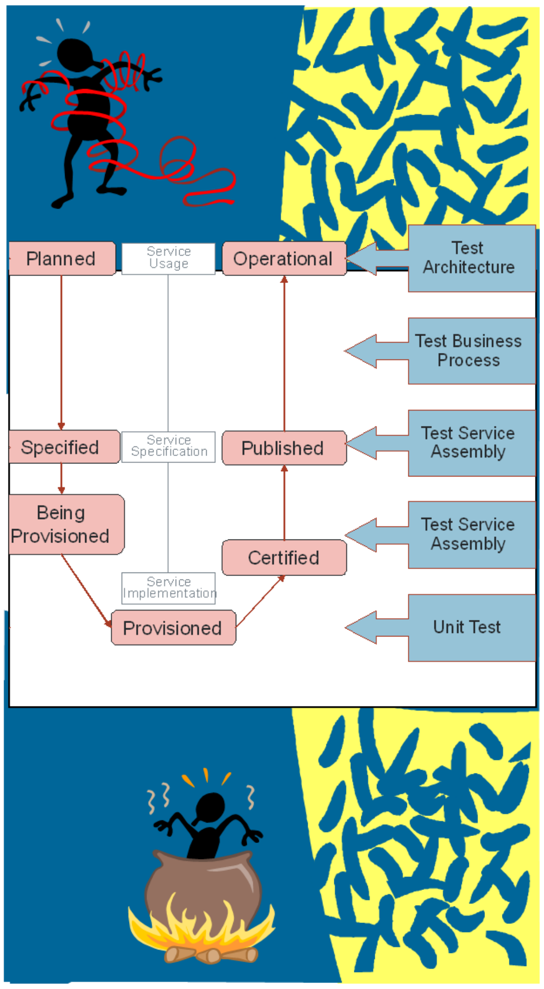
CBDI Journal



SOA Methodology Report

Service Contracts in the Service Oriented Process

At the heart of the service concept is conformance with the principle of the "contract based" capability. Whilst many SOA principles will be optional depending upon context, the use of service contracts is likely to be required, because a well formed service is likely to be the primary enabler of the agile business. At the same time it is important to use a pragmatic level of specification detail that is appropriate to the context of use. Our practice research to date has focused on the Rich Service Specification and the Service Level Agreement. In this report we extend that guidance to cover evolution of specification artifacts across the full service life cycle, in an integrated way, to include all the involved parties.

By Paul Allen

Originally published in the May 2008 edition of the CBDI Journal

Independent Guidance *for* Service Architecture and Engineering

SOA Methodology Report: Service Contracts in the Service Oriented Process

At the heart of the service concept is conformance with the principle of the “contract based” capability. Whilst many SOA principles will be optional depending upon context, the use of service contracts is likely to be required, because a well formed service is likely to be the primary enabler of the agile business. At the same time it is important to use a pragmatic level of specification detail that is appropriate to the context of use. Our practice research to date has focused on the Rich Service Specification and the Service Level Agreement. In this report we extend that guidance to cover evolution of specification artifacts across the full service life cycle, in an integrated way, to include all the involved parties.

By Paul Allen

Introduction

The SOA architectural style is fundamentally about separation; establishing smaller, separate units of capability that create a more agile application and infrastructure environment. Most of the core SOA principles such as loose coupling, abstraction, virtualization etc depend upon the existence of a contract.

The concept of service contract appears in various guises at different points in both the software process and the service lifecycle. Contractual mechanisms are needed 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.

In simplest terms each of these three types of contract relate to the specification, implementation and deployment views of a service. But this is a little simplistic and we need to define in detail the relationships between these different types of contract to ensure a coordinated contractual position between provider and consumers..

In addition there are timing, level of detail, and role related considerations around the different types of contract.

Timing considerations center on when different specification types are appropriate in the software process and service lifecycle. Level of detail considerations have to do with achieving an appropriate degree of rigor and depth in the specifications that supports the relevant project phase activity. For each type of contract we require a progressive development approach that provides just enough information to support planning with a reasonable level of confidence and stability, and a continuous development process that involves multiple parties in reaching agreements as details of the design, deployment and usage are finalized. In fact it sounds more than a little like any contractual agreement process.

Consequently it's essential that participant involvement in the contract development process is clearly defined and communicated. A service contract, by definition, implies it is a contract between parties. But are parties always necessary and who are these parties exactly? We need to be very clear about the roles and responsibilities involved.

Practical Requirements

The need for guidance on service contracts surfaces in terms of questions such as:

- Does every service require a Service Specification?
- How much detail do I need – can this be graduated somehow so I don't have to jump in at the deep end so to speak?
- Can a Service Specification be broken down into more manageable chunks, each with a different audience?
- Does the idea of specification apply only to services – what about other types of software and how about assets in general?
- How does Service Specification relate (if at all) to the world of SLAs and ITIL?
- What about the implementation design of a service – if it's not documented in a Service Specification, then where?

More broadly, there are a range of practical requirements surrounding the requirement for broader contractual mechanisms including:

- Clarification on the key concepts involved; in particular the differences and relationships between Service Specification, Automation Unit Specification and Service Level Agreement.
- Clarification on the building blocks of the service specification itself and on how these relate to other models and artifacts.
- Justification of the "overhead" of creating "all this documentation"; common questions are "Why do we have to create such a lot of documentation?" "We don't have time on our project for this level of detail."
- Guidance on using a weight of specification that is both appropriate to the task in hand, and not too onerous as to be unworkable. In particular, there is a need to find a way to "get to first base".
- Real world scenarios involving whether (and, if so, how) to document the following in the same or a similar way:
    - Existing services
    - Acquired services
    - Service-like software (e.g. not fully encapsulated, or exposing internals, but otherwise a service)
    - Legacy systems
    - And so on!

The Three Dimensions of Specification

Figure 1 shows the key concepts of service specification using a UML class diagram; see table 1 for definitions. Roles are clearly central to this picture and we include an overview below

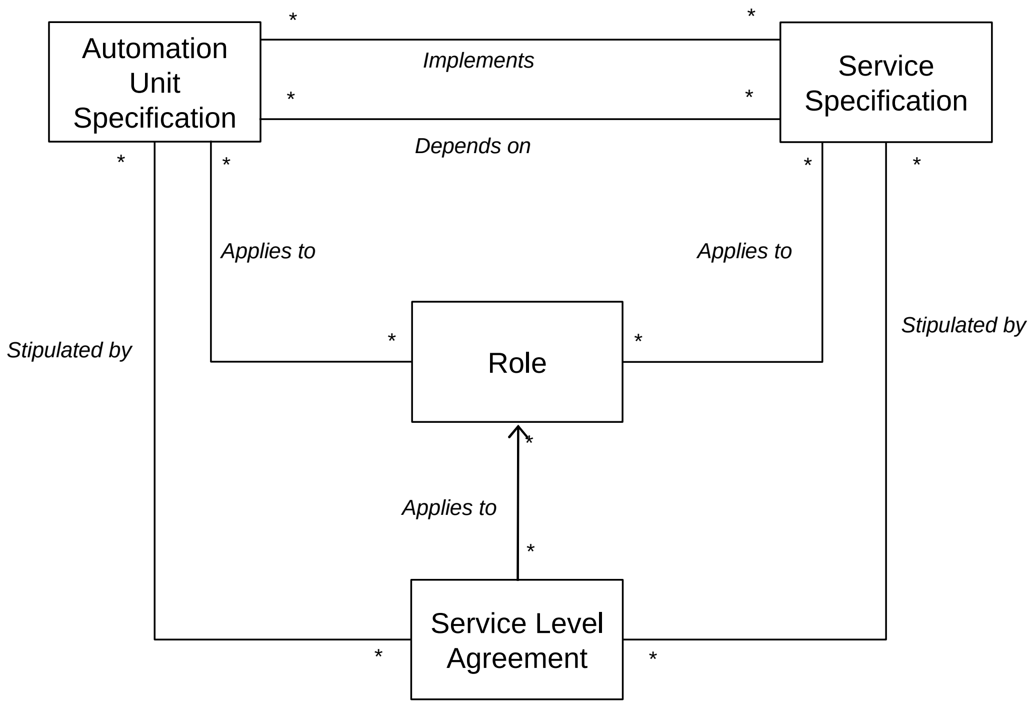


Figure 1 – Key Concepts involved in Specification of Services

Associated definitions are shown in table 1.

Concept Name	Definition
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. An Automation Unit can be decomposed into several distributed Automation Units are each hosted on a separate Node of a computing network. So an Automation Unit might also represent a part of the implementation of a Service, or several Services or an Application. *The full internal design of an Automation Unit, and its source code, are not represented.* *An Automation Unit may do very little; it might simply delegate processing to another Automation Unit.*
Role	A set of related skills and responsibilities, which a participant in the process may offer. A worker may typically offer several roles, and several participants will be able to perform the same role.
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.

Concept Name	Definition
	The Specification must supply all the information that a consumer of a deployed service needs to know. So when the Specification is complete, it must specify the Service's Endpoints.
Service Level Agreement (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. The SLA can refer to or subsume the Service Specifications of the Services which are the subject of the agreement. The SLA will state various measurable obligations, such as: the percentage of the time services will be available; the number of users that can be served simultaneously; specific performance benchmarks to which actual performance will be periodically compared; the schedule for notification in advance of network changes that may affect users; help desk response time for various classes of problems; dial-in access availability; the usage statistics that will be provided; penalties for non-conformance.

Table 1 – Definitions of Specification Concepts

Overall Applicability

All three types of specification will normally be required for all services that are implementing clear separation. Each enterprise will need to establish policy governing usage, for example all three specification types required for services:

- published on the ESB or
- used across project boundaries
- represent important business flexibility points, such as boundary of integrity unit such as test or deployment

In practice these guidelines may be further qualified by levels of completion.

An exception to this general guideline is Automation Unit Specifications are unlikely to be required where there is no means of control over the way in which the implementation of the software is designed, for example for consumers of "Software as a Service" (SaaS).

Clearly, it is important to weigh up the pros and cons carefully in each case; market context[1] is especially important when considering SLAs for example..

Certification of Services

The three dimensions of specification of services cannot operate in a vacuum. The quality of specifications, and more importantly the service itself must be certified according to SOA policy as illustrated below. In addition SOA policy provides a means of reusing common elements of specification, such as QoS level requirements, across services. As well as reducing redundancy SOA policy also provides a key governance mechanism, which is increasingly important as early low risk projects scale up to bigger higher risk ones; see my article last month for further details[2]

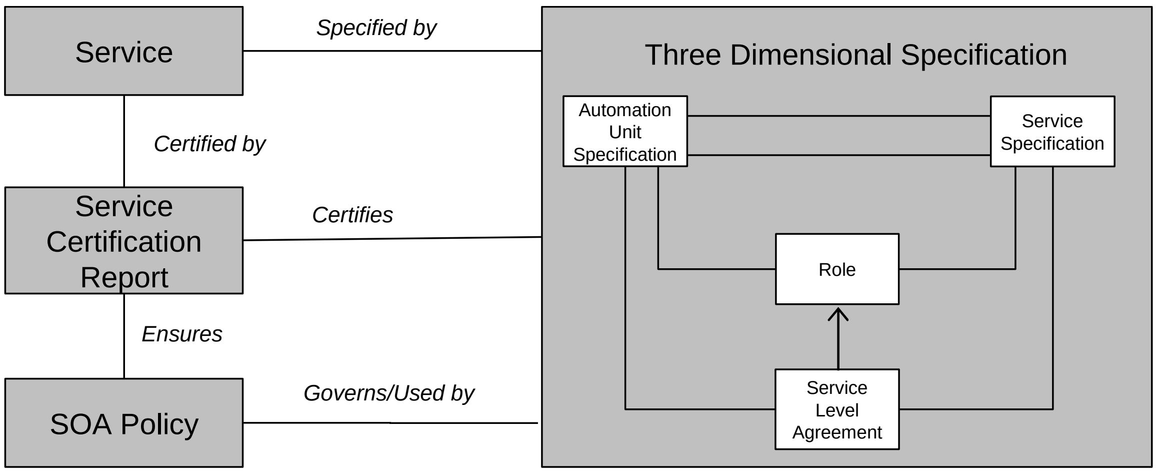


Figure 2 – The Concept of Service Certification Report in Context

Structuring the Artifacts used in Specifications

In this section we provide an overview of the elements involved in each type of service contract. Our main focus is to set out a structure that helps the practitioner produce "just enough" detail. At the same time it is important to have the option of producing the detail where it is really needed. Clearly it is important to introduce the various forms of service contract in a graduated fashion.

In the case of Service Specifications and Automation Unit Specifications we suggest an overall approach to service contracts that involves three tiers:

- Descriptions describe the basic properties of services.
- Lite Specifications omit key aspects of a specification template. For example the Service Information Model and some pre and post conditions may be optional for some Process Services. Similarly for algorithmic Utility Services.
  Similarly for Underlying Services that are assessed as stable details of behavior may be omitted, as the investment to create in not warranted.
- Specifications rigorously define the agreed service behaviour and information, and define measurable QoS requirements such that the services can be tested.

The level of detail, or the number of tiers used in a given case, should be appropriate for the type of service, the applicable process or SDLC stage, the service lifecycle stage and the context in which the specification is to be used. We will need to profile a service and make a quick call on these factors as described later in this article.

Then, having decided on an appropriate level of detail, the service contract needs to be developed involving the right audience, according to which roles are involved in the contract; again we will come back to that in a while.

Specifying Services

In CBDI-SAE^{TM} Service Descriptions are distinct from, but closely related to Service Specifications, as shown in figure 3 below. A Service Description includes the key properties of a service without going anywhere near the level of detail that a Service Specification would go to.

The Service Portfolio Plan includes a set of Service Descriptions. In principle the Service Description is a subset of the meta data that will eventually be delivered as a fully detailed Service Specification. However some of the properties will be forecasts, speculative or initial assessments. For example:

- Target Consumers that will only become clear as SLAs are developed
- Indicative Operations that will only be confirmed following detailed requirements analysis and design
- Specified dependencies that will be refined during detailed design

And some of the properties will change depending on life cycle state, such as:

- Technical Owner
- Providing Organization

Figure 3 provides an overview of the key elements involved. Detailed definitions of the concepts involved are available in earlier reports[3].

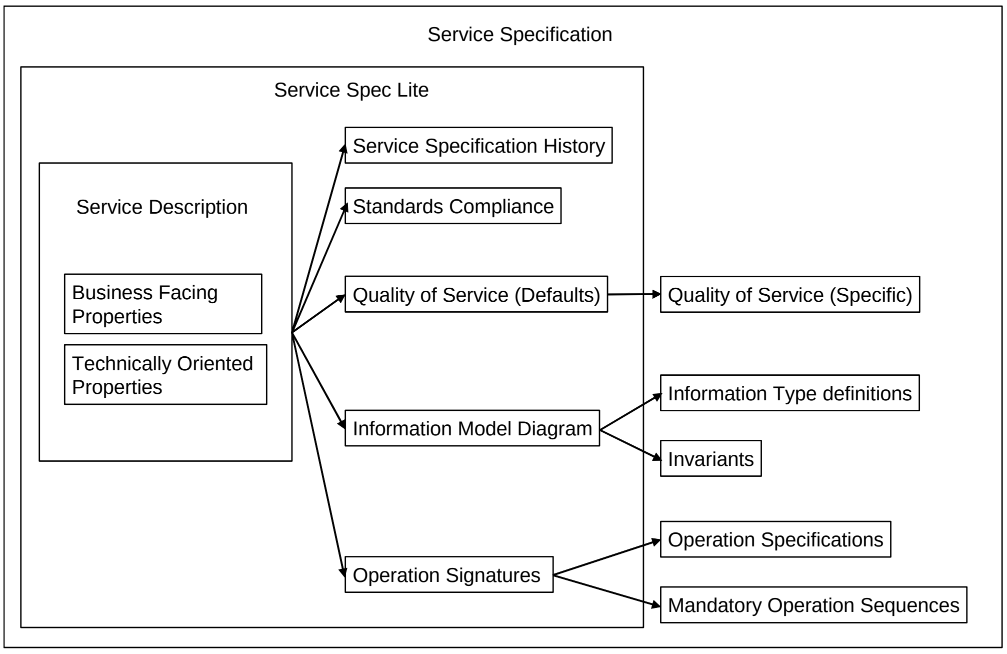

Service Specification
Service Spec Lite
Service Description
Business Facing Properties
Technically Oriented Properties
Service Specification History
Standards Compliance
Quality of Service (Defaults)
Quality of Service (Specific)
Information Type definitions
Information Model Diagram
Invariants
Operation Specifications
Operation Signatures
Mandatory Operation Sequences

Figure 3 – Elements of Service Specification

Specifying Automation Units

The provisioned services and their consumption by assembled application solutions will ultimately determine whether your SOA lives up to the business case expectations. This requires a level of formality over the contracting process with the building party. If conducted properly this naturally leads us towards subsequent stages where the high quality services are realized into concrete services and consumed into solutions within the ecosystem of the business.

In particular, the provisioner needs to exert some architecture and design control over the automation unit builder. For example, whether the build process is undertaken on or offshore, it's critical that issues of architectural integrity, replacement and or upgrade time and cost are determined.

Figure 4 provides an overview of the key elements involved. Detailed definitions of some of the concepts involved are available in Denzil Wasson's September 2006 report[4]. Note that an example of an SOA Policy Default might be Service Behavioral Design Policy; see relevant section on reusing SOA policy a little further on.

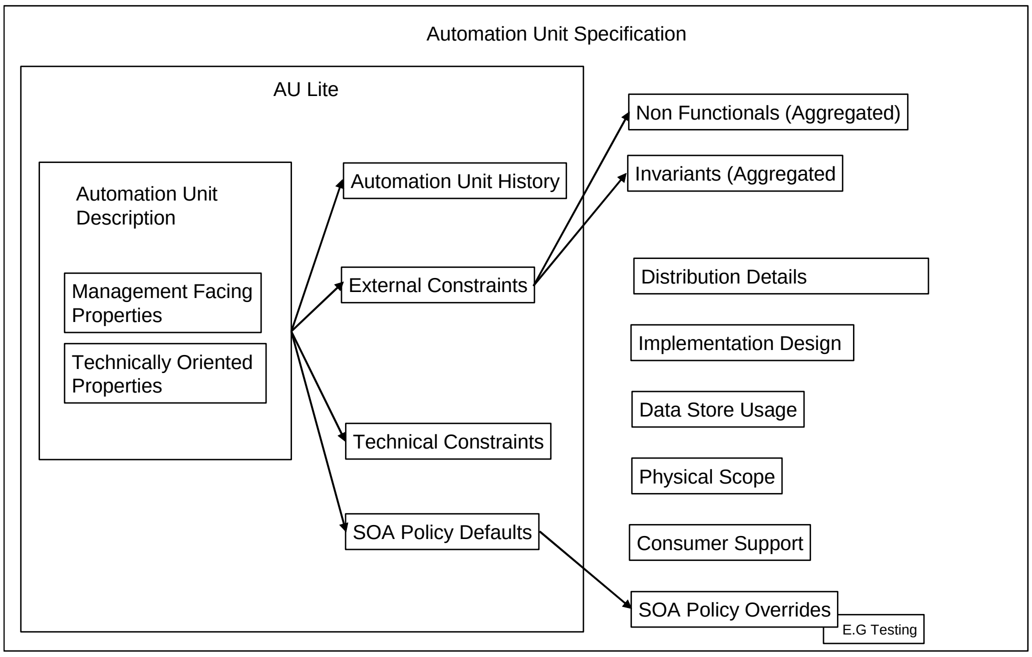


Figure 4 – Elements of Automation Unit Specification

Specifying SLAs

An SLA is an agreement between a customer and provider about what services are to be offered by the provider and to be used by the customer. It must specify the measurable levels of those services that the provider must achieve, and the terms of use that the customer must comply with. Notice that the SLA is a back to back agreement that covers obligations on both the provider and the customer and actually sits at the intersection of potentially many roles.

An SLA applies the service specification to a particular customer – provider pair (or set of pairs) in terms of a commercial and binding contractual agreement that governs the provision and the use of a service (or set of services). The service specification acts as a base role neutral specification and may be the subject of different SLAs involving different role participants. In this way the service specification supports reuse of specifications in different contexts, helping to address the problem of duplicated and reinvented information that is common in many organizations today.

The SLA also covers various forms of delivery patterns which may involve several parties and therefore multiple, potentially linked SLAs. For example:

- Single Full Lifecycle SLA: Single instance service reused by (one or more) consumers. Provider responsible for entire service certification and delivery at design and run time
- Multiple SLAs: Service provided for independent deployment by consumer. For example Service Oriented Business Application (SOBA) such as SAP or Oracle Fusion. SLA for maintenance and support with the SOBA developer; SLA with hoster for runtime operation.

- Separate Design and Runtime SLA: Service provided for discrete deployment by (one or more) consumers. Applicable where service operates on partitioned data and or where performance or trust requirements cannot be established on the single instance. Provider responsible for delivery of design time service and certification. Consumer responsible for runtime SLA. Support and maintenance provided by either provider or consumer.
- Separate provider and hoster. Separate SLAs for provision, design maintenance and runtime support. Possibility also for multiple certification reports.

Figure 5 provides an overview of the key elements involved. Detailed definitions of the concepts involved are available in my December 2006 report[5]. Note that "Subject" refers to the associated service or set of services that are the subject of the SLA.

It is also worth mentioning that we have taken a different approach to structuring the SLA than we did with the Service Specification and Automation Unit Specification, since the commercial details, quality levels and business levels are all likely to have distinct, though not entirely separate audiences. For example a business process analyst and business sponsor is likely to be interested in business targets addressed but not necessarily the enabling quality levels (hoster) and commercial details (sourcing and usage manager).

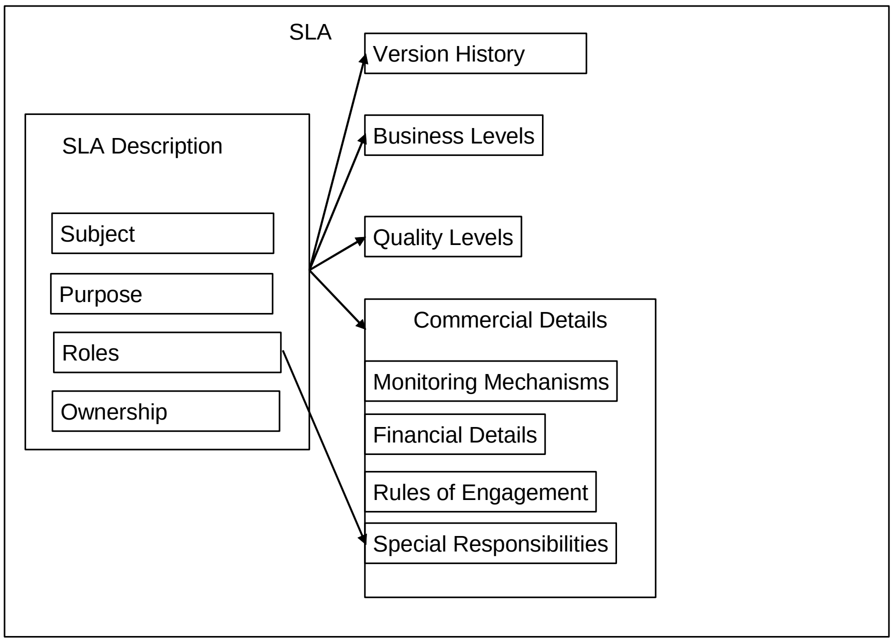


Figure 5 – Elements of the SLA

Certifying Services

A service in the "certified" state is not only deemed to meet its requirements as detailed in its Service Specification, Automation Unit Specification and SLA, it is also confirmed to be compliant with all associated standards and SOA policies. The service must be certified by provisioning specialists who are likely to be independent from the service implementers, such as developers and testing specialists. A separate certifier role is recommended for this purpose.

Service certification consists of two sets of interrelated activities:

- checking the consistency between the various artifacts involved in specifying the service; for example the QoS levels specified in the SLA should not be more stringent than those specified in the Service Specification and Automation Unit Specification.
- checking that the service itself conforms with SOA policy and with its Service Specification, Automation Unit Specification and SLA, for deployment and consumption in its intended runtime environment. The usage context is important here; see Richard Veryard's article SOA Testing[6] (page 18), as the service can only be certified for use in certain contexts, not all imaginable scenarios Note that the service being certified may already be in operation; for example an external service that the organization is subscribing to over the Internet.

The Service Certification report details the findings and recommendations arising out of certification of a service previously tested by the service implementers, and includes the following elements:.

- Level of certification (for example bronze, silver or gold)
- Specification Consistency Checks
- Conformance to SOA policies and standards; for example security, best practices, asset management.
- Certification Test Plans
- Certification Test Results

Reusing SOA Policy in Specification of Services, Automation Units and SLAs

At the top of this article we asked whether SOA and Service Specification make sense without each other. Well, from a practical point of view we contend that they don't. An SOA without a set of Service Specifications is simply a set of drawings without real substance.

Equally where information is common to several services a good SOA should provide a means of recording that information as policy. For example a quality attribute, such as performance, is likely to apply across the board to a whole set of services in an enterprise or domain. There needs to be a means of specifying the required performance levels as policy with an appropriate scope.

There are good practical reasons for this approach. For example commonly occurring information is created and maintained just the once. And more significantly a measure of architectural control is achieved over critical areas thus ensuring against software anarchy with developers creating disparate services.

Table 2 provides an example of SOA Service Specification policy, which provides a more formal means of controlling levels of detail in specification artifacts, than the simple Description-Specification idea described earlier

Attribute	Definition
Definition	The policy stipulates the format of a Service Specification in terms of a recommended Service Specification Template and indicates the kinds of service that require a Service Specification with expected level of detail.
Policy Strategy Area	Standardized Deliverables
Parent Policy Type	
Guidance	It is recommended initially to use the SAE™ Service Specification Template. As the company gains experience of SOA this template can be adjusted and extended to suit the organization's requirements. A Service Specification can potentially include a great deal of detailed information, which may well prove too onerous for this organization's level of SOA governance maturity. It is therefore important to take a view on which types of service require specification and to introduce the concept in graduated fashion. It is recommended that all core business services are specified using the Service Specification Template and that the following sections of the Service Specification Template are completed for each service requiring specification: • Service Description • Service Specification History • Quality of Service; note these are "default policies" as a starter in the absence of formal policies recorded within the Operational governance area. • Standards Compliance • Information Model Diagram • Operation Signatures • Further detail should be added as experience is gained in this area: • Quality of Service (Specific) • Information Type Definitions • Invariants • Operation Specifications • Mandatory Operation Sequences
Suggested Assertion Language	Use Service Specification Template.
Likely to be Automatable?	Yes

Table 2 – Service Specification Policy Example

In addition SOA policies play an important role with respect to the service contracts themselves. For example a SOA Design Policy such as Transaction Integrity could be inherited as a Default SOA Policy, unless specified otherwise, by all Automation Unit Specifications falling within the scope of the policy.

Relevant Artifacts in the Service Lifecycle

As a general principle expect to see increasing detail of artifacts used to describe and specify services as the service passes through its lifecycle states as shown in table 3 below. However, notice that detail does not increase at a constant rate.

Service State	Service Description	Service Specification	Automation Unit Description	Automation Unit Specification	Service Certification Report	SLA Description	Detailed SLA
Planned	Outline of Service Need		Outline of AU Need			SLA Patterns defined	
Specified		Comprehensive definition		Comprehensive definition	Defined test strategy		
Being Provisioned		Agreement with builder		Agreement with builder			Agreement with relationship owner
Provisioned							
Certified							
Published							
Operational							
Retired							

Table 3 – Generalized Applicability of Different Specifications by Service State

In particular the Specified state involves the Automation Unit Description and SLA Description in order to check for technical and commercial feasibility respectively.

This approach is also in sharp contrast to the traditional IT mindset in which development and operations tend to be isolated from one another. Not only is the service oriented software process distinctively iterative across the traditional software lifecycle stages, but SLAs, which tend to be treated as a purely operational concern in earlier approaches, assume significance much earlier in the piece.

Roles involved in Service Contracts

As a general guideline the Service Architects produce Service Descriptions (and Automation Unit Descriptions) whereas the Service Specification Designer produces Service Specifications. Service Descriptions help regarding the earlier issue of getting to first base. Figure 6 illustrates the point.

Typically the service architect will work collaboratively with the build party in developing the Service Specification. Similarly the service specification designer will work collaboratively with the build party in developing the Automation Unit Specification. Then, at a certain point in time they will be signed off and represent a contract between the parties. In the case of offshore of outsourced relationships, the contracts may well form attachments to a master contract. Other roles, such as Sourcing and Usage Manager and Certifier will come into play at various points. In addition, where an SLA is required, negotiation and agreement must take place under the stewardship of the service level manager. Ownership or custodianship of the various artifacts must also be established. Table 4 provides a generalized overview of the roles involved, showing responsibility, approval and review activities.

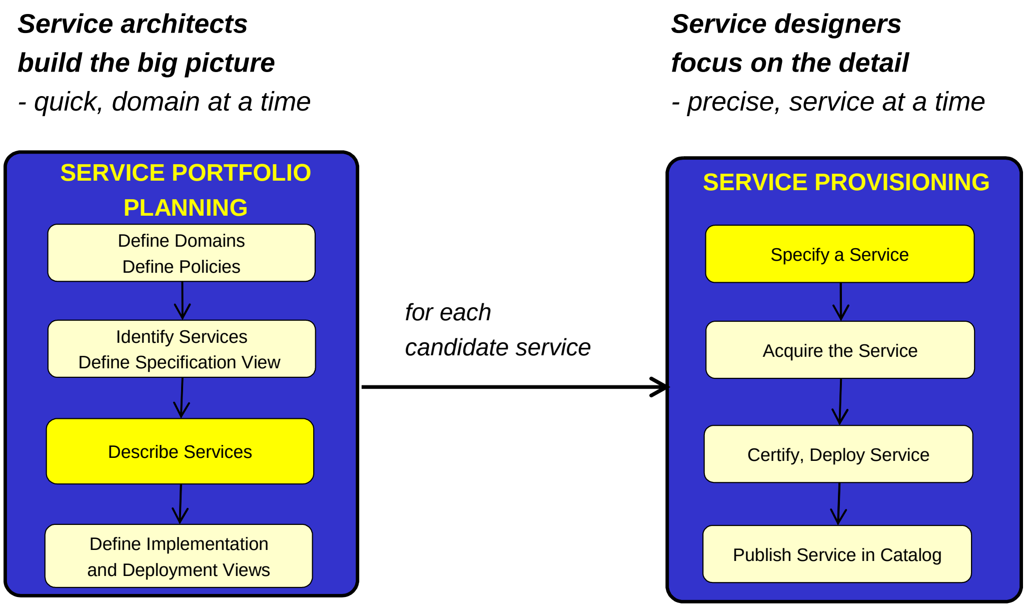


Figure 6 – From Service Description to Service Specification

.As alluded to earlier the responsibility characteristics of an SLA are potentially quite complex. The service level manager takes the central role but further roles may be involved as footnoted. Notice also that we are focusing on responsibility, not ownership. Sometimes ownership is associated with both responsibility and authority. In our view ownership centers on the latter. For example, a service level manager may be responsible for ensuring an SLA is acceptable to all parties, but does not own the service.

Role	**Service Description**	**Service Specification**	**Automation Unit Description**	**Automation Unit Specification**	**Service Certification Report**	**SLA**
Sponsor	Reviews		Reviews		Reviews	Reviews
Service Architect	Responsible for	Approves	Responsible for			
Solution Designer		Approves				
Solution Architect	Approves		Approves			
Service	Approves		Reviews			

Role	Service Description	Service Specification	Automation Unit Description	Automation Unit Specification	Service Certification Report	SLA
Infrastructure Architect						
Service Specification Designer	Approves	Responsible for	Approves	Responsible for		
Service Supplier				Approves		Approves
Sourcing and Usage Manager	Reviews		Approves			Responsible for[1]
Certifier or Certification Authority					Responsible for	
Service Level Manager						Responsible for
Hoster						Responsible for[2]
Consumer (User)						Approves
Service Provider			Approves	Approves		Approves
SOA Governance Lead	Approves	Approves	Approves	Approves	Approves	Approves

Table 4 – Generalized Role Involvement in Different Specification Types

The way in which the roles and service contracts in Table 4 relate to the service oriented process advocated by CBDI[7] is shown in Figure 7. Role definitions, with responsibilities pertinent to service contracts, are included in table 5.

Role	Responsibility
Certifier	Tests services against their Service Specification, Automation Unit Specification, SLA, inspecting the different specifications for adherence to agreed standards. Ensures compliance of services to agreed standards and SOA policy criteria, such as QoS policy.
Sponsor	Funds the service. The role works closely with the business process architect and the service architect (to identify services), and with the service level manager (to agree SLAs).
Service Architect	Plans software services, understanding the business requirement for services in balance with both technology limitations and opportunities. Identifies and

1 The Sourcing and Usage Manager is responsible for the commercial details of the SLA
2 The hoster is responsible for the QoS levels, such as availability and performance.

Role	Responsibility
	structures services, and automation units. Specifies related SOA policy
Solution Designer	Details out solution specifications and designs looking for service creation and reuse opportunities.
Solution Architect	Plans the services to be used by a project. Ensures proposed service usage by a project is communicated to the Service Architect
Service Infrastructure Architect	Plans service infrastructure design. Identifies and specifies infrastructure software units, ensuring consistency of usage of across projects. Specifies related SOA policy. Tunes the service infrastructure technology in line with service execution needs.
Service Specification Designer	Details out service specifications in liaison with service architect.
Service Supplier	Implements and executes services in compliance with SLA; can be different roles for implementing and executing services.
Sourcing and Usage Manager	Negotiates the sourcing and usage aspects of services. There is a very close relationship between this role and the service level manager role. Whereas the latter focuses on SLAs in themselves this role focuses on the implications of SLAs from a customer or a provider relationship viewpoint.
Service Level Manager	Acts as coordination point between customers and providers reviewing SLAs regularly with customers and providers, and ensuring SLAs are communicated clearly to service execution management. The role must work closely with service architects to achieve balanced SLAs and with the sourcing and usage manager in choosing suitable providers
Hoster	Hosts the runtime service with responsibilities for QoS levels.
Consumer (User)	Uses services in compliance with SLA. Where software services are involved usage is via software.
Service Provider	Negotiates and agrees SLAs, including provider responsibilities, and makes services available to users in compliance with SLAs. The provider must possess the ability to offer services in accordance with SLAs, acting as a broker between suppliers and users
SOA Governance Lead	Develops and maintains the SOA Governance Framework. Manages overall SOA governance activities from both a cultural and technology perspective. Monitors and reviews SOA related activity for compliance with SOA Governance Framework. In particular, ensures ownership of SOA related deliverables by appropriate roles.

Table 5 – SOA Role Definitions

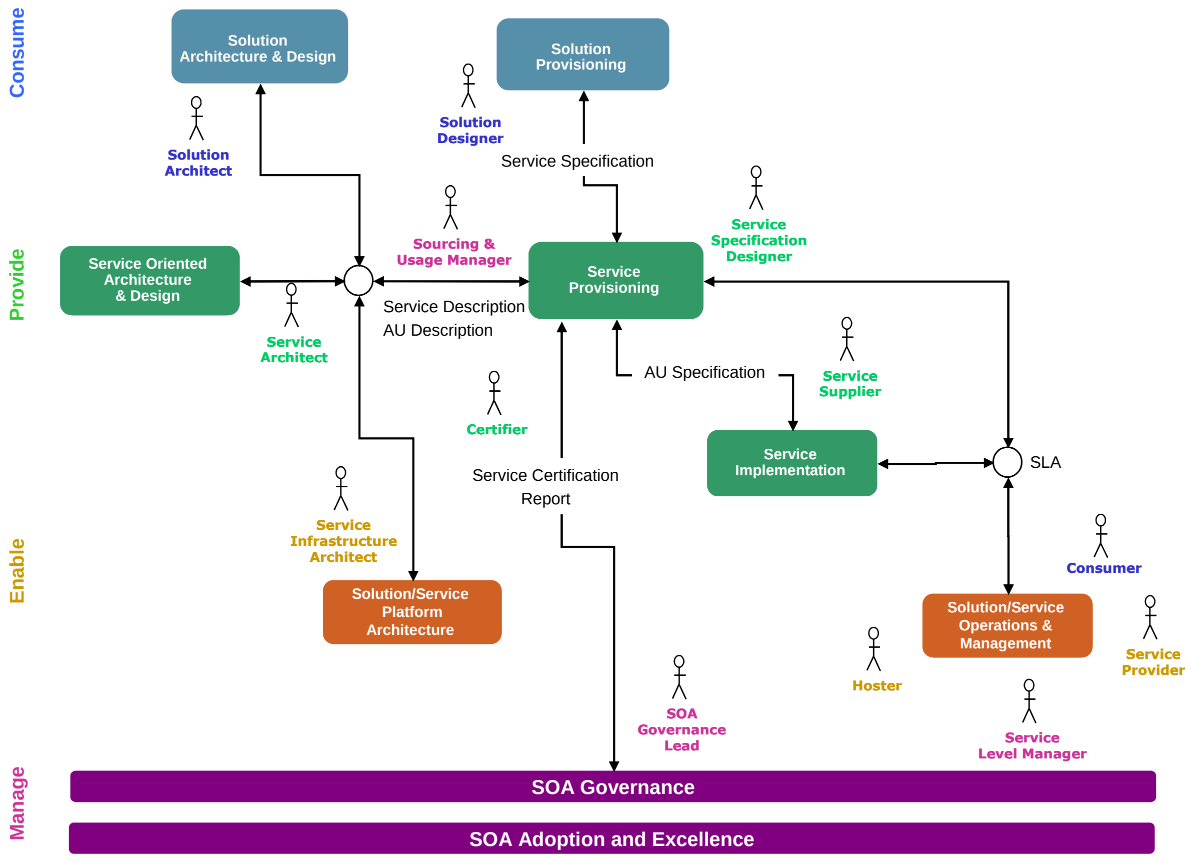


Figure 7 – Service Contracts and Roles in the SO Process

Concluding Remarks

Many organizations are tempted to short cut clear definition of service contracts, faced as they often are with short term horizons based on business planning cycles. It is essential that management provide the support for developing richer service contracts that at the end of the day enable a more agile business based on trust across supplier-consumer boundaries. Fail to prepare – prepare to fail!

At the same time the approach to specification must be accessible, workable and "joined up". In this article we have suggested practical ways of structuring the service contracts that are so central to the success of any approach deemed worthy of the title "service oriented" and provided guidance on the evolution of specification artifacts across the full service life cycle, in an integrated way, to include all the involved parties.

---

1 A Pragmatic Guide to Service Level Agreements, Paul Allen, CBDI Journal, January 2007
2 SOA Governance – Challenge or Opportunity? , Paul Allen, CBDI Journal, April 2008
3 Practical Service Specification and Design Part 3: Specifying Services
4 Service Portfolio Planning – The Implementation View, Denzil Wasson, CBDI Journal, September 2006
5 Service Level Agreements, Paul Allen, CBDI Journal, December 2006
6 SOA Testing – Managing Testing in an SOA Environment, Richard Veryard, CBDI Journal, April 2008
7 Architected Solution Delivery: Extended the Service Oriented Process, Paul Allen and Paul Brown, CBDI Journal, November 2007

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