



California
DEPARTMENT OF TECHNOLOGY

California Enterprise Architecture Framework

Service-Oriented Architecture (SOA) Reference Architecture (RA)

Version 1.0 Final
January 2, 2014



This Page is Intentionally Left Blank



TABLE OF CONTENTS

1	Introduction.....	1
1.1	Purpose	1
1.2	Limitations.....	1
1.3	Intended Users.....	2
1.4	Document Organization	2
1.5	Future Directions	2
2	SOA Overview.....	3
2.1	Definitions.....	3
2.1.1	<i>SOA as Paradigm and Service Orientation</i>	3
2.1.2	<i>SOA as Style</i>	4
2.1.3	<i>SOA as (Architectural) Model</i>	5
2.1.4	<i>SOA RA and CEAF</i>	6
2.2	Business Benefits of Adopting Service Orientation and SOA.....	7
2.3	Main SOA Usage Scenarios	8
2.4	Key Capabilities of SOA Solution.....	8
2.5	Components of SOA Solution.....	9
2.5.1	<i>Consumer Interfaces</i>	10
2.5.2	<i>Business Processes</i>	12
2.5.3	<i>Services</i>	14
2.5.4	<i>Service Components</i>	15
2.5.5	<i>Enterprise Systems</i>	16
2.5.6	<i>Enterprise Decision Management</i>	17
2.5.7	<i>EAI and EII</i>	18
2.5.8	<i>IdAM Platform</i>	19
3	SOA Reference Architecture Description.....	21
3.1	SOA RA Conceptual View	21
3.2	SOA RA Logical View	23
3.2.1	<i>High-Level Interactions</i>	23
3.2.2	<i>Business Process-Driven Interactions with Services</i>	25
3.3	Deployment View of SOA RA	29
4	Implementation Notes and Guidelines	30



4.1	SOA Patterns and Anti-Patterns.....	30
5	Glossary.....	33
6	References.....	34
6.1	Federal and State Documents.....	34
6.2	Books and Papers.....	34
6.3	Referenced Web Sites and Web Resources.....	34
7	Document History.....	36



LIST OF FIGURES

<i>Figure 2-1 Reference Architecture and SOA in CEAF 2.0</i>	6
<i>Figure 2-2 Groups of Architectural Components in SOA RA</i>	9
<i>Figure 2-3 SOA Consumer Interfaces Layer Overview</i>	11
<i>Figure 2-4 SOA Business Processes Layer Overview</i>	12
<i>Figure 2-5 SOA Services Layer Overview</i>	14
<i>Figure 2-6 SOA Service Components Overview</i>	16
<i>Figure 2-7 Overview of Enterprise Systems in SOA</i>	16
<i>Figure 2-8 SOA Enterprise Decision Management Overview</i>	17
<i>Figure 2-9 SOA EAI Platform Overview</i>	18
<i>Figure 2-10 SOA EII Platform Overview</i>	19
<i>Figure 2-11 SOA IdAM Platform Overview</i>	20
<i>Figure 3-1 SOA Reference Architecture – Conceptual View</i>	22
<i>Figure 3-2 SOA Component Interactions</i>	23
<i>Figure 3-3 Typical Interactions during Execution of Business Process as a Service</i>	26



LIST OF TABLES

<i>Table 3-1 High-level interactions among SOA components</i>	24
<i>Table 3-2 Interactions during Execution of Business Process as a Service</i>	27
<i>Table 4-1 Common SOA Anti-Patterns</i>	31
<i>Table 7-1 Document History</i>	36



This Page is Intentionally Left Blank

1 Introduction

Introducing Service Oriented Architecture (SOA) is actually not easy: the term means different things to different people. To an industry analyst, SOA may mean an increasingly popular paradigm for making sense of the ever-growing complexity of the enterprise and for organizing its supporting information systems. To a decision maker, SOA may mean a promise of harnessing complexity and costs of IT in the enterprise. To an architect, SOA may mean a specific style for creating architectures of systems and applications that address specific business needs. To a developer, SOA may mean using Web Services and SOAP over http.

Regardless of the particular viewpoint, two things about SOA seem shared: one is the central concept of “service”, and the other is the belief that SOA is the main – if not only – valid and validated way of coping with current business challenges for the enterprise and for information systems providing the backbone of today’s enterprise. The adoption of SOA has been growing, and SOA has matured over the last decade. Even though adopting SOA is not without its share of challenges, the key question about SOA now-a-days is not whether to adopt it or not, but rather how to do that effectively, keeping in mind short- and long-term business and technical concerns.

SOA is important to the state as a key element in Target Enterprise Architectures. Consequently, CEAF 2.0 includes SOA into its scope by presenting a Reference Architecture for it, considered in the context of the business needs of the state and state agencies. This context shapes the way SOA is viewed in CEAF 2.0, both as a paradigm applicable to a number of domains and as an architectural style that facilitates practical adoption of Business Process-oriented enterprise architecture in the enterprise.

1.1 Purpose

The SOA Reference Architecture document provides guidelines and options for making architectural decisions when implementing SOA-based solutions.

The objectives for the document include the following:

- To introduce key terms and distinctions relevant for the topic
- To provide inputs for creating or evaluating architectures in the Service Oriented Architectural *style*
- To identify building blocks (architectural layers, services, components) for integrating elements of an SOA-based solution
- To communicate the key architectural decisions relevant for creating or evaluating SOA-based solutions
- To communicate opportunities for solution and/or platform sharing at agency, cross-agency and/or state levels.

1.2 Limitations

The document focuses on SOA in context of CEAF 2.0 and it may not cover all concepts related to SOA in other contexts. The document is concerned with Service Oriented architectural (and to some degree, also business) models and it is not intended as a product guide or an endorsement of any specific product.

1.3 Intended Users

The primary intended users of this document are Enterprise Architecture practitioners and other architects that contribute to enterprise architecture. This broad group includes architects from other domains/disciplines such as Security, Application, Information, Business, Technology, Infrastructure, and Solution Architects. It is also beneficial to Managers, at senior or operational levels, who are involved with SOA or related areas, such as Enterprise Application Integration, Master Data Management, Cloud Computing, Identity and Access Management, and similar areas.

1.4 Document Organization

The SOA Reference Architecture documentation is organized as follows:

- Section “SOA Overview” provides background for the SOA RA by introducing descriptions and definitions of SOA, discusses the main usage scenario types found in SOA implementations, and identifies architectural components for respective usage scenarios.
- The section “SOA Reference Architecture Description” elaborates RA for SOA using the following architectural views:
 - The Conceptual View (in the subsection “SOA RA Conceptual View”) introduces the necessary capabilities for an SOA and how they are supported by Architectural Building Blocks (ABBs)
 - The Logical View (in the subsection “SOA RA Logical View”) describes key interactions among Layers and/or ABBs to realize functionality specific to SOA systems
 - The Deployment View (in the subsection “Deployment View of SOA RA”) focuses on system topologies and deployment facets of ABBs.
- The section “Implementation Notes and Guidelines” provides initial guidance for SOA implementation, concentrating on common anti-patterns
- The section “Glossary” provides description of the terms and abbreviations used in the document
- The section “References” lists publications used for preparation of the document.

1.5 Future Directions

Future evolution of the document includes the following steps:

- Addition of existing best-practice-based realizations of the SOA RA
- Identification and elaboration of solution sharing opportunities
- Formulation of implementation guidelines for SOA RA

2 SOA Overview

This section provides a description of SOA to address different understandings of the term present in the industry and to clarify some of the common misunderstandings that can affect the creation of Target EA and related roadmaps. This section also discusses the intended business benefits of SOA, key capabilities of an SOA solution, and its main usage scenarios. Key components of the solution are described at a high level in this section.

2.1 Definitions

The term “Service-Oriented Architecture” (or “SOA”) has been introduced by Gartner in 1996. Early on (2003), the term was explained as follows:

Essentially, SOA is a *software architecture* that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls. SOA would be better-named *interface-oriented architecture*. SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function.

Nowadays, Gartner provides a shorter definition for SOA that says:

Service-oriented architecture (SOA) is a *design paradigm* and discipline that helps IT meet business demands.

As can be seen from the above example, the meaning of the term evolved over the years. It has become very popular and it accrued a number of other meanings in different contexts. This can lead to unnecessary confusion, so this section attempts to distinguish the variations in the meaning of SOA that are relevant to the Reference Architecture, such as “Service Orientation”, “architectural style”, and similar.

There are at least three distinct meanings that became attached to the term “SOA”. Depending on the context, “SOA” may mean one of the following:

- **Paradigm** for analysis and design of information systems
- **Style** for creating models (typically for architecture, but also for design, or functional domains), or
- **Model** of a domain *created* using the above style.

The above distinction helps avoid some of the misunderstandings when discussing SOA-related topics and when trying to make sense of the existing publications of SOA, marketing literature and the media, where mixing of the meanings is common. As SOA is viewed in CEAF 2.0 as an important element for target Enterprise Architectures and related Roadmaps, it is useful to distinguish the meanings of SOA that are useful in that context.

The discussion of relevant terms is presented in the following subsections.

2.1.1 SOA as Paradigm and Service Orientation

In cases when SOA is meant to be a *paradigm*, it actually means **Service Orientation**, defined as follows:

Service-orientation is a *design paradigm* intended for the creation of solution logic units that are individually shaped so that they can be collectively and repeatedly utilized in support of

the realization of the specific strategic goals and benefits associated with SOA and service-oriented computing¹.

Service Orientation forms the conceptual basis for Service Oriented *architectures* and SOA-as-Style, but it is also applicable to a number of domains other than architecture. It is also applicable to business and functional models, where it can be used as the basis for productive analyses and specifications. To avoid possible confusion, when *SOA-as-paradigm* is meant, the term “Service Orientation” is used in this Reference Architecture.

2.1.1.1 Services in Service Orientation

Service in Service Orientation is the “solution logic unit” that has the following attributes:

- Services provide a well-defined, coarse-grained unit of functionality
- Services are autonomous, mutually independent building blocks, interactions among which can be used to obtain specific target function or process
- Services can be very complex, but their complexity is not visible to their consumers; internally services can be either Atomic or Composite (that is, built from other atomic or composite services), but in either case, the internals are encapsulated for the consumer of the service
- Consumers (or “clients”) of services are only aware of the contracts that services make publicly available and interact with the services using those contracts.

There are a number of kinds of Services depending on the type of functionality they provide:

- *Business Service* encapsulates a specific business operation and is technology-agnostic (from the point of view of their consumer).
- *Technical Service* that have no business meaning but provide reusable technical capabilities.
- *Process Service* represents a Business Process or its sub-process or a task (most often, reusable part), by encapsulating a sequence of operations; when doing so, it may involve - behind the scenes - a number of Business and Technical Services.

It is important to note that Service in Service Orientation does not directly map and should not be directly mapped to an application or a system. Even though no service can be effectively provided without underlying technical components (such as an application), the distinction between the two should be maintained.

2.1.2 SOA as Style

SOA can also be understood as a *style* of conceptualizing information systems in which the concept of *business-aligned service* is the main organizing concept. This style can be seen as the application of Service Orientation paradigm to a particular domain. Depending on the domain, there are variations of the meaning of the term:

- In the *business analysis and process re-engineering domains*, SOA as a style means that analysis and specifications of business functions and business processes are done using the concept of “service” as the unit of functional decomposition. Such an approach is valid regardless of whether the IT buildings blocks for realization of services exist or not, and regardless of whether the process in question is manual or not.

¹ Note that Gartner defines SOA as “design paradigm”, while Erl defines Service-Oriented as “design paradigm”, which produces two distinct terms for the same content.

- In the *architectural domain*, SOA is an *architectural style* in which the concept of business-aligned service and the contract that expresses it are the basic facets of architectural building blocks to be identified and related to one another.
- In *software design*, SOA typically means an approach (or a “software design principle” to *designing* applications as a set of loosely-coupled, autonomous components orchestrated to support an end-to-end business process, and *building* those applications using specific components implementing services and contracts that express them.
- In *software construction*, SOA typically means using specific standards (such as SOAP and WSDL) and technologies available (such as Java- or .Net-based) for realization of SOA-based designs.

When this meaning is intended in the document, the expression “Service Oriented style” is used to make the intended meaning explicit. The additional qualifications indicating the domain of application of the style (enterprise or solution architecture, business analysis, etc.) are added as needed.

2.1.3 SOA as (Architectural) Model

SOA is also used to refer to the result of *applying* Service Oriented style to a domain (most often, but not only – architecture) that results in a concrete model. The following definition captures this meaning:

SOA represents an implementation-agnostic architectural *model*. However, Web services currently provide the foremost means of implementing services.

The difference between Service Oriented style and SOA as model is analogous to the difference between a recipe for doing something and the outcome of applying that recipe.

Given that architecting systems produces models and architectures are communicated using models, this understanding of SOA is the main one in this document; consequently, when “service oriented architecture” is mentioned, it is intended to mean architecture (that is, a model) compliant with the Service Oriented style.

Service Oriented Architecture - when treated as architecture (i.e., a model) created using specific *architectural style* – can be characterized by adoption of the following architectural principles when specifying architectures:

- **Loose Coupling:** Services maintain a relationship that minimizes dependencies among them; they may retain awareness of each other but under no circumstances the knowledge of their respective internals
- **Standardized Service Contract:** Services adhere to an agreement about mutual interactions; the agreement is defined collectively by one or more service descriptions and related documents
- **Service Abstraction:** Beyond what is described in the service contract, services hide their internal logic and implementation from the outside world, including consumers of the service
- **Service Reusability:** Logic is divided into services with the intention of promoting reuse
- **Service Autonomy:** Services have control over the logic they encapsulate
- **Service Statelessness:** Services avoid retaining information between subsequent invocations and minimize retaining information specific to an activity in progress

- **Service Discoverability:** Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms
- **Composability:** Collections of services can be coordinated and assembled to form composite services.

2.1.4 SOA RA and CEAF

Using the distinctions provided above, it becomes easier to be clearer what “SOA” means in the context of this Reference Architecture. Given that this document accompanies CEAF 2.0 - an Enterprise Architecture framework, its natural area of concern is twofold: *models* for Architecture and Business domains together, when approached using the principles of Service Orientation as previously described. The following figure depicts this:

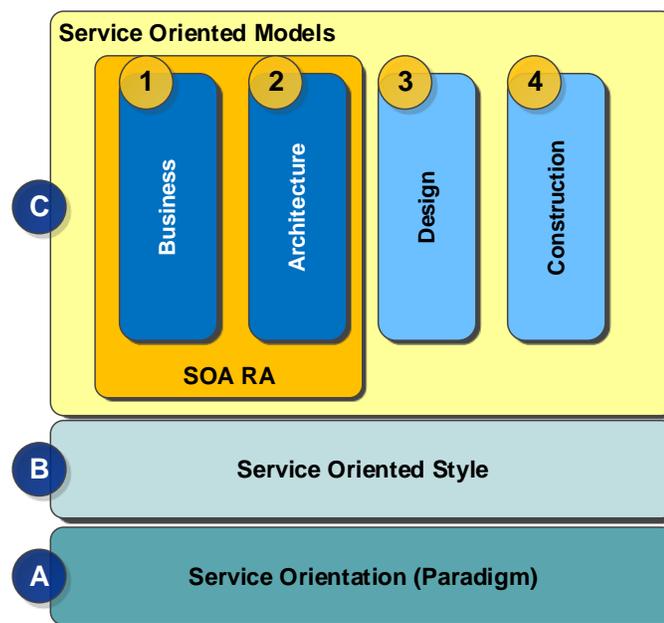


Figure 2-1 Reference Architecture and SOA in CEAF 2.0

The above figure shows the following:

- (A) Service Orientation (or SOA-as-paradigm), which provides key concepts for the remaining elements (as described in the section “SOA as Paradigm and Service Orientation”)
- (B) Service Oriented Styles (as described in the section “SOA as Style”):
- (C) Models as results of applying Service Oriented style (as described in the section “SOA as (Architectural) Model”)
 - a. Models of Business domains
 - b. Architectural models
 - c. Design models
 - d. Software Construction models.

As marked in the figure, the areas (C)(1) and (C)(2) mark the scope of “SOA” for this Reference Architecture:

- Service Orientation as applied to Business Domain, with focus on Business Process models

- Service Orientation as applied to Architecture, with emphasis on Services as architectural building blocks and mechanisms of orchestrating their interaction as a way of carrying out business processes.

2.2 Business Benefits of Adopting Service Orientation and SOA

Some of the benefits of successful adoption of Service Orientation in public sector are listed below:

- Increase organizational agility (ability to respond to change) in the face of changing business processes and regulatory constraints, by allowing the business to decide on the key drivers of solutions and allowing IT to rapidly implement required capabilities
- Reduce time to production by structuring business solutions based on a set of business and IT services in such a way as to facilitate the rapid restructuring and reconfiguration of the business processes and solutions that consume them
- Support intrinsic interoperability, or ability to share data within the organization due to common standardized contracts
- Increase vendor diversification options by allowing the organization to pick and choose “best- of-breed” vendor products and technology innovations and use them together within the enterprise
- Reduce or rationalize costs, by consolidating or eliminating redundant application functionality and by decoupling functionality (“services”) from obsolete and increasingly costly applications while leveraging existing investments
- Reduce IT burden, by making it possible to reduce duplication and redundancy, size, and operational cost
- Increase ROI, by enabling measuring of ROI for automated solutions
- Support federation, by sharing or re-using services (and potentially other resources) among state departments or agencies, while maintaining their individual autonomy; also, by integrating across silo-like solutions and organizations

Business and technology domain alignment can be significantly improved by adopting Service Orientation, because of the following benefits for the area:

- Service Orientation provides an intuitive and common language for communicating business-centered needs, namely the language centered on functional services
- It helps define the services that are required in terms of concrete functional needs, therefore narrowing the gap between the business and the technological perspectives in the organization
- It provides a way for assessing how these needs can be matched by services provided by existing or planned applications or systems
- It facilitates better understanding – for the business and technical stakeholders alike - of the status quo, the desired target and existing constraints

Applying Service Orientation style to business domain typically results in better functional specifications and better requirements than before. In case of architectural and design domains, it results in creating blueprints that are more adaptable and more resilient to change than traditional techniques and this alone makes it possible to shorten time to value.

There are areas, however, for which even successful introduction of service orientation and SOA should not be expected to provide immediate improvements:

- One of those is the expectation that adopting Service Orientation results in simple or markedly simpler than before implementations - especially in cases where legacy systems and their integration are concerned: on one hand, encapsulating legacy applications in order to mold them into the overall service oriented paradigm increases complexity. On the other hand, however, this encapsulating of legacy systems makes it possible to plan for orderly and timed substitution of the legacy system components that need replacing.
- Another area in which immediate benefits cannot be reasonably expected is growing the skills of subject matter experts, business analysts, designers and developers so that SOA can be successfully adopted in the organization. Growing such skills takes months or longer, even when effectively supported by training, mentoring and internal collaboration. Successful adoption of SOA, however, requires these skills in order to deliver business benefits

2.3 Main SOA Usage Scenarios

The following are common usage scenarios for SOA:

- Designing and deploying composite applications (or system of systems) as an assembly of services and service components
- Creating a Business Process in which process steps make use of available services – either internal to the organization, or externally available
- Encapsulating an external system or application as a locally available service or services
- Encapsulating a legacy application within the organization as a set of components providing a well-specified set of services

Please refer to the section “SOA RA Logical View” for description of representative interactions among architectural building blocks for the scenarios.

2.4 Key Capabilities of SOA Solution

The core capabilities of an SOA solution considered in CEAF are as follows:

- Provide for separation of Consumer Interfaces, Business Processes, Services, Services Components and supporting Enterprise Systems one from another
- Provide support for building distributed software that facilitates loosely coupled integration and resilience to change through the use of Services
- Support common interaction patterns (e.g., synchronous (request/response), asynchronous (fire and forget), and publish/subscribe) between service consumers and service providers
- Leverage the Enterprise Application Integration, Enterprise Information Integration, and Enterprise Decision Management Platforms to provide communication, mediation, information access and business rule enforcement functions
- Support integrated Single Sign-On (SSO), access management, directory services, and federated trust across heterogeneous systems by leveraging the enterprise Identity and Access Management platform
- Supporting standard (typically open) protocols and standards for interactions between components – especially in interactions between Consumer Interfaces, Business Processes and Services

- Support externalization and formal orchestration of business processes including collaborative workflows
- Interoperability among platform components

2.5 Components of SOA Solution

Applying Service Oriented style in the architectural domain can result in a number of architectures, which differ in the layering scheme they adopt and their respective responsibilities. Regardless of the particular arrangement, however, it is useful in the Enterprise Architecture context to distinguish between the SOA architectural components (even if they are considered at a logical level) and the components of other related architectures that enable their functions as shown in Figure 2-2 below.

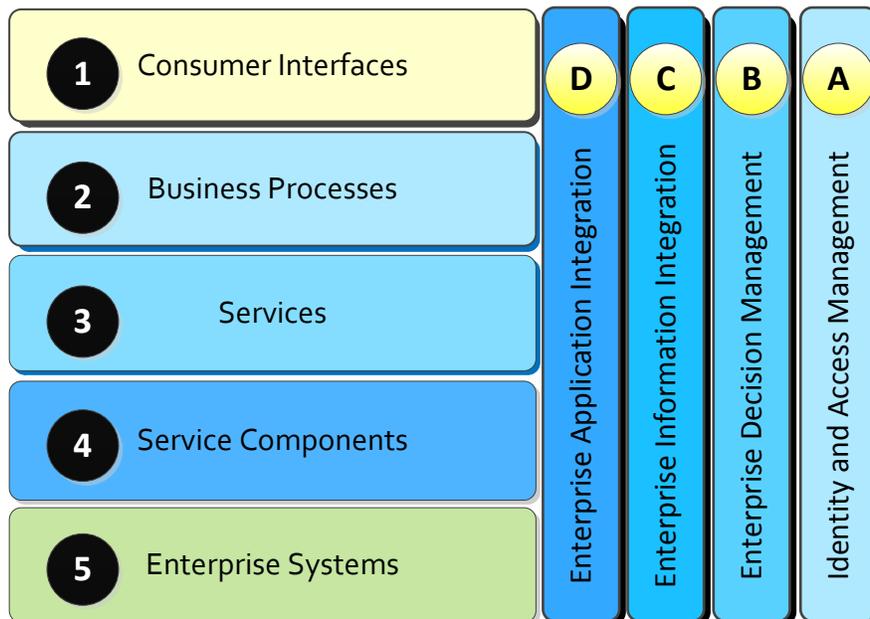


Figure 2-2 Groups of Architectural Components in SOA RA

The above figure shows the following components arranged vertically, in layers:

- (1) **Consumer Interfaces**, which provide the ability for consumers (human users, other systems, other SOAs, cloud service consumers, etc.) to interact with the SOA. It is the point of entry for interactive consumers and for service requests from external sources
- (2) **Business Processes**, which cover business process representation and composition, and provide building blocks for aggregating loosely-coupled services into sequencing processes aligned with business goals
- (3) **Services**, which provide units of functionality, *specified by contracts* about the interaction, and expressed as APIs or interfaces. Services, in the context of this SOA RA, can be thought of as *service descriptions* for business capabilities including contracts and policies. Service Components or existing enterprise applications (legacy systems, packaged applications, etc.) are responsible for the actual implementation or realization of a service. Service components are not visible to the consumers, but the *exposed service descriptions* are, and therefore from a consumer point of view, the *exposed service descriptions are the services* available for consumption.

- (4) **Service Components**, which consist of actual software components and/or existing enterprise applications implementing one or more services (i.e., service contracts)
- (5) **Enterprise Systems**, which are the existing enterprise applications/investments that contain service logic that can be exposed as services (through Wrapper Service Components and/or Service Implementation Adaptors etc.)

The capabilities of the above components are enabled by the supporting *platforms* (some of which are from other CEAF RAs), as follows:

- (A) **Identity and Access Management (IdAM)**, which provides the ability to manage identities, roles, access rights and entitlements, protected resource provisioning, run-time policy decision and enforcement services, and protect services from unauthorized access
- (B) **Enterprise Decision Management (EDM)**, which supports defining of business rules in application-agnostic terms and provides a standard way for accessing them from business processes and service components
- (C) **Enterprise Information Integration (EII)**, which provides the ability to integrate information across the enterprise in order to enable information services capability
- (D) **Enterprise Application Integration (EAI)**, is a key enabler for an SOA as it provides the capability to *mediate* which includes transformation, routing, and protocol conversion to transport service requests from the service requester to the correct service provider

The above figure is further developed in the section “SOA RA Conceptual View”. Here, the figure provides an overview and a context for the elements described in the following subsections.

2.5.1 Consumer Interfaces

Consumer Interfaces components provide entry points for various types of consumers of SOA, including human users, other systems or applications (service oriented or not), external sources (e.g., Business-to-Business (B2B) scenarios), other SOAs, and cloud service consumers. Consumer Interfaces provide for decoupling between the consumer and the rest of the underlying SOA, in order to achieve the following:

- Support agility and enhanced re-use
- Improve quality and consistency of consumer interactions

From business perspective, the importance of supporting multiple channels of interaction with the consumers is difficult to overestimate: in the public sector, a significant part of efficiency savings from adoption of SOA results from shifting away from traditional ways of interacting with service consumers (e.g., the public visiting government offices) to cheaper and more consumer-friendly electronic access.

The following diagram shows the main internal components of Consumer Interfaces in SOA:

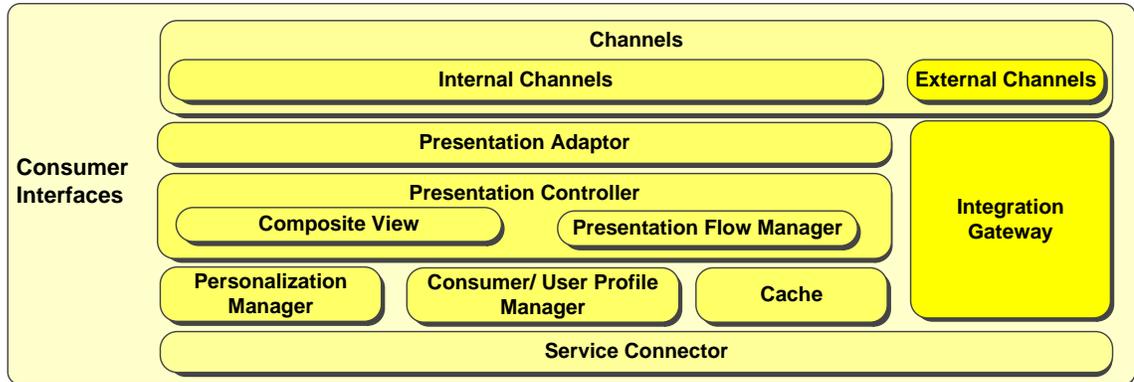


Figure 2-3 SOA Consumer Interfaces Layer Overview

The above figure shows the following components:

- **Channels** are distinctive types of platforms by which SOA consumers access services through the SOA. Examples of channels are web front-ends, rich clients, mobile applications, IVR (Interactive Voice Response) systems, and Electronic Data Interchange (EDI) messages which could all leverage the same core functionality within the SOA. Channels can be further characterized by the technology they use (such as JVM-based, .Net-based, raw sockets, etc.) and the manner in which they are used (always connected or sporadically connected, using a public kiosk or a private device, etc.).
 - **Internal Channels** are the channels internal to the organization (service provider) but external to the SOA
 - **External Channels** are the channels external to the organization (service provider). Service requests from these external sources (e.g., Business-to-Business (B2B) scenarios) are usually subject to additional perimeter security enforcement and are supported by federated identity and access management capabilities
- **Presentation Adaptor** is responsible for adapting presentation views to a given interaction channel
- **Presentation Controller** is responsible for managing the flow between presentation views and the composition of views into aggregates
- **Personalization Manager** is responsible for maintaining and applying user-specific choices or customizations to the views and user interaction
- **Consumer/User Profile Manager** is responsible for persisting user-specific personalization information and for supporting standardized method of accessing that information
- **Cache** stores pieces of data or implementation artifacts in order to improve overall performance of the system; the caching may use complex algorithms to determine what and for how long data needs to be cached
- **Service Connector** makes *services* easier to use by encapsulating the specifics of communicating with *services*. Connectors are responsible for discovering the addresses of services, establishing connections to the services, preparing requests, dispatching requests, receiving responses, transforming responses into client-specific local formats, and handling all service invocation related exceptions
- **Integration Gateway** serves as the ingress for service requests from external consumers, which are typically the transactions via the Internet. They can also be used to serve as the ingress for service requests from multiple security zones across the lines of business of an

organization. The primary purpose of the Integration Gateway is to secure third-party access to services and other internal applications. Integration Gateway interfaces with the Identity and Access Management platform to provide authentication and authorization, access control, digital signature processing, and encryption/decryption functions.

2.5.2 Business Processes

In the traditional (legacy) IT applications, business process flows were typically embedded in software components and user interfaces which constrained the ability to change the business processes as requirements changed. With the advent of SOA came the promise of agility and flexibility. An organization that has embarked on the journey of SOA would be successful in delivering the promise of agility and flexibility only when its business processes and associated flows are realized in the architecture in a fashion that allows rapid changes in the implementation of these business processes². This SOA RA recognizes the promise of SOA and allows *externalization of the business process flows* in a separate layer in the architecture and thus provides better ability to rapidly change the business processes as the requirements change.

The Business Processes cover the process representations and compositions, and provide building blocks for aggregating loosely-coupled services as a sequencing process aligned with business goals. Data flow and control flow are used to enable interactions between services and business processes within an enterprise or across multiple enterprises.

Business Processes components support the following:

- Defining business processes and their composites (i.e., composite business processes)
- Managing and persisting the definitions
- Instantiating and executing business process definitions at runtime
- Managing information flow in the context of business process
- Integrating capabilities provided by disparate components and services in order to realize business process
- Monitoring and management of process execution
- Logging, auditing, and other functions as required by security and compliance.

The following diagram shows the main components of Business Processes in this SOA RA:

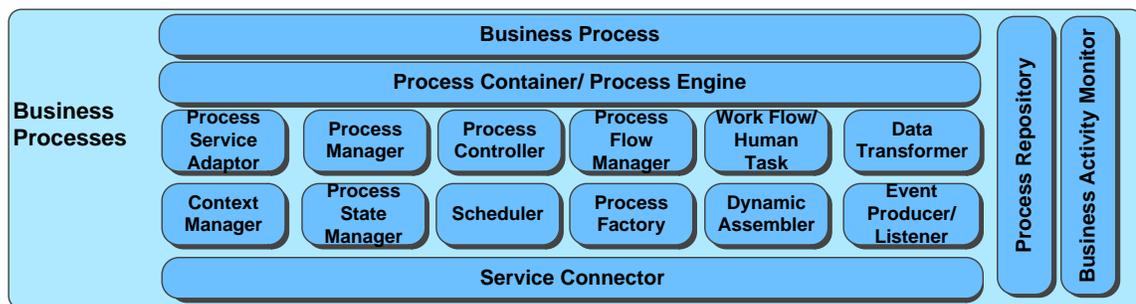


Figure 2-4 SOA Business Processes Layer Overview

The above figure shows the following components:

² TOGAF SOA Reference Architecture, http://www.opengroup.org/soa/source-book/soa_refarch.

- **Business Process** represents a process that a business performs. Business process is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm. Business Process, as a building block, as shown in the above figure is the *definition* that specifies - typically in implementation technology-agnostic manner - the end-to-end processing path of a business process that starts with a consumer request or an event and ends with a result for the consumer (or creation of business value)
- **Process Repository** stores business process definitions
- **Process Container/ Process Engine** provides an environment to manage the execution and flow of business processes (instantiated using a given business process definition) and to manage the interaction of humans with business processes. It is also responsible for managing the instances of running processes and their context.
- **Process Controller** is responsible for the execution of a single business process in a single session, and the interactions of the business process instance with other components involved in the execution of that instance (Context Manager, Flow Manager, etc.)
- **Process Manager** is responsible for deploying processes in the process container
- **Scheduler** is a component responsible for triggering the execution of business processes based on predefined time patterns
- **Process Factory** is responsible for instantiating a business process execution at runtime, given the process definition and input data and/or events
- **Dynamic Assembler** is responsible for invoking the appropriate end-point that needs to serve the request based on the context. Context provides the information that the Dynamic Assembler needs to make intelligent decisions on which end-point to be invoked
- **Process State Manager** provides the ability to retain and manage the process state during the execution of a business process. It manages the process state within each business process instance
- **Process Context Manager** manages the context of the various instances of the business process, which includes business data and technical, non-business elements required for the execution of the process
- **Business Activity Monitor** supervises running processes to produce logs and audits, and information about failures
- **Process Service Adaptor** provides the mechanism to expose business processes as services
- **Process Flow Manager** is responsible for creating process instances from process definitions using the *Process Factory* and for managing one or more process instances concurrently. A process instance is the representation of a single running business process. Process Flow Manager supports multiple interfaces that interact with business processes. It manages all the interactions and the decomposition relationships with the underlying sub-processes and enabling services
- **Workflow/Human Task** is a step or activity in a business process that involves interaction with human users, which usually involves various forms of input validation, often involving business rules
- **Data Transformer** is responsible for transforming data formats from source to target format, e.g., transforming data from a step or activity in a business process into the format necessary to invoke a service or the next process step
- **Event Producer and Listener** are responsible for generating and publishing events during the execution of a business process and for subscribing to events that may trigger a business process execution or alter its flow

- **Service Connector** makes *services* easier to use by encapsulating the specifics of communicating with *services*. Connectors are responsible for discovering the addresses of services, establishing connections to the services, preparing requests, dispatching requests, receiving responses, transforming responses into client-specific local formats, and handling all service invocation related exceptions

2.5.3 Services

Services are all the services defined within the SOA. They are the units of functionality, specified by contracts about the interaction, and expressed as APIs or interfaces. Services, in the context of this SOA RA, can be thought of as **service descriptions** for business capabilities including contracts and policies. Service Components or existing enterprise applications (legacy systems, packaged applications, etc.) are responsible for the actual implementation or realization of a service.

Services can be grouped in a number of ways:

- Based on the business meaning of the service (or absence of it), there are **Business** and **Technical Services**. Business Services directly provide for realization of a specific business function, such as *legal presence verification, address validation, eligibility determination, and income verification*. Technical Services provide supporting technical functions, such as *data access, persistence, and communication-related services*
- Depending on the type of the business function supported, Business Services can be further grouped as **Process Services** (supporting the execution of a business process), **Information Services** (having to do, typically, with search and retrieval of information), and similar. Note that interacting with Business Services typically makes use of Enterprise Business Objects.
- **Technical services** can be further classified according to their technical role, their granularity and composition, such as core/infrastructure services (such as persistence services) or utility services
- Depending on composition of the service, there are **atomic services and composite services**. Atomic services are not composed and not further divisible (as services). Composite services are composed of atomic and/or other composite services

The following diagram shows building blocks of Services Layers in SOA RA:

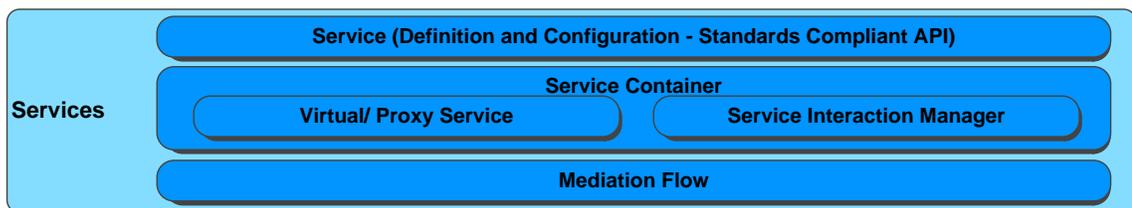


Figure 2-5 SOA Services Layer Overview

The above figure shows the following building blocks:

- **Service** represents an externally visible “**published service**” that offers certain functionalities the business performs to achieve a business outcome or a milestone. Externally visible does not necessarily mean external to the organization; it includes consumers internal to the organization or to a specific business unit within the organization. Service is the means by which the needs of a consumer are brought together with the capabilities of a provider. As

- such “service” represents a “definition”, typically represented in a standard description language (e.g., WSDL) describing its accessible interfaces (e.g., function or method signatures). The interfaces encapsulate Service Component (which performs the actual functions made available by the API). **Service Configuration** provides support for service versioning, including mapping between a versioned contract and the service component that realizes it. Service is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm
- **Service Container** provides the environment with the ability *to invoke and run services* (manage their runtime invocation, lifecycle, etc.). A Service Container may be contained within an Integration Bus (a part of the EAI platform), in a separate Java EE environment or a .NET environment, within a hardware device, or on another platform as long as it provides the ability to support runtime invocation and running of services and integration with other platforms. Within the Service Container are the components that enable it to invoke and execute service components, and support integration. Service Container leverages the Service Repository and Service Registry to support service versioning and virtualization
 - **Virtual/ Proxy Services** are the intermediary services that the Service Container implements locally. They are the gateways through which service consumers are connected to the managed back-end service components. They look like the actual target services to the consumers and thus provide a layer of indirection which enables internal services/ service components to be added, upgraded, or replaced while minimizing the impact on service consumers. Virtual Services also allow a collection of internal services to be combined *to provide an external service* to a service consumer. They enable mediation to be done between the virtual services and the services they consume and/or the service components they invoke by offering a facade over those services/ service components
 - **Service Interaction Manager** is contained within the Service Container and in general manages the interactions required to the Service Components
 - **Mediation Flow** defines the flow of messages between service consumers and service components, allowing for the mismatches such as protocol and interface differences between the two to be handled. They allow the request messages to be validated (for format), enriched, transformed (into the format of the target service/service component), and routed to the service end-point (which is typically not exposed to external service consumers)

2.5.4 Service Components

The Service Components provide the implementation or realization for services and their operations. Service Components reflect the definition of the service they represent, both in terms of functionality and Quality of Service. They bind the service contract/specification to the implementation of the service. Service Components are hosted on platforms which support their execution.

Figure 2-6 below shows the typical components of Service Components. It should be noted that the IT Governance has a significant influence on the Service Components. The choice of implementation technology for Service Components, the decisions about the manner in which Service Components may or may not consume behaviors from other Service Components, and the decisions about where to place the integration logic are some of the examples of this influence. For example, the implementation options for a Service Component may include BPEL, a Session EJB, a POJO, a Message Broker Flow, a SOAP/CICS operation, etc. Some of these

alternatives may eliminate the need for some of the components shown in Figure 2-6. For example, consumers may be exposed to a *Virtual Service* and the realization of that service may be provided by a Mediation Flow (which runs in (the service container of) an Integration Bus) which invokes the service logic contained in an Enterprise System through an Adaptor (which is a part of the EAI platform or the Integration Bus) eliminating the need for *separate* Service Components (because their functionality is provided through the Integration Bus, its Adaptors/Connectors, and existing Enterprise Systems).



Figure 2-6 SOA Service Components Overview

The above figure shows the following building blocks:

- **Service Implementation Adaptor** passes on the service call to the Service Component or to the Enterprise Systems that contain the service logic
- **Service Component Invoker/ Controller** supports the invocation of the Service Components by the Services or the Integration Bus. This includes invoking the Service Implementation Binder to convert the in-bound call to the Service Component call and invoking the Service Implementation Adaptor to pass on the service call to the Service Component or to the Enterprise Systems that contain the service logic
- **Service Implementation Binder** provides the technical runtime plumbing needed to execute the required service component within its container
- **Service Component** is a software building block which provides the implementation or realization for one or more services and their operations. It provides a façade behind which IT is free to do what is wanted and/or needed; as a result, it encapsulates business-specific logic

2.5.5 Enterprise Systems

Enterprise Systems are the existing applications that contain *service logic* which can be exposed as services using applicable techniques (wrappers, adaptors, mediation flows etc.).

The following diagram shows the typical components of Enterprise Systems Layer:

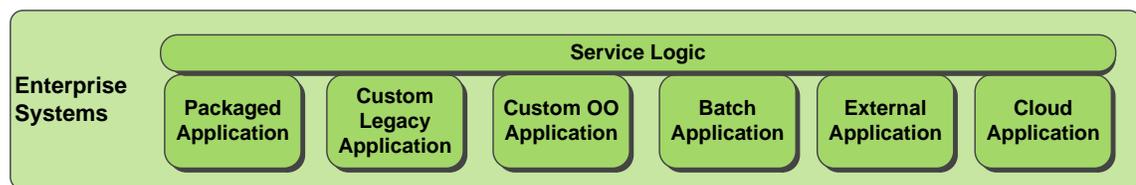


Figure 2-7 Overview of Enterprise Systems in SOA

The above figure shows the following types of Enterprise Systems:

- **Packaged Applications** are commercial software packages, such CRM, or ERP systems that are often tailored or customized (through configuration) to meet specific needs

- **Custom Legacy Applications** are usually applications developed in-house using legacy technologies such as mainframe-centric technologies
- **Custom OO Applications** are developed using Object Oriented technology (most often, using Java EE- or .Net-based technologies), which typically require specialized application containers to run (such as Java EE or Servlet containers in case of Java EE-based technology)
- **Batch Applications** are typically developed in-house and are executed at pre-defined schedules. These applications may contain service logic that can be exposed to support real-time or scheduled execution of business processes
- **External Applications** are externally deployed applications that are accessible to the organization; these types of applications typically introduce constraints in testing, scalability and Quality of Service for their consumers
- **Cloud Applications** are applications deployed in the Cloud including Software-as-a-Service (SaaS) which contain service logic accessible to other on-premises or hosted systems. The Cloud need not be public (please refer to the “CEAF 2.0 Cloud Computing Reference Architecture” for discussion of the Cloud deployment models)

2.5.6 Enterprise Decision Management

The following figure shows the main components of Enterprise Decision Management which support SOA.

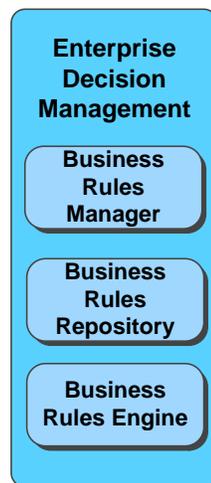


Figure 2-8 SOA Enterprise Decision Management Overview

Enterprise Decision Management Platform allows externalization of key business rules, and doing so provides a number of advantages:

- Adopting an Enterprise Decision Management Platform simplifies business rule development through business-oriented statements that encode business decisions.
- It provides an effective way to make rapid changes to the logic of systems/services.
- It allows for separation of business rules from core programming.

Components of Enterprise Decision Management Platform which support SOA include:

- **Business Rules Engine** is a generic component capable of evaluating available Business Rules when provided required context (data in predefined format).

- **Business Rules Repository** is responsible for persisting Business Rule definitions and for providing access to them to Business Processes, Services, and Service Components through the Business Rules Engine.
- **Business Rules Manager** is responsible for supporting Business Rules lifecycle, covering creation, modification, validation, and finally retiring of business rule definitions.

2.5.7 EAI and EII

The following diagram shows components of EAI which support SOA.

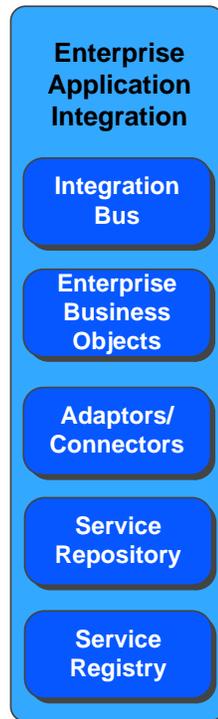


Figure 2-9 SOA EAI Platform Overview

Components of EAI Platform which support SOA include:

- Integration Bus (a single or federated set of Integration Busses as a part of the overall Enterprise Application Integration Platform)
- Enterprise Business Objects
- Adaptors/ Connectors
- Service Registry
- Service Repository

For more information about the above components and the other components of the EAI platform not shown here, please refer to the *Enterprise Application Integration Reference Architecture*.

The following diagram shows components of EII which support SOA.

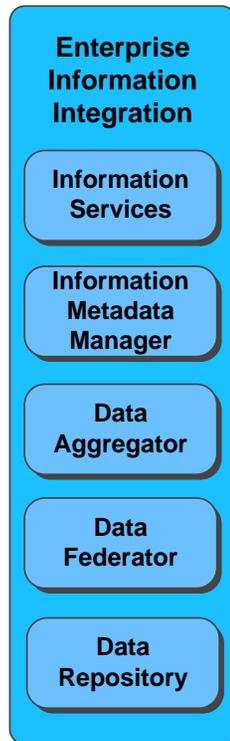


Figure 2-10 SOA EII Platform Overview

Components of EII Platform which support SOA include:

- Information Services
- Information Metadata Manager
- Data Aggregator
- Data Federator
- Data Repository

For more information about the above components and the other components of the EII platform not shown here, please refer to the *Business Intelligence Reference Architecture*.

2.5.8 IdAM Platform

SOA solution integrates with the Enterprise Identity and Access Management Platform in order to provide for the following key groups of functions:

- Authentication and authorization services
- Policy definitions and their enforcement

The following figure shows elements of IdAM Platform which support SOA:

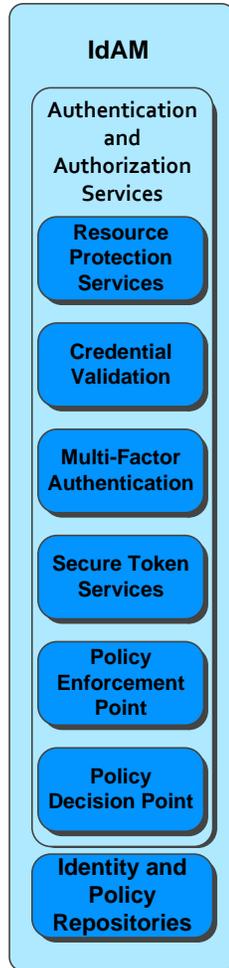


Figure 2-11 SOA IdAM Platform Overview

Components of IdAM Platform which support SOA include:

- Resource Protection Services
- Credential Validation
- Multi-Factor Authentication
- Secure Token Services
- Policy Enforcement Point
- Policy Decision Point
- Identity and Policy Repositories

For more information about the above components and the other components of the IdAM platform not shown here, please refer to the *Identity and Access Management Reference Architecture*.

3 SOA Reference Architecture Description

This section describes SOA Reference Architecture (RA) using three views:

- *Conceptual View*, which provides a summary of logical-level building blocks for SOA as presented in the Section 2 above
- *Logical View*, which provides an overview of relationships and interactions between components in an SOA solution for a specific usage scenario
- *Deployment View*, which illustrates the distribution of processing and components across nodes in the system

Each of the above views is presented in the subsections that follow.

3.1 SOA RA Conceptual View

The following Conceptual View figure brings together all major components of an SOA solution that have already been described in the section “Components of SOA Solution”. The diagram shows key SOA components organized in subsequent horizontal layers. The layers placed vertically (such as Enterprise Application Integration) represent components that enable the functions of the components in the horizontal layers.

Service-Oriented Architecture (SOA) Reference Architecture (RA)

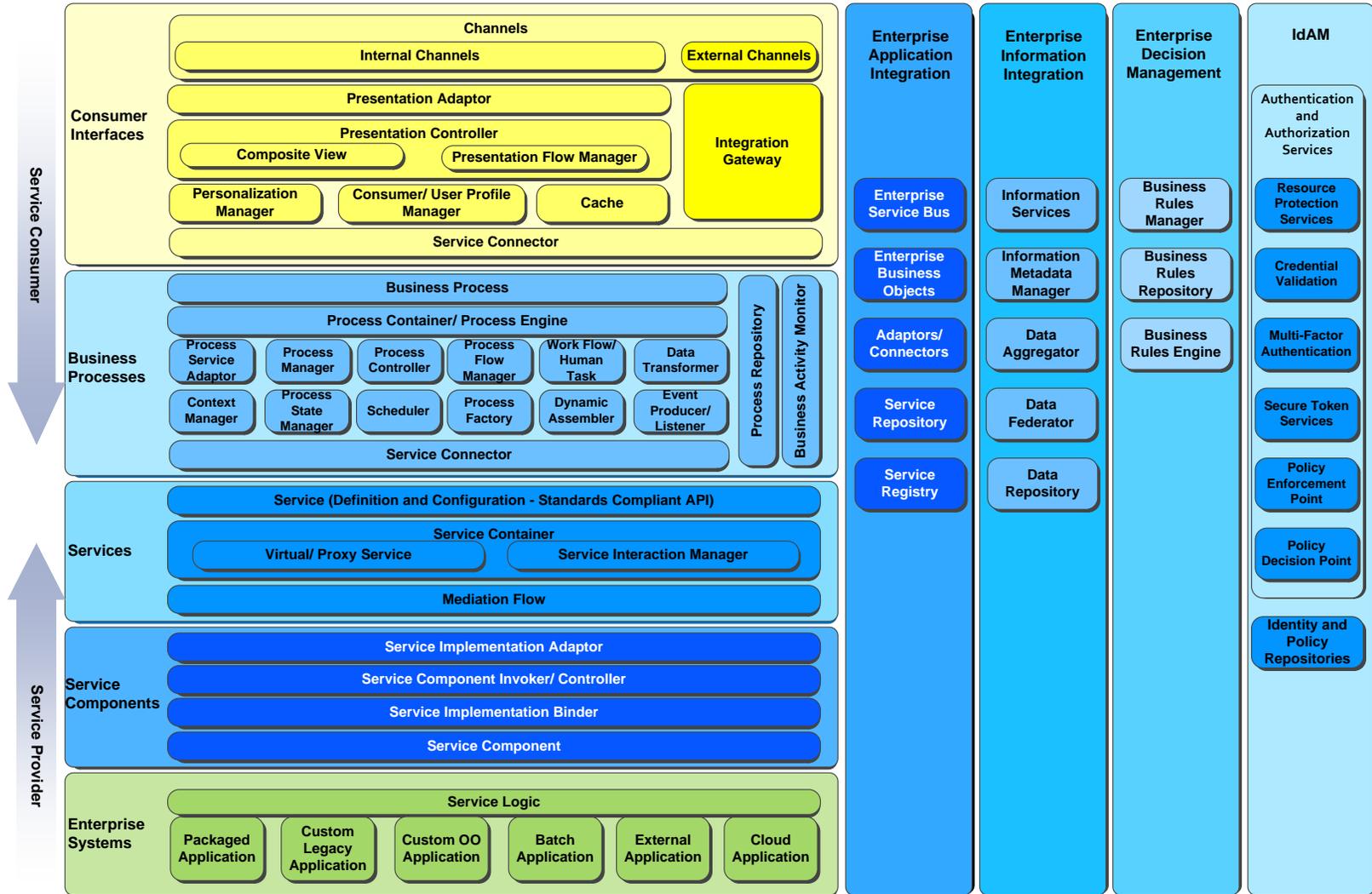


Figure 3-1 SOA Reference Architecture – Conceptual View

3.2 SOA RA Logical View

This section shows relationships and main interactions between components of SOA. First, high-level interactions are presented. They are followed by a specific SOA scenario to illustrate logical interactions among the various SOA components to realize the specific scenario.

It should be noted that the interactions shown in this section are logical abstractions representing typical interactions. In practice, however, due to the physical packaging of SOA logical components and services in the *specific system software products* selected, the component interactions (and also the names of components and services) may change. Therefore, this section is intended to enable the reader understand how SOA components logically realize a given scenario or use case, and use that knowledge to evaluate specific technology choices to provide desired capabilities.

3.2.1 High-Level Interactions

The following diagram shows high-level typical interactions among components of an SOA solution:

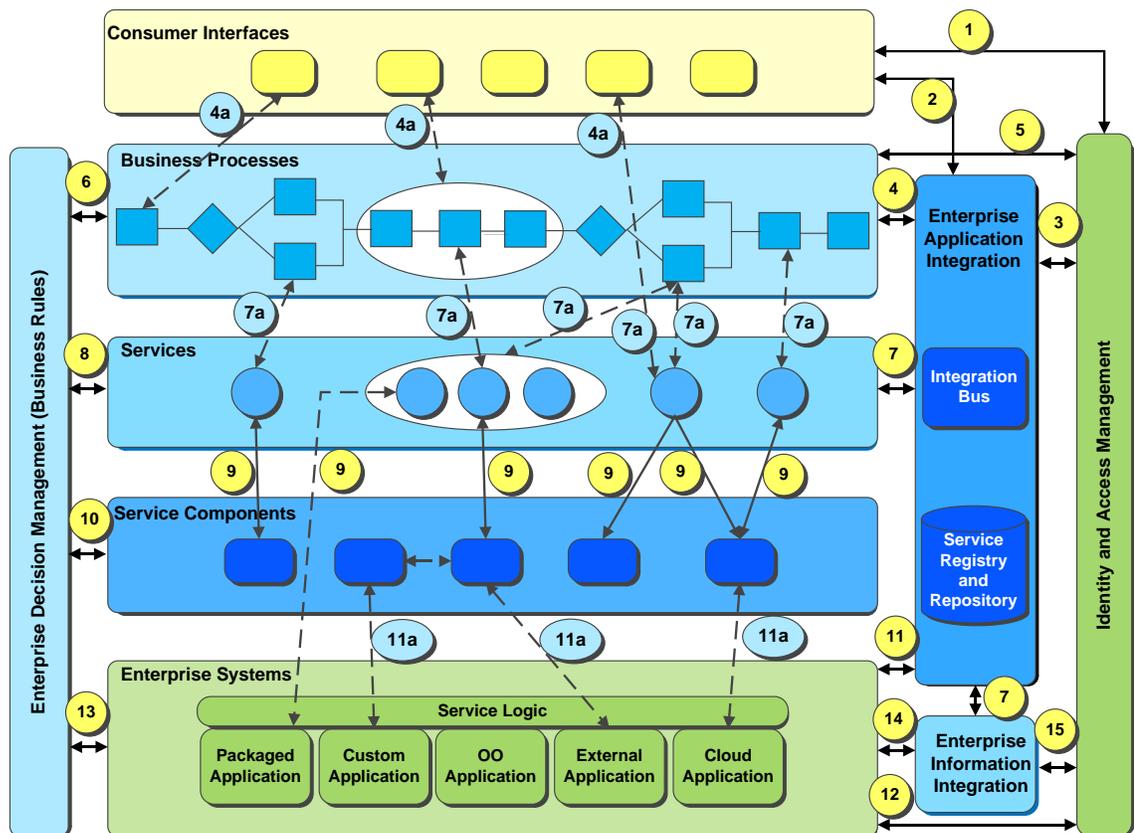


Figure 3-2 SOA Component Interactions

In the above diagram, continuous lines show direct interactions, whereas dashed lines indicate indirect interactions (i.e., through the EAI platform). While interactions with services (and service components) are preferable through the EAI platform, in practice, direct interactions are (technically) possible. Such direct interactions should be limited to situations where the following points apply:

- When a local service consumer interface (or application) components and the service components are located in the same container (i.e., there is no inter-machine communication) and security mediation, indirection, protocol conversion and mediation are not required
- When an organization's or a business unit's services are consumed locally by that organization's or business unit's consumer interfaces and security mediation, indirection, protocol conversion and mediation are not required. Additionally cross aspects such as logging and monitoring are addressed by the runtime container of service components

The following table summarizes interactions portrayed in the above diagram.

Table 3-1 High-level interactions among SOA components

Label	Description
(1)	<i>Consumer Interfaces</i> interact with <i>Identity and Access Management Platform</i> for user authentication and initial authorization.
(2)	<i>Consumer Interfaces</i> send request for a <i>Business Process</i> (available as a <i>Service</i>) or a <i>Service</i> (Atomic or Composite) through the <i>EAI Platform</i> (please refer to the points related to direct interactions described earlier in this section).
(3)	<i>EAI Platform</i> interacts with <i>Identity and Access Management Platform</i> for <i>Business process</i> or <i>Service</i> level authorization decision. Alternately the <i>Identity and Access Management Platform</i> may intercept the request and enforce this authorization.
(4)	<i>EAI platform</i> invokes the <i>Business Process</i> as a service.
(4a)	This is an indirect interaction between the <i>Consumer Interfaces</i> and the <i>Business Process</i> . In some SOA deployments, <i>Consumer Interfaces</i> may directly invoke the <i>Business Process</i> or a <i>Service</i> (please refer to the points related to direct interactions described earlier in this section).
(5)	<i>Business Process</i> layer interacts with <i>Identity and Access Management Platform</i> for fine-grained process/task level authentication decisions.
(6)	<i>Business Process</i> layer interacts with <i>Enterprise Decision Management Platform (Business Rules Engine)</i> for business rule execution (to direct the business process control flow).
(7)	<i>Business Process</i> layer invokes atomic or composite <i>Services</i> (mapped to process steps) or <i>Information Services</i> (exposed through <i>EII Platform</i>) through the <i>EAI Platform</i> .
(7a)	This is an indirect interaction between the <i>Business Processes</i> and <i>Services</i> . In some SOA deployments, <i>Business Processes</i> may directly invoke the <i>Services</i>

	(please refer to the points related to direct interactions described earlier in this section).
(8)	<i>Services</i> (virtual services or the mediation steps) interact with <i>Enterprise Decision Management Platform (Business Rules Engine)</i> for business rule execution (typically to direct the control flow of the composite services).
(9)	<i>Services</i> invoke <i>Service Components</i> through the Service Interaction Manager and Service Implementation Adaptor.
(10)	<i>Service Components</i> (optionally) interact with <i>Enterprise Decision Management Platform (Business Rules Engine)</i> for business rule execution.
(11)	<i>Services or Service Components</i> invoke the service logic encapsulated in the existing operational systems through the <i>EAI platform</i> .
(11a)	This is an indirect interaction between the <i>Services/ Service Components</i> and <i>Operational Systems</i> . In some SOA deployments, <i>Service Components</i> may directly invoke the <i>Service Logic</i> contained in the <i>Operational Systems</i> , but it is not a recommended practice (please refer to the points related to direct interactions described earlier in this section).
(12)	<i>Enterprise Systems</i> interact with <i>Identity and Access Management Platform</i> for security token validations and/or fine-grained authorization decisions.
(13)	<i>Enterprise Systems</i> interact with <i>Enterprise Decision Management Platform (Business Rules Engine)</i> for business rule execution.
(14)	<i>Enterprise Systems</i> interact with <i>EII Platform</i> for information services and data federation services.
(15)	<i>EII Platform</i> interacts with <i>Identity and Access Management Platform</i> for fine-grained authentication decisions.

3.2.2 Business Process-Driven Interactions with Services

The Business Process Layer of this SOA RA covers the representations and compositions of business processes, and provides the building blocks for aggregating loosely-coupled services into a business process to fulfill customer requirements. The Business Process Layer leverages the Services Layer to compose and choreograph services. Interactions between services and business processes are enabled by data flows and control flows. These interactions may exist within an organization or across multiple organizations. Figure 3-3 shows the typical interactions involved in executing a Business Process as a Service (BPaaS) in an SOA environment.

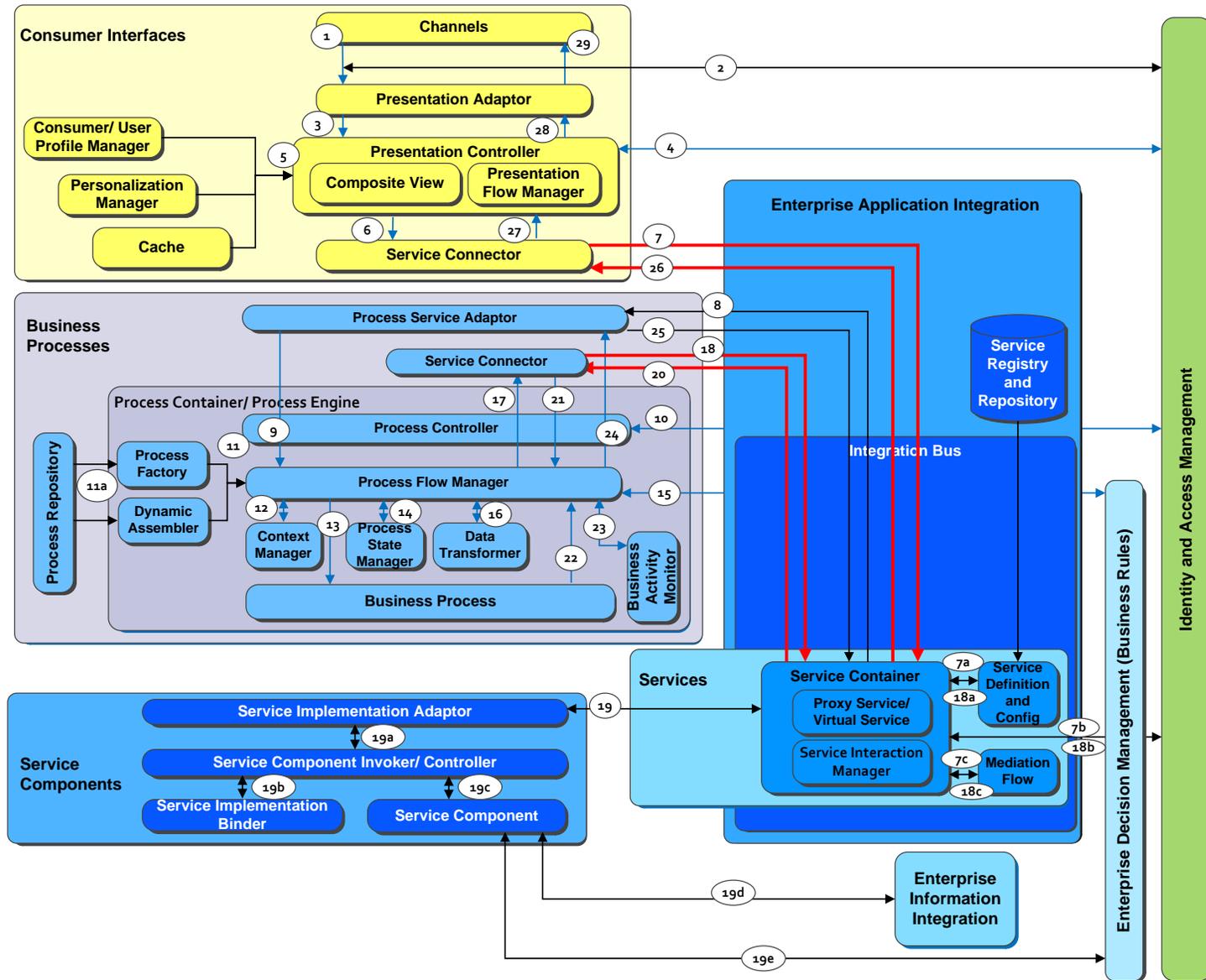


Figure 3-3 Typical Interactions during Execution of Business Process as a Service

The following table describes the interactions shown in the above figure:

Table 3-2 Interactions during Execution of Business Process as a Service

Label	Description
(1)	A user request through appropriate <i>Channel</i> is sent to the <i>Presentation Adaptor</i> .
(2)	The <i>Resource Protection Services</i> component of the Identity and Access Management Platform intercepts the user access request and verifies if the user has already authenticated and has an active session. If the user has an active session, user request is sent to the <i>Presentation Adaptor</i> . If the user does not have an active session, then the Identity and Access Management platform collects user credentials, authenticates the user, and then sends the user request to the <i>Presentation Adaptor</i> .
(3)	<i>Presentation Adaptor</i> invokes the <i>Presentation Controller</i> .
(4)	<i>Presentation Controller</i> interacts with <i>Identity and Access Management Platform</i> for fine-grained authorization decisions.
(5)	<i>Presentation Controller</i> obtains user profile information, cached information and user preferences, if any.
(6)	<i>Presentation Controller</i> (or the application component (not shown in figure)) sends a service request to the Business Process made available as a Service through the <i>Service Connector</i> .
(7)	The <i>Service Connector</i> invokes the Business Process corresponding to the user request as a Service through the <i>EAI Platform</i> .
(7a)	The <i>Integration Bus</i> (of the EAI Platform) initiates the Service Execution. The <i>Service Container</i> retrieves the <i>Service Definition and Configuration</i> and instantiates a local <i>Virtual Service</i> or <i>Proxy</i> to execute the <i>Mediation Flow</i> .
(7b)	The <i>Service Container</i> interacts with <i>Identity and Access Management Platform</i> for Business process level authorization decision. Alternately the <i>Identity and Access Management Platform</i> may intercept the request and enforce this authorization.
(7c)	The <i>Service Container/ Virtual Service</i> begins the execution of the <i>Mediation Flow</i> .
(8)	The <i>Service Container</i> invokes the <i>Business Process</i> as a service through the <i>Service Interaction Manager</i> . The <i>Process Service Adaptor</i> is triggered in this invocation.
(9)	The <i>Process Service Adaptor</i> invokes the <i>Process Engine</i> . This adaptor may use the <i>Data Transformer</i> to transform the request data to process specific data. The <i>Process Flow Manager</i> is triggered in this invocation.
(10)	The <i>Process Flow Manager</i> interacts with <i>Identity and Access Management Platform</i> for authorization decisions (during process initiation and execution).

Label	Description
(11), (11a)	The <i>Process Flow Manager</i> manages the execution of the business process. It creates an instance of the business process (if not already created for the current session) using the <i>Process Factory</i> and/or the <i>Dynamic Assembler</i> .
(12)	The <i>Process Flow Manager</i> uses the <i>Context Manager</i> to create or update the <i>Process Context</i> during the execution of the Business Process. This Process Context contains the data elements and other contextual information necessary to direct the execution of the Business Process.
(13)	The <i>Process Flow Manager</i> starts (or resumes) the <i>Business Process</i> execution.
(14)	The <i>Process Flow Manager</i> uses the <i>Process State Manager</i> to create or update the <i>Process State</i> during the execution of the Business Process.
(15)	The <i>Process Flow Manager</i> interacts with <i>Enterprise Decision Management Platform (Business Rules Engine)</i> for business rule execution (to direct the business process control flow).
(16)	The <i>Process Flow Manager</i> uses the <i>Data Transformer(s)</i> to transform the process context data in preparation for invoking a Service.
(17)	The <i>Process Flow Manager</i> calls the <i>Process Controller</i> to invoke a Service. <i>Process Controller</i> invokes the requested service through the <i>Service Connector</i> .
(18)	The <i>Service Connector</i> invokes the requested service through the EAI Platform. The <i>Service Container</i> (of the Integration Bus) receives this request.
(18a)	The <i>Integration Bus</i> (of the EAI Platform) initiates the Service Execution. The <i>Service Container</i> retrieves the <i>Service Definition and Configuration</i> and instantiates a local Virtual Service or Proxy to execute the <i>Mediation Flow</i> .
(18b)	The <i>Service Container</i> interacts with <i>Identity and Access Management Platform</i> for Service level authorization decision. Alternately the <i>Identity and Access Management Platform</i> may intercept the request and enforce this authorization.
(18c)	The <i>Service Container/ Virtual Service</i> begins the execution of the <i>Mediation Flow</i> .
(19)	The <i>Service Container</i> invokes the <i>Service Component</i> through the <i>Service Interaction Manager</i> . The <i>Service Implementation Adaptor</i> is triggered in this invocation which in turn executes the <i>Service Component</i> and sends the results from the service execution to the <i>Virtual Service</i> .
(19a), (19b), (19c), (19d),	Represent the interactions during the execution of the <i>Service Component</i> .



Label	Description
(19e)	
(20), (21)	The results from the execution of the Service are returned to the <i>Process Flow Manager</i> through the <i>Service Connector</i> and <i>Process Controller</i> .
(22)	Business Process execution completes.
(23)	The <i>Process Flow Manager</i> calls the Business Activity Monitor to record the process execution information.
(24), (25), (26)	Business process Execution results are sent back to the <i>Service Connector</i> .
(27)	The <i>Service Connector</i> sends the service response back to the <i>Presentation Controller</i> .
(28)	<i>Presentation Controller</i> invokes the <i>Presentation Adaptor</i> to provide response to the user request.
(29)	User receives response.

3.3 Deployment View of SOA RA

This subsection is to be completed in a future release. It is intended to show best-practice-based system topologies and implementation patterns of SOA, based on existing realizations of the SOA RA in the state.

4 Implementation Notes and Guidelines

This section is intended for SOA Reference Architecture-related notes and guidelines that can help when architecting SOA-based solutions, or when assessing such solutions in the context of a target Enterprise Architecture. This section is expected to grow over time as feedback is collected and experiences in the state agencies get analyzed. In the current version, the focus is on main SOA-related anti-patterns that should be identified and removed, to successfully incorporate SOA into target Enterprise Architectures.

4.1 SOA Patterns and Anti-Patterns

A “pattern” is a general, successful and reusable solution to a commonly occurring problem within a given context. There have been a number of SOA-related patterns identified and a number of papers and books devoted to them. These patterns fall into the following main groups:

- **Governance-level patterns** that include introducing SOA to the organization, supervising its implementation and assessing the outcomes
- **Business-level patterns** that involve business processes
- **Architecture-level patterns** are present in the fundamental organization of a system, including determination of components and their interactions
- **Application-level patterns** that pertain to technology-specific issues

Many (if not most) of the commonly described or most popular patterns are of the application-level type. These kinds of patterns are of interest primarily to the Application Architect or SOA Designer but less so to the Enterprise Architect. The opposite of the “pattern” is the “anti-pattern”; it can be described as a seemingly good solution that is *repeatedly used but known not to produce satisfactory* or acceptable outcomes. The usefulness of anti-patterns goes beyond designing and architecting; discovery and subsequent removal of anti-patterns can (and should) also be used for effective risk mitigation, e.g., when introducing SOA-based solutions to the organization.

The following table (based on [8], [9], and [10]) summarizes several common anti-patterns related to SOA. The table contains instances of governance-, business-, and architecture-related patterns. In cases where data is available, the pattern is assigned an indication of its frequency (or how likely it is to occur) and of the level of negative impact (ranging from low to “lethal”).

Table 4-1 Common SOA Anti-Patterns

Anti-Pattern Label	Description	Mitigation	Frequency	Negative Impact
Application Silos to SOA Silos	Services are built for use from the perspective of a single organizational unit – their definition and scope reflecting existing organizational silos. Absent central authority to promote shared specification and reuse which perhaps is the most pervasive counter-force to EA.	<ul style="list-style-type: none"> Doing SOA right requires target Enterprise Architectures that span multiple organizational units and involve maturing away from silos. 	High	Very High
Business Process Forever	Not planning for changing the existing definition of business processes in the organization and/or obscuring those business processes/rules within code. Conviction that well defined business processes are immutable.	<ul style="list-style-type: none"> Business sponsor and end-user involvement is necessary, and beyond the initial stage of defining the process. Allowing resources for business process modifications – most business processes do change over time. 	High	Very High
On-Line Only	Forcible elimination of batch-based systems as technologically obsolete.	<ul style="list-style-type: none"> Analyze business impact of elimination of batch systems. In many cases, these systems are necessary for the business. 	Low	Very High
Technology Altar	Current state of technology (and technology in general) driving specification of the target EA.	<ul style="list-style-type: none"> Target Enterprise Architectures should be primarily driven by business considerations. In case of technology shift, plan replacing potentially affected components, depending on what are the business services they provide. 	High	Very High

ESB is SOA	Identifying SOA with Enterprise Application Integration (EAI), or a subset of it (e.g., SOA = ESB + Web Services).	<ul style="list-style-type: none"> • Effective SOA realization requires an appreciation that SOA is more than an ESB deployment, and that it is not simply a matter of using Web services for interfacing between systems. • Set expectations clearly about which SOA strategies and technologies drive which benefits. 	Medium	High
No Legacy	Forcible elimination of all legacy components when introducing SOA, regardless of the impact on the business	<ul style="list-style-type: none"> • Analyze business impact of elimination of legacy systems • Encapsulate required legacy components as a component with well-defined services • Follow EA Roadmap to phase out legacy components 	High	High
Standardization Paralysis	Delaying solutions until respective standards appear or stabilize. Also, requiring compatibility with standards that undergo frequent modifications or are immature.	<ul style="list-style-type: none"> • Use mature standards whenever possible. • Avoid stopping or delaying of business functions solely for the reason of immature or unavailable standards. • Contain visibility of non-standard solutions to make future evolution possible. 	High	High
Misuse of Atomic Service	Insistence on assigning a single service to a single component.	<ul style="list-style-type: none"> • Architectural guidelines for mapping services to components are needed. A given component can realize multiple services. 	Low	Very High

5 Glossary

Applications Architecture defines the major kinds of applications or service components needed to support enterprise business functions and manage data.

Business Process is an end-to-end processing path that starts with a customer request or an event, and ends with a result for the customer or creation of business value.

Business Process Definition specifies a given Business Process, typically in implementation technology-agnostic manner, which can be executed by a Business Process Engine.

Reference Architecture models the abstract architectural elements in the domain independent of the technologies, protocols, and products that are used to implement the domain.

Service Component is an actual application, program, or a subsystem providing implementation of a Service which is treated as a *contract*.

Service-Orientation is a design paradigm intended for the creation of solution logic units that are individually shaped so that they can be collectively and repeatedly utilized in support of the realization of specific strategic goals and benefits associated with SOA and service-oriented computing.

Service-Oriented Architecture has a number of meanings – see the section “Definitions”.

Service-Oriented Computing is a computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments.

Web Service is (1) a software concept for program-to-program communication and application component delivery, where software is treated as a set of services accessible over ubiquitous networks using Web-based standards and protocols; (2) a concrete software component, defined in a standard way (using WSDL), which can be accessed by another application in a standard way (using SOAP) using a standard protocol (e.g., HTTP).

6 References

6.1 Federal and State Documents

- A. State of California, California State Information Technology Strategic Plan, November 2004
- B. State of California, California Performance Review Report
- C. Chief Information Officers Council, Federal Enterprise Architecture Framework, Version 1.1, September 1999
- D. Chief Information Officers Council, A Practical Guide to Federal Enterprise Architecture, Version 1.0, February 2001
- E. Federal Enterprise Architecture Program Management Office, The Business Reference Model, Version 2.0, June 2003
- F. Federal Enterprise Architecture Program Management Office, The Service Component Reference Model, Version 1.0
- G. Federal Enterprise Architecture Program Management Office, The Technical Reference Model, Version 1.1, August 2003
- H. Federal Enterprise Architecture Program Management Office, The Data Reference Model, Version 1.0, September 2004

6.2 Books and Papers

- 1. T. Erl, "SOA Design Patterns", Prentice Hall PTR, 2009.
- 2. T. Erl, "SOA: Principles of Service Design", vol. 1. Prentice Hall Upper Saddle River, 2008.
- 3. S. Jones, "SOA Anti-Patterns: How Not to Do Service-Oriented Architecture", 2010.
- 4. N. M. Josuttis, "SOA in Practice: The Art of Distributed System Design", 2007.
- 5. P. Liegl, "The strategic impact of service oriented architectures", in Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the, 2007, pp. 475–484.
- 6. M. H. Ibrhaim, K. Holley, N. M. Josuttis, B. Michelson, D. Thomas, and J. Devadoss, "The future of SOA: what worked, what didn't, and where is it going from here?", in Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion, 2007, pp. 1034–1038.
- 7. Y. V. Natis, "Service-Oriented Architecture Scenario", Gartner, 2003.
- 8. J. Krai and M. Zemlicka, "The most important service-oriented antipatterns," in Software Engineering Advances, 2007. ICSEA 2007. International Conference on, 2007, pp. 29–35.
- 9. J. Ang, L. Cherbakov, and M. Ibrahim, "SOA Antipatterns", 2005, <http://www.ibm.com/developerworks/webservices/library/ws-antipatterns/>.
- 10. M. H. Dodani, "Patterns of Anti-Patterns", Journal of Object Technology, vol. 5, no. 6, pp. 29–33, 2006.
- 11. Oracle, "SOA anti-patterns: How Not to Do Service-Oriented Architecture", Oracle White Paper in Enterprise Architecture, 2010.
- 12. Heather Kreger et al., "The IBM advantage for SOA reference architecture standards", 2012, <http://www.ibm.com/developerworks/webservices/library/ws-soa-ref-arch/index.html>.
- 13. Robert Daigneau, "Service Design Patterns", Addison Wesley, 2012.

6.3 Referenced Web Sites and Web Resources

- a. California State Chief Information Officer, California Information Technology Council, Committees <http://www.cio.ca.gov/ITCouncil/Committees/Committees.html>



- b. Gartner IT Glossary, <http://www.gartner.com/it-glossary>
- c. TOGAF SOA Reference Architecture, http://www.opengroup.org/soa/source-book/soa_refarch.
- d. Steve Jones, SOA anti-patterns, <http://www.infoq.com/articles/SOA-anti-patterns>.



7 Document History

Table 7-1 Document History

Release	Description	Date
Version 1.0 Draft	Initial creation	06/19/2013
Version 1.0 Second Draft	Revised based on internal review comments	06/21/2013
Version 1.0 Final Draft	Addressed EAC review comments	10/21/2013
Version 1.0 Final	Final version	01/02/2014