Websphere User Group

WPS Implementation Experiences

Author: Kim Clark
Email: kim.clark@uk.ibm.com
Date: 27th Feb 2007

Objective

- **Leverage project WebSphere Process Server experiences to provide design and implementation guidance on future projects**

Overview

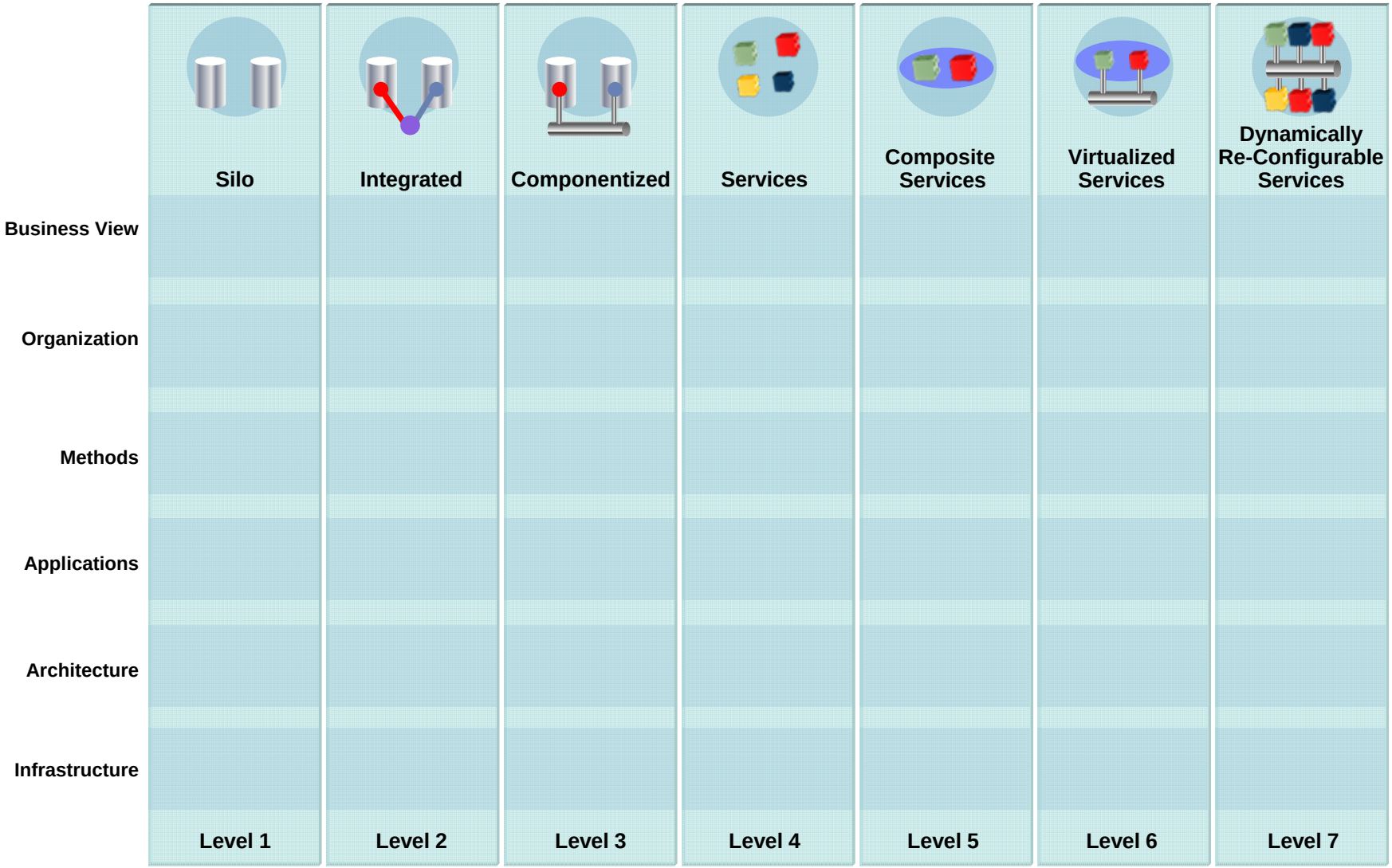
- **Example usage scenarios for WPS**
- **Common experiences from SOA/WPS projects**
- **Features introduced in WPS 6.0.2**
- **Design work products**
- **Design tips**

Know your audience!

- **How many people have been trained in, or have hands on experience with WebSphere Process Server**

- **How many people are involved with a WebSphere Process Server project leading to, or actually in production**
  - If you don't yet know me, I'd like to speak to you today!

Service Integration Maturity Model (SIMM)



Common usage types for WPS/WESB

- **SOA - Business Process Automation / Service Chorography (~ SIMM Level 5)**
  - "Headless" straight through processes
  - Task based process portals
  - GUI centric integration portals
- **SOA – Service Exposition (~ SIMM Level 4)**
  - Exposing of existing services
  - Creating and exposing of new services
- **EAI (~ SIMM Level 3)**
  - Event driven data synchronisation (BPEL)
  - Asynchronous data aggregation (BPEL)
- **Brokerage and Connectivity (~ SIMM Level 2)**
  - Routing/Transformation (WESB)
  - Connectivity across platforms/protocols (HTTP/JMS/MQ…)
  - Application specific connectivity (Websphere Adapters)
  - Content based routing

Usage types are typically combined. In fact, many usage types used in isolation actually represent an "anti-pattern"!

Potential Anti-Patterns for WPS

- Very long-lived processes (years)
- Extract Transform Load / Data Migration
- Over complex individual BPEL processes
- High java content in BPEL
- Point to point connectivity only
- Movement of very large objects
- BPEL for complex algorithms better suited to java code
- Fine grained services
- Services based on complex GUI data aggregation, coupled to individual UIs
- GUI navigation controlled by BPEL/BSM
- B2B handshaking, where other products such as WebSphere Partner Gateway are specifically appropriate

Throughput vs. Response time

Throughput based applications are architected, designed, implemented, optimised and managed differently to **Response Time** based applications

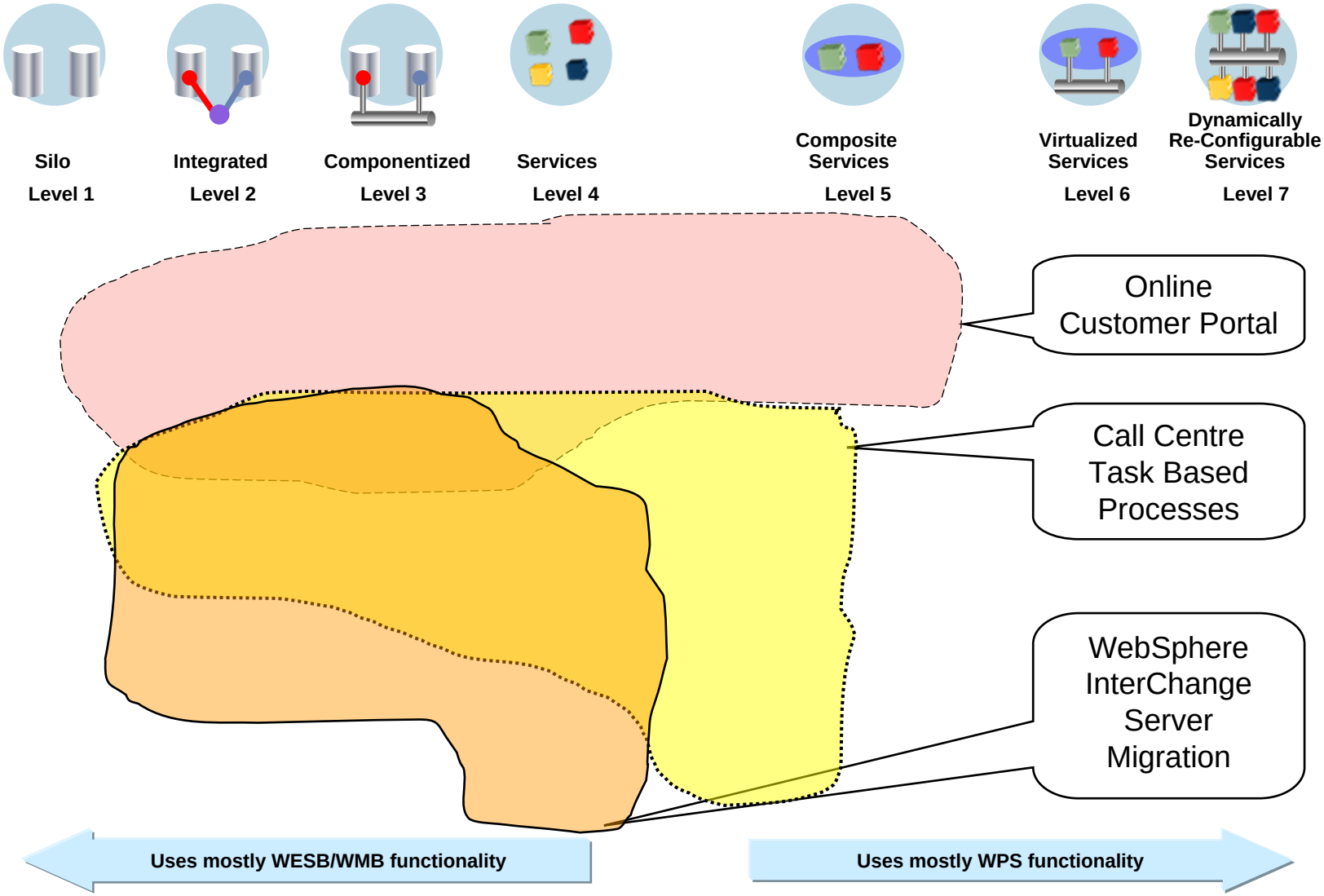
- **Throughput based**
  - Typically designed asynchronously using messages
  - No user awaiting response
  - SLA Measured in events per time period (e.g. processes 100,000 messages per hour)
  - Pace of an individual process irrelevant
  - Performance tuned for high CPU utilisation
  - Thread count should be minimised.
  - Concurrency only related to performance tuning.
  - Errors must be handled/resolved
  - Servers handle retries
  - Typically stateful. Survives server restart.

- **Response time based**
  - Typically designed synchronously
  - Typically a user awaits the response
  - SLA Measured in time taken for response to arrive (e.g. user waits no more than 3 seconds)
  - Performance tuned to handle peaks
  - Queue depths should be minimised
  - Concurrency relates closely to the number of expected clients.
  - Errors can often be thrown away
  - Clients handle re-tries
  - Largely stateless. Transaction doesn't survive server restarts – although the session might.

Analyse Usage Patterns from Two Dimensions

- **Dimension 1: SOA Maturity**
  - based on the 7 levels in the "Service Integration Maturity Model" (SIMM)
- **Dimension 2:**
  - Response Time Based
    - typically synchronous traditional J2EE
  - Throughput Based
    - typically asynchronous message based

WPS Usage Analysis – Common Scenarios



Experiences from other Geographies

- **Asia**
  - Labour is cheaper so more focus on Human Task Management (HTM)
  - More focus on management control rather than process automation
  - ERP is relatively new so less focus on Enterprise Application Integration (EAI) / legacy integration
  - Commonly use Business State Machine + HTM

Design Issues

Application Design

- Business Processes
- Dynamicity in Long Lived Processes
- Data Models
- Dynamic Logic
- Layered Architecture
- Service Composition/Component Boundaries
- Graphical User Interfaces
- Exception Handling
- Logging
- Batch/Staging Considerations
- …

Infrastructure Design

- Deployment Topologies
- Sizing
- Performance
- Resilience
- Disaster Recovery
- Security
- Monitoring
- Scripted Installation
- …

Development Env.

- Test Automation
- Build Automation
- Version Control
- Team Structure
- …

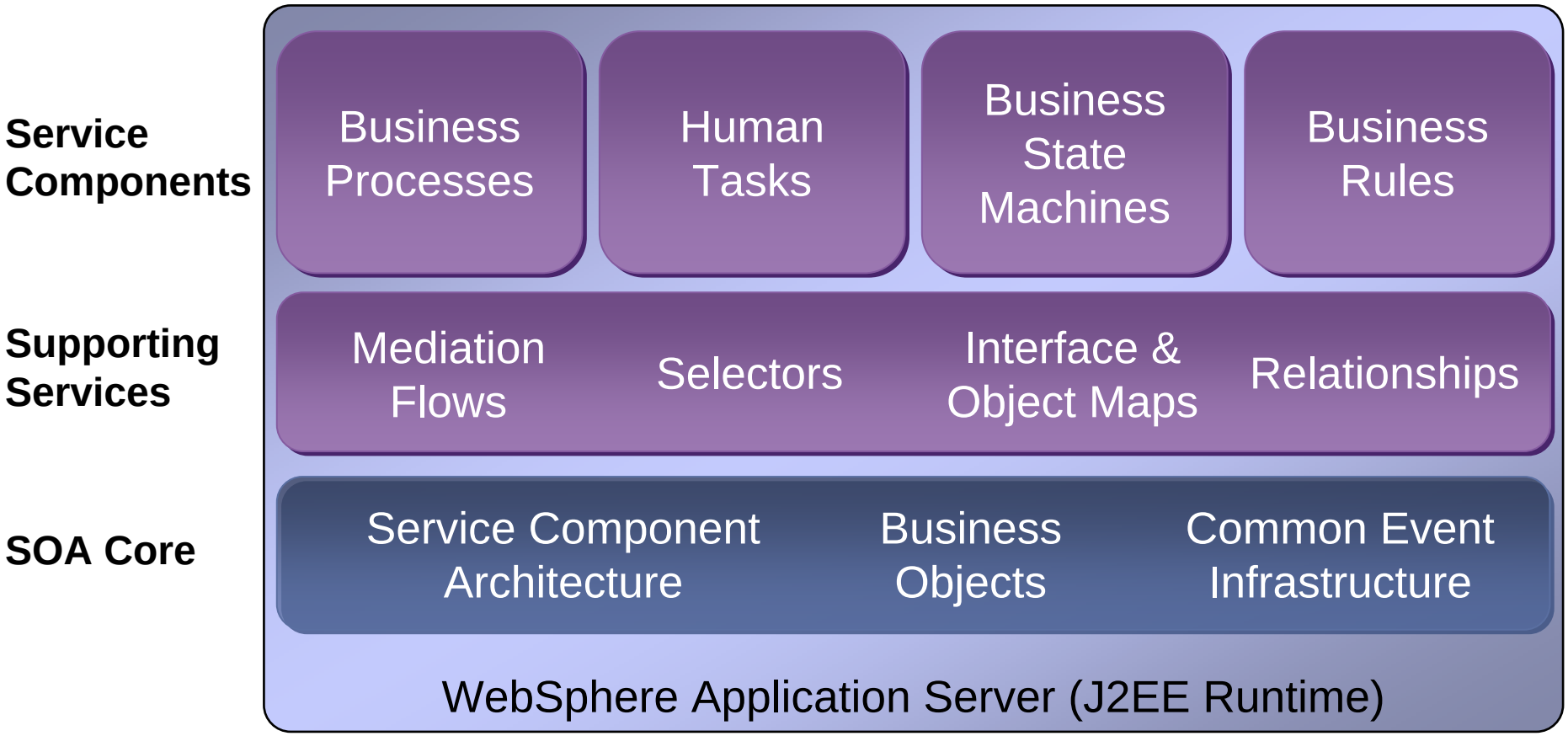
Methodology

- Phases
- Process
- Work Products
- …

Example Design Issues – Business Processes

- **Modeling Techniques**
  - What are the options for modelling business processes and when is each of them appropriate. e.g. WBI Modeler, BPEL, UML, Model Driven Business Transformation (MDBT)/Artefact Based Modeling
- **Process Types**
  - When to use long lived rather than short lived. When to use a state machine rather than a business process.
- **Process Granularity**
  - What are appropriate techniques for handling business processes that are very complex and/or long lived? What are the best ways to reduce or distribute the complexity e.g. Process with Callable Sub-processes, Chained Processes
- **Process Versioning**
  - The requirements for Services and Business processes change over time. We need the ability to adjust the business logic. This presents a number of challenges, especially regarding long-lived processes that have instances in flight. What mechanisms are available to introduce flexibility in the right places.
- **Exception Handling**
  - What guidance do we have in the choice and use of the various exception handling mechanisms in BPEL. e.g. Fault Handlers, Compensation, Failed Events, Event Handlers for termination, Correlation Exceptions, Housekeeping of failed processes, use of Process Administrators, Transaction scope.

Websphere Process Server v6.0.2 – What's new



WebSphere Process Server 6.0.2 – top enhancements

- **Native bindings for MQ**
- **EJB consumption**
- **Process variable visibility**
- **Custom cleanup of completed processes**
- **Human ad-Hoc collaboration**
- **Extended BPC client support to Web Service clients**
- **Productionisation of BPC Observer**
- **Graphical view of processes and instances**
- **Queriable Business State Machine variables**
- **Auditing, and import/export of Business Rules**
- **Event sequencing**
- **Dynamic endpoint addressing**
- **GUI generation for Human Task**
- **Follow-up tasks - scheduling**
- **Server controlled page flow**
- **Uninstall of BP templates with instances**

http://www.ibm.com/fcgi-bin/common/ssi/ssialias?infotype=an&subtype=ca&appname=Demonstration&htmlfid=897/ENUS206-244

Design Issues

Application Design

- Business Processes
- Dynamicity in Long Lived Processes
- Data Models
- Dynamic Logic
- Layered Architecture
- Service Composition/Component Boundaries
- Graphical User Interfaces
- Exception Handling
- Logging
- Batch/Staging Considerations
- …

Infrastructure Design

- Deployment Topologies
- Sizing
- Performance
- Resilience
- Disaster Recovery
- Security
- Monitoring
- Scripted Installation
- …

Development Env.

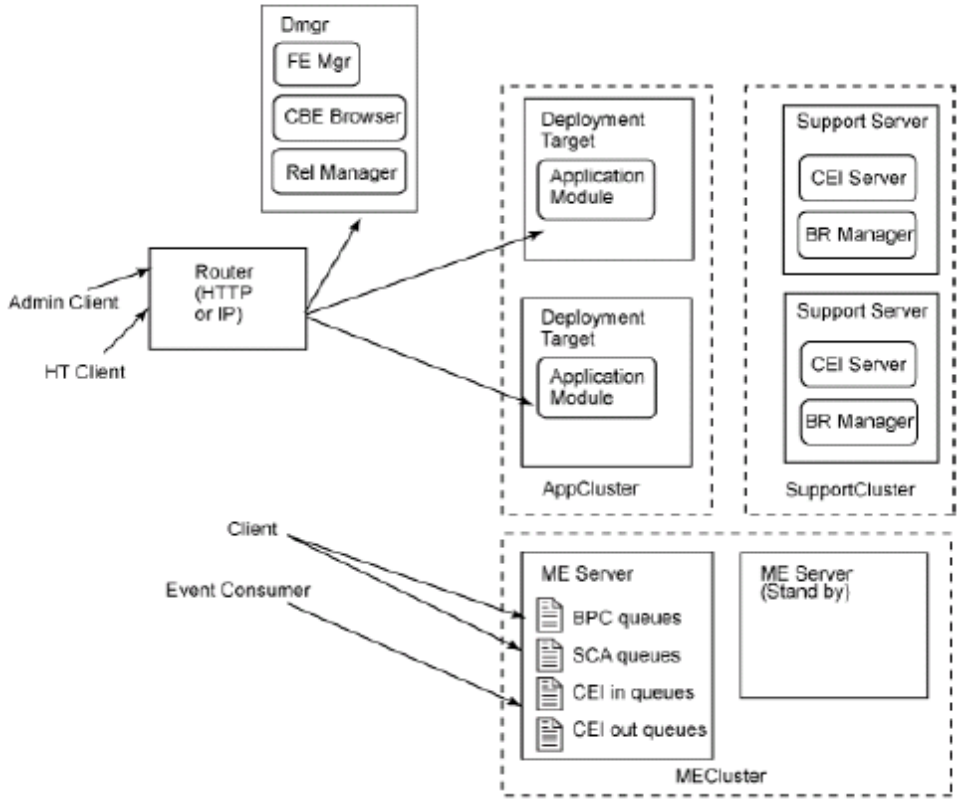
- Test Automation
- Build Automation
- Version Control
- Team Structure
- …

Methodology

- Phases
- Process
- Work Products
- …

Installation Topologies

- **Several topologies**
  - Install for the future
- **Performance**
  - Consider response time vs. asynchronous traffic
  - Hardware sizing
- **Failover**
  - Use HA execution groups for aspects that can't join WAS clustering.
- **Redbook in progress**



Reference: WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns
http://www-128.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html

Automated Testing

- **Process integration applications have a significant number of test paths, especially when you include exception paths**
- **Testing will be required at many levels. At least unit testing, connectivity testing and end-to-end testing.**
- **End system emulation will become critical as you move through environments towards production**
- **Use of iterative development cycles should mandate test automation**
- **See example test framework in 3 part developerWorks article**
  - http://www.ibm.com/developerworks/websphere/techjournal/0608_artus/0608_artus.html

Application Build

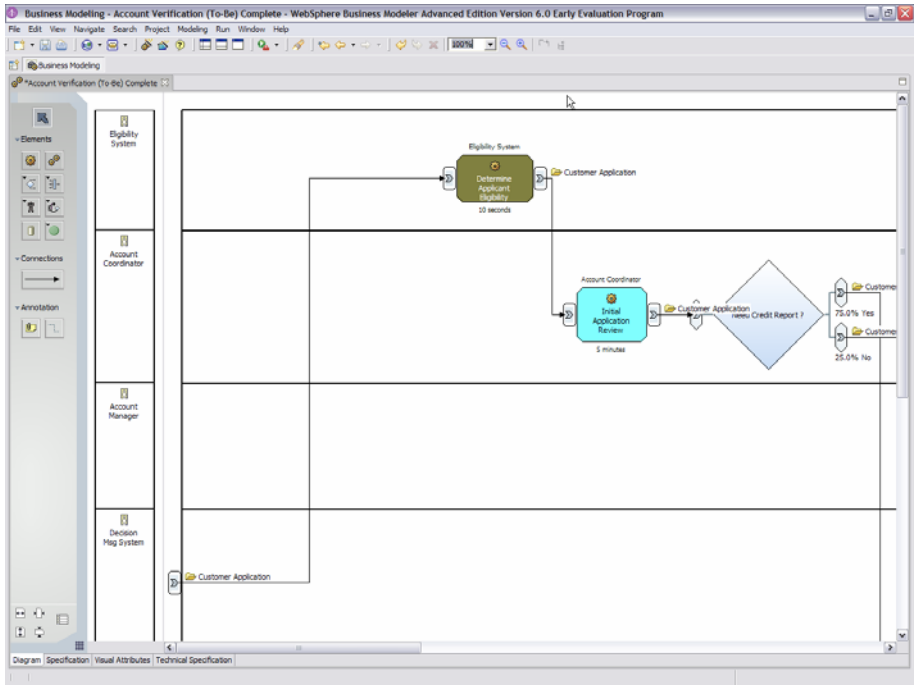
- **Unix based build box**
  - Releases the CPUs from GUI processing
- **Build Automation using “Service Deploy”**
  - Command line based build
  - Also available as an Ant task
  - Enables unattended build

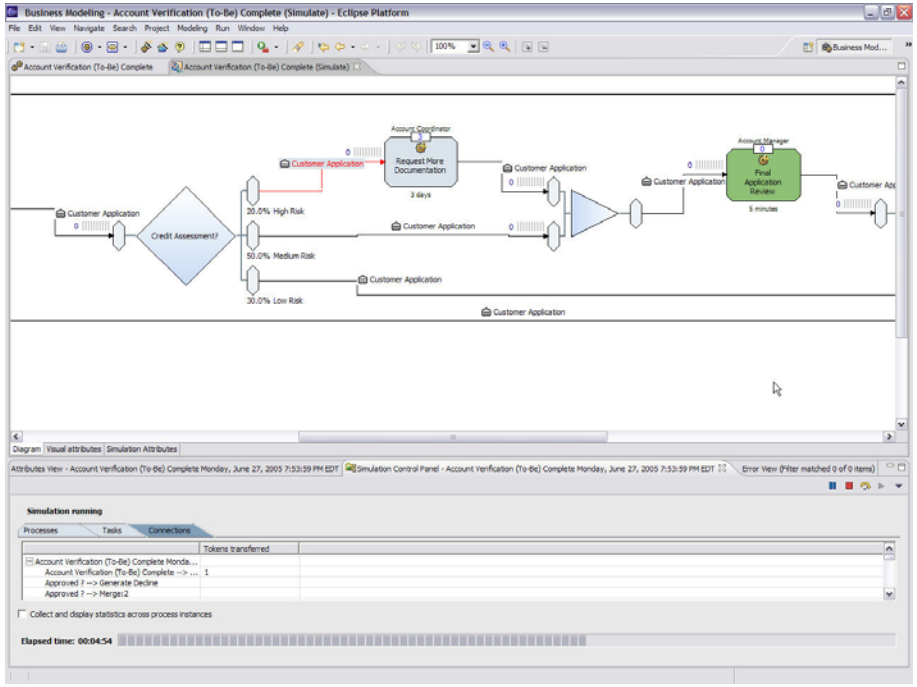
Design Work Products

- **Business Process**
  - Business Process Model or Activity Diagram
  - BPEL Draft
- **Services/Interfaces**
  - Context Diagram
  - Service/Interfaces Catalogue
- **Composition**
  - Data Model
  - Component Model

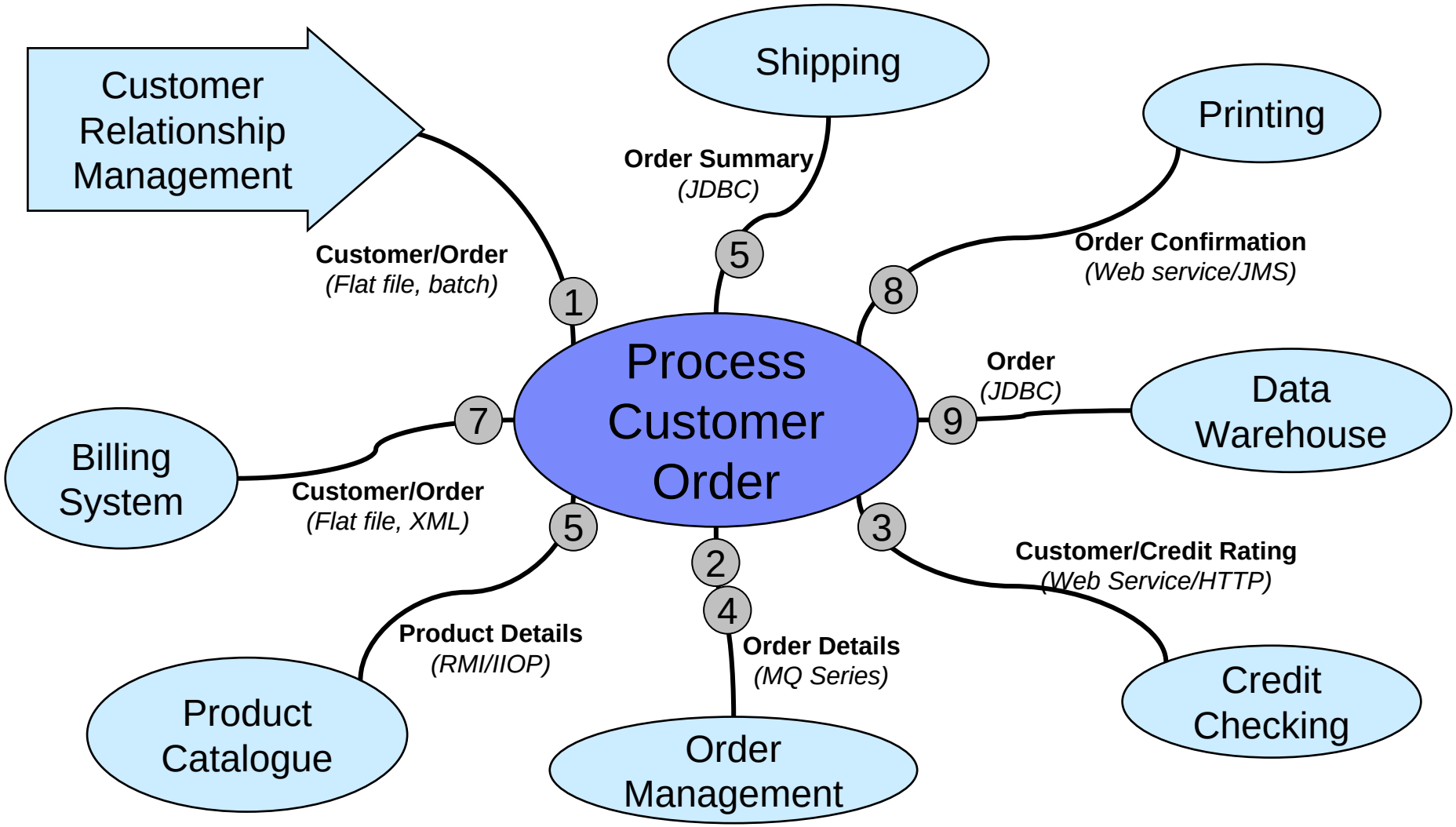
Business Process Modeling

- **Roles**
- **Swim-lane View**
- **Simulation**
- **As-Is/To-Be Comparison**





Basic Context Diagram for a Process

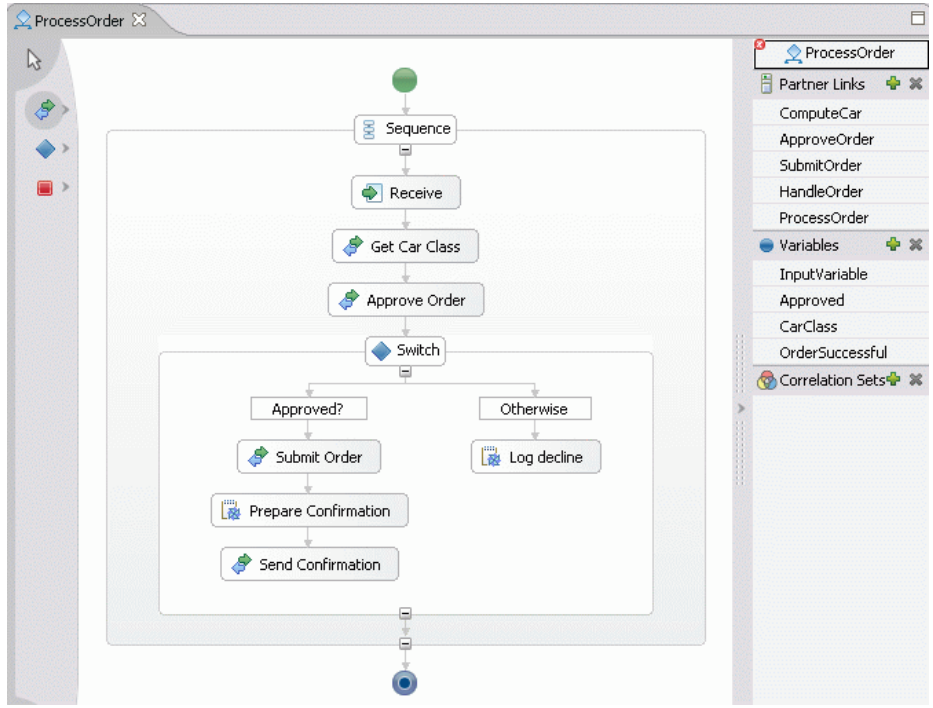


Interface Catalogue - Characteristics

- **Interaction type**
- **Transport**
- **Protocols**
- **Data formats**
- **Principal data objects**
- **Re-usability**
- **Reliability**
- **Availability**
- **Volumes**
- **Time windows**
- **Response times**
- **Service level agreements**
- **Message size**
- **Throughput**
- **Batch or real-time**

Draft Process in BPEL

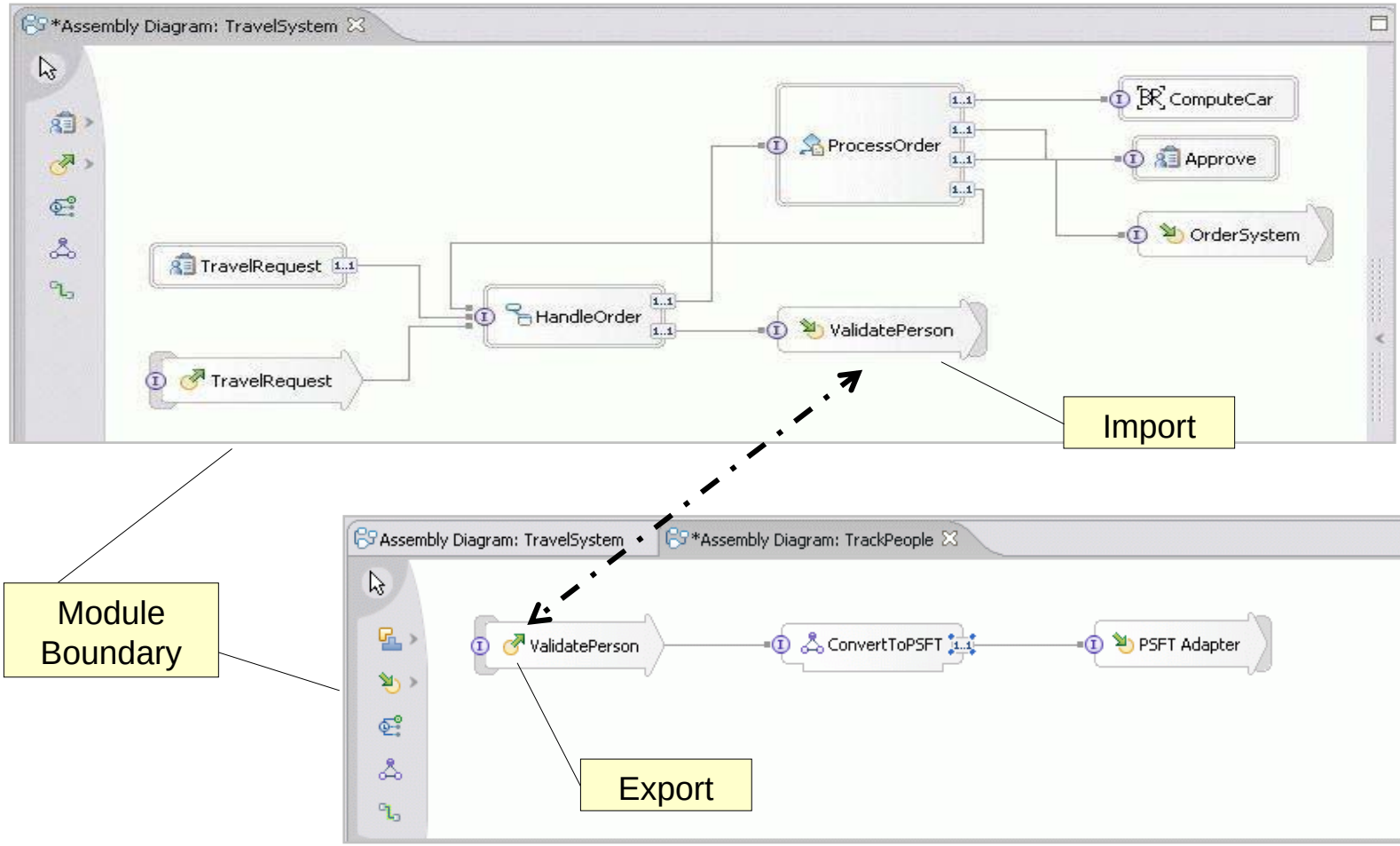
- **BPEL itself can be an effective tool for capturing processes in a workshop**
- **Don’t be to concerned as to whether a construct will “work”, just capture the knowledge of the process.**



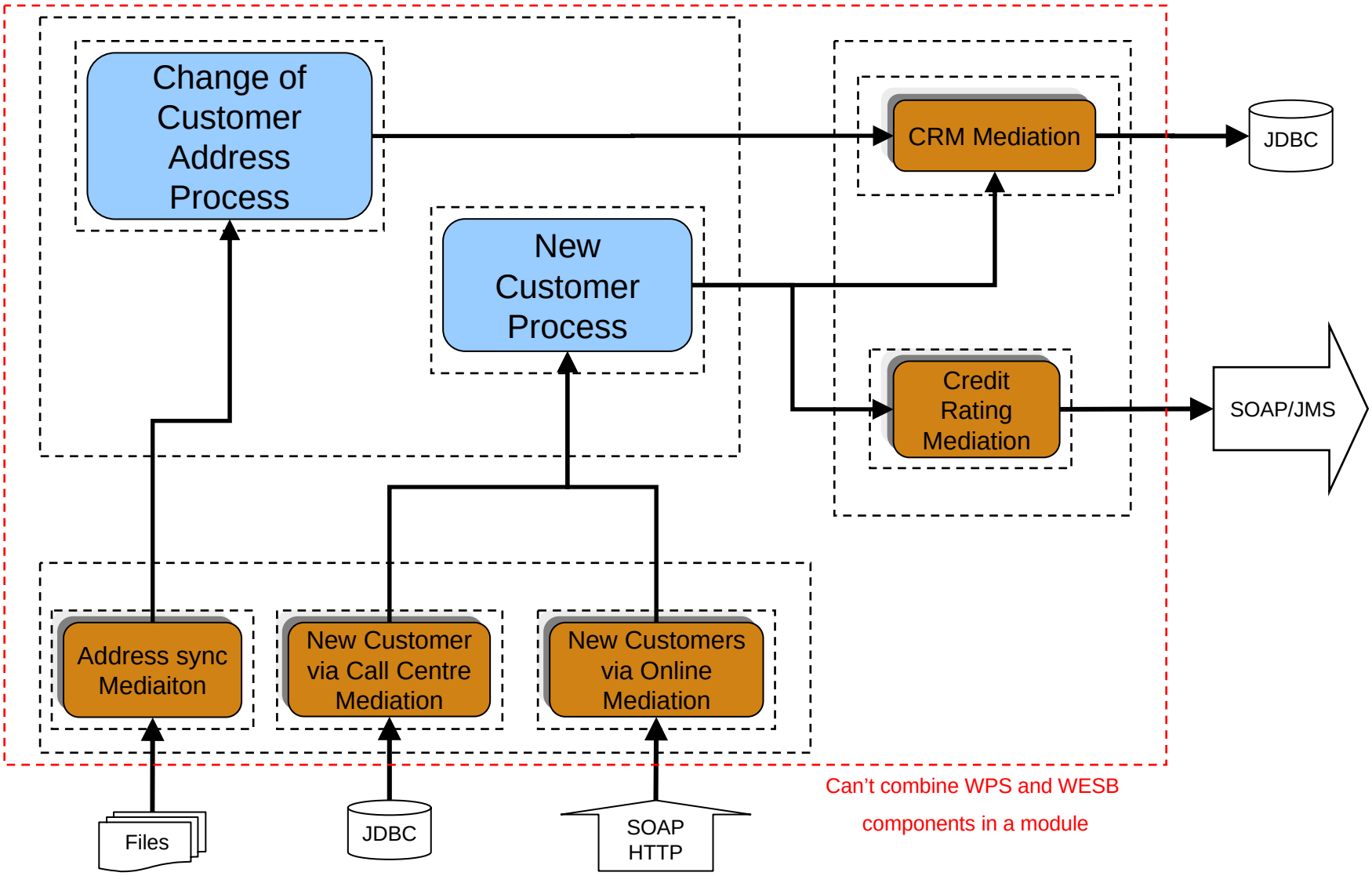
Data Model

- **Create a canonical model for the process that is independent of the services called**
- **At a minimum, copy to a new namespace**
- **Use industry standard models if available**
- **Business analysts should define the logical mapping between namespaces**
- **Create and document the models using WebSphere Integration Developer**

Component Assembly Editor



High Level Component Model – Module Boundaries



Process Considerations

- **Dynamicity and Versioning**
- **Reducing the complexity of Long Lived processes**
- **Container variable pattern**

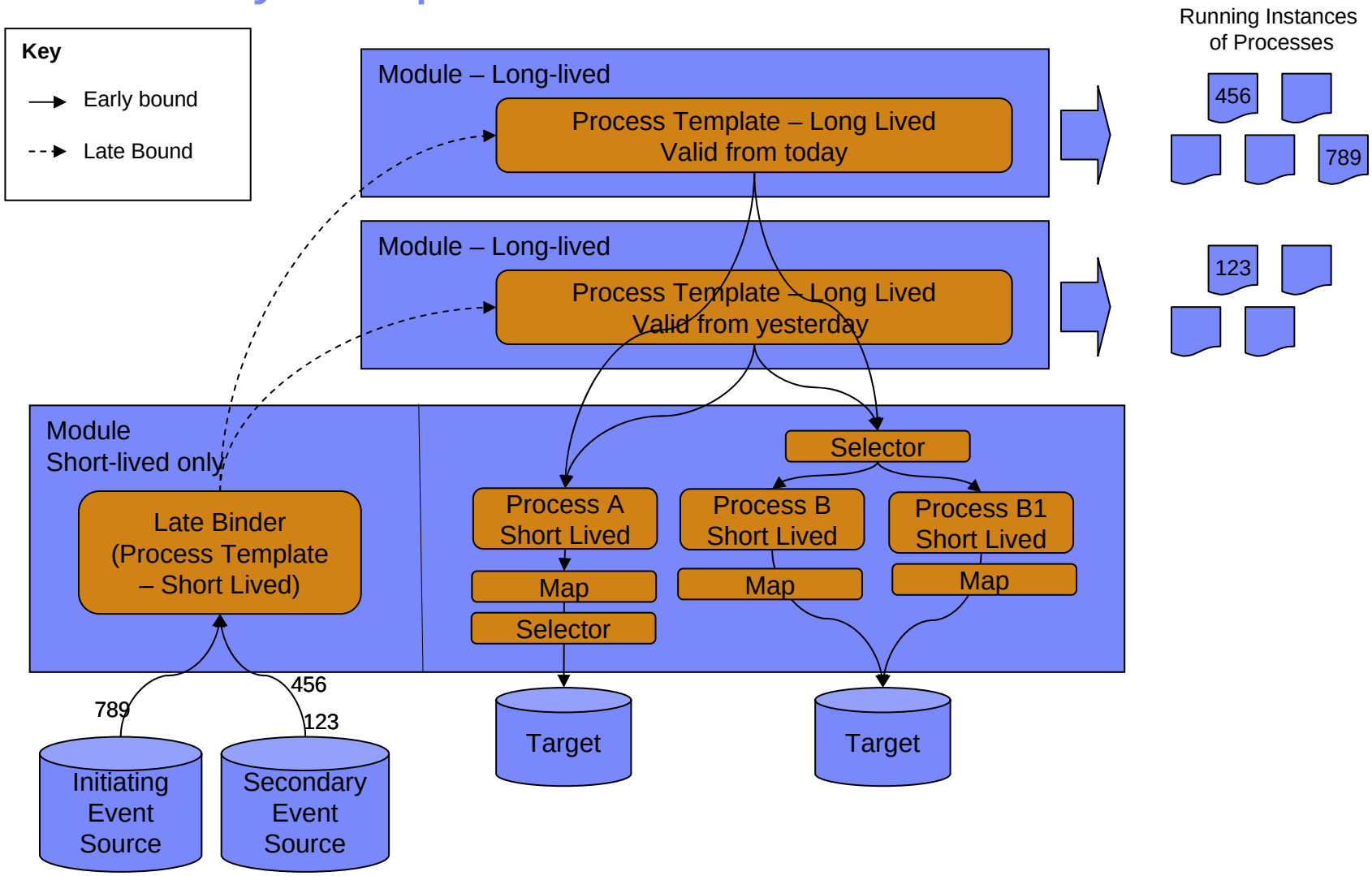
Dynamic Logic

- **Short lived BPEL**
- **Business rules**
- **Relationships**
- **Selectors**
- **Registry lookups**
- **Database lookup**
- **JNDI properties**
- **Java component**
- **Late binding to BPEL (next slide)**

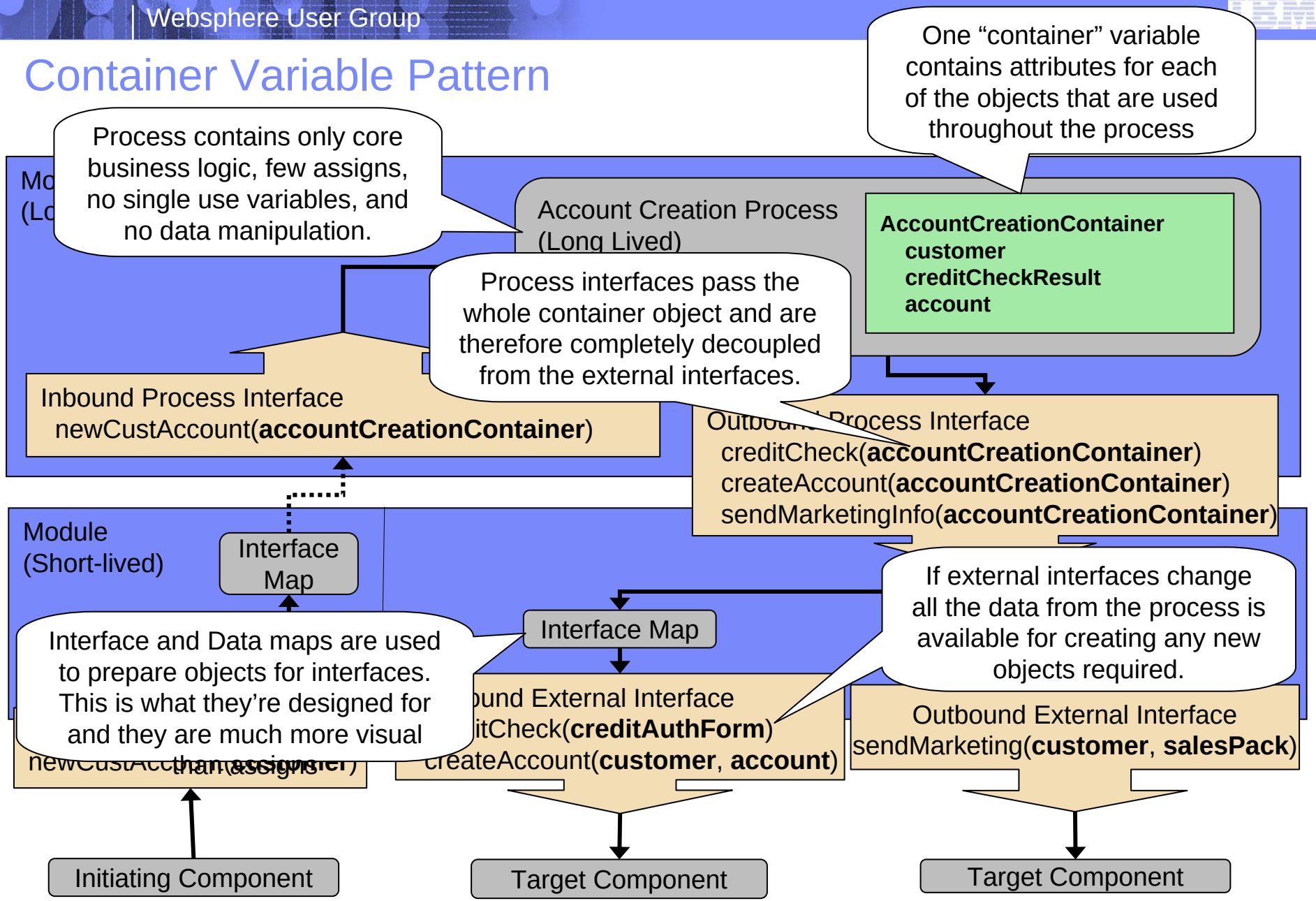
Reference

http://www.ibm.com/developerworks/websphere/library/techarticles/0602_brown/0602_brown.html

Dynamicity – Associated Short-lived Module Other likely components



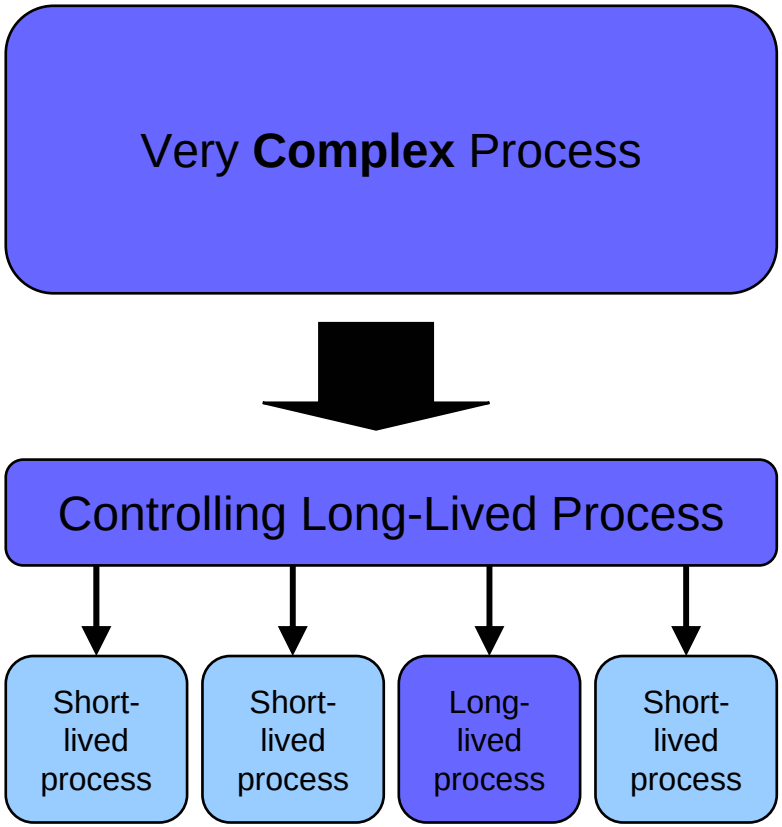
Container Variable Pattern



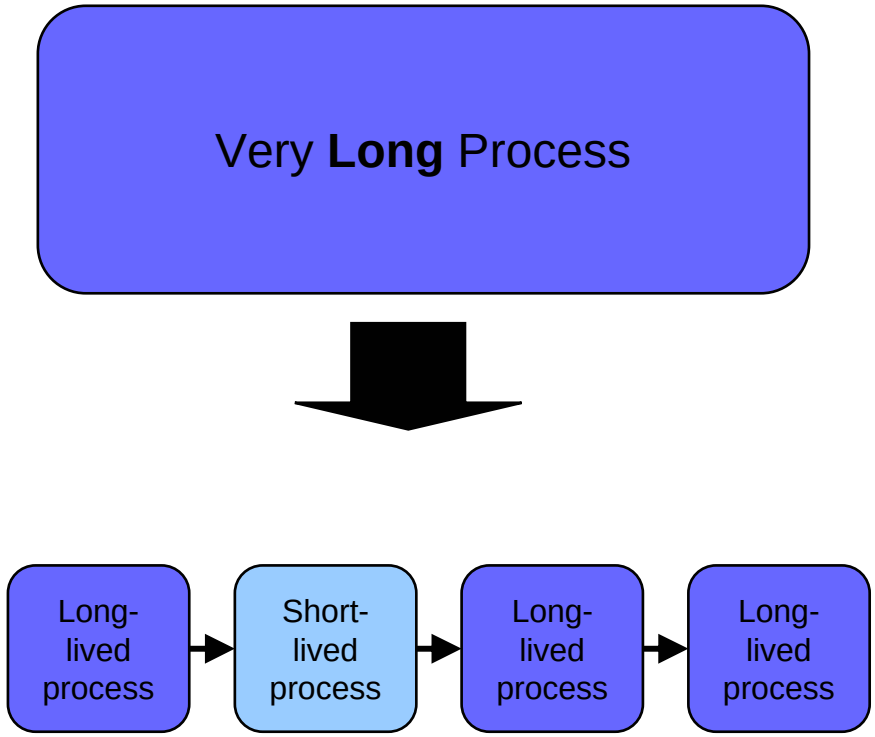
Versioning and Dynamicity: It doesn't have to be this hard!

- **If long-lived is chosen simply to ensure persistence over minutes, hours, or a small number of days**
  - then it may be possible to allow process instances to complete, and the application can be replaced.
- **You can decree that any change to the service will be a new service interface.**
  - but be aware of the version sensitivity that you have pushed back to the client.

Handling Very Long Lived or Complex Processes

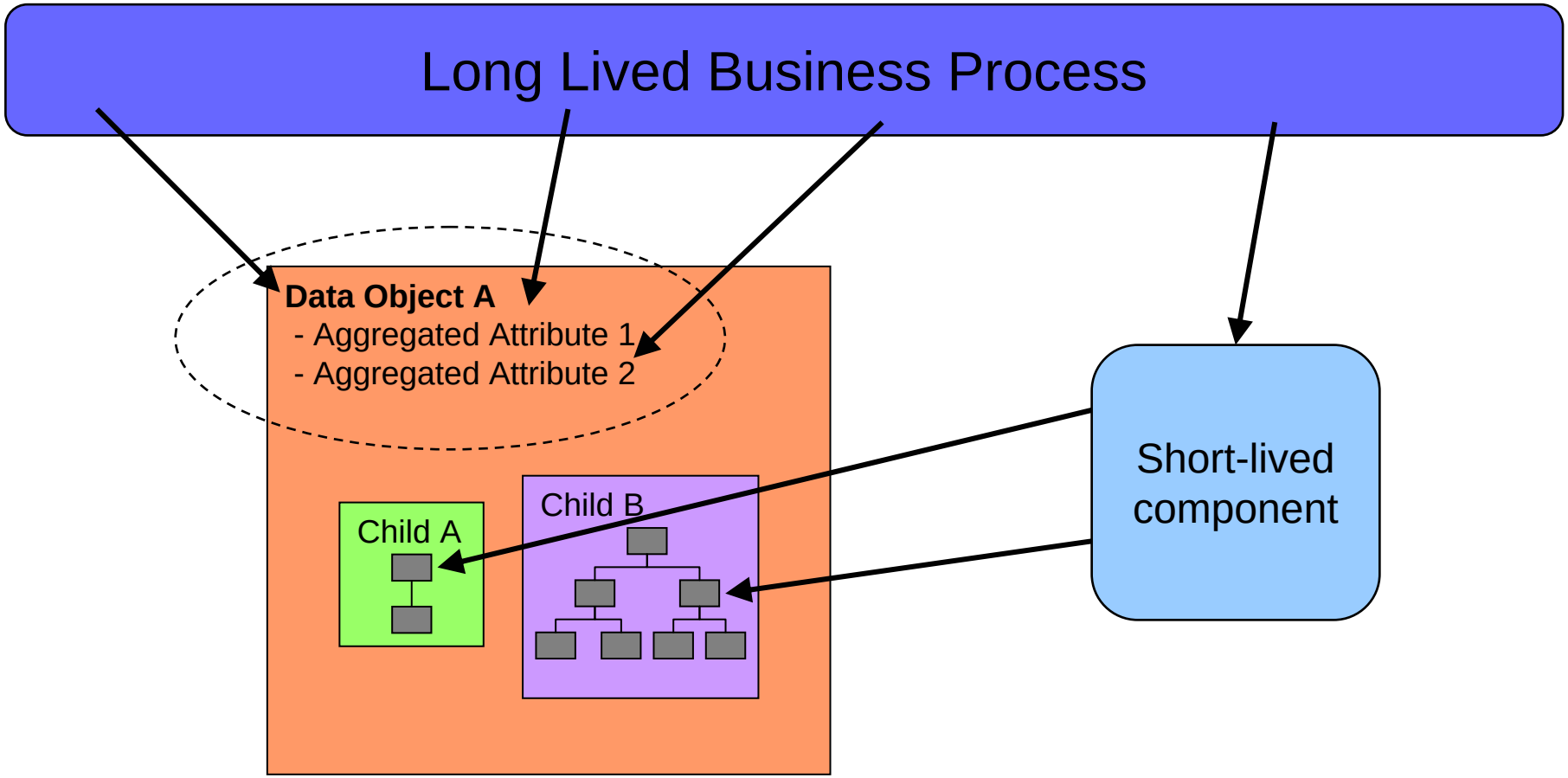


- **Callable Sub-processes**



- **Chained Processes**

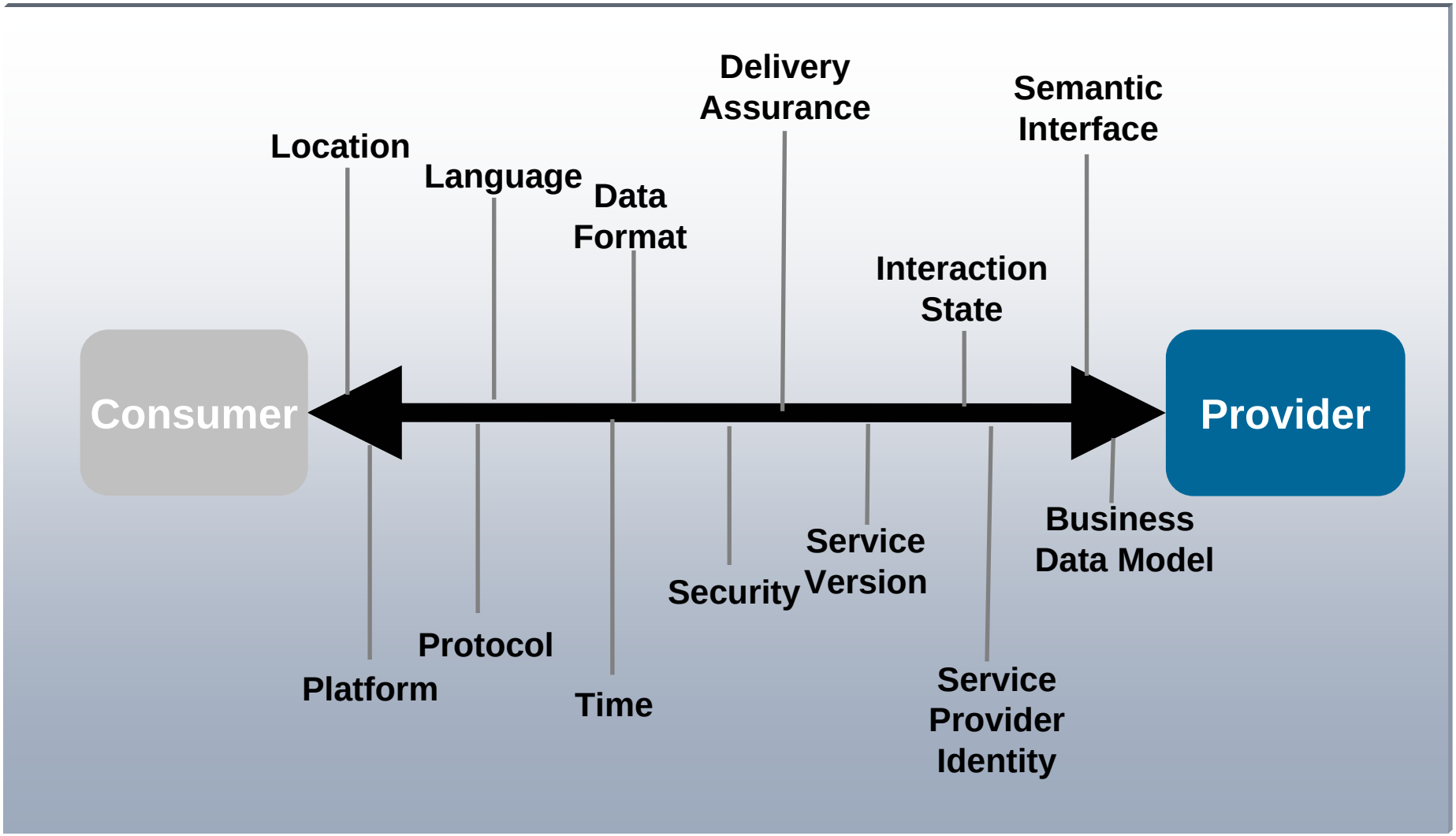
Extracting complexity from BPEL processes



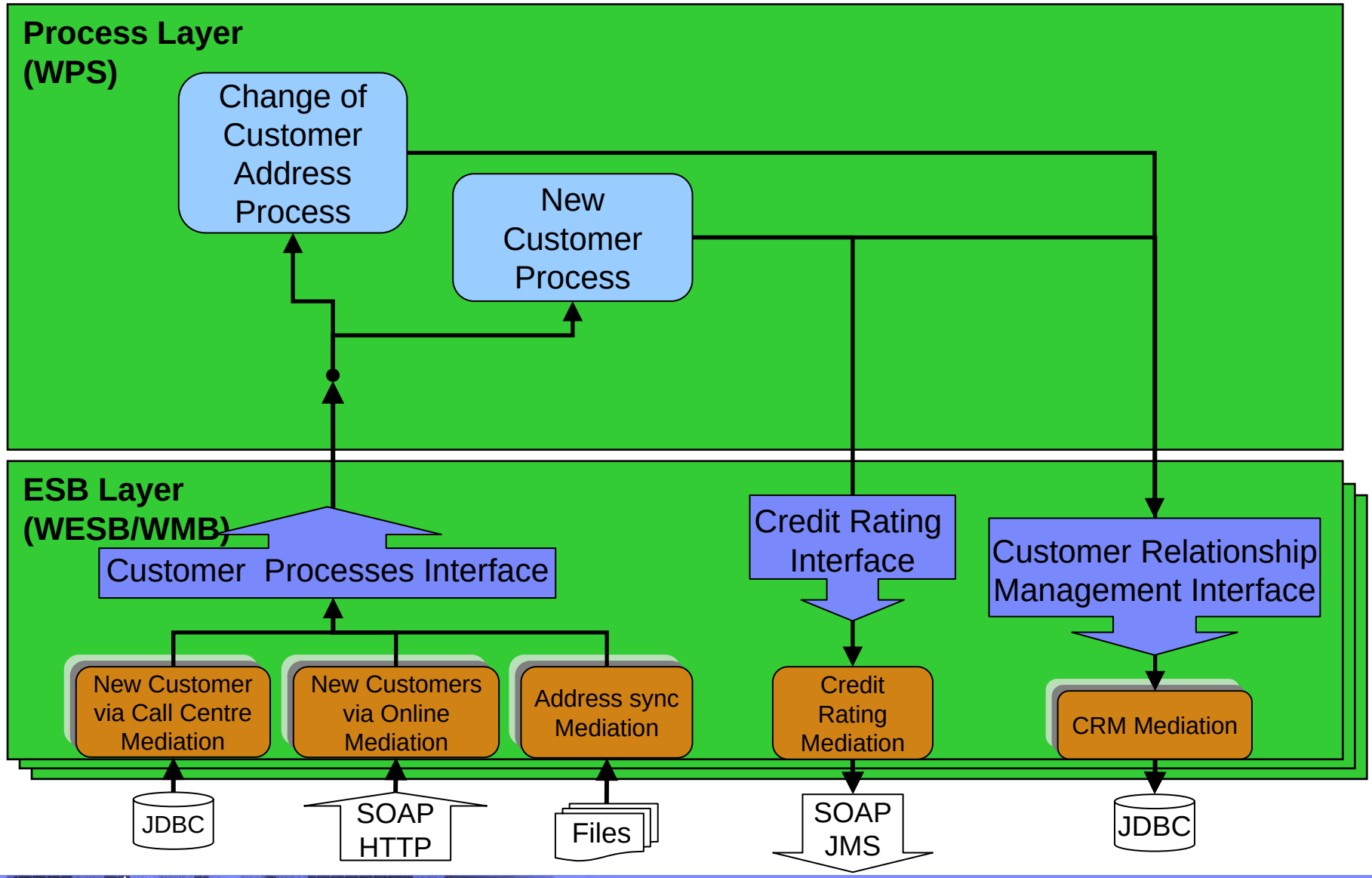
De-coupling

- **Layered Architecture**
- **De-coupling Process from Enterprise Service Interface**
- **When are “Local” Services valid**

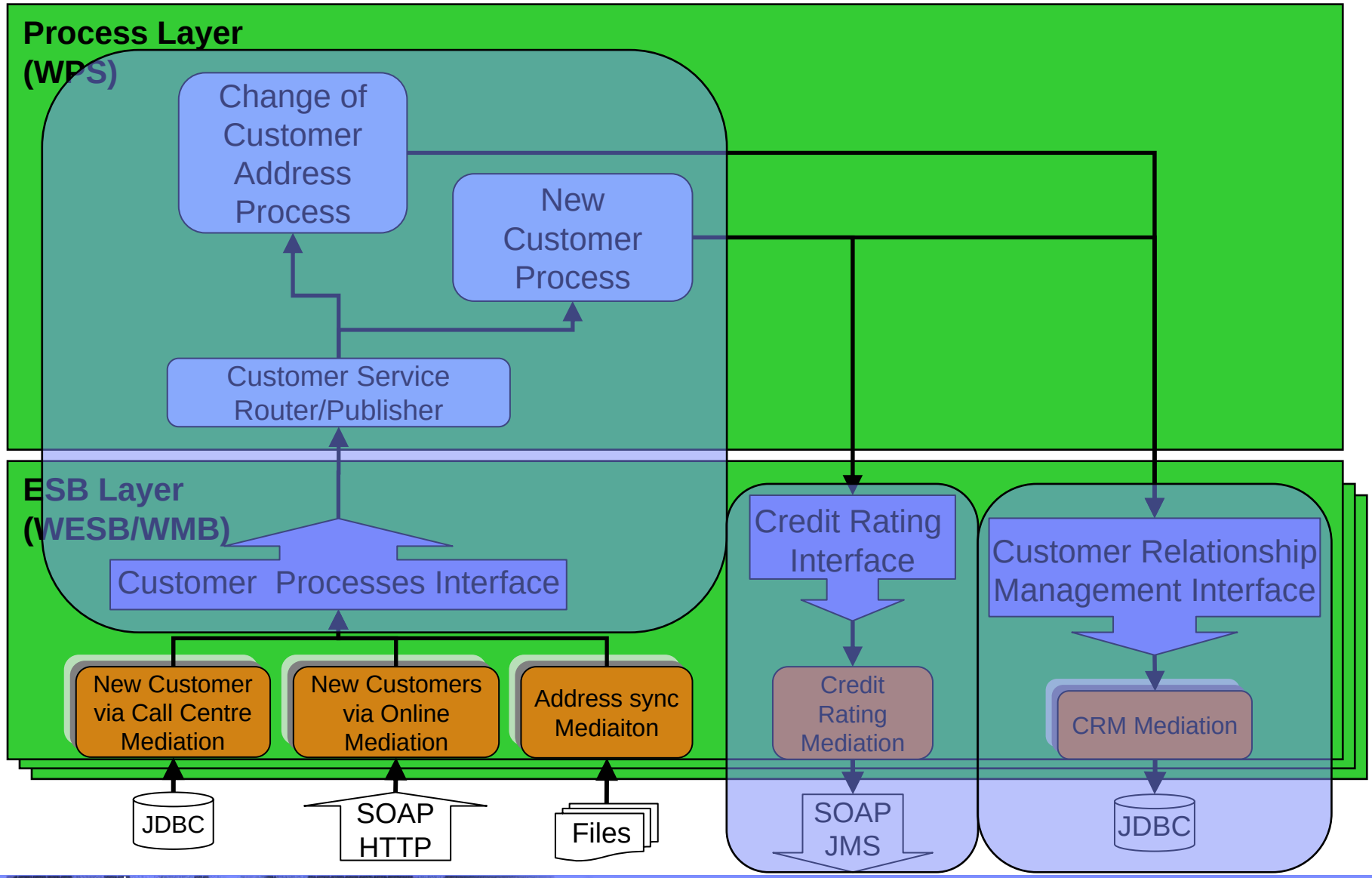
Loose coupling aspects of service interactions



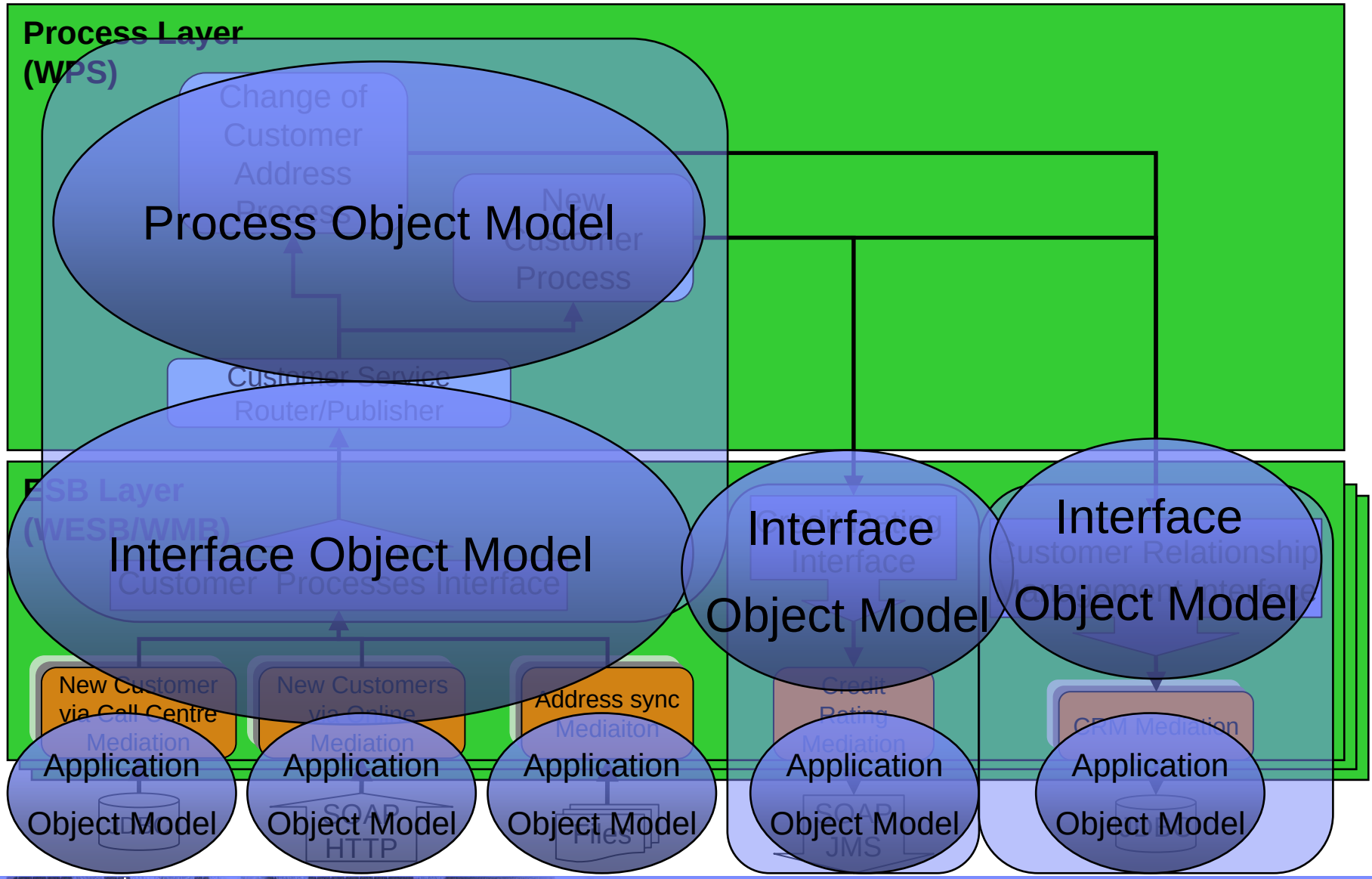
Layer de-coupling



Service Boundaries



Full Object Model De-coupling



Graphical User Interfaces

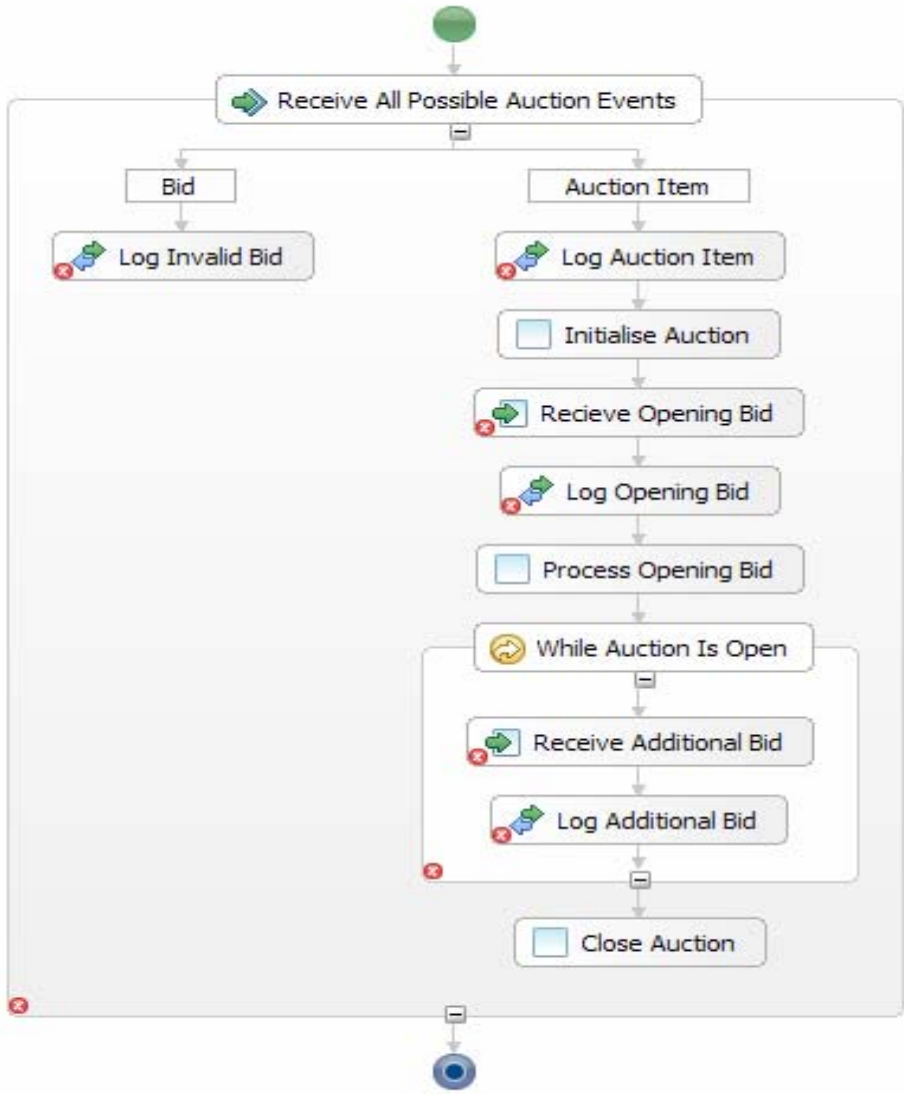
- **What type of GUI is required**
  - BPC Explorer HT
  - JSF
  - Portlets
  - JSP
- **Decoupling the GUI from the process**
  - Façade/proxy to BP/HTM
  - BP/HTM API exposed via web services
  - Page flow, generally should live in the presentation tier

Exception Handling

- **Fault Handlers**
- **Compensation**
- **Failed Event Management**
- **Event Handlers for termination**
- **Correlation Exceptions**
- **Housekeeping of failed processes**
- **Housekeeping of related Human Tasks**
- **Process Administrators**
- **Transaction scope**

Correlation – Avoiding Exceptions

- **Catch all possible events/operations in the initialising receive.**
- **Ensures that no correlation failures will occur avoiding unnecessary exception handling in the server.**
- **Searching for existing processes is possible but requires extra code and may hint at poor design**



Overview

- **Example usage scenarios for WPS**
- **Common experiences from SOA/WPS projects**
- **Features introduced in WPS 6.0.2**
- **Design work products**
- **Design tips**

- **Questions, anytime during the presentation**

Questions?