

INFS 3204/7204 Service-Oriented Architecture



A/Prof Heng Tao SHEN
ITEE, UQ
Semester 2, 2011

M7: SOA Development

1

M7 Topics

- SOA
 - Concepts
 - Development of SOA
 - Industrial standards

2

Concepts

- **Service-oriented architecture**
 - An **architectural style** whose goal is to achieve “loose coupling” among interacting and contracted services via communication protocols
- **Architecture**
 - A software architecture is a structure or abstraction of a software system, including:
 - Software components
 - Externally visible properties of those components
 - Relationships among them
 - Constraints on their use

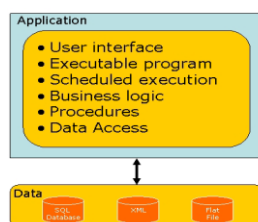
3

Loose Coupling

- Coupling is dependency between interacting systems
 - **Real dependency** is the set of features or services that a system consumes from other systems. It always exists and cannot be reduced.
 - **Artificial dependency** is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. It always exists, but can be reduced.
 - language dependency
 - platform dependency
 - API dependency, etc.
- Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum

4

Traditional Architecture



combines everything into a single program

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/FoodMovers3.asp>

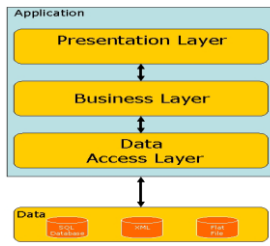
5

Its Problems

- The application's functions cannot be re-used, e.g., the business functions are written for this particular application/platform only
- It is difficult to debug the program as it grows, and maintain it as it is deployed. A change to one part of the code could affect other code
- Security is another problem because the user interface cannot be isolated from the rest of the program
- Scalability is almost impossible because it is difficult to spread any part of the application

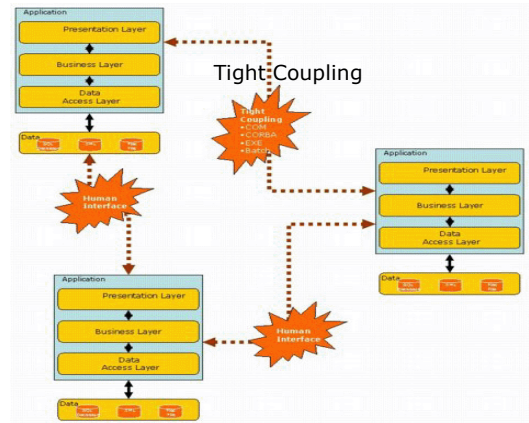
6

Component-Based Application Architecture



First addressed the problem of integrated code by defining layers of functionality

7



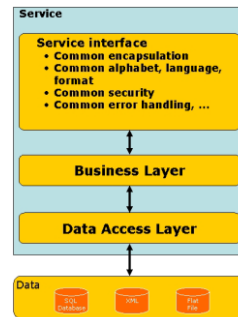
8

Its Problem

- A tightly coupled solution, in which two applications that are being integrated are aware of each others' implementation details, requires custom code that one a human, who reads the output from one application and keys it into another application
 - Allows the components to be re-used on the same platform and usually the same programming language
- **What if...?**
 - we need to share the business functions throughout heterogeneous platforms?
 - we want to share information with our external partners?

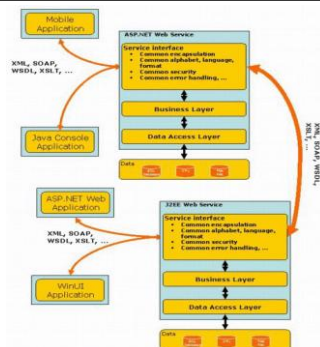
9

Service-Oriented Architecture



10

SOA Integration



11

A Service:

- Has a clearly defined interface, which is exposed through some kind of standard contract
- Is usually discoverable, but not always
- Can correspond to real-life business activities
- Interacts with other services and components using loosely-coupled, message-based architecture
- Uses standards for communication for interoperability
- Is up and running all the time, unlike components that must be instantiated before use

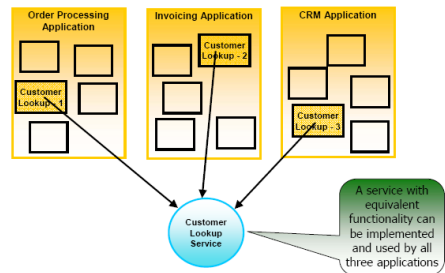
12

SOA Enables:

- Reusability
- Legacy leverage
- Agility
- Loose coupling
- Interoperation

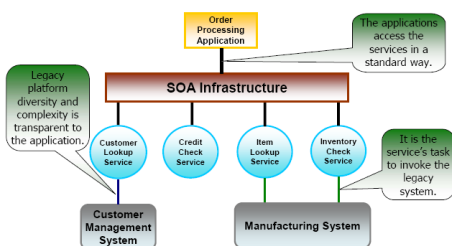
13

Reusability



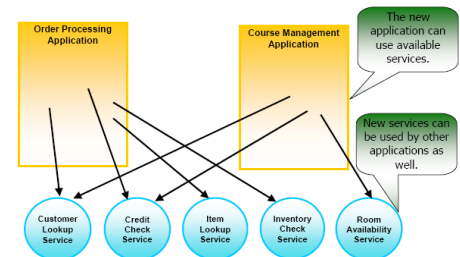
14

Legacy Leverage



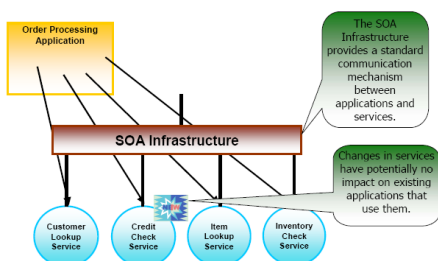
15

Agility



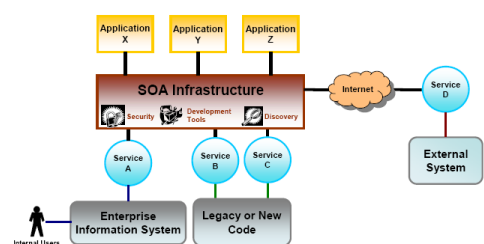
16

Loose Coupling



17

Interoperation



18

Components of SOA

- **Service** in SOA is an exposed piece of functionality with 3 properties:
 - Platform-independent
 - Dynamically located and invoked
 - Self-contained

19

Replaceable Components

- Often need to represent several different implementations of a particular component, where you need to
 - isolate requestors from change in the implementation
 - provide access to many different "flavors" of that component, either at once or over time
- For example – Insurance Broker site
 - You're building a web site that will provide customers with Insurance quotes. To do so you have an agreement with several different Insurance companies to provide you with quotes. But you have a problem; each of the Insurance companies has a different mechanism for accepting quote requests and providing quotes to you

20

Composable Components

- Do you need to take parts of your system and then compose them into a larger system in a configurable way?
- Invoking WS's in a specified order – some services may be skipped (or rerun) depending on results of previous WS
- For example, the Business Process Execution Language for Web Services (BPEL4WS) specification describes how to compose Web Services into workflows

21

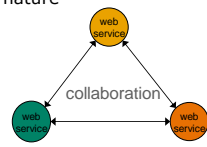
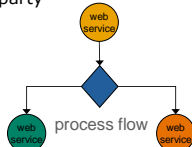
Integration with SOA

- Internal Integration
 - Only worthwhile if the system is already functional and robust
 - Can be done with custom interfaces (e.g. screen scraping) or
 - Using loosely coupled architectures and message-based protocols – internal WS
- External Integration
 - Companies communicate with messages such as
 - contracts
 - purchase orders
 - invoices
 - monetary transactions
 - insurance claims, and many more ...

22

Orchestration vs. Choreography

- Includes execution order of web services interactions
- Describes process flow
- Can include both internal and external web services, but process is always controlled by one party
- Tracks the sequence of messages involving multiple parties and sources
- Associated with public message exchanges, not executable processes
- More collaborative in nature



23

Granularity of Integration

- One great difficulty in developing a SOA lies in determining what the right level of integration granularity
- Needs to be relevant to the service consumer – and combine appropriate flexibility with ease of assembly into the business process
- Often a more "large-grained" approach is better
 - For efficiencies sake you should avoid a fine-grained structure (network latency etc)
 - Protects the consumer from "unnecessary detail"
- A common model for each services is to manipulate significant documents using "query, update, add, and delete" methods (object model approach with separate accessory methods for each property)

24



Granularity Issues

- SOA allows fairly large chunks of functionality to be strung together to form ad hoc applications which are built almost entirely from existing services
 - The larger the chunks, the fewer the interface points required to implement any given set of functionality
 - However, very large chunks of functionality may not be granular enough to be easily reused
- The great promise of SOA is that the marginal cost of creating the n-th application is almost zero, as all of the software required already exists to satisfy the requirements of other applications
 - Only orchestration is required to produce a new application

25



SOA Development

- Service-oriented Analysis
- Service-oriented Design
- Implementation

26



Service-Oriented Analysis

- Step 1: define analysis scope
 - Identify system specifications
- Step 2: define affected systems
 - Identify affected legacy systems and reusable services
- Step 3: perform service modelling
 - Identify services considering
 - Service Reusability
 - Service Autonomy
 - Service Discoverability

27



Service Modelling

- Entity Services
- Task Services
- Utility Services

28



Entity Services

- Defines the organization's relevant business entities
 - E.g., customer, employee, invoice, and claim
- Represents a business-centric service that bases its functional boundary and context on one or more related business entities
- Considered a highly reusable service
- Known as entity-centric business services or business entity services

29



Task Services

- A business service with a functional context and scope
 - E.g., add, search, submit and cancel
- Tends to have less reuse potential and is generally positioned as the controller of a composition responsible for composing other services
- Task services are also known as task-centric business services or business process services

30



Utility Services

- Dedicated to providing reusable, cross-cutting utility functionality
 - E.g., event logging, notification, and exception handling
- Utility services are also known as application services, infrastructure services, or technology services

31



Delivery Processes

- Top Down
 - Tactically focused on that it makes the fulfilment of immediate business requirements a priority and the prime objective of the project
 - Avoids the extra cost, effort, and time
 - Imposes increased governance burden as bottom-up delivered services tend to have shorter lifespans and require more frequent maintenance, refactoring, and versioning

32



Delivery Processes

- Bottom Up
 - Advocates the completion of an inventory analysis prior to the physical design, development, and delivery of services
 - Demands more of an initial investment because it introduces an up-front analysis stage focused on the creation of the service inventory blueprint
 - Individually defined as part of this blueprint so as to ensure that subsequent service designs will be highly normalized and standardized

33



Service-Oriented Design

- Orchestrate the business logic, based on entity, utility, and task services
 - Different methods can be used, e.g., activity diagram (using task services)
 - Complex systems often mix all types
- Service analysis results in a collection of services that establish the starting point for service design

34



Implementation

- Choosing the development environment
 - platform, language, technologies etc.
- Coding & documentation
- Testing
- Running
- ...

35



General Guiding Principles

- For development, maintenance, and usage of the SOA
 - Reuse, granularity, modularity, composability, and interoperability
 - Compliance to standards (both common and industry-specific)
 - Services identification and categorization, provisioning and delivery, and monitoring and tracking

36

Specific Architectural Principles

- For design and service definition focusing on specific themes that influence the intrinsic behaviour of a system and the design style
 - Encapsulation: hide implementation
 - Loose coupling: minimizes dependencies
 - Contract: communications agreement
 - Abstraction: hide logic from the outside world
 - Reusability: logic is divided into services for reuse
 - Composability: services can be assembled
 - Autonomy: services have control over the logic
 - Optimization: high-quality services are preferred
 - Discoverability: services are descriptive

37

Industrial Standards

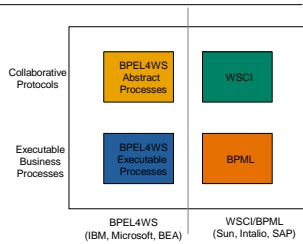
- Orchestration standards can simplify the creation of business processes involving web services
- Many "standards"
 - No clear winner. It is unclear where the industry is going with the various standards
- Many proposals of "standards" – every player tries to get the biggest piece of cake
 - **BPML4WS** (IBM, Microsoft, BEA) – merging of XLANG (Microsoft) and **WSFL** (IBM)
 - **WSCI + BPML** (Sun, SAP, Intalio and BEA!!!) – as W3C Technical report
- Limited number of supporting products (Collaxa, IBM BPWS4J)

From: Chris Paletz "Web Services Orchestration. A review of emerging technologies, tools and standards", Hewlett Packard Co, January 2003

38

Popular Standards

- WSCI/BPML has much richer choreography support and backing by W3C working group
- BPML4WS has major supporters behind it, with developer tools and documentation already available



39

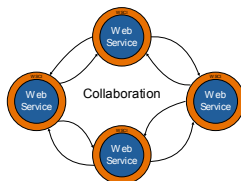
BPML

- Business Process Modeling Language
 - developed by BPMI.org (Intalio, Sterling, Sun, CSC)
 - meta-language for describing business processes
 - language executed by a BPMS system
- Key features
 - basic activities for sending, receiving, and invoking services
 - handles conditional, sequential, and parallel activities
 - persistence, correlation, and composition support
 - supports short and long-running transactions
 - robust exception handling mechanisms
- Incorporated **WSCI** for web services **choreography**

40

WSCI

- Web Services Choreography Interface
- Specification from Sun, SAP, Intalio, and BEA
- Defines an XML language for web services collaboration
- Describes the overall choreography, or observable behavior, between services
 - Does not define executable processes
 - Each partner exposes their WSCI interface
- Layered on top of WSDL



41

WSCI Features

- Support for basic activities:
 - each activity specifies the WSDL operation involved
 - use `<action>` to define a basic request/response message
 - use `<call>` to invoke external services
- Support for structured activities:
 - sequential, parallel, and conditional looping
 - use `<all>` to specify an unordered actions to perform
- Support for business transactions and exceptions:
 - transactional contexts can be defined in WSCI
 - any failure in a context will result in all transactions in context being rolled back

42

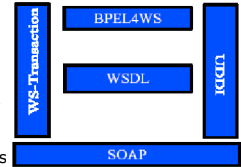
BPEL4WS

- Business Process Execution Language for Web Services
 - specification from IBM, Microsoft, and BEA
 - XML grammar describing the logic required to coordinate web services in a process flow
 - interpreted and executed by an orchestration engine
- Layered on top of WSDL
 - every process is exposed as a web service using WSDL
 - WSDL types are used to describe persistent information
 - WSDL references specify calls to external services
- Orchestration features:
 - executable processes model an executable private workflow
 - abstract processes specify a public message exchange

43

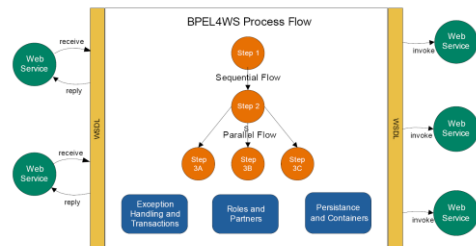
BPEL4WS

- A layer on top of WSDL
- Models the behaviour of web services in a business process interaction
- Control logic required to coordinate web services
- In current implementations interpreted and executed by an orchestration engine (centralized!)
- Support for long transactions (compensations)
- Combine activity diagram and activity hierarchy



44

BPEL4WS Process Flow



45

Summary

- This week:
 - SOA:
 - Concepts
 - Development of SOA
 - Industrial standards
- Next week:
 - **Cloud Computing Basic**

46

References

- Microsoft FoodMovers example
 - <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/FoodMovers1.asp>
- Service-Oriented Architecture Explained
 - http://www.ondotnet.com/pub/a/general/print_code.html
- Web services
 - <http://msdn.microsoft.com/webservices>
- An introduction to SOA
 - <http://www.whatissoa.com/>
- Web Service and SOA
 - <http://www.service-architecture.com/>

47