INTERNATIONAL
Systems Group, Inc.

Modeling, Designing and Implementing Service Based Applications

2-day seminar

Give Your Business the Competitive Edge

SOA has gained momentum because it is seen as the key for enterprises to achieve business agility, improved quality of service and lowered total cost of ownership. Over the recent years, many Fortune companies have started to embrace a SOA for initial development and integration projects. However, just utilizing technologies like Web Services and the latest generation of development tools are not sufficient for successful implementation of an enterprise SOA.

What is required is a consistent approach to architecture, modeling and design that takes a cross-project view on services, providing guidance to critical concepts like service layering and design for reusability. At the same time, traditional object oriented development methodologies such as Rational Unified Process (RUP) need to be significantly adapted, making sure your services are more than just objects with a Web Services interface.

A Service Oriented Software Development Lifecycle

Following a brief introduction the seminar first discusses key architectural guidelines for service oriented design. It covers the important characteristics of loosely coupled architectures and compares today's two mainstream approaches to SOA, namely traditional Web Services based architectures and RESTful architectures. It also outlines the options for moving processing from the server to the Web Services client.

The seminar then addresses how a typical object oriented application development methodology can be modified and adapted such that is suitable for implementing services. This is followed by an illustration of a layered services model, which fosters the separation of services into three distinct layers (orchestration, application, and infrastructure services layer). This separation is key to designing loosely coupled services and to ensure reusability of services across applications or business processes.

A case study is used for a detailed illustration of the modeling and design of a SOA-based B2B gateway. It includes the design of service interfaces, the encapsulation of a legacy system, the definition of XML S1chemas that are broken down into reusable components, the development of business processes, and a walk-through of the complete B2B gateway architecture.

Benefits of Attending

- Learn the key aspects of modeling, designing and implementing services in a SOA
- Understand how a service oriented software development lifecycle is different from its object oriented predecessors.
- Be able to define practical guidelines that can help different project teams make the best architecture, design and implementation choices for SOA.
- Learn about proper service layering and service design for reusability.
- Gain insight into how Data Architecture relates to SOA
- Learn how to design XML Schemas for componentization and reuse.

Who Should Attend

- Architects who want to adopt a Service Oriented Architecture.
- IT professionals who need to see how SOA can be applied to development projects.
- IT Managers who want to adopt a service oriented software development lifecycle.
- Consultants who need to recommend and use different implementation strategies for building a SOA.

Prerequisite: This class requires attendees to have an architectural understanding of Service Oriented Applications and core Web Services standards like XML Schema, SOAP, WSDL, etc.

1. Agenda & SOA Overview

- Agenda
- ISG overview
- SOA defined
  - The changing notion of applications
- Next generation SOA
  - Moving from Client/Server SOA to Event and Service Oriented Architecture (e-SOA)

2. Architectural Guidelines for Service Design

- Overview of loosely coupled architectures
  - Interface style and service interaction metaphor
  - Client/server vs. event-driven SOA
  - Degrees of service orientation
- Traditional Web Services based vs. RESTful architectures
  - Characteristics of traditional Web Services based architectures
  - Characteristics of RESTful architectures
    - The uniform interface
    - The concept of resources
    - Architectural constraints
  - Why WSDL – and the REST alternative
  - Why SOAP – and the REST alternative
- High-level application architecture
  - Considerations for orchestration and application services
  - Stateful vs. stateless services
  - Data access services
- The client tier
  - How clients relate to services in a “Web Oriented Architecture”
  - Format considerations for the data exchange between client and service
    - JSON vs. XML

3. Service Oriented Analysis and Modeling

- Overview of a service oriented Software Development Lifecycle
  - Component vs. service oriented development
- Extending a object oriented software development methodology for SOA
  - Business Modeling Activities
  - Requirements Activities
  - Analysis & Design Activities
    - Mining existing systems
    - Verifying service definitions against SOA principles
  - Implementation Activities
  - Test and Deployment Activities
  - Project Management Activities
  - New roles & responsibilities
- Top-down vs. bottom-up approach
- The service layer model
  - Orchestration services layer
  - Application services layer
  - Infrastructure services layer

4. Analysis and Modeling - Customer Case Study Part I

- Customer overview
- B2B Integration Strategy
- Business Process Walk-Through
- Defining Project Scope
- Business Modeling
- Definition Of Services & Layering
- Verify SOA Principles

5. Service Oriented Design - Case Study Part II

- Guiding principles for service design
- Putting it all together: a customer case study – Part II
  - Designing service interfaces
  - Encapsulation of existing business logic
  - Designing the Schemas
  - Determining the right degree of service orientation
  - Designing the components
  - Composing business processes
  - Detailed B2B Gateway Logical Architecture
  - Conclusions

6. Conclusions

- Summary of lessons learned
  - Summary of architecture choices
  - Where to process – on the client or the server?
  - Layers are good for decoupling and reusability
  - Service development needs a methodology
  - Case study highlights
- Challenges
  - Composite applications
  - Asynchronous service interaction