WebSphere Expert Call Series

WebSphere BPM Error Handling and Solution Recovery

Bryan D.Brown
Advisory Engineer
IBM WebSphere Process Integration - SWAT

22 September 2010

An Exclusive series presented by the IBM Software Accelerated Value Program

Objectives

- Answer some questions about Error Handling Strategy
- Error handling strategy stages
  - Recognize
  - Record
  - React

Murphy's Law

- Things that can go wrong will go wrong
- Consider a typical SOA solution that interacts with internal and external services, typical errors that happens at all levels of the system but often missed during solution design
  - Services will become unavailable
    - Network failure
    - Regular off-line maintenance
  - Bad data
  - Application Errors - Divided-by-zero
  - Request appears 'hung'
- Need to “engineer for failures”!

Why do I need an Error Handling Strategy?

- A strategy is needed to standardize development practices.
- The lack of standardization leads to instability and unnecessary gaps in transactional integrity.
  - This is especially common when production projects grow from proof of technology pilots.
- To promote consistent and successful recovery of business transactions and data

What is an Error Handling Strategy?

- An error handling strategy is developed by understanding the business needs and applying patterns of development to manage volatility.
- By using these patterns, integration developers are able to quickly assemble consistent applications that provide and consume services from common end points.
- The strategy establishes a recovery framework and standards enforce the usage of the framework.

Who develops the Error Handling Strategy?

- SOA Architects are responsible for defining a strategy for service interactions
  - Meet nonfunctional requirements
  - Promote system recovery and data integrity.
- Service architects/developers should define service error handling mechanism to conform with the strategy

When is an Error Handling Strategy Developed?

- Error handling and prevention is the most often overlooked non-functional requirement.
- These strategies must be developed in the design phase of the system development life cycle
- Position to leverage integration patterns that promote recovery and problem determination.

Where do I begin?

- Define and record the objectives and service level agreements for error handling and recovery.
- Gather requirements from the business, IT and operations teams regarding:
  - How the system should behave?
  - What data requires persistence?
- The creation of the error handling strategy will feed into the development of Operations Manuals and recovery procedures.
  - Assertion – we can document procedures if a solution is developed to behave consistently

Three Objectives of Error Handling

- There are many ways to approach an error handling strategy
- However, there are always 3 main objectives
  - Recognize
  - Record
  - React

Recognition

- Recognizing that there is a problem is the first line of defense.
- Applications should be structured in such a way that exceptions are properly classified.
- A classification strategy is critical to subsequent actions.
  - We have to understand what condition we are dealing with first.
- There are 2 types of Exceptions
  - Declared – withdrawal exceeds balance
  - Undeclared – connection not available

Recognition

- There are two classes of error detection for service development
  - Internal sanity checks/failures
    - An request user not in my database
  - External service failures
    - Services that you interact with are not doing its jobs
- A well design service needs to consider both classes to properly handle errors

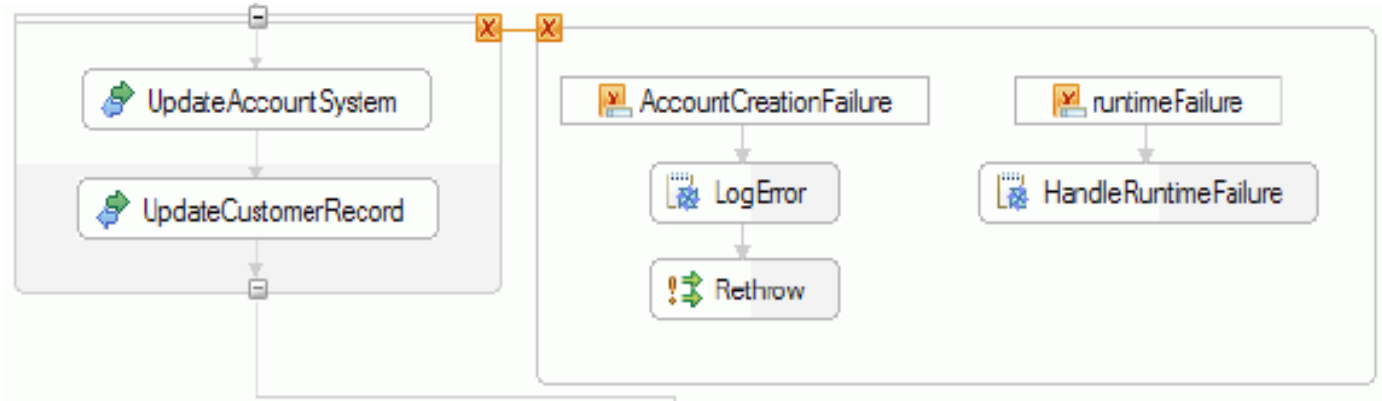
Recognition

- After determining which types of errors may be expected to occur (and defining how these will be represented), there are a variety of tools and techniques available to developers for identifying errors and classifying them within their application.

- An error handling strategy should define which of these methods to use and when to use them, based on the application requirements, design and architecture.

Techniques for Recognizing Exceptions

- Fault Handlers in BPEL
  - When using BPEL to choreograph service invocations, we use fault handlers to define how a business process should behave when specified faults occur.
  - Fault handlers may be configured to catch faults defined on the process component's reference interface(s), or one of the built-in faults.

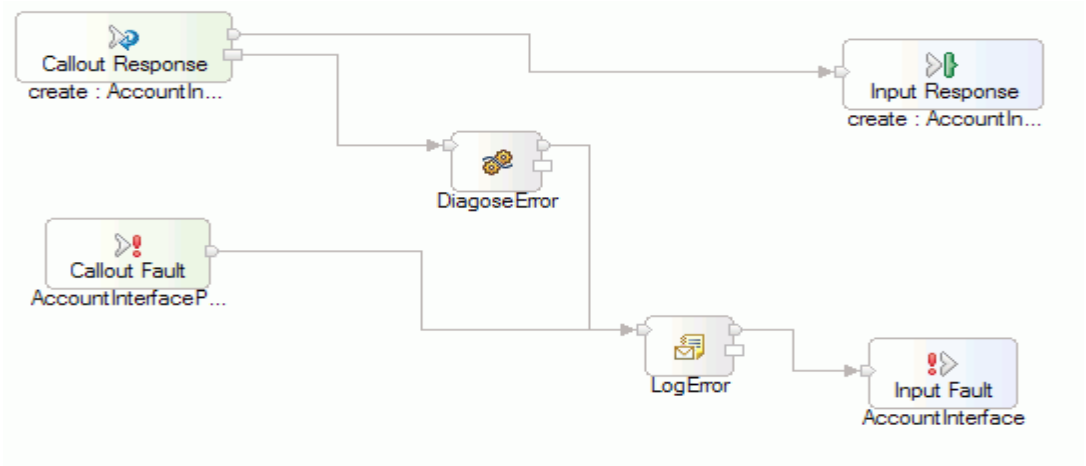


Techniques for Recognizing Exceptions

- Fault Handlers in BPEL
  - WebSphere Process Server's Process Engine defines a number of built-in faults.
  - These faults should be understood and used consistently across the development effort.
  - Additionally, there are a number of samples and examples available for how to set up and define a fault handling routine.
  - Advanced BPEL features> Fault Handling: http://publib.boulder.ibm.com/bpcsamp/index.html

Techniques for Recognizing Exceptions

- WESB – Failure nodes and unmodeled faults
  - The mediation tooling in WebSphere Enterprise Service Bus (WESB) provides features for handling both declared and undeclared faults.
  - To define how your mediation behaves when this fault occurs, simply create a wire from the callout fault to the appropriate node in your flow.



Techniques for Recognizing Exceptions

- Using Java
  - Java components use the same try catch keywords provided by the Java runtime.
  - WebSphere Process Server's programming model provides an implementation of declared and undeclared exceptions.
    - ServiceRuntimeExceptions- Undeclared and unexpected error types. Typically, these are created only by the runtime.
    - ServiceBusinessExceptions - declared and expected error types. These are defined on the service interface and programmatically created.

Record

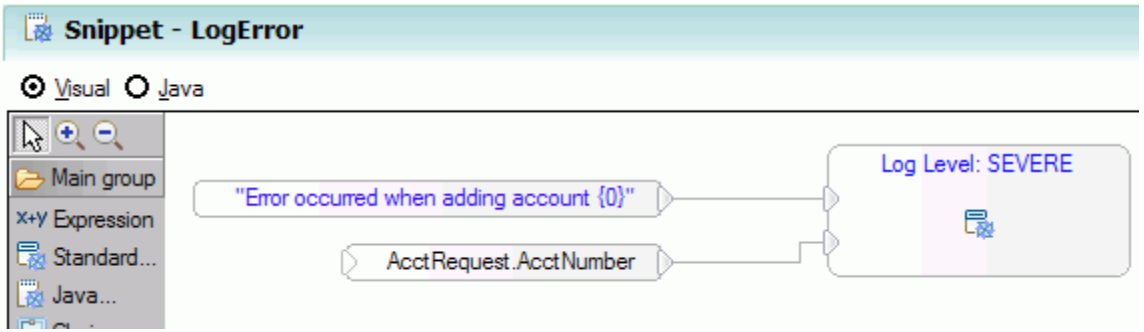
- After the appropriate recognition and classification of a problem or exceptions, it is critical to record data about the state of the system for future recovery.
- This data is essential to the problem determination effort that will follow in the React phase of the strategy.
- Our error handling strategy should consider the differences between recording business and IT events.
- We should also understand the audience of the error recording: business user, IT administrator, or IT developer
- The mechanism that we will use to record the incident will likely be different.

Recording the Exception

- IT events are recorded to a log and/or emitted for IT monitoring products like Tivoli.
- Since debugging and other problem determination activities, may required a precise and detailed record of what has happened, the volume of detail we chose to record are better suited for logs and collectors.
- The project should set up a common logging framework and standard.
- The logging frameworks can be implemented using configurable tools such as java.util.Logging (JSR47 logging).

Recording the Exception

- The standard defined by your team should define the frequency and depth of each log entry.
- There are many things to consider:
  - Where are trace entries standard?
    - Method entry and exits?
  - How do we standardize the usage of message severity?



Recording the Exception

- Business Events can be emitted via in a WebSphere environment using the Common Event Infrastructure.
- It is important to establish a standard for when/where/why these business events are emitted.
- Emitting too much or too little will have different but adverse effects.
  - If the entire event digest is included in each event and a given event is logged a number of times, the additional load on the system would be significant enough to consider the scenario during load and capacity planning.
  - If events are not emitted enough, then the value of the data will suffer.

React

- The general principles are
  - First make sure that we are in a consistent state
  - Then try to either report the error/exception back or trigger corrective actions
- In general there are 3 types of corrective actions
  - Automated Recovery
  - Trigger manual corrective procedures (by a person such as an administrator)
  - Switch to an alternate processing path

Don't Panic!

- Have a plan and execute!

React - Automated

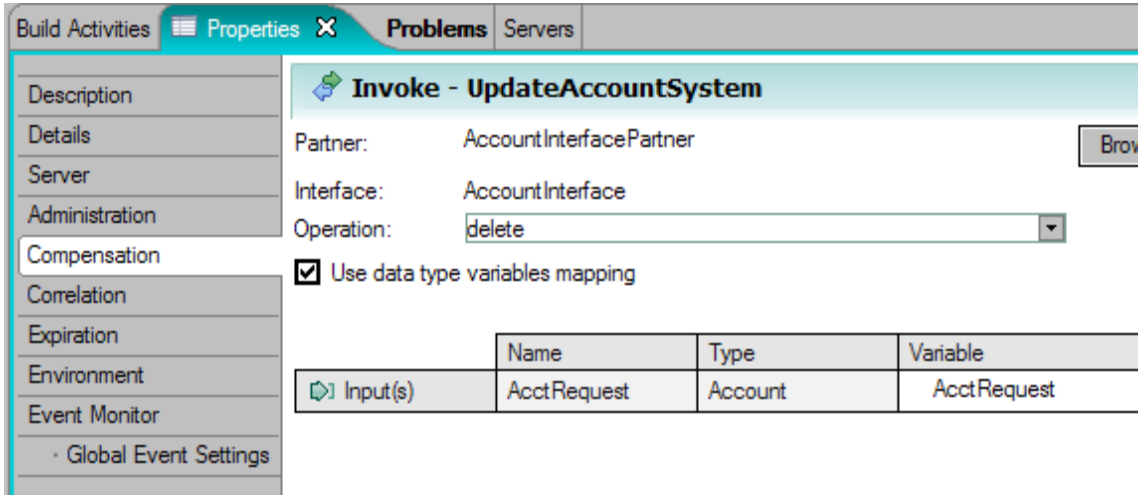
- Automated recovery within the “React Phase” of the error handling strategy implies that where possible, the application should automatically ensure that the system stays in a consistent and predictable state.

- Transaction scopes can be configured and used to maintain the consistent state of resources during the course of an unexpected condition. XA Transactions may be used to create scopes around capable resources.

React - Automated

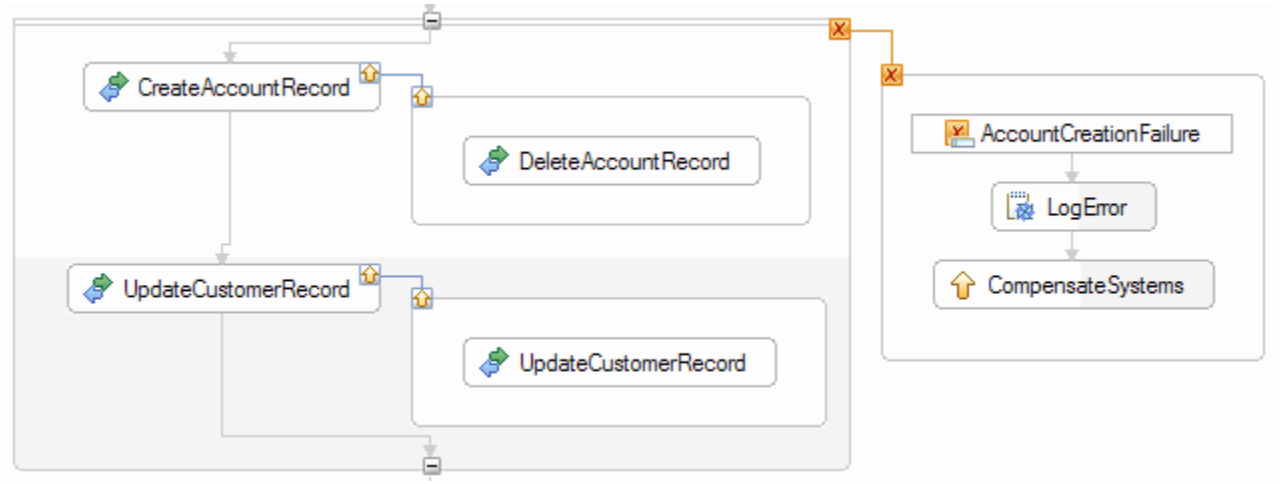
- Compensation of microflows and long-running processes can be used to “undo” the outcome of service invocations that have already completed.

- It is used when choreographing non-transactional services.



React - Automated

- In long-running processes, compensation of activities that have successfully executed is initially triggered by a fault raised in the process, or can be explicitly triggered using a compensation activity. This is a useful technique for reversing the effects of already-committed transactions within a long-running process.

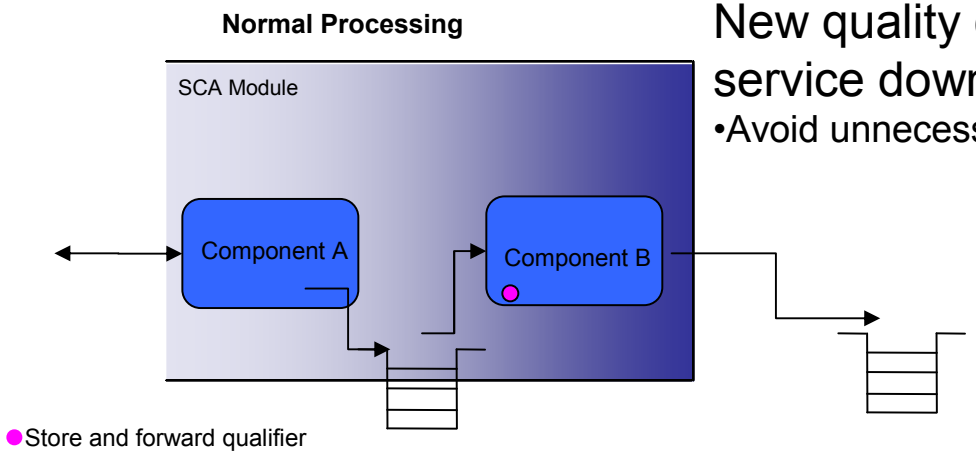


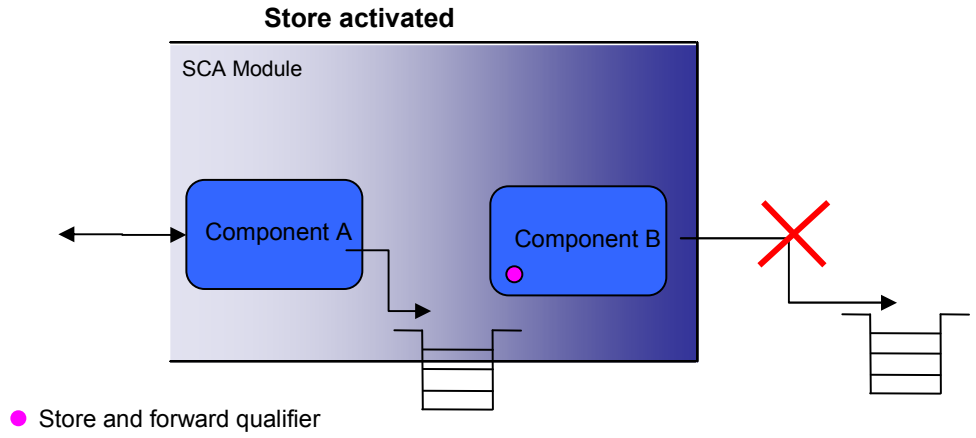
React – Automated

- Automatic retries can be used to increase the resilience of the application to intermittent errors.

- WebSphere Process Server and Enterprise Service Bus provide a retry capability automatically under some circumstances; such as when making asynchronous invocations.
  - Service Integration Bus (SIB) retries on asynchronous invocations
  - BPE retries
  - Configurable retries in mediation flows

React: Store and Forward





New quality of service for handling unexpected service down scenario

•Avoid unnecessary failed events

- Automatically detect service down based on runtime exception specifications
- Modeled in WID as qualifier on interface/operation
- Works on all imports and asynchronous hops
- Failed event manager shows events that activates store-forward
- Admin widgets that can manage store-forward points

React - Manual

- There will always be unforeseen errors, or errors that the system cannot handle automatically.
- In these cases, the application developer may choose to have the application “hold” and wait, to allow an administrator (or other human) to manually diagnose and correct the error.

React - Manual

- BPEL – Continue on Error
- On BPEL activities, a developer may set the “continue on error” property to “false”.
- If the activity fails at runtime, the Business Process engine will transition the activity to the “stopped” state.
- The Business Process Choreographer Explorer will show the process with the stopped activity under the Critical Processes view
- Stopped activities may be resubmitted or terminated by the process administrator.

- Note: For long running process instances only

React - Manual

- Administer Failed Events

- Failed events are created automatically by the WebSphere Process Server product. They will be stored in the system until they are dispositioned by an administrator or programmatically by the Failed Event Manager APIs.

- Failed Event APIs are documented in the web/ directory of your WPS test server installation: <WID-install-dir>/runtimes/bi_v62/web/mbeanDocs/FailedEventManagerMBean.html

React - Manual

- In the case of manual intervention, administrators or the staff responsible for the manual activity would need to be notified.
- The error handling strategy needs to document how the notifications would be sent, how frequently, who is responsible (role) for acting upon the event.
- The notification mechanism will likely differ based on whether the error condition is an IT level error or a business related problem.

React – Business level errors

- Sometimes, the business process definition accounts for error conditions. These are truly business-level errors, rather than IT level errors, and should be handled accordingly.

- The logic for handling these error conditions is part of the modeled business process flow, rather than one of the techniques described previously.

- For example, in a human-centric workflow, all exception cases may be routed to a senior employee for processing. Or, in a business process defined using BPEL, you can take advantage of the "Case" and "Otherwise" clauses of the "Choice" construct to model If-Then-Else conditions.

Evaluate and establish a system framework

- The service level agreements will have a direct impact on the types of error handling frameworks that are required.
- After we have established the requirements for the framework, we can then begin to evaluate the provided product capabilities that we can leverage for error handling and recovery.
- You can use and combine each of the following product capabilities to establish a framework within the strategy:
  - Business processes
  - Human tasks
  - Failed Event Manager
  - Common Event Infrastructure
  - Business Activity Monitoring

Evaluate services and service providers

- The selected framework could be affected by characteristics of the service such as:
  - Batch windows and system availability
  - Lack of transaction support
  - Message tolerance/Assured Delivery: At least once, once and only once
  - Reliability of the protocol

Evaluate services and service providers

- Evaluate each endpoint by a number of characteristics. Then the architecture team can develop common patterns and templates for dealing with the common sets of issues.
  - Operation type
  - Binding
  - Invocation pattern
  - Availability
  - Security
  - Performance and throughput

Examples

- At this point with time permitting we will review some applications in the BPM tooling.
- We will walk through some good examples and bad examples of an error handling strategy and have an open discussion about the ramifications of the implementation and administration of the solution

What is wrong with this picture?

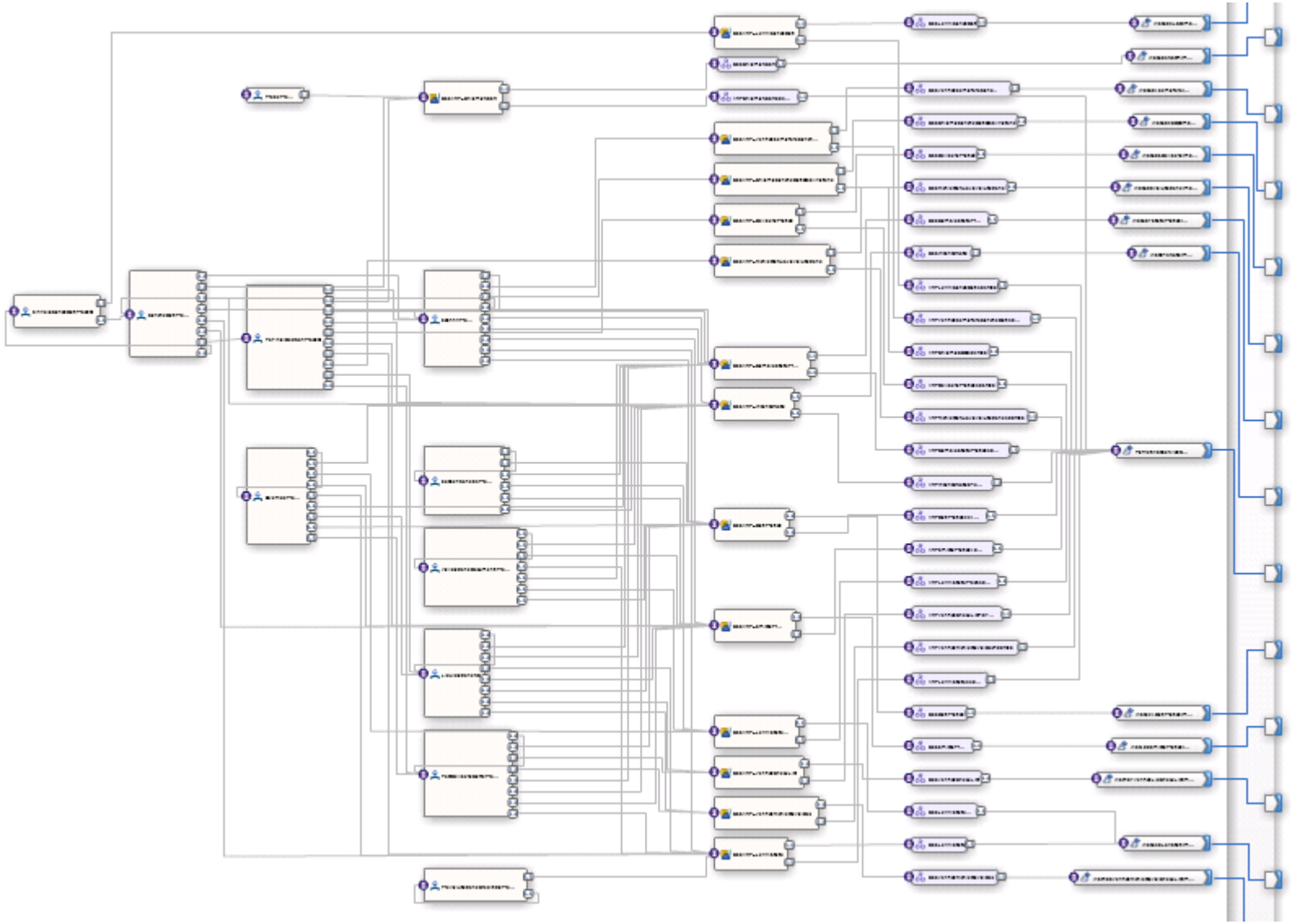
PI Created with tooling version(s): 6.1.x

- 369 Exports
  - 369 Web Service (JAX-RPC based)
  - 4 Web Service (JAX-WS based)
- 609 WSDLs
  - 561 use web services
  - 566 total operations
  - 0 total faults
- 41 SCA Modules
  - 0 mediation modules
  - 41 business modules
- 1 WPS Libraries
- 41 Components
- 28 XSDs
  - 763 BOs
  - 7522 fields
- 286 Imports
  - 286 Web Service (JAX-RPC based)
- 42 Business Process
  - 42 long-running
  - 0 with compensation
  - 916 total activities
  - 0 Java snippets
  - 1 Inline human tasks

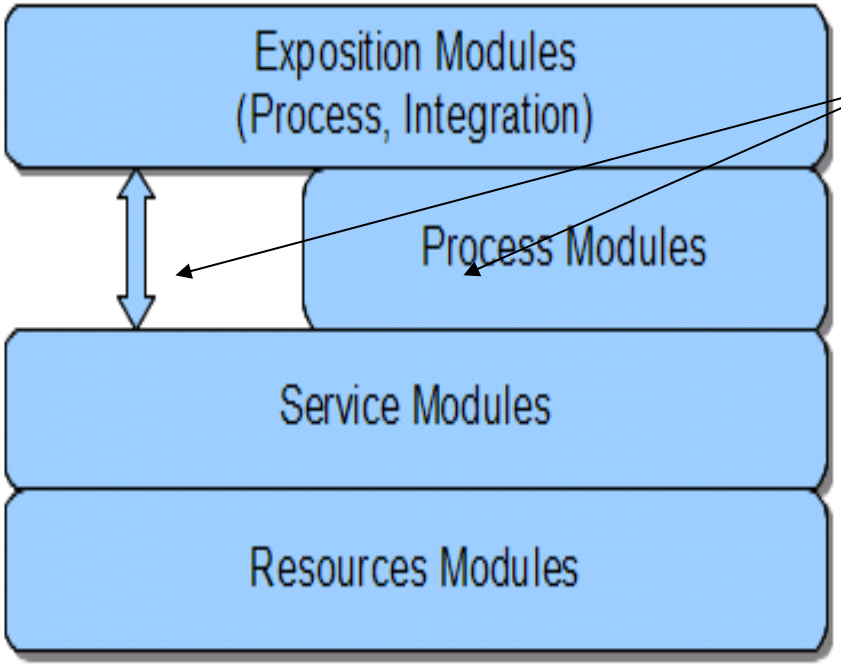
Statistics

On average each WSDL contains 1.0 operation(s).
On average there are 0.0 faults/WSDL.
On average there are 0.0 faults/operation.
On average each XSD contains 268.64285 field(s).
On average there are 27.25 BOs/XSD.
On average there are 9.858454 fields/BO.
On average there are 21.809525 activities/BPEL.

What is wrong with this picture?



How should I structure my solution?



Failed Event and Store and Forward

Fault Handling and Human Intervention

Unit of Work (Compensation & XA)

Retry and Time Management

- Structure the solution in layers
- Isolating and normalizing at each layer
- Stateful processes interact with purified services and Business Entities

Summary

- Define an error handling strategy during the design phase
- Remember the three “R”s
- Enforce the use of the error handling strategy during the implementation by executing peer reviews and proper governance

References

- ***Product Information Centers & Samples:***
- BPM Samples: http://publib.boulder.ibm.com/bpcsamp/index.html
- Dealing with faults in your business process, http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/topic/com.ibm.wbit.620.help.bpel.ui.doc/concepts/cdealfault.html
- Failed event manager, http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wbpmhelp.620.doc/com.ibm.ws.console.wbi.wbifailedeventmanager/index.html

- ***Whitepapers and developerWorks articles***:
- Exception handling in WebSphere Process Server and WebSphere Enterprise Service Bus, Jeff Brent and Pamela Fong, http://www.ibm.com/developerworks/websphere/library/techarticles/0705_fong/0705_fong.html
- Handling unmodeled faults within WPS V6.1, Marco Lezajic and Boris Feist, http://www.ibm.com/developerworks/websphere/library/techarticles/0802_lezajic/0802_lezajic.html
- Using compensation in business processes with Business Process Choreographer, Anke Robeller and Claudia Zentner, http://www.ibm.com/developerworks/websphere/library/techarticles/0604_robeller/0604_robeller.html
- JSR-000047 Logging API Specification, http://jcp.org/aboutJava/communityprocess/review/jsr047/index.html

Questions ?

Please send your feed back and follow up questions to gouni@us.ibm.com