Principles and Tools for SOA Governance

May 22, 2005

Brent Carlson
VP of Technology
LogicLibrary, Inc.

http://www.logiclibrary.com
brent.carlson@logiclibrary.com

http://www.logidexassetcenter.com
http://lab.msdn.microsoft.com/logiclibrary/logiclibrary.aspx

Agenda

- Setting the Context
  - Service-oriented architecture (SOA) definitions – common ground
    - **What is SOA?**
    - **What are software development assets (SDAs)?**
  - Working architectural contexts for SDA reuse
    - **Application**
    - **Technical**
    - **Business**
  - Introducing the service production/distribution/consumption lifecycle
- SDA reuse best practice 1
  - Production: Pragmatic service interface modeling and definition
- SDA reuse best practice 2
  - Production: Lifecycle review points
- SDA reuse best practice 3
  - Production: Managing produced services as internal "products"
- SDA reuse best practice 4
  - Distribution: Effectively delivering Web services and other SDAs to potential consumers
- SDA reuse best practice 5
  - Consumption: Service usage traceability, impact analysis and ROI

SOA Concepts

SOA Concepts

- What is service-oriented architecture?
  - Definition and delivery of core application functions through a series of coarse-grained services meant to be assembled through a message-oriented infrastructure
- Not all Web services fit into an SOA
  - e.g., fine-grained RPC-type services
- Not all SOAs use Web services technology
  - e.g., guaranteed messaging using message-oriented middleware (MOM) technology
- Services may be implemented using a variety of techniques
  - COM/.NET/J2EE components, integration server adapters, …
- Services are inherently reusable software development assets (SDAs) and should be managed as such

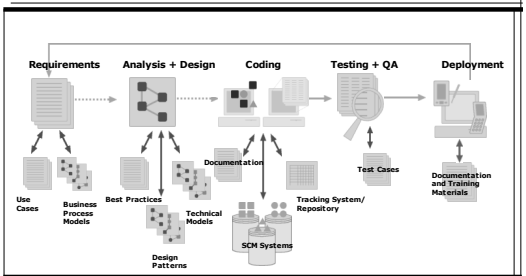
What Industry Experts are Saying about SOA

- "Service orientation and Web Services are two powerful application architecture trends that, individually and in combination, promise to improve business flexibility and responsiveness by reducing the time required to build, change and integrate enterprise applications both internally and across company boundaries" – Randy Heffner (Forrester)
- "At the core of any service-oriented development of applications effort is an institutionalized reuse program." – Gartner Group
- "SOA is an architecture -- not a framework. It is a style of design -- not a tool. On the other hand, "web services" is a framework -- a set of tools (middleware) to help you implement applications and systems that adhere to SOA design principles" – Anne Thomas Manes (Burton)

What is an SDA?

- An SDA is "something of value to an IT organization," including:
  - **Knowledge** assets
    - **Architectures, best practices and processes, design patterns, …**
  - **Executable** assets
    - **Web services, data views, components, applications, …**
- SDAs are composed of ***metadata***, such as:
  - **Artifacts:** work products, including code, schemas, models, executable modules, …
  - **Classifiers:** searchable and reportable values such as keywords, development effort, owner, language, …
  - **Relationships to other assets:** dependencies, prerequisites, other asset versions, …

SDA Development Lifecycle

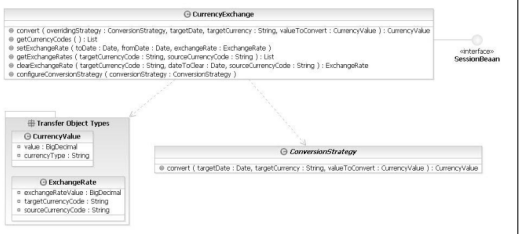


SDAs within an SOA

SDAs are key building blocks within an SOA

- Services themselves are SDAs
  - **Meant to be published and consumed**
  - **Need to be managed over their typically multi-versioned lifecycle**
- … and services consume other SDAs
  - **Other services**
  - **Components (both business and technical)**
  - **Legacy APIs**
  - **Knowledge assets (e.g., Enterprise Solution Patterns, such as Service Interface and Service Gateway)**
  - **…**

An Example SDA: CurrencyExchange



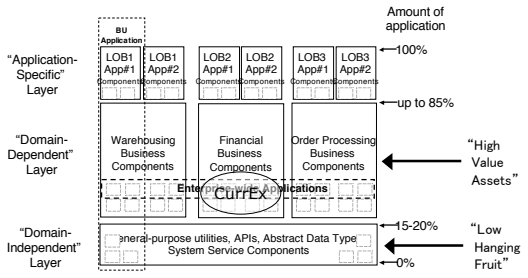
- How do we get to this design?
  - We will explore in more detail in Best Practice 1…

SDAs Reuse Working Contexts

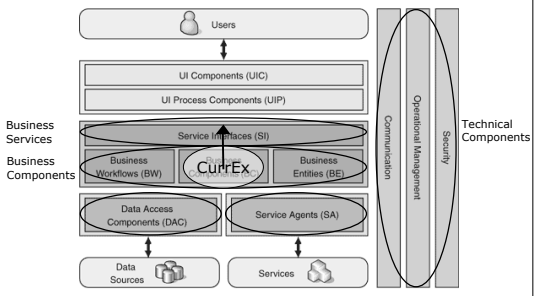
SDA Working Contexts

- SDAs need to be managed simultaneously from three primary working contexts
  - **Application/integration context:** core patterns and reference implementations to be used in assembling enterprise application capabilities to support the business architecture
  - **Technical context:** underlying technology stacks to be used in implementing and deploying enterprise applications
  - **Business context:** business process modeling -> functional service and component identification and normalization
- These contexts are best described with a combination of models and taxonomy specs

Application Context Example

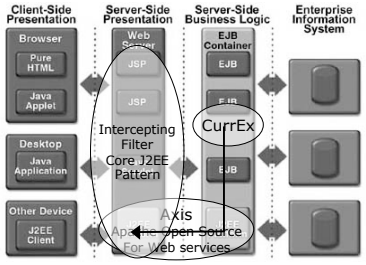


Technical Context: .NET



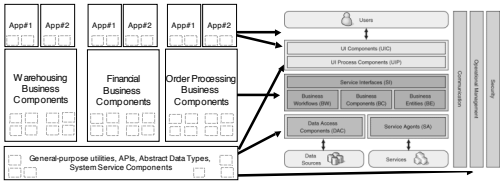
* http://msdn.microsoft.com/architecture/patterns/

Technical Context: J2EE



* http://java.sun.com

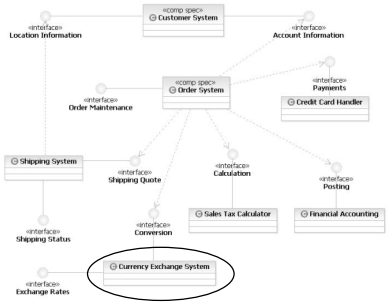
Mapping SDAs to the .NET Technical Context



Business Context

- Technical context is only part of the story
  - It's just as important to align your reusable asset development work with your company's business context
- What is a business context?
  - The overarching business requirements driving development projects
    New component/service development, application integration, new/reworked application development
  - In other words:
    What business processes really matter to our company?
    And what business functions are demanded by those business processes?

Our SDA's Business Context: Retail (including e-commerce)



- Best Practice 1 will show us how to derive this business context/architecture

Business Context Elements

- Primary UML construct is class diagram laying out coarse-grained reference components/services
- Other UML constructs to consider:
  - Use Cases
    Establish initial requirements for business function
    Actors
    Preconditions
    Functional scenario / use case steps
  - Activity Diagrams
    Describe detailed process or subprocess flow underlying a use case
    Specific activities can be mapped to functional capabilities to be implemented as Web services

Effective Reuse Initiatives: Architecture, Governance, and Tools

What are the major challenges to delivering a successful reuse initiative?

- Enterprise Architecture
  - which leads to…
- Process and Governance
  - which is supported by…
- SDLC Tooling
  - including an SDA library

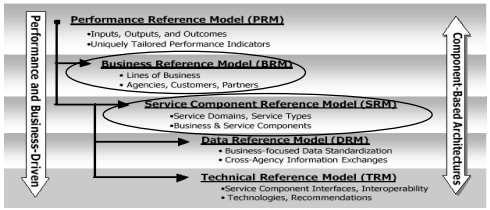
What are the major challenges to delivering a successful reuse initiative?

- Architecture:
  - ***Establishing*** a standard architecture
  - ***Communicating*** the architecture
  - ***Aligning*** architecture and functional capabilities to provide flexibility
  - ***Understanding*** the business requirements to assure that IT projects meet the requirements
- Process and Governance:
  - Defining a process to ensure maximum buy-in and minimum political friction
  - Driving governance so people will follow the architectural and development guidelines
- SDLC Tooling:
  - Understanding the landscape and using the right tool for the right job (don't drive a nail with a screwdriver)

Enterprise Architecture (EA) Scope and Objectives

- EA is a relatively new discipline
  - Scope and objectives are in flux
- My view – EA has three primary responsibilities:
  - Define architectural taxonomies
  - Deliver supporting knowledge and executable assets to downstream consumers
  - Govern appropriate use of those assets within the scope of IT projects

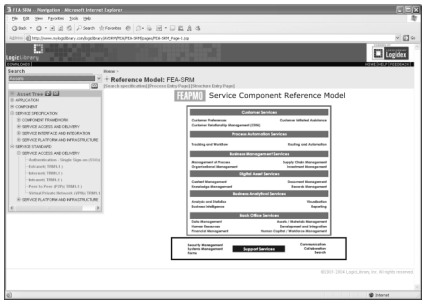
Case Study: The FEA And Agency Business Process Standards



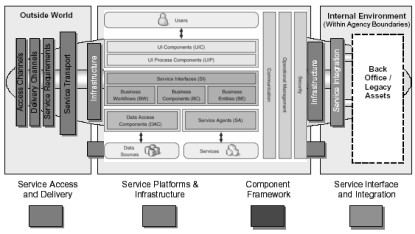
- The FEA provides high-level business process guidance to agencies
  - BRM and SRM provide structure
  - Agencies are responsible for:

    Populating processes within that structure
    Managing those processes and their alignment to SDAs

Tool Example: FEA Service Component Reference Model Within Logidex

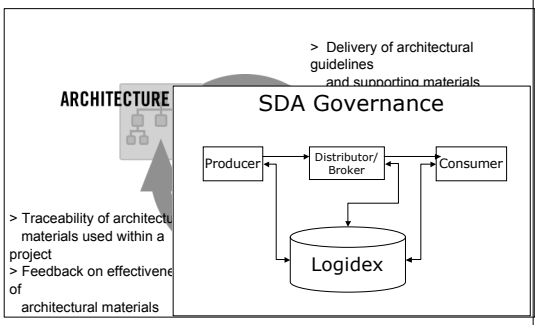


Case Study: FEA Technical Reference Model



- The FEA TRM defines a coarse-grained distributed application architectural framework that supports service-oriented and component-based development
  - The .NET (and J2EE) technical architecture fits nicely within this architectural framework

Architecture and Governance



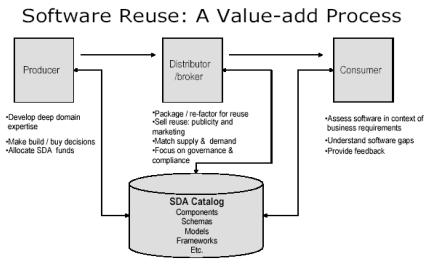
What are the major challenges to delivering a successful reuse initiative?

- Architecture:
  - Establishing a standard architecture
  - Communicating the architecture
  - Aligning architecture and functional capabilities to provide flexibility
  - Understanding the business requirements to assure that IT projects meet the requirements
- Process and Governance:
  - ***Defining*** a process to ensure maximum buy-in and minimum political friction
  - ***Driving*** governance so people will follow the architectural and development guidelines
- SDLC Tooling:
  - Understanding the landscape and using the right tool for the right job (don't drive a nail with a screwdriver)

SDA Governance And Reuse

- SDA reuse is supported by production/distribution/consumption roles
  - **Production:** Identification and preparation of existing and newly-defined candidate reusable SDAs
  - **Distribution:** Publication of those SDAs into SDA library
  - **Consumption:** Use of SDA library to discover appropriate reusable SDAs on a per-project basis
- SDA reuse effectiveness is assessed by management governance and ROI review

SDA Reuse Lifecycle Revisited…



What are the major challenges to delivering a successful reuse initiative?

- Architecture:
  - Establishing a standard architecture
  - Communicating the architecture
  - Aligning architecture and functional capabilities to provide flexibility
  - Understanding the business requirements to assure that IT projects meet the requirements
- Process and Governance:
  - Defining a process to ensure maximum buy-in and minimum political friction
  - Driving governance so people will follow the architectural and development guidelines
- SDLC Tooling:
  - ***Understanding*** the landscape and using the right tool for the right job (don't drive a nail with a screwdriver)

Which Comes First: Reuse Process or Reuse Tooling?

This is a false dichotomy: the right answer is "both!"

- Organizations need to avoid "Analysis Paralysis" when defining process
  - So much focus on defining the perfect process that nothing concrete ever gets done
- But deploying a library without a process will result in:
  - Bad first experiences (which are difficult to recover from)
  - Mixed-bag results – assets of all shapes, sizes, and levels of quality with nothing to differentiate them

Which Comes First: Reuse Process or Reuse Tooling?

- Process and tooling should be defined and deployed iteratively and in parallel
  - Establish "the way to do things" up front so you don't have to break bad habits later
  - Implement an environment that supports iterative development of your Process

    Manual implementation of your reuse process will result in too much perceived additional work and will be too easy to bypass
  - Execute a Pilot that provides feedback about the Process that can then be used to determine the next iteration of the process

Tools Enablement of SDA Reuse

- Traditionally reuse has occurred via **word-of-mouth**
  - Effective for very small groups but doesn't scale
- Various **tool-based attempts** have been made in the past, including:
  - Spreadsheet "registries"
  - Full CASE tooling
  - Version control repositories
  - Ad-hoc collaboration databases
- New **SDA metadata library (repository)** space has emerged in past 2-3 years
  - Gartner March 2004 report and Magic Quadrant on Metadata Repositories
  - These **tools**, plus **rationalization** of development space into **.NET** and **J2EE** component and service architectures and dramatically **improved IDE tooling** (e.g., SAP NetWeaver Studio, VS, WSAD, Eclipse) enable for the first time effective SDA reuse practices and governance

How can an SDA metadata library enable software reuse?

- An SDA metadata library should:
  - Pull the pieces (i.e. artifacts) together to support the software development process and provides a searchable environment that links to the other systems such as:

    Source Code
    Requirements management
    Document management
    Defect tracking
    Etc.
  - Provide an environment where it is easy to find high quality assets within preferred tools (such as IDEs) so that they will be reused
  - Make it easy to implement governance policy through configurable automation and audit trail support

BREAK

Reuse Best Practice 1

Production Best Practice: Pragmatic Service Modeling/Definition

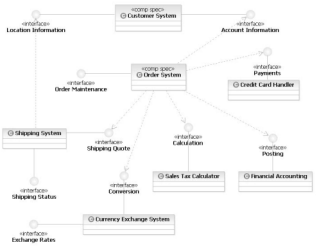
- Services within an SOA cannot be developed in a "bottom-up," ad-hoc manner
  - Become YALOT (yet another layer of technology)
  - More spaghetti code of a different form
- But services also cannot be defined solely in a "top-down" manner, which leads to either:
  - Analysis paralysis: continual refinement of a model hoping to reach perfection (which never comes), *or*
  - "Big-Bang" projects: trying to implement everything at once, usually with disastrous consequences

Balancing Top-Down and Bottom-Up Service Definition

- Objective: striking a pragmatic balance between where the business is and where it needs to go
- Where the business is:
  - Current set of strategic applications
    - **Usually implemented on different technologies**
    - **Often locked into rigid business processes**
- Where the business needs to go:
  - Loosely-coupled business services
    - **Exposed via a common technical infrastructure**
    - **Supporting flexible business process definition**

Top-Down Analysis

- Establish a coarse-grained business model
  - Driven by key business processes
  - Laying out a roadmap for prioritized service definition and development
- How do we build this reference model?
  - Extracting requirements from prioritized use cases

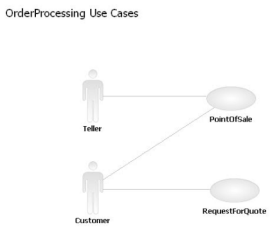


Case Study: Evaluating Currency Conversion Requirements

- Apply our top-down requirements assessment principle by selecting two business processes to evaluate:
  - Point Of Sale
  - Request For Quote
- Processes are diverse enough to explore different aspects of currency conversion within the context of the e-commerce domain

Deriving our Reference Model from Use Cases

- Sample Use Case Diagram

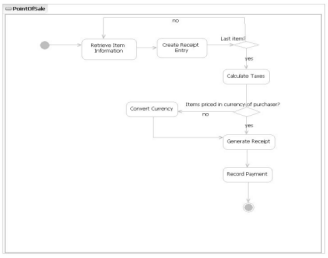


Point Of Sale Requirements

- Customer selects items and presents to Teller
- Teller processes items serially
- Process must take into account Customer's desire to pay in alternative currency
  - e.g., airport duty free shop

Use Case 1: Point Of Sale

- Drilling into Point Of Sale Activity Diagram…

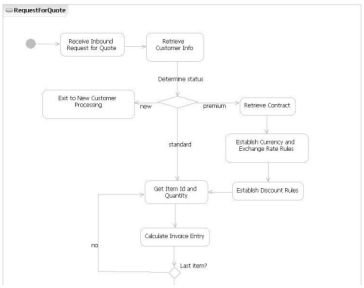


Request For Quote Requirements

- Inbound Customer requests provided electronically
- Customer must be evaluated for priority
  - New
  - Standard
  - Premium
- Premium Customer is eligible for policy overrides
  - Discounting rules
  - Currency conversion (e.g., contractual conversion rate)
- Quote response must be electronically returned to Customer

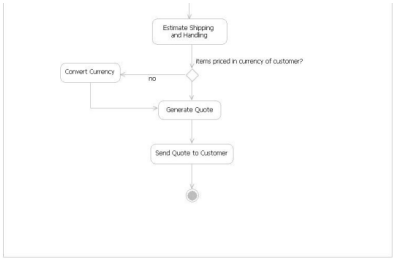
Use Case 2: Request For Quote

- Drilling into Request For Quote Activity Diagram…

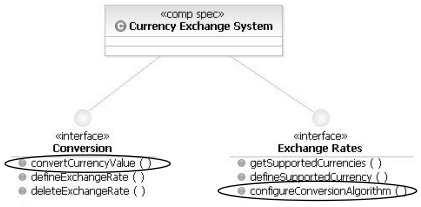


Use Case 2: Request For Quote

- Drilling into Request For Quote Activity Diagram…



Deriving Our Coarse-Grained Reference Component

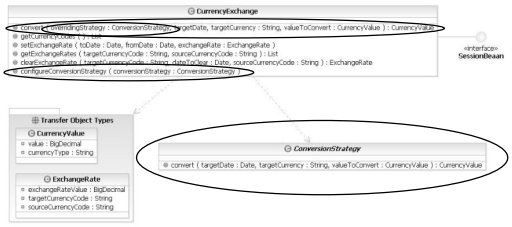


- Both use cases require a conversion operation
- Request For Quote use case requires configurable conversion algorithm

Bottom-up Analysis

- Use the coarse-grained, top-down business model to prioritize service development
  - Formalize service definition for prioritized services
  - Assess current set of production applications to understand which aspects of those applications are candidates to support required services
  - Combine and re-factor application capabilities by implementing adapters
    Adapters provide necessary glue and compensation logic
- Patterns provide excellent guidance – use them!
  - .NET: Enterprise Solution Patterns
  - J2EE: Core J2EE Patterns
  - General: Enterprise Integration Patterns (Gregor Hohpe, et al)

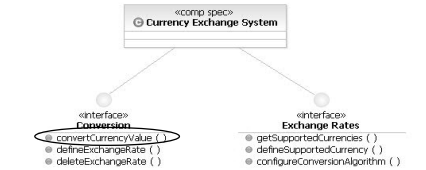
CurrencyExchange Design Revisited



- How was our design affected by requirements?
  - Point Of Sale use case defined basic convert operation
  - Request For Quote use case introduced requirement for Conversion Strategy

Aligning our SDA with our Reference Model

- Returning to our component model…
  - Map our reference component to our example asset



Reference component operation	SDA operation
getSupportedCurrencies	getCurrencyCodes
defineSupportedCurrency	no direct equivalent
convertCurrencyValue	convert
defineExchangeRate	setExchangeRate
deleteExchangeRate	dearExchangeRate

Tool Example: Use Case and Reference Model Development within Rational System Architect

BREAK

Reuse Best Practice 2

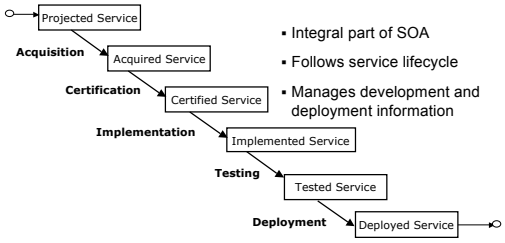
Production Best Practice: Review Points and Teams

- Best practice recommendation: *virtual/matrixed SDA architectural review team*
  - **Team members:**
    - *Team leader* **dedicated to SDA reuse program**
    - *Matrixed team members* **drawn from project teams**
      - Lead designer/developer skills required
      - 10%-20% job responsibility
  - **Team objectives/responsibilities:**
    - *Identify* **candidate reusable SDAs – "active discovery"**
    - *Review* **proposed reusable SDAs – asset hardening**
      - Adherence to architecture
      - Necessary functionality implemented and supported
      - Mandatory artifacts provided
    - *Publish* **approved SDAs into SDA library for consumption**
    - *Recommend* **future resource allocation for key reusable SDAs**
      - Expanded funding for key SDAs
      - Transfer of key SDAs to common SDA support group

Recommended Review Points

- At a minimum, organizations should review services under development at these points in the SDLC:
  - ***Requirements complete:*** all business requirements documented and initial service definition specified (ideally as WSDL), allowing reviewers to validate service against business context
  - ***Design complete:*** Implementation approach defined with sufficient documentation (e.g., UML design models completed, relevant legacy APIs identified) to allow reviewers to validate design against technical and application/integration contexts
  - ***Implementation complete:*** Service implemented and deployed in a test environment, with sufficient supporting documentation (e.g., sample client code, automated test cases, usage guide) to enable a potential consumer to understand the service
- Other review points may also be appropriate based on organizational needs and objectives
  - Example: CNA Insurance case study (coming right up…)

Case Study: CNA Insurance and Logidex



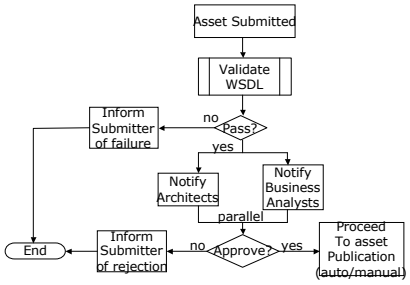
- Integral part of SOA
- Follows service lifecycle
- Manages development and deployment information

Applying Our Recommended Review Points

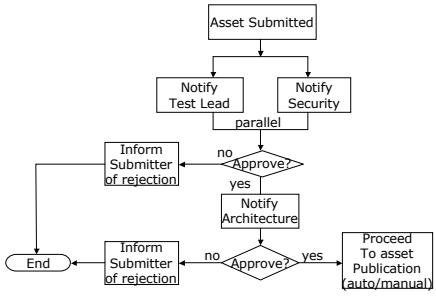
Enterprise has defined their services SDLC to include reviews at these points in the lifecycle:

- ***Requirements complete:*** all business requirements documented and initial service definition specified (ideally as WSDL), allowing **business analyst** and **architecture** reviewers to validate service aglainst business context
- ***Design complete:*** Implementation approach defined with sufficient documentation (e.g., UML design models completed, relevant legacy APIs identified, test plan defined) to allow **architecture, security** and **test lead** reviewers to validate design against technical and application/integration contexts
- ***Implementation complete:*** Service implemented and deployed in a test environment, with sufficient supporting documentation (e.g., sample client code, automated test cases, usage guide) to enable a potential consumer to understand the service; **architecture**, **performance** and **operations** reviewers complete final review before deployment

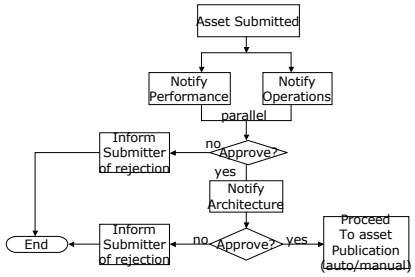
Requirements Complete Review Process Scenario



Design Complete Review Process Scenario



Implementation Complete Review Process Scenario



Tool Example: SDA Production Review and Governance using Logidex

Reuse Best Practice 3

Production Best Practice: Managing Services as "Products"

- Service producers need to treat their services as "products"
  - Regular and well-defined release cycle
    Often enough to meet consumer needs on a timely basis
    But not so often as to churn existing consumers
  - Backward compatibility wherever possible
    Give existing consumers time to migrate off of deprecated operations
    n-1 version support at a minimum
  - Requirements gathering mechanisms from current and potential "customers"
    Consider establishing a "product manager" role that manages the aggregate set of business requirements for the service and works to prioritize requirements with its current and potential consumers

What does a Product Manager do?

- Product Manager defines the "what"
- Development/project teams define the "how"
- Product Manager must be able to bridge the business and technical worlds
  - Not necessarily an expert in either but able to quickly grasp concepts and effectively communicate in both directions to both audiences
- What drives a Product Manager's decision making process?
  - Competitive feature (e.g., new business process)
  - Customer enhancement (e.g., enabling a new UI approach)
  - Strategic initiative (e.g., business process refinement)
  - Architectural enhancement (e.g., restructuring application infrastructure to enable flexibility or reduce costs)

"Service as Product" Impacts on SDLC Tools

- Version Control Repository
  - Establish a baseline whenever a new version of a service is released into production – must be able to simultaneously maintain production service while developing next version of that service
- Requirements Management / Defect Tracking
  - Manage your requirements and defects at a version basis – both originating version and target version for resolution
- SDA Management
  - Maintain all "valid" versions of a service within your SDA library
    "Under Development" – available for requirements gathering and application development team planning purposes
    "Production" – mainline version for use in new development
    "Retired" – still in use by existing apps but not allowed for use by new apps
    "Obsolete" – all apps should be migrated off this version; version metadata is maintained for traceability / audit purposes only

Version Control Repository Rules of Thumb

- Keep a main code line and branch versions off of it
  - As opposed to branching each new version and working on the branch
  - This is fundamental to deferring the branch as long as possible
    Why defer? Branching results in multiple maintenance…
- Branch only on version boundaries
  - Thread a label upon entry to final QA cycle but defer creating a branch for as long as possible to avoid dual maintenance
    Labels establish a compatible "path" through a set of related files – in effect "wiring" those files together to support a build or branching process
  - Only branch when you close down the release or are forced to branch to avoid conflict between current version defect resolution and next version new development, whichever comes first
- For "current" version of SDA, establish labels to represent different levels of code maturity
  - e.g., "dev" (head revision), "QA" (promoted jobs ready for system test), and potentially additional levels depending upon SDLC process
  - Establish a job promotion process to move revisions through the label stack

http://www.perforce.com/perforce/bestpractices.html

Tool Example: Versioning Strategy within Perforce Source Code Repository

BREAK

Reuse Best Practice 4

Distribution Best Practice: Integrated Asset Library

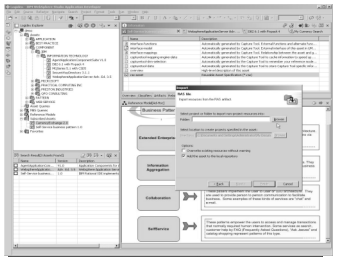
- Ad-hoc distribution schemes are sufficient for two or three services used by a small community, but don't scale
  - Spreadsheets and static websites get out of date
  - "Call the architect" turns the architect into an information bottleneck
  - UDDI registries are not suited for development use
    - **Limited service metadata**
    - **Often difficult search UI**
    - **Not well suited to managing other SDA types**
- Best practice recommendation: *SDA library (e.g., Logidex) to distribute assets*

Important SDA Library Features

- Consider these important features when evaluating SDA libraries:
  - Governed and configurable asset metadata ***assembly*** and ***validation***
    - **Standardized metadata definition**
    - **Per-asset-type metadata validation and enforcement**
  - Configurable (manual vs. automatic) asset publication
    - **Newly defined SDAs**
    - **Updated SDAs**
    - **New versions of existing SDAs**
  - Passive and active distribution modes
    - **User-based SDA subscriptions**
    - **Automated search notifications during asset creation/update**
    - **Mutliple search modes**
  - Multiple UI options
    - **Thin-client**
    - **Deep IDE integration**
    - **API-based integration**

Example: Logidex Deep IDE Integration

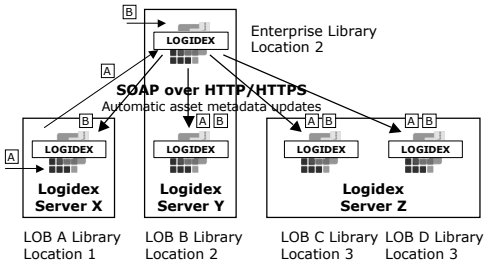
- SAP NetWeaver Studio
- Microsoft VS
- IBM RAD/WSAD/Eclipse



Federated Library Scenario

- Scenario: large enterprise with multiple lines of business establishing an SOA/reuse initiative
  - **Geographically dispersed IT teams**
  - **Differing levels of development process maturity across LOBs**
  - **LOB-specific sensitive assets/applications (e.g., privacy, legal restrictions, security levels)**
- Issues:
  - Establishing an asset maturity model
    - **LOB-specific assets (both as-is and hardened)**
    - **Enterprise-wide assets**
      - Promoted from LOB
      - Driven/built by enterprise architecture team
  - Supporting a seamless end-user search experience
  - Providing sufficient local asset library performance without duplicating asset work products across geographies

Federated Library Example



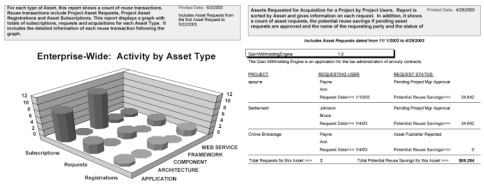
Reuse Best Practice 5

Consumption Best Practice: Project-based SDA Usage Tracking

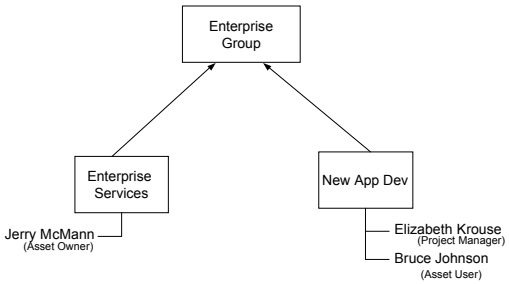
- Best practice recommendation: ***named user consumption***
  - **Project-scoped interactions/tracking**
    - Enables reporting and management of asset consumption
  - **SDA acquisition**
    - Configurable approval signoffs based on organization's process
  - **Project and asset-specific collaboration**
    - Discussion forums
    - Persistent searches
    - Asset notifications
- Access alternative: ***anonymous consumption***
  - **Suitable for lightweight/casual users**
  - **Read-only interaction with library**
  - **Restricted tracking and collaboration activities**

SDA Reuse Management Example: Logidex ROI and Asset Usage Reports

- SDA ROI reporting examples:
  - "Big Picture" rollup
  - Project-specific SDA reuse ROI



Case Study: SDA Acquisition Governance



Case Study: SDA Acquisition Governance

1. Jerry McMann publishes approved component to Logidex (completed prior to demo)
2. Bruce Johnson searches for component, finds it and determines it is appropriate for his project (at this point Bruce is allowed to view documentation only and is not allowed to access source code or executables)
   a. Fills out justification comment
   b. Initiates asset acquisition
3. Elizabeth Krouse is notified of pending asset acquisition for her project
   a. Elizabeth reviews justification comment and approves asset for use in her project
4. Jerry McMann is notified of planned use of the component
   a. Jerry reviews justification audit trail and also approves asset for use in the requesting project.
5. Bruce can now retrieve executable file for use in his development work

Tool Example: Asset Consumption Governance using Logidex within Rational Software Architect

Session Summary

To be effective, your SOA initiative needs to:

- **Define application, technical and business contexts**
  - Technical architectures (J2EE and .NET) and patterns do a great job of defining technical context
  - Once business analysts have defined your business context for you – listen to them!
- **Align existing application inventory against prioritized business processes**
  - Don't "boil the ocean." Pick the key systems that support your business needs and blend top-down and bottom-up approaches to service definition and implementation
- **Define and manage your services production/distribution/ consumption process lifecycle**
  - Tools are key to delivering quality assets to consumers and tracking where assets are used
  - Well-defined processes are just as important

    "Just enough process" – don't overwhelm your project teams with unnecessary workload, but also provide enough guidance at key points in the production and consumption lifecycles to make sure things stay on track