Enterprise –

Business Process Management/
Business Performance Management:
Architecture, Technology, Standards

Donald F. Ferguson
IBM Fellow, SWG Chief Architect,
dff@us.ibm.com
Agenda

- Introduction
  - Disclaimer
  - A sample business problem
  - Context and major trends

- Business Process/Performance Management
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- Summary, Discussion and Challenges, and “A Grand Challenge!”
Agenda

- **Introduction**
  - Disclaimer
  - A sample business problem
  - Context and major trends

- **Business Process/Performance Management**
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- **Summary, Discussion and Challenges, and “A Grand Challenge!”**
A Sample Business Problem – HW PO/Supply Chain

Manual Entry and Tracking

Customer POs for PCs, SAN, …

Customer System

Manufacturing

Warehouse

Assembly

Warehouse/Staging

Shipping
Example: Complexity is Forcing Change

Actual Application Architecture for Consumer Electronics Company
Some Challenges and Trends

- Challenges
  - Seams between
    - “people activities,” “information integrated” and “automated SOA activities”
    - Choice may change in a solution over time
  - Reusable solution templates with customization/configuration.
  - Policy and rules
  - Information –
    - Processes are a mix of “documents,” “people” and choreography.
    - Rich information model – PO, Customer, … …
  - Legacy integration
  - Federated/decentralized control and goals

- Some trends
  - SOA and Web services, obviously.
  - Improved but fragmented formal modeling standards
  - Coming together of IT processes and business processes (MUWS)
  - Coherent models for EDA and {BPM, EAI}
  - Domain standards
  - Governance.
Business Process and Performance Management: Model – Assemble – Deploy – Manage/Monitor

- Discover
- Construct & Test
- Compose

- Integrate people
- Integrate processes
- Manage and integrate information

- Gather requirements
- Model & Simulate
- Design
- A “better napkin”

- Financial transparency
- Business/IT alignment
- Process control
- Charge back
- Who saw what and did what for whom?

- Manage applications & services
- Manage identity & compliance
- Monitor business metrics
Agenda

- Introduction
  - Disclaimer
  - A sample business problem
  - Context and major trends

- Business Process/Performance Management
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- Summary, Discussion and Challenges, and “A Grand Challenge!”
Model

- BPM is more than process
  - Business Objects, Content Model (e.g., scans)
  - Business and object state
  - Organization (people)
  - Interaction
  - Policy
  - Key Performance Indicators/Observation, Events
  - Business Vocabulary

- Standards matter
  - Runtime interoperability
  - Federated tools
  - Portability
  - Monitoring and reporting
  - Evolution, Substitutability

- Currently a bit of a mess
  - UML
  - E-R
  - WSDL, BPEL, …
  - BPMN
  - SBVR

Model in “business user” terms and concepts, not XSD or com.payroll. …

Standards are more than runtime protocols

Design Time
- What does company A’s tools give to B’s?
- How does the caller know invocation seqs?
- What CA’s does the service support?

Portability
- Flexible placement over disparate product choices
- Decouple BPM from infrastructure evolution

Real programmers “extreme” program
Information as a Service – An Example of Going Beyond Process

Information as a Service
(Information Virtualization)

Standards based: e.g., XQuery, JSR170, JDBC, Web Services...

Data & Content

Master Data

Insightful Relationships

Master Data, Entity Analytics, Decision Portals, Exec Dashboards, Industry Data Models

Extracted or Real-Time

Heterogeneous Applications & Information

SAP

External Supplier / Business Partner

Point Content Management Application

Internally Developed

PeopleSoft

DB2

abc...

IBM Content Manager

xyz...

Oracle

and more...

Information as a Service – An Example of Going Beyond Process
Why Use Information Services in SOA?

- Efficiency
- Sophisticated clean/link/… functions
- Simplifies process design
- Reuse in other processes, as well as applications, portal, … …
Agenda

- Introduction
  - Disclaimer
  - A sample business problem
  - Context and major trends

- Business Process/Performance Management
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- Summary, Discussion and Challenges, and “A Grand Challenge!”
Service Components

“Web services describes the outsides. How do you implement a service? How do you compose services?”

- Encapsulate Components for Reuse; All look the same from outside
- Components may be wired together and aggregated via flow
- Business Objects are the data flowing on wires between Components
- Enable type, role and skills specific tools.

![Diagram of Service Components](image)
Web services describes the outsides. How do you implement a service? How do you compose services?

- Encapsulate Components for Reuse; All look the same from outside
- Components may be wired together and aggregated via flow
- Business Objects are the data flowing on wires between Components
- Enable type, role and skills specific tools.

### Interface:
- How to call this component

### Reference:
- What this component calls

#### Java Interface

#### WSDL Port Type

#### Business Process
- Business State Machine
- Business Rule
- Human Task
- Interface Map
- Decision Table
A Simple Example and Some Concepts

Something a DB dude recognizes

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

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

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

Deployment Tools

Author Tools
or
Text Editor
Assembling Services – Modules and Subsystems

UDDI, Service Registry
Project, SCCS
ad hoc exchange,

May “drill into” a specific component’s details

Simple wiring metaphor and tools.
Unaware of “how” the component works.

If Approved then
Send letter offering gold
If NOT Approved
Send letter offering Credit counseling service
Services Oriented Architecture and Mediations

Commerce Service

Credit Card Service

Needs a CC Service

Found a CC Service

But it’s not quite right. Darn

Transform
Route
Augment
Side Effect
Policy Selection
Matching

Mediation
Wiring is a simple model but is limited.

Transparent Mediation

Emit Event

Topics

Filter

Analyze

Rules

Subscription

Filter
Customizing Services – A “Design Pattern”

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

**Required “Services”**
- document POVs

**Prolog**
- Types
- Messages
- Port Types
- Operations
- Bindings

**Mediate**

**Impl**

**Web Service Binding (Find)**

POV = Impl. Model for calling a service, e.g. JAX RPC Stub, BPEL Invoke

**Control Descriptors**

**Deployment Descriptors**

**Simple rule templates**
- Java, JavaScript
- Rule Engines
- Decision Table
- Decision Tree
  - … …

**Just another “service”**

**Just another “service”**
Services and Components

- There is a set of standards (emerging) for formally representing a service component’s behavior
  - WSDL, XSD
  - BPEL, UML
  - WS-Policy
  - WS-ResourceFramework

- There is an emerging, extensible set of service component kinds that provide a natural mapping for model elements
  - Process, Business State Machine, Selector
  - *Eliminates the “miracle happens here” model*
  - High level, portable implementations emerging

- Support for very dynamic
  - Structural composition
  - Behavioral composition
  - Configuration/customization

- Bridge to existing skill sets.

This “BPM” solution is a set of documents enabling a “wiki like” BPM evolution.
Some Perspective

- Haven’t we heard this before? OO, RPC, MDP, … …?
  - This time we really mean it.
  - Can’t you take a joke?
- There are some differences
  - XML is language neutral;
    previous approaches implied a language model.
  - WSDL and XML are more forgiving of changes
  - Supports RPC and message/document approaches from beginning
  - Common type model for applications/servers, message systems and DBs
  - Builds on Internet protocols already deployed for “Web browsing.”
  - Uniform model for events/pub-sub, message routing and RPC
  - More focus on logical behavior – WSDL, Policy, etc.
  - SOA component model derives from business modeling, making MDD simpler and eliminating “spooky transformations!”
Agenda

- **Introduction**
  - Disclaimer
  - A sample business problem
  - Context and major trends

- **Business Process/Performance Management**
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- **Summary, Discussion and Challenges, and “A Grand Challenge!”**
A Patch Management – Implementing the Process

Questions:
- How best to implement the process and make it operational?
- What tools should be used for sequencing work between people?
- What tools for automating particular activities in the process?

Need to have:
- Logical Process implemented in executable workflows
- User Interfaces for process steps
- Adapters to Management Apps
- Documentation to customize steps

SOA Infrastructure

(Business) Process Engine

Change Mgmt

Release Mgmt

Open RFC
Categorize RFC
Assess Impacts
Review & Approve Change
Deploy Change
Close RFC

Package the change
Test the change
Document Impacts
Open other RFCs
Apply Patch
Verify Patch

Select Servers
Select Subset
Select Change Window
Schedule Change
Deploy Patch

Check Results (verify) Change
Update Config DB

Done?

Rollback
Manual Patching

Infrastructure Mgmt tools automate some process steps

Human review and approval steps

Infrastructure Mgmt
Building IT Process Flows

- Model
- Assemble
- Deploy

- Provide portal console and run process using Process Choreographer (WPC)

Human review and approval steps

Infrastructure Mgmt tools automate process steps
The Role of the Container

Security Header
Reliable Messaging Header
Atomic Transaction Header

SOAP Message

This is fragile,
changes over time,
complex for business programmers,
error prone,
etc.

Policy Declarations

double deposit(Message m) {
    checkForDuplicate(m.seqNo);
    registerForTransaction(m.context);
    isCAValid(m);
    checkSignature(m);
    updatePerformanceInfo();

    balance += m.amount;

    // … …
    updatePerformanceInfo();
}
The Role of the Container

Security Header
Reliable Messaging Header
Atomic Transaction Header

SOAP Message

Wrapper
The Impl.

Container
Security
Reliability
Transactions

Before
After

Check Certificate
Challenge
etc.

Ack.
Retransmit

Container is a set of policy driven functions.
Interceptor pattern for business logic and “stubs.”
Before and After factoring of code.
4 Principles Of Application Management Confidence

A repeatable approach to sense and respond to performance problems within the composite application infrastructure.

Measure Response Time
Trace Transactions
Mediate Services & Enforce Policies
Monitor & Adjust Resources
A Typical Management/Monitor Portal

- **Topology View:** aggregate interactions among services.
- **Sequence Diagram:** Shows exact sequence of messages over time
- **Statistics View:** A table view of the raw data collected by the monitoring agent at each interception point
- **Content View:** Shows content of a SOAP message
Business Process/Performance Manage/Monitor

- **SOA infrastructure monitoring**
  - Dynamic and policy driven
  - “Transparent” to application logic
  - Sense – Respond – Act via MUWS

- **Observation/Performance Modeling**
  - Observation points are part of modeling
  - Define scorecard view of Key Performance Indicators

- **SOA event infrastructure and event database**

- **Enables**
  - Dashboard, monitoring, management.
  - Ability to intervene in deployed processes
    - Set situational triggers and notifications
    - Dynamically respond to these alerts
  - Supporting continuous process improvement
    - Monitor in-flight business processes
    - Make process modifications based upon real-time data sent back to the Modeler for simulations
Deploy, Manage and Monitor

- System and application management is a *business process*
  - Complex, multi-step process with compensation for change management
    - Patches, OS/middleware upgrades, application enhancements
    - Automated steps, manual approval
    - Compensation, recovery, retry
  - Common Management (Information) Database
  - Events, Sense/Respond, Monitor, Act, History, reports

- Using “standard” BPM technology
  - Enables existing skills (e.g. no arcane SM language and APIs)
  - First class tool support

- The line between systems/application management and BPM is an illusion
  - Why are PO submissions failing? IT error or process design?
  - Impact analysis
  - Think about “customer on-boarding.”
    - Some updates to CRM, account system, etc.
    - Also calls to security for UID, ACLs, … …
Agenda

- **Introduction**
  - Disclaimer
  - A sample business problem
  - Context and major trends

- **Business Process/Performance Management**
  - Model, and why more than “process.”
  - Assemble and Customize; EDA and SOA
  - Deploy
  - Manage/Monitor

- **Summary, Discussion and Challenges, and “A Grand Challenge!”**
Summary and Directions

- **Business Process/Performance Modeling/Management**
  - Is a broad space with many “sub-models.”
  - We are only beginning to
    - Bring them into a coherent whole
    - Without overwhelming business professionals and IT professionals
  - We must drive standards and convergence. Standards are broader than runtime formats and protocols.
    - Tool federation
    - Flexible function placement
    - Evolution

- **There are many open areas**
  - Governance
  - Web 2.0
  - Explosion of casual programmers in the workplace
  - Modeling support for *Recipes, Patterns and Templates*
  - Service/component identification, factoring, good “size,” … …
A Grand Challenge

- There is good and improving models for Process, Information and State,
  - Events
  - Components
  - Use cases
  - Component collaboration
  - … …

- Policy is where we are the most “broken”
  - Examples
    - “All POs over $10,000 must be approved by regional sales manager.”
    - “An employee cannot close a customer complaint that he created or marked complete.”
    - “Business class is authorized for flight more than 8 hours or overnight.”
  - We typically write these down in text.
  - Programmers read the text and write code.

- Can we do better?
  - Many domains have nascent business vocabularies, e.g. law, dentistry
  - We write specs with nascent “grammars,” e.g. MUST, CANNOT, RECOMMEND
  - Documents are often simple combinations of policy
    - Nested lists (Decision Trees)
    - Tables/Forms (Decision Tables)
    - Hyperlinks (Decision Flows)
  - Can we improve hand-offs and traceability for rules and codes through structured language, business vocabularies
Some Clarification

- An Asset is, well an Asset. Can be anything
  - Word document, Powerpoint Presentation
  - Handy code that I keep lying around
  - Excel spreadsheet for costing a project
  - ……

- A Pattern is a recurring solution for solving a problem
  - Patterns for eBusiness (http://www-106.ibm.com/developerworks/patterns/)
  - Enterprise Integration Patterns (http://www.eai-patterns.com/)
  - J2EE Patterns (http://corej2eepatterns.com/index.htm)
  - ……
  
  Read the book and start typing!

- 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 a directed graph of Templates that direct a user to
  - Which arcs to follow
  - Metadata flows through the graph
  - 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

We should be more helpful than this

This is my terminology. We are trying to come to a simple, common terminology.
Pattern Authoring in Rational Software Architect

1. Create Plugin Project with Patterns Templates

2. Create the skeleton of the pattern. Define parameters for the pattern and constraints on the parameters

3. Add your custom code using the RSA Pattern Framework. It provides many extension points

4. Test pattern by launching a new runtime workbench

5. Export the pattern as a RAS asset. Classify your pattern to facilitate searching

6. Publish Asset to a RAS repository
Example of applying a Pattern

1. Locate and Import the Asset from a RAS repository

2. Select the Pattern from Pattern Explorer

3. Specify Pattern Parameters

4. Run a Transformation to convert models into Artifacts (code, scripts, docs)
Some Principles

- Minimize concepts and rely on *patterns*
  - Pub-sub is a SOA pattern
  - Mediation is a *style* of service
  - Business rules are *a way* to implement services … …

- There are very few *new things*
  - Focus less on code reuse and more on pattern reuse
  - There is no difference between pattern, code, developer tool
  - Basic building blocks (e.g. SCA) enable flexible patterns

- Benefits
  - Productivity
  - *Reduce risk, more predictable projects, technical community, …*

- There are two tiers of programmer
  - Architecture/Pattern/Template provider
  - Template/Pattern user
    - Instantiate
    - Compose
    - Configure

- Code for customization
To Do

- Governance
- ITSM
- Haven’t we heard this before?
- Web 2.0/ESRI
- Service Identification and Factoring
- Component Business Model and abstract models
- Everyone is a “casual programmer”
- Patterns