Topic: Frameworks

Date: May 2012

Author: Lawrence Wilkes

Service Specification Reference Framework. An Overview

This research note explores aspects of a reference model for service specification, outlining several important concepts.

Introduction

You might be tempted to comment, “Reference Models are for academics and the birds!” And I would have some sympathy. It’s true that a huge amount of hot air gets expended on the topic in industry standards bodies, and sometimes you look at these models and wonder how anyone could come up with something so complex, just to solve what is really a fairly simple problem. Did someone mention over engineering?

But the core of successful service specification is actually getting a sensible reference model that allows everyone to understand what they are doing!

Elements of a Reference Model described in this report include

- A **meta model**. A meta model defines the rules for building a model of a business or software -- or anything else for that matter. The meta model for service specification is vital because it assists architects to define the meta data that needs to be collected in the specification, and establish consistent terminology
- **Life cycles.** To define the various life cycle states a service specification will pass through, and to map the specific content required and the policies that apply at each state.
- **Classification systems.** Establishing different types of service will allow specialization of the service specification approach for each type.
- **Patterns.** That may be applied to improve productivity and consistency.

Meta Model Concepts

Figure 1 provides an overview of some of the key service specification elements that should be defined in the Reference Model. It represents a gross simplification of the CBDI-SAE Meta Model for SOA[i].

- A **Service Automation Unit** provides one or more **Services**, which provides one or more **Service Interfaces**. Both the Service and the Service Interface can be understood as providing one or more **Service Operations**.
- A **Service** is a Capability – the ability to do something, which is required by a **Service Consumer**.
- The capabilities of the Service are accessed via a **Service Interface**.
- The **Service Operations** provide individual **Behavior** – abstractions of a single ‘ability’ provided by the Service.
- A **Service Automation Unit** (SAU) is a collection of one or more software module or other artifacts that automates or ‘provides’ the capabilities offered by a Service.

The separation of Service from the SAU is essential to delivering the ‘loose coupling’ promise of SOA. Service Consumers and Providers contract to the capability provided by the Service, not a specific software implementation of that capability as this may change over time.

To gain a deeper understanding of these concepts we recommend reading the meta model report[ii] and a worked example[iii] available on the website.

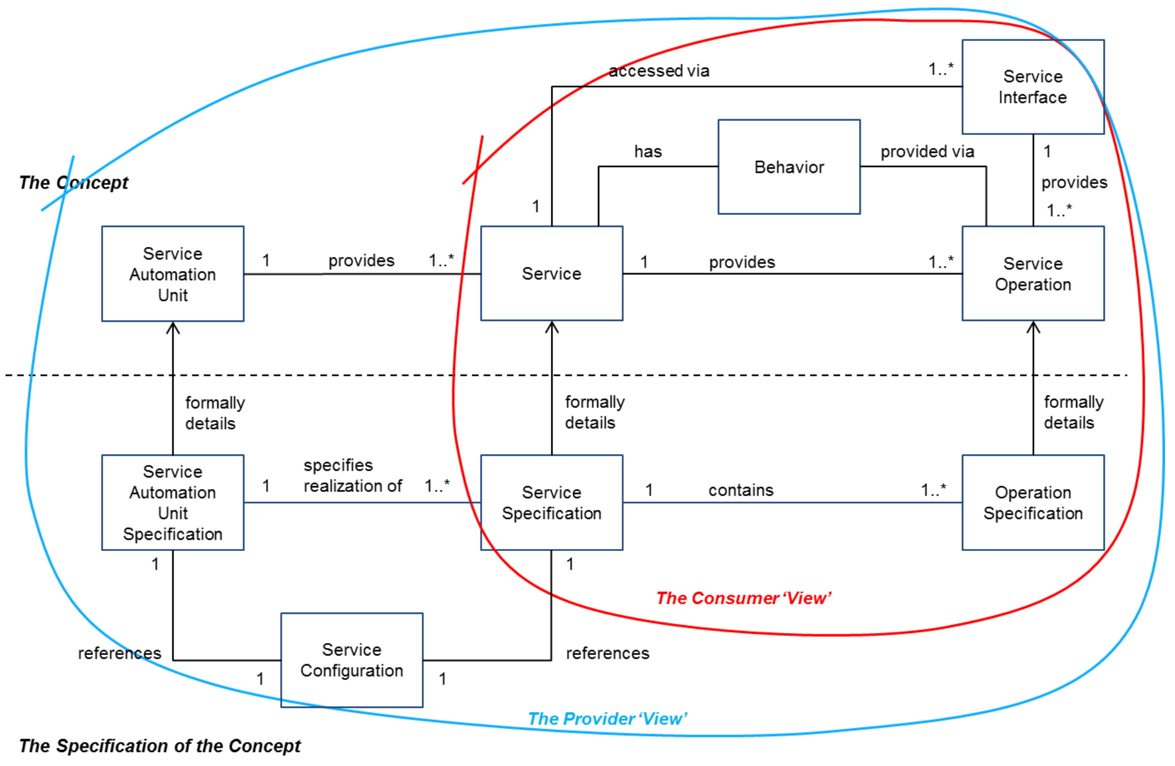

Figure 1 – Service Specification Concepts

Specification Artifacts

Each of these concepts has a corresponding specification artifact. The specification details the corresponding concept.

The **Service and Operation Specifications** provide the Service Consumer's view of the Service. The Service Consumer should only be interested in the 'external' view of the Service – *what* it does, and how it is used – and not the 'internal' view – how it is implemented.

Whereas the Service Provider's view also includes the **SAU Specification**. This details *how* the Service is implemented. The job of the developer therefore is to provide an implementation that conforms to both the Service Specification and the SAU Specification.

The reference model should identify a deliverable template for each of the specification types.

A detailed Service Specification Template[iv] is provided on the website. A detailed SAU Template[v] was published in the CBDI Journal and is available to Knowledgebase subscribers.

I also recommend reading Service Contracts in the Service Oriented Process[vi] available on the website, as this provides a further breakdown of the content of the specifications and explores their relationship.

Versions and Configurations

Specifications are versioned. Hence, 'version 2.0' of a Service, really means version 2.0 as defined by a versioned *specification* of that Service. That is, the 'notional' concept of the Service doesn't change – e.g. the Customer Service still manages customer information - but the detailed behavior provided does change and so hence also the specification changes to reflect that. For example, an operation signature may change, or a new operation may be provided.

It is that versioned specification of the service that the Service Consumers and Providers actually contract to. Therefore a SAU should also realize a specific versioned Service Specification.

Consequently it is useful to deal in **Service Configurations**[vii]. That is, a configuration that reflects a specific version of a SAU specification that realizes a specific version of a Service Specification. (Note: You won't find Service Configuration in the current CBDI-SAE meta model)

Service Classification

Service classification is an important governance mechanism, as policy assertions can then be geared to specific service types.

The Reference Model should identify and define the permitted classification mechanisms and permitted values. See Appendix A4 of the CBDI Meta Model for SOA V2[viii] for some example Service Classifications and values.

In terms of Service Specification, the classification of different service types can be used to determine

1. The specific meta data required. For example process-centric services will require process or interaction models as part of the specification, whereas information-centric services may not.
2. The level of completeness required. For example, core differentiating capabilities that require custom development will require completion to a greater level of detail than common utility or context capabilities that are planned to be acquired 'off the shelf'.

As this implies, useful classification mechanisms with regard to specification are

- **Architecture Layer.** A Service is classified according to the architectural layer it is placed in the Service Architecture. Classification by layers enables Providers and Consumers to determine exactly what the purpose and role of a Service is in the overall Service Architecture.

  Example layers include Process Services, Core Business (or entity) Services, Utility Services, Data or Information Services. See Appendix A1 of the CBDI Meta Model for SOA V2 for further detail and definitions of example service architecture layers.

- **Commoditization Level.** Services are classified according to the level of commoditization that is judged by the organization. Commoditization reflects the importance of a capability to the business. Normally used to determine appropriate Service sourcing. The sourcing approach is matched to the commodization level. For example, resources are not wasted building Services in-house when a capability is seen by the business as Standardized and hence the Services to support that capability are usually available "off the shelf"

  Examples include Core Services that differentiate you in the market place, or Context Services that can be standardized.

Service Automation Unit Classification

Similarly SAUs can be classified. We deliberately chose the term 'Automation Unit' as a way of avoiding any connotations of using a term like 'component' which might be interpreted by some as implying a specific style of implementation.

However, the SAU can still be classified as a 'component' if that does accurately reflect the style or pattern of implementation. Other implementation pattern examples might include 'script', 'external', or 'package'.

Design patterns can also be used to classify the SAU. Design patterns include 'façade' or 'wrapper'.

Consequently, the specification approach and content can be tailored to each pattern.

Classifying SAUs by using patterns is a useful mechanism as it quickly conveys to the reader what the nature or style of the SAU.

Extensive coverage of this is covered in a CBDI Journal report[v] and is available to Knowledgebase subscribers.

Service Life Cycle

Services don't miraculously appear in the published state in a Service Registry. Rather they go through a complex life cycle spanning planning to retirement.

Hence some properties of the Service Specification may manifest themselves at different life cycle states. For example, the Service Definition Language File (e.g. WSDL) may only be delivered once the Service is *Provisioned*. Similarly, the actual Service Endpoint where it is deployed at runtime won't be delivered until it is *Operational* (and may change dynamically)

Consequently, as shown in Table 1 it should be understood that the Service Specification is developed iteratively

- First as a 'requirements' document, that is used to drive Provisioning
- Then second as a 'provisioned' document that reflects what has actually been built or acquired
- Some of the more 'technical' elements may not be detailed as part of 'requirements', but only added once 'provisioned'
- Additional Specified Operations may be added over time as new requirements surface

State	**Deliverables**	**Comments**
Planned	Service Description Service identified in - Service Portfolio Plan - Service Specification Architecture (high level)	Essentially a 'to-be' or requirement. Specification is just an outline description
Specified	Service Specification (as required) Service identified in - Service Specification Architecture (detailed)	Requirement. Specification reflects ideal or desired Service.
Provisioned	Service Specification (as provided)	Actual. Service Specification can reflect actual or provisioned Service.
Certified	Certificate	May demonstrate compliance with industry standards
Published	Design-time Directory/Catalog entry Service Definition Language File (SDLF)	Catalog structure should reflect Service Specification meta data Directory may be more limited set of meta data. e.g. WSDL entry SDLF reflects interface technology. May use WSDL if binding available.
Operational	Run-time Directory entry	Directory contains minimum set of meta data to support run-time endpoint resolution. e.g. WSDL entry
Retired	Updated Directory/Catalog entries	Archived Service Specification may indicate replacement or alternative

Table 1 – Service Lifecycle

An extensive Service Life Cycle has been published in the CBDI Journal[ix] and is available to Knowledgebase subscribers.

Service Automation Unit Life Cycle

Like any other artifact, the SAU also has a life cycle as shown in Table 2.

Again, the specification will evolve in line with the life cycle state.

State	Deliverables	Comments
Planned	SAU identified in Service Implementation Architecture SAU Description	Essentially a 'to-be' or requirement. Specification is just an outline description
Specified	SAU Specification	Requirement. Specification reflects ideal or desired SAU.
Designed	SAU Internal Architecture	Documents the component and service architecture *within* the SAU Primarily where the SAU is to be developed, not acquired.
Provisioned	SAU Implementation (Software Executables, etc.)	External provisioning decisions may pre-determine SAU packaging. Actual. SAU Specification can reflect actual or provisioned SAU.
Retired		Retirement of the SAU is not necessarily in line with the life cycle of the Service.

Table 2 – Service Automation Unit Life Cycle

Again this detailed in the CBDI Journal[v], available to Knowledgebase subscribers.

[i] CBDI-SAE Meta Model for SOA Version 3 Specification http://everware-cbdi.com/mm-v3
[ii] CBDI-SAE Meta Model for SOA Version 3 Draft Report http://everware-cbdi.com/index.php?cID=21&cType=document
[iii] Update to the Example Model Based on V3 of the CBDI-SAE Meta Model for SOA http://everware-cbdi.com/index.php?cID=20&cType=document
[iv] Rich Service Specification Template http://everware-cbdi.com/rss-template
[v] Service Implementation Architecture and Automation Unit Specification. CBDI Journal, June 2009
[vi] Service Contracts in the Service Oriented Process. CBDI Journal, May 2008. http://everware-cbdi.com/index.php?cID=31&cType=document
[vii] See Service Life Cycle Configuration Management. CBDI Journal, Jan 2009.

[viii] CBDI-SAE Meta Model for SOA Version 2 Specification http://everware-cbdi.com/index.php?cID=16&cType=document
[ix] The Service Life Cycle. CBDI Journal, Nov 2005.