Service Oriented Architecture
CIO Guide
Roman Hayer, SAP AG
2010
This document is intended to outline future product direction, and is not a commitment by SAP to deliver any given code or functionality. Any statements contained in this document that are not historical facts are forward-looking statements. SAP undertakes no obligation to publicly update or revise any forward-looking statements. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. The timing or release of any product described in this document remains at the sole discretion of SAP. This document is for informational purposes and may not be incorporated into a contract. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

The document is still under construction and has been added to the starter kit as preview. The final SOA CIO Guide is planned for end of 2010.
This document introduces reference architectures for various SOA Process Pattern in Service Oriented Architecture based on SAP products.

The reference architectures aim at companies that already have deployed the SAP Business Suite (or parts of it) and want to deploy or extend their footprint in the SOA area. The document addresses multiple reader groups with slightly varying targets:

- CIOs will be supported to translate business requirements into IT strategic decisions.
- Enterprise Architects will be supported to plan the overall system architecture and derive/optimize project specific variants.

The reference architectures represent SAP’s view on the current state of technology as well as sustainable ways going forward and should guide the customer on the way forward with SAP solutions.
Table of Contents

- **Definitions & Value Proposition**
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- SAP Product Mapping & Implementation Guidance
- Business & Technology Trends

----------------- Appendix -----------------------
- Architecture Pattern Catalogue
Definitions & Value Proposition

Definition SOA

- **Service-Oriented Architecture (SOA)** is an architectural style that supports service orientation.
- **Service orientation** is a way of thinking in terms of services and service-based development and the outcomes of services.
- A service is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports), is self-contained, may be composed of other services, is a “black box” to consumers of the service.
- An architectural style is the combination of distinctive features in which architecture is performed or expressed.
- The **SOA architectural style** has the following distinctive features:
  - It is based on the design of the services – which mirror real-world business activities – comprising the enterprise (or inter-enterprise) business processes.
  - Service representation utilizes business descriptions to provide context (i.e., business process, goal, rule, policy, service interface, and service component) and implements services using service orchestration.
  - It places unique requirements on the infrastructure – it is recommended that implementations use open standards to realize interoperability and location transparency.
  - Implementations are environment-specific – they are constrained or enabled by context and must be described within that context.
  - It requires strong governance of service representation and implementation.
  - It requires a “Litmus Test”, which determines a “good service”.

Accelerating Innovation

- **Product and service innovations**, which drive top-line growth through increased volume or prices
- **Process innovations**, which help lower costs to raise the bottom line
- **Technology innovations**, which help IT respond more quickly to changing business requirements

Enhancing Operational Excellence

- **Reliability**
- Support of rapid business and organizational change
- Delivery of predictable results with high quality

Empowering the Information Worker

- Automating common processes related to specific business roles
- Providing collaborative work environments for improved communication
- Improving access to business information

Definitions & Value Proposition

SOA Value Proposition

Accelerate Innovation

- **Innovation**

Enhance Operational Excellence

- **Operational Excellence**

Empower Information Worker

- **Information Worker**
Table of Contents

- Definitions & Value Proposition
- **Solution Space & Key Capabilities**
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- SAP Product Mapping & Implementation Guidance
- Business & Technology Trends

----------------- Appendix -------------------

- Architecture Pattern Catalogue
The following partitioning of SOA capabilities does neither reflect a lifecycle nor an architecture layer view. It is used to introduce the major capabilities that are required to adapt an IT landscape to the SOA paradigm.

**PLAN**
Comprises capabilities, tools and methodologies to document, analyze, plan and optimize the enterprises business process evolution, including manual and automated processes and governance over time.

**BUILD**
Provides the environment to model, develop and test the automated process support based on the planning in iterative steps.

**RUN**
Enumerates the capabilities that are required to run the planned and built automated process support by IT infrastructure.

**MANAGE**
Comprises all tools and procedures to manage the running automated business processes.
Solution Space & Key Capabilities

Capability Overview - PLAN

- **Enterprise Modeling**
  - **Enterprise Architecture**
    - Comprises all activities to define, analyze and optimize the system landscape and the interoperability protocols between systems
  - **Organizational Planning**
    - Structures - Standards, tasks and roles (e.g. SOA competence team), etc.
    - Processes - BPM, service management, etc.
    - People - Skill and knowledge management, etc.
  - **Governance**
    - Across all other topics to make results sustainable

- **Methodology**
  - Enterprise Architecture Framework, TOGAF*, ...

- **Portfolio Management**
  - Transparent investment decision process
  - Ensure balance between strategic priorities, market, external opportunities, architectural innovation and available resources

*http://www.opengroup.org/togaf/
Solution Space & Key Capabilities
Capability Overview – BUILD

Integrated Development Environment
Domain specific Designtime Editors
Designtime Repository
Integrated Test Environment

Provides the environment to model, develop and test the automated process support based on the planning in iterative steps.
Solution Space & Key Capabilities
Capability Overview - RUN

Enumerates the capabilities that are required to run the planned and built automated process support by IT infrastructure.

Service Consumption
- Human Interaction Channels
- System Interaction Channels
- Business Process Management
- Business Rules Management

Service Integration
- Service Discovery and Virtualization
- Mediation (Connectivity, Adaptation, Routing, Mapping)
- Event Processing
- Business Rules Management

Service Provisioning
- Service Inventory
- Encapsulation
- Eventing
**Solution Space & Key Capabilities**

**Capability Overview - RUN - Service Consumption**

- **System** acts as placeholder for the non human interaction channel. Services can be consumed by either company internal applications (A2A) or participate in business networks (B2B).
- **User** denotes the human interaction channel, which is exposed via user interfaces.
- **UI Framework** represent an environment to create a user centric interaction channel with services.
- **Consumption UI** - is typically deployed to a consumption system as part of a composite application and connects the Composition Logic with the user.
- **Consumption Logic** - comprises all development artifacts, which are deployed on to the consumption server to automate the consumption process.
- **Process Engine** enables subsequent processing rules in a structured and easily adaptable way.
- **Rules engine** is the runtime of a intuitive and thus highly productive capability to express business rules to support decisions within consumption scenarios. Agility to adapt business changes is one of their key benefits.

Service Consumption provides capabilities to connect **users** and other **systems** or **applications** to the service providers.
Solution Space & Key Capabilities

Capability Overview - RUN - Service Integration

- **Responsibility Determination** is a service for routing or event processing to determine the receiver of a request or message.
- **Event Processing** supports detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.
- **Service Discovery and Virtualization** drives decoupling between consumer and provider from design-time to runtime.
- **Service Registry** provides publication and discovery functionalities.
- **B2B Gateway** enables integration with standardized protocols like RosettaNet, etc.
- **Messaging** denotes the basic infrastructure to move a message through various integration processing steps.
- **Mapping & Transformation** of data structures must be provided to ensure semantical alignment between consumer and provider.
- **Routing** of messages must be performed. Can also lead to copies, splits or joins of messages to handle transport to multiple consumers.
- Technical **connectivity** must be ensured, if consumer and provider reside on different systems supporting different standard or legacy protocols.
- **Rules engine** offer intuitive and thus highly productive capabilities to model business rules to support decisions within integration and workflow scenarios. Agility to reflect business changes is one of their key benefits.
- **Process Engine** enables subsequent processing rules in a structured and easily adaptable way.

Service Integration mediates decoupled and bidirectional communication between Consumer and Provider on syntactical and semantical level.
Service Providers do have the capabilities to fulfill the needs of service consumers. Encapsulation must be done by service facades or external service integration.

- **Service Facade** describes the encapsulation of business semantics in reusable services, delivering dedicated Quality of Services.
- **Business Functionality** denotes the core business functionality exposed via proprietary legacy interfaces. This comprises on-premise OLTP and analytical systems as well as cloud provisioning of business functionality.
Solution Space & Key Capabilities
Capability Overview - MANAGE

Comprises all tools and procedures to manage the running automated business processes.

Software Logistics
Lifecycle Management
Monitoring
Technical Configuration
Business Configuration
Landscape Management
Process Analytics
Solution Space & Key Capabilities
Capability Overview - Details

PLAN
- Enterprise Modeling
  - Enterprise Architecture
- Organization
  - Structures
  - Processes
- People
- Governance
- Methodology
- Portfolio Management

BUILD
- Design Time Repository
- Integrated UI Environment
- Line Interface
- Processes
- Business Rules
- Authorization
- Service Definitions
- Transformation
  - UI
- Event Correlation
- Code

RUN
- Service Consumption
  - UI Framework
- Consumption UI
- Process Engine
- Rule Engine
- Service Integration
  - Mapping/Data Transformation
  - Routing
  - Policy Enforcement
- Publication/Subscription
  - Rules Engine
  - Service Registry
  - Event Handling
- Connectivity
  - EAI Gateway
- Messaging

MANAGE
- Software Lifecycle Management
- Application Management
- Service Management
- Monitoring
- Technical Configuration
- Business Configuration
- Process Analytics
- ...
Table of Contents

- Definitions & Value Proposition
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- SAP Product Mapping & Implementation Guidance
- Business & Technology Trends

------------------ Appendix ------------------

- Architecture Pattern Catalogue
Reference Architectures & Maturity Model
Architecture Definition Framework – Architecture Pattern

Description

- Decomposition of a typical SOA stack into 3 layers
  - Service Consumption
  - Service Integration
  - Service Provisioning

- For each layer we describe
  - Mutual exclusive Architecture variants for the system topology including quality assessments and constraints
  - Additional architecture options to add further qualities
Architecture Definition Framework

- Introduction of **SOA Process Pattern**
- SOA Process Pattern specific **Reference Architectures** based on
  - Functional Requirements
  - Quality Requirements
  - Design Constraints
- Quality based decision framework guides from Use case to reference architecture based on reference architectures
- **SAP Product Portfolio** mapping to architecture pattern to support make or buy decisions for Use Case specific architecture
Reference Architectures & Maturity Model
Architecture Definition Framework

1. Identify SOA Use Case
2. Select SOA Process Pattern
3. Derive Architecture Pattern
4. Fine tune Design Architecture

- Identify Functional Requirements, Quality Requirements, Design Constraints
- Select Functional Requirements, Quality Requirements, Design Constraints
- Derive Functional Requirements, Quality Requirements, Design Constraints
- Fine tune Design Architectures
Reference Architectures & Maturity Model
Architecture Definition Framework

**SAP**

A. describes a set of typical architecture pattern and assesses key qualities and drawbacks
B. categorizes typical SOA use cases into a manageable number of process pattern
C. creates architecture blueprints/reference architectures for a typical set of process pattern under typical constraints
D. maps the SAP product portfolio to the reference architectures

**Customer**

1. analyzes the use case to be implemented and identifies the best matching process pattern
2. selects the best fitting reference architecture by using his quality requirements and design constraints as a filter criterion
3. derives the use case specific architecture from the selected reference architecture
4. finishes the major architecture definition work by finetuning the architecture based on the underlying pattern framework
Ahhh, typical UI simplification on a single backend.

OK, here is the closest ref arch.

OK, need composition environment and Process Integrator.

To speed up, I add backend service adaptation.
Reference Architectures & Maturity Model

Notation

SAP Technical Architecture Modeling Notation (TAM)

Communication Channel
(R-arrow indicates the request direction, can also be unspecified or in both directions)

Communication Channel
(arrows indicate data flow)

Read/Write Access
Agent accesses Storage

Read Access
Agent reads from Storage

Agent
active part / component

Storage
passive part / component

Multiplicity
Entity occurs multiple times

Browser

Web Server

Static Page

Web Application

Datastore
Layer focus

- Whenever we describe one of the layers variants or options we use grey boxes to denote hosting systems and colored boxes to describe the capabilities or functional entities.

- As layers cannot be described completely independent from each other we use simplified representation of a whole layer to show the interaction of the currently described layer as colored boxes without surrounding grey boxes.
Reference Architectures & Maturity Model

Architecture Pattern Catalogue (Variants & Options)

Consumption
- Single System Consumption
- Asymmetric Consumption
- Symmetric Consumption
- Control Flow
- Replication
- Transaction Coordination

Integration
- Peer to Peer
- ESB Hub
- Distributed ESB
- Central Service Inventory
- Service Virtualization
- Control Flow

Provisioning
- Legacy Interface
- Service Enabled Interface
- Backend Service Adaptation

Pattern Catalogue (see Appendix)
Reference Architecture
Illustration of Architecture Construction

UI simplification with process flow across 2 service provider backends with legacy interfaces
SOA Process Pattern

Composition
- Backend Abstraction
  - Service enabling of backendsystem
- Dashboard
  - Information access to single or multiple systems
- UI Simplification
  - Role based UI Simplification on single system
- Backend Multiplexing
  - Role based UI Simplification on multiple systems
- Offline Client
  - Occasionally offline role based UI Simplification on single system
- Offline Multiplexing
  - Occasionally offline role based UI Simplification on multiple systems

Integration
- A2A Integration
  - Integration of internal systems to connect business activities and exchange information.
- B2B Partner Integration
  - Integration of systems across company boundaries based on common standards
- Dominated B2B Integration
  - Integration of systems across company boundaries with one dominating partner, e.g. manufacturer - supplier integration

"Transactionality"

Online transaction
- Staged/Offline transaction
- Information only

Key:
- Composition
- Integration

Reach of SOA Application Landscape

- Single systems
- Multiple internal systems
- Multi Company landscape
Each SOA Process Pattern might have multiple reference architectures depending on diverging constraints and requirements.

- Input describes all available information to take the architecture decisions
- Architecture Decisions lists all architecture decisions (following a →). Architecture decisions are typically trade-off* decisions!
- Architecture Blueprint shows the composed architecture blueprint with additional information about SAP products

Assessment is based on 5 major qualities
- End user performance
- UI Flexibility
- TCO
- TCD/required skills
- System Landscape flexibility

*http://en.wikipedia.org/wiki/Trade-off
SOA Process Pattern - Reference Architectures

UI Simplification - I

**Input**

- Functional Requirements
  - Role specific UI simplification of aggregation of 1 service provider system
  - Readwrite Access
  - Simple process flows required
  - Integration into other Uis of service provider backend required
- Context Assumptions
  - Service provider is not service enabled, and does allow modifications/add on development
  - Service provider backend offers process engine
- Quality Requirements
  - High end user performance requirements
  - Operational data access without latency
  - Developer skills focus on existing backend technologies
  - Development cost must be minimized

**Architecture Decisions**

- Service Consumption
  - Single service provider, internal UI integration ➔ use consumption possibilities of backend
  - Simple process flows required ➔ use backend process engine to control process flows
- Service Integration
  - Provider not service enabled ➔ Develop/model local mediation/adaptation to connect services to consumption logic
- Service Provisioning
  - Adapt proprietary application programming interface via mediation into required interface pattern

**Architecture Blueprint**

- Single service provider, internal UI integration
- Use consumption possibilities of backend
- Use backend process engine to control process flows
- Develop/model local mediation/adaptation to connect services to consumption logic
- Adapt proprietary application programming interface via mediation into required interface pattern
SOA Process Pattern - Reference Architectures

UI Simplification - II

**Input**

- **Functional Requirements**
  - Role specific UI simplification of aggregation of 1 service provider system
  - Readwrite Access
  - Simple process flows required
  - Integration into other Uis required

- **Context Assumptions**
  - Service provider service enabled, and does allow modifications/add on development

- **Quality Requirements**
  - High end user performance requirements
  - Operational data access without latency
  - Developer skills in modern programming languages (e.g. Java) available

**Architecture Decisions**

- **Service Consumption**
  - Single service provider, external UI integration ➔ use dedicated consumption frontend
  - Simple process flows required ➔ use consumption system with workflow engine

- **Service Integration**
  - Provider service enabled ➔ Mediation is done on frontend server
  - Single backend ➔ no transaction coordination required

- **Service Provisioning**
  - Provider service enabled ➔ Peer to peer connection possible

**Architecture Blueprint**

- WebDynpro Java
- NetWeaver BPM
- Composition Environment
- Mediation
- Service Facade
- Service Provider
- Provisioning Backend
- Business Suite

© SAP 2010 - Page 31
SOA Process Pattern - Reference Architectures

UI Simplification - III

**Input**

- **Functional Requirements**
  - Role specific UI simplification of aggregation of 1 service provider system
  - Readwrite Access
  - Simple process flows required
  - Integration into other UIs required

- **Context Assumptions**
  - Service provider is not service enabled, and does allow modifications/add on development
  - Consumption server will be connected to other backends as well

- **Quality Requirements**
  - Moderate end user performance requirements
  - Operational data access without latency
  - Developer skills in modern programming languages (e.g. Java) available

**Architecture Decisions**

- **Service Consumption**
  - Single service provider, external UI integration ⇒ use dedicated consumption frontend
  - Simple process flows required ⇒ use consumption system with workflow engine

- **Service Integration**
  - Provider not service enabled ⇒ introduce ESB as mediation hub
  - Single backend ⇒ no transaction coordination required

- **Service Provisioning**
  - Service Provider not service enabled ⇒ Deploy matching adapter on ESB

**Architecture Blueprint**
<table>
<thead>
<tr>
<th>Quality</th>
<th>Arch. Blueprint</th>
<th>I - Single System</th>
<th>II - Peer to peer</th>
<th>III ESB</th>
</tr>
</thead>
<tbody>
<tr>
<td>End user performance</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td></td>
</tr>
<tr>
<td>UI Flexibility</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td></td>
</tr>
<tr>
<td>TCO</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td></td>
</tr>
<tr>
<td>TCD/required skills</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td></td>
</tr>
<tr>
<td>System Landscape flexibility</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td></td>
</tr>
</tbody>
</table>
**Input**

**Functional Requirements**
- UI Visualization of aggregation of >1 service provider information
- Readonly Access
- No process flows required

**Constraints**
- Service provider are NOT service enabled, instead they do offer proprietary application programming interfaces and do not allow any modifications/add on deployments
- Information to be displayed comes from multiple service providers with approp. equal distribution
- Similar projects will follow

**Quality Requirements**
- Moderate end user performance requirements
- Operational data access without latency
- Development cost should be minimized

**Architecture Decisions**

**Service Consumption**
- No dominating system ➔ Symmetric Consumption Variant
- Readonly ➔ no transaction coordination required

**Service Integration**
- Proprietary backends without modification possibilities ➔ backend abstraction via ESB hub
- More coming similar projects ➔ service registry to keep track on service provisioning and start design time governance

**Service Provisioning**
- Legacy interfaces given ➔ no possibilities for backend service adaptation

**Architecture Blueprint**

[Diagram of SOA Process Pattern - Reference Architectures]

- **User** ➔ **Consumption Ui** ➔ **Consumption Logic**
- **UI Framework** ➔ **Service Consumption** ➔ **Adapter** ➔ **Message Bus** ➔ **Adapter** ➔ **ESR**
- **Service Integration - ESB** ➔ **Adapter** ➔ **Message Bus** ➔ **Adapter** ➔ **ESR**
- **Service Registry** ➔ **Process Integrator** ➔ **R/3**
- **SAP Business One**
- **SAP Business All in One**
SOA Process Pattern - Reference Architectures

UI Dashboard - II

Input

Functional Requirements
- UI Visualization of aggregation of >1 service provider information
- Readonly Access
- No process flows required

Constraints
- Service provider are service enabled, and do allow modifications/add on deployments
- Information to be displayed comes from multiple service providers with approp. equal distribution
- Similar projects will follow

Quality Requirements
- High end user performance requirements
- Operational data access without latency
- Development cost are flexible to foster end user performance

Architecture Decisions

Service Consumption
- No dominating system ➔ Symmetric Consumption Variant
- Readonly ➔ no transaction coordination required
- No latency ➔ replication is not an option

Service Integration
- Backend systems can be extended and performance is key ➔ Distributed ESB
- More coming similar projects ➔ service registry to keep track on service provisioning and start design time governance

Service Provisioning
- High end user performance requirements ➔ backend service adaptation to speed up tailored service provisioning (Backend extensibility is given)

Architecture Blueprint
**Input**

**Functional Requirements**
- UI Visualization of aggregation of >1 service provider information
- Readonly Access
- No process flows required

**Constraints**
- Service provider are service enabled
- Information to be displayed comes from multiple service providers with approp. equal distribution
- Similar projects will follow

**Quality Requirements**
- High end user performance requirements
- Operational data access without latency
- Development cost must be minimized

**Architecture Decisions**

**Service Consumption**
- No dominating system ➔ Symmetric Consumption Variant
- Readonly ➔ no transaction coordination required
- No latency ➔ replication is not an option

**Service Integration**
- Backend systems are service enabled ➔ no mediation hub (ESB) necessary
- More coming similar projects ➔ service registry to keep track on service provisioning and start design time governance

**Service Provisioning**
- N.a.

**Architecture Blueprint**

- User ➔ Consumption Server
- Composition Environment ➔ UI Framework
- Consumption UI ➔ Consumption Logic
- Mediation ➔ Service Registry
- ESR ➔ Publish ➔ R/3
- SAP Business One ➔ SAP Business All in One
- R ➔ Service Provider Backend
- Service Provider ➔ ESR

---

**SOA Process Pattern – Reference Architectures**

**UI Dashboard - III**
## SOA Process Pattern - Reference Architectures

### UI Dashboard - Quality Assessment

<table>
<thead>
<tr>
<th>Quality</th>
<th>Arch. Blueprint</th>
<th>I - ESB</th>
<th>II - Distributed ESB</th>
<th>III Peer to peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>End user performance</td>
<td></td>
<td>★★☆☆☆</td>
<td>★★☆☆☆</td>
<td>★★☆☆☆</td>
</tr>
<tr>
<td>UI Flexibility</td>
<td></td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
</tr>
<tr>
<td>TCO</td>
<td></td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
</tr>
<tr>
<td>TCD/required skills</td>
<td></td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
</tr>
<tr>
<td>System Landscape flexibility</td>
<td></td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
<td>★★★☆☆</td>
</tr>
</tbody>
</table>
SOA Process Pattern - Reference Architectures
To be cont’d

- Offline Client
- Offline Multiplexing
- A2A Integration
- Dominated B2B
- B2B Partner Integration
- Backend Multiplexing
- Backend Abstraction
Customers should plan their SOA Adoption along a multidimensional (e.g. Architecture, Governance, etc.) maturity model

- Determine your current maturity level
- Make small steps along all dimensions
- Use frontrunner projects to evolve single dimensions
- Balance Project Portfolio decisions between business value and SOA evolution steps
  - No big bang service enablement
  - No ultra pragmatic proprietary integration solutions
Customers should plan their portfolio and not single project architectures

- Combine architectural decisions along all projects within portfolio
- Try to find a global optimum instead of (project) local optimum
- Build and maintain an architecture vision along the SOA maturity path and reflect the project portfolio
Service Enabling is a strategic goal, not a short term decision

- Do service enabling, where required
- Establish governance in parallel
  - Service provisioning Guidelines & Policies
  - Establish controlling mechanisms
  - Define deviation management
Strategic Guidance

Golden rules

- Plan for, continually monitor, and drive the reuse of enterprise services. Reuse doesn't just happen; it requires commitment and control. Keep assessing your progress and regularly check your enterprise SOA scorecard.
- Start with a readiness assessment. Determine what you already have in place before you move forward.
- Begin with small and simple steps that deliver fast results. Get some quick wins by leveraging your current portfolio of applications and IT projects. Identify enterprise services that are ideal for reuse. Reflect upon your experiences.
- Keep your service models as plain, simple, and well documented as possible. Establish and ensure adherence to your company's unique enterprise SOA architecture and design methodology.
- Implement an enterprise services repository. Implementing a repository provides design-time and runtime governance automation, life-cycle management, and greater reuse.
- Define a number of core competencies and associated roles. These roles drive architectural development and organizational procedures, and they overcome functional silos.
- Build and actively participate in an enterprise services community. Your company can benefit from best practices that you learn from others and accelerate advancement along the enterprise SOA learning curve by using those best practices.
- Establish a change management process. Change management procedures brace your organization for anticipated changes as well as for the unexpected. Make sure that a defined process governs enterprise service reuse (including the first instance of reuse).
- Establish effective escalation procedures. Opinions can and will differ. Be sure to set up an enterprise service design steering committee that is staffed by people and stakeholders other than those of the enterprise SOA competency center.
- Don't reinvent the wheel. You are not alone. Trusted advisors and a community of enterprise SOA peers can provide support and guidance and share best practices with you.
Table of Contents

- Definitions & Value Proposition
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- **Enterprise SOA Governance**
- SAP Product Mapping & Implementation Guidance
- Business & Technology Trends

----------------- Appendix ----------------------
- Architecture Pattern Catalogue
Enterprise SOA Governance

Definition

“Governance refers to the processes that an enterprise puts in place to ensure that things are done right, where ‘right’ means in accordance with best practices, architectural principles, governmental regulations, laws, and other determining factors. SOA governance refers to the processes used to govern adoption and implementation of SOA.”

Anne Thomas Manes
Vice President and Research Director, Burton Group
Enterprise SOA Governance

Key Elements

- Governance management: the organizational structures, skillsets, and procedures that are aligned with the specific needs of the company
- Tool sets and life-cycle management: all the tools required to support good governance and to achieve the ultimate goal of automated governance
- A design and modeling methodology: a methodology that spans all phases of service design, harmonization, and implementation
- Community building: continually sharing ideas and best practices, inside and outside of the company, for faster time to value and sustained success.
Enterprise SOA Governance

Governance Domains

- Managing the portfolio of services
- Planning development of new services and updating current services
- Managing the service lifecycle
- Ensuring that updates of services do not disturb current service consumers
- Using policies to restrict behavior
- Ensure consistency of services using rules that all services need to conform to
- Establish performance monitoring of services
- Loosely-coupled nature of service composition, means the consequences of service downtime or underperformance can be severe.
- By monitoring service performance and availability, action can be taken instantly when a problem occurs.
Table of Contents

- Definitions & Value Proposition
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- **SAP Product Mapping & Implementation Guidance**
- Business & Technology Trends

----------------- Appendix -----------------------

- Architecture Pattern Catalogue
NetWeaver BPM
- Is designed to orchestrate automated processes across multiple systems. Both, human interaction and service orchestration are focused activity types.

SAP Business Workflow
- Has significant strength for tight integration into a single ABAP based system.

Guided Procedures
- Was primarily designed to guide users to subsequent interactive processing steps across multiple systems

ccBPM
- Is optimized to orchestrate services, especially for integration processes.
WebDynpro ABAP

- Is optimized to provide user interface access to ABAP functionality within one system. Tabular configuration supported with some graphical enrichments.

WebDynpro Java

- Provides graphical modeling capabilities with seamless transition to programming.

Visual Composer

- Designed for easy graphical modeling without breakout to programming.
Enterprise Modeling

- Aims to model all automated and manual processes in an enterprise. Documentation, analysis and optimization are purely paperwork jobs as prerequisite for change management. SAP’s offering includes Solution Manager, Enterprise Service Repository and ARIS (partner product) for this task.

Service Modeling

- Helps to provide contract capable services, which encapsulate reusable business functionality following the SOA definition. SAP supports service modeling with Enterprise Service Repository.

Process Modeling

- Transfers parts of the modeled enterprise processes to automated systems. Depending on the target system, an appropriate design time is available within SAP’s portfolio:
  - NetWeaver BPM
  - SAP Business Workflow
  - Guided Procedures
  - ccBPM
Table of Contents

- Definitions & Value Proposition
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- SAP Product Mapping & Implementation Guidance
- **Business & Technology Trends**

----------------- Appendix -----------------------

- Architecture Pattern Catalogue
SOA/ BPM Trends

Technology Trends
- Enterprise Service Bus
  - Peer to peer Communication
  - Enterprise Service Bus Central Hub
  - Distributed Enterprise Service Bus
- Continuous UI technology evolution (Flex, Silverlight, Ajax)
- On Demand
  - Service in the Cloud
  - Service Virtualization
  - Automatic self configuration of service networks
- BPMN 2.0
- Continuous Communication protocols evolution
  - CORBA/COM/RFC
  - SOAP
  - RESTful services

Business Trends
- LOB empowerment
- Continuous cost pressure
- Best of breed consolidation
- Acquisitions by large vendors
Table of Contents

- Definitions & Value Proposition
- Solution Space & Key Capabilities
- Reference Architectures & Maturity Model
- Enterprise SOA Governance
- SAP Product Mapping & Implementation Guidance
- Business & Technology Trends

----------------- Appendix ----------------------

- Architecture Pattern Catalogue
Appendix

Architecture Pattern Catalogue (Variants & Options)

Consumption
- Single System Consumption
- Asymmetric Consumption
- Symmetric Consumption

Integration
- Peer to Peer
- ESB Hub
- Distributed ESB
- Central Service Inventory
- Service Virtualization
- Control Flow

Provisioning
- Legacy Interface
- Service Enabled Interface
- Backend Service Adaptation
Architecture Pattern
Service Consumption

Service Consumption
Service Integration
Service Provisioning
3 – Symmetric Consumption

Service Consumer resides in a dedicated system, separated from all service providers.

2 – Asymmetric Consumption

Service Consumer resides in one of the Service Provider systems and connects to all other service providers.

1 – Single System

Service Consumer resides in the same system as the service provider. This topology results in very high performance, but also very tight coupling and restrictions to the available backend consumption technologies (e.g. UI).
Reference Pattern
Service Consumption – Architecture Variants – Single System

Precondition
- Only one backend system as service provider
- Backend system allows additional consumption development

Description
- Service Consumption is deployed to the backend server and uses the intrinsic consumption technologies.

Quality Assessment
- Pros
  - Very good performance for consumption of the backend services, because no additional network latency
  - Easy UI integration with other UIs of the backend system
  - Low learning curve with the assumption that skills for backend technology are already available
- Cons
  - Limited UI technology availability
  - Tight coupling to backend system
Reference Pattern
Service Consumption - Architecture Variants - Asymmetric Consumption

Precondition
- Backend system allows additional consumption development

Description
- One Service Provider is treated as the dominating one and its system is used to host the consumption. Local and remote service providers are connected to this consumption via Service Integration.

Quality Assessment
- Pros
  - Very good performance for service consumption of the dominating backend services
  - Easy UI integration with UIs of the dominating backend system
  - Low learning curve with the assumption that skills for backend technology of dominating system are already available
- Cons
  - Limited UI technology availability
  - Tight coupling to dominating backend system
  - Usage of service integration on a hub is challenging, because service integration with local services cannot use the hub functionality (without losing the performance advantage) and causes redundant implementations of mediation capabilities. Implementing all service integration into the dominating backend systems bares the disadvantage of tight coupling (System takes 3 responsibilities in the landscape and thus evolves to bottleneck or single point of failure)
Reference Pattern
Service Consumption - Architecture Variants - Symmetric Consumption

Precondition
- Frontend Server with Consumption Capabilities

Description
- A dedicated consumption frontend accesses multiple service provider backends via Service Integration.

Quality Assessment
- Pros
  - Consumption Centralization
  - User decoupling from backend
  - High UI Flexibility ➔ availability of state of the art UI technologies, integration into user productivity tools
- Cons
  - No local service consumption, all services are accessed remotely ➔ moderate end user performance
One of the service provider systems plays a dominant role and is used as encapsulating composition container. User interface may be provided in a separate system or also within the service provider system.

Service Consumer resides in a dedicated system and provides Transaction Coordination mechanisms to ensure transactional integrity.

Consumer system deals with multiple service providers without transaction management, if:
- Access is readonly or change actions happen only to one service provider
- More than one service provider with write access, but no required transaction management (Fire & Forget)
Reference Pattern
Service Consumption – Architecture Options – Symmetric Transaction Coordination

Precondition
- Symmetric Consumption Variant
- Service Provider systems provide either Compensation services or Distributed transaction support

Description
- Transaction coordination mechanisms help to improve transactional consistency across multiple systems. There are primarily two basic mechanisms available depending on the offering of the service providers
  - Multi-phase commit mechanisms (e.g. 2-phase commit or 3-phase commit) extend the local transaction management though a sophisticated handshake protocols to global transaction. Example can be found in WS-Transaction spec.
  - Compensation uses 'compensating' services to 'rollback' the effects of a previous service call in a service provider system, if one of the associated service calls of another provider fails.

Quality Assessment
- Multi-phase commit
  - Pros
    - Ensures transactional consistency
  - Cons
    - High implementation effort, because today's business systems do not support it out of the box.
    - Stateful communication, tighter coupling between systems

- Compensation
  - Pros
    - Simple implementation of compensating services
  - Cons
    - Supports transactional integrity, but does not ensure it (what happens, if a compensation action fails?)
Reference Pattern
Service Consumption - Architecture Options - Asymmetric Transaction Coordination

Precondition
- Asymmetric Consumption Variant
- Service Provider systems provide either Compensation services or Distributed transaction support

Description
- On the protocol level very similar to the Symmetric Transaction Coordination. If one backend system is used to encapsulate other service provider, it should act as a consumption environment, which exposes itself meaningful services to e.g. a user interface. The user interface can be provided in an external system or the backend system itself.

Quality Assessment
- See symmetric transaction coordination
Reference Pattern
Service Consumption – Architecture Option – Replication

Replication

Service Consumer interacts with a local storage, which is replicated to one or more service providers. Offline scenarios or very high end user performance requirements (for the price of latency) justify this pattern.
Reference Pattern
Service Consumption – Architecture Options – Replication

Precondition
- Consumption System with local storage

Description
- A dedicated consumption system replicates data from multiple service providers to a local store and provides Service Consumption capabilities on the local storage

Quality Assessment
- Pros
  - Very high end user performance
  - User decoupling from backend
  - High UI Flexibility ➔ availability of state of the art UI technologies, integration into user productivity tools
  - Offline usage possible
- Cons
  - Information Latency (service provider information is replicated to consumer)
  - Very high development effort
  - Increased TCO
Objects in the consumption logic interpret rules and interact with each other to execute business processes.

Reference Pattern
Service Consumption – Architecture Option – Control Flow

1 – Process Engine driven processes
A process engine interprets a process flow model and triggers actions within the underlying objects in the consumption logic.

2 – Object driven Processes
Objects in the consumption logic interpret rules and interact with each other to execute business processes.

Architecture Overview
Reference Pattern

Service Consumption – Architecture Options – Process Engine driven processes

Precondition
- Consumption System process engine and rules engine

Description
- A dedicated process engine is used to execute modelled processes. The process engine controls and drives activities of associated objects taking part in the consumption logic. A consumption logic object may also be a placeholder for externally provided services. Business decisions are based on a rules engine to provide flexible adaptations of business rules.

Quality Assessment
- Pros
  - Easily adaptable and extensible
  - High transparency of business processes as base for process analytics and optimizations
- Cons
  - High development effort to reflect flexible or unstructured processes. A very flexible process modeling and execution environment is required.
Reference Pattern
Service Consumption – Architecture Options – Object driven processes

Precondition
- Consumption System with rules engine

Description
- Process flows are controlled peer to peer by the consumption objects. Business rules are provided by a rules engine to ease flexible adaptations to change

Quality Assessment
- Pros
  - Very high flexibility

- Cons
  - Transparency of business control flows not automatically given
Reference Pattern
Service Consumption - Architecture Variability Matrix

Choose one of the architecture variants

Architecture Variants
- Single System Consumption
- Asymmetric Consumption
- Symmetric Consumption

.. and add some options
### Service Integration – Architecture Variants

<table>
<thead>
<tr>
<th>Description and Key Qualities</th>
<th>1 – Peer to Peer (P2P)</th>
<th>2 – Enterprise Service Bus</th>
<th>3 – Distributed Enterprise Service Bus</th>
</tr>
</thead>
</table>

**1 – Peer to Peer (P2P)**

Service Consumer and Service provider communicate via an agreed protocol. Typically mediation on both sides is necessary to adapt to this protocol.

**2 – Enterprise Service Bus**

An Enterprise Service Bus is used as central mediation hub to translate all system messages into an “Enterprise message language” and vice versa. Only one adapter per connected system is necessary.

**3 – Distributed Enterprise Service Bus**

The logical ESB is configured like a hub, but deployed de-centrally into the connected systems.

#### Architecture Overview

- **System 1**: Service Consumer, Mediation, Service Provider
- **System 2**: Service Consumer, Mediation, Service Provider
- **ESB**: Service Consumer, Service Integration, Service Provider
- **Distributed ESB**: System 1, Service Consumer, Distributed ESB, Service Integration, Service Provider, System 2
Reference Pattern
Service Integration – Architecture Variants – Peer to Peer

Precondition
- Consumer system and provider system allow mediation to adapt a potentially proprietary agreed protocol

Description
- Consumer and provider negotiate an integration protocol.

Quality Assessment
- Pros
  - Low cost integration
  - Possible with ‘any’ service provider technology that offers remote API capabilities
- Cons
  - Does not scale for more than 2 systems
  - Proprietary and thus potentially high TCO over time
  - Tight coupling ➔ changes must be synchronized between all communicating parties
Reference Pattern
Service Integration – Architecture Variants – Enterprise Service Bus

Precondition
- Consumer/Provider allow for 'any' remote API technology

Description
- Central Enterprise service bus defines message 'enterprise message syntax and semantics'. Each system is connected via a specific adapter to translate system specific messages into 'enterprise language'

Quality Assessment
- Pros
  - Non invasive integration with 'star' topology (n adapter instead of ~n²)
  - Possible with 'any' service provider technology that offers remote API capabilities can be connected
  - Centralized integration configuration/Administration
- Cons
  - Additional network latency
  - Additional hub in system landscape ➔ TCO increase
Reference Pattern
Service Integration – Architecture Variants – Distributed Enterprise Service Bus

Precondition
- Consumer/Provider allow additional deployment of distributed ESB adapter
- Distributed ESB product available

Description
- Distributed Enterprise service bus overcomes some network latency drawbacks of ESB hub, if preconditions match. No runtime hub, but configuration and administration is organized like in hub scenario.

Quality Assessment
- Pros
  - Integration with 'star' topology (n adapter instead of \( \sim n^2 \))
  - Centralized integration configuration/administration
  - No additional network latency at runtime
- Cons
  - Expensive/difficult in very heterogeneous system landscapes (distributed ESB must be deployed and run on any participating system)
Reference Pattern
Service Integration - Architecture Option - Central Service Registry

1 – Service Registry

- Service Provider publishes a service definition into a central inventory (Service Registry)
- Service Consumer finds a service definition in the Service Registry.
- Service Consumer binds to the service endpoint, which is associated with the service definition

Architecture Overview

Service Consumer

Service Provider

Service Registry

Reference:
- Bind (R)
- Find (R)
- Publish (R)
**Reference Pattern**

Service Integration – Architecture Option – Central Service Inventory

**Precondition**
- Standardized Service Registry available (e.g. UDDI)

**Description**
- Service provider publish their service definitions and endpoint information to a central service registry including QoS. Service consumer use this inventory (typically at design time) to find & locate the necessary services and service endpoints for the intended consumption and bind at runtime to these endpoints.

**Quality Assessment**
- **Pros**
  - High transparency of service provisioning and consumption in big landscapes
  - Excellent starting point for service governance
- **Cons**
  - Additional administration effort ➔ TCO increase
Service Provider publishes a service definition into a central inventory (Service Registry).

Additional virtual service façade is registered and abstracts some concrete endpoints based on QoS or other criteria.

Service Consumer binds to virtual service, which is translated from infrastructure to ‘real’ service.
Reference Pattern
Service Integration – Architecture Option – Service Virtualization

Precondition
- Standardized Service Registry available (e.g. UDDI)
- Virtualization infrastructure available

Description
- Service provider publish their service definitions and endpoint information to a central service registry (e.g. in Web Service Description Language). A virtual service definition is added, which is more decoupled from the underlying provider. Service consumer bind against this virtual service and the infrastructure resolves at runtime the currently most appropriate implementation of the virtual service regarding availability, Quality of Service or other criteria. Any necessary transformation between the virtual service and the 'real' services are done by the infrastructure.

Quality Assessment
- Pros
  - High transparency of service provisioning and consumption in big landscapes
  - Excellent starting point for service governance
  - Strong decoupling from real services leads to very good agility and flexibility on the provisioning side
- Cons
  - Rare and immature vendor offering
A process engine interprets a process flow model and triggers actions within the underlying objects in the integration logic.
Reference Pattern

Service Integration – Architecture Options – Process Engine driven processes

Precondition

- Integration System process engine and rules engine

Description

- A dedicated process engine is used to execute modelled processes. The process engine controls and drives activities of associated objects taking part in the integration logic. An integration logic object may also be a placeholder for externally provided services. Business decisions are based on a rules engine to provide flexible adaptations of business rules.

Quality Assessment

Pros

- Easily adaptable and extensible
- High transparency of integration processes as base for process analytics and optimizations

Cons

- High development effort to reflect flexible or unstructured processes. A very flexible process modeling and execution environment is required.
Choose one of the architecture variants

- P2P Integration
- ESB Hub Integration
- Distributed ESB Integration

.. and add some options
Architecture Pattern
Service Provisioning
Reference Pattern
Service Provisioning – Architecture Variants

1 – Legacy Interface
Service Provider comes with a legacy interface. Mediation to the service consumer happens outside the provisioning system.

2 – Service enabled
Service Provider comes with a standardized service interface or offers possibilities to add service enabling to the provisioning system.

Architecture Overview

- Service Consumer
- Service Integration
- Business Functionality
- Service Provider

- Service Consumer
- Service Integration
- Service Facade
- Business Functionality
- Service Provider
Reference Pattern
Service Provisioning – Architecture Variants – Legacy Interface

Precondition
- Service Provisioning system offers a non WS standard remote API and cannot be service enabled

Description
- A service provisioning system offers a proprietary remote API, which cannot be treated as service enabling. The typical solution is service enabling with a service integration hub, e.g. Enterprise Service Bus and appropriate adapter to transform the proprietary API into a standardized format.

Quality Assessment
- Pros
  - Complete non invasive service enabling solution
- Cons
  - Performance Drawbacks due to additional network latency, if service enabling is done via a service integration hub
  - Non SOA standard integration, if the integration is done p2p with a consumption system and the transformations happen in the consumption system.
Reference Pattern
Service Provisioning - Architecture Option - Backend Service Adaptation

Backend Service Adaptation

- Service Provider opens up to host consumer specific mediation to speed up transformation. Service adaptation can be anticipated (flexible service interface, e.g. sorting, filtering via parameter) or unanticipated (additional deployment to service provider system necessary)

Description and key qualities

Architecture Overview
Reference Pattern
Service Provisioning – Architecture Option – Backend Service Adaptation

Precondition
- Service Provisioning system offers
  - Anticipated service adaptation like generic sorting, filtering parameters
  - Possibilities to develop and deploy unanticipated (via encapsulation or user exit) service adaptation

Description
- If the provisioning system offers anticipated service adaptation, parameterization of service calls is utilized to improve performance and limit data transfer. If additional deployment is possible, we can provide consumer specific encapsulation of service calls (filtering, sorting, etc.) or implement predefined user exits to reduce network traffic and thus improve performance.

Quality Assessment
- Pros
  - Reduction of network traffic and improvement of end user performance
- Cons
  - Anticipated: none
  - Unanticipated: Consumer specific deployment on service provisioning system
Reference Pattern
Service Provisioning - Architecture Variability Matrix

Choose one of the architecture variants

- Legacy Interface
  - Service Consumer
  - Service Integration
  - Business Functionality
  - Service Provider

- Service enabled Interface
  - Service Consumer
  - Service Integration
  - Business Functionality
  - Service Provider

.. and add some options

Backend Service Adaptation

- Service Consumer
- Service Integration
- Standard or Legacy
- Backend Adaptation
- Business Functionality
- Service Provider
**ESB**
- Enterprise Service Bus

**OSIMM**
- OpenGroup Service Integration Maturity Model

**DTR**
- Design Time Repository

**ESR**
- Enterprise Service Repository

**Anticipated Service Adaptation**
- Service implementations foresee variabilities in their delivery by additional parameters. Consumer specific responses can be generated by provisioning of appropriate parameters. Typical representatives are sort or filter (rows/columns) parameter.

**Unanticipated Service Adaptation**
- Service definition and implementation do not offer consumer specific service adaptation. Nevertheless the service has to be transformed/adapted to the consumers need. This can be done by an encapsulating service or, if the service implementation offers some generic user exit mechanism, by implementing some consumer specific hooks.