Service-Oriented Architectures

Basic Elements of SOA

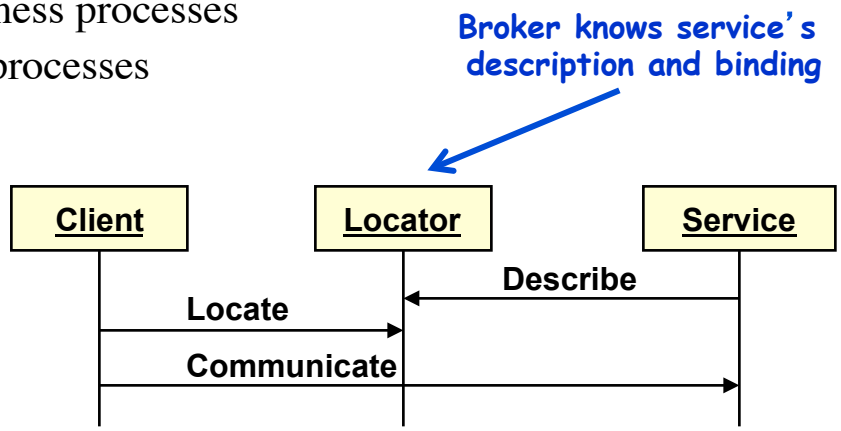
SOA versus REST

Revisit SOA Principles

Trends and Summary

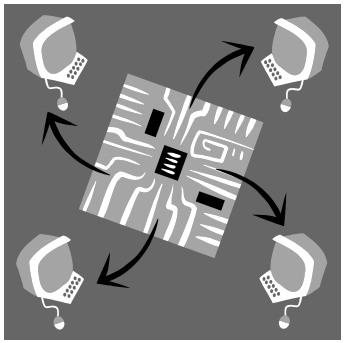
Basic Elements of SOAs

1. Communication infrastructure (transport)
   - The wire protocol for communication – e.g., SMTP/HTTP
2. Description of service specification
   - Specifies service, messages, responses, QoS guarantees, permissions, access control, etc.
3. Discovery and location
   - Allows clients to discover existing services
   - Criteria may include services properties (e.g., attributes, QoS)
4. Composition
   - Composes services to create higher-level business processes
   - Orchestration defines business and workflow processes
5. Additional infrastructure specifications
   - Security, transactions, event notification, etc.



1) Communications Infrastructure

- Infrastructure built on the layered network model
- Transports message and data between client & service
  - Passes message and data and returns data result or fault
- Defines how clients denote service endpoints
  - Technology examples – remote object reference, URL, etc.
- Defines a normalized message format
  - Communication protocol (we have been doing HTTP)
  - Technology examples – JSON, SOAP, etc.
  - Message transported with any protocol
    - May use multiple transports
- Addressing, routing, security
  - May be provided by messaging protocol (WS-* standards)
  - May be provided by transport (e.g., URL, https)



1) Communications Infrastructure

REST

- HTTP
- URIs (URLs) to locate
- Message formats are usually JSON with varying levels of semantic info
- Addressing, routing, security not directly in scope, rely on HTTP mechanisms and best practices like OAuth

SOA

- Variety of protocols
- Yes URLs to WSDL, but also registry and naming services
- SOAP (XML) as the message format, ESBs may normalize
- Addressing, routing, and security are done via a plethora of standards

2) Service Description

- Defines meta information describing a service's characteristics
  - Information necessary to deploy and interact with a service
- Characteristics
  - Functional characteristics defines messages the service sends & receives
  - Policy characteristics defines the service's execution context
    - Does service require security, transactions, etc.?
    - May also define QoS parameters, e.g., minimal encryption strength

REST

- Usually it is API documentation on the web
- Level 3 self-describing services
- Meta-language standards for tool developers, like WADL
- Additional QoS out of scope

SOA

- WSDL
- Additional QoS through standards-based extensions
- ESBs typically provide additional QoS

3) Service Discovery

- Mechanism by which clients discover & bind to services
  - Must uniquely identify service
  - May also include meta-information about the service

REST

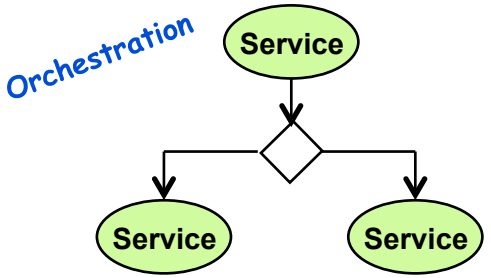
- single endpoint + navigation + semantic formats
- Or web page for documentation, describes where endpoints are
- Consistent formatting of endpoints.

SOA

- UDDI
- WS-Policy
- Or like REST, just publish your WSDLs to a URL

4) Service Composition

- Create a service from the composition and orchestration of lower-level services
  - Orchestration defines centrally controlled process flow for services
  - Choreography coordinates distributed services

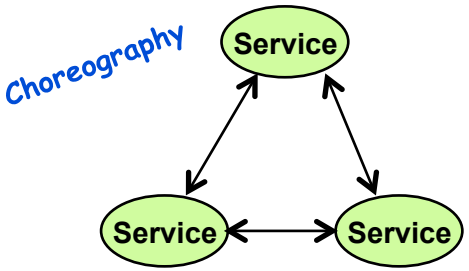


SOA

- In your app
- Choreography suggests services know about each other.
- Orchestration suggests a single conducting service, usually in an ESB
- Standards like BPeL or WS-Flow

REST

- In your app



5) Other Common Services Aspects

- Leasing
  - Service usage granted for fixed period
    - Allows service implementation to expire information it may be managing
    - Allows SOA systems to self-heal when services and connections die
- Events
  - Asynchronous communication eliminates need for polling
- Security, reliable messaging, transactions
  - Provides common credentialing and state for these commonly used features

REST

- Relies on HTTP or best practices
- For example, A&A is not in REST per se, but many use Oauth with a temporal token to approximate leasing

SOA

- Again, typically a standard for everything
- SOA is message-based so has more natural means for providing asynchronous behavior

Principles of SOA Revisited

Stateless interactions

- Services are self-contained and do not store state between invocations
- State should be managed in the business process (e.g. orchestration or choreography) and provided to individual services

3 Modern Flavors

1. Communicate through normalized message broker (ESB)
   - SOAs provide an independent protocol layer (normalized messages)
   - Adapters may be required to convert legacy applications to participate
   - Message-oriented, stateless communication; focus on the message, not the protocol
2. Use REST or similar API-driven services
   - Rely on established best practices instead of heavy standards-based technology stacks
   - Ease the burden of deployment and troubleshooting on your ops staff
   - Scale-up/down (elasticity, cloud) on demand
3. Microservices as a way to promote full lightweight decoupling
   - Assign a team, choose a language/platform, expose a behavior
   - Lightweight migrant deployment – the rise of the container

Trends

The Recent Past

- Application Containers
- Components behind Services
- Assembly and Composition
- Ops: Virtualization
- Values
  - Flexibility (dev & ops)
  - Scalability
  - Decomposition

We join this show already in progress…

The Next Wave

- YAGNI
- Single-process, Single-server
- CCC – Commoditization, the Cloud, and Containerization
- Lean Configuration
- Values
  - Speed (DevOps)
  - Scalability
  - Decomposition – yes, but inversion of the process space

Summary

- Services-oriented computing is an exercise in managing complexity
- Understand how the 5 SOA Architecture Principles apply in each context
- Solutions driven by economics as much as by technology