Software Engineering Institute

Suggestions for Documenting SOA-Based Systems

Stephany Bellomo

September 2010

TECHNICAL REPORT
CMU/SEI-2010-TR-041
ESC-TR-2010-106

Acquisition Support Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library/).

Table of Contents

List of Figures

Acknowledgements

The report is based upon a presentation given at the 27th SIGDOC[1] in October 2009. The presentation was developed in collaboration with Ed Morris and Paul Clements, both of the Carnegie Mellon® Software Engineering Institute (SEI). In addition, the information presented in this report draws largely from the book titled *Documenting Software Architecture: Views and Beyond (V&B)* [Clements 2010] and the corresponding course, developed by the SEI.[2]

1 The 27th ACM International Conference on Design of Communication Special Interest Group on Design of Communication (SIGDOC) was held in Indianapolis, Indiana in October 2009.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

2 The course titled “Documenting Software Architecture” is offered regularly at the SEI. For more information, visit the SEI website (www.sei.cmu.edu/training/p33.cfm).

Abstract

This report provides suggestions for documenting service-oriented architecture-based systems based on the Views & Beyond (V&B) software documentation approach. The V&B documentation approach is a lightweight and flexible approach to documenting software architecture developed by Carnegie Mellon University's Software Engineering Institute.

This report also includes an overview of several well-known service-oriented architecture (SOA) documentation challenges and suggestions for tailoring and augmenting the V&B approach to address those challenges.

The author hopes that the suggestions presented in this report will help SOA developers avoid some of the common documentation pitfalls and produce higher quality SOA documentation.

1 Introduction

This report introduces an approach for documenting service-oriented architecture-based systems based upon the Views & Beyond (V&B) software documentation approach, which was developed by Carnegie Mellon's Software Engineering Institute (SEI). The V&B documentation approach is a lightweight and flexible approach to documenting software architecture [Clements 2010]. This report also includes an overview of several well-known SOA documentation challenges followed by a section containing suggestions for tailoring and augmenting the V&B approach to address those specific challenges.

The structure of this report is as follows:

- Section 2 briefly outlines the evolution of software documentation practice.
- Section 3 provides an overview of the SEI's V&B approach for documenting software.
- Section 4 briefly discusses SOA as an enabling technology.
- Section 5 presents some of the unique challenges to documenting SOA-based systems.
- Section 6 includes suggestions for documenting SOA-based systems.

The author hopes that the suggestions presented in this report will help SOA developers to produce higher quality SOA documentation and avoid some of the common documentation pitfalls.

2 The History of Software Documentation Practice

To illustrate where we have been and where we are today with respect to evolving documentation practices, this section provides a brief history of the evolution of software documentation practice from the 1980s to today.

2.1 Software Documentation Practices in the 1980s and 1990s

The practice of software architecture documentation has matured significantly over the past few decades. In the 1980s, the software development community began developing standalone systems, and architecture documentation practices were ad hoc and informal. At that time, software requirements were essentially lists of system specifications or "shall statements." Software documentation was an afterthought.

In the 1990s, the interest and focus on software documentation and software modeling approaches continued to increase. Software modeling notations such as the Unified Modeling Language (UML) specification and requirements documentation techniques such as use cases as defined by the Object Management Group also began to emerge [OMG 2010].

2.2 Software Documentation Practices Today

As shown in Figure 1, software documentation practices must now extend to the enterprise level. Today software documentation must accommodate notions of system-of-systems interoperability and reuse across an enterprise. The software community has begun to reevaluate documentation practices in an effort to identify which conventional practices are still useful, which require change, and which are now irrelevant.

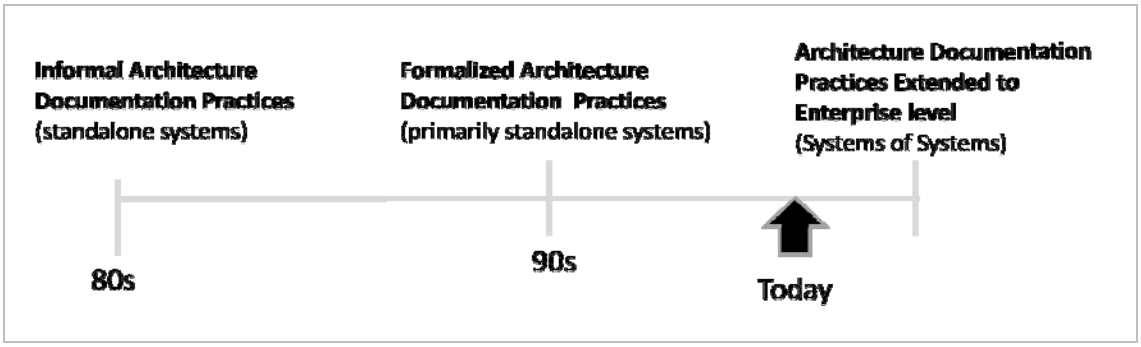


Figure 1: Software Architecture Documentation Timeline

SOA is an architectural style that can promote interoperability; as such, it is an enabling technology for systems-of-systems development. Because the SOA-based architectural style is a relatively new software development approach, documentation practices for SOA-based systems are not well established.

3 Overview of V&B Key Concepts

For those unfamiliar with it, V&B is a lightweight and flexible approach for documenting software architecture. Because the V&B approach is not architecture specific, the SEI's V&B approach for documenting software works well for documenting SOA-based systems.

As an introduction to the V&B approach, a brief overview of some of the key V&B principles are described in the next sections. It is important to note that this overview covers just a few key points from the V&B method. The V&B approach is described in detail in a book in the SEI Software Engineering Series titled *Documenting Software Architectures: Views and Beyond* [Clements 2010].

3.1 A Consistent Documentation Approach

A key concept of documenting software architecture emphasized in the V&B approach is that organizations should *use a consistent documentation approach* across an enterprise. To illustrate why this is important, we will use the Department of Defense Architecture Framework (DoDAF) as an example.

While the DoDAF prescribed the use of a set of views, often the organization provided little guidance about suggested or valid elements that could be used in views. In other words, although two parts of an organization created the same type of view, the views were not necessarily understandable by the other party because element types were not clearly defined.

Suggestions for improving consistency in documentation practices include advocating the use of style guides and enterprise-wide templates. Style guides improve understandability across independently developed views by defining suggested elements and notations for each view type. Precisely defined notations not only provide both guidance and structure for the reader, they can also improve accuracy and completeness in their design documentation [Bass 2003]. In addition to using a style guide to define a common notation for your views, the authors of the SEI book titled *Documenting Software Architectures: Views and Beyond* recommend basing your documentation on an agreed-upon template or format [Clements 2010].

The V&B view template is shown in Figure 2. It is not important that you adopt this specific template, but it is important to use a template. Choose the template that best fits your needs. The key is to have and follow a consistent format for documenting your architecture across the enterprise.

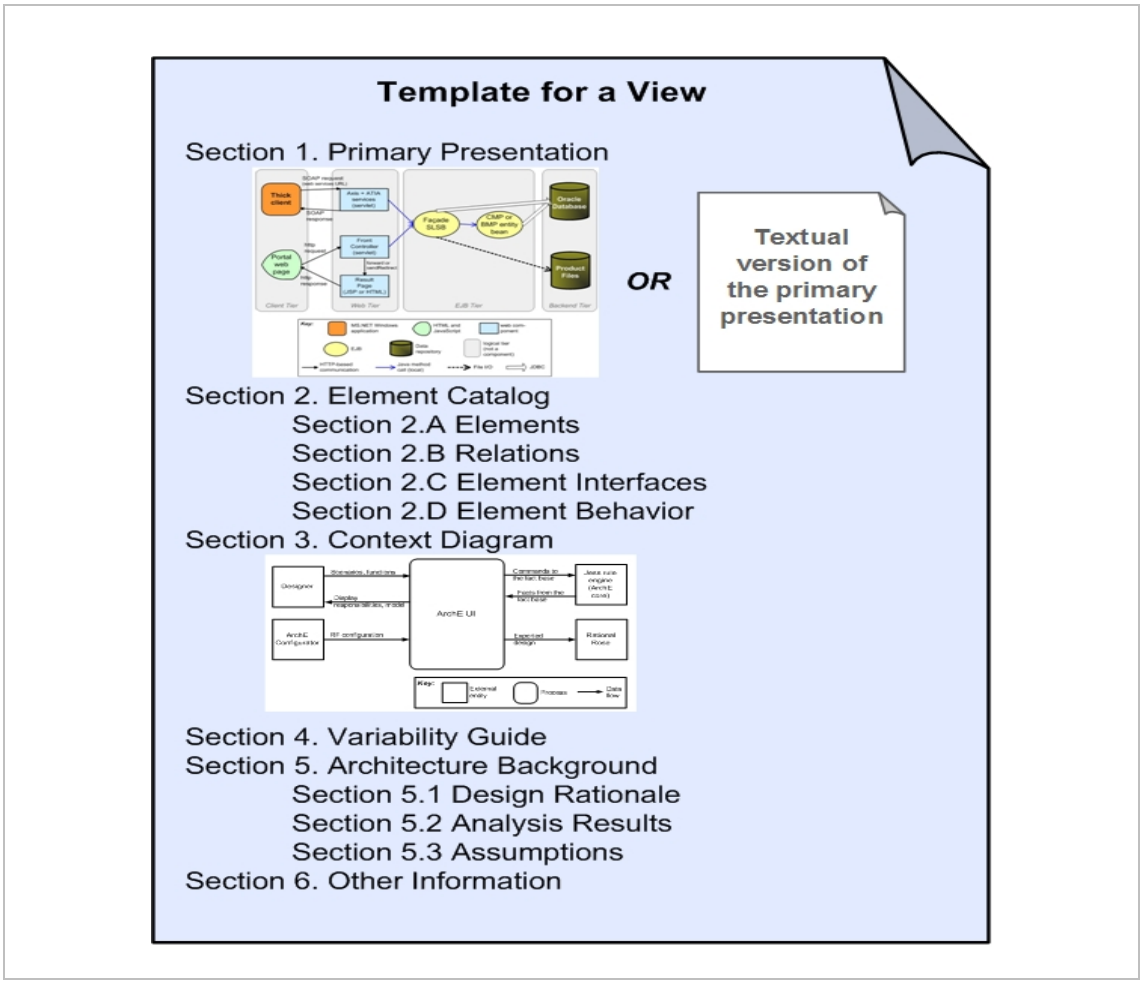


Figure 2: Template for a View

Although not required, most modern software documentation templates use a view-based documentation approach. The V&B approach emphasizes that a view is not just a graphic or picture. There must be accompanying text to describe the important architectural information the view is intended to convey to the reader, as well as any key design decisions.

Views should also contain overarching documentation that provide system context and describe how the views fit together. The V&B approach includes a template for documentation (shown in Figure 3) that captures this type of information

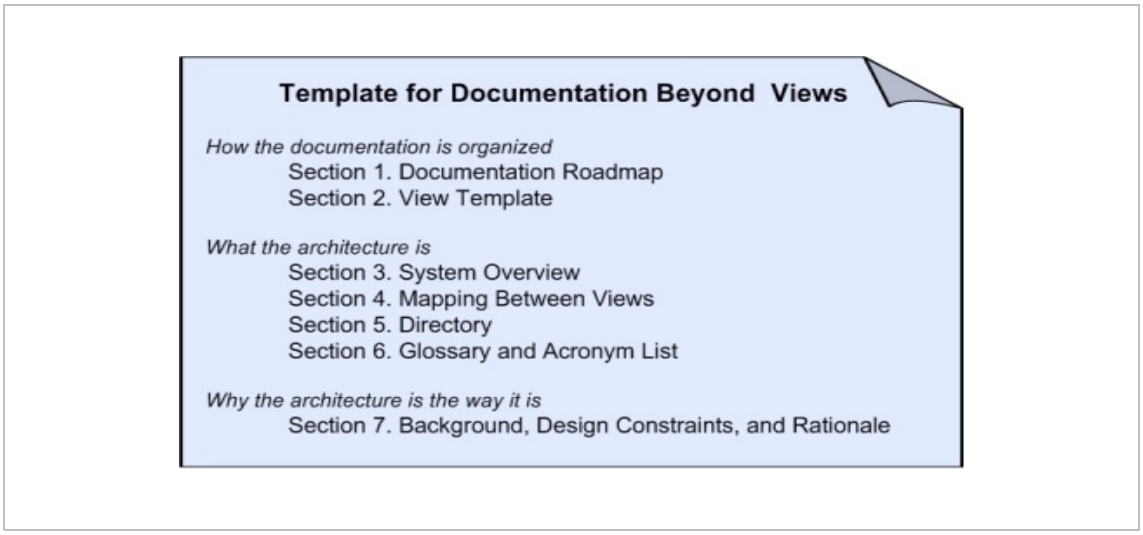


Figure 3: Documentation Beyond Views Template

3.2 A View-Based Approach

Another key concept from the V&B approach, and a well-accepted practice in the software community, is to *use a view-based documentation approach*. Views provide an illustration or snapshot of a key piece of the architecture. In the V&B method, the view is captured in the primary presentation graphic. The primary presentation is the picture described in the first part of the view template (see Figure 2). The primary presentation should be a graphic that is intended to communicate something important about the architecture. For example, a view may describe how the architecture supports achievement of critical security or performance requirements.

Different types of views are appropriate for communicating different aspects of the architecture. For example, module views (see Figure 4) are good for reasoning about how the software architecture supports qualities such as portability and modifiability. Process views (see Figure 5) are good for reasoning about how the software architecture supports performance. Allocation views (see Figure 6) are good for reasoning about the relationship between the physical environment and software components in the architecture.

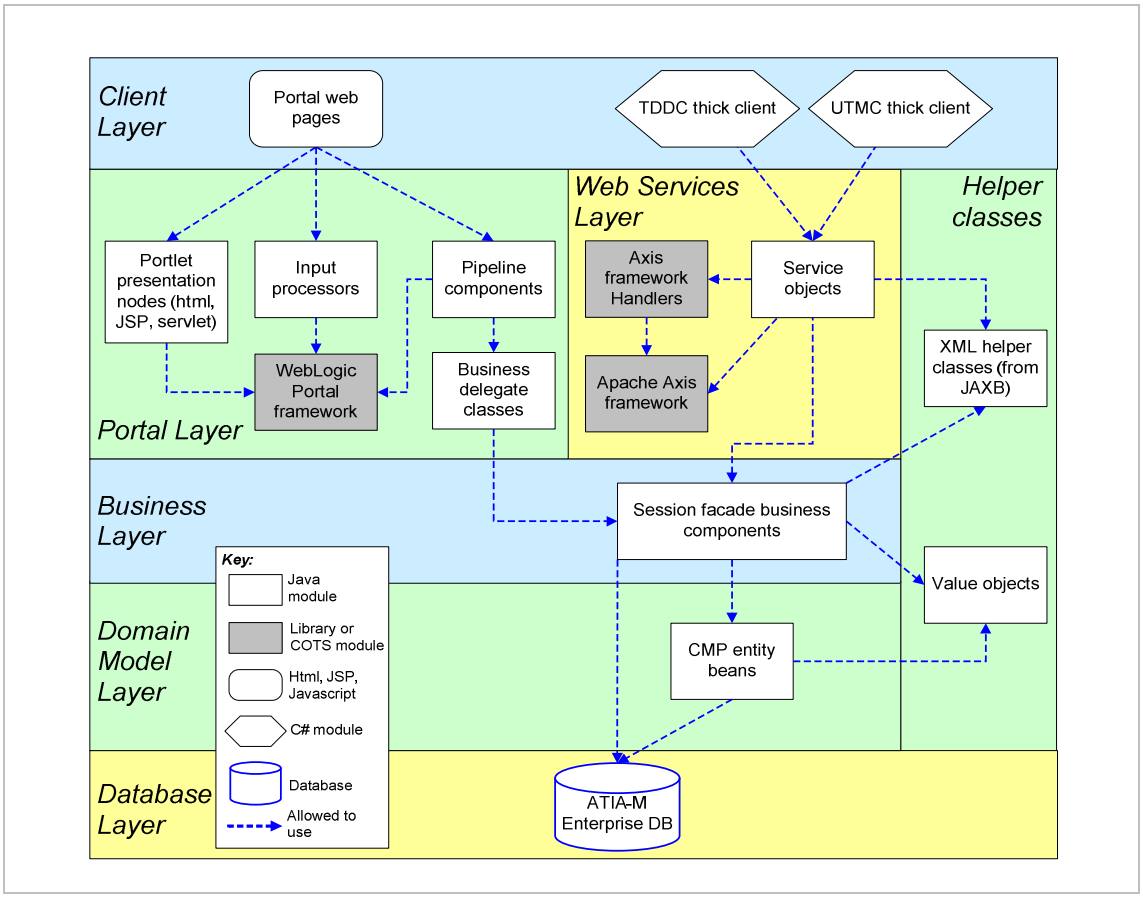


Figure 4: Module View

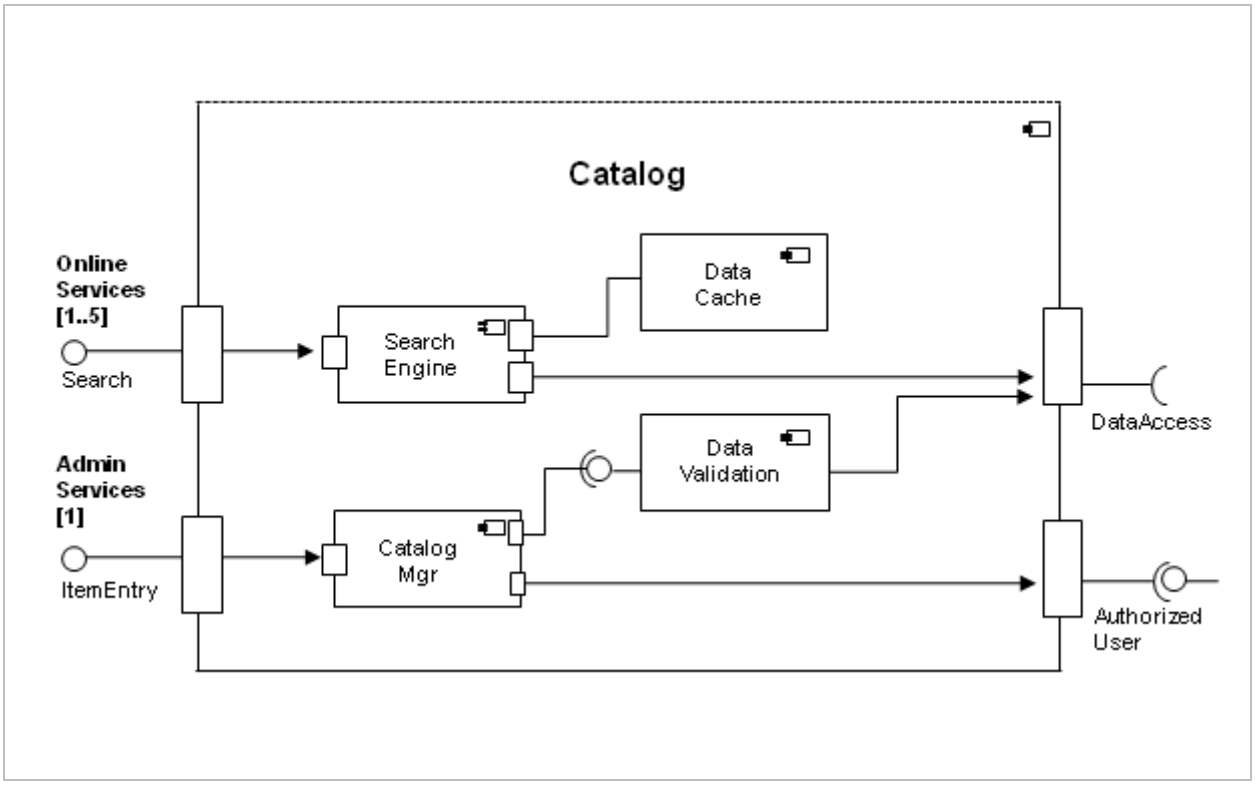


Figure 5: Component and Connector View (Process View)

People often wonder how to know what views to create. A rule of thumb is to develop the views that meet the needs of the highest number of key stakeholders. For example, if a deployment view helps the systems administrator reason about physical configuration and helps the software architect reason about performance, then it makes sense to create a deployment view.

The V&B approach suggests that documenting a software architecture is a matter of documenting the relevant views and then adding information that applies to more than one view. The organizational structure for the V&B approach is illustrated in Figure 7.

Documentation beyond views provides overarching information that describes how the documentation is organized, what the documentation is, and why the documentation is the way it is.

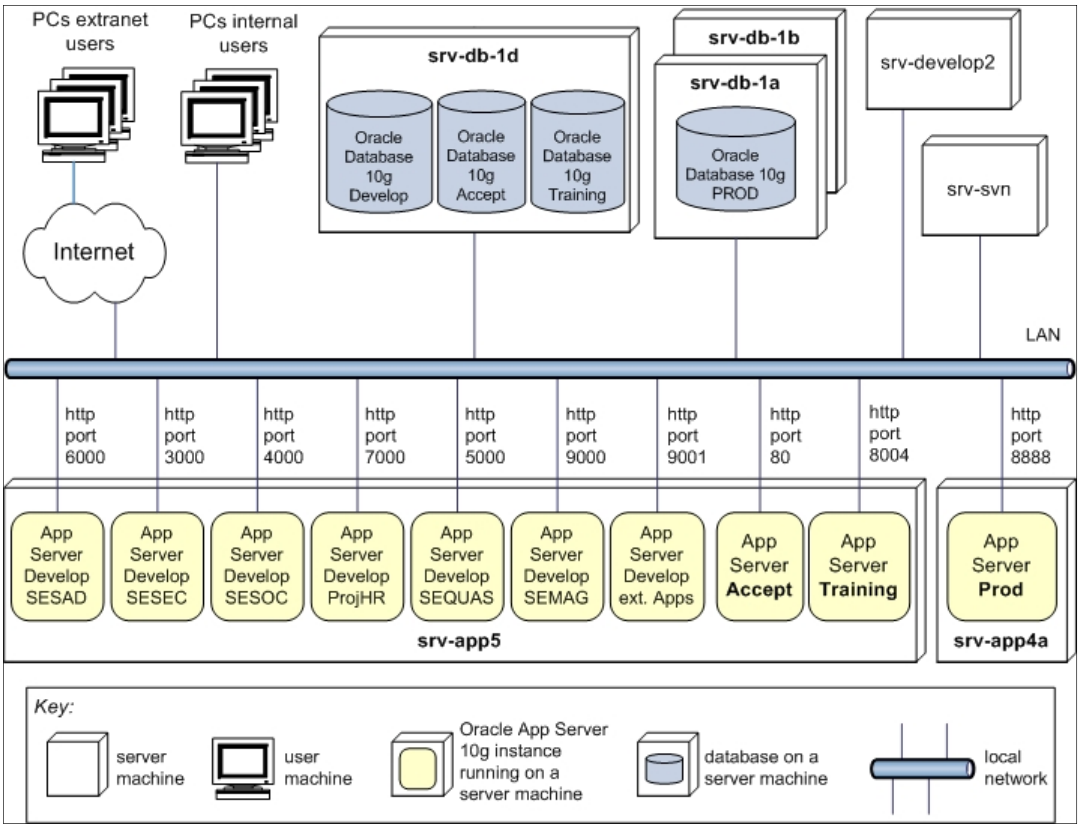


Figure 6: Allocation View

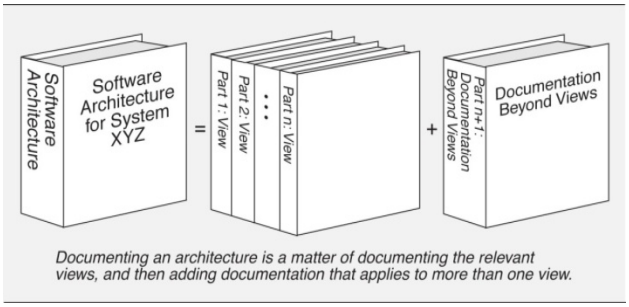


Figure 7: The Organizational Structure for the V&B Approach

3.3 Views That Describe the Key Aspects of the Architecture

It is essential that the views communicate something that is important to the stakeholders who are designing, developing, and/or maintaining the system. This seems like an obvious suggestion; however, too often a list of required views (e.g., DoDAF views) are mandated by an organization and developing views becomes a checklist activity rather than an important step in capturing key architectural decisions.

Figure 8 shows that key aspects of the architecture may be illustrated in the primary presentation. Good architectural views are important artifacts for educating people about the system, analyzing for change, and maintaining the system.

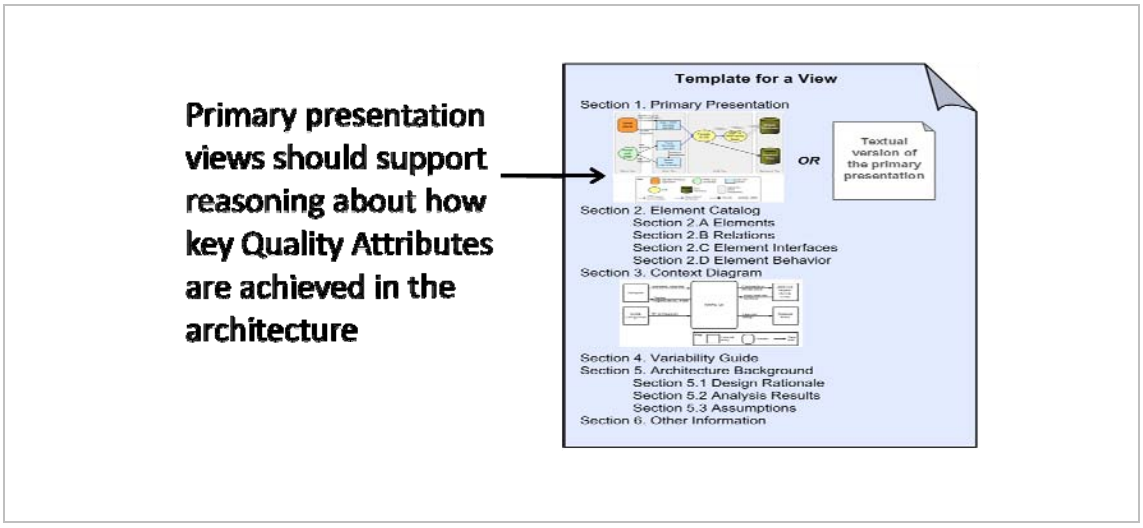
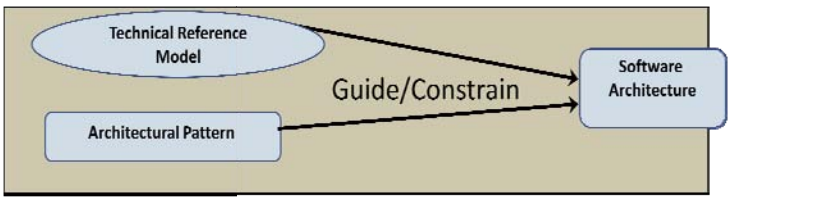


Figure 8: Primary Presentation with Emphasis on Quality Attributes

3.4 Architectural Patterns and Reference Models

Another key concept for documenting software architecture is to *use architectural patterns and reference models.* Architectures are seldom built from scratch, but rather evolve from solutions previously applied to similar problems. Architectural patterns represent a current approach to reusing architectural design solutions. Taking this approach in architectural documentation can improve understandability and foster architectural consistency across systems, and sometimes, even across an enterprise.

Technical reference models (see Figure 9) are typically well established and environment specific (e.g., Oracle database management system reference model). Architectural patterns are more general in nature, and they typically do not include technical details. SOA is an example of an architectural pattern.



Adapted from SEI Software Architecture Principles and Practices Course

Figure 9: Technical Reference Models and Architectural Patterns

4 A Brief Introduction to SOA

SOA is not a technology or a piece of software; rather it is an architectural pattern. As with any architectural pattern, a service-oriented architectural pattern can be instantiated in many different ways for different purposes. Therefore, there is no one service-oriented architecture to use or recommend. There are, however, systems that are based upon the SOA architectural pattern. A runtime depiction of a SOA pattern is shown in Figure 10.[3] Key elements in the SOA architectural pattern include Service Consumers, SOA Infrastructure (ESB) and Services (e.g., Customer Lookup Service).

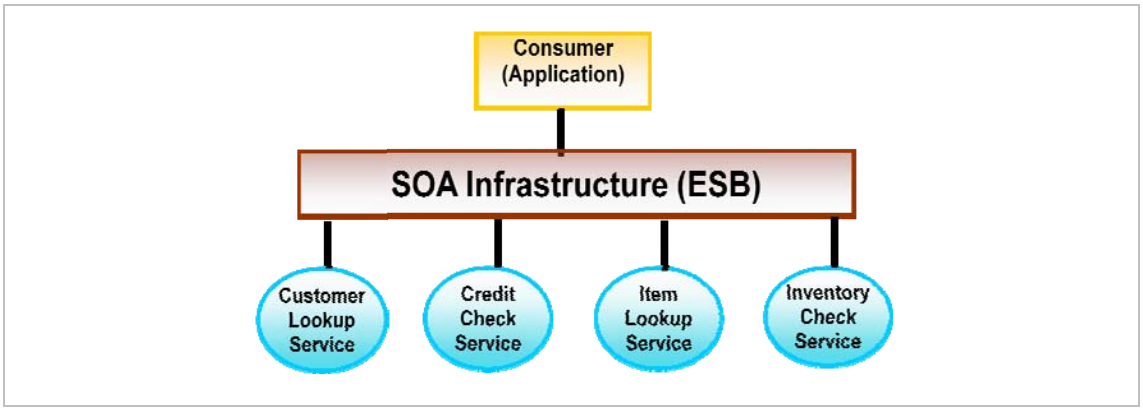


Figure 10: SOA Runtime Diagram Example

One of the reasons for the current popularity of SOA is that it is an enabling technology for integrating systems of systems.[4] SOA can allow disparate systems to exchange data through service interfaces and a common communication infrastructure. Benefits of the service-oriented architectural style include the following:

- There is software reuse and rapid application development.
- Cross-enterprise communication is supported.
- An enterprise can expose legacy functionally through service interfaces.

You can learn more about SOA on the SEI's website[5] or in the new release of *Documenting Software Architectures: Views and Beyond*, which contains a section on documenting SOA-based systems [Clements 2010].

[3] This graphic was taken from the SEI's "Migrating Legacy Systems to SOA Environments" course [SEI 2010].

[4] For more information about systems of systems, refer to Section 2.2 on page 3 of this report.

[5] www.sei.cmu.edu/architecture/research/soa/index.cfm

5 Challenges in Documenting SOA

There are several unique characteristics of the service-oriented architectural style that make documenting SOA-based systems challenging. Some of the more common SOA documentation challenges are listed below.

5.1 SOA Views Are Often Vague or Incomplete

There are a couple of reasons that SOA views can be vague, confusing, and incomplete. One problem is that often SOA views only provide a picture and do not include the supporting textual description that is necessary to understand why the view is important. Readers are then left to guess why the architect chose to capture a particular view. The reader might wonder why a picture-only view was created, whether it is intended to convey an important architectural decision, and whether it communicates something important about how a quality attribute is achieved.

With SOA, the audience for runtime views is often expanded to other stakeholders, such as enterprise leaders, business analysts, or requirements managers since they are trying to reason about how SOA will promote reuse or interoperability. Since these people are not engineers, the cryptic and confusing formal notations for documenting runtime views can be frustrating. To compensate for this, SOA documenters tend to use informal views. The increased use of informal notation in SOA views helps readers create pictures that are easy to understand. However, this type of notation is sometimes vague and that often confuses people (such as architects) who need a more precise understanding of the architecture.

5.2 Future QoS Needs Are Not Known at Design Time

Another challenge is that, unlike traditional systems development, SOA software architects do not necessarily know how their services will be used by all potential future consumers when the service is developed. This means that service developers must speculate about future service consumers and allow for some degree of variability in the quality of service (QoS) provided by the service.

For example, imagine that a credit card service provides credit card authentication to a large online store such Amazon.com. Then imagine that the same credit card service is used by a small "mom and pop" business. Amazon.com may require a fast response time for credit card authentication, such as 5 seconds per transaction, and in order to get that quality of service Amazon.com may be willing to pay the credit card authentication provider $1.00 per credit card authentication.[6] The mom and pop business, on the other hand, may be able to live with a 10-second response time. Because of the reduced quality of service (10 seconds instead of 5 seconds per transaction) the mom and pop store may only be charged $.50 per transaction. The point is

[6] Please note that these are not real response times and fees.

that SOA documentation approaches should provide a way to compensate for unpredictability and variability in QoS needs.

5.3 Addition of Future Consumers May Necessitate Extensive Rework

Designers of SOA-based systems expect that there will be future consumers that are unknown at design time. Because of this, SOA documenters know that the documentation approach they use for SOA-based systems must accommodate future consumer usage scenarios without requiring a "ground up" overhaul of the documentation. This means that the structure of the documentation requires forethought and, like SOA, needs to be extensible and adaptable.

5.4 What Documentation Can (or Should) Be Auto-Generated?

SOA vendor tools allow developers to auto-generate some aspects of SOA documentation. For example, Web Services Description Language (WSDL) files are typically auto-generated and are human readable artifacts. Because of this, WSDLs can be used as documentation. While WSDL files do a good job of describing the syntax of a service interface, they do not effectively describe the quality attributes supported by the service. Examples of this are response time or security levels that are supported by a particular service interface. Additionally, automatically generated files will not describe the design rationale behind a specific service implementation. Because of this, SOA documenters should determine what to auto-generate versus what they should document manually.

5.5 SOA Vendor Tool Access Challenges

It is good that there is much tool support for developing SOA-based systems; this speeds up development and provides a seamless, integrated environment for the developers. However, these tools can lead to isolated "SOA islands," which limit the sharing of documentation and artifacts outside the development team if access to documentation requires a vendor software license or login. If licensing or access controls restrict people who need access to the documentation from getting to artifacts, then regardless of its positive attributes you may need to consider alternative approaches to sharing documentation. An example of a critical artifact that must be widely accessible to service consumers is the list of available services. If access to service lists is limited to a select set of users, this severely limits the likelihood that the services will be reused across an organization. Therefore, SOA documenters need to reason about whether or not the tools they are using to develop and share documentation allow for adequate access to documentation for all potential documentation users.

5.6 Lack of Technical Governance Limits Interoperability

SOA is an architectural pattern that fosters interoperability. However, if there is no governance over things like enterprise-wide SOA security architecture or web service standards, it is entirely possible to end up with a group of independent service-oriented systems that cannot effectively or safely interact. If this happens, the organization may be worse off than before it moved to a service-oriented development approach. For that reason, it is important to consider establishing

technical governance—including recommended standards, patterns, and SOA reference models—as part of the SOA Governance Framework.

5.7 Semantic Mismatch for SOA System Data Exchange

Many organizations adopt a service-oriented architectural approach to foster interoperability. For example, organizations may plan to exchange data among independent systems by exposing legacy functionality through service interfaces. Organizations might also share data with multiple consumers through that same set of services. While interoperability through services is a major benefit to organizations that have critical data residing in stove-piped systems, there are also challenges with achieving interoperability. One such challenge is getting both parties to agree on the semantics of the data being exchanged using services.

6 Suggestions for Documenting SOA-Based Systems

In this section, we provide suggestions for documenting SOA-based systems. Some of these suggestions are based on best practices defined in the V&B documentation approach (with a specific focus on documenting SOA-based systems). Other suggestions are based upon common industry practices or SEI experiences with the Department of Defense (DoD) and other government programs developing SOA-based systems.

Several suggestions for documenting SOA-based systems are included below.

6.1 Include SOA Style Guide and Documentation Guidelines in Governance Strategy

Defining consistent documentation guidelines should be part of an overarching SOA governance strategy that is managed by a governance body at the enterprise level. Two artifacts that need to be developed to establish consistency across the enterprise are the SOA Documentation Style Guide and SOA Architecture Documentation Guidelines, shown in red in Figure 11.[7]

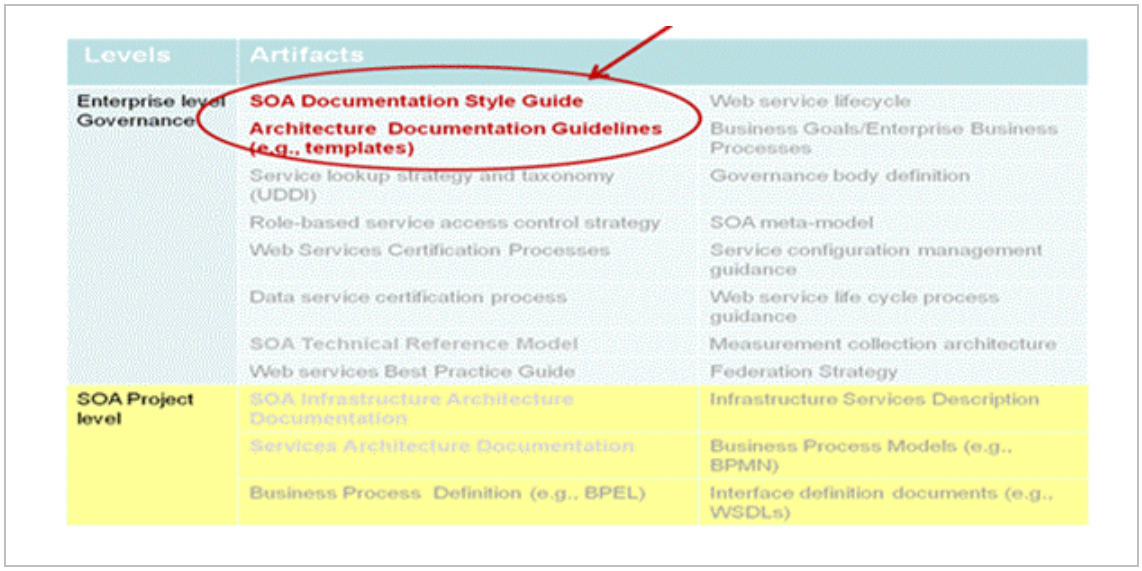


Figure 11: SOA Governance Artifacts for Consistency

The enterprise SOA style guide should define informal notations used in SOA views, SOA-related elements, and relationships among elements. SOA architecture documentation guidelines should provide information related to structuring SOA documentation such as suggested templates. In addition, it is recommended that enterprise-wide SOA standard (e.g., WS* standards) guidelines also be provided as part of the SOA Governance Framework.

[7] Items in Figure 11 that are not emphasized have been intentionally grayed out.

As part of the SOA governance strategy, it is important to establish a suggested enterprise SOA view template. Because it is lightweight, flexible, and easily adapted to SOA-based system documentation, we suggest using the V&B approach for SOA-based systems. One of the benefits of the V&B template is that it promotes inclusion of accompanying text sections with views that describe key SOA architectural decisions, elements (service interfaces), and relationships (SOA messages). In addition, the template promotes completeness and accuracy in developing SOA views.

There are three places in the V&B template where one can capture information about the quality attributes that are promoted in the architecture. Quality attributes are attributes of a particular architecture. Examples of quality attributes are performance, reliability, and security. Quality attributes are often referred to as QoS characteristics in SOA-based systems.

The first place to document how qualities are achieved using the V&B template is the primary presentation. The second place to capture QoS information is in the element behavior section of the template. For example, you may specify that a service provides a response time of two seconds. Service Level Agreements (SLAs) may also be captured in the Element Behavior section. The third place to capture QoS information is the Design Rationale section of the V&B template. This section may contain design rationale behind selected SOA patterns or impactful architectural decisions.

In SOA-based systems, informal notation is often used to create runtime views. Figure 12 shows an example of a SOA runtime view using informal notation from the J2EE Adventure Builder.[8]

As discussed in the previous section, informal views such as the one shown in Figure 12 are popular with SOA architects because they are easy to understand. However, informal documentation can lead to vague or confusing views. For that reason, it is imperative that all elements and relationships among elements are described in a SOA enterprise style guide or, at a minimum, a SOA view key.

[8] The J2EE Adventure Builder is available on the Java website (https://adventurebuilder.dev.java.net/).

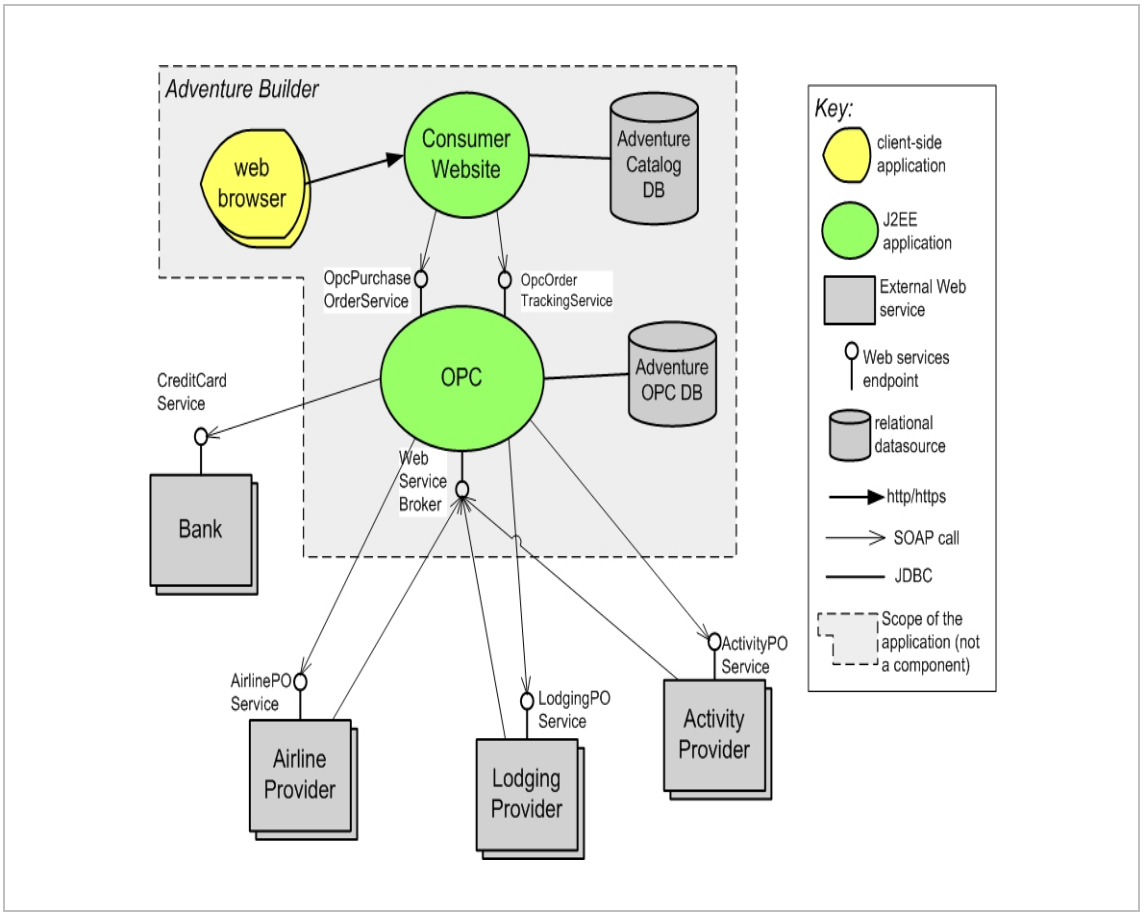


Figure 12: An Example SOA Runtime View from the J2EE Adventure Builder Tutorial

6.2 Use Ranges to Capture Service QoS Characteristics

As mentioned earlier, one challenge in documenting SOA-based systems is that the QoS needs for future consumers are unknown at design time; these needs may also vary among consumers. One way to deal with this is to provide a QoS range instead of a discrete value for the QoS variables. For example, a service may provide a response ranging between 2 and 10 seconds depending on factors ranging from how much the consumer is willing to pay for the service to how much bandwidth is available between the service consumer and service provider. These ranges can be included in the element behavior section of the template described above. More about this topic is available in Chapter 6, "Advanced Concepts" in the book *Documenting Software Architecture: Views and Beyond* [Clements 2010].

6.3 Structure SOA Documentation to Accommodate Future Consumer Service Uses

It is relatively straightforward to describe how services are used by known consumers in a view. The tricky part is making SOA views flexible enough to accommodate new, unknown consumers easily. To make the documentation approach easily extendible, we suggest creating a view package for each service-consumer capability.

In Figure 13, the Patient Portal consumer usage (circled in red) shows one service-consumer combination. For each service-consumer combination, one may provide overarching documentation to describe a set of associated views and how the views fit together. For example, there may be a set of views that describe how the patient portal leverages services such as the Create Lab Test Order Service. One might create a second service-consumer package called Hospital App. The views in this package may reuse services used in the Patient Portal or may include new ones. Again, there would be overarching documentation for the Hospital App usage scenario to describe how the views fit together. Each set of packages will evolve independently with their own version control.

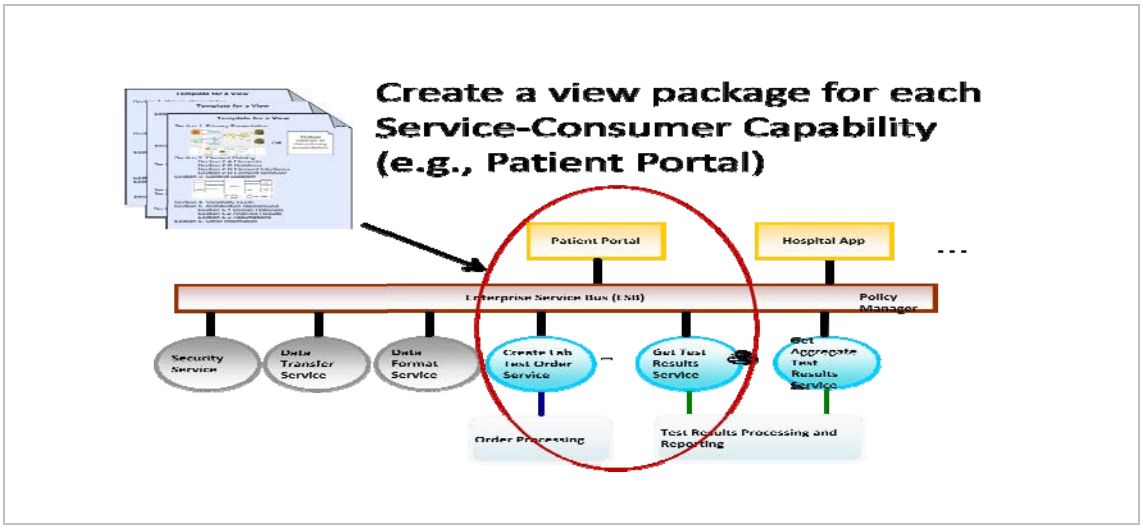


Figure 13: View Package Service-Consumer Capability

In addition to providing the ability to easily add new service-consumer usages, it is also necessary to provide an extendible way to document the SOA infrastructure architecture. The SOA infrastructure is referred to in Figure 14 as the Enterprise Service Bus (ESB) and is circled in red. Keep in mind that the SOA infrastructure may contain more than just the ESB; it may also contain utility services, data transformation services, and security services.

The SOA infrastructure may leverage many patterns to accomplish various service consumer and provider needs. As with the consumer-service usage documentation described above, it is also necessary to provide the overarching documentation, which will describe how the infrastructure-related views fit together to form the SOA infrastructure services. As with the service-consumer usage packages, each set of packages will evolve independently with their own version control.

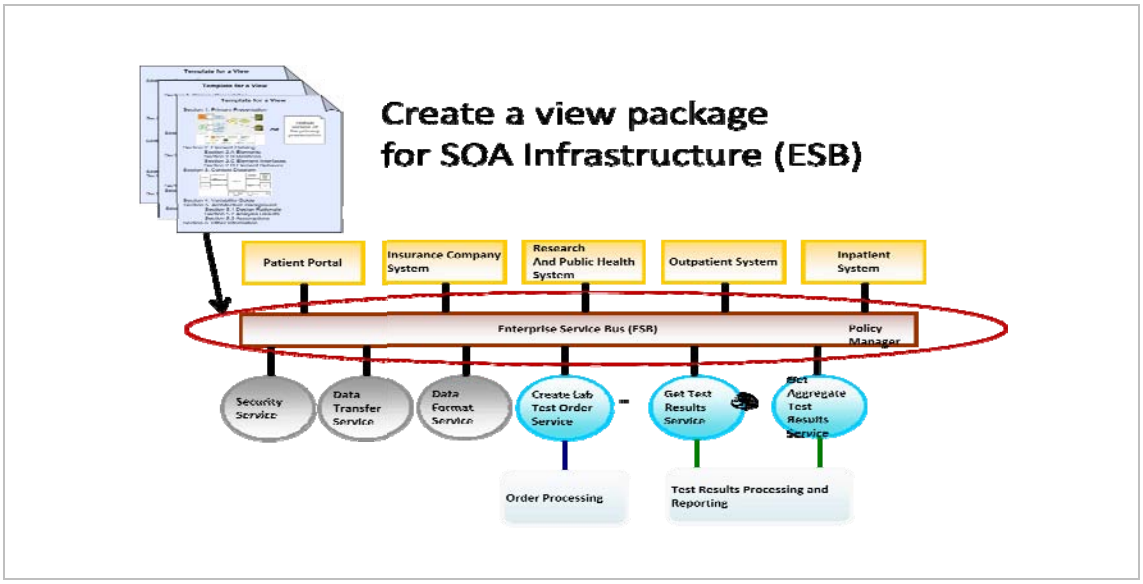


Figure 14: View Package SOA Infrastructure View Package

6.4 Auto-Generate Static Information; Manually Document Architectural Decisions

To minimize documentation maintenance, tools can be used to auto-generate some parts of your documentation from the actual SOA operational artifacts. Documentation based on static files (e.g., interface specifications) is a good example of the type of documentation that can be automatically generated. Interface specifications can be captured in WSDLS using XMI (open standard for generating documentation). Alternatively, you can use JavaDoc and Business Modeling tools to auto-generate some of your SOA documentation.

However, there are limitations to auto-generated SOA documentation. Auto-generated documentation needs to be augmented in some places with manually created text descriptions of the rationale for key design decisions and information about how key quality attributes are achieved through the architecture. We suggest auto-generating static information, such as service interface information, and then manually augmenting that auto-generated documentation with information about key architectural decisions.

6.5 Use SOA Vendor-Agnostic Tools to Make SOA Documentation Accessible

One of the SOA documentation challenges introduced in the previous section is that it is important to make sure that the SOA documentation that is developed using SOA vendor tools is widely accessible. A good alternative to using proprietary SOA tools to capture and share SOA documentation is to use SOA vendor-agnostic tools like IBM's System Architect to share your SOA-based documentation. These tools allow you to create an overarching structure for your documentation and then link back to source documents. As you can see in Figure 15, Gartner

Group believes that Telelogic's System Architect is arguably the most capable for documenting SOA-based systems.[9]

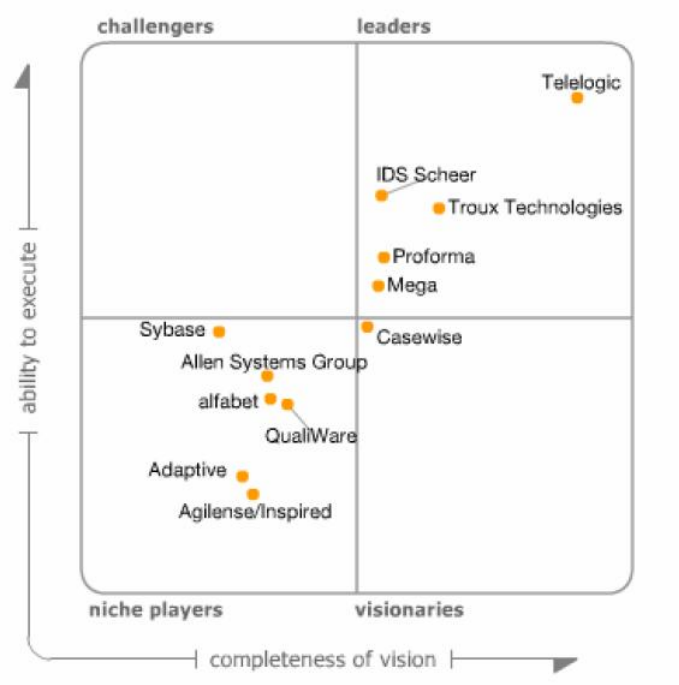


Figure 15: Magic Quadrant for Enterprise Architecture Tools, April 2006

Another option for making your SOA documentation widely accessible is to create your own web structure based on the V&B templates. The SEI report titled *Creating and Using Software Architecture Documentation Using Web-Based Tool Support* provides some examples [Stafford 2004]. Whether you use web pages or an enterprise architecture tool, it is important to remember that the documentation needs to be widely available.

6.6 Establish Technical Governance as Part of the SOA Governance Framework

To promote interoperability for SOA-based systems, it is helpful to define recommended SOA patterns and technical reference models for the enterprise. The idea is not to mandate that only certain SOA patterns must be used within an enterprise; rather it is to provide guidance to lead architects toward a common way of doing things to promote interoperability across independently developed SOA systems. SOA developers will likely leverage many other patterns or tactics to achieve functional requirements and quality attributes. Some of the patterns used in a SOA-based architecture may be SOA-specific and others may not. A data service hosted as part of the SOA

[9] Figure 15 was excerpted from the Gartner RAS Core report titled *Magic Quadrant for Enterprise Architecture Tools* [Handler 2009].

infrastructure using a publish-subscribe pattern is an example of a pattern that is often used to support SOA-based design. As illustrated in Figure 16, technical reference models can provide reference architectures and package patterns for SOA developers [OASIS 2006].

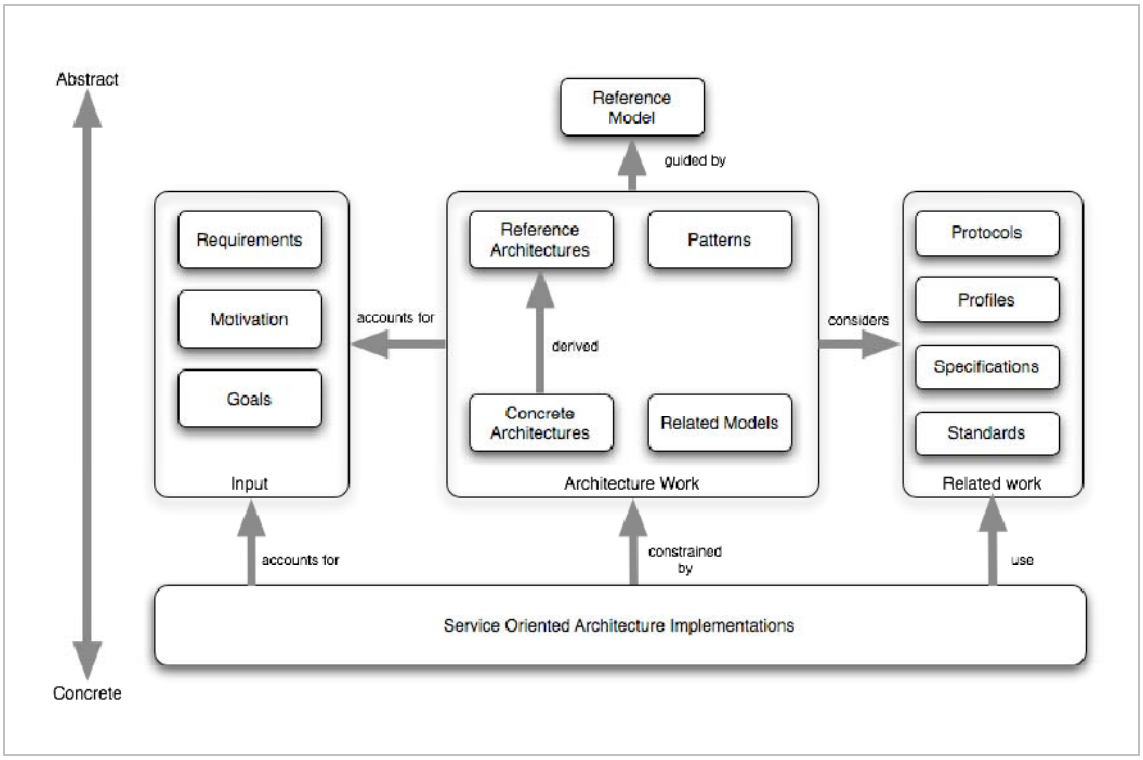


Figure 16: OASIS Technical Reference Model Descriptive Diagram

6.7 Develop a Common Semantic Ontology for Core Data

A significant challenge for those developing SOA-based systems to support interoperability is getting both parties to agree upon the semantics of the data being exchanged between systems. More on interoperability challenges can be found in the SEI report titled *System-of-Systems Navigator: An Approach for Managing System-of-Systems* [Brownsword 2006]. Some enterprises invest significant resources developing a common ontology that can be shared among organizations or within an enterprise. If an enterprise-wide common ontology for a core set of data is developed, it should be captured in documentation, accurately maintained, and made widely accessible. The development of an enterprise ontology should also be included in the SOA governance strategy.

7 Summary

Much research remains to be done in the field of SOA documentation practices. The SEI is researching the viability of using tool-based SOA documentation techniques as well as concepts for designing and documenting using QoS ranges. While many questions are yet to be answered, we hope that this report presented a useful approach for structuring SOA-based documentation. We also hope the report provided useful suggestions that developers may use to help SOA documenters avoid common SOA-documentation pitfalls.

8 References

URLs are valid as of the publication date of this document.

[Bass 2003]
Len Bass, Paul C. Clements, & Rick Kazman. *Software Architecture in Practice*, Addison-Wesley Professional, 2003 (ISBN: 0321154959).
www.sei.cmu.edu/library/abstracts/books/0321154959.cfm

[Brownsword 2006]
Lisa Brownsword, David Fisher, Edwin J. Morris, James Smith, & Patrick Kirwan. *System-of-Systems Navigator: An Approach for Managing System-of-Systems Interoperability* (CMU/SEI-2006-TN-019). Software Engineering Institute, Carnegie Mellon University, 2006.
www.sei.cmu.edu/library/abstracts/reports/06tn019.cfm

[Clements 2010]
Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, & Judith A. Stafford. *Documenting Software Architectures: Views and Beyond, Second Edition.* Addison-Wesley Professional, 2010 (ISBN: 0321552687).
www.sei.cmu.edu/library/abstracts/books/0321552687.cfm

[Handler 2009]
Robert A. Handler & Chris Wilson. *Magic Quadrant for Enterprise Architecture Tools* (Gartner RAS Core Research Note G00172491). Gartner Research, November 2009.
http://imagesrv.gartner.com/media-products/pdf/reprints/ibm/external/volume4/article28.pdf

[OASIS 2006]
OASIS. Reference Model for Service Oriented Architecture 1.0," August 2006.
www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[OMG 2010]
Object Management Group Website, September 2010. www.uml.org/

[SEI 2010]
Software Engineering Institute. "Migrating Legacy Systems to SOA Environments." Software Engineering Institute, Carnegie Mellon University, 2010. www.sei.cmu.edu/training/V06.cfm

[Stafford 2004]
Judith A. Stafford. *Creating and Using Software Architecture Documentation Using Web-Based Tool Support,*" (CMU/SEI-2004-TN-037). Software Engineering Institute, Carnegie Mellon University, 2004. www.sei.cmu.edu/library/abstracts/reports/04tn037.cfm

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Suggestions for Documenting SOA-Based Systems	FA8721-05-C-0003

6. AUTHOR(S)

Stephany Bellomo

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2010-TR-041

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2010-106

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report provides suggestions for documenting service-oriented architecture-based systems based on the Views & Beyond (V&B) software documentation approach. The V&B documentation approach is a lightweight and flexible approach to documenting software architecture developed by Carnegie Mellon University's Software Engineering Institute. This report also includes an overview of several well-known service-oriented architecture (SOA) documentation challenges and suggestions for tailoring and augmenting the V&B approach to address those challenges.

The author hopes that the suggestions presented in this report will help SOA developers to avoid some of the common documentation pitfalls and produce higher quality SOA documentation.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Service-oriented architecture, SOA, SOA-based systems, SOA documentation	42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL