Tips and Tricks for Oracle SOA Suite by Oracle A-Team, Engineering, and Customers

Joseph Kardamis, IT Manager, The Sherwin-Williams Company
Sherwood Zern, Consulting Solution Architect, Oracle
Antony Reynolds, Product Strategy Director, Oracle
David Shaffer, Managing Partner, Middleworks

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Blending SOA and Agile practices

Joe Kardamis, Sherwin-Williams
IT Manager, SOA/Integrations
SOA + Agile = ???

◆ Why would you?
◆ How would you?
Why would you mix SOA and Agile?

◆ Business Agility!
◆ SOA’s aim (among other things) is to flatten the cost of change by making change smaller and less risky, thereby enabling the business to change as often as it needs
◆ Agile’s aim (among other things) is to deliver value as often as possible so as to elicit continuous feedback from the business and enable change
◆ Doing both will give you Even More Agility! And, as many teams are trending Agile, being able to blend the methodologies is beneficial to collaborating
How we have historically “done SOA”

◆ Wait, who’s “we”?  
  – The Sherwin-Williams Company (so, paint)  
  – Corporate IT, Internet Services (custom applications and solutions for various aspects of the company – external web sites and mobile tools, internally facing business process improvement, et cetera)  
  – SOA/Integrations Team (think: “Integration Competency Center”, or “SOA Center of Excellence” as defined by Wikipedia)
Okay, back to SOA then

From Oracle SOA Engineering Framework
What had to change?

- **Service Identification and Discovery**
  - Input: Requirements
  - Action: Analysis of Requirements to determine Service Candidates that are “worth it”, and discover what, if any, Services already exist which can fulfill the need
  - Output: Service candidates
  - Agile Analog: Epic/Feature Decomposition and Spikes
    - Risk: “Design by Decomposition” – if this step is not done deliberately, or done by the wrong parties, it may be missed
    - Mitigation: Ensure “SOA System Architect” involved in decomposition
    - Acceptance: The process is set up to embrace change
What had to change?

◆ Service Release Planning
  – Input: Current Services in flight plus new Service Candidates
  – Action: Whether or not existing Services are to be continued, and what Service Candidates are to be accepted by the team
  – Output: Updated Release Plan
  – Agile Analogs: Release Planning, Sprint Planning, Backlog Grooming
What had to change?

- **Service Definition**
  - Input: Fully understood functional and non-functional requirements associated with Service Candidate
  - Output: Service Contract
  - Agile “Analog”: Roughly, user story Acceptance Criteria
    - Inversion: Requirements for services in flight are collected, and published as a contract (specification) at release/promotion time
    - More change is expected, so less time is spent on definition/analysis
      - Generally, the service consumers are expecting this as well, so it’s not as pain-causing as with traditional waterfall
      - Interesting: Service “taxonomies” (e.g. app vs. shared, connectivity vs. data) are more fluid, and services tend to “evolve” from tier to tier
What had to change?

- Service Design, Implementation, and Testing
  - Input: Service Contract
  - Output: Service Interface and functional Service
  - Agile Analog: Implementation of user stories, regression testing
    - Story implementation covers design, implementation and testing
    - Generally each story will correspond to a single operation.
    - Implication: The Service Interface is in flux until it’s all done
    - Inherently iterative, so care needs to be taken when defining the Versioning Policy. As other teams are also iterative, they are more able to flex to your changes, especially while still in development
    - Design-Time Governance is fragmented across stories. Performing a pre-release check (e.g. Code Review) is recommended
What had to change?

- Service Deployment and OA&M
  - Agile Software Development Methodologies are more concerned with how to build the software, not how to support it, and as such, existing methods to perform these activities are fine as is
Did it blend?

- So far, yes! We’re roughly 8 months in to this “experiment”
- But there have been additional challenges as it relates to inter-team dependencies
  - Accommodating unplanned work is far easier in Kanban than in Scrum, which reduced the impact of inter-team dependencies
  - We hypothesize that shifting from centralized SOA CoE (Component Team) to “embedding” SOA developers in attempt to create “Feature” teams will result in a reduction of the number of inter-team dependencies
Thank you!

◆ Any Questions?
◆ Email: joe.r.kardamis@sherwin.com
Logistics and challenges

- Ideally, teams should be “Feature” teams
  - Feature teams can work a Feature from end to end entirely itself, realizing the business value without requiring additional coordination with other teams
  - Cross-team dependencies are reduced

- However, most of our teams were “Component Teams”
  - Standing up a new platform tends to gravitate towards systems
  - Our team (SOA CoE) is a Component Team, and is also the team through which many of the cross-team dependencies flow

- Therefore – LOTS of dependencies
Experiment 1: Scrum to Kanban

- Due to the large amount of churn and unplanned/unplannable work, a system with less heaviness on the planning side may be appropriate.
- Kanban is more about flow and less about “the plan for the timebox”.
- Kanban allowed us to flag when items are needed to help with prioritization, but allowed us to introduce new work mid-sprint without wrecking “the plan” (because “the plan” was never “done”).
Did it work?

- Yes! We were more productive

- No! Cross team dependencies still present challenges
Experiment 2: CoE to CoP

◆ “Dissolution” of physical SOA CoE
  – Sending out SOA developers to “embed” with other teams in attempt to promote “Feature” teams

◆ Concerns identified around keeping quality, cohesion among SOA team

◆ Mitigation: Creation/formalization of “SOA Community of Practice” to meet periodically to cover things like best practices, design time governance, code reviews, et cetera
Did it work?

- Don’t know yet! This experiment has just started...
SOA Stuff

- Oracle Practitioner Guides
More SOA Stuff

- **Oracle Reference Architectures**
  - SOA Foundation:  
  - SOA Infrastructure:  
Agile Stuff

- Agile Manifesto/Principles
  - http://www.agilemanifesto.org/
  - http://agilemanifesto.org/principles.html
  - Ooh look a SOA one... http://www.soa-manifesto.org/

- Scrum and Kanban stuff
  - http://www.collab.net/services/training/agile_e-learning
  - http://jpattonassociates.com/kanban_oversimplified/
  - http://leansoftwareengineering.com/ksse/scrum-ban/

- Agile at scale
  - http://www.scaledagileframework.com/
  - http://www.disciplinedagiledelivery.com/
SOA + Agile Stuff

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Program Agenda

1. Service Invocation Actions
2. SOA Direct-Binding
Program Agenda

1. Service Invocations

2. SOA Direct-Binding
Service Invocation Options

- Oracle Service Bus supports several different ways to invoke an external service – understand the differences and choose the right one for your requirements:

1. Service Callout
   - Enrichment of the payload
   - Blocking Call

2. Publish
   - Invoke the service asynchronously
   - Non-blocking call - continue with the message flow without waiting on a response
Service Invocation Options (continued)

3. **Routing**
   - Common mechanism to invoke a service
   - Demarcation between request and response pipelines
   - Uses asynchronous servlet to wait for response from HTTP-based service

4. **Java Callout**
   - Should only be used when invoking very fast services
   - examples: Read or Update a cache, quick calculations
A Common Challenge with Service Callout

• Symptom
  – OSB encounters STUCK threads

```
"[STUCK] ExecuteThread: '1' for queue: 'weblogic.kernel.Default (self-tuning)'' waiting for lock java.lang.Object@1de6564b WAITING
java.lang.Object.wait(Native Method)
java.lang.Object.wait(Object.java:503)
com.bea.wli.sb.pipeline.PipelineContextImpl$SynchronousListener.waitForResponse(PipelineContextImpl.java:1620)
com.bea.wli.sb.pipeline.PipelineContextImpl$dispatchSync(PipelineContextImpl.java:562)
stages.transform.runtime.WsCalloutRuntimeStep$WsCalloutDispatcher$dispatch(WsCalloutRuntimeStep.java:1391)
stages.transform.runtime.WsCalloutRuntimeStep$processMessage(WsCalloutRuntimeStep.java:236)
```
A Common Challenge with Service Callout

• Resolution
  – Assign Minimum Constraint Work Manager to Invoked Business Service
  – Set the number of threads (0 < Threads <= 2)

• Work Managers used should be unique per Business Service

• References
  – http://docs.oracle.com/middleware/1213/wls/CNFGD/self_tuned.htm#CNFGD112
Service Callout Tips

• Error Handling
  – Put the Service Callout in its own Stage
  – Add a Stage Error Handler

• The fault will not be returned in the SOAP Body (HTTP)
  – The fault will be returned in the $fault variable
  – Update the SOAP Body with the $fault contents
A Common Challenge with Publish

• Fire-And-Forget; works great, but ...
• Potential Issue: What the fetch happened to my messages?
  – Errors from remote service are not returned
  – Messages are not guaranteed delivery
A Common Challenge with Publish

• Resolution
  – Adding Routing Options to the Request Pipeline in the Stage
  – Set the Quality of Service to Exactly-Once
  – Add Error Handler to Stage
  – Errors encountered by remote service get returned to the error handler
  – Error handler needs to ensure that the messages are not lost
Routing Tips

• Can only have one Route per pipeline-request pair
• Use Route versus Publish if no need to continue processing through the request pipeline
• With HTTP transport the response is handled by an asynchronous servlet
  – Request thread is returned to the thread pool
• Setting the Quality of Service to Exactly-Once will use the same thread for the response as the request
  • Helps control the number of concurrent threads to a remote service
Java Callout Tips

• The invoked Java resource should be deployed within the OSB domain
  – Able to control its availability

• The invoked method must be static

• Make sure the operation executes quickly
  – The request thread is the same thread executing the Java method
Service Invocation Tip: Understand the OSB Threading Model

Program Agenda with Highlight

1. Service Invocation Actions
2. SOA Direct-Binding
SOA Direct-Binding

- Is the right choice for invoking a SOA composite IF your use case involves one of these three requirements
  - You need to propagate the Security Subject to SOA
  - You need to propagate the transaction context to SOA
  - You have an asynchronous process with a callback to the OSB from SOA
SOA Direct-Binding

• What you need to understand
  – SOAP/HTTP should be used if the use case does not need to provide the previously mentioned requirements
  – Timeouts cannot be set on this transport; could potentially lead to long running processes
  – Thread is blocked until a response is returned
  – The underlying XML object must be converted from OSB Xquery Tokenized Stream to a DOM object for outbound and vice versa inbound
Integrated Cloud
Applications & Platform Services
SOA Development Best Practices

Antony Reynolds
Product Strategy Director
SOA Suite Product Management Team
Program Agenda

1. Reducing your code
2. Reduce and Manage Storage (aka dehydration store)
Faults: Categories and Strategies

• System Errors
  – Potentially retry able e.g temporary network glitches, endpoint not available
  – Actionable - Bad composite code/transform

• Business Faults
  – Actionable - Related to bad payloads, dodgy code – you should try to catch all these so engine won’t retry them

• Identify common fault handling strategy, for each of two possible cases:
  1) If in-flight business transaction can be modified and continue:
     • Use Fault Policy Human Intervention to pause the flow at faulting invoke
     • Use User Action to:
       – Address error/modify data
       – Retry/Continue
  2) If business semantics require abort and restart/resubmit
     • Use fault policy Java Action to cause resubmit/log, followed by Terminate current transaction

Handling Faults with the Fault Management Framework doc:
http://docs.oracle.com/middleware/11119/soasuite/SOASE/bp_faults.htm#SOASE9904
Reusable Error Handling

- Reduce code with externalized fault/error handling
  - Avoid duplicating catch blocks all over process
  - Can be a common strategy for all composites or override for individual composites

- Use SOA Auto Recovery
  - Applies to faults which you have not caught and don’t have a fault policy for
  - Intermittent network related Retry able – let engine handle them
  - Configure auto recovery to run 24x7

- Use Fault Policy
  - Do not Retry Payload related faults (or business faults)
  - Use human intervention action in fault policy
  - Once in human intervention, data can be modified and instance can be aborted, retried or continued
Reusable Error Handling

• Feature: Fault Policy
  • For error handling external to SOA which does not impact the SOA/BPEL design or runtime
  • Policies are defined in XML
  • Re-useable across composites and components
  • Pre-defined actions
    retry, human intervention, replay scope, rethrow fault, abort,
    and custom Java actions

Using Fault Handling in a BPEL Process doc:
http://docs.oracle.com/middleware/11119/soasuite/SOASE/bp_faults.htm#SOASE478
Reduce Storage
Long Running Processes and Audit Trails Can Fill a lot of DB..

• May set completionPersistPolicy to Faulted
  
  • Audit composite applications at BPEL activity level
    – Set Audit Level to minimal
    – Identify mile stone activities in your process
    – In Java Embedded Activity use addAuditTrailEntry to store custom mile stone based audit
  
  • Another alternative is to use sensors on important mile stone related activities.
Reduce Storage

- Audit Policy feature in 12c will allow to further reduce audit trail data, where you choose to

- Configure audit trails for composite applications at BPEL activity level
  - Identify milestone activities in your process
  - see documentation: http://docs.oracle.com/cloud/latest/soacs_gs/SOASE/soa-debugger-jdev.htm#SOASE89046

```xml
<auditPolicies ..>
  <auditPolicy name="whilePolicy">
    <!-- do not audit assign-->
    <activity type="bpel:assign“ auditLevel="off"/>
    <!-- do not audit while-->
    <activity type="bpel:while" auditLevel="off"/>
    <activity type="bpel:exec“ auditLevel="production"/>
  </auditPolicy>
</auditPolicies>
```
Manage Storage

- Know your data - inflow rate, completion rate, fault rate
- Identify storage used
- Build your purge strategy and get your DBA familiarized with the SOA Suite metadata repository schema
- For 12c: review the documentation: [http://docs.oracle.com/cloud/latest/soacs_gs/SoAAG/soa-database-manage-plan.htm#SOAAG97777](http://docs.oracle.com/cloud/latest/soacs_gs/SoAAG/soa-database-manage-plan.htm#SOAAG97777)
- Automate Purge and plan/measure/test well in advance of production
  – 12.1.3 comes with out of box auto purge
Working With Very Large Schemas

David Shaffer
Managing Partner
Middleworks
dave@middleworks.com
Agenda

0  About Middleworks
1  Very Large Schemas
2  JDev Bundle Patches
3  Sparse Mappings
4  Non-Sparse Mappings
5  Additional Tips

For full XSLT tips and tricks doc, see: http://bit.ly/1M92Cq3
About Dave Shaffer / Middleworks

• Acquired into Oracle in 2004 via Collaxa, where BPEL engine came from - ran product mgmt for Oracle’s Integration products through 2011

• Founded Middleworks in 2011 to help customers and partners be more successful with FMW and specifically Oracle SOA / Integration technologies

• Remain very closely connected to Oracle product teams

• Offer independent strategy, architecture, best practices, issue resolution, referral and advisory services

• No implementation team – pull in partners as appropriate. This is a unique differentiator since we are not incented to always staff our own people...

• Can help find the right expertise, training, staffing (what happens if you ask your current partners “can you help us do <aaa> or <bbb> or ...?”)

• Contact me at dave@middleworks.com or C: 415.652.9712
Large / Complex Schemas

• A very common issue with packaged apps
• Mapping / data transformation can be the hardest part of integration
  – Understanding schema (sorry, can’t help with this…)
  – Tools issues with schemas, WSDLs, etc
  – Hard to traverse schemas for mapping, finding the right elements, etc
• This customer was using OTM’s Glog schema
• Just navigating Glog schemas in JDev 12c XSLT mapper was painfully slow
  – Everyone assumed this meant doing mapping manually and minimizing use of Glog schemas
JDev Bundle Patches

• 12c Bundle Patch 2 (and beyond) *immediately resolved* our JDev XSLT mapper issues

• Not everyone is aware of them and instructions for installing helpful

• Need metalink to access, which some developer downloads may not have

• For instructions to install bundle patch 3 (latest as of 10/15/15), see Jan van Zoggel’s blog post at:

  – [https://jvzoggel.wordpress.com/2015/08/14/patching-your-jdeveloper-12-1-3-with-soa-bundle-patch-12-1-3-0-3/](https://jvzoggel.wordpress.com/2015/08/14/patching-your-jdeveloper-12-1-3-with-soa-bundle-patch-12-1-3-0-3/) (or [http://bit.ly/1LPJZRI](http://bit.ly/1LPJZRI) for shortened link)
Sparse Mappings

• XML trees can be very large and you often only use a very small number of the nodes in your mappings
  – Constantly scrolling through nodes that are of no use to you

• Two strategies for dealing with sparse mappings

  1. Generate a schema from sample XML – can be done in XSLT mapper when selecting schema for source or target for your map
     – Can then Replace/Add Source or Target schema to switch back and forth

  2. Use XSLT view (new in JDev 12c)
     – Provides condensed look at the mappings you are creating – only see target nodes which are used
Sparse Mappings: XSLT View
Non Sparse Mappings
Non Sparse Mappings - Tips

- Right-click on target node and Set Display Scope to only show mappings to that node (and its descendents)
- Right-click any hidden area and use text search to find and display additional nodes you need
Additional Tips

• Reduce textual clutter with Abbreviate Text to make mapper hide prefixes and also set up custom abbreviations (e.g. “CPL” ⇔ “CustomerPartyList”)

• Searches can be cancelled (added in bundle patch 2) and you can reduce the Expension Depth (maybe set to approx 10)

• Don’t use automap to copy large input XML documents – use identity template instead

• Use the XSLT editor to generate test documents (rather than the XSLT testing tool), *if* your schema contains abstract elements or overloaded types

For full documentation of these tips, see: [http://bit.ly/1M92Cq3](http://bit.ly/1M92Cq3)