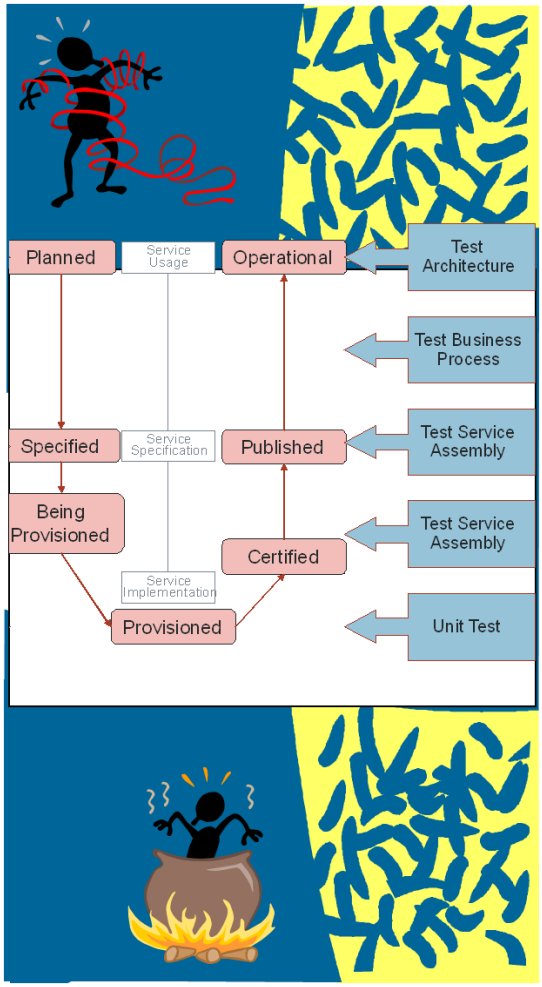
CBDI Journal



SOA Process Report

Establishing the SOA Reference Architecture Framework

In a collaborative, shared, reusable SOA world the reference framework provides a common backplane for federated service and solution development and assembly. Without a consistent framework, service orientation will remain an interesting concept but deliver suboptimal business value. With an appropriate reference framework the work of individual projects, programs divisions and partners will be coordinated with just enough formality to ensure that the many moving parts can fit together when and were needed. This article provides a process structure for the creation and evolution of the model and architecture elements of the SOA Reference Framework.

By David Sprott

Originally published in the May 2008 edition of the CBDI Journal

Independent Guidance *for* Service Architecture and Engineering

SOA Process Report: Establishing the SOA Reference Architecture Framework

In a collaborative, shared, reusable SOA world the reference framework provides a common backplane for federated service and solution development and assembly. Without a consistent framework, service orientation will remain an interesting concept but deliver suboptimal business value. With an appropriate reference framework the work of individual projects, programs divisions and partners will be coordinated with just enough formality to ensure that the many moving parts can fit together when and were needed. This article provides a process structure for the creation and evolution of the model and architecture elements of the SOA Reference Framework.

By David Sprott

Introduction

In the past the concept of the architecture framework was a way to share good experience in organizing and planning. Well known architecture frameworks such as Zachman and TOGAF provided useful perspectives that allowed architects to develop and communicate structured approaches and plan strategy.

In the SOA world the requirement for reference architecture is very different requiring higher levels of rigor and precision. To deliver inherently loose coupled, agile applications and infrastructure, key aspects of architecture need to be implemented in a consistent manner. Whilst industry standards such as SOAP, WSDL, WS-Security etc provide a useful basis for consistency these de jure standards are simply one layer focused on interoperability in a much broader set of enterprise policies and standards.

An SOA Framework is needed to:

- Nail down those things that need to be common
- Create consistency where needed
- Leave individual projects to do DIY where appropriate
- Provide a structured approach to managing standards, policies, patterns in order to deliver on SOA objectives

CBDI has described a reference framework for SOA shown in Figure 1. Previous CBDI Journal reports have further detailed the process elements of the framework[1], the model elements[2] and the architecture elements[3].

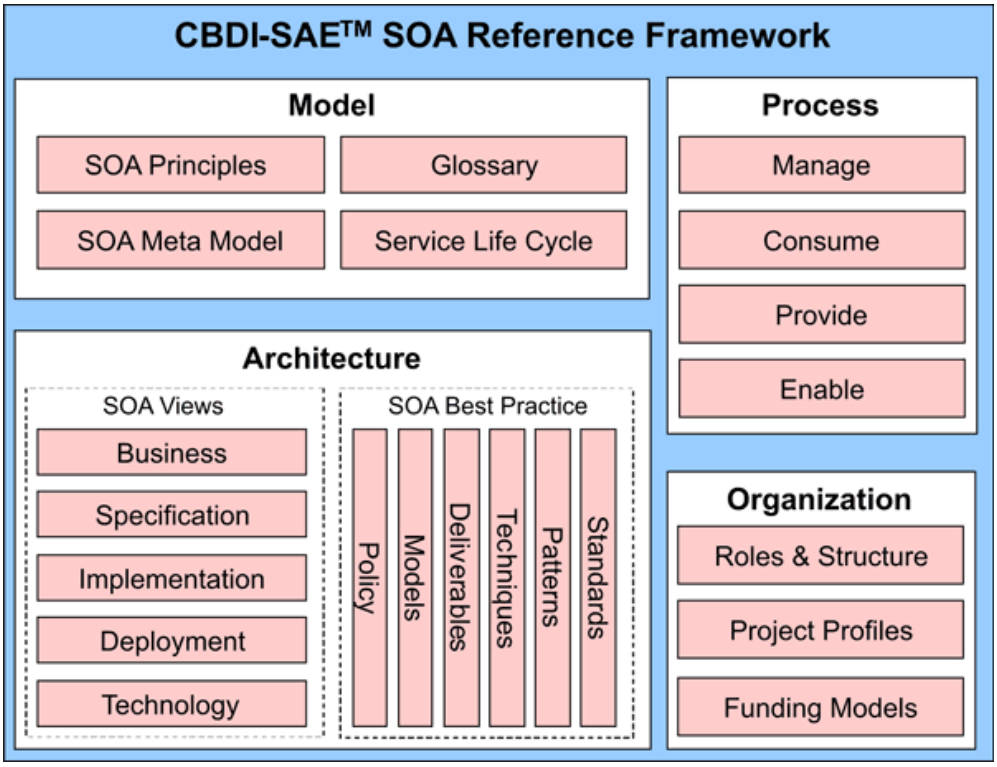


Figure 1 – The CBDI SAETM SOA Reference Framework

The CBDI-SAE™ SOA Process describes the Discipline of Service Oriented Architecture & Design with a simplified context diagram for this shown in Figure 2. This identifies a workflow covering the activities of assessing and Evolving SOA Reference Framework.

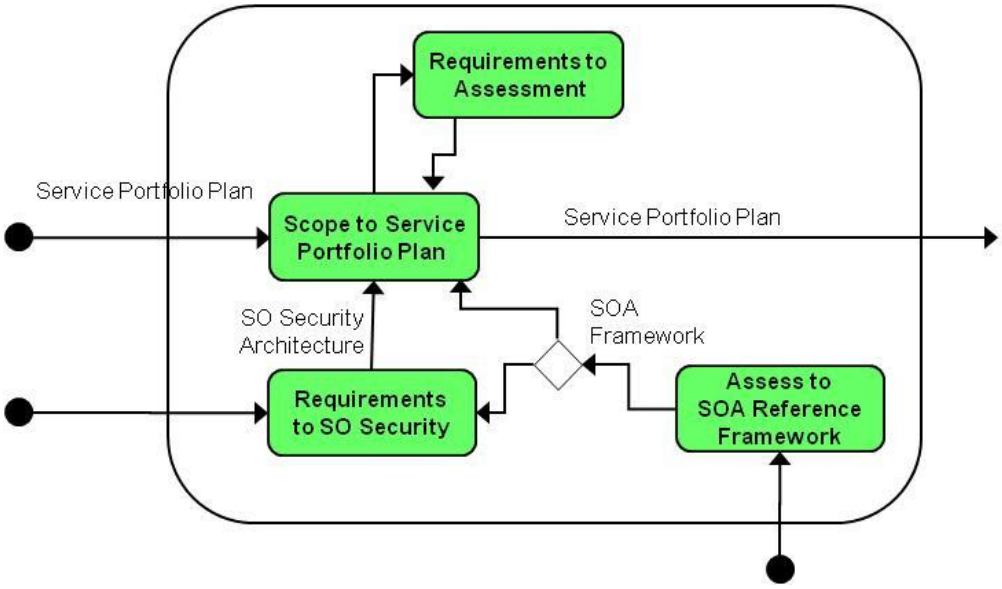


Figure 2 – Simplified Context Diagram for the Service Oriented Architecture & Design Discipline

This report focuses on a description of the activities required to establish the architecture and model aspects of the framework. It aims to answer the question - what is the task structure to establish a fit for purpose SOA Architecture Framework?

Figure 3 illustrates a first level decomposition of the Assess to SOA Reference Framework workflow. The tasks commence with assessing the requirements for the framework, because a) it is likely most enterprises will already have existing frameworks and b) the creation of the SOA Architecture Framework is going to be an evolutionary process over several phases of the Adoption Roadmap.

Define the Model is a crucial foundational activity that establishes the vocabulary and things that need to be managed – the assets.

The Define View Details task then populates the details of layers, deliverables, policies, patterns etc together with parallel tasks that define the contract types and architectural assets.

Finally there is a task to integrate the SOA Framework with other existing or planned frameworks because SOA is not a standalone concept.

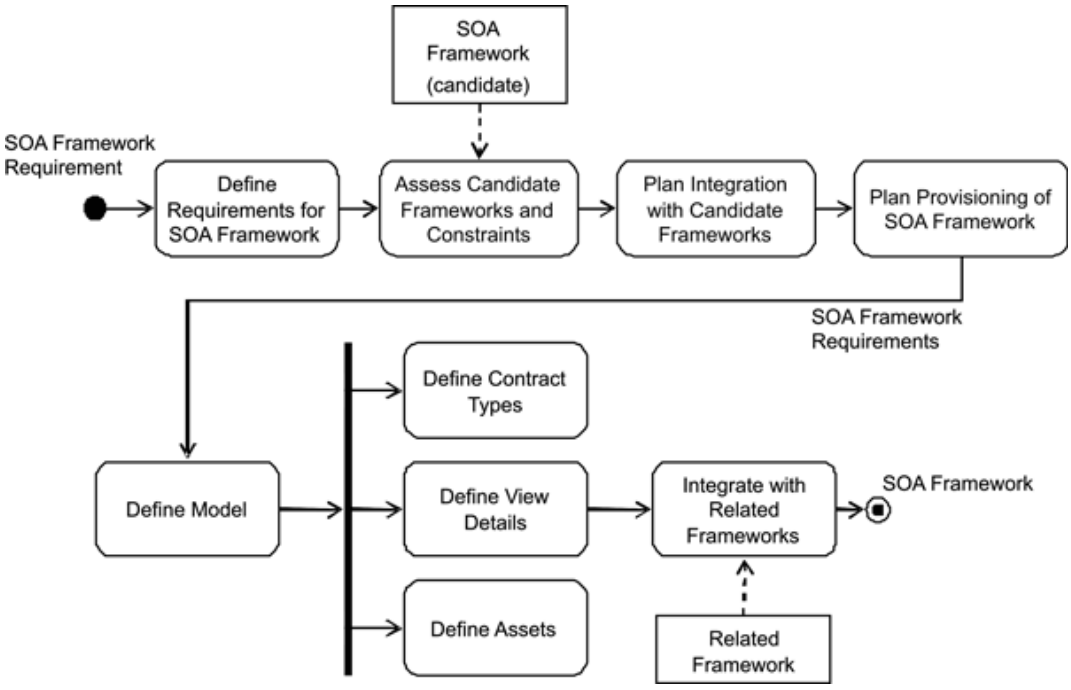


Figure 3 – Simplified Activity Diagram for Assess to SOA Reference Framework

SOA Framework Assessment and Requirements

It is unlikely that any enterprise will need to invent a green field SOA framework. There are various framework initiatives that are in varying stages of maturity and have varying characteristics.

This first set of activities is documented as part of the Everware-CBDI SAE™ framework. It is quite likely that enterprises will wish to integrate with pre-existing frameworks, and also possibly coordinate with other SOA frameworks. These may also include fragments of de facto frameworks such as provided by a SOBA[4].

Steps	Definition	Requires	Produces
Define Requirements for SOA Framework	Establish basis for SOA framework taking into account organizational scope, objectives and constraints	Understanding of Objectives for SOA SOA Adoption Roadmap	SOA Framework Requirements
Assess candidate Frameworks and Constraints	Review status, quality and completion of frameworks.	Access to existing frameworks Relevant legislation	Framework review
Plan Integration with Existing Frameworks	Define how the SOA Framework will align, integrate, coexist and or supersede existing frameworks.	Details of Existing Frameworks usage and assets	Plan for Integration or Alignment Update of Adoption Roadmap Capability Plan
Plan provisioning of SOA Framework	Select/Acquire/Plan Customization of SOA Framework Establish sourcing of the SOA Framework and Plan customization for enterprise specific needs	SOA Framework Requirements Plan for Integration or Alignment with existing frameworks	Framework Customization Plan
Evolve/refine SOA Framework	Establish system for evolving and refining the SOA Framework	Initial Framework delivery	Ongoing refinement and improvement

Table 1 – Assess SOA Reference Framework Options Tasks

Define Requirements for SOA Framework

The place to start evolving a framework is to articulate the business value that may be derived from the SOA framework in order to support decisions relating to collaboration and sharing. This is essential in order to justify the investment. Fundamentally commonality is important first to facilitate shared services that deliver an appropriate level of business consistency and support for cross business processes. But equally there is an opportunity to break down silo behavior, where individual divisions and geographies all reinvent their own wheel and replicate skills that could easily be shared enterprise wide.

In principle an enterprise should strive to have one common SOA framework and manage divisional requirements on an exceptional basis so it is important to determine the organizational scope, identifying divisions, geographies, partners, related ecosystems that will need to have some level of alignment with some aspects of the framework.

Very large organizations or ecosystems, such as national governments or collaborations such as supply chains and partner networks, will inevitably have multiple frameworks that are aligned to some extent. In this case it will be necessary to identify those elements of the shared framework that should be shared.

Within an enterprise, items that really should be common should include the model, glossary. life cycle, asset classes, core policies that impact cross enterprise service usage, core aspects of contracts particularly service specification and SLA. Also templates should be standardized wherever possible in order to maximize organizational learning and reuse.

These items form the basis for a common vocabulary that establishes the foundations for an inherently agile enterprise – that can more easily respond to business and technology change because there is a shared language.

Assess Candidate Frameworks and Constraints

An organization may already use one or more frameworks and they need to be reviewed to assess:

- Quality and completeness of existing models and tooling
- Compliance with regulatory requirements

Plan Integration with Existing Frameworks

Existing architectural frameworks may already contain extensive details of existing assets and provide valuable input to the SOA Framework. Some elements of existing frameworks may be immediately useful, particularly data architectures and logical models, application asset registers and integration schema definitions. However there will usually be differences and gaps in the underlying meta model that will render the information less than complete. For example conventional integration points may be linked simply to an application, with no linkage to business process or purpose.

Existing frameworks may also exist because the organization uses packaged applications that already implement or are planning to implement a SOBA. These applications bring an implied SOA framework which will need alignment in areas such as:

- Layer policy to establish desired level of agility and reuse by layer.
- Cross application integration – requiring agreements on gateways between contract and profile formats and deployment environments
- Inter domain policy relating to treatment of dependency and change management

Phasing of integration with existing framework(s) should be aligned with requirements identified in the Adoption Roadmap phasing.

Plan Provisioning of SOA Framework

The selection, acquisition and or creation of the SOA Framework is a discrete task. It is likely that existing framework(s) can be utilized to a significant extent, but that an enterprise framework will be unique.

Selection criteria and drivers for uniqueness may include:

- Scope and coverage of business and or ecosystem
- Alignment with methodology
- Alignment with required meta model
- Independence from specific technologies
- Alignment with industry standards
- Alignment or integration with acquired frameworks (SOBAs)
- Alignment with existing frameworks

Most enterprises will acquire a significant proportion of the SOA Framework because it makes sense to avoid reinventing the wheel, and it's faster, cheaper and likely to reach mature more quickly. An acquisition of this nature is also likely to provide an injection of external knowledge that will accelerate the organization's capability and avoid repeating mistakes that others have made.

Some level of customization is however always going to be necessary and is likely to address:

- Customization of vocabulary to align with widely used current conventions
- Choice of layering approach
- Extension of meta data, particularly asset attributes
- Contract templates and schemas, often to integrate with offshore partners
- Specific definitions of deliverables

Evolve SOA Reference Framework

The first iteration and delivery of the SOA Framework is the most important because it establishes a baseline on which early projects will be based. However the first delivery will and should continue to evolve in the light of project experience.

The wise organization will create a social system that encourages a disparate, evangelical group of architects and designers to engage in the initial delivery, but once the first delivery is made, to then encourage a more representative group of architects to assist in refining on the basis of real experience.

The framework evolution should also be phased according to the needs identified in the Adoption Roadmap Plan.

- Early framework applicable to exploratory SOA project activity
- Need to align maturing of the framework with higher risk projects
- Opportunity to evolve practices to policy status and to progressively mature policy in line with experience

Define Model

Conventional enterprise architecture describes an information system in terms of structural properties of the system. The architecture identifies components, building blocks, standards, policies and products which form the basis for planning and guiding systems delivery.

Not surprisingly SOA introduces change to the structural properties. There are new and different building blocks, standards etc. These don't necessarily replace the existing properties, mostly they complement and extend. However there are also areas where fundamental differences apply, for example in areas such as scoping and applicability, security models, reuse policies and so on, and we need a new model that combines the old and new in a cohesive whole.

Steps	Definition	Requires	Produces
Define Principles and Architectural Style	Establish Architectural Style appropriate to the enterprise	Objectives for SOA (outcome of Adoption Roadmap)	Definition of SOA for an enterprise Guidelines or policies relating to principles
Define Meta Model	Establish integrated model of meta classes and attributes to manage the life cycle of a service and its dependencies.	Industry model(s) Understanding of Objectives for SOA (from SOA Adoption Roadmap) Details of Existing Frameworks usage and assets (implied or actual meta model)	Coherent meta model for SOA in an enterprise Glossary of Terms
Define Life Cycle	Establish service life cycle states and transition conditions	SOA Meta Model Asset Management Strategy	Defined Life Cycle States State transition conditions

Table 2 – Define Model Tasks

Define Principles and Architectural Style

SOA architectural principles have much broader applicability than the conventional IT architectures that we are more familiar with. For example CICS, Client/Server, J2EE, .NET have quite constrained purpose, and they can be acquired as platforms and tools. But enterprises acquiring an ESB discover very rapidly that what they have bought is a technology backplane with no instructions on how to deliver the business benefits.

SOA principles are not inherent to any technology, product, platform or tool. They require enterprise specific architectural and engineering thinking to utilize them.

Most projects will conform with a subset of the core SOA principles. Most are attempting to address some level of loose coupling because this is the obvious place that agility may

be achieved. However the extent to which the loose coupling is built into the business design is likely to be more variable. An engineered approach to service architecture would review the broader range of principles and patterns to develop a practical design that delivers a level of agility that is fit for purpose and makes economic sense. Considerations may include:

- Loose Coupling
- Standardization
- Abstraction
- Composability
- Modularity
- Virtualization

The definition of principles provides an important foundation for documenting patterns. If principles provide a theoretical requirement for architectural style, patterns provide the organizational (and possibly industry) experience to show how the principles can be complied with.

Define 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 SOA is vital because it assists architects to:

- define the SOA concepts
- establish consistent terminology
- facilitate elimination of redundant concepts
- define the attributes and properties of SOA concepts
- define the relationships between concepts which in turn define dependencies
- establish integrated tool support

The CBDI-SAE Meta Model for SOA provides a comprehensive starting point for this, and is likely to be sufficient for many organization's needs.

Define Life Cycle

An understanding of the service lifecycle helps an organization to better organize the provisioning and operation of software services.

The software service is a formal asset, which is quite separate from the underlying software. It must therefore be traceable, manageable and governed from the time it is planned through to retirement.

The life cycle requirements for an enterprise should ideally be standardized in order to have a consistent approach. It also assists in managing the relationships between providers and consumers as they can share a common view of the asset and its life cycle state.

Each enterprise therefore needs to assess the life cycle states that need to be tracked and managed, and the conditions that trigger state transition. The resulting model will be an

integral part of the delivery life cycle and the asset tracking and management environment

See the Service Life Cycle[5] defined by CBDI for more detailed coverage of this

Define Assets

A Service is an asset. In the SOA meta model it is separate from the technical interface and supporting automation unit(s) and must be tracked, managed and versioned independently. In addition the relationships (dependencies) between services and relationships to other classes of asset are all extremely important to manage. In an SOA environment we typically have many more moving parts, which is what can potentially deliver greater agility. But it's therefore absolutely critical that asset inter-dependency is:

- subject to architectural decision making to create integrity units or clusters of assets that have a high level of internal cohesion and a low level of external dependency and
- managed throughout the life cycle so that architectural decisions are not compromised, reducing the planned agility characteristics.

To manage the SOA therefore we require a framework that defines and manages assets throughout the entire life cycle. In establishing the SOA Framework we need to:

- Define asset types and schemas (services, automation units, components, patterns, code frameworks)
- Define Publishing Policies (publishing profile, asset classification)
- Define Version and Release Management Policies

Steps	Definition	Requires	Produces
Define asset types and schemas	Establish asset management environment	SOA Meta Model	Defined Schemas in Asset Repository Pattern Templates
Define Publishing Policies	Define what, when and how assets are published	Adoption Roadmap strategy relating to use and reuse of defined types of asset	Asset publishing policy
Define Version and Release Management Policies	Define version and release management policy that establishes protocols for the required range of use and reuse patterns allowing providers and consumers to establish trust relationships	Use and Reuse patterns to be employed Defined provider consumer relationships	Version and release policy that defines an orderly approach to change management that providers a trust framework for providers and consumers

Table 3 – Define Assets Tasks

Define asset types and schemas

Superficially it might be thought that SOA is simply about managing service assets. However at all stages of the maturity model enterprises will use the SOA architecture to classify, manage and use a wide range of related assets. To create an agile business it is insufficient simply to manage service reuse and upgrade. The entire range of supporting assets from components, automation units, SOBAs, schemas, patterns, executables, code blocks etc all need to be organized and managed so that are related to the SOA layering and dependency taxonomy.

Define Publishing Policies

In the early stages of enterprise adoption of SOA it is quite likely that discovery and use of services will be highly controlled. The idea of dynamic discovery will be seen as high risk and reuse of design time assets will be a major improvement for most organizations. More advanced forms of publishing, discovery and reuse are generally behaviors for mature organizations that can handle automation of SLA, scalability, support etc.

At the same time it will be important that a consistent approach is adopted in publishing, that all services are published, even if they are not intended by design for reuse. So while projects creating services may be very reluctant to publish details of work they only intend to be used locally to a project, there is real merit in making details of the work much more broadly available which may lead to unplanned business opportunity arising through collaboration.

Define Version and Release Management Policies

Many SOA initiatives have foundered on inadequate version and release management. The primary issue that must be addressed is how to support multiple concurrent users of a single service providing reasonable response to functional and non functional change requests, while minimizing the impact of change on service consumers.

Define View Details

One of the defining characteristics of any methodology is the structure used to capture the relevant aspects or perspectives of a system, whatever system that may be – business, information system, hardware, or what have you. The CBDI-SAE™ Reference Framework includes five views – Business, Specification, Implementation, Deployment, and Technology. These views comprise a consistent level of abstraction for deliverable artifacts that relate to distinct set of stakeholders. This provides an effective mechanism for grouping practices, deliverables, policies etc based on a particular stakeholder view or perspective.

The framework is organized by view, and within each view there are key architectural elements that together provide a coherent approach.

To establish the framework each of these architectural elements must be defined and guidance provided sufficient for the purpose of the specific enterprise, to guide or govern appropriate levels of consistency.

The tasks involved in defining the views are:

- Define Views
- Detail Layers
- Detail Reference Architecture Policies
- Detail Models
- Detail Patterns
- Detail Deliverables
- Detail Standards

Steps	Definition	Requires	Produces
Define Views	Define primary perspectives of the reference architecture	Knowledge of pre existing frameworks Current practice	Agreed view structure
Detail Architecture Layers	Define sequenced subdivision within the software architecture. The members of a layer (for example software services) play similar roles, are at the same level of abstraction, and should conform to all rules defined for that layer.	Motivation of architectural community to establish common perspectives that enable reuse	Common approach
Detail Reference Architecture Policies	Definition of strategies or directives for Reference Architecture. These are defined independently from how they are carried out. Some policies are mandatory while others are guidelines.	Adoption Roadmap capability requirement plan	Requirement for consistent architecture
Detail Models	An abstract depiction of a problem or solution. In the context of SAE, a model must contain objects defined by the SAE meta model; e.g Business Type Model. A model can optionally also be a deliverable.	Methodology foundation	Consistent set of models and deliverables
Detail Patterns	A structured description of generic problem and a recommended solution,	Defined principles	Reusable organizational experience and

Steps	Definition	Requires	Produces
	thus representing reusable best practice knowledge	Pattern template	knowledge
Detail Deliverables	Defined deliverables that are recommended or mandated to be delivered at specified project phase specifically to document the SOA	Chosen Methodology Chosen techniques	Defined Deliverables
Detail Standards	Requirement for use of de jure, de facto and enterprise specific standards	External guidance on industry standards status Adoption Roadmap standards capability plan by phase	Standards policies

Table 4 – Define Views Tasks

Define Views

There is a fairly high level of consensus in the industry about architectural views. Most architectural frameworks share a similar approach to creating stakeholder perspectives, however unsurprisingly there is wide divergence on naming.

In CBDI-SAE™ CBDI has diverged somewhat from convention by introducing a "specification" view. Superficially this appears to be similar if not identical to a logical view. However there are differences insofar as the specification view is implementation independent, but not necessarily design independent where it is essential for the consumer to be aware of design details.

So CBDI's guidance is to define the external consumer perspective which is logical plus any behaviors the consumer needs to be aware of in order to formalize the separation of concerns between the provider and consumer.

In defining the views considerations will be made relating to naming and consistency with existing frameworks and asset stores, in order to sensibly integrate with other initiatives.

Detail Architecture Layers

The CBDI-SAE™ methodology provides guidance on software architecture layering.

Other organizations also provide guidance in this area. CBDI suggest that layers should specialize behaviors that maximize opportunity for reuse of layer contents as well as provide commonality of patterns, policies etc

The purpose of a common approach to layering is to establish maximum opportunity for cross division/LOB reuse. If the layering is common, then firstly framework elements can be reused, but equally delivered instances of components, services etc. Equally if

different approaches to layering are adopted then many opportunities for reuse will probably be lost.

In our experience many organizations will customize these layers:

- Certain layers such as capability services may be simply omitted. Some customers report they believe the capability layer to be really important, but only once they are more mature in their use of SOA. Specifically this is likely to be appropriate at the Enterprise Stage of the Adoption Roadmap, when there is strong business involvement.
- Certain layers may be further decomposed to provide subclasses that share common patterns, policies, change management protocols. For example the Process Services Layer may be subdivided into Solution Specific Process Services and Common Process Services. The Core Business Layer may be subdivided into Master Data Management Services and others.

Detail Reference Architecture Policies

The purpose of the Reference Framework is to provide guidance on how key aspects of architecture should be implemented in a consistent manner in order to ensure the objectives of the SOA are met.

In practice certain aspects of the architecture will be mandatory within an enterprise and implemented in policy. Others will be recommendations for best practice and will be attributed as guidelines.

Examples of policies include:

- Architecture Views and Layers
- Usage Patterns
- Domains
- Asset descriptions
- Versioning and Release Management Protocol
- Contract templates

Examples of guidelines include:

- Layer Dependency
- Principles
- Contract completion

Detail Models

Create detailed guidance on models that may be used to establish architecture and deliverables. Examples include:

- Business Process Model
- Event Model
- Business Type Model
- Service Specification Dependency Diagram
- Service Information Model

Detail Patterns

Patterns are the primary way the principles will be implemented. If principles are the theoretical objective and style, patterns are the documentation of practical knowledge that is relevant to the problem.

In practice many patterns will be available in the general industry domain, but enterprises will customize and extend them to make them relevant to the enterprise task.

Patterns will evolve both individually and collectively. An individual pattern will mature as it is used on successive occasions. Collectively an organizations requirement for patterns will evolve in line with the Adoption Roadmap Phases.

- In the early stages simple patterns relating to service style and usage, integration and loose coupling may dominate.
- In the integration phase patterns relating to specification, security, performance, testing will become important
- In the Enterprise phase business relevant patterns will come into play

Define Contract Types

As discussed in the CBDI Report also published this month[6] most of the core SOA principles such as loose coupling, abstraction, virtualization etc depend upon the existence of a contract. We therefore need three different contracts that create the necessary level of formality needed to establish the layer separation that delivers inherent business agility. The three contact types needed are to:

- specify the service that will be provided to the consumer regardless of how the service is implemented
- specify constraints on how the service is to be implemented.
- specify the commercial terms and associated logistics that govern the way in which the service is provided and used.

Not surprisingly we have three closely related subtasks.

Steps	Definition	Requires	Produces
Define Service Specification	A thorough description of what a service does, which avoids defining how it is realized or deployed. This description includes operation behavior and service quality levels.	Mapping to project management process and service life cycle Policy or best practice guidelines on service specification requirements for different classes and planned usage of services	Service Specification template Detailed guidance for completion Role responsibilities

Steps	Definition	Requires	Produces
		Adoption Roadmap capability plan	
Specific Automation Unit Specification	An Automation Unit Specification describes the implementation of a single Service, several Services, or an Application. It is itself a specialized form of Versioned Specification. It consists of a collection of Deployable Artifacts.	Defined contract relationships to be used (on/offshore, inhouse, outsourced) Mapping to the project management process and service life cycle Policy on architectural control and governance over build parties Adoption Roadmap capability plan	Automation Unit Specification Template Detailed completion guidance Defined role responsibilities Mapping to build contracts
Specify SLA	A contract that a service provider and a consumer agree to, which defines the provider‘s and the consumer‘s obligations with respect to one or more Services.	Defined delivery patterns to be supported Default SLA Adoption Roadmap capability plan	SLA template Detailed completion guidance Defined role responsibilities

Table 5 – Define Contract Types Tasks

The three types of contract will evolve according to the Adoption Roadmap phase capability plan.

- Early Learning and Applied phase: Contracts are minimalist. Reuse is relatively low. SOA activity is primarily applied to project activity and contracts will be within project teams. Utility and independent service contracts will be relatively straightforward, although utility contracts will need to address potentially large constituencies of consumers. Automation Unit Specification is of prime importance to ensure architectural governance.
- Integrated phase: Contracts required to cover multi party use of shared services across silos replacing point to point integration contracts.

- Enterprise phase: Widespread reuse of business facing shared services requires contracts to be back to back with business SLA.

Integrate With Related Frameworks

As mentioned earlier, SOA is not a standalone concept. It is essential that there is integration with existing frameworks.

Steps	Definition	Requires	Produces
Integrate with Project Management	Integration of architecture and project management processes to ensure project activity delivers architectural integrity	Architectural deliverables and dependencies Reference architecture policies Standards policies Versioning and release management policies	Enhanced project management guidance
Integrate with EA Framework	Execute plan to integrate with existing and planned EA framework(s)	Plan for Integration or Alignment Update of Adoption Roadmap Capability Plan	Common architectural perspective
Integrate with IT Service Management Framework (ITIL or similar)	Versioning and release management policies and best practices	Reference architecture policies Standards policies Version and release policy that defines an orderly approach to change management that providers a trust framework for providers and consumers	Integrated management practices for services

Table 6 - Integrate With Related Frameworks Tasks

Related Processes and Tasks

- Implement Knowledgebase
- Implement Communications and Collaboration Systems
- Implement Registry and Repository
- Establish Patterns Library
- Create education materials and rollout
- Implement Management Information System

Summary

The SOA Reference Architecture will not emerge by accident from pilot or pathfinder projects. Larger enterprises need to take a structured approach to delivering the levels of architectural commonality and diversity that are appropriate for the organization at evolving states of maturity.

---

[1] The Service Oriented Process, CBDI Journal February 2007
[2] A Meta Model for Service Architecture & Engineering, CBDI Journal October 2006
[3] The Architecture Component of the SAE™ Reference Framework for SOA, CBDI Journal March 2007
[4] SOBA – Service Oriented Business Application (source Gartner Group)
[5] The Service Life Cycle, CBDI Journal Nov 2005
[6] Service Contracts in the Service Oriented Process, CBDI Journal May 2008

About CBDI

CBDI Forum is the Everware-CBDI research capability and portal providing independent guidance on best practice in service oriented architecture and application modernization.

Working with F5000 enterprises and governments the CBDI Research Team is progressively developing structured methodology and reference architecture for all aspects of SOA.

CBDI Products

The CBDI Journal is freely available to registered members. Published quarterly, it provides in-depth treatment of key practice issues for all roles and disciplines involved in planning, architecting, managing and delivering business solutions.

Visit www.cbdiforum.com to register.

Platinum subscription – A CBDI Forum subscription provides an enterprise or individual with access to the CBDI-SAE Knowledgebase for SOA delivering ongoing practice research, guidance materials, eLearning, tools, templates and other resources.

Everware-CBDI Services

At Everware-CBDI we enable large enterprises and governments to become more agile by modernizing their business systems. We have repeatable processes, resources, tools and knowledge-based products that enable enterprises to transform their current applications in an efficient, low risk manner, into an optimized service-based solutions portfolio that supports continuous, rapid and low cost evolution. Our consulting services range from providing practices and independent governance to architecture development, solution delivery and service engineering.

Contact

To find out more, and to discuss your requirements visit www.everware-cbdi.com or call

USA and Americas: 703-246-0000 or 888-383-7927 (USA)

Europe, Middle East, Africa, Asia, and Australasia: Telephone: +353 (0)28 38073 (Ireland)

www.everware-cbdi.com