Effective Fault Handling in SOA Suite 11g

Ronald van Luttikhuizen
Vennster
Utrecht, The Netherlands

Keywords:

Fault handling, fault prevention, faults, Oracle SOA Suite 11g, Oracle Service Bus 11g, BPEL, SOA, Web Services

Introduction

It is one thing to design and code the "happy flow" of your automated business processes and services. It is another thing to deal with situations you do not want (or expect) to occur in your processes and services. This session will dive into fault handling in Oracle Service Bus 11g and Oracle SOA Suite 11g using a case study based on an ordering business process. First the session investigates what can go wrong in automated processes and services. Then it categorizes these situations and dives into the mechanisms Oracle Service Bus and Oracle SOA Suite offer to handle these different scenarios. These mechanisms include Retry and Throttling capabilities of Oracle Service Bus, BPEL activities such as Throw and Catch activities, and SOA Suite's fault handling framework. The session will wrap up by introducing a generic fault handling framework for technical faults used in a real-life project that is realized using a Java fault handler and SOA Suite's fault handling framework.

What is fault handling?

A fault is something that is unusual and happens outside normal operational activity or the "happy flow" of a process. Examples are network failures causing messages to be lost, tablespaces that are full causing database failures, division by zero in a software component ordering the incorrect amount of goods in a web shop, and processing an invoice that has an incorrect invoice amount.

Fault handling should focus on prevention first since faults are time-consuming and expensive to deal with. It is better to prevent fault situations then having to deal with them. However, it is impossible to prevent all faults, effective fault handling measures should therefore be in place to deal with faults when they occur.

Faults in IT systems can be categorized into the following types:

- Technical errors. Faults caused by errors in the underlying infrastructure or middleware such as network errors, server failures, corrupt disks, full tablespaces, and so on. Technical errors can be prevented by creating robust and hardened infrastructure and middleware.
- Software errors. Faults caused by programming errors such as division by zero's, infinite loops, memory leaks, null pointer exceptions, and so on. Software errors can be prevents by applying software development methodologies and best-practices such as pair-programming and collegial review.
- Faulty operation by users. Faults caused by human errors that use IT systems such as entering a wrong credit card number, switching the to and from date when booking flight tickets, and

so on. Fault user operation can be prevented by applying good user experience techniques so that IT systems are well designed and understandable for end users.
- Exceptional business behaviour. Failure to meet a certain business criteria or rule such as a bad credit rating, an unknown customer that wants to purchase something and incorrect invoice amount.

This presentation focuses on technical and business faults in the context of a SOA environment.

Zooming in on Business Faults and Technical Faults

The two types of faults that are discussed in the context of SOA are business faults and technical faults. They differ in the following aspects:

Business faults have the following characteristics:

- Business faults are faults that service clients can expect and recover from;
- They indicate a failure to meet a particular business requirement;
- They are often expected, have business value, are contractual and recoverable.

An example of a business fault as returned by a Web Service and that indicates that a particular customer is not found in the backend system might look like the following:

```
<soap:Envelope>
  <soap:Header/>
  <soap:Body>
    <soap:Fault>
      <faultcode>CST-1234</faultcode>
      <faultstring>Customer not found</faultstring>
      <detail>
        <CustomerNotFoundFault>
          <CustName>John Doe</CustName>
          <City>Long Beach</City>
        </CustomerNotFoundFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

We might recover from this fault by registering the new customer in the backend CRM system and continue business as usual.

Technical faults have the following characteristics:

- Technical faults are faults that service clients do not expect and cannot (easily) recover from;
- They results of unexpected errors during runtime, e.g. null pointer errors, resources not available, and so on;
- They are often unexpected, technical, implementation and non-recoverable.

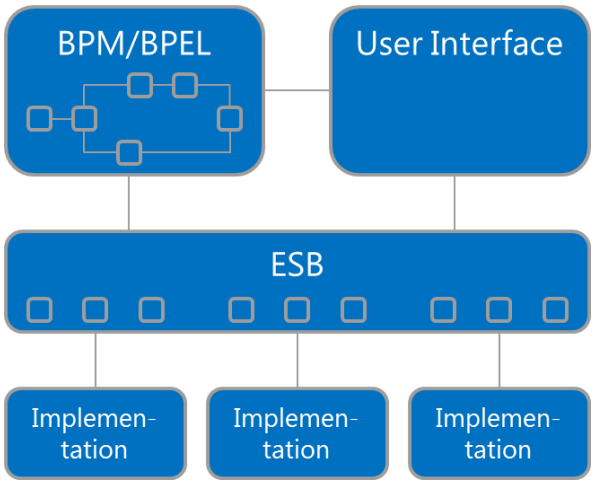
An example of a technical fault as returned by a Web Service and that indicates that a particular endpoint is unavailable might look like the following:

```
<soap:Envelope>
  <soap:Header/>
  <soap:Body>
    <soap:Fault>
      <faultcode>S:Server</faultcode>
      <faultstring>Could not connect to URL 127.0.0.1 on port
8001</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

We cannot easily recover from this error, but need to inform IT operations that manages the server we try to connect to so they can restore the server to a normal operational condition. We can handle the fault by for example retrying at a later time.

Fault Handling in SOA versus traditional systems

A typical SOA and BPM environment encompasses backend systems exposed by an Enterprise Services Bus, has a User Interface and BPM component that both consume services. Such an environment might look like the following:



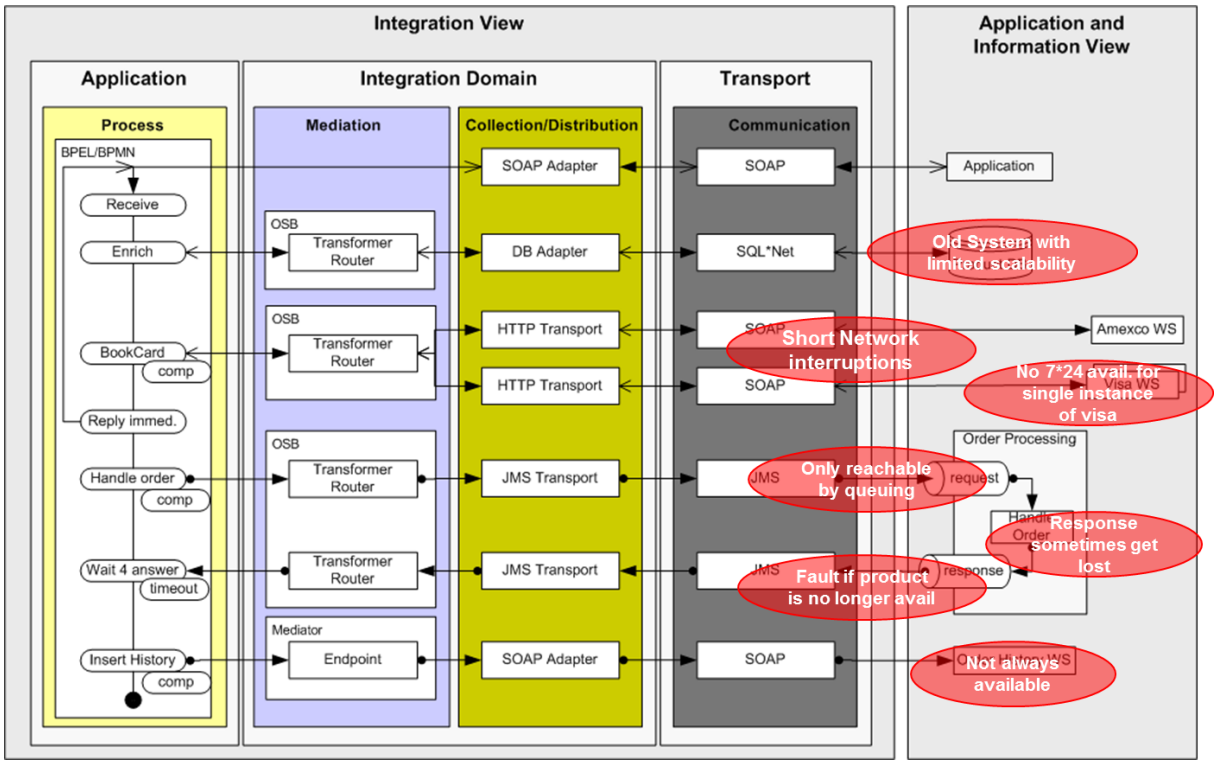
The following aspects of a SOA have impact on fault-handling:

- There are multiple service consumers, also external clients;
- Services are part of a larger unit of work, we might need fault handling in a larger scope than a service alone;
- A SOA contains of heterogeneous and external components making standards for fault handling important;
- A SOA and BPM environment contains long running processes that impact the way we handle faults;
- Services can be asynchronous;
- There are timed events that might, or might not occur;
- SOA and BPM efforts are often rolled-out often enterprise-wide.

Scenario

To explain the fault prevention and handling capabilities of Oracle Service Bus and Oracle SOA Suite a scenario is used. The following figure shows an ordering process that is implemented in a BPM and SOA environment. The left hand side shows the process steps, the middle shows the services that are invoked from the process, and the right hand side shows the backend systems that are accessed through these services. In this scenario Oracle Service Bus is used to provide the services and Oracle SOA Suite to implement the ordering process.

The red ovals represent faults that can occurs during the execution of the process such as limited scalability of old systems, lost messages, and so on.



The remainder of the presentation discusses how these faults can be prevented and handled using the capabilities of Oracle Service Bus and Oracle SOA Suite.

Oracle Service Bus 11g

The Oracle Service Bus provides several mechanisms to prevent faults from happening or to correct faults that have happened. The following list contains solutions that can be applied per problem:

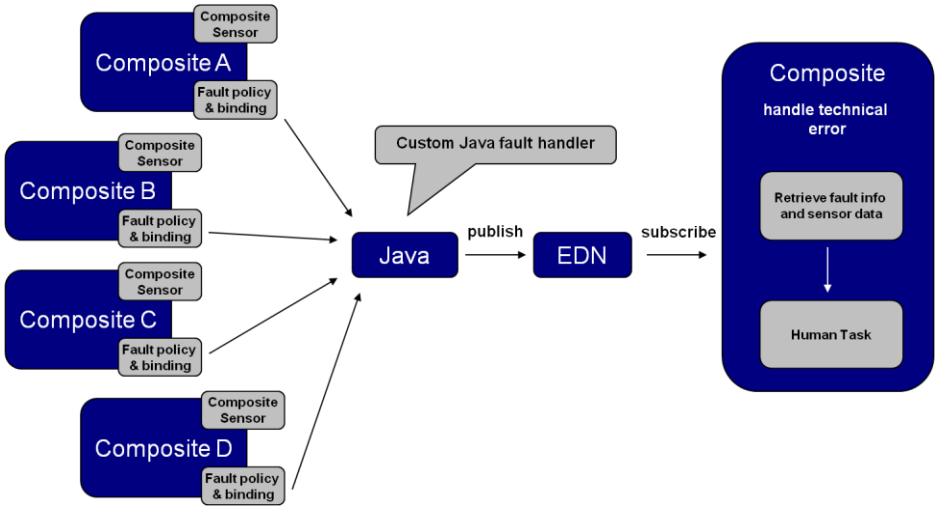
- Problem: Not overloading old non-scalable systems with high, or increased load.
  - Solution: Use result caching to cache the information (read-only operation) so the backend system is not invoked.
  - Solution: Use throttling to limit the number of concurrent requests to the non-scalable system.
- Problem: Unstable network between the service consumer (ordering process) and the external services.

    - Solution: Use retry mechanism of OSB to invoke the backend system multiple times. No fault management is necessary for the service consumer if the network interruption is only for a short time.
- Problem: Services do not guarantee 7*24 availability for one single instance.
    - Solution: Deploy multiple instances (endpoints) of the service and use the service pooling feature. No fault management is necessary for the service consumer if at least one endpoint is available
- Problem: Guarantee that the message will be delivered to a backend system.
    - Solution: Use queues and make sure that the queues are available, even if the Handle Order system is not.
    - Solution: Make sure that queuing run's in the same transaction as the service consumer so messages are not lost.

SOA Suite 11g

The Oracle SOA Suite provides several mechanisms to prevent faults from happening or to correct faults that have happened. The following list contains solutions that can be applied per problem:

- Problem: The process needs to return a Business Fault as result of an asynchronous message exchange as opposed to returning a fault as part of a synchronous response.
    - Solution: Design a separate Fault Message and Operation on the Callback contract (WSDL) and use that. The "Business Fault" is modeled as another operation on the Callback WSDL.
- Problem: When a backend system is not available, it should have no impact on Business Processes.
    - Solution: Use SOA Suite's fault management framework configure retries for the Mediator invoking the backend system. The fault management system is shipped with out-of-the-box recovery actions such as retry, terminate, and human intervention.
- Problem: Several services can return different types of faults, while the process that invokes these services only returns one type of fault.
    - Solution: Catch and handle the errors in the process and map them to the fault type that is agreed upon in the WSDL and returned to the service consumer (i.e. the caller of the process).
- Problem: Response messages from services can get lost in the backend system, i.e. the callback message will never arrive in the business process.
    - Solution: Define a timeout on the wait for answer with a BPEL Pick activity containing a timeout.
    - Undo the process by executing compensation that handles the fault by replaying the service invocations, but then with the opposite input or operation to reverse actions.
- Problem: How to handle unexpected technical faults in a generic fashion.
    - Solution: Use fault handling mechanism of SOA Suite to catch technical faults, execute a Java action that enqueues the fault on an error queue without adding process logic.
    - Create one process to listen to error queue and handle faults.
    - Retrieve process information by using (composite) sensors. This is shown in the following figure:



Summary and best-practices

The following table lists the issues that are addressed in this manuscript, the proposed solutions, and the product that offers the solution:

Issue	Solution	Product
Overloading product management system	ThrottlingResult cache	OSB
Service does not guarantee 7*24 uptime due to e.g. network problems	Muliple endpoints Service pooling	OSB
Guarantee message delivery to backend system	Availability of queues Enqueue and dequeue in service consumer transaction	OSB (and SOA Suite for XA propagation to OSB)
Returning business fault over asynchronous exchange from backend system	Separate operation and fault message	OSB and SOA Suite (callback contract between the two)
Backend service not available	Retry in Mediator using fault policy framework	SOA Suite
Business fault handling from service to process to consumer	Catch faults in process and reply fault to consumer	OSB and SOA Suite (correct contracts)
Detect missing response message	Timeout in pick activity	SOA Suite
Handle business faults	Compensation	SOA Suite
Avoid calling service twice	Set non-idempotent property	SOA Suite
Processes needing to deal with unexpected technical faults. All processes solving it in their own way using process logic.	Fault policy frameworks, error queue, generic error handler & (composite sensors.	SOA Suite

When implementing fault prevention and handling consider the following best practices:

- Differentiate between business and technical faults;
- Design with criticality, likeliness to fail, and cost in mind;
- Design service contracts with faults in mind: formally describe business faults in service contracts;
- Differentiate fault patterns in OSB and SOA Suite:
    - OSB: Retry, Throttling, transactions;

    - BPM/BPEL: Compensation, business fault handling, generic fault handler, and timeout.
- Handle unexpected errors generically;
- Don‘t use exceptions as goto’s;
- Make services autonomous and idempotent;
- Use eventing for decoupling;
- Fault-handling should be done on the scope of services as well as in wider perspective.

Contact address:

Name
Vennster
Postbus 31457
6503 CL, Nijmegen, the Netherlands

Phone:	+31 (0)6 52456043
Email	ronald.van.luttikhuizen@vennster.nl
Internet:	vennster.nl
Twitter:	rluttikhuizen
LinkedIn:	http://nl.linkedin.com/in/soaarchitecture