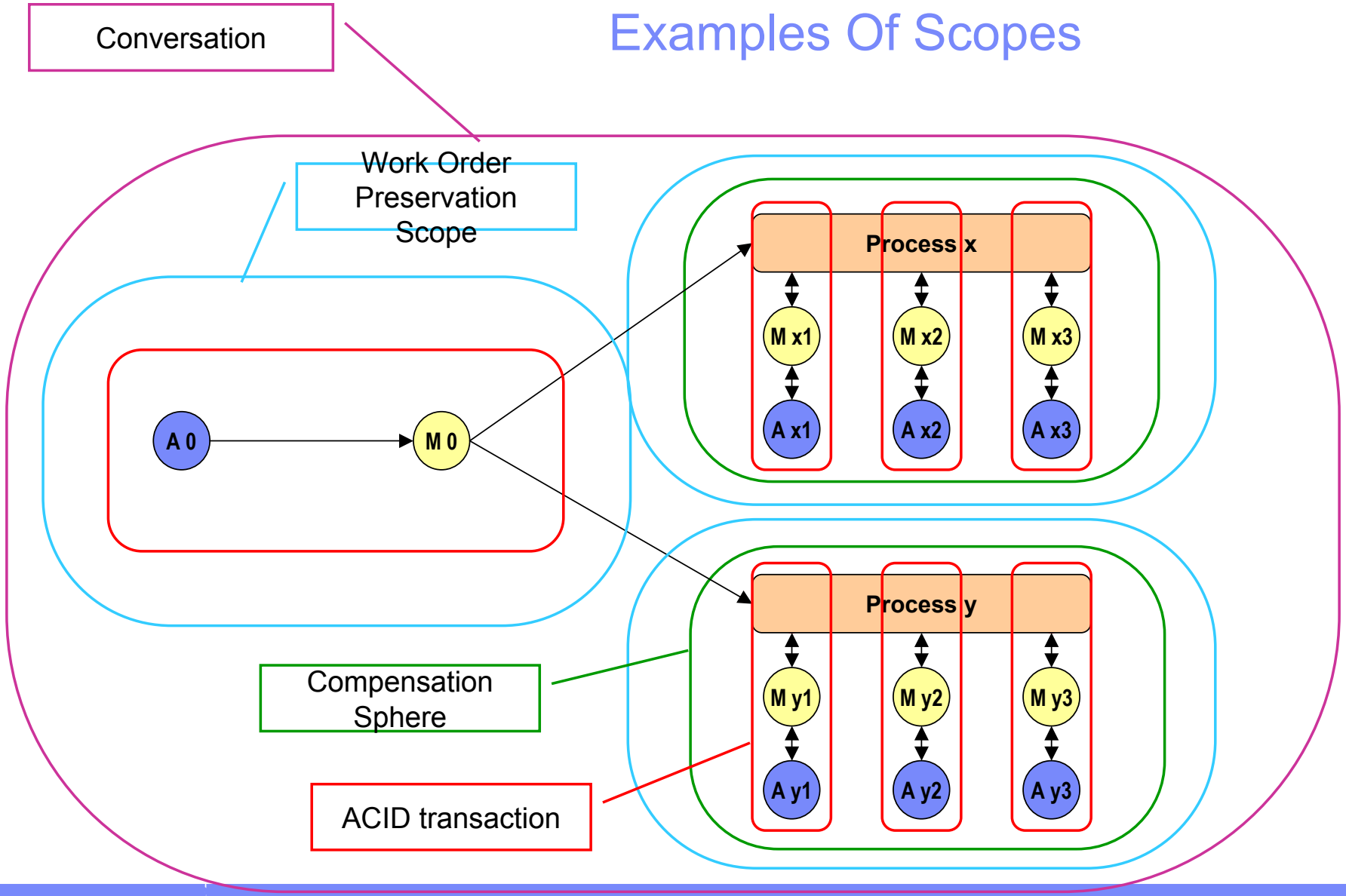
- SOA, BPEL, etc. allow “On Demand” business processes
- Sometimes, some one is going to need to *personally* do something.
- The “RUX” programming model provides an “On Demand” Workspace
  - Dynamic discovery of “applications”
    - Business Processes
    - Portlets from partners
  - Business rules allow
    - Site to show what you need to see.
    - You to declare what you would like to see
- Standards (Portlets, HTML, JavaServer Faces, Struts) provide a standard model for customers, SIs an ISVs to contribute to each persons workspace.

Programming Model: Business Transactions

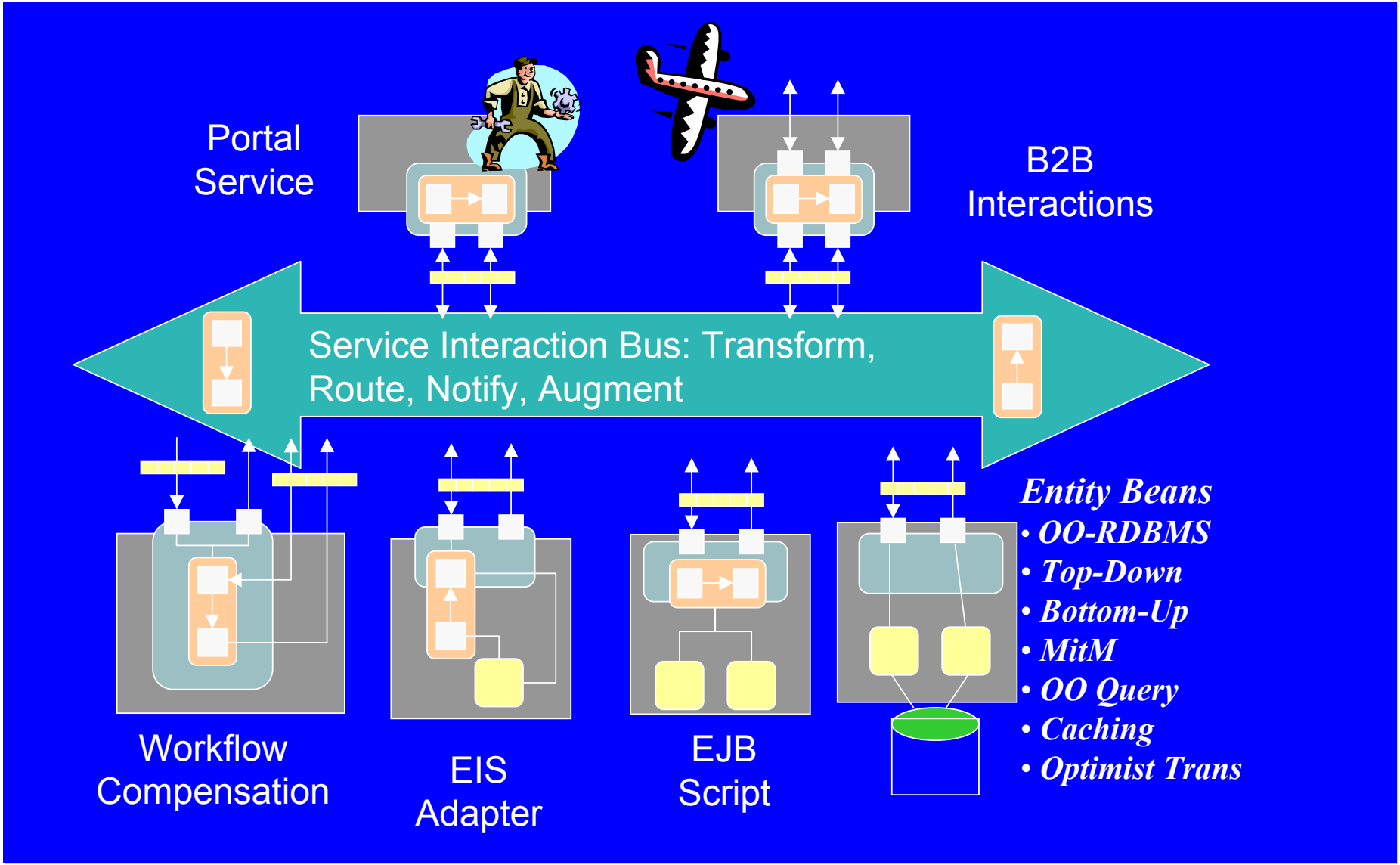
- Long-running scopes with nesting
- Different scope types:
  - Conversation
  - Work Order Preservation
  - Compensation Sphere
  - Locking Scope
  - Activity Session
  - ACID (J2EE Transactions)
- Parent scopes are not suspended when child scopes are started
  - Thus, context is available, e.g., for monitoring
- Scope begin and end follows the J2EE pattern. Delcared in Component deployment descriptor.

Programming Model: Business Transactions

Examples Of Scopes

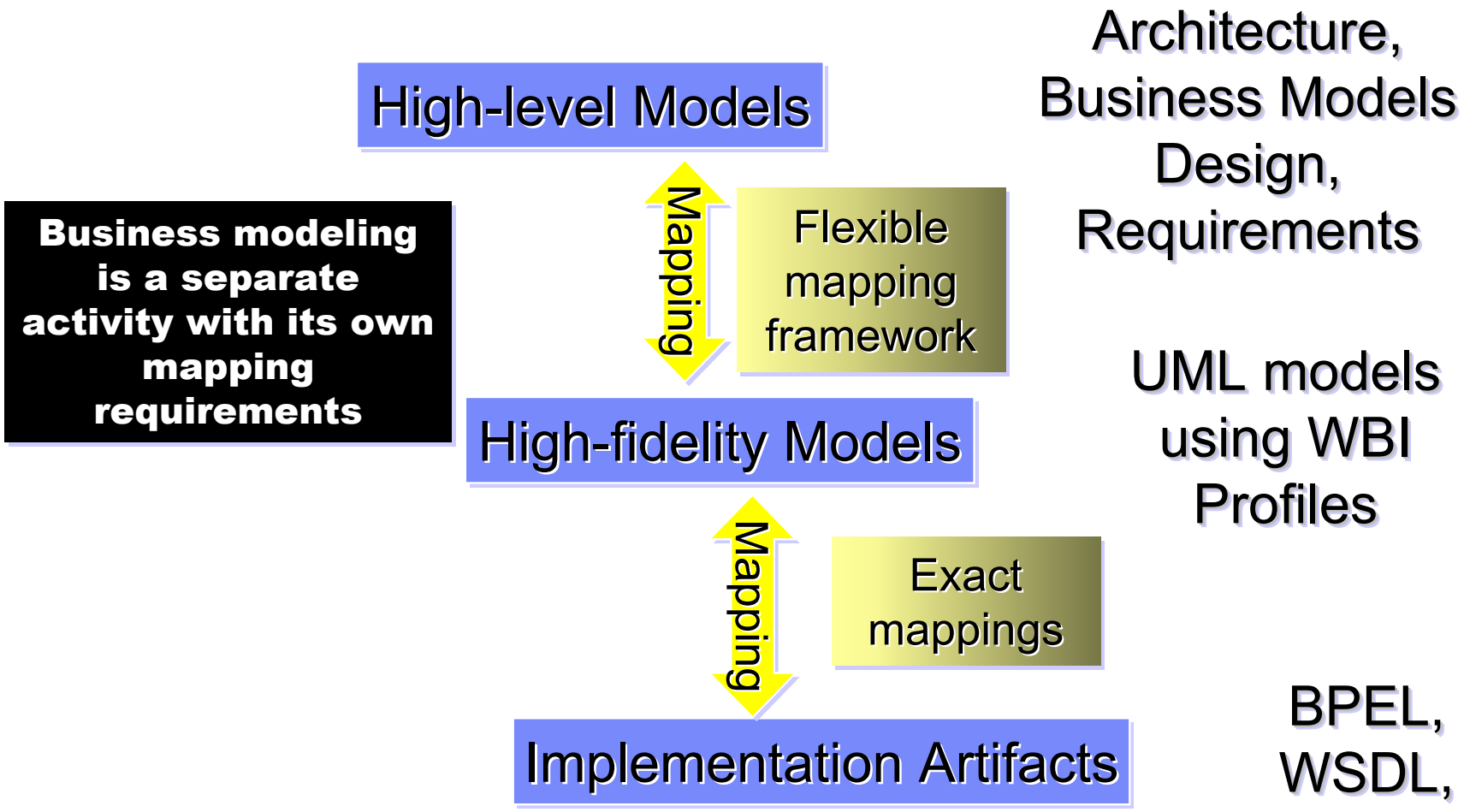


End to End Model – All in a Bag

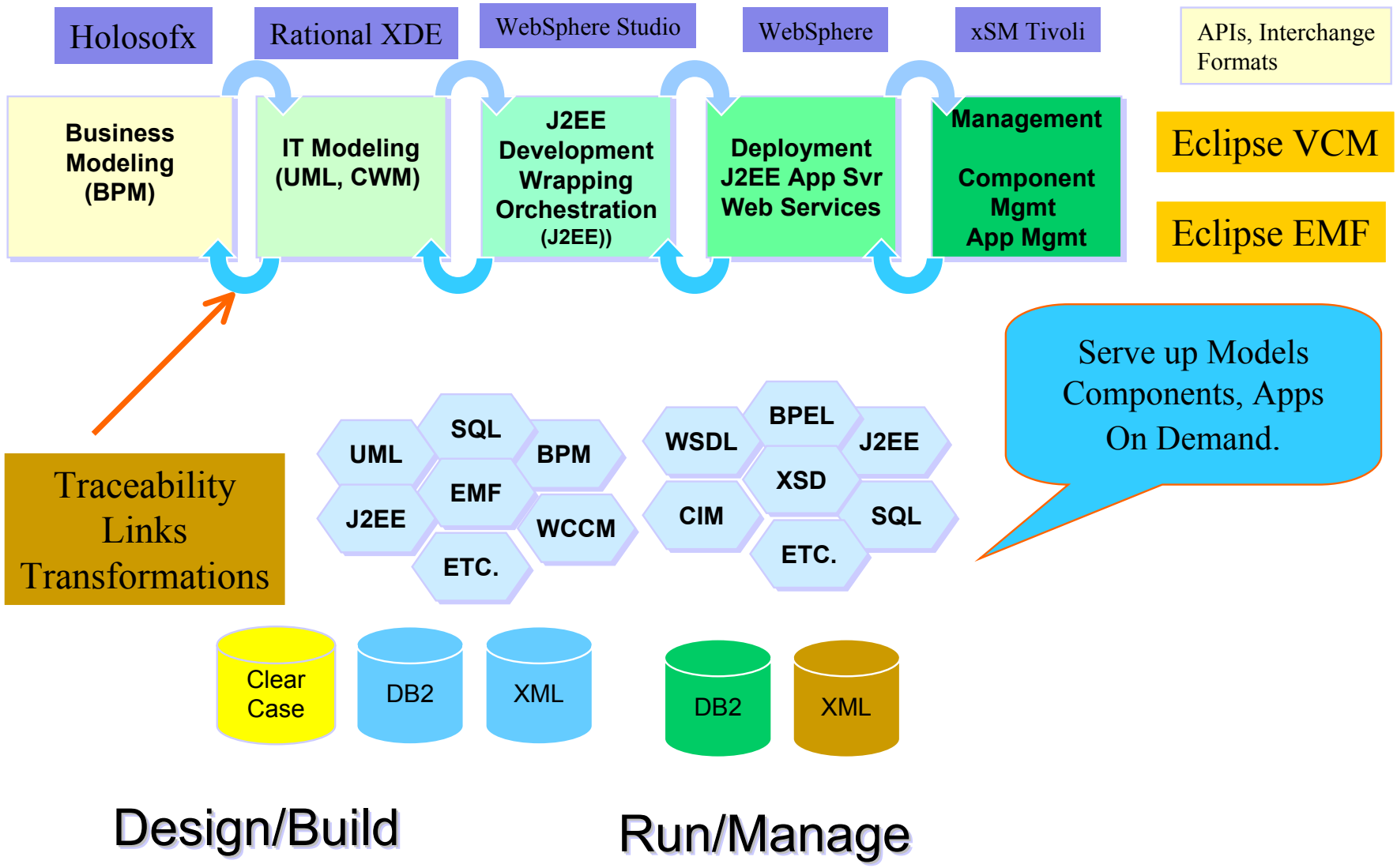


Tooling and Methodology

Different Levels of Abstraction supported by Tools

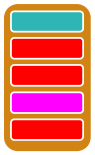


Model Driven Business Integration – Tool Chain Example

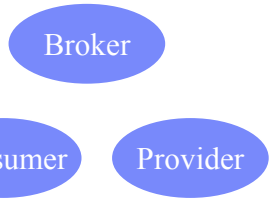


Programming Model Elements

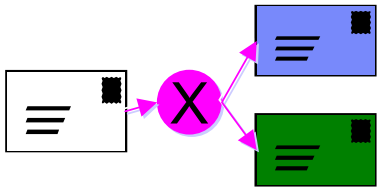
Component Model



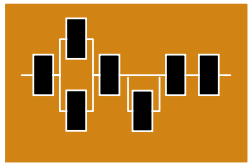
Service Oriented Binding



Mediation

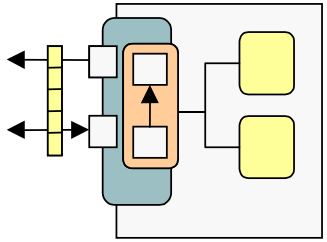


Flow Composition

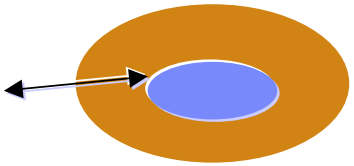


Policy Enabled Business Rules

Connector/Adapter



Container Based



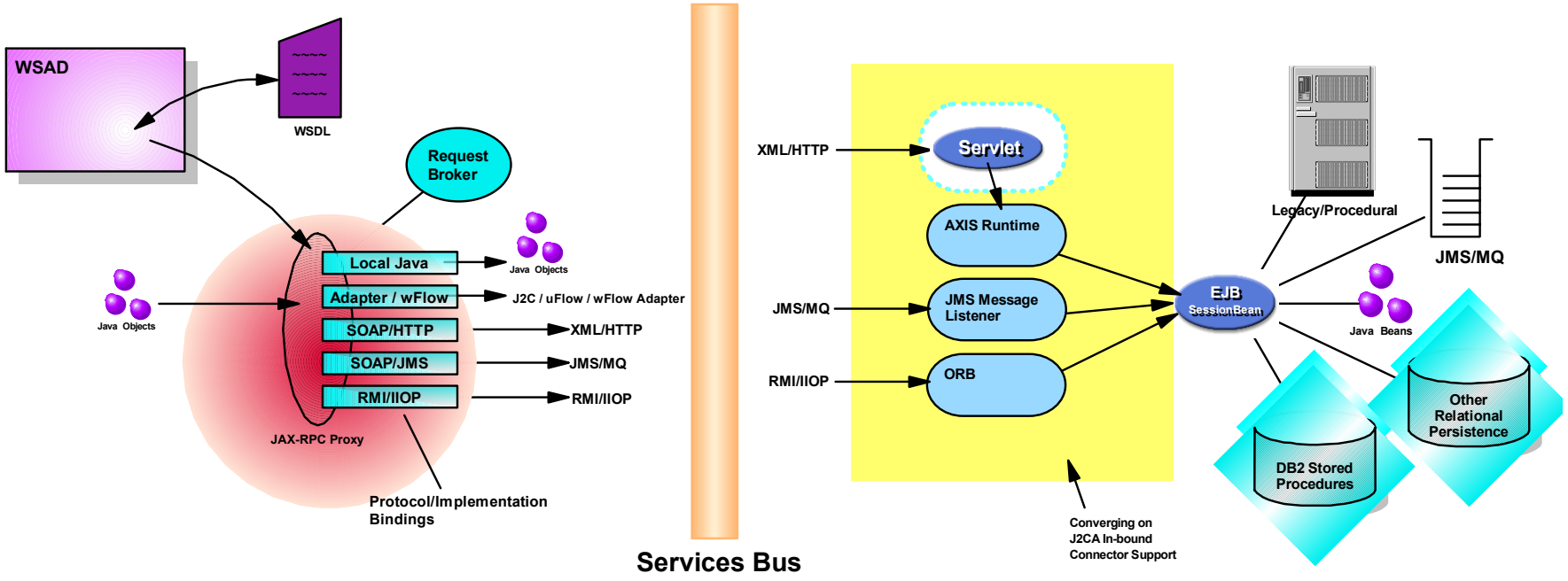
Web Services Technical Strategy

Leveraging Simple Beginnings

- Tactical thinking:
  - Web services are a way of gaining access to business function over the internet through programmatic interfaces enabled by HTTP and SOAP
- Strategic thinking:
  - Web services are a way of describing business function that can be accessed over a variety of communication protocols
  - Focus is on WSDL
  - Different binding protocols offer different qualities of service -- loose vs. tight coupling
    - SOAP/HTTP -- un-reliable, but ubiquitous, internet enabled
    - SOAP/HTTP-R -- reliable, but less ubiquitous, internet enabled
    - SOAP/SMTP -- mail-oriented, high latency, store-forward
    - SOAP/JMS or XML/JMS -- asynchronous, high latency, leverages existing providers
    - RMI/IIOP -- synchronous, low latency, standardized service contexts, transactions
    - J2CA Adapters -- legacy integration and microflows
    - Local Java -- locally optimized
    - Other: RosettaNet, ebXML, etc.
  - Choose the binding protocol that meets the needs of the business situation
  - Resolve the programming model issues of protocol transparency
  - Service context definition and propagation

J2EE is a component technology for implementing business services; web services are a distributed systems technology for loosely-coupled access to business serivces

Services Oriented Architecture



- Enables a service-bus of loosely-coupled components
- Standardized services specification
- Multi-protocol support -- separates developer and deployment concerns

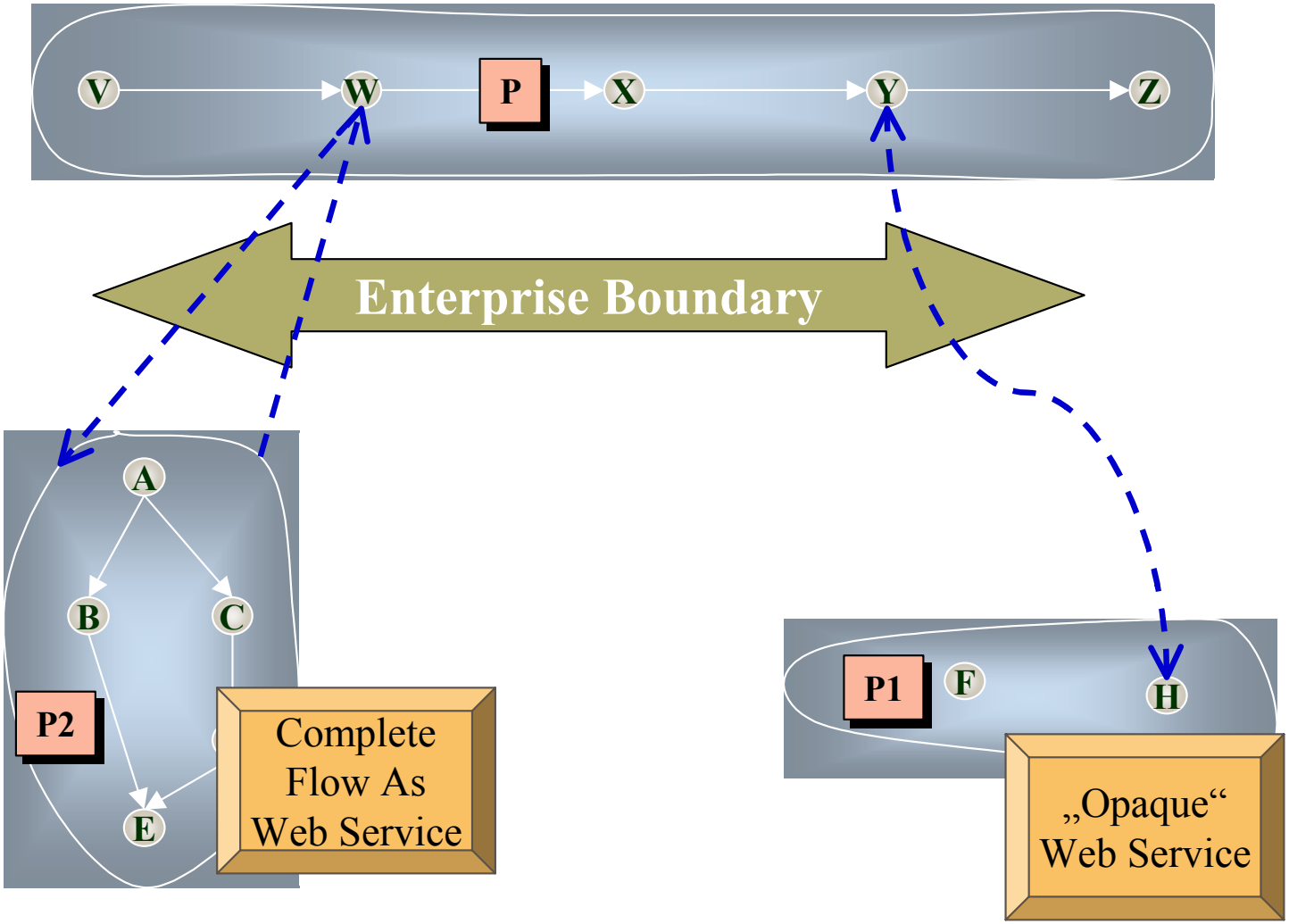
Web Services Invocation Framework

- Framework and runtime support in
  - WebSphere Application Server
  - Client environments
- Client Side
  - JAX-RPC Stub
  - Binding selection establishes “protocol” and “format”
  - Transparent to caller
- Server Side
  - Prebuilt request dispatchers for SOAP/HTTP, IIOP, JMS (P2P, Pub/Sub), Java
  - Extensible model for adding request dispatchers
  - Isolates service implementation from channel
- Open-source donation to Apache

BPEL4WS

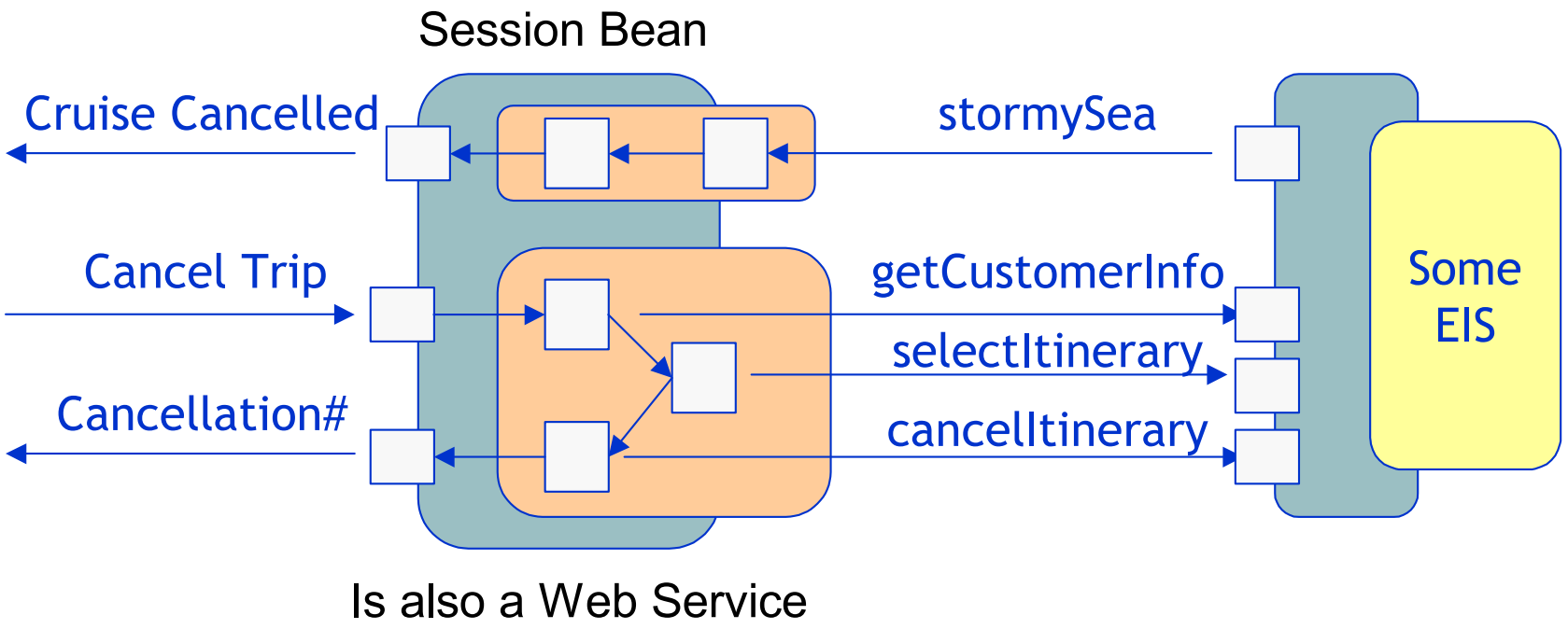
- A language to specify behavior of business processes
  - Between Web services...
  - ...and as Web services
- Same language to define executable processes and business protocols
- Executable processes
  - Can be performed at all compliant environments
  - Interoperability of heterogeneous environments
- Abstract processes
  - Specify constraints of message exchange
  - Expresses external interaction requirements for 3rd-party collaboration
  - Enables encapsulation of implementation details in the executable process definition
- Extends and subsumes previous specifications published individually by Microsoft, XLANG (05/01) and IBM, WSFL (05/01) into one defined specification (BPEL4WS).
  - Defines an XML-based workflow language which allows businesses to describe rich business processes that can both consume and publish Web services in a reliable and dependable manner.
  - Enables portability and interoperability by defining a set of constructs to implement executable business processes and to implement message exchange protocols.
  - Defines operational semantics for the execution of business processes in a well-defined manner for business processes and business partners in heterogeneous environments.

Web Services Composition



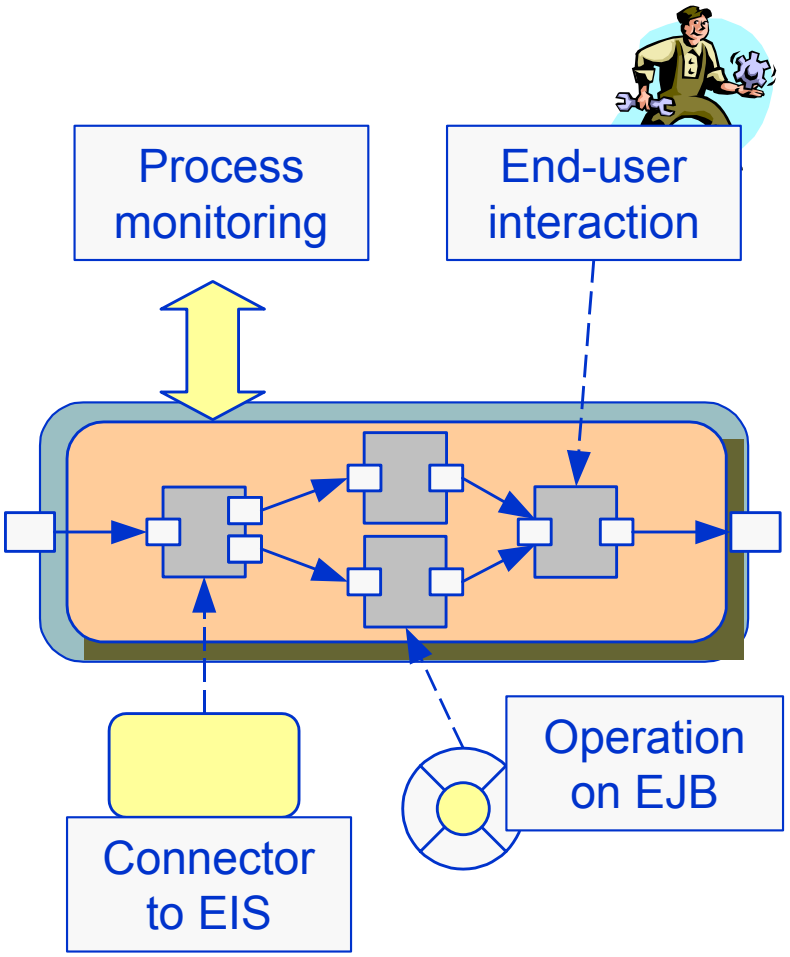
Adapters

- JCA supports shallow encapsulation of backend systems
- Microflows support adaptation of 'native' EIS interface to process requirements
- Inbound and outbound adapters



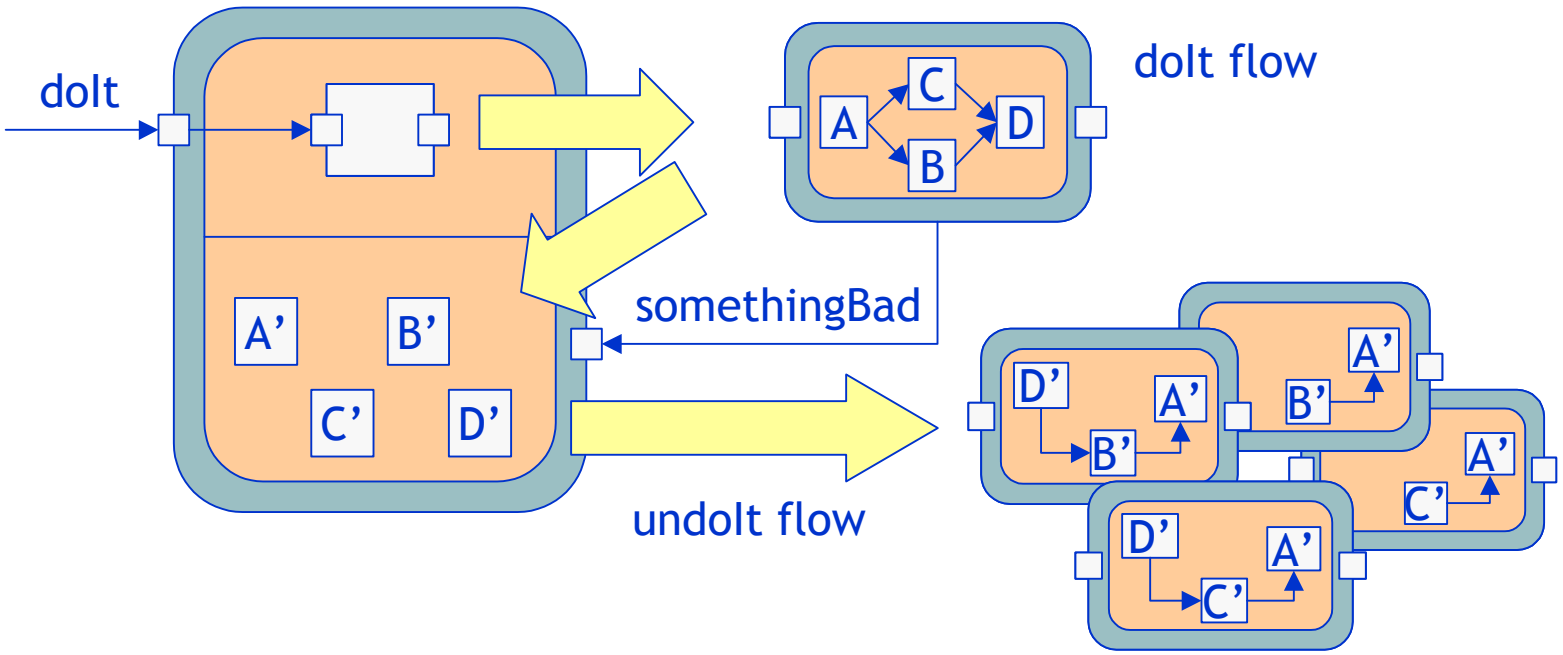
Work Flows

- Coarse-grained, potentially long running recoverable flows
- Interactive workflow processes
  - Coordinate interactions with end users
  - Staff assignment, worklist management
- Automated, interruptible process flows
  - Script stateful interactions of application components



Advanced Composition Patterns

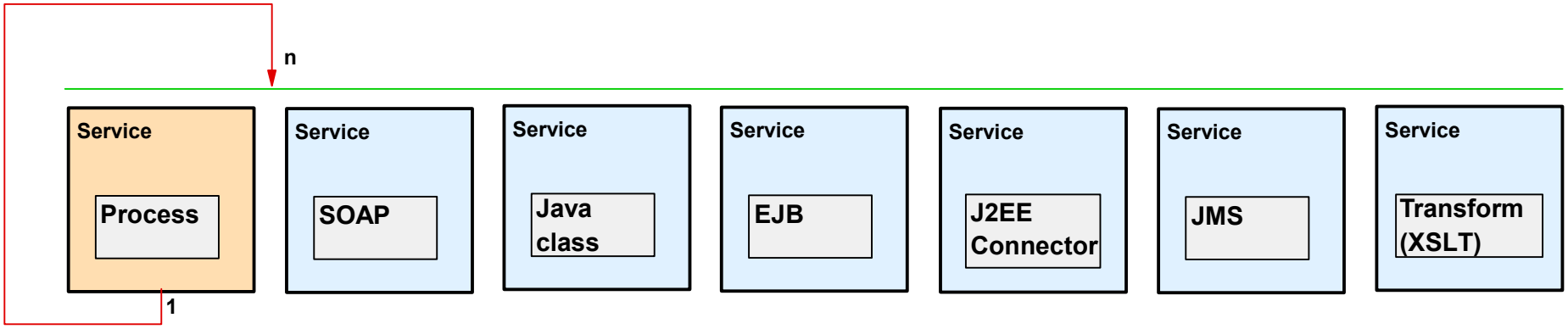
- Compensation-Flows provides framework for realising additional composition patterns
- Most of them are not very attractive from a ‘programming in graphics perspective’ – process behaviour encoded in pattern
- Example: flow compensation pattern



Business Process Management

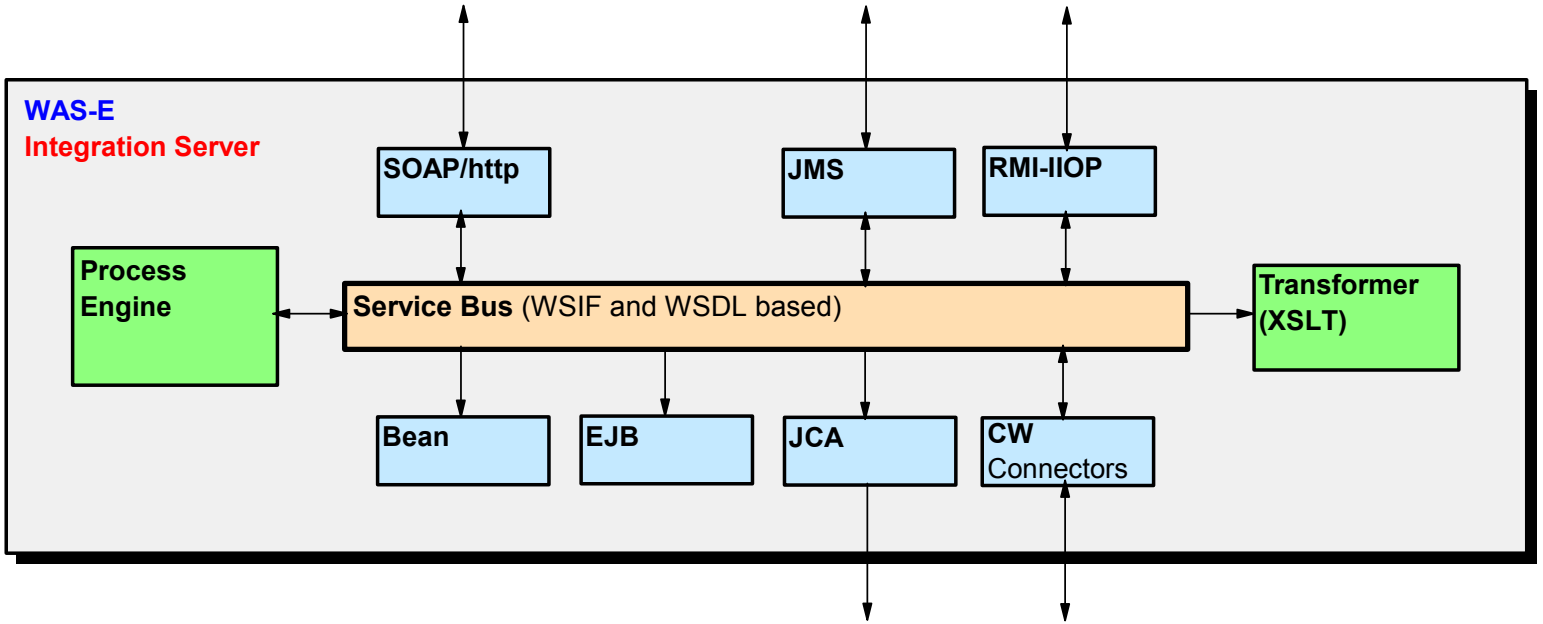
Overview

Its all about Services



Overview

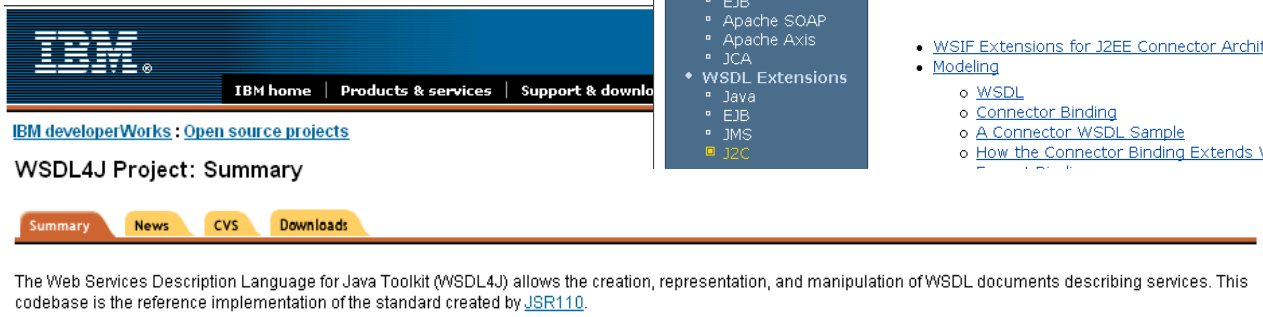
WebSphere Application Server Enterprise – The Integration Server



- J2EE & Services
- Standards based Environment for Business and Enterprise Application Integration

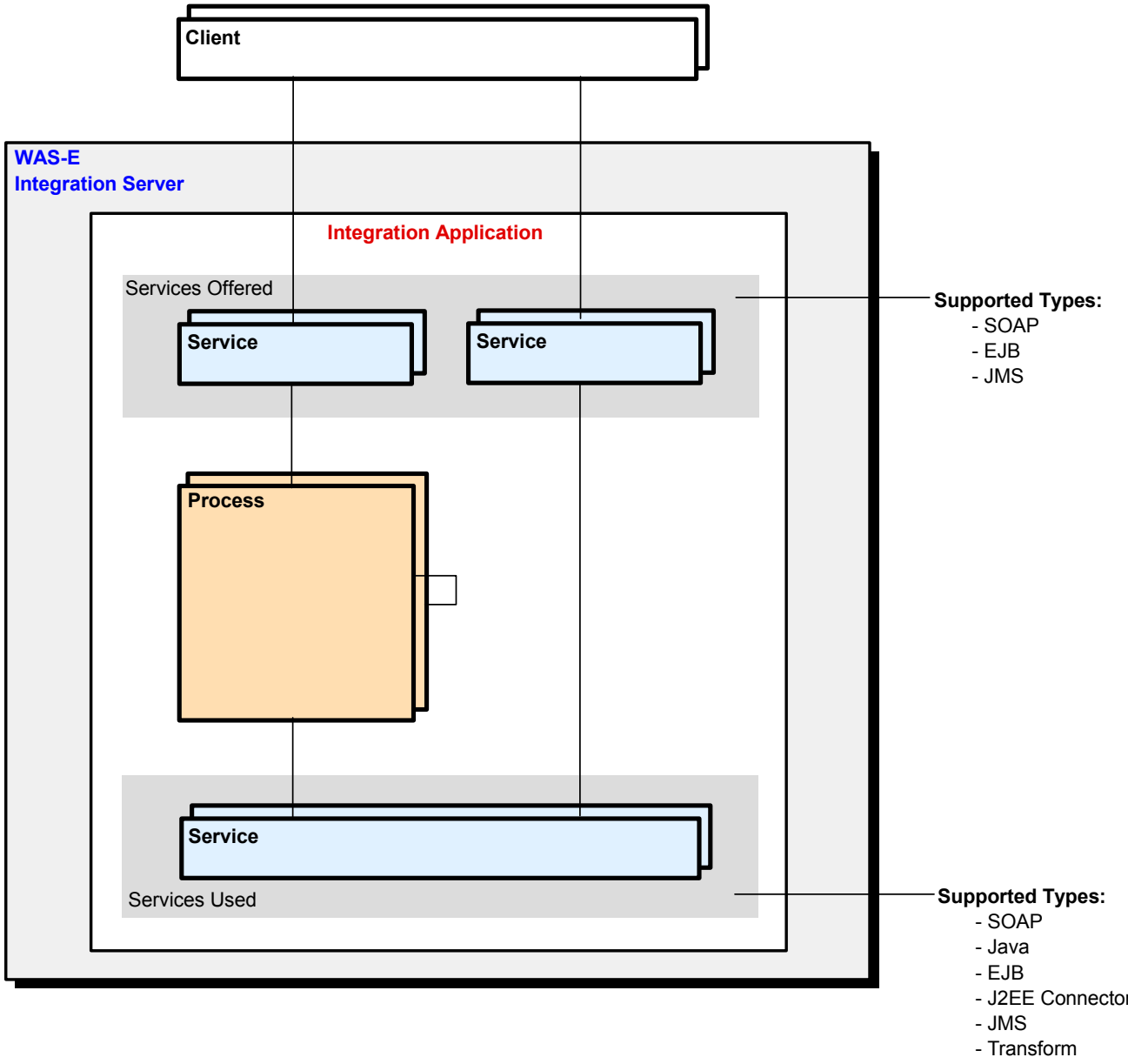
Overview

WSIF, WSDL, J2C Plugin - http://ws.apache.org/wsif/



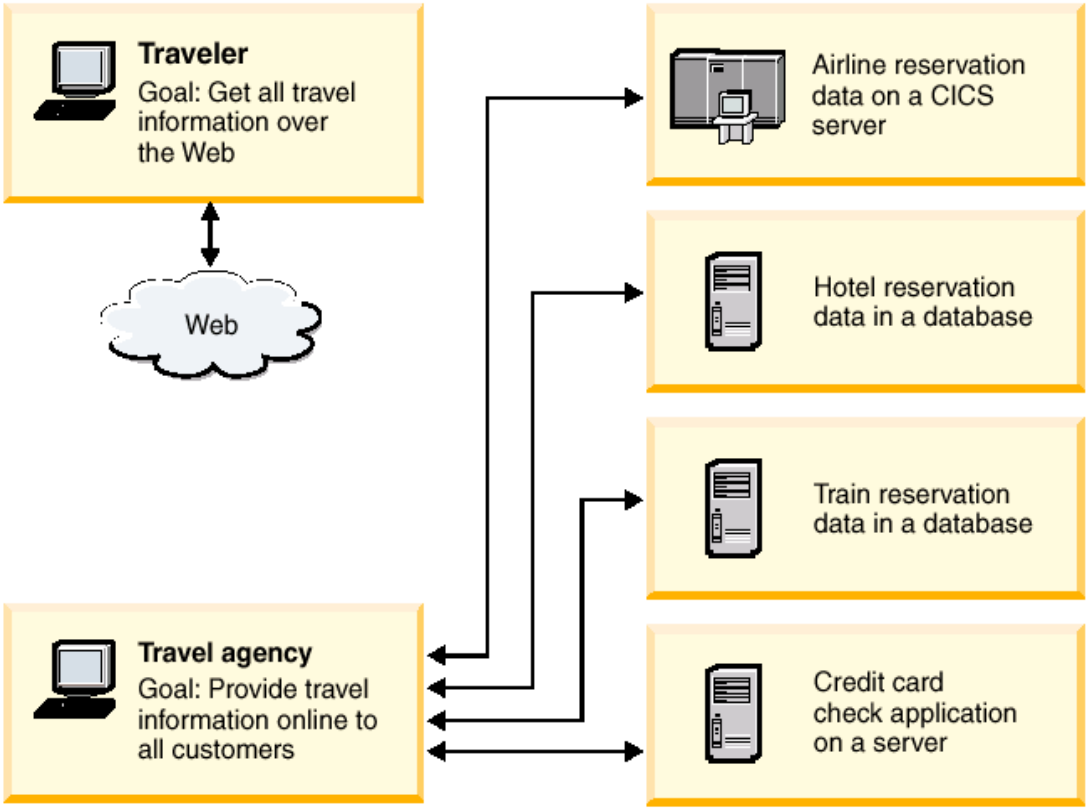
Overview

Integration Application



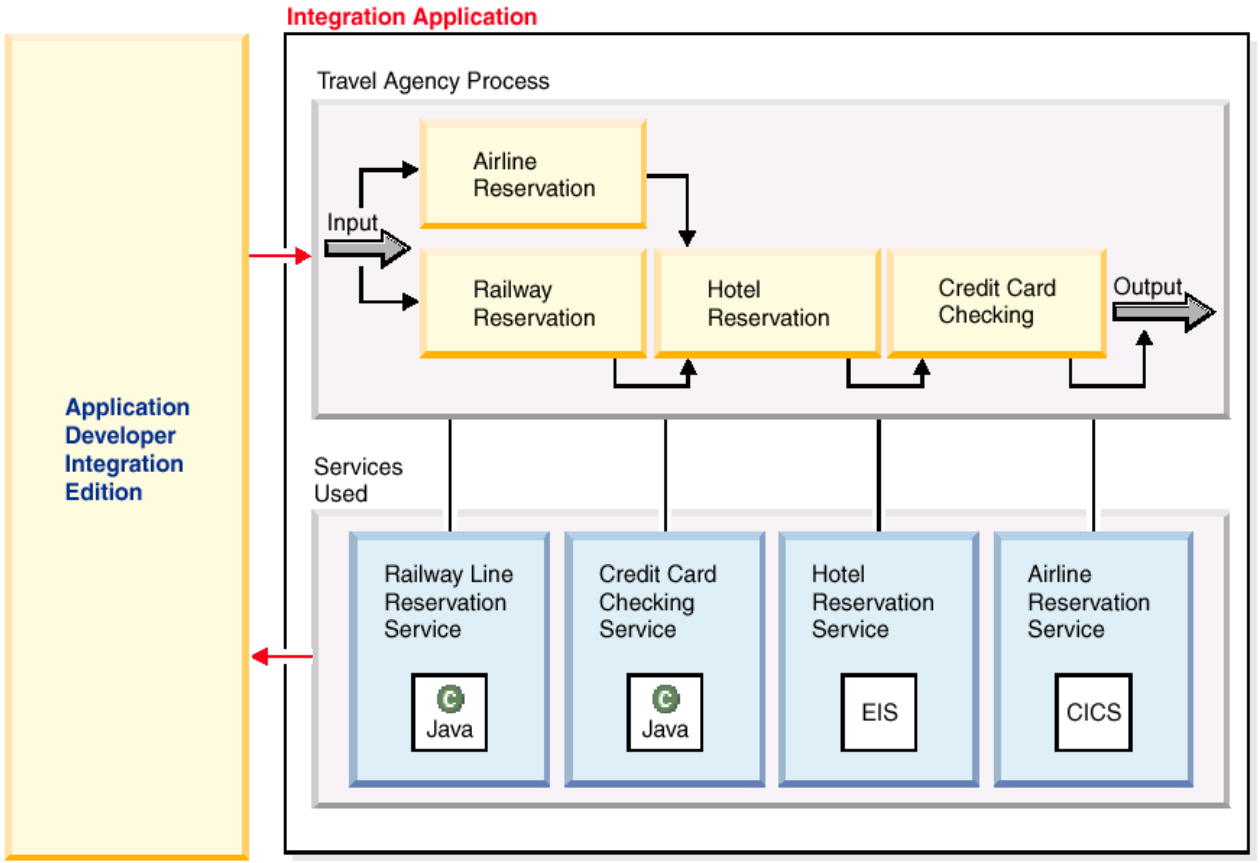
Travel Scenario

The Problem



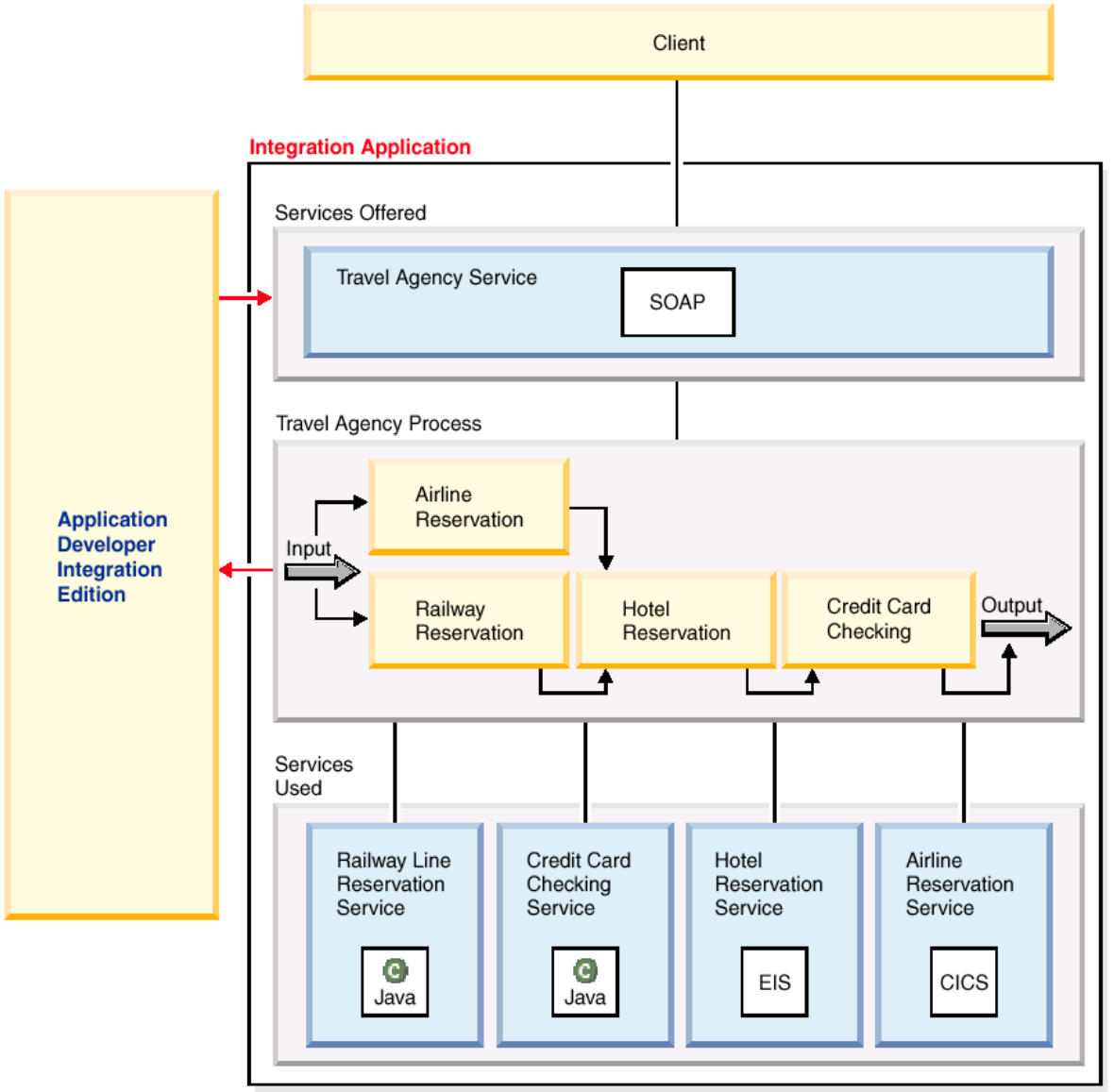
Travel Scenario

Creating a business process



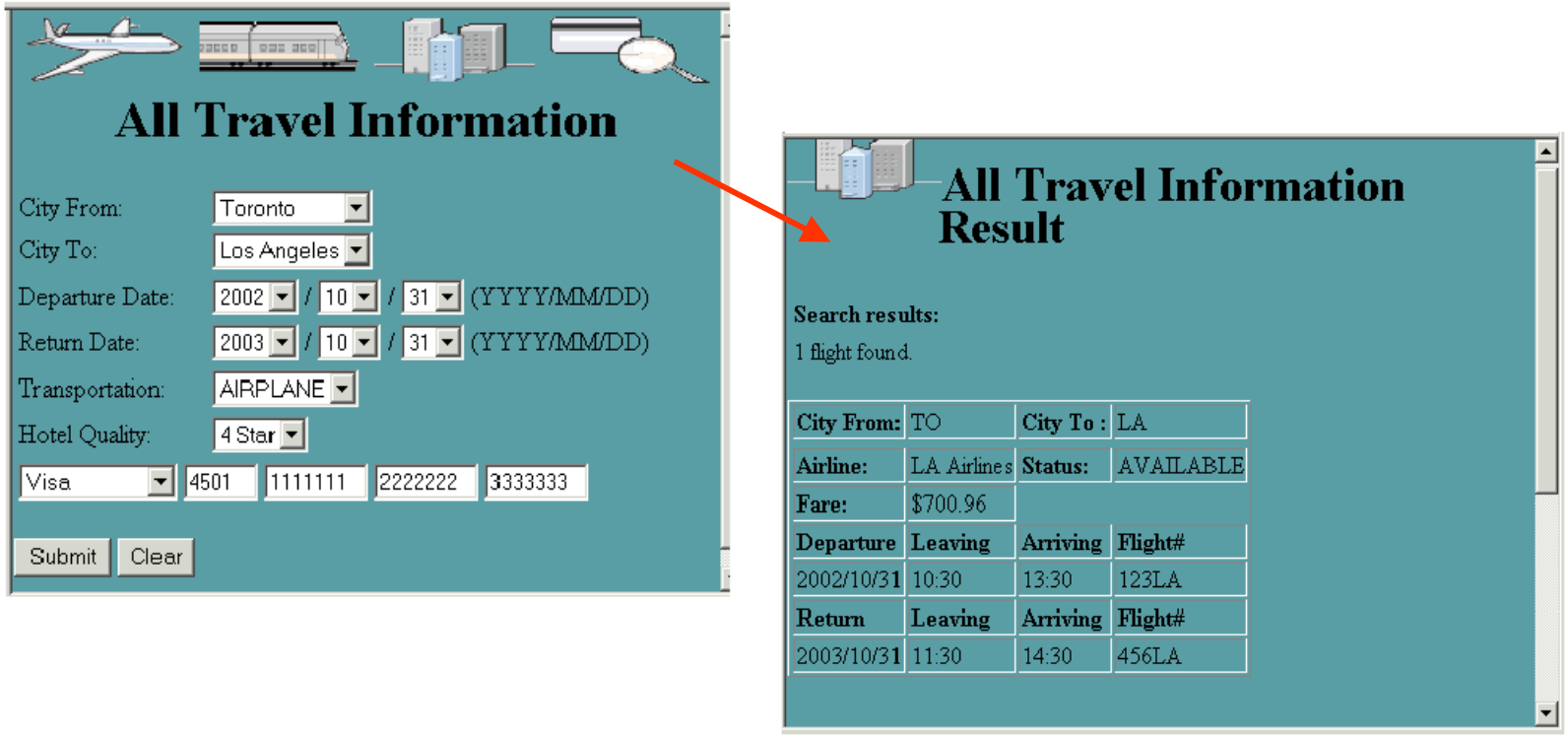
Travel Scenario

Creating the service you want to offer



Travel Scenario

Using the new service



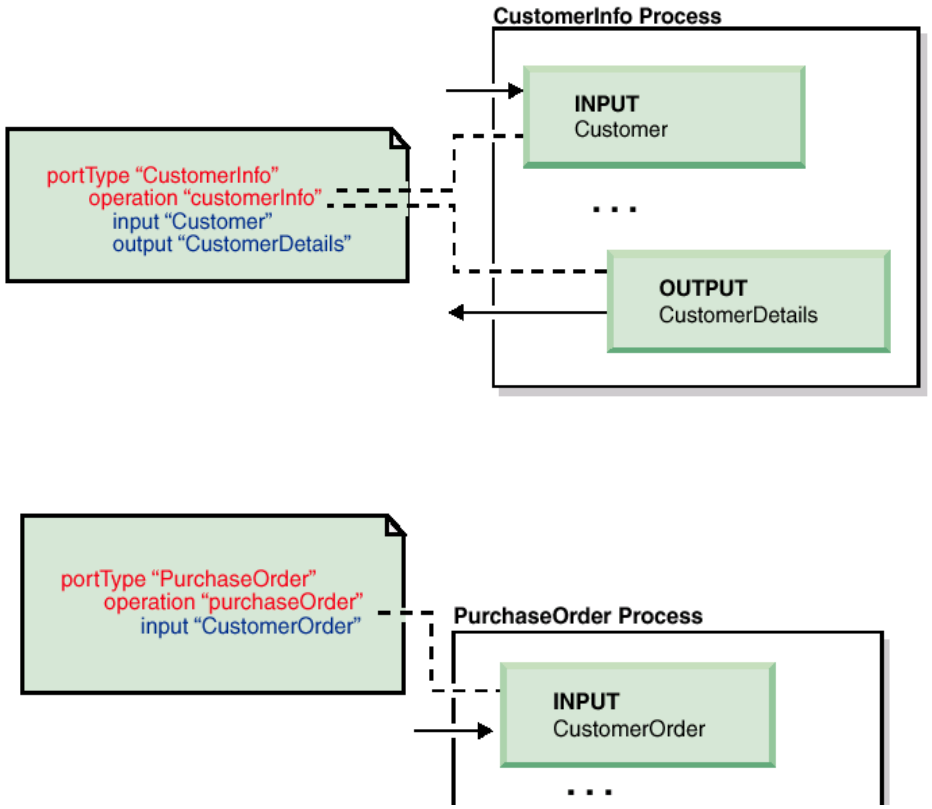
Process Programming Model

Process

- Process Interface
- Process Implementation
  - Short running, non-interruptible process (aka Microflow)
  - Long running, interruptible process (aka Macroflow)

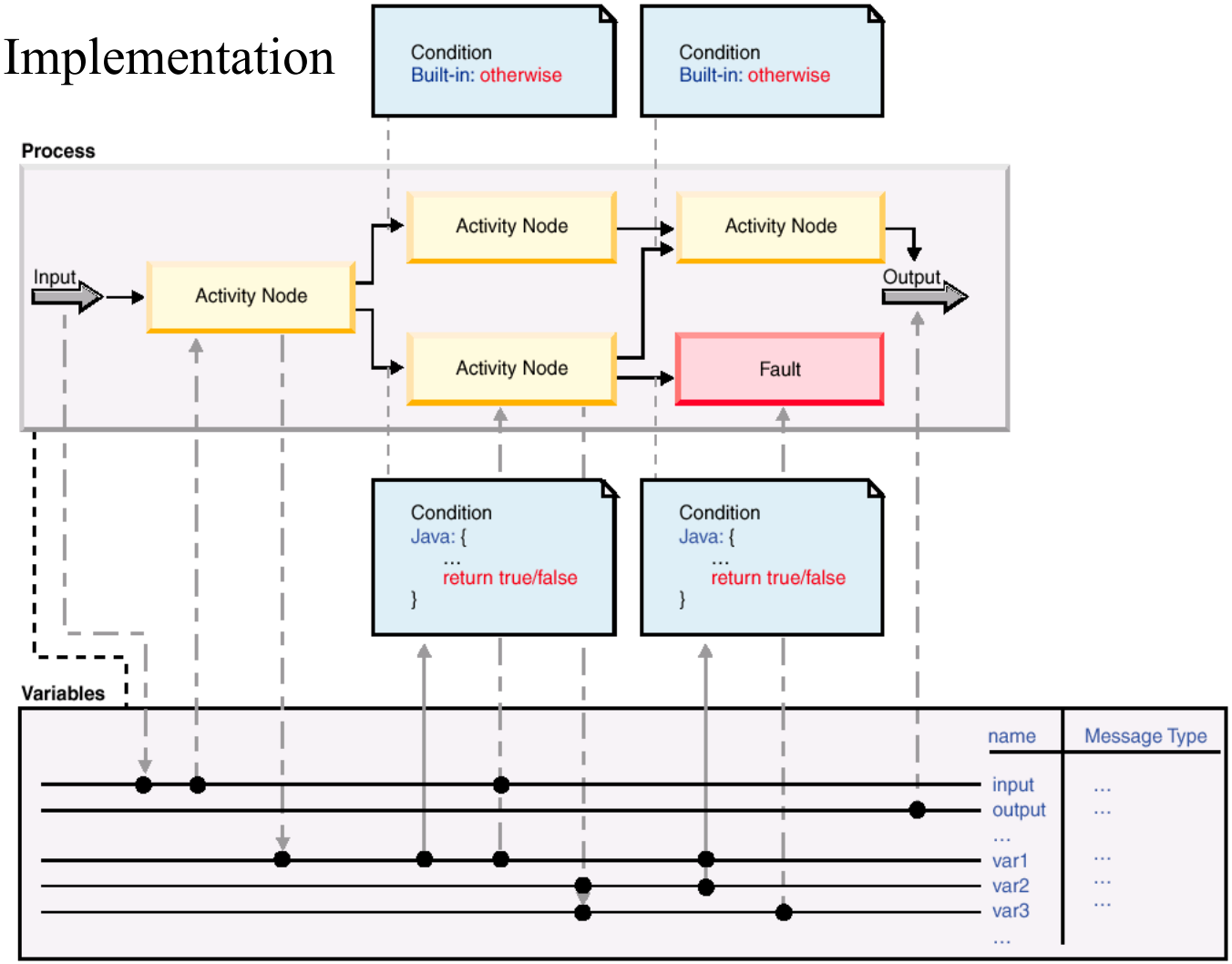
Process Programming Model

Process Interface – sync and async



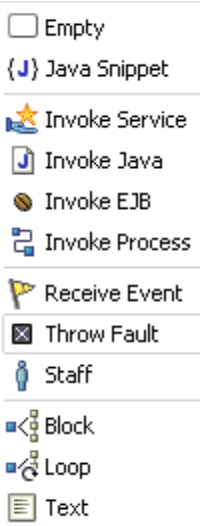
Process Programming Model

Process Implementation



Process Programming Model

Process Implementation



Basic Activities

- Input
- Output
- Fault
- Empty
- Service
- Java Snippet
- Java
- EJB

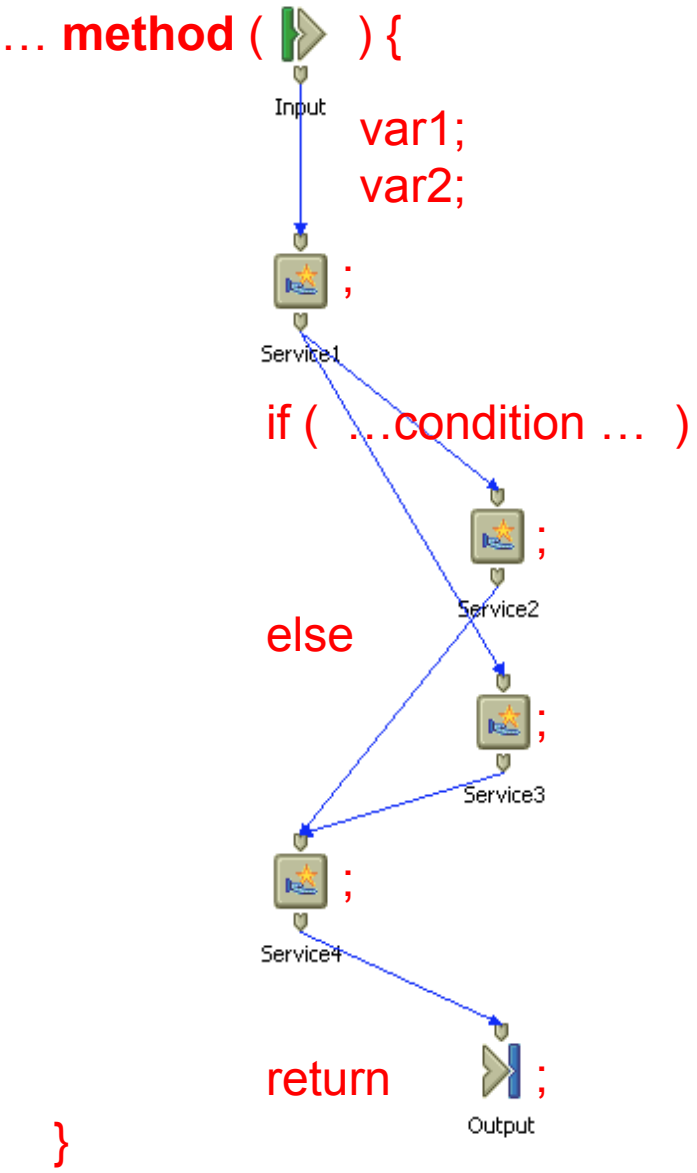
Structured Activities

- Blocks
- Loops
- Processes
- Events (staff queries, expiration)
- Staff (staff queries, expiration)

Process Programming Model

Process Implementation

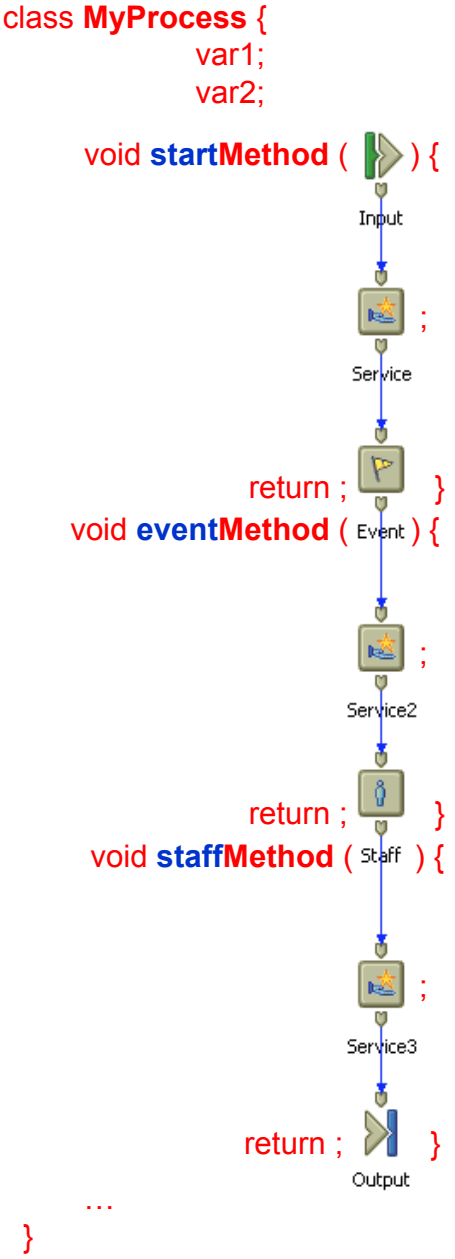
- Non-interruptible process
  - think visual method implementation



Process Programming Model

Process Implementation

- Interruptible process
  - think entity with multiple state changing methods



Process Programming Model

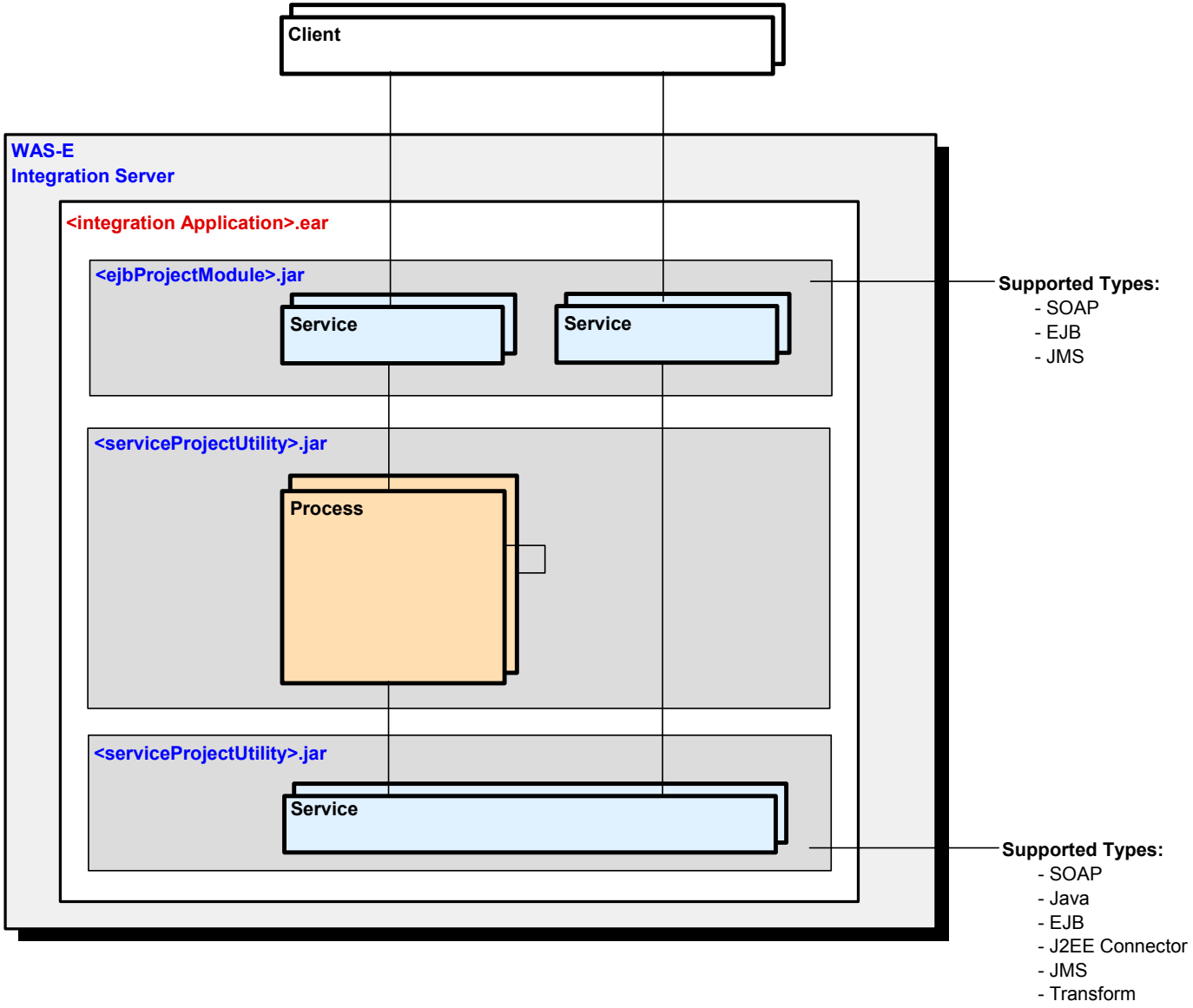
Process Implementation – Process level implementation concepts

- Correlation sets, i.e. process keys
- Compensation

- Valid from
- Audit
- Staff queries for start and administration

Deployment Model

Integration Application



BPEL4WS support – tech preview

- Create BPEL process from WSADIE process
- Create WSADIE process from BPEL process

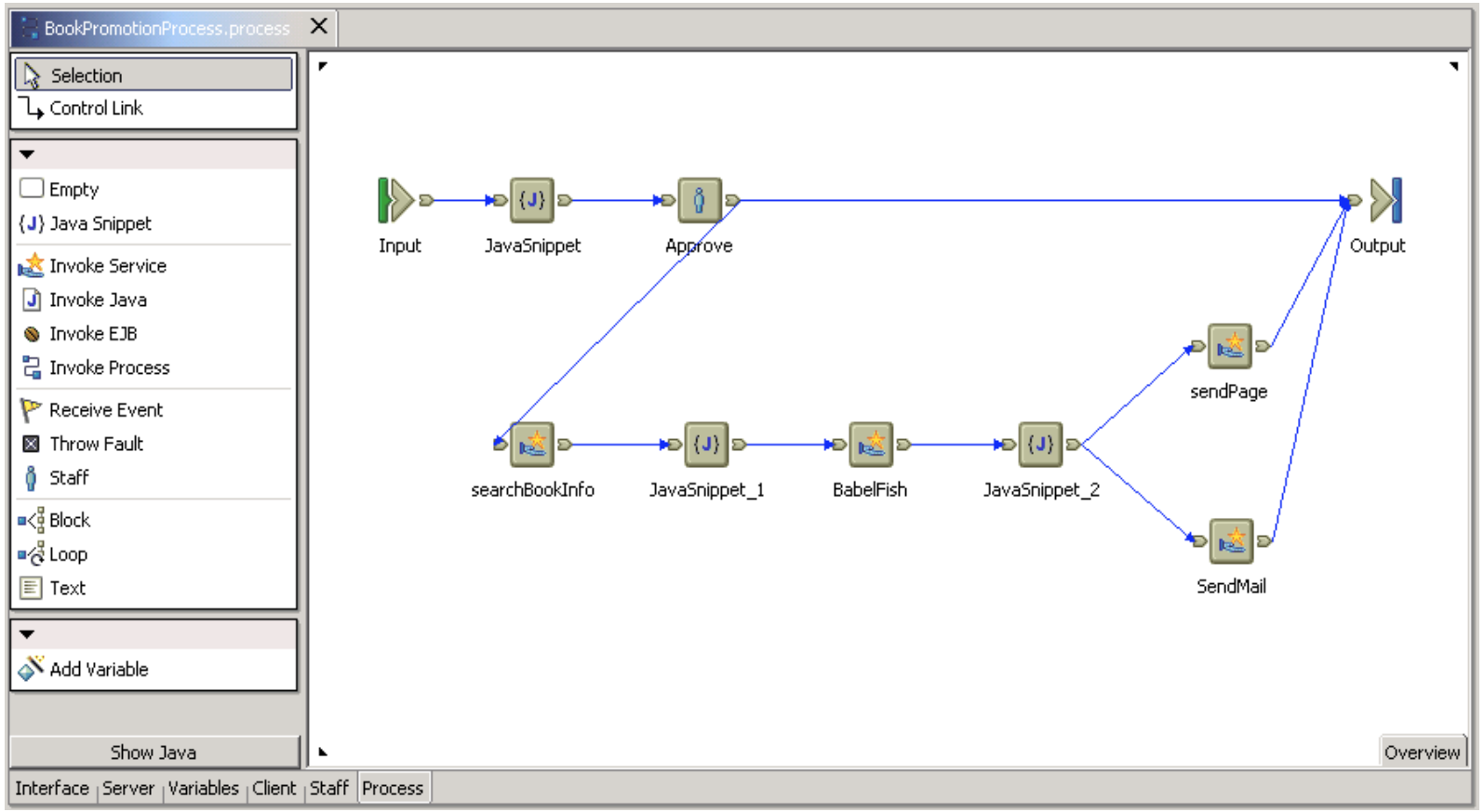
➢Focus on mapping WSADIE process features on BPEL patterns

➢BPEL documents have to adhere to these patterns to be importable

v5 Process Feature	BPEL Support
Input Node	yes
Output Node	yes
Fault Node	yes
Service Node -> one-way	yes
Service Node -> req-resp	yes
Empty Node	yes
JavaSnippet Node	yes
Block	yes
Loop	limited
Receive Event Node	yes
Compensation	yes
CorrelationID	limited
Transition Conditions	limited
Faults (on blocks, loops and service nodes)	yes
Variables	yes
Service Information	yes
Synchronous process	yes
Asynchronous processes	yes
Staff	no
Process Node	no
EJB Node	no
Java Node	no
Text Comments	no
All Graph Layouts	yes

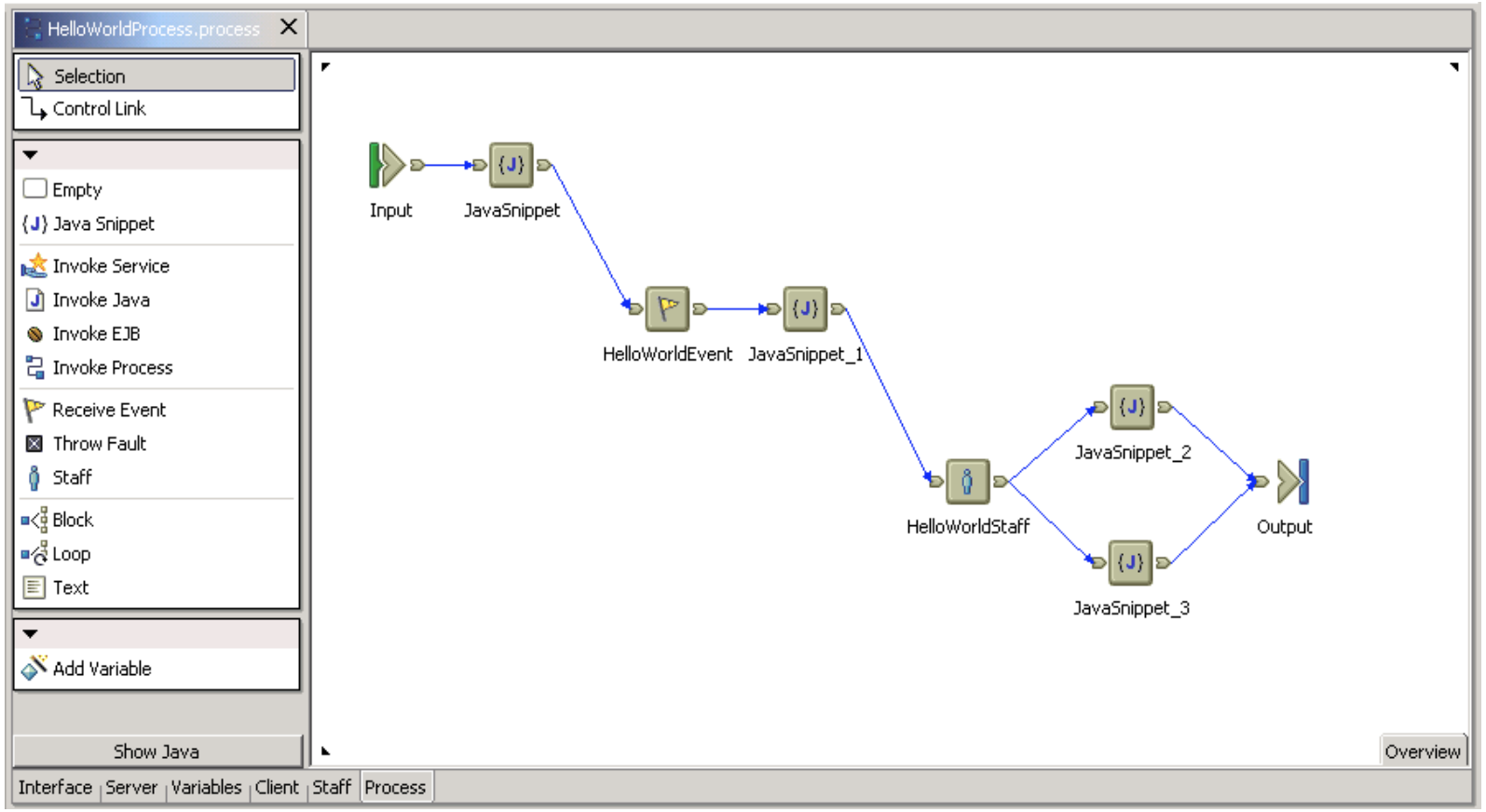
Demo

books4less.processes.BookPromotionProcess.process



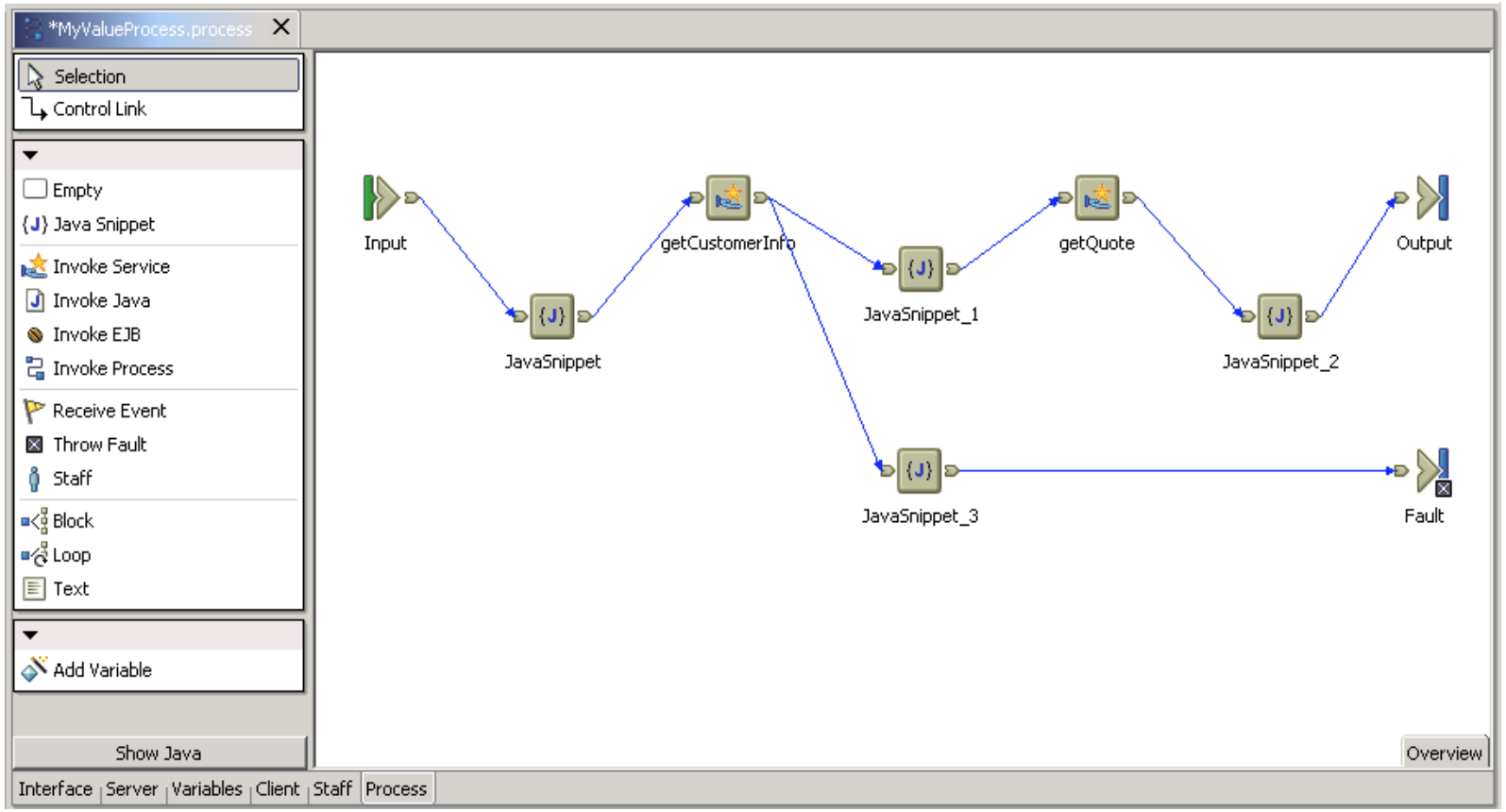
Demo

helloworld.processes.HelloWorldProcess.process



Demo

myvalue.processes.MyValueProcess.process



Getting your hands on it

- WSADIE V5
  - http://rolo.torolab.ibm.com/index_v5.0.html
  - http://rolo.torolab.ibm.com/builds/releases/bpel-TechPreview/
- Product Documentation Online
  - http://bidoc.torolab.ibm.com:9080/help/help.jsp
- BPEL4WS
  - http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
- WSIF
  - http://ws.apache.org/wsif/
- WSDL4J
  - http://www-124.ibm.com/developerworks/projects/wsdl4j/
- J2C Plugin
  - http://ws.apache.org/wsif/providers/wsdl_extensions/j2c_extension.html