Grundlagen des Software Engineering
Fundamentals of Software Engineering

Winter Term 2011/12

Prof. Dr. Dr. h. c. Dieter Rombach

Chapter 5.3:
Software System Engineering – Information System Architecture

Last update: 13/11/2011
The goals of this chapter are to be able to:

- Characterize information systems
- Categorize information systems
- Understand architectural styles for information systems
- Understand the concept of Service-Oriented Architectures

Literature

- World Wide Web Consortium (W3C): http://www.w3.org
Introduction and motivation
- Before we start

- We have described
  - Requirement engineering for IS
    → Software engineering principles
    → TORE
  - View styles for architecture
    → Software engineering principles (Divide and conquer, Information Hiding …)
    → Module
    → C&C
    → Allocation

- How to verify and validate the requirements and the system
  - These notions are prerequisites for this chapter

- What we want to explain in this chapter
  - How to realize the requirement for IS
  - How are view styles tailored for IS
  - What techniques exist for implementing IS
"Information systems are implemented within an organization for the purpose of improving the effectiveness and efficiency of that organization. Capabilities of the information system and characteristics of the organization, its work systems, its people, and its development and implementation methodologies together determine the extent to which that purpose is achieved." [silver et al.]
Introduction and Motivation

- Definition
- Application Domains
- Challenges

Architecture styles for IS
- 1-Tier
- 2-Tier
- 3-Tier
- Example

IS and E-Business

Service Oriented Architecture
- Definition
- Services
- Example
- Characteristics
- Implementation
- Web Services

Executive Information Systems
Decision Support Systems
Management Information Systems
Transaction Processing Systems

[Laudon et al.]
Introduction and Motivation

Application Domains

- Expert systems
  - Health information systems
  - AI information systems

- Geographic Information systems
  - Google hearth

- Office automation

- Web information systems

- Date warehouses
  - Inventory management
  - Financial forecasting

- Enterprise systems
  - SAP systems

- …
Introduction and motivation

- Challenges

- High availability

- Security
  - Confidentiality
  - Integrity
  - Availability
  - Authenticity
  - Non-repudiation

- Performance and scalability

- Complexity

- Heterogeneity

- Distribution

...
Introduction and Motivation

Definition

Application Domains

Challenges

Architecture styles for IS

Layered organization of the architecture

- Functionalities are separated into three layers
- Separation of concerns
- Information Hiding
- Distribution

Presentation Layer

- Presents information to the clients
- Accepts requests from the client
- It formats the data set so that it can be displayed on the client’s user’s interface

Application Layer

- Implements the business core of the information system
- Answers the operations requested by the clients

Resource management layer

- Manages and implements the data source of the information system

Layers can be combined or distributed
**Architecture styles for IS - 1-Tier**

- **Presentation, Application and Resource Layer are combined on the same tier**

- **Mainly used on mainframe-based computer architectures**
  - Clients is a dumb terminal

- **Monolithic systems**
  - No entry points from outside
  - Black-box

- **Advantages**
  - Performance optimization
  - Cheap client development
  - Deploying and maintaining the client is not an issue

- **Disadvantages**
  - Difficult and expensive to develop and maintain the single tier
  - Single point of failures
  - The failure of a layer on the mainframe will affect other layers
Architecture styles for IS
- 2-Tier

- Motivated by the emergence of PCs
- Typically realized as Client-Server
  - The presentation layer resides in the client PC
    - Exploit the resources of PCs (Processing power, memory …)
    - More resources are allocated to the application and resource management layer
- Types of clients/server:
  - Thin client / Fat server
  - Fat client / Thin server

- Advantages
  - Portability
  - Performance optimization with a fat server
  - Support of different clients

- Disadvantages
  - Scalability limited
  - Expensive development of fat clients
  - Integration clients are complex
SE – Software Architecture for IS

- Introduction and Motivation
  - Definition
  - Application Domains
  - Challenges

- Architecture styles for IS
  - 1-Tier
  - 2-Tier
  - 3-Tier
  - Example

- IS and E-Business

- Service Oriented Architecture
  - Definition
  - Services
  - Example
  - Characteristics
  - Implementation
  - Web Services

### Architecture styles for IS - 3-Tier

- **Motivated by the increase in network bandwith**
- Clearly separation between the presentation, application and resource management layer
  - Move presentation layer to the client
  - Middleware realizes the application layer
  - Resource management layer is composed of database servers

- **Advantages**
  - Scalability
  - Portability
  - Support of multiple resource managers

- **Disadvantage**
  - Communication overhead

- **N-Tier**
  - Generalizes the 3-Tier architecture
  - The client is a web browser
  - Resource layer may include 1-, 2-, 3-, N-tiered systems
  - Presentation layer may includes a web browser and a HTML filter
  - Commonly 4-Tier architecture

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12
Introduction and Motivation
- Definition
- Application Domains
- Challenges

Architecture styles for IS
- 1-Tier
- 2-Tier
- 3-Tier

Example

IS and E-Business

Service Oriented Architecture
- Definition
- Services
- Example
- Characteristics
- Implementation
- Web Services
Introduction and Motivation

Definition

Application Domains

Challenges

Architecture styles for IS

1-Tier

2-Tier

3-Tier

Example

IS and E-Business

Service Oriented Architecture

Definition

Services

Example

Characteristics

Implementation

Web Services

SE – Software Architecture for IS

Example - Web System - Application layer
Introduction and Motivation
- Definition
- Application Domains
- Challenges

Architecture styles for IS
- 1-Tier
- 2-Tier
- 3-Tier

Example

IS and E-Business

Service Oriented Architecture
- Definition
- Services
- Example
- Characteristics
- Implementation
- Web Services
Web technologies have been widely used for implementing information systems

E-business systems
  – Information systems that supports **business processes** using electronic media or **web technologies**
  – **Stakeholders**
    → Companies
    → Suppliers
    → Customers

Types of E-business information systems
  – Business to Business (B2B)
    → support transaction between business processes of different companies
    → Between a manufacturer, a wholesaler and a bank
  – Business to Consumer (B2C)
    → Support transaction between enterprises and their customers
    → Between a supplier and its customers via the supplier’s website
  – Consumer to Consumer (C2C)
    → Support online dealing of goods among individuals

For each type of E-business services are exchange via web technologies

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12
"Service-oriented architectures are a ways of developing distributed information systems where the system components are stand-alone services, executing on geographically distributed computers," [sommerville]

- **Services** are well defined Business functionalities
Services
- Definition

- Introduction and Motivation
  - Definition
  - Application Domains
  - Challenges

- Architecture styles for IS
  - 1-Tier
  - 2-Tier
  - 3-Tier
  - Example

- IS and E-Business

- Service Oriented Architecture
  - Definition
  - Services
  - Example
  - Characteristics
  - Implementation
  - Web Services

- "A service is a software building block that is well-defined, self-contained" [Erl]

- A service can encapsulate
  - a single standalone unit (process step) of a business automation
  - some or all of the business automation logic

- Example
### Services - Design Principles

- **Loose coupling**: The dependencies on other services should be minimal.

- **Service contract**: Service descriptions should describe a communication agreement to what all services should adhere.

- **Autonomy**: Services have control over the logic they implement.

- **Abstraction**: The logic implemented in the service should be hidden to the outside world. Only the service contract is visible.

- **Reusability**: Enhance reuse by dividing logic into services.

- **Composability**: A set of services can be coordinated and assembled to form composite services.

- **Statelessness**: Services minimize retaining information specific to an activity.

- **Discoverability**: Services should be described so that they can be found and assessed via available discovery mechanisms.

---

**Architecture styles for IS**
- 1-Tier
- 2-Tier
- 3-Tier
- Example

**IS and E-Business**

- **Service Oriented Architecture**
  - Definition
  - Services
  - Example
  - Characteristics
  - Implementation
  - Web Services

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12
- Introduction and Motivation
  - Definition
  - Application Domains
  - Challenges

- Architecture styles for IS
  - 1-Tier
  - 2-Tier
  - 3-Tier
  - Example

- IS and E-Business

- Service Oriented Architecture
  - Definition
  - Services
  - Example
  - Characteristics
  - Implementation
  - Web Services
Ticket Service

Ticket information request by driver

Manager → Look up ticket → Ticket pool

Ticket information not found

Manager → Notify Bus Driver

Ticket information sent

Bus Driver → Get information → Bus Driver is informed

Ticket information found

XOR

XOR

Ticket Service
Introduction and Motivation

Definition

Application Domains

Challenges

Architecture styles for IS

1-Tier

2-Tier

3-Tier

Example

IS and E-Business

Service Oriented Architecture

Definition

Services

Example

Characteristics

Implementation

Web Services

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12

Ticket Service
How should SOA be built? [Erl]

- **Increasing quality of service**
  - Tasks to be performed securely and reliably
  - Performance requirements (avoid overhead when tasks are executed)
  - Transactional capabilities (when a task fail, an exception logic is executed)

- **Autonomous**
  - An individual service is as independent and self-contained as possible

- **Using open standards**
  - Standard open technologies are used within and outside of solution boundaries
  - Data exchange is governed by open standards.
  - Vendor-neutral communications framework.
How should SOA be built? \cite{Erl}

- **Supporting vendor diversity**
  - The open communications framework allow the heterogeneity within (and between) corporations.

- **Promoting discovery**
  - SOA supports and encourages the advertisement and discovery of services throughout the enterprise and beyond.
  - Service descriptions are managed by a service registry or a directory.
How should SOA be built? \cite{Erl}

- **Fostering intrinsic interoperability**
  - When building a SOA application from the ground up, services with intrinsic interoperability become potential integration endpoints.
  - This reduces the cost of future cross-application integration.

- **Promoting federation**
  - Services enable standardized federation of disparate legacy systems.
  - To encapsulate legacy and non-legacy application logic, SOA uses a common, open, and standardized communications framework.
How should SOA be built? \[Erl\]

- **Fostering inherent reusability**
  - reuse on many levels
  - The creation of services that are unforeseen to both the business processes and the automation solutions.

- **Emphasizing extensibility**
  - Extensible services can expand functionality with minimal impact (without breaking the established interface).
How should a SOA be built? [Erl]

- **Supporting a service-oriented business modeling paradigm**
  - Business processes can be represented and expressed through services or a collection of services

- **Implementing layers of abstraction**
  - Business and application logic can be *abstracted* through a dedicated service layer
  - The functionality of services then are used via *service interfaces*
How should a SOA be built? [Erl]

- Promoting loose coupling throughout the enterprise
  - Through the implementation of service layers that abstract business and application logic, the loose coupling paradigm can be applied to throughout the enterprise.
How should a SOA be built? [Erl]

- **Promoting organizational agility**
  - A loosely coupled relationship between business and application technology allows each end to more efficiently respond to changes in the other.
  - Cost of change is reduced.
SOA Implementation

- SOA can be implemented using a wide range of technologies: CORBA, RPC, DCOM, JINI, Web Services, etc.

- No technology has been so suitable and successful in manifesting SOA principles than Web services.

- SOA is an architectural style, whereas Web services is a technology that can be used to implement SOAs.

- Distributed systems technologies of the past, such as CORBA, didn’t achieve broad adoption because standards were not widely endorsed by CORBA vendors.
Web services \[\text{[Erl]} \, \text{[w3c]}\]

- Introduction and Motivation
- Definition
- Application Domains
- Challenges

- Architecture styles for IS
  - 1-Tier
  - 2-Tier
  - 3-Tier
  - Example

- IS and E-Business

- Service Oriented Architecture
  - Definition
  - Services
  - Example
  - Characteristics
  - Implementation

SE – Software Architecture for IS

- “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.”

- WSDL described the service.
- SOAP provided the messaging format used by the service and its requestor.
- UDDI provided the standardized service registry format.

![Diagram of Web Service Architecture]

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12

31