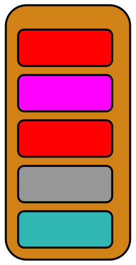
IBM's Service Oriented Architecture: Programming Model and Architecture

Donald F. Ferguson

IBM Fellow
SWG Chief Architect and Chair, SWG Architecture Board

What's this "Service Thing?"

WSDL



Message M1, M2, … …

Op1, inMsg1, outMsg1, faultMsg1

Op2, inMsg2, outMsg2, faultMsg2

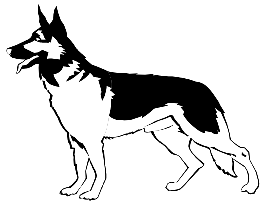
… … …

"That's my simul. package!"

Dude!
A SessionBean

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

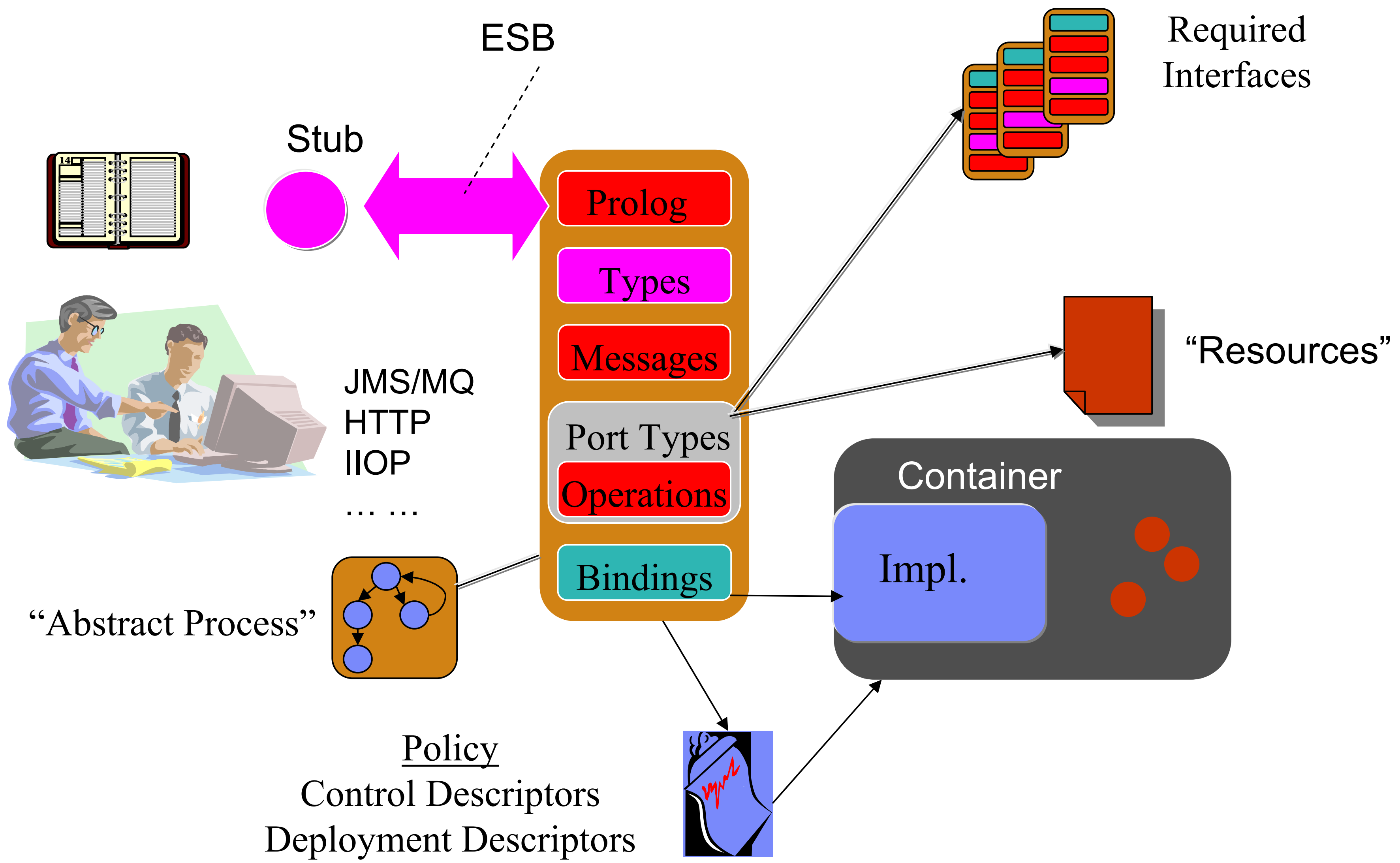
Narr! Das ist eine IDOC



".NET. I like it."

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

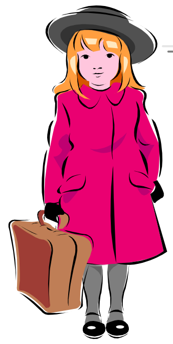
"Web services" is our "Component" Model



Summary

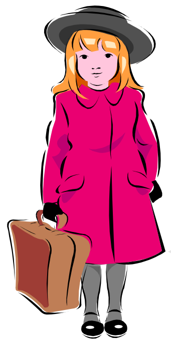
- SOA
  - SOA is a conceptual model
  - Web services is a set of standards that IBM uses in its SOA
- A Service (A Component)
  - Has a well-defined interface and implementation
  - MAY have a WSDL interface –
    - Our tools automate this requirement.
    - We also support Java, and will add COBOL, etc.
  - MAY be accessible through SOAP/HTTP –
    - Our runtime automates this requirement.
    - We also support MQ, IIOP, etc.
  - MAY define Declarative Policies for callers (WS-Policy, WS-PolicyAttachment)
    - I only processes WS-SecureMessaging
    - I support WS-AtomicTransactions
  - MAY be Stateful
  - MAY define an Abstract Process (or State Machine) for Partner Links
- A Component builds on this with
  - MAY declare requirements on other components
    - I "call" this interface
    - I must be bound to an instance that supports these policies
  - MAY define declarative policies for Container Management

Simplifying Development



Daddy,
Mommy gave me these
documents to convert.

What type of EJB
do you want to build?



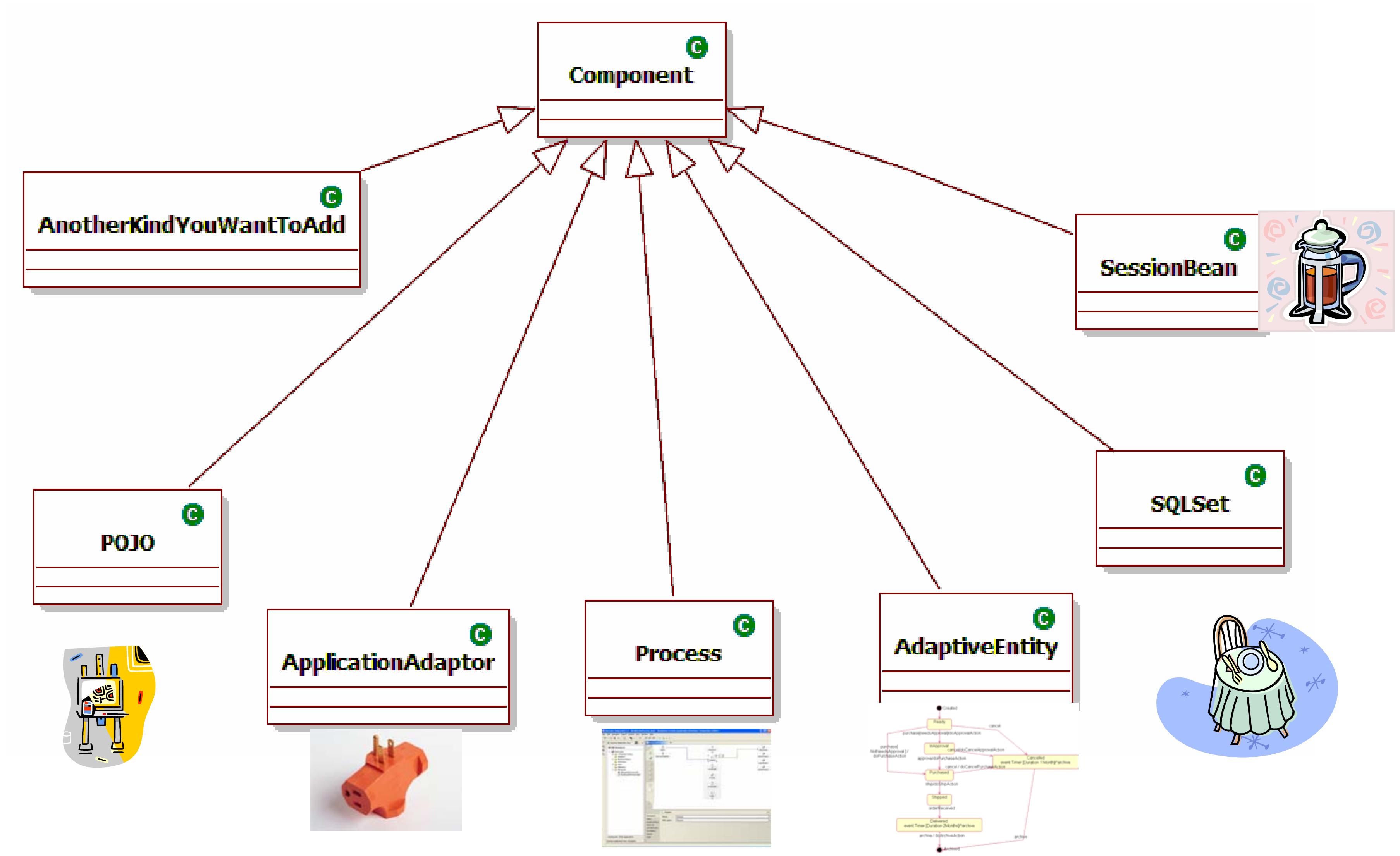
Um. I do not want to build
an EJB. You see, Mommy gave me these …

Maybe you didn't understand the question.
Your choices are SLSB, SFSB,
CMP Entity, BMP Entity, MDB

You're not very nice.

- This is crazy.
- Programmers want to build a "part" that implements a "basic building block" and then aggregate them together

Component Model – Examples of Types and Tools



A Simple Example and Some Concepts

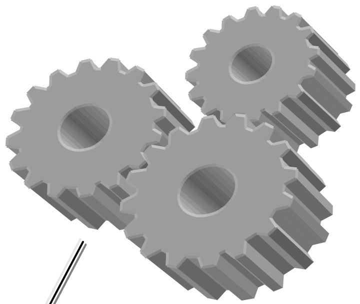
Something a DB dude recognizes

Author Tools
or
Text Editor

```
/*
Pragma This;
Pragma That;
*/
SELECT
            ticker,value, activity
FROM
            StockQuotes
WHERE
            ticker == QuoteRequest.ticker
INTO
            quoteResponse.ticker
            quoteResponse.value,
            quiteResponse.sharesTraded;
```

Attribute/Property/Metadata

Deployment Tools

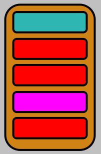


Deployment Package

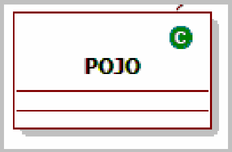
SDOs

```
public class QuoteRequest {
        String          ticker;
        Date            when;
}

Public class quoteResponse {
        String          ticker;
        float           value;
        float           sharesTraded;
}
```



Interface(s)



Data Svc.

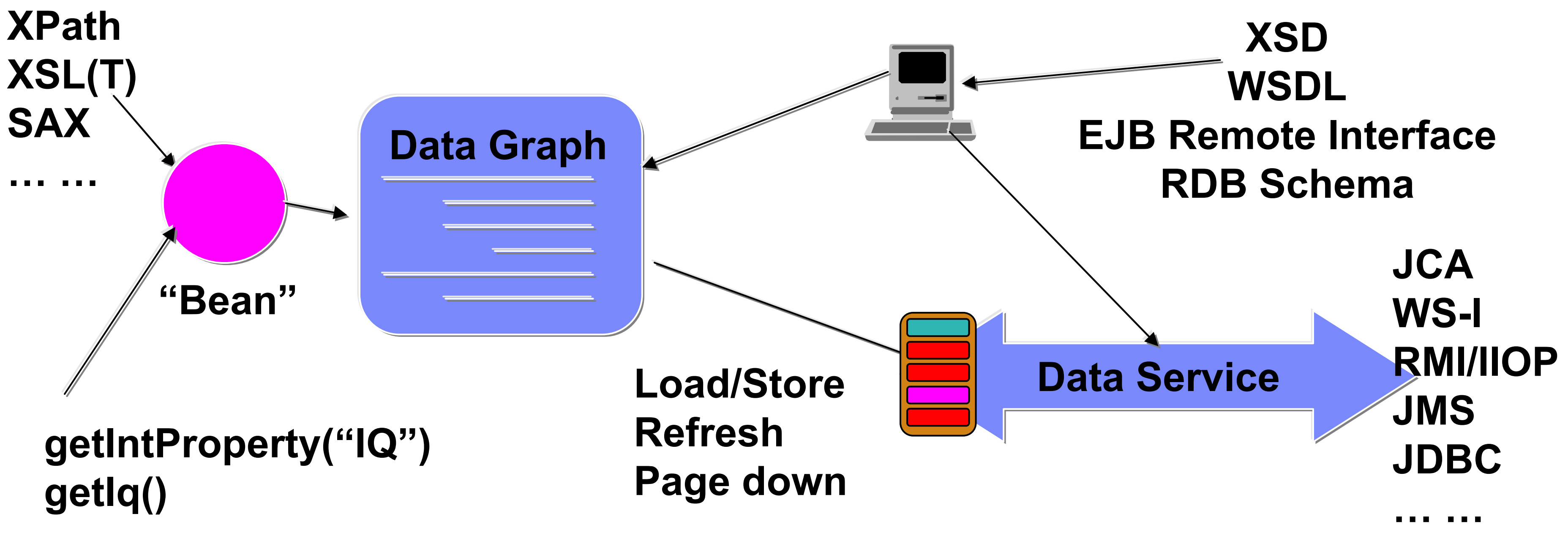
Generated Code
Interpreted Metadata

Component (Service) Types

- There is an extensible set of Service Types
- Each has own
  - Service Component Description Language (SCDL) format to pull together/reference bits and pieces
    - WSDL
    - Java classes and interface
    - Policy
    - etc.
  - Annotation based model extension to source file, e.g. POJO, .SQL, etc.
  - Valid set of policies
  - **Tool – Focuses on the task at hand and the skills**
- Some examples
  - Plain Old Java Object; Stateless SessionBean
  - BPEL Process Definition; Adaptive Entity = {Event, State, Action, newState}++
  - CICS, IMS TP programs
  - Application Adaptor, e.g. JCA → EIS
  - ESB Mediation
  - .SQL files; Stored Procedure
- Defining an extensible set of patterns/templates for each type.

Programming Model: Service Data Objects

- Once upon a time we had "commands."
- We also had data access beans (JDBC)
- Then we had three or four types of EJB Access Beans
- We have JAX RPC
- There is also JROM, JAXB, etc.
- Too many ways to do nearly the same thing.

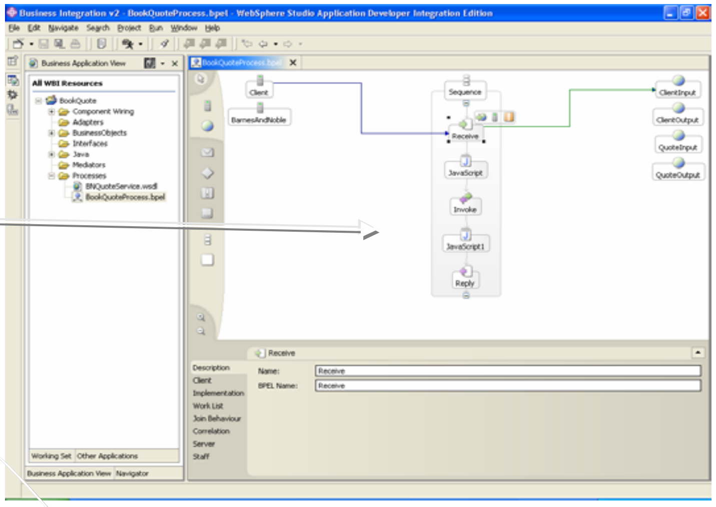


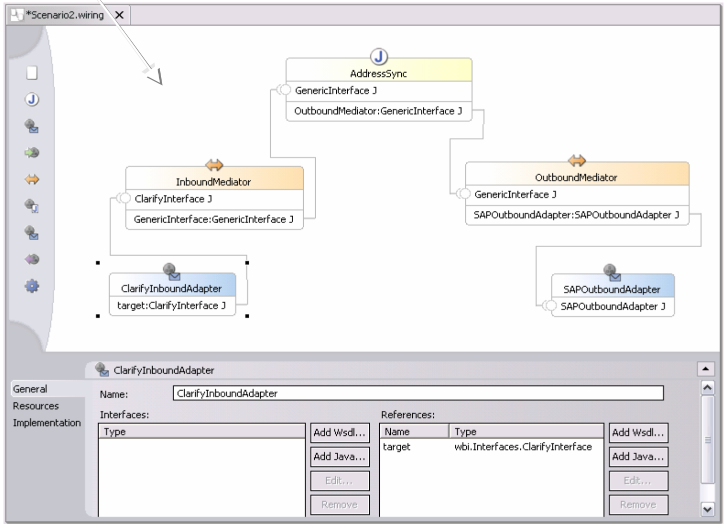
Programming Model: Service Data Objects

- There are many "client programming models,"
  - e.g. JDBC, JAX-RPC, RMI/IIOP, JMS, JCA, … …
  - We have added many different types of "helper" classes
- SDO unifies and simplifies these models for the most common use case of Retrieve, Display, Update, Write back
- SDOs are are a uniform abstraction and realization for "sources of data" and "in-flight messages"
  - WSDL operations, EJB method calls
  - JDBC/SQLJ Rowsets (query results)
  - XML documents, BPEL4WS Containers
  - JMS, Message Driven Applications, JCA
  - ESB Messages
  - … …
- Supports advanced features
  - Both pessimistic and optimistic transactions
  - Integrated with distributed, coherent cache
  - Both XML and Java access; Supports "by name" and "static methods"

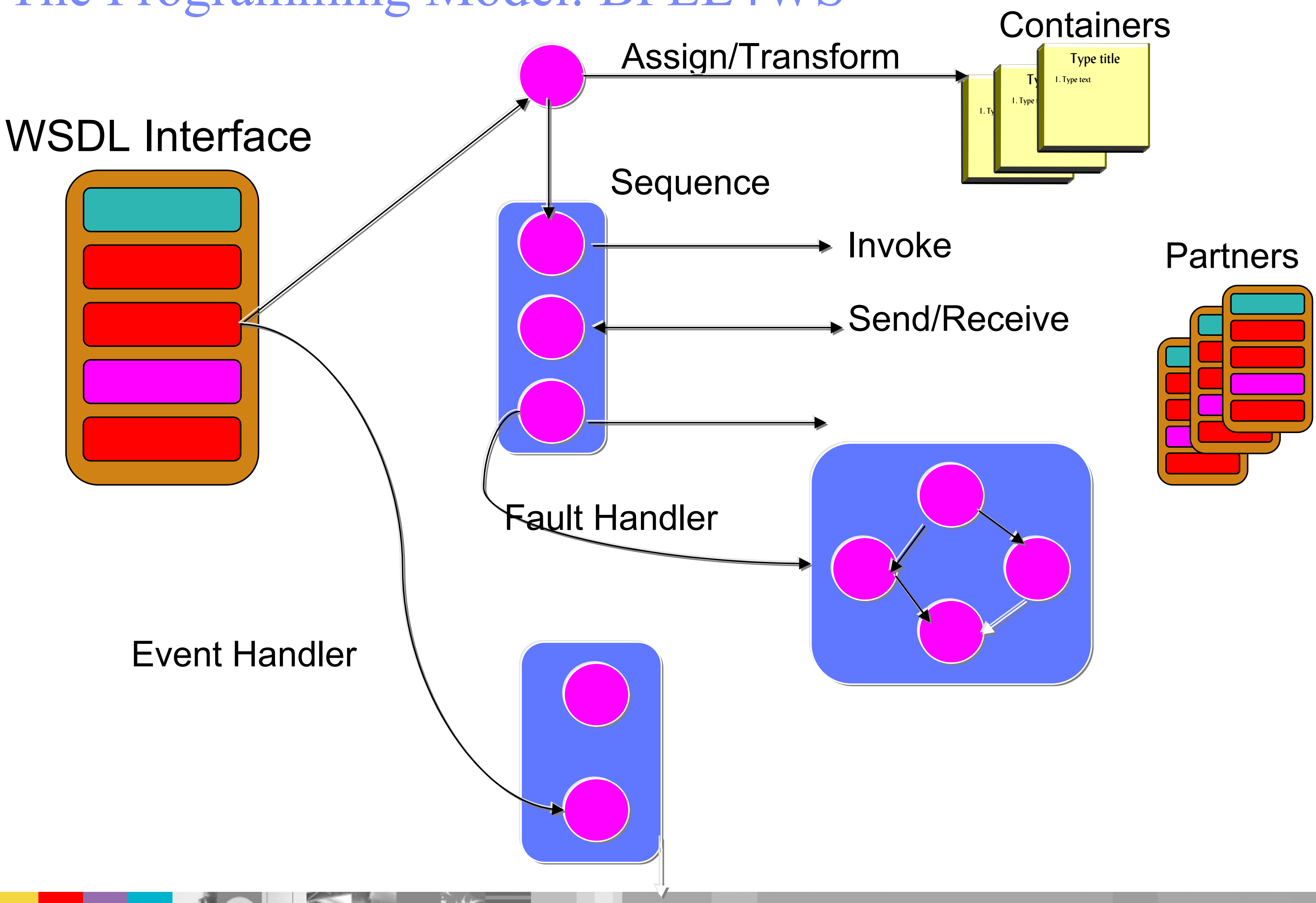
A Component Model Enables Assembly

- Simplifies tools
  - Asset view
  - Flow view (and invoke)
  - Wiring view
- Maps to/Imports from multiple file formats
  - EAR files
  - Java class with annotations

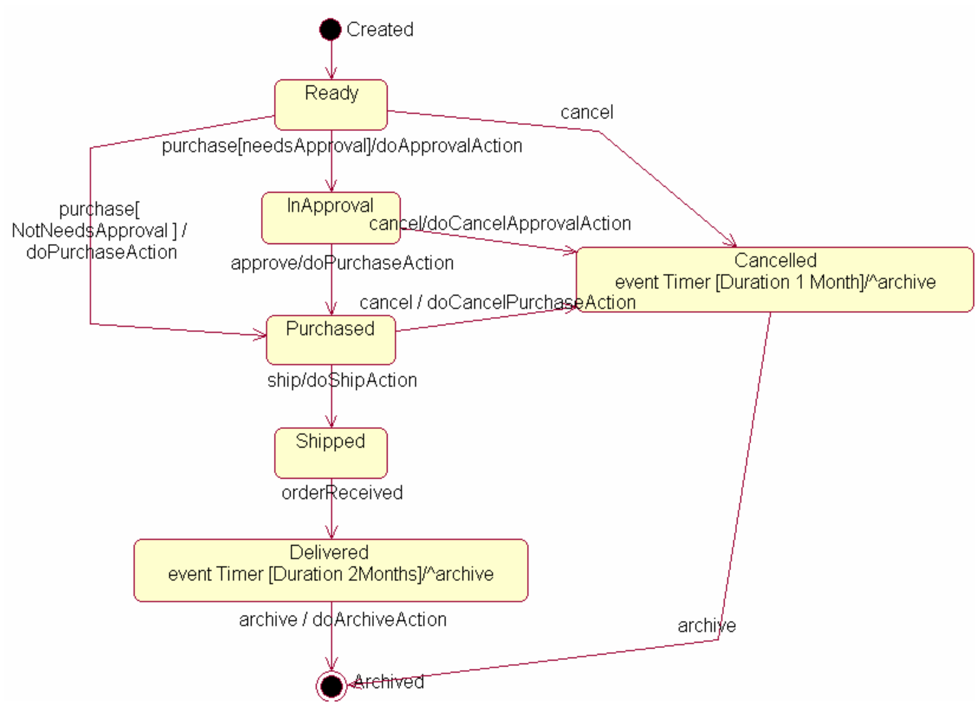




The Programming Model: BPEL4WS



Adaptive Business Object/Adaptive Entity

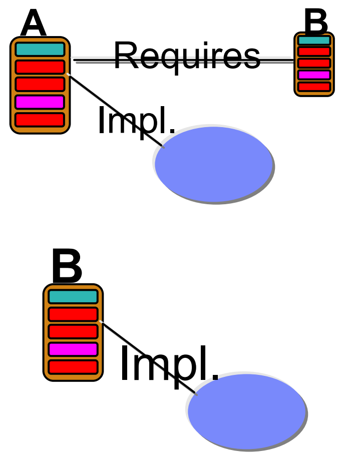

Created
Ready
cancel
purchase[needsApproval]/doApprovalAction
purchase[NotNeedsApproval] / doPurchaseAction
InApproval
cancel/doCancelApprovalAction
approve/doPurchaseAction
Cancelled
event Timer [Duration 1 Month]/^archive
cancel / doCancelPurchaseAction
Purchased
ship/doShipAction
Shipped
orderReceived
Delivered
event Timer [Duration 2Months]/^archive
archive / doArchiveAction
archive
Archived

- Thing – An EntityBean, e.g.
  - PurchaseOrder
  - Customer
  - Invoice

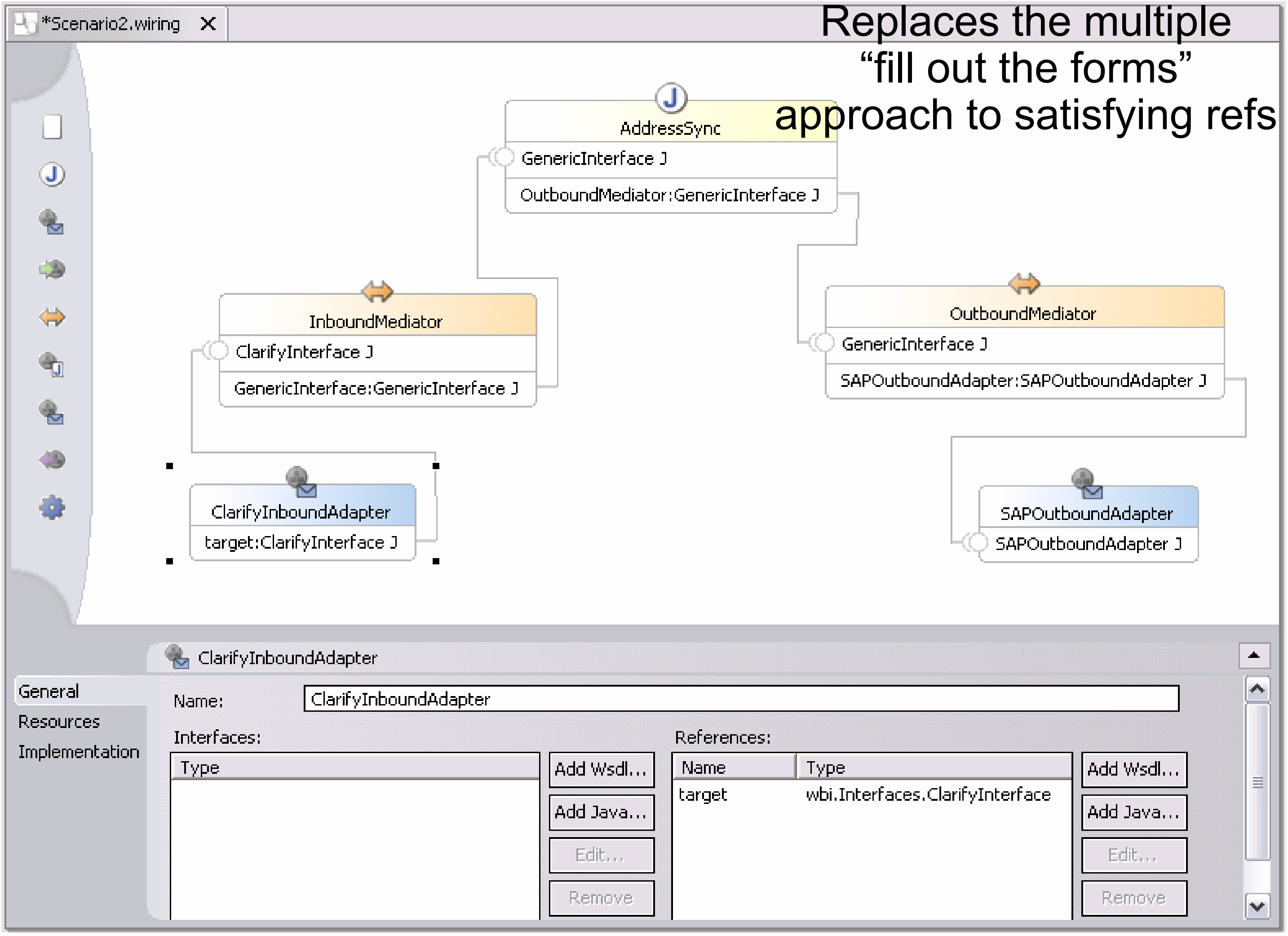
  Has state data, just like any "thing"
  - IQ
  - Date
  - Balance
  - … …

  Has a **state machine**
  - State
  - Valid Operations (Events) for the state – WSDL, Remote Interface
  - Actions (private methods) on exit, transition, entrance
  - Exit/Entry guards
- UML and SACL representations

Wiring Replaces Filling Out Forms"



Replaces the multiple "fill out the forms" approach to satisfying refs



Name: ClarifyInboundAdapter

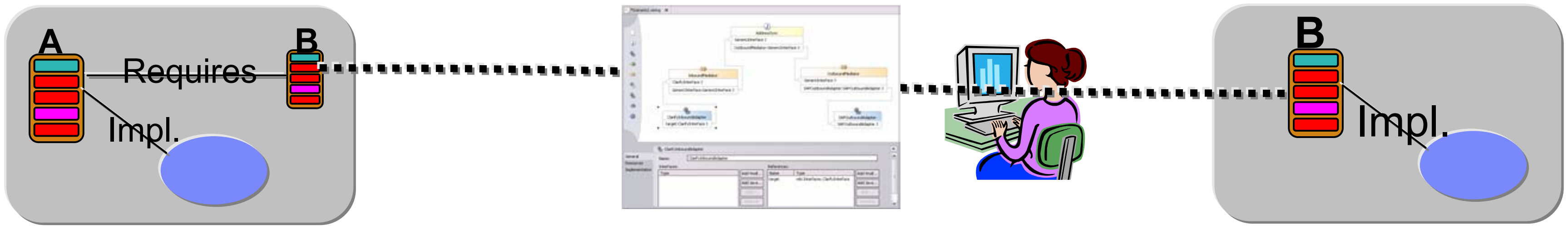
Interfaces:

Type

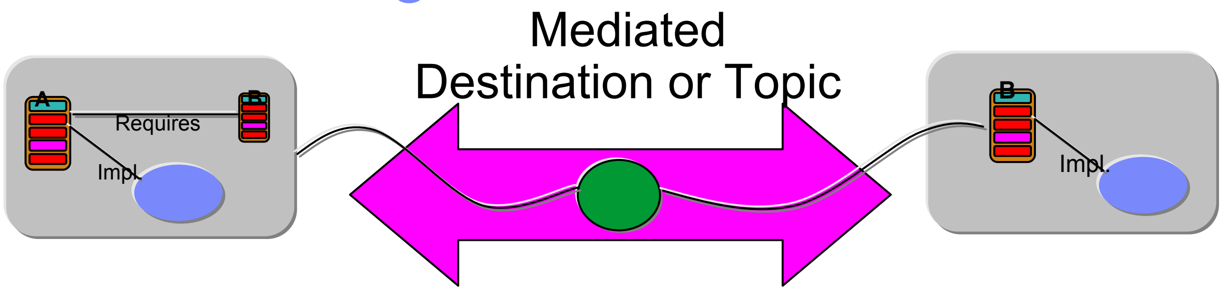
References:

Name	Type
target	wbi.Interfaces.ClarifyInterface

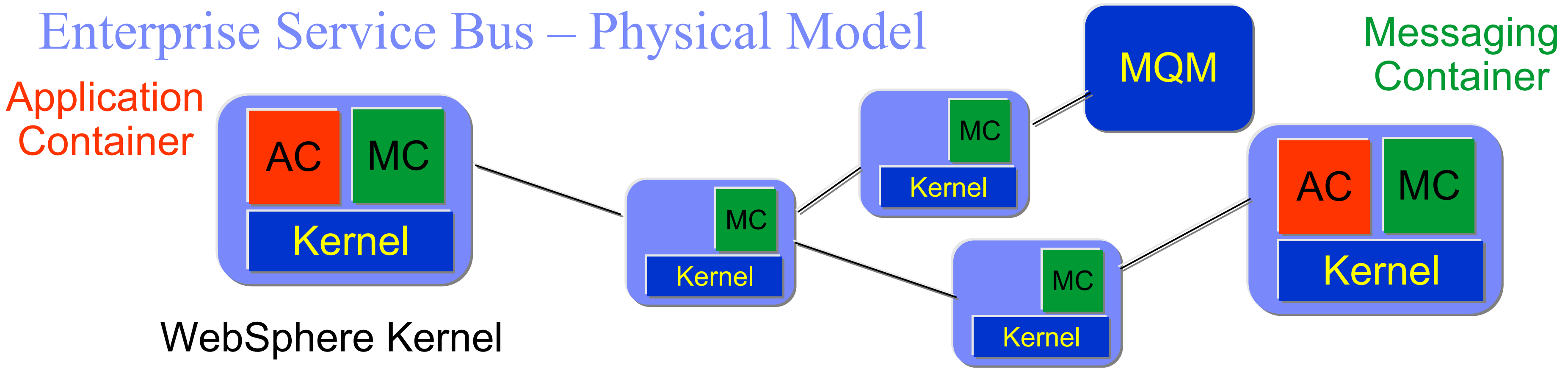
Enterprise Service Bus – Design Model



Enterprise Service Bus – Logical Model



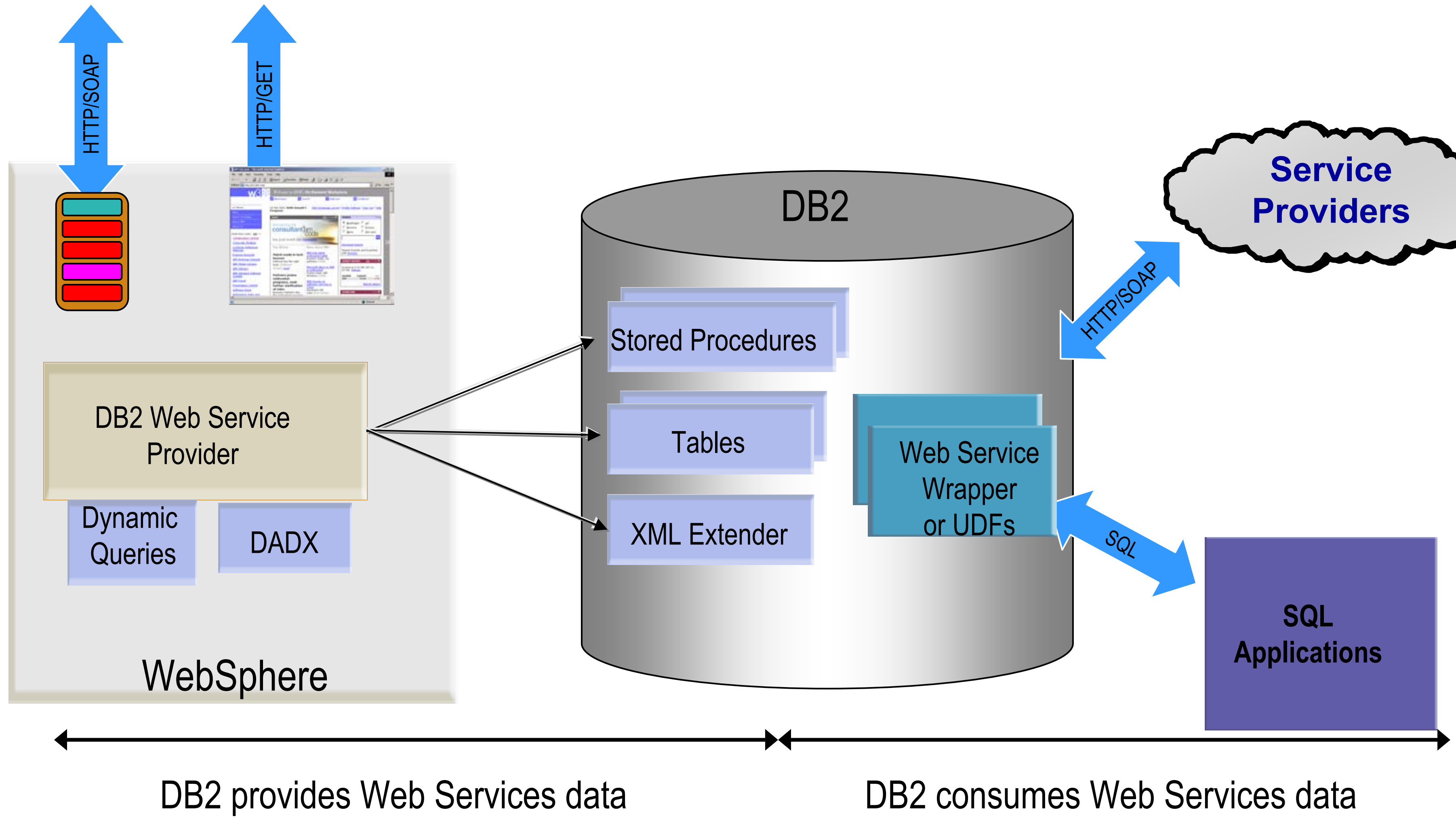
Enterprise Service Bus – Physical Model



Enterprise Service Business

- Currently a *Pattern* on existing products, evolving to an integrated product.
  - WMQ, WMQI
  - WebSphere Platform Messaging
- WebSphere Platform Messaging
  - Based on JMS
  - Currently two implementations
    - JMS → MQ Series
    - All Java/JDBC impl. In WebSphere 6.0
  - Integrates into MQ Networks (MQMs)
- Enterprise Service Bus
  - Adds support for *mediated destinations* and *topics*
  - Transform, route, augment, side effects, etc. for in-flight messages
  - Support for WSDL, WS-Policy and WS-I protocols
  - "Wires" between services logically flow through the "bus."

DB2 (II) Web Services Overview

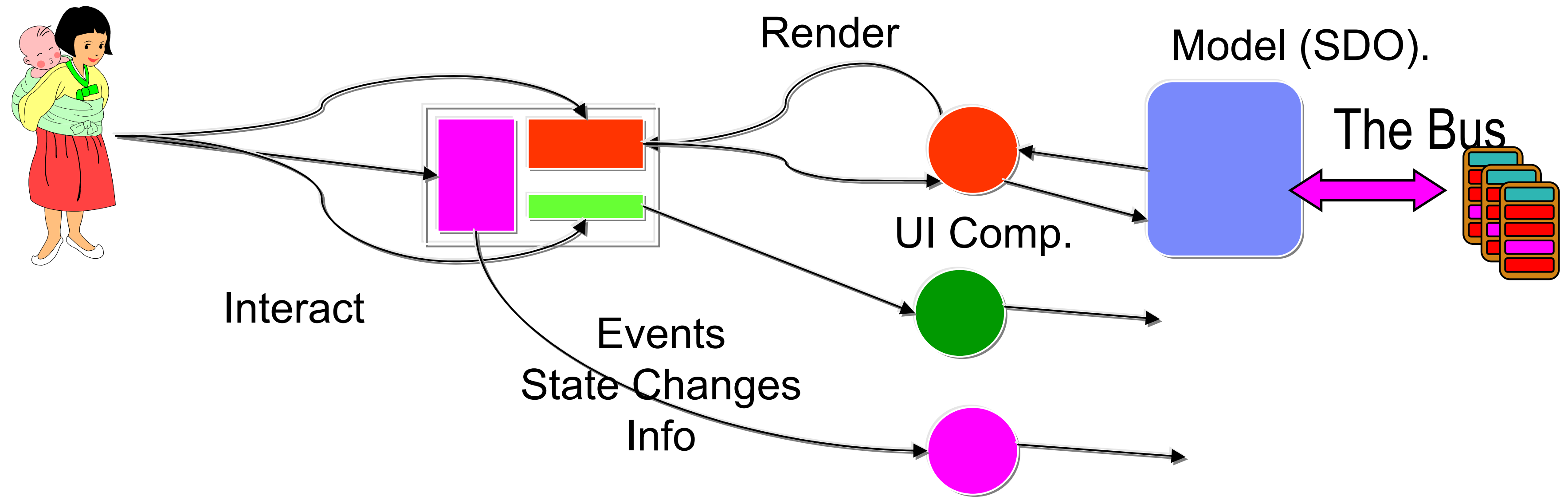


SOA, Web services and "Data"

- Simple tool support (no-programming) to expose
  - SQL Queries
  - Stored Procedures
  - XML Extender

  as Web Services.
- DB2 II provides support for *consuming* Web services
  - Integrated "XML" data sources into the information model.
  - DB2 programmer sees a "SQL" model only
  - Web service parameters/operations appear in SQL through
    - Nicknames and/or
    - UDFs
  - Tool support for bridging between WSDL and SQL
- Content Manager
  - BPEL support for approval workflows
  - Moving to WBI Modeler integration
  - Moving to exploitation of WBI engine

Java Server Faces



- 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.); WYSIWYG Tool
- Builds on what we do today (JSP, Struts, Portlets)

Some Clarification

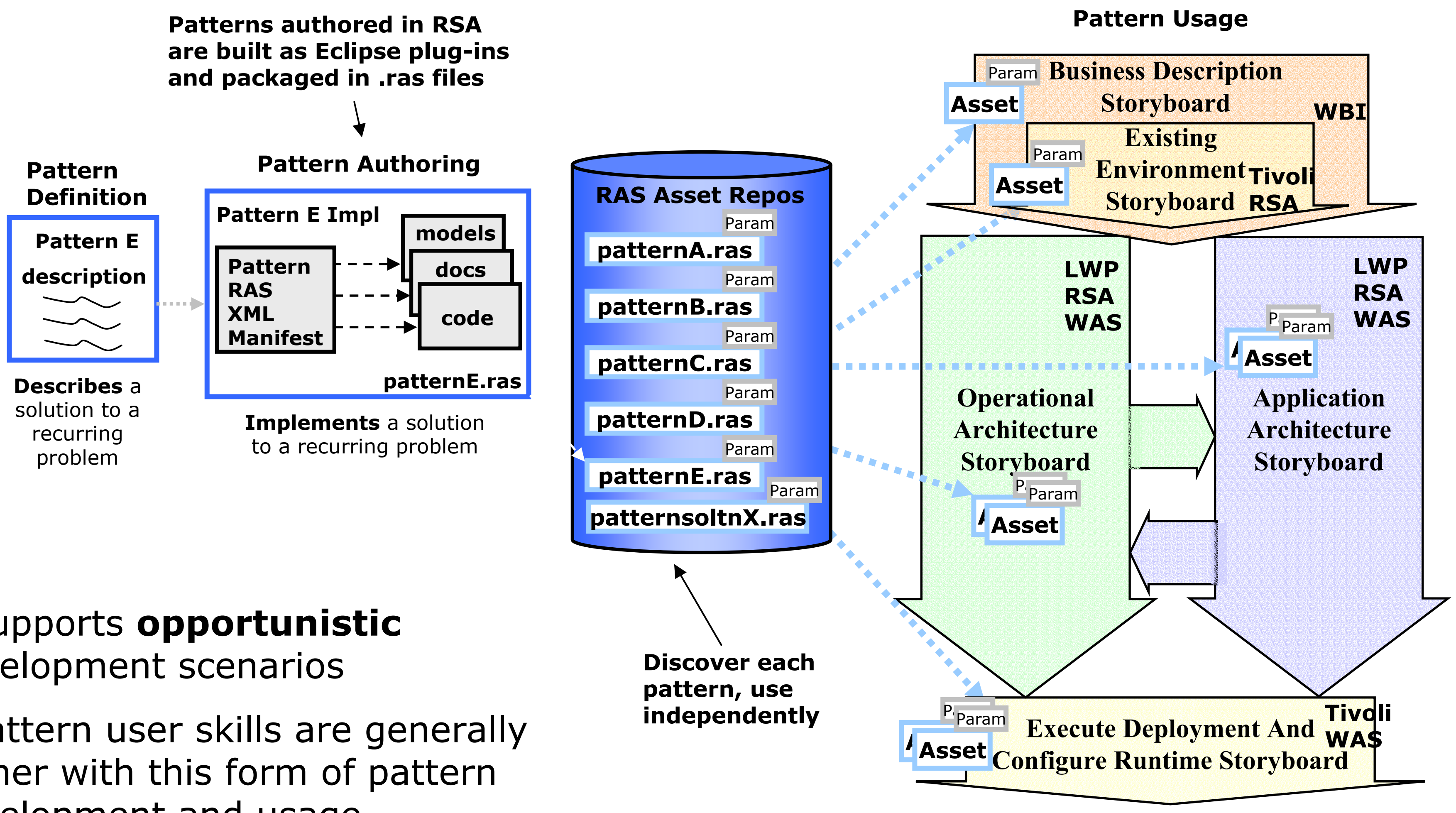
- An *Asset* is, well an Asset. Can be anything
  - Word document, PowerPoint Presentation
  - Handy code that I keep lying
  - Excel spreadsheet for costing
  - … …
- A *Pattern* is a recurring solu
  - Patterns for eBusiness (http:
  - Enterprise Integration Patter
  - J2EE Patterns (http://corej2e)
  - … …

  Read the book and start typing!

This is not really as helpful as we can be!

- A *Template* is a Pattern (or sub-pattern) that
  - Has associated metadata
  - Comes with a design time control (Wizard)
  - Uses code generation or "data driven behavior" to convert to an instance.
- A *Recipe* is an directed graph of *Templates,* with composite controls
  - Which arcs to follow
  - Metadata flows through the graph as you follow the recipe
  - Subsets, augments, modifies the constituent patterns.
- A Solution Template is
  - A complete solution, with install images
  - Well-defined POVs for tailoring the elements and wizards

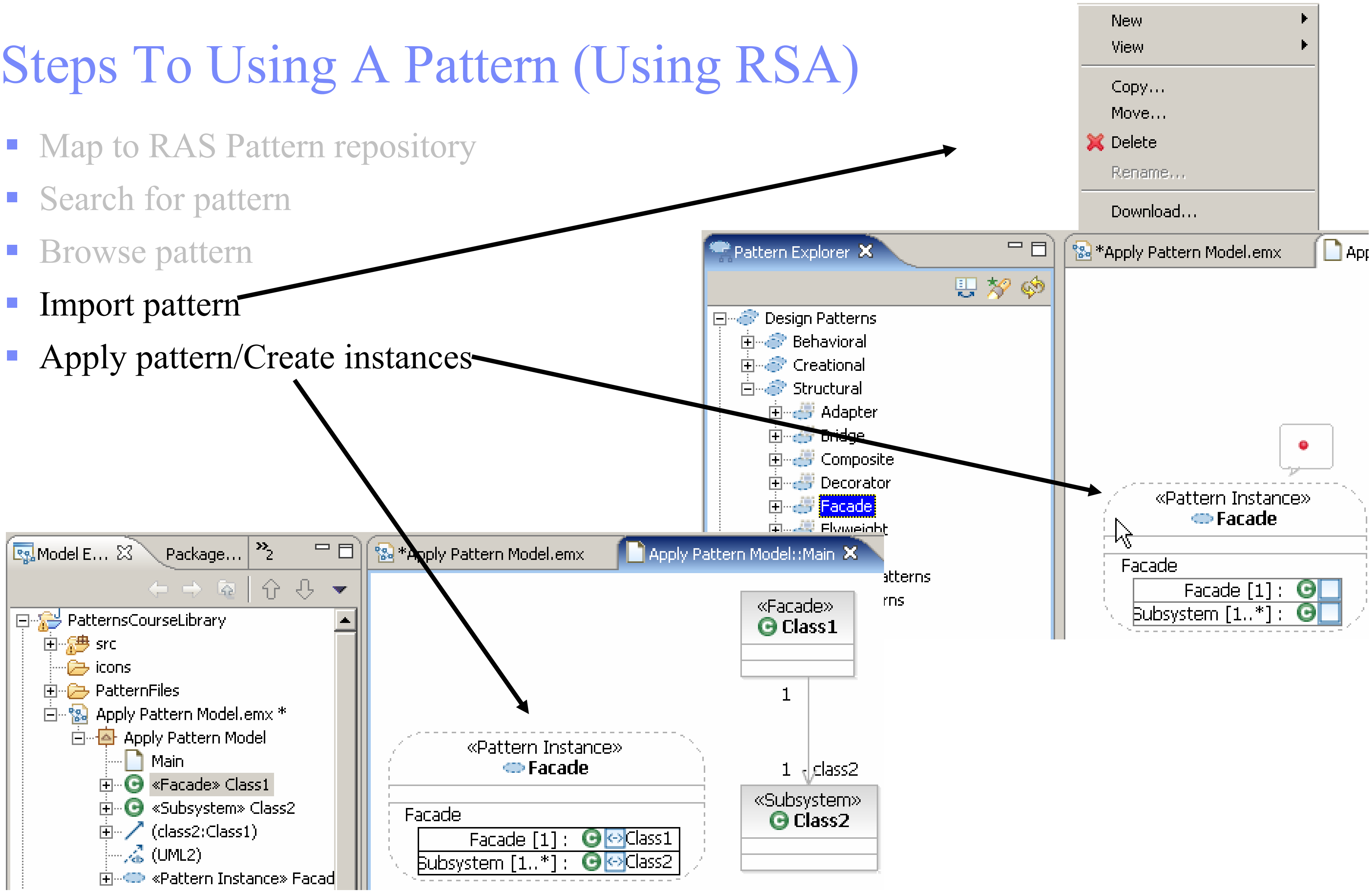
Implementing And Using Patterns


Patterns authored in RSA are built as Eclipse plug-ins and packaged in .ras files
Pattern Definition
Pattern E description
Describes a solution to a recurring problem
Pattern Authoring
Pattern E Impl
Pattern RAS XML Manifest
models
docs
code
patternE.ras
Implements a solution to a recurring problem
RAS Asset Repos
Param
patternA.ras
patternB.ras
patternC.ras
patternD.ras
patternE.ras
patternsoltnX.ras
Discover each pattern, use independently
Pattern Usage
Asset
Business Description Storyboard
WBI
Existing Environment Storyboard
Tivoli
RSA
LWP RSA WAS
Operational Architecture Storyboard
Application Architecture Storyboard
Execute Deployment And Configure Runtime Storyboard
Tivoli WAS

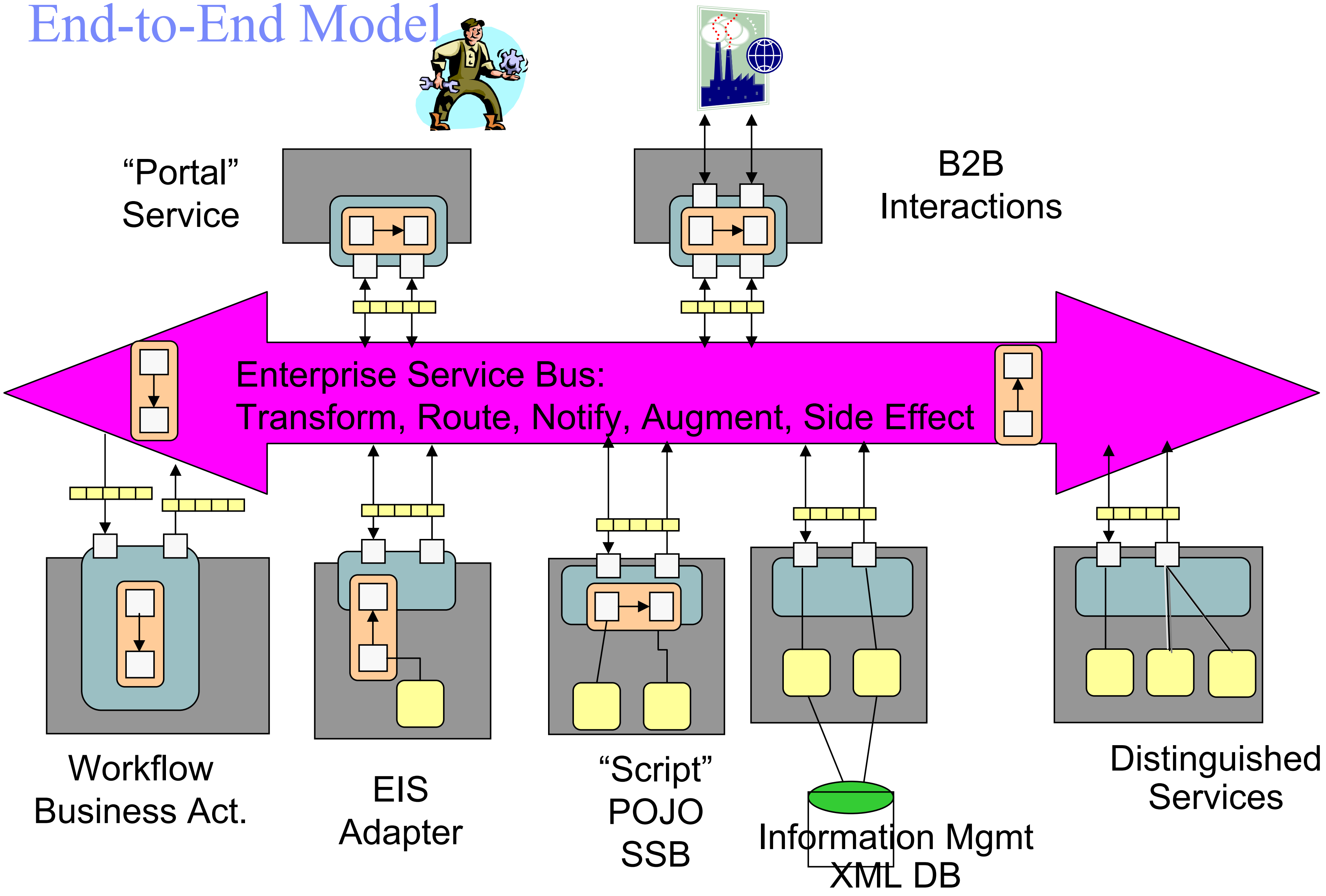
- Supports **opportunistic** development scenarios
- Pattern user skills are generally higher with this form of pattern development and usage

Steps To Using A Pattern (Using RSA)

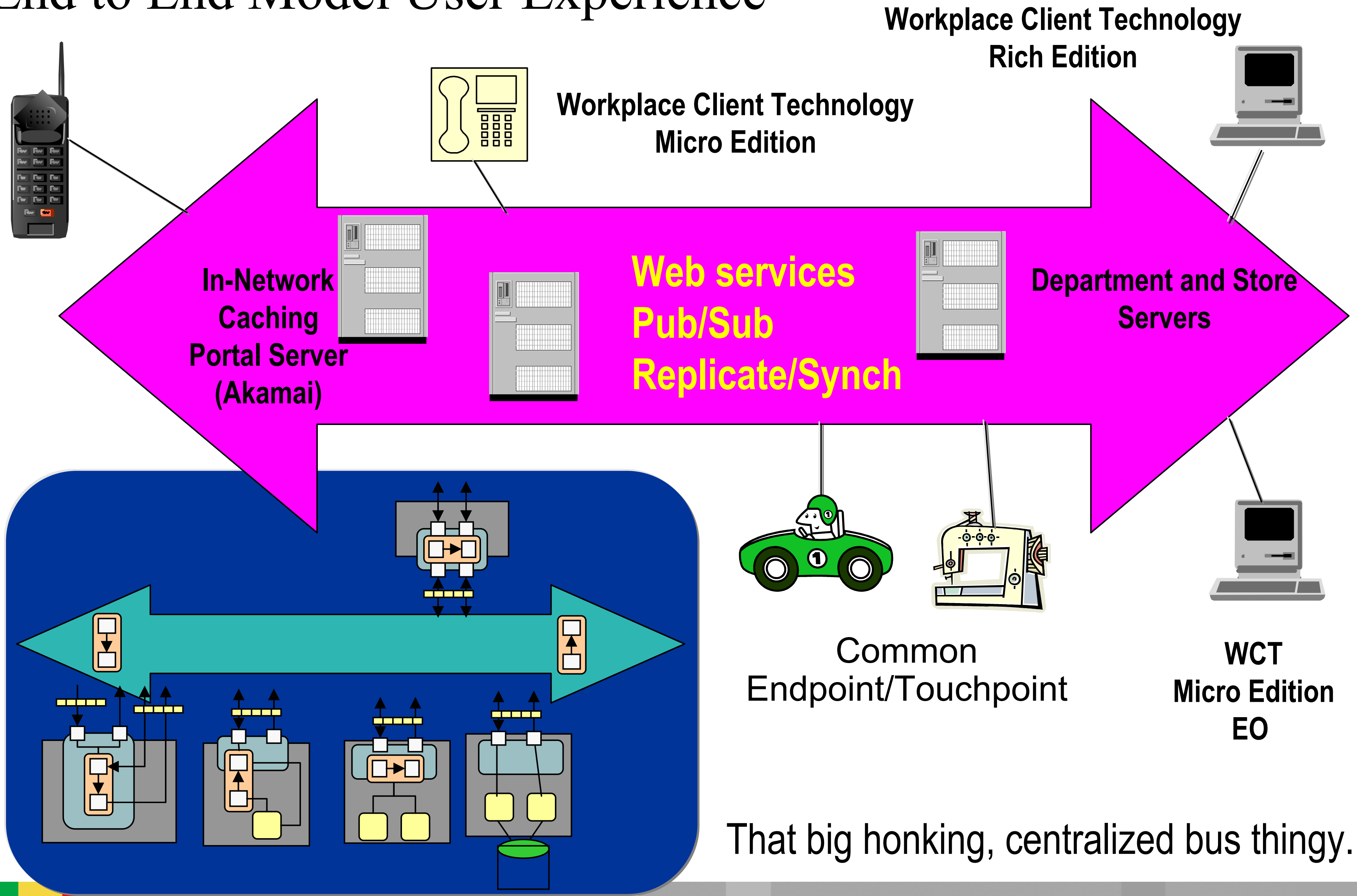
- Map to RAS Pattern repository
- Search for pattern
- Browse pattern
- Import pattern
- Apply pattern/Create instances



End-to-End Model



End to End Model User Experience



That big honking, centralized bus thingy.

Product Architecture

- Our products are
  - A set of containers that host Service Components
  - Several different types of Container
    - Business Process Choreographer
    - CICS
    - WebSphere
    - DB2
    - … …
- The Enterprise Service Bus connects all components (and containers)
  - Logical concept {WebSphere, WPM, MQ, WMQI}, evolving to
  - A product – Whitewater
- The model supports "clients"
  - WCT
  - Touchpoints
  - In-network servers
- Our tools are evolving to support patterns, recipes and templates.
- There will increasingly be a set of *distinguished services/containers*

Summary and Discussion