Establishing a Service Factory

Federal SOA Community of Practice
14th SOA for eGovernment Conference
October 2, 2012

Dave Mayo

www.everware-cbdi.com
Topics

- Corporate Profile
- Architecture – Engineering Model
- What is a Service Factory?
- Twin-Track Development
- Product Line Engineering
- Governance
- Service Factory Methods & Tooling
  - Service Architecture & Specification
  - Model Driven Development
  - Agile Methods
Everware-CBDI is a Leader in Applied SOA

**Worldwide SOA Reputation**
- Knowledgebase of SOA Best Practices – CBDI-SAE™
- Keynote Speakers on SOA
- SOA Metamodel Submission to OMG (SoaML)

**Leadership in Advisory Groups & Standards Bodies**
- Industry Advisory Council
- Federal SOA Community of Practice
- Object Management Group (OMG)

**Authors/Publications**
CBDI Journal (130+ Editions)
Articles (EA Journal, Microsoft Journal)
White Papers (Federal CIO Council, IAC, AFEI)
Books (Service Orientation, Information Modeling)

**Profile**
- Small Business
- Founded in 1997
- Offices in
  - Fairfax, Virginia
  - London, UK

**Offerings**
- Architecture Services
- SOA Enablement Services
- Application Modernization & Development Services
Implementable Architecture: Standard Engineering Model

Customer
- Define & Communicate Needs
- Provide Funding

Architecture
- Balance Demands
- Rationalization/Alignment
- High Level Structure
- High-level Solution Design

Engineering
- Apply Constraints
- Detailed Specification

Construction
- Build & Assemble
- Hand off to QA

Functional Responsibilities

Remember: Architect, Invest & Implement?
Implementable Architecture: IT Roles

Business Model (CIM)
- Business/Mission
  - Business Capability Requirements
- Enterprise Architect
  - Analytical Products
  - IT Innovation

Solution Model (PIM)
- Solution Architect
  - Solution Portfolio
  - Service Portfolio
  - Architecture Policies
  - Design Guidance
- Service Architect
  - Service Specification

Platform Model (PSM)
- Software Engineer
  - Design Patterns
  - Technical Patterns
  - Domain Knowledge
- Application Developer
  - Engineering Design Specification

Contract-Based Interfaces

Each level guides and constrains the next level.
Implementable Architecture: IT Roles

Business/Mission
- Business Capability Requirements
- Analytical Products
- IT Innovation

Enterprise Architect
- Solution Portfolio
- Service Portfolio
- Architecture Policies
- Design Guidance

Solution Architect
- Solution Design
- Design Patterns

Service Architect
- Service Portfolio

Solution Engineer
- Service Specification

Service Developer
- Service

Solution Assembler
- Engineering Design Specification

Contract-Based Interfaces

Each level guides and constrains the next level.
What Is a Service Factory?
Service Factory: What is the Problem to Solve?

- IT Solution Delivery Issues
  - Quality
  - Cycle time
  - Risk
- Inconsistency of approach
  - Across teams (skills)
  - Semantics
  - Delivery technology
  - Methods
- Lack of reuse of implementation and process assets
  - Architectural mismatch
  - Ad hoc / program focused reuse
- Solutions Development projects do not have budget, charter or business scope to deliver shared services
Service Factory: Concepts & Drivers

- **Purpose**
  - Execute repeatable process to produce shared services in efficient manner

- **Specialization of Software Factory**
  - Manufacturing analogy
  - Assembly from components
  - Automation of standardized, repeatable processes
  - Mass customization
  - Produces families of products

- **Factory Components**
  - Skilled workers with diverse roles
  - Automated tools for design, construction and coordination
  - Engineered processes, techniques and materials
Service Factory

- **Systematic approach to reuse**
  - Implementation Assets – tools, design patterns, frameworks
  - Process assets – test cases, harnesses, design methods
  - Production assets – software artifacts providing ROI through reuse

- **Highly automated service software development**
  - Building families of similar software service products
  - Assembling self describing service components
  - Developing DSL and tools using language definition, code generation, tool development technologies – massive reduction in handwritten code

- **Massive scalability**
  - Larger projects
  - Geographical, Sector distribution
  - Extended product life cycles without loss of agility
Service Factory Fundamentals

- **Frameworks**
  - Bootstrap implementations of services
  - Based on common architectural styles

- **Language based tools**
  - Support development of services
  - Adapting, configuring and assembling framework based components

- **Using tools to engage customers**
  - Respond to changes in requirements rapidly
  - Building software services incrementally

- **Capturing design decisions in a form that directly produces executables**

- **Brings all skillsets, disciplines, tools together in a single organization**

- **Centralized funding**
Solution Provisioning: Twin Track Process

Current: Build to Order

Future: Assemble to Order

- Application Based Business Demand
- Capability Based Business Demand
- Architecture and Portfolio

SDLC

Solution Assembly

Service Provisioning

Service Inventory

Service Factory

Solutions
Service Factory Supports the “Provide” Swim Lane of the CBDI-SAE™ Process

Consume

Solution Provisioning

Solution Component Requirement

Solution Assembly/Implementation

Legacy Application Reengineering

Solution Provisioning

Solution Component Requirement

Service Implementation Requirement

Solution (implemented)

Service Implementation

Provide

Service Provisioning

Service Implementation Requirement

Service (implemented)

Legacy to Service Reengineering

Service Provisioning

Service Implementation Requirement

Application Modernization Plan

KDM Packages

Application Modernization Plan

KDM Packages

Application Modernization Plan

KDM Packages

Solution Architecture

Service Portfolio Plan (Service Architecture, Service requirements)

(KDM Packages)

Service Portfolio Plan (Service Architecture, Service requirements)
**Service Factory: Context**

- **Business Capability Priorities**
- **Solution Portfolio Plan**
- **Service Portfolio Plan**
- **Legacy Portfolio Plan**
- **COTS / GOTS Components**

**Service Factory**

- **Reference Architecture**
- **Service Registry / Repository**
- **Solution Assembly Teams**
- **Service Specification / Contract**
Service Factory: Concepts & Drivers

Reference Framework

Software Factory Schema

Service Specification
Customized Tools
Service Implementation
Service Delivery

Reusable, Configurable Assets

Product Line or Family Analysis
Product Line or Family Design
Product Line or Family Implementation
Product Line Development
Software Product Line Principles

- A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

  -- Software Engineering Institute

- Strategic & systematic reuse
- Goal is to manage the product line, rather than products individually
- Configuration management becomes critical
- Commonality and Variability
Commonality
- Implement common service behavior patterns as reusable components
- Raise level of abstraction of reuse

Architect for Variability
- Deliver policy driven service architecture
- Specialize family reference architecture to separate the aspects of the service that change at different rates

Prediction
- Deliver generalized service components for family that are designed to be configured to create variants

Exploit the Predicted Changes
- Standard infrastructure
- Platform for service reuse – SDK, SLA plus related processes
- Upgradeable designs and components
**Product Line Engineering Enables Mass Customization**

- Employ base products designed to be customised for individual customer / process requirements
- Engineer for faster delivery of finished product
- Enable concurrent delivery for multiple customers
- Maintains integrity of base product
- Uses range of assembly techniques to customise
SOA Governance

- Shared Services Don’t Happen by Accident
- Policies to Mandate Sharing & Reuse
- Architected Services by Domain
- Use of Reference Architectures
- Use of Canonical Data Models (MDM)
- Use of Standard Technology Stacks
- SDLC Checkpoints for Services
- Funding Mechanisms

CBDI-SAE™ Governance Framework
Service Factory
Methods & Tooling
Service Oriented Application Modernization (SOAM) Provides Methods & Tools for the Service Factory

- Modularization
- Reuse
- Rapid assembly

- Rapid development
- User feedback
- Continuous testing

- Maximum leverage of legacy assets
- Reduced risk
- Orderly retirement of legacy apps

- Rigorous specification
- Coordination
- Automated patterns

Service Oriented Architecture
Model-Driven Development
Agile Methods
Portfolio Transition Engineering
SOA: Domain-Based Service Architecture
Patent Application Management (partial)
Model-Driven Development (MDD) Provides Accelerators

Software delivery approach where specification models and other abstract artifacts are created to describe the structure and behavior of a system or module. Models are either executed directly or transformed into implementation code or environment control.
Why Model-Driven?

- **Productivity**
  - Increase output (e.g., function points)
  - Reduce time & cost of development

- **Coordination**
  - Across multiple (parallel) development teams
  - Replicate modifications across modules
  - Rigorous specifications

- **Application of Patterns**
  - Encapsulate knowledge as reusable assets
  - Automate the application of patterns

- **Traceability & Reconstruction**
  - Easily identify impacted components when requirements change
  - Document & preserve design decisions
MDD: Model Transformation and Code Generation

- Model Design
  - UML Model

- Validation
  - Model Parser

- Transformation Engine
  - Cartridges
    - Java Cartridge
    - SOAP Cartridge
    - XML Schema Cartridge
    - XML DB Cartridge
    - REST CXF Cartridge
    - Backbone.js Cartridge

- Code
  - .java
  - .wsdl
  - .xsd
  - .xqy
  - .java
  - .js
Agile Methods: Scrum

Customer Requirements

Product Backlog

Sprint Backlog

Potentially Shippable Product Increment

Feedback

24 hours

2-4 weeks
MDD for Agile life cycle

Knowledge Discovery

Enterprise and Solution Architecture

User Stories, Use Cases

Business Improvement

Reference Model

UML Models

Iteration and redeployment

Cartridge Development

Code and Schema Generation

Test, Configuration and Deployment

Code generation enables rapid and frequent iteration and evolution with integrity without the cost/time overhead
Service Tooling for CMS/FFE Factory
Service Factory: Summary of Results

- Implementation of services based on architecture
- Product Line approach ensures consistency
- Facilitates Twin-Track provisioning
- Practical implementation of service policies (governance)
  - Semantics
  - Technology
  - Reuse
- Automates & accelerates provisioning process
- Enables enterprise scale production
Thank You.

Questions?

Dave Mayo
dmayo@everware-cbdi.com

www.everware-cbdi.com