Abridged from the book

…and modified for presentation

by Al Smith, Jr. www.TheJavaArchitect.com

The Goal of SOA

Abridged and Modified from the book 'SOA Right Away' by Al Smith, Jr.
Published by: TheJavaArchitect.com

The Goal of SOA

- **Develop Applications that Provide:**
  - Standardized Service Contracts
  - Loosely Coupled Components
  - Abstracted Components
  - Reusable Components
  - Autonomic Components
  - Stateless Components
  - Discoverable Components
  - Composable Components

History of Distributed Computing

- **Distributed computing software components**
  - CORBA
  - DCOM
  - Web Services
- **Interface description languages**
  - XML-RPC
  - SOAP

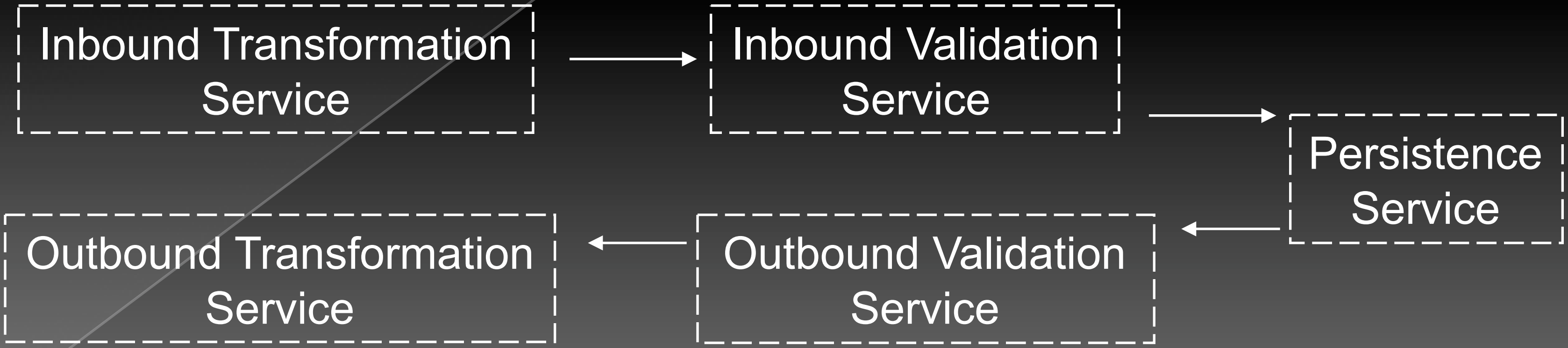
What Makes SOA Different?

- **Service-Orientation**
  - A design paradigm that provides a means to achieve "separation of concerns" through the use of services
    - Defined differently between SOA vendors
  - Derived from
    - Object Orientation
    - Web Services

"Ironically this is one of the problems SOA contracts attempt to prevent"

What is Service Oriented Architecture

- **Service**
  - An abstract definition of a function to be performed within a business process
    - Defines a *contractual* responsibility that a piece of software will perform within the constraints of a given process


Inbound Transformation Service
Inbound Validation Service
Persistence Service
Outbound Validation Service
Outbound Transformation Service

The State of SOA Adoption

- **SOA in a Box**
  - To address many of SOA's new infrastructure requirements, vendors have brought several new product categories to market:
    - SOA repositories
    - Enterprise service buses (ESBs)
    - SOA appliances
    - SOA and Web services management solutions

The State of SOA Adoption

- **SOA in a Box** **(continued)**
  - Most architects understand how to deploy a tactical, project-by-project approach
    - Buying whatever seems to fit the needs
  - Potential Issues:
    - Duplicate investments
    - Incompatible infrastructure
    - Brittle solutions

The State of SOA Adoption

- **Big Bang Adoption**
  - Few organizations can afford to fund a large, upfront, strategic investment approach:
    - Building an SOA platform is best done via incremental steps toward a strategic vision
    - The SOA platform evolves in steps rather than attempting one big jump

The State of SOA Adoption

- **Incremental Adoption**
  - To successfully execute an incremental SOA platform evolution, you need a coherent approach for:
    - Envisioning
    - Designing
    - Evolving your platform

The State of SOA Adoption

- **The Recommended Approach Consists of:**
  - SOA platform definition
  - A functional planning model
  - An evolutionary manner of building from your existing technologies
  - Integrating SOA specialty products as needed

Build or Buy

- **The Two Extremes are to Provide:**
  - A suite approach (full stack), tightly-integrated, all -embracing technology stack
  - To deliver a set of 'best-of-breed' point solutions that the organization assembles to meet their specific requirements

Build or Buy

- **Build or Buy: Suite Approach**
  - Represents:
    - Simple negotiation with a single vendor
    - Less work in setting up the environment
    - Knowledge that all components will interoperate
    - Simpler ongoing management

Build or Buy

- **Build or Buy: Suite Approach**
  - Considerations:
    - It creates a level of lock-in to a 'strategic' vendor that might be unacceptable
    - It limits the ability to extend the functionality (or performance) in a way that is not within the product development plans for that suite

Build or Buy

- **Build or Buy: Best-of-Breed**
  - Represents:
    - Minimal start-up cost
    - Open Source technology
    - Services are acquired as needed
    - Not obligated to fit a square peg into a round hole

Build or Buy

- **Build or Buy: Best-of-Breed**
  - Considerations:
    - Specifications aren't always industry compliant
    - Limited service domain knowledge and technical expertise
    - Requires an in-house development team with SOA expertise

Defining an SOA Strategy

Defining an SOA Strategy

- **Strategic Implementation Paths**
  - Simple Legacy Access
  - Build composite services
  - Achieve mature enterprise SOA delivery

Defining an SOA Strategy

- **Governing SOA Services**
  - Operation Control
  - Utilizing Governance to Accelerate Agility
  - Developing a Smarter SOA with Governance
  - Enabling Policy Management

Governing SOA Services

- **Operation Control**
  - SOA needs to be a cooperative venture with buy-in and participation from all the people who will be working on or with the business applications
    - Governance teams overseeing their initiatives
      - Critical in identifying required common services

Governing SOA Services

- **Operation Control** **(continued)**
  - SOA needs to start from the beginning with the services and applications that the business ***needs***
    - Little value in developing cool services undesired on business side

Governing SOA Services

- **Utilizing Governance to Accelerate Agility**
  - SOA is about business as much as it is about traditional application development
    - Mistake to leave implementation to coders
    - Get business people involved in SOA projects from the start

Governing SOA Services

- **Developing a Smarter SOA with Governance**
  - To get the scope and focus of SOA right it is important to remember that:
    - The A in SOA stands for "architecture"
    - Architects and Project Managers must play a key role

Governing SOA Services

- **Developing a Smarter SOA with Governance** **(continued)**
  - Requires investing in management and design to make sure the SOA project fits the business needs
    - *Vendors urge investing in software infrastructure*

Governing SOA Services

- **Enabling Policy Management**
  - Policy management ensures that policies approved by the governance framework, covering areas such as compliance, conformance, security, etc. are enforced throughout the lifecycle of the SOA initiative

Financial Justification

- **Benefits of Implementing SOA:**
    - Increased Flexibility
    - Increased Extensibility
    - Increased Robustness
    - Increased Reusability and Productivity
    - Increased Business Requirements Fulfillment

Managing SOA Risk

- **Risk Identification**
  - Implementations can get bogged down:
    - Project leaders try to do too much too quickly
    - Get overwhelmed with enormity of tasks
    - Taking on too many SOA projects with too many services

Managing SOA Risk

- **Risk Identification** **(continued)**
  - Great flux in Web services specifications and rapid growth in product functionality
  - Dynamic vendor relationships
  - SOA best practices growing in maturity

Managing SOA Risk

- **Risk Identification** **(continued)**
  - **Building ahead of your own maturity**
    - Increases risk of issues when building too many services into your platform
    - Requires experience to understand how to use them
    - May not properly prioritize evaluation criteria or may include criteria of non -value

SOA Migration Strategy

- **SOA Migration Strategy**
  - Extract Information and Put it to Use
  - Promote Reuse and Eliminate Redundancies
  - Increase Visibility of Services Across Heterogeneous Platforms

SOA Migration Strategy

- **Extraction of Information and Put it to Use**
  - Requires a carefully thought out data model
    - Traditional applications commonly require data from ***external*** business partners
    - By contrast, SOA creates cohesive silos which requires a strategy for integrating new and existing technologies

SOA Migration Strategy

- **Promote Reuse and Eliminate Redundancies**
  - Important to move beyond the old concept that new applications require new code
    - Old view defeats purpose of service reuse
      - *Discourage people from re-inventing the wheel*

SOA Migration Strategy

- **Increase Visibility of Services Across Heterogeneous Platforms**
  - Requires ability to configure business orchestration and propagate it to the designated components across application deployment
    - Must apply the right operational rules before any communication occurs
  - Provide real-time information about components such as performance and, security

SOA Centric Analysis and Design

Performing SOA Centric Analysis and Design

- **What Is the Best Solution for Building an SOA Platform?**
  - There isn't a best single sequence or solution for building an SOA platform
    - An SOA platform can start with messaging technologies such as HTTP, SOAP, REST, and message queuing
    - Factors such as diversity among organizations, and existing software infrastructures leads different *flavors* of SOA

Performing SOA Centric Analysis and Design

- **Identify your Existing Infrastructure's SOA Capabilities**
  - Forms a basis for ensuring that existing capabilities are fully leveraged
    - Prevents wasteful spending on new products that duplicate existing functions

Performing SOA Centric Analysis and Design

- **Identify your Existing Infrastructure's SOA Capabilities** **(continued)**
  - Examine existing application infrastructure and management products against a functional model of an SOA platform
  - Identify functions provided by existing products

Performing SOA Centric Analysis and Design

- **Identify your SOA Priorities**
  - Provides a list of potential options for ensuring near-term success with SOA -based solution delivery projects
  - Work from application road maps and service portfolio plans
  - Identify types of service implementations required for high-priority services necessary to build over near term

Performing SOA Centric Analysis and Design

- **Identify SOA Priorities** **(continued)**
  - Determine whether existing infrastructure can fulfill requirements
    - If gaps, investigate SOA specialty product categories, such as ESBs or integration -centric business process management suites (IC-BPMSes)
    - Analysis may highlight ways specialty products may fulfill some requirements more effectively than existing infrastructure

Performing SOA Centric Analysis and Design

- **Match Platform Plans to Organization's Investment Strategy and Risk Profile**
  - Determine where actual investments in an SOA platform will fall along a continuum from:
    - Small
    - Single-project purchases to large,
    - Cross-project infrastructure build-outs

Service-Orientation and Object-Orientation

Service-Orientation and Object-Orientation

- **A Comparison of Goals and Concepts**
  - Object-oriented analysis and design was responsible for popularizing the vision of applications designed to be:
    - Reusable
    - Flexible

Service-Orientation and Object-Orientation

- **A Comparison of Goals and Concepts** **(continued)**
  - OOAD grew out of a need for:
    - Service federation
      - Attempts to bring order to unstructured development processes
    - Instinct interoperability
      - Emphasizes the creation of code that closely mirrors real world

Service-Orientation and Object-Orientation

- **A Comparison of Goals and Concepts** **(continued)**
  - Provides rules and guidelines that govern careful separation of application logic and data into objects
    - Can be individually maintained
    - Helps minimize the impact of change on the application as a whole

Service-Orientation and Object-Orientation

- **A Comparison of Goals and Concepts** **(continued)**
  - Service-orientation shares similar goals as OOAD
    - Seeks to establish flexible design
    - Allow for agile accommodation of business requirements

Service-Orientation and Object-Orientation

- **Service-Orientation**
  - Similarly, service-orientation design is concerned with minimizing impact of change
    - Service Loose Coupling
    - Service Composability

Implementation

Implementation

- **Web Services**
  - Is an implementation of SOA that enables *service* to be distributed across a network
    - Uses Interface definitions to achieve, **Abstracted Components, Loosely Couple Components**
    - Uses XML to achieve **Autonomic Components, Stateless Components**
    - Uses WSDL to achieve **Standardized Service Contracts**

Implementation

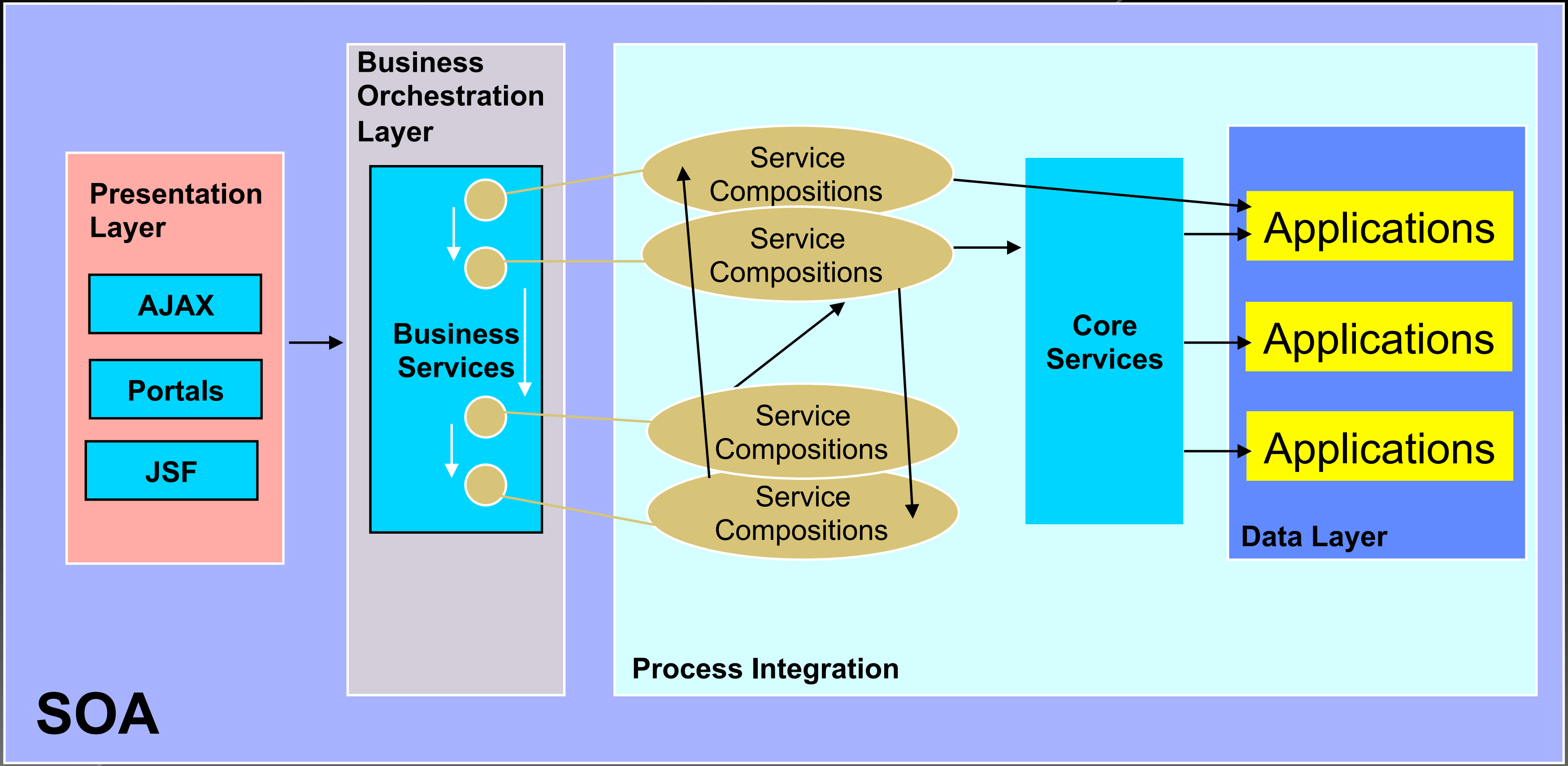
- **Web Services** **(continued)**
  - Is an implementation of SOA that enables *service* to be distributed across a network
    - Uses UDDI to achieve **Discoverable Components**
    - Use Repositories to achieve **Reusable Components**

Implementation

- **Services Layers**
  - Consequence of deploying SOA:
    - Heterogeneity across legacy systems is likely to increase
  - Benefit of deploying SOA:
    - Easier to manage heterogeneity and focus leveraging existing infrastructure instead of replacing it
  - ***Wrapper Services*** enable legacy applications to be reused and integrated into new SOA platforms, homogenously

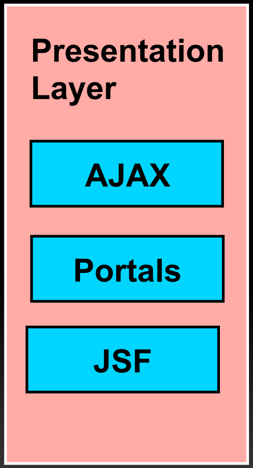
Implementation

Service Layers (continued)



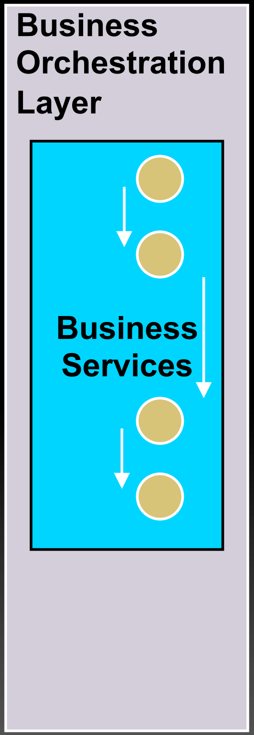
Implementation

- **Presentation Layer Wrapping**
  - Technologies such as AJAX, Portals and JSF make it possible for the UI to interact directly with backend services



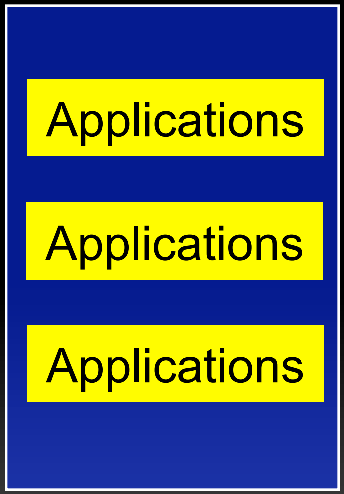
Implementation

- **Business Layer Wrapping**
  - Wrap business logic as services to communicate with other internal and external business services



Implementation

- **Data Layer Wrapping**
  - Wrap databases and legacy applications behind services so they can be exposed to other services

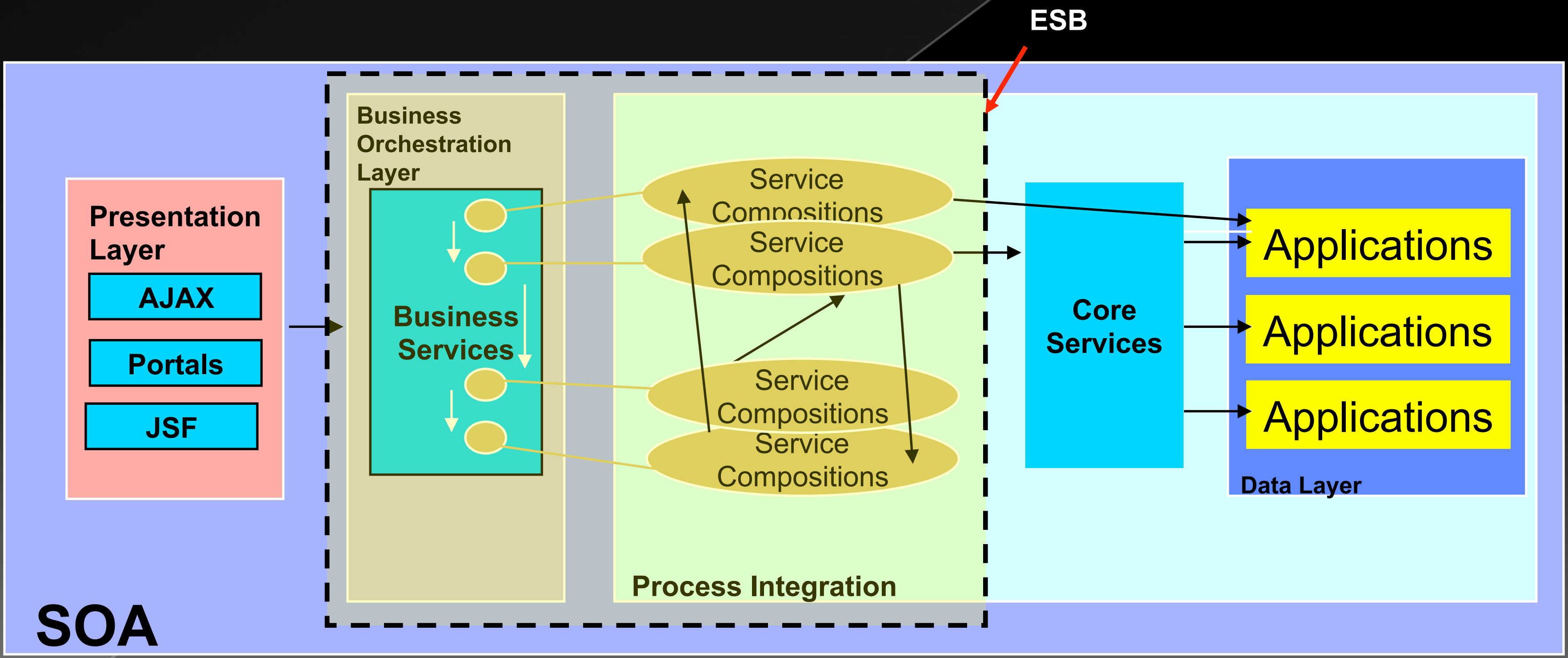


Implementation

- **Enhancing Connectivity and Flexibility with an Enterprise Service Bus (ESB)**
  - Enterprise Service Bus (ESB): essential run-time infrastructure for SOA
  - Primary functions of an ESB:
    - Ensure messages sent to and from services
    - Ensure messages arrive reliably at the right endpoint and in the right format
    - The business logic to be performed and messaged to be delivered in the correct sequence.

Implementation

- **Enhancing Connectivity and Flexibility with an ESB** (continued)



Implementation

- **Enhancing Connectivity and Flexibility with an ESB**
  - Provided in a highly-distributed and dynamically-changing environment
  - Support the performance, scalability and fault tolerance requirements
  - Vendors bundle different combinations of capabilities to orchestrate
    - Sequencing of business rules
    - Execution of business rules
    - Security

Conclusion

- Invest in management and design to make sure the SOA project fits the business needs.
- Important to move beyond the old concept that new applications require new code.
- SOA needs to start from the beginning with the services and applications that business needs.

This presentation and the complete book,
"SOA Right Away!"

Are available at SoaRightAway.com

SOAi

Service Oriented AI Architecture

Bridging the Gaps between Business & AI

Adapted from 'SOA Right Away!' by Al Smith Jr. · TheJavaArchitect.com

The Goal of SOAi

Designing AI Systems that are modular, persistent, and composable

The Goal of SOAi

Develop AI Applications that Provide:

Standardized Agent Contracts

Well-defined input/output schemas for every AI service (tools, agents, APIs)

Reusable & Composable Agents

Prompt templates, tools, and reasoning chains reused across products

Loosely Coupled Agents

LLMs, tools, and memory layers that operate independently and swap freely

Stateful / Persistent Memory

Agents that remember context across sessions (the key SOA limitation AI must solve)

Abstracted AI Components

Model-agnostic interfaces; consumers don't care if it's GPT-4o or Claude 3.5

Discoverable AI Services

Agent registries, MCP servers, tool catalogs — services AI can find and call

History of AI
Distributed Systems

History of AI Distributed Systems

2017	2020	2022	2023	2024+
Transformer	**GPT-3 / API Era**	**ChatGPT / RLHF**	**Agents & RAG**	**MCP & Agentic SOA**
'Attention is All You Need' — the architectural event that made modern LLMs possible	LLMs exposed as REST services; AI becomes a callable component for the first time	Instruction-tuning makes models conversational; mass adoption begins	LLMs gain tools, memory, and retrieval — stateful AI services emerge	Standardized protocols (MCP, OpenAPI tools) enable true service-oriented AI composition

Just as SOA evolved from CORBA → DCOM → Web Services, AI services evolved from APIs → Agents → Composable Agentic Architecture

What Makes SOAi Different?

AI-Orientation

- A design paradigm where intelligence is a service, not a feature
  - Each AI capability is independently deployable, versioned, and callable
  - Defined differently across AI vendors and platforms (OpenAI, Anthropic, Google, Mistral)
- Derived from
  - Service-Orientation (SOA) — modular, loosely-coupled design patterns
  - Transformer architecture — attention-based intelligence as a callable API

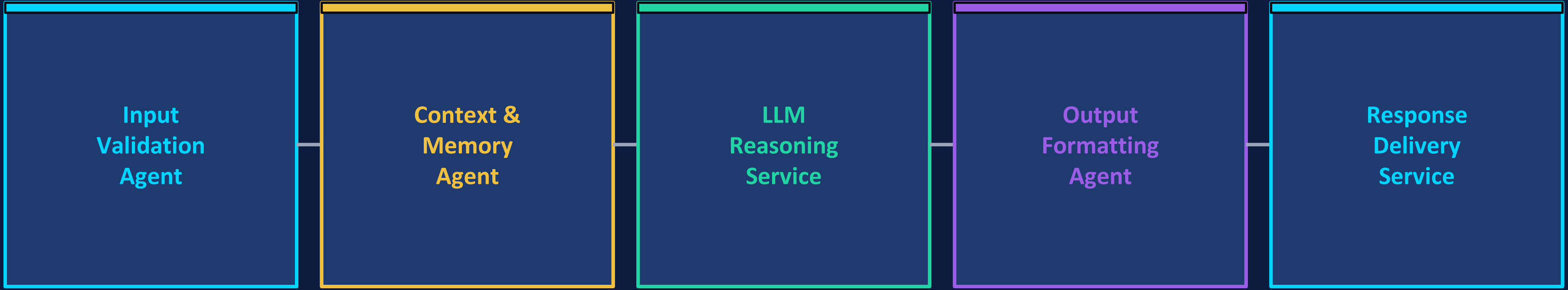
"Ironically, AI vendors define AI-orientation differently — exactly the problem AI service contracts attempt to prevent"

SOA Concept	SOAi Equivalent
WSDL Service Contract	OpenAPI / MCP Tool Schema
ESB (Enterprise Service Bus)	AI Orchestrator / LangGraph
SOAP/REST Interface	LLM Tool Call / Function Calling
Service Registry / UDDI	Agent Registry / MCP Server List

What is Service Oriented AI Architecture?

AI Service

- An abstract definition of an intelligence function within a business workflow
  - Defines a contractual responsibility that an AI model or agent will perform
  - Scoped to a specific domain: summarization, extraction, reasoning, generation



AI Service Pipeline — each stage is independently deployable, testable, and replaceable

The State of AI Adoption

The State of AI Adoption

AI in a Box

To address AI's new infrastructure requirements, vendors have brought new product categories to market:

AI Orchestration Platforms

LangChain, LangGraph, CrewAI, AutoGen

Vector / Memory Stores

Pinecone, Weaviate, Chroma, pgvector

LLM Gateways & Routers

LiteLLM, PortKey, OpenRouter, BedRock

AI Observability & Evals

LangSmith, Arize, Weights & Biases, Braintrust

Agent Runtime Platforms

Vertex AI Agents, AWS Bedrock Agents, Azure AI Foundry

Model Context Protocol (MCP)

Anthropic's open standard for tool/agent interoperability

The State of AI Adoption

AI in a Box (continued)

- Most teams understand how to deploy a tactical, project-by-project approach
  - Buying whichever model or platform seems to fit the immediate need
- Potential Issues:
  - Duplicate AI subscriptions and overlapping vendor capabilities
  - Incompatible context formats, tool schemas, and memory APIs
  - Brittle prompt pipelines tightly coupled to a single model provider

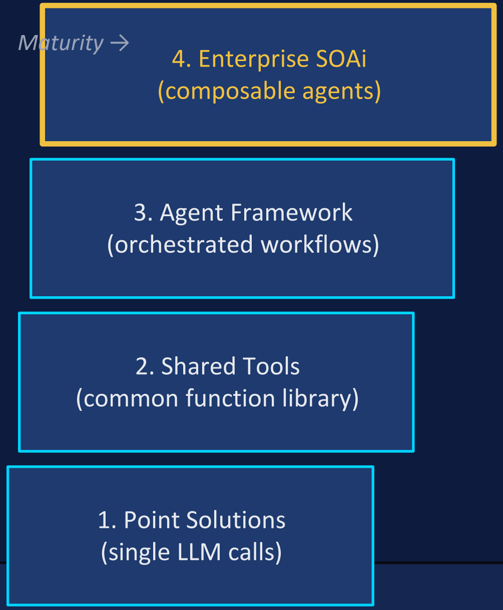
⚠ Common Pattern

- One team uses GPT-4o via OpenAI SDK
- Another team uses Claude via Bedrock
- Both reinvent memory, tool calling, and orchestration from scratch
- Result: 3× cost, 0× reuse

The State of AI Adoption

Incremental Adoption

- Few organizations can afford a full AI platform overhaul upfront
  - Building an SOAi platform is best done via incremental steps toward a strategic vision
  - The AI platform evolves in stages, not one big rewrite
- To successfully execute an incremental AI platform evolution, you need a coherent approach for:
  - Envisioning — mapping AI capabilities to business processes
  - Designing — defining agent contracts, memory models, and orchestration
  - Evolving — replacing point solutions with shared AI infrastructure over time



Build or Buy Your AI Stack

Build or Buy

Suite Approach — Full AI Platform

(Azure AI Foundry, AWS Bedrock, Google Vertex AI, Salesforce Einstein)

✓ Represents:

- Single vendor negotiation and billing
- Pre-integrated model + memory + tooling
- Guaranteed interoperability within the platform
- Simpler governance and compliance management

⚠ Considerations:

- Vendor lock-in — switching models requires platform migration
- Limits ability to use best-of-breed models as they emerge
- Innovation pace tied to one vendor's roadmap
- Often opinionated about data residency and privacy

Build or Buy

Best-of-Breed AI Stack

(Open-source orchestrators + swappable LLM providers + independent memory/tool layers)

✓ Represents:

- Minimal lock-in — swap models as better ones emerge
- Open-source tools (LangGraph, Ollama, ChromaDB)
- Acquire AI capabilities exactly when and where needed
- Mix providers: Claude for reasoning, Gemini for multimodal, GPT-4o for code

⚠ Considerations:

- Specifications and tool schemas vary across providers
- Requires in-house AI engineering expertise to integrate
- Higher integration complexity — you own the glue code
- Eval and observability must be assembled independently

Defining an SOAi Strategy

Defining an SOAi Strategy

Strategic Implementation Paths

1. **Simple AI Access**

   Expose existing LLMs as internal REST endpoints. Standardize the model interface so application teams call one contract, not five SDKs.

2. **Build Composite AI Services**

   Combine memory, retrieval, tool-use, and reasoning into reusable agent components. Business logic assembles agents like LEGO bricks.

3. **Achieve Mature Enterprise SOAi**

   Full agentic platform: governed agent registry, standardized tool schemas (MCP), shared memory stores, and model-agnostic orchestration.

Governing AI Services

Operation Control

- AI deployment needs buy-in from business, legal, security, and engineering
  - AI governance teams must oversee model selection, data handling, and output quality
  - Critical for identifying shared AI capabilities that multiple teams need
- SOAi needs to start from the beginning with AI services the business actually needs
  - Little value in building impressive AI demos that solve no real business problem
  - Get business stakeholders involved in AI workflow design from day one

⚙ **Operation Control**

🛡 **Model Governance**

📋 **Prompt Policy Mgmt**

✓ **Output Quality Eval**

Governing AI Services

Enabling AI Policy Management

AI policy management ensures that policies approved by the governance framework — covering compliance, safety, fairness, data residency, PII handling, and model versioning — are enforced throughout the lifecycle of every AI service.

Prompt Safety Policies

Guardrails and content filtering applied before and after every LLM call

Data Residency Rules

Which data can be sent to which external model provider, in which region

Model Version Pinning

Production AI services must declare the exact model version they depend on

Evaluation Gates

No AI service ships without passing automated evals on golden test sets

Cost & Rate Controls

Token budgets, rate limits, and fallback routing to prevent runaway spend

Audit Logging

Every prompt/completion pair logged for compliance, debugging, and drift detection

Managing AI Risk

Risk Identification

[HIGH] Model Hallucinations

AI confidently generating incorrect outputs without flagging uncertainty

[HIGH] Prompt Injection

Adversarial inputs hijacking agent behavior through malicious prompt crafting

[HIGH] Data Leakage

Sensitive data inadvertently included in prompts sent to external model APIs

[MED] Model Drift

Model providers silently update behavior, breaking previously validated pipelines

[MED] Cost Runaway

Agentic loops with no token budgets exhausting API quotas in minutes

[MED] Building Ahead of Maturity

Deploying complex multi-agent systems before your team can operate or debug them

AI Migration Strategy

Migration Strategy

Extract Intelligence & Put it to Use

Audit existing ML models, heuristics, and manual processes • Identify which can be replaced or enhanced by LLMs • Requires a careful data model — SOA's lesson about external partner data applies equally to training and RAG data

Promote Reuse & Eliminate Redundancy

Move beyond the old concept that each AI feature requires a new custom model • Build shared prompt libraries, tool registries, and evaluation harnesses • Discourage teams from reinventing memory stores and RAG pipelines

Increase Visibility Across AI Services

Centralized observability: trace every agent action, tool call, and LLM response • Apply operational rules (rate limits, guardrails) before any AI service call occurs • Real-time dashboards for latency, cost, error rates, and hallucination detection

SOAi-Centric
Analysis and Design

SOAi-Centric Analysis and Design

What Is the Best Solution for Building an AI Platform?

- There isn't a single best sequence or solution for building an AI platform
- **An AI platform can start with any of these entry points:**
  - A single-model RAG pipeline serving one product team
  - A shared tool library (MCP server) used by multiple agents
  - A model gateway abstracting OpenAI / Anthropic / local models
  - Factors like existing data infrastructure, team expertise, and compliance needs lead to different 'flavors' of SOAi

SOAi Design Principles

- Model-Agnostic Contracts
- Stateful Memory Layer
- Shared Tool Registry
- Unified Observability
- Incremental Capability Growth

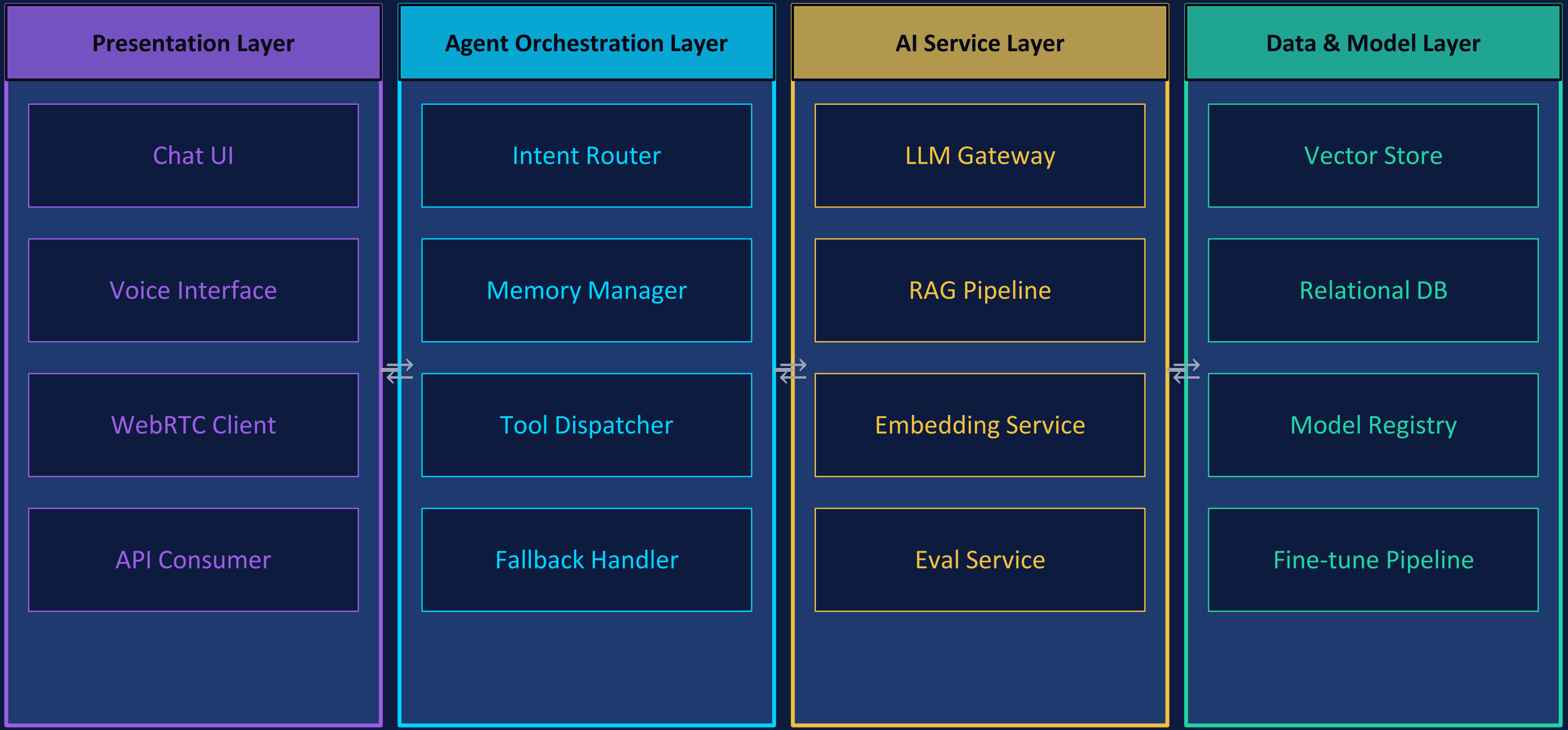
AI-Orientation and Service-Orientation

A Comparison of Goals and Concepts

Concept	Service-Orientation (SOA)	AI-Orientation (SOAi)
Core Unit	Service / Component	Agent / Model / Tool
Interface Contract	WSDL / OpenAPI Schema	Tool Schema / MCP Spec / System Prompt
State Management	Stateless (by design)	Stateful memory (required)
Reuse Mechanism	Service Registry / UDDI	Agent Registry / MCP Server
Composition Pattern	Orchestration / Choreography	Agentic Workflow / Multi-Agent System
Coupling Principle	Loose coupling via interfaces	Model-agnostic via unified contracts
Failure Recovery	Circuit breaker / retry	Fallback model / prompt retry / eval

Implementation

Implementation — AI Service Layers



WebRTC / Voice / Chat → Agent Orchestration → AI Services → Data & Models

Implementation

Model Context Protocol (MCP) — The AI Enterprise Service Bus

MCP is to SOAi what the ESB was to SOA — the essential runtime infrastructure for agent interoperability.

Standardized Tool Contracts
Every tool (function, API, data source) exposes a machine-readable schema agents can discover and call

Reliable Message Delivery
Ensures tool calls reach the right service in the right format — analogous to ESB message routing

Business Logic Sequencing
Agents compose tool calls into workflows without hardcoding API integrations

Security & Access Control
MCP servers enforce auth, rate limits, and scope — agents get only what they need

Distributed & Dynamic
New tools can be registered without redeploying agents — runtime discoverability

Vendor-Neutral Standard
Works across Claude, GPT-4, open-source models — breaks the vendor lock-in cycle

Conclusion

- Invest in AI architecture and governance from day one — not as an afterthought after the first production incident.
- Move beyond the old concept that every AI product requires a new model, new stack, and new integrations.
- SOAi starts with the intelligence services the business actually needs — not the most impressive demos.
- The principles of SOA are alive in AI: loose coupling, reusability, and standardized contracts are just as critical for agents as for web services.

SOAi

Service Oriented AI Architecture

This presentation is an adaptation of 'SOA Right Away!' by Al Smith Jr.