Service-Oriented Computing (SOC)

Service-Oriented Computing (SOC)

Software as a Service

- A computing paradigm revolving around the concept of software as a service
- Assumes that entire systems of software are built and deployed as network-accessible services
- Focus is placed on the utility of software components, rather than on mechanisms of software
- The value of an application is measured in terms of ability to integrate into the environment rather than the application's functional capabilities

SOC Software Requirements

- Technology neutral
  - use widely available, standardized technology
- Loosely coupled
  - minimize formal knowledge required to use a service
- Location transparent
  - be publicly discoverable
  - provide QoS regardless of location of service or clients

Web Services

- Platform independent, network accessible service
- Typically stateless (extensions exist)
- Name derived from traditional use
- Firewall friendly
- Machine-to-machine interaction (not human-to-machine)
- Focused on integration rather than performance
- Two main types: SOAP and REST
Service-Oriented Computing (SOC)

Web Service Technologies

Extensible Markup Language (XML)

Platform independent
- Text resolved
- Short lived
- Limited in size (kilobytes)
- Descriptive

Designed to structure and describe data
- A family of related technologies
- Widely used in a range of technologies today
- Often used for data validation and migration

XML

- Tags are user specified (not predefined)
- XML documents are hierarchical and extensible
  - applications can parse extended documents
  - unrecognized extensions are ignored
- XML is a platform, software, and hardware independent
- XML is used to structure, store, and send information
- XML is both human and machine-readable

Well-formed: conforms to XML syntax specifications
e.g., tags properly nested, tags closed, attributes quoted

Valid: conforms to a namespace-defined syntax
i.e., document matches a syntax definition (schema)

XML Example

XML Parsers

- Programs that traverse XML documents and retrieve data
- Usually transforms data into a custom format
- Can ignore data and insert default values
- Usually slow & memory consuming
- Should incorporate extensive fault tolerance techniques
Simple API for XML (SAX)

- Uses an event-driven parse model
- Performs online parsing of data streams
- Requires very little memory
- Somewhat cumbersome to use

Document Object Model (DOM)

- Constructs a node tree from a document
- Allows off-line processing of data
- Requires large amounts of memory
- Simple to use

XML Path Language (XPath)

- Expression language for addressing parts of XML documents
- Used for pattern-based searching of XML documents
- Possible to search on elements, attributes, and tag data
- Integrates well with the DOM API

XML Query Language (XQuery)

- An query language for XML documents (SQL for XML)
- Used for structured data access in XML documents
- Uses the same data model as XPath
- Contains simple programming language constructs
- Integrates well with the DOM API

eXtensible Stylesheet Language Transform (XSLT)

- A transformation language for XML-based data
- Used to restructure XML documents
- Can produce data in any (text-based) format
- Uses XPath internally to address data
- Typically use to create data presentation templates

XML Schema

- Describes the structure of an XML document
- Is used to validate XML documents
- Is itself an XML document
- Typically difficult to read
- Defines elements, attributes, and data types
SOAP Web Services

Technologies

XML Schema Basics

- Schemas validate documents
- Schemas contain
  - Type definitions
  - Element declarations
  - Attribute declarations
- Simple types
  - Cannot have sub-elements (no child elements or attributes)
  - Predisefined type or derived from predefined type
- Complex types
  - Can have sub-element(s) - elements and/or attributes
- Element declarations can reference both simple and complex types
- Attributes can only reference simple types

SOAP

- Formerly known as Simple Object Access Protocol
- Originally designed as a communication protocol without symmetrical requirements, i.e., a protocol that didn’t require communicating nodes to use the same distributed object model
- XML-based protocol to invoke Web Services
  (XML-serializes Web Service requests / responses)
- Usually transported via HTTP (in HTTP body)
- Can send messages
  - point-to-point (directly to receiver)
  - via intermediaries (via chains of actors)
- Typically uses request-reply messaging, but can use any communication pattern

Soap Communication Stack

Service Descriptions

- Provides all information required to invoke a service
- Defines a type system for payload data
- Hosted by service registries
- Provides a mechanism to decouple services and clients
  - interface abstraction
  - modularization
  - separation of concerns

Service Description Content

- Machine-understandable interface specification
- Details structure, operational characteristics, and non-functional properties of a service
- Describes wire format and transport protocols
- Payload data type system
- Metadata (optional)
  - e.g., behavioral data, policy descriptions

Web Service Description Language (WSDL)

- XML (Schema)-based language for describing Web Service interfaces
- Standardized by the World Wide Web Consortium (W3C)
  - XML Schema provided for WSDL validation
- Completely describes the Web Service interface
- Constitutes a “contract” between the client and the service
- Describes what, where, and how of Web Services
  - what the service does (what functionality is offered)
  - where the service is located
  - how to invoke the service (data formats, protocols)
- Two major parts
  - abstract / interface (types, messages, interfaces)
  - concrete / implementation (bindings, services)
WSDL Elements

WSDL Types

WSDL Messages

WSDL Interfaces

WSDL Bindings
Today
Service-Oriented
Computing (SOC)
Web Service
Technologies
Extensible Markup
Language (XML)
SOAP Web Services
Web Service Description
Language (WSDL)

- Two main approaches
  - generate WSDL from code
  - generate code (stubs) from WSDL
- Generated WSDL tend to be platform / tool-dependent
  (quick and easy, but incompatibility issues may arise)
- Generating stubs from WSDL ensures compatibility
  (but require more work from all parties involved)
- **GOAL:** interoperability (favor the WSDL approach)

Web Service Invocation (stubs)

1. Locate Web Service (discovery)
2. Obtain WSDL description
3. Generate message stubs from WSDL description
4. Use stubs to invoke Web Service methods
5. Use stubs to parse response

Web Service Invocation (API)

1. Locate Web Service (discovery)
2. Obtain WSDL description
3. Create request message document
4. Use API to invoke Web Service methods
5. Parse response message document

Communication Patterns
SOAP Terminology

- Method
  - HTTP message (request / response) that complies with the SOAP encoding rules
- Endpoint
  - HTTP-based URL identifying method invocation target
- Intermediary
  - SOAP node operating on a SOAP message
- Message path
  - the SOAP message route (node list)
- Includes sender, intermediaries, and endpoint.
- Fault
  - Detailed error message
  - Comparable to a Java exception

SOAP Messages

- Outer layer
  - technically not part of the SOAP message
  - e.g., HTTP, SMTP, RPC data (SOAP overhead)
- Envelope (message root element)
- Header (optional)
  - message delivery and processing metadata
  - e.g., factorization information, recipient list
  - extensions for security, transactions, QoS, etc. exist
- Body
  - application specific data (message payload)
  - e.g., XML elements, Faults (error messages)

SOAP Message

- Envelope (message root element)
  - http://schemas.xmlsoap.org/soap/envelope/
- Body
  - Request/Response data
- Header (optional)
  - service information, requestor credentials, etc.
- Faults
  - SOAP faults are propagated back to sender along the same path as the message was delivered

SOAP Faults

- Faults reported in SOAP message body
- Error messages
- Comparable to Java exceptions
- Fault information
  - faultcode / Code: error identifier
  - faultstring / Reason: human readable identifier
  - faultactor: origin of error
  - detail / Detail: additional fault information (optional)

SOAP Fault Example

- E.g., XML elements, Faults (error messages)

SOAP Message Processing

- SOAP nodes may intercept, alter, and send messages
- SOAP intermediaries take on roles in message processing
- Roles are assigned in SOAP message headers
- Messages marked using the header attribute mustUnderstand must be processed according to specification by all nodes, or a SOAP fault must be returned
- Message payloads in the SOAP body may also be altered or processed by any node in the intermediary chain
- Message bodies carry either data or faults, not both
- SOAP faults are propagated back to sender along the same path as the message was delivered
SOAP Web Services

SOAP Communication Styles

- **RPC**
  - Utilizes automated serialization of a method invocation
  - Details serialization rules for method invocations
    - URI identifying transport address for the call is required
    - All parameters modeled as fields in a single struct
    - Parameter names and order must correlate to that of the method involved
  - Enforces tightly coupled, bilateral interaction between client (requestor) and service

- **Document (Message)**
  - Represents arbitrary message serializations
  - Typically sends entire documents rather than discrete sets of parameters in message payloads
  - Assumes messages are well-formed XML documents
  - Results in lossy coupled, message-driven interactions

SOAP Encoding Styles

- **Literal**
  - XML messages encoded as-is
    - Recognized in WS-I Basic Profile (WS-Interoperability)
- **Encoded**
  - Messages encoded using proprietary encoding schemes
    - Prohibited in WS-I Basic Profile (WS-Interoperability)

SOAP Message Processing

Wire Protocols

- **Wire protocols**
  - Concerned with data format and payload structure
    - e.g., method parameter serialization, message interpretation, etc
- **Transport protocols**
  - Concerned with payload delivery
    - e.g., connection establishment, routing, package size, etc

SOAP is a wire protocol in that it handles XML serialization and encoding of Web Service invocation data

SOAP usually utilizes HTTP as a transport protocol

Wire Protocol Design Criteria

- Compactness
- Protocol efficiency
- Coupling
- Scalability
- Interoperability

A number of trade-offs between these criteria exist and typically results in that no wire protocol performs well on all criteria

SOAP Advantages

- Simplicity
- Portability
- Firewall-friendly
- Open standards
- Interoperability
- Acceptance
- Resilience to changes
SOAP Disadvantages

- Stateless
- Serialization by value
- Inefficient transport (HTTP)
- Protocol overhead
- Memory requirements
- Poor implementations
- Attack sensitivity
- Inefficient representations (binary data)

SOAP vs. REST

- Passionately discussed, polarized debate
  - Google “SOAP vs REST” -> 150k hits
  - Similar to vi vs. emacs discussion
  - Some vendors support both
- Use cases
  - REST
    - Ad hoc Web-style integration
    - Simplicity, scalability
  - SOAP
    - Enterprise integration
    - Advanced capabilities (QoS, transactions, ...)
- Convergence?
  - WSDL 2.0 supports REST
  - Read up, create your own opinion

SOAP Web Service Technology Stack