Service Computing: Basics

Monica Scannapieco
Generalities: Defining a Service

• Services are
  • self-describing, open components that support rapid, low-cost composition of distributed applications.

• Since services may be offered by different enterprises and communicate over the Internet, they provide a
  • distributed computing infrastructure for both intra and cross-enterprise application integration and collaboration.

*Guest editorial. In [CACM03]*
Generalities: Defining a Service

- Service descriptions are used to advertise the service capabilities, interface, behavior, and quality.
  - Publication of such information about available services provides the necessary means for discovery, selection, binding, and composition of services.
  - In particular, the service capability description states the conceptual purpose and expected results of the service (by using terms or concepts defined in an application-specific taxonomy).
  - The service interface description publishes the service signature (its input/output/error parameters and message types).

Guest editorial. In [CACM03]
Generalities: Defining a Service

- Service descriptions are used to advertise the service capabilities, interface, behavior, and quality.
  - The (expected) behavior of a service during its execution is described by its service behavior description.
  - The Quality of Service (QoS) description publishes important functional and nonfunctional service quality attributes
- Service clients (end-user organizations that use some service) and service aggregators (organizations that consolidate multiple services into a new, single service offering) utilize service descriptions to achieve their objectives.

*Guest editorial. In [CACM03]*
Generalities: Defining a Web Service

• The application on the Web of a service is manifested by Web services

• Web service

  • **Software component available on the Web, to be invoked by some other client application/component**

  • **A way of building Web-scale component-based distributed systems**
Generalities: Principles of Service Computing

- Self-describing: description of their functionality and the location (the application server to which the service is deployed) from which they can be accessed so that applications can use them.
- Service-oriented: Web services are developed to offer a particular service over the Internet. For example, a currency converter can be a Web service.
Generalities: Principles of Service Computing

• Component-based:
  • **Web services are the evolution of some distributed computing models such as**
    • Common Object Request Broker Architecture (CORBA)
    • Component Object Model (COM)
    • Distributed Component Object Model (DCOM)
  • **Web services are platform and language independent. But unlike these technologies, Web services are not accessed via specific protocols but rely on standard protocols**
SOA (Service Oriented Architecture)

• Service Oriented Architecture (SOA) is an architectural approach to viewing and creating a business solution as a network of services.
• In a SOA, services can be
  • distributed across geography, enterprises, and disparate IT systems
  • reconfigured into new business processes
SOA (Service Oriented Architecture)

- These services are built on open standards and loosely coupled
  - easily combined both within and across enterprises to create new business processes
SOC (Service Oriented Computing)

• Service Oriented Computing (SOC) is a new generation computing paradigm to building distributed applications
Putting pieces together...

1. **Service-orientation** is a way of thinking in terms of services and service-based development and the outcomes of services.

2. **Service-Oriented Architecture (SOA)** is an architectural style that supports service orientation.

3. **Service Oriented Computing (SOC)** revolves around the realisation of Service-Oriented business solutions and its associated strategic goals, by applying a Service-Oriented architectural model that complies with Service-Orientation design principles.
Principles of Service Design
Contracts - 1

• A service contract is comprised of one or more documents that provide information about the service

• The fundamental document is the service interface, a standards based technical specification which defines the capabilities of the service to the service consumer
Contracts - 2

• A service provider can also publish non-technical documents as part of the service contract
  • SLA (Service Level Agreement) which provides additional service metadata such as the non-functional, QoS (Quality of Service) features, service availability, behaviours and limitations
  • For example, a service implemented as a web service uses a WSDL (Web Service Description Language) to describe the functionality of the service.
Service coupling

• Coupling refers to a connection or relationship between two things.

• Loosely coupling prescribes minimum dependency between the service contract, its implementation and its service consumers.

• Loosely coupled architectures provide several benefits

  • Loosely coupled components are less dependent on their environment and capable of autonomous functionality, thereby increasing their potential for reuse.
Service abstraction

- This principle advocates that services be regarded as **black boxes** whose functionality is published to service consumers through well-defined service contracts and the underlying implementation be hidden from the consumers.

- Abstraction enables developers to leverage technological advances to provide new improved service implementations, with **minimal impact** to service contract and service consumers.
Service statelessness

• In order to design **scalable** services by freeing up services from storing their **state information** whenever possible

• “Functional” style for service behaviour
Service reusability

- This principle is strongly emphasised in the design of services
- Designing services as generic, reusable enterprise resources with agnostic functional context maximises their potential for reuse as they will be less dependent on any one particular business process
Service autonomy

- Different levels:
  - Shared logic
  - Shared infrastructure
  - Autonomous
Service Composability

- Services are expected to be effective composition members, as part of a richer, complex composite service
- Services must be very composable regardless whether or not immediate composition requirements are already in existence
Quick overview of Web Services (main) standards
A Minimalist Infrastructure for Web Service

SOAP-based middleware

SOAP messages exchanged on top of HTTP, SMTP, or other transport

converts procedure calls to/from XML messages sent through HTTP or other protocols.
SOAP

- SOAP is based on **message exchanges**.
- Messages are seen as **envelopes** where the application encloses the data to be sent.
- A SOAP message consists of a SOAP of an `<Envelope>` element containing an optional `<Header>` and a mandatory `<Body>` element.
- The contents of these elements are application defined and not a part of the SOAP specifications.
- A SOAP `<Header>` contains blocks of information relevant to how the message is to be processed. This helps pass information in SOAP messages that is not application payload.
- The SOAP `<Body>` is where the main end-to-end information conveyed in a SOAP message must be carried.
SOAP Communication Model

- SOAP supports two possible communication styles:
  - remote procedure call (RPC) and
  - document (or message).
**RPC-style SOAP Services**

- A remote procedure call (RPC)-style Web service appears as a remote object to a client application. The interaction between a client and an RPC-style Web service centers around a service-specific interface. Clients express their request as a method call with a set of arguments, which returns a response containing a return value.
Document (Message)-style SOAP Services

- In the document-style of messaging, the SOAP <Body> contains an XML document fragment. The <Body> element reflects no explicit XML structure.
- The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message.
Web Service Definition Language (WS-DL)

- WS-DL provides a framework for defining
  - **Interface**: operations and input/output formal parameters
  - **Access specification**: protocol bindings (e.g., SOAP)
  - **Endpoint**: the location of service
<operation name="orderGoods">
  <input message="OrderMsg"/>
</operation>

From Interfaces to Stub/Skeleton

WSDL of service provider

WSDL compiler (client side)  WSDL compiler (server side)

service requestor

application object (client)

stub

SOAP-based middleware

SOAP messages

service provider

application object (service provider)

skeleton

SOAP-based middleware
**Business Process Execution Language for Web Services (WS-BPEL)**

- Allows specification of composition schemas of Web Services
  - **Business processes as coordinated interactions of Web Services**
  - **Business processes as Web Services**
- Allows abstract and executable processes
- Influenced from
  - **Traditional flow models**
  - **Structured programming**
  - **Successor of WSFL and XLANG**
RESTful Services (1)

- REST refers to simple application interfaces transmitting data over HTTP without additional layers as SOAP
  - Web page meant to be consumed by program as opposed to a Web browser or similar UI tool
  - require an architectural style to make sense of them (the REST one), because there’s no smart human being on the client end to keep track
RESTful Services (2)

- Metaphor based on nouns and verbs
  - URIs ~ nouns
  - Verbs describe actions that are applicable to nouns
    - GET -- retrieve information / READ, SELECT
    - POST (PUT) – add/update new information / CREATE, INSERT, UPDATE
    - DELETE -- discard information / DELETE

- State means the application/session state, maintained as part of the content transferred (in XML) from client to server back to client
RESTful Services (3)

• REST is, in a sense, a kind of RPC, except the methods have been defined in advance
  • Consider the stock example of a remote procedure called “getStockPrice”
  • It's not clear what it means to GET, PUT, and POST to something called "getStockPrice"
  • But if we change the name from "getStockPrice" to "CurrentStockPrice" all is well !!
Statistical Services: Design issues
Statistical Services Design Issues: data awareness?

- Statistical services do have to take exchanged data into account
  - A step beyond generic services
- Statistical data standards
  - Conceptual level: GSIM
  - Logical/implementation level: SDMX and DDI
Statistical Services Design Issues: strong or weak data awareness?

- A «strong» data aware service integration must take fully data exchanged by services into account
  - Format
  - Schema (i.e., domain specific concepts)
  - Model (i.e., dataset structure)
- A «weak» data aware service integration can focus on some aspects of data (mainly format)
  - Tradeoff with automation
Statistical Services Design Issues: Service granularity

• Fine-grained services (e.g., a function sorting a dataset with respect to a set of variables)

• Course-grained services (e.g., a full-fledged GUI-based record linkage system)

• Is it really an issue?
  • Conformance to a «contract»
Statistical Services Design Issues: Contracts

- Example of contract rules for statistical services
  - Design-time service output definition
  - Number and nature of service inputs
  - ...
References


• http://www.whatissoa.com/

• Thomas Erl: SOA: Principles of Service Design