Introduction to Service Oriented Architectures (SOA)

Responsible Institutions:

ETHZ
(Concept)

ETHZ
(Overall)

ETHZ
(Revision)
# Content

1. Introduction to Service Oriented Architectures (SOA) ................................................................. 3  
   1.1. Architectural constraints ........................................................................................................ 4  
      1.1.1. Interfaces ...................................................................................................................... 5  
      1.1.2. Messages .................................................................................................................... 6  
   1.2. Service Oriented Architecture ................................................................................................ 7  
      1.2.1. Basic SOA .................................................................................................................... 8  
      1.2.2. Extended SOA ........................................................................................................... 9  
      1.2.3. Web Services ............................................................................................................. 10  
   1.3. SOA and ORCHESTRA .......................................................................................................... 11  
      1.3.1. ORCHESTRA is SOA! ............................................................................................... 12  
   1.4. SOA and OASIS .................................................................................................................. 13  
      1.4.1. OASIS Definition of SOA .......................................................................................... 14  
   1.5. Summary ............................................................................................................................ 16
1. Introduction to Service Oriented Architectures (SOA)

Introduction

Service Oriented Architecture (SOA) is a key concept of modern information technology. This lesson explains the basic ideas behind SOA and shows how they are implemented in ORCHESTRA. Finally, the advantages and disadvantages of SOA for risk management are discussed.

Current software has to cope with the heterogeneous nature of the internet, addressing complex issues such as distributed software, application integration, diverse platforms and protocols, and various devices. SOA along with web services allows seamless integration by abstraction from complexity thus providing an approach to deal with the challenges of such complex software environments.

Service Oriented Architecture (SOA) is an architectural style based on loosely coupled interacting software components that provide services. A service is a piece of functionality made available by a service provider in order to deliver end results for a service consumer. A service consumer sends a service request to a service provider. The service provider returns a response to the service consumer containing the expected results.

In service oriented computing (SOC), services are the crucial element to develop applications. SOC applies SOA to organize software applications and infrastructure into a set of interacting services.

References to this lesson:

Learning Objectives

- Architectural constraints
- Service Oriented Architecture
- SOA and ORCHESTRA
- OASIS Definition of SOA
1.1. Architectural constraints

Introduction

In order to achieve loose coupling between components, SOA is based on two major architectural constraints where:

1. Interfaces are defined for all participating services. Only generic semantics are encoded at the interfaces. The interfaces should be universally available for all providers and consumers.
2. Messages are described and constrained by an extensible scheme and delivered through the interfaces.
1.1.1. Interfaces

The interface constitutes a contract defining the functionality of the service in a platform-independent manner. This implies that the invocation mechanism (protocols, descriptions, and discovery) must comply with widely accepted standards enabling a client to use the service from anywhere applying any OS or programming language.

A discovery service (e.g. a directory service) provides clients with a look-up mechanism supporting dynamic locating and invoking.

Services are self-describing. They advertise the service capabilities, interface, behaviour, and quality. Services may publish several descriptions. The service interface description publishes the service signature, e.g. its input, output, and error messages. The (expected) behavior is described by the behavior description and the QoS (Quality of Service) describes both functional and non-functional service quality attributes, e.g. performance, security attributes, reliability, etc.

Services exhibit several other properties. They are stateless, this means that users can use them without knowing the current conditions of the service, the service maintains its own state. The interaction between services is loosely coupled, that is the services must not share common modules (e.g. GUI or storage) or data model. The usage of services is location-transparent, e.g. clients do not have to know if the service is local or only accessible over a network. These properties enable and support rapid and low-cost composition of services for distributed applications.
1.1.2. Messages

Message passing is a form of communication for inter-module interaction. Processes communicate with each other by sending and receiving messages, where each sent mechanism must match the corresponding receive mechanism. Services communicate with each other and with consumers using messages. The service interface defines the messages a service can process. To achieve platform-and language-independency, messages are typically constructed using XML documents that comply with the corresponding XML Schemas. In contrast to Remote Procedure Call (RPC) the mechanism is an asynchronous communication, directly supported by message passing.

A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without modifying existing services.
1.2. Service Oriented Architecture

Introduction

SOA applies a service model consisting of a set of interconnected services communicating through standard interfaces and messaging protocols. Basic services, their descriptions and basic operations as publish, discovery, selection, and bind constitute the basic SOA.
1.2.1. Basic SOA

SOA constitutes a concept to provide services to clients through published interfaces and to coordinate interaction through the exchange of messages. Generally, the basic SOA describes the relationship between three kinds of participants: the service providers, the registry, and the service requestors. The service represents a logical separation of declaration and implementation, its implementation is hidden from the client and can be subject to changes which may not influence the client so long as the service interface stays unchanged.

An important mechanism in a SOA is the Dynamic Discovery of services:

The interaction model of the basic SOA consists of three key players, the service providers, the service requestors, and the intermediating directory service. First, the service providers register with the directory service, then clients can query the directory service for providers and browse the exposed service capabilities. Typically a directory service supports:

- a look-up service for clients
- scalability of the service model: services can be added incrementally
- dynamic composition of the services: the client can decide at runtime which services to use

### 1.2.2. Extended SOA

The higher layer of the SOA pyramid provides support for service composition and management, and service orchestration, transaction, and security. In the composition layer several atomic services can be consolidated into one composite service. Depending on their requirements clients apply atomic or composite services as applications and/or solutions. Service aggregators may utilize such composite services as components in further service compositions thus becoming service providers by publishing the service description they create.

A composer of several services must encompass functionalities such as:

- **Coordination**: establish and manage the control of data flow among the services.
- **Monitoring**: subscribe to events generated by component services.
- **Conformance**: ensure integrity of composite service by controlling conformance of component services.
- **Quality of Service (QoS)**: bundle QoS of component services to derive the composite QoS, e.g. performance, security, authentication, privacy, scalability, availability, etc.

---

**Extended SOA**

1.2.3. Web Services

The application of Web services allows the constitution of an SOA. In general, a Web service is a specific kind of service which can be identified unambiguously by an URI and which uses Internet standards such as HTTP for transport.

The World Wide Web Consortium (W3C) provides a more specific and accurate definition: “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.” [W3C Web Services Architecture Group]

According to this definition Web services are built on top of well-known and platform-independent protocols fulfilling the key requirements of an SOA: the dynamic discovery and invocation of a service is provided by UDDI, WSDL, and SOAP. The usage of XML supports the required platform-independence, and HTTP offers internet-wide interoperability.

In conclusion, web services typically interact applying SOAP messages to exchange XML data. The web services interfaces can be described using the Web Service Definition Language (WSDL) while the Universal Description, Discovery, and Integration (UDDI) standard constitutes a protocol for directory services enabling clients to locate web services and examine the details. By chaining different web services clients can generate new composited web services defining the interaction through the Business Process Language for Web Services (BPEL4WS).
1.3. SOA and ORCHESTRA

Introduction

For each ORCHESTRA service, a platform-neutral abstract specification and a specific implementation specification are available. By defining an implementation specification for the Web services such as platform mapping (WSDL/SOAP/UDDI), ORCHESTRA provides well defined service interfaces, service and data definitions.
1.3.1. ORCHESTRA is SOA!

ORCHESTRA follows the principles of SOA as described above and has all its elements:

- The ORCHESTRA catalogue service acts as a service registry. A difference with respect to SOA theory is that the catalogue service administrator registers available services. In SOA theory, new services subscribe themselves at the service registry.
- Service providers: The ORCHESTRA project also provides an implementation of ORCHESTRA services
- Service requestors: Client applications or ORCHESTRA services using other services. The ORCHESTRA pilots are examples of client applications.

ORCHESTRA uses elements of extended SOA. Services may be composite and therefore invoke other ORCHESTRA services. For example the Map and Diagram Service may visualize geographic objects accessed through the Feature Access Service, which in turn may request input features from a Sensor Access Service. Monitoring of ORCHESTRA services can be done using the Service Monitoring Service. Security is provided by the User Management, Authorisation and Authentication Services.
1.4. SOA and OASIS

Introduction

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. In general, entities (people and organizations) create capabilities to solve or support a solution for the problems they face in the course of their business. It is natural to think of one person’s needs being met by capabilities offered by someone else; or, in the world of distributed computing, one computer agent’s requirements being met by a computer agent belonging to a different owner. There is not necessarily a one-to-one correlation between needs and capabilities; the granularity of needs and capabilities vary from fundamental to complex, and any given need may require the combining of numerous capabilities while any single capability may address more than one need.
1.4.1. OASIS Definition of SOA

The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs. Visibility, interaction, and effect are key concepts for describing the SOA paradigm. Visibility refers to the capacity for those with needs and those with capabilities to be able to see each other. This is typically done by providing descriptions for such aspects as functions and technical requirements, related constraints and policies, and mechanisms for access or response. The descriptions need to be in a form (or can be transformed to a form) in which their syntax and semantics are widely accessible and understandable.

Whereas visibility introduces the possibilities for matching needs to capabilities (and vice versa), interaction is the activity of using a capability. Typically mediated by the exchange of messages, an interaction proceeds through a series of information exchanges and invoked actions. There are many facets of interaction; but they are all grounded in a particular execution context – the set of technical and business elements that form a path between those with needs and those with capabilities. This permits service providers and consumers to interact and provides a decision point for any policies and contracts that may be in force.

The purpose of using a capability is to realize one or more real world effects. At its core, an interaction is “an act” as opposed to “an object” and the result of an interaction is an effect (or a set/series of effects). This effect may be the return of information or the change in the state of entities (known or unknown) that are involved in the interaction.

We are careful to distinguish between public actions and private actions; private actions are inherently unknown by other parties. On the other hand, public actions result in changes to the state that is shared between at least those involved in the current execution context and possibly shared by others. Real world effects are, then, couched in terms of changes to this shared state.

The expected real world effects form an important part of the decision on whether a particular capability matches similarly described needs. At the interaction stage, the description of real world effects establishes the expectations of those using the capability. Note, it is not possible to describe every effect from using a capability. A cornerstone of SOA is that capabilities can be used without needing to know all the details.

This description of SOA has yet to mention what is usually considered the central concept: the service. The noun “service” is defined in dictionaries as “The performance of work (a function) by one for another.” However, service, as the term is generally understood, also combines the following related ideas:

- The capability to perform work for another.
- The specification of the work offered for another.
- The offer to perform work for another.

These concepts emphasize a distinction between a capability and the ability to bring that capability to bear. While both needs and capabilities exist independently of SOA, in SOA, services are the mechanism by which needs and capabilities are brought together.

SOA is a means of organizing solutions that promotes reuse, growth and interoperability. It is not itself a solution to domain problems but rather an organizing and delivery paradigm that enables one to get more value from use both of capabilities which are locally “owned” and those under the control of others. It also enables one to express solutions in a way that makes it easier to modify or evolve the identified solution or to try alternate solutions. SOA does not provide any domain elements of a solution that do not exist without SOA.

Note that while an SOA service brings together needs and capabilities, the provider of the
underlying capability may not be the same entity that eventually provides the service which accesses that capability. In reality, the entity with the domain expertise to create, maintain, and evolve a given capability may not have the expertise or the desire to create, maintain, and evolve its service access.

The concepts of visibility, interaction, and effect apply directly to services in the same manner as these were described for the general SOA paradigm. Visibility is promoted through the service description which contains the information necessary to interact with the service and describes this in such terms as the service inputs, outputs, and associated semantics. The service description also conveys what is accomplished when the service is invoked and the conditions for using the service.

In general, entities (people and organizations) offer capabilities and act as service providers. Those with needs who make use of services are referred to as service consumers. The service description allows prospective consumers to decide if the service is suitable for their current needs and establishes whether a consumer satisfies any requirements of the service provider.

(Note, service providers and service consumers are sometimes referred to jointly as service participants.)
1.5. Summary

Advantages and disadvantages:
SOA in combination with Web services is an approach to deal with the increasing complexity of current software environments. Organisations applying software applications interacting over the internet must cope with complex problems such as system availability, reliability, scalability, and application integration. SOA provides a mature and scalable solution to the occurring problems by reducing their complexity.
SOA can be slow if large datasets have to be transferred between services; a situation which is typical for geospatial services. SOA services are independent of each other and consequently they cannot exchange information by passing storage addresses such as monolithic programs do. However, a possible solution to this problem is to pass references to data processing services instead of the data itself.
Client applications may not be able to work if a service on a remote server is temporarily not available.