

# Information Systems Integration and Testing

## Module 1. Foundations and Integration Mindset

These notes establish the terminology, motivation, and architectural background of the course.

- [Introduction](#) - Concept page introducing software integration, interface-driven design, data exchange, and testing concerns.
- [Software Integration](#) - Core theory page on integration problems, legacy systems, middleware, file-based integration, databases, and integration strategies.
- [Evolution of Software Integration methods](#) - Historical and architectural overview showing how integration moved from tightly coupled systems toward middleware and enterprise integration approaches.
- [12-Factor App Methodology](#) - Operational design principles for modern service-based systems, deployment, configuration, and runtime behavior.
- [Semantic Versioning](#) - Compatibility-focused versioning rules for APIs, components, and evolving systems.

## Module 2. Classical Integration and Low-Level Communication

This block covers foundational distributed communication techniques and older but historically important integration models.

- [Object Request Broker \(CORBA\)](#) - Classical middleware and distributed object communication with ORB, IDL, and broker-based invocation.
- [Integration based on TCP/IP Sockets](#) - Low-level point-to-point communication and the basic mechanics of network integration.
  - [Java example for Blocking and Non-Blocking Socket](#) - Practical Java socket example for synchronous and asynchronous communication patterns.
  - [Python example for Blocking and Non-Blocking Socket](#) - Practical Python socket example for blocking and non-blocking communication.
  - [Socket Exercises](#) - Practice tasks for reinforcing socket-based communication.
  - [HTTP server](#) - Example page connecting socket-level understanding with a simple application protocol.
- [Java - Remote Method Invocation](#) - Java-specific remote invocation model, useful for understanding language-bound distributed calls.
  - Original paper: [A Note on Distributed Computing](#)
  - Oracle tutorial: [Java RMI Tutorial](#)
  - [Group chat](#) - Applied example built around the RMI communication model.
  - [Group chat - Nodejs](#) - Comparative communication example using Node.js.
- [Integration based on RPC Remote Process Call](#) - Conceptual transition from raw communication to call-oriented distributed interaction.
  - [XML-RPC example](#) - Structured RPC example using XML-based message exchange.

## Module 3. Modern APIs, Contracts, and Remote Calls

These topics focus on structured service contracts, modern remote procedure calls, and API-oriented integration.

- [Modern Data Integration based on Protocol Buffer](#) - Schema-driven data exchange and compact cross-language serialization.
- [Google's modern remote procedure call technique](#) - Modern RPC based on Protocol Buffers and service contracts.
- [JSON-RPC](#) - Lightweight remote procedure calls over JSON, useful for comparing RPC with REST-style APIs.
- [REST API](#) - Resource-oriented integration with HTTP methods, representations, and stateless interaction.
- [graphql integration](#) - Query-driven API design that lets clients request exactly the data they need.

## Module 4. Deployment and Runtime Integration

Integration is not only about protocols. It also depends on packaging, deployment, runtime isolation, and service-to-service topology.

- [Docker integration techniques](#) - Container-based integration and deployment as a practical runtime foundation.
  - [Simple multi container example](#) - Hands-on example of multi-service container composition.
  - [Load balancing with haproxy](#) - Example of scaling and traffic distribution across service instances.

## Module 5. Messaging, Web Services, and Enterprise Integration

This module moves from direct request-response integration toward asynchronous communication, XML-based service standards, and enterprise-scale middleware patterns.

- [Messaging systems](#) - Asynchronous communication, queues, brokers, and message-based decoupling.
  - [RabbitMQ simple producer and consumer](#) - Introductory example of message publishing and consumption.
  - [RabbitMQ complex example](#) - More advanced messaging scenario and message-flow design.
- [Web services](#) - XML-centered service integration and standards-oriented interoperability.
  - [JAX-WS web service](#) - Java SOAP-style service example.
  - [JAX-RS web service](#) - Java RESTful service implementation example.
- [Enterprise Service Bus \(ESB\)](#) - Centralized enterprise integration with routing, transformation, protocol mediation, and QoS concerns.

From:

<https://edu.iit.uni-miskolc.hu/> - **Institute of Information Science - University of Miskolc**

Permanent link:

[https://edu.iit.uni-miskolc.hu/tanszek:oktatas:iss\\_t:lecture\\_notes?rev=1774895063](https://edu.iit.uni-miskolc.hu/tanszek:oktatas:iss_t:lecture_notes?rev=1774895063)

Last update: **2026/03/30 18:24**

