Service-centric Software Engineering

Objectives

- To explain the notion of a reusable service, based on web service standards, that provides a mechanism for inter-organisational computing;
- To describe the service engineering process that is intended to produce reusable web services;
- To introduce service composition as a means of application development;
- To show how business process models may be used as a basis for the design of service-oriented systems.

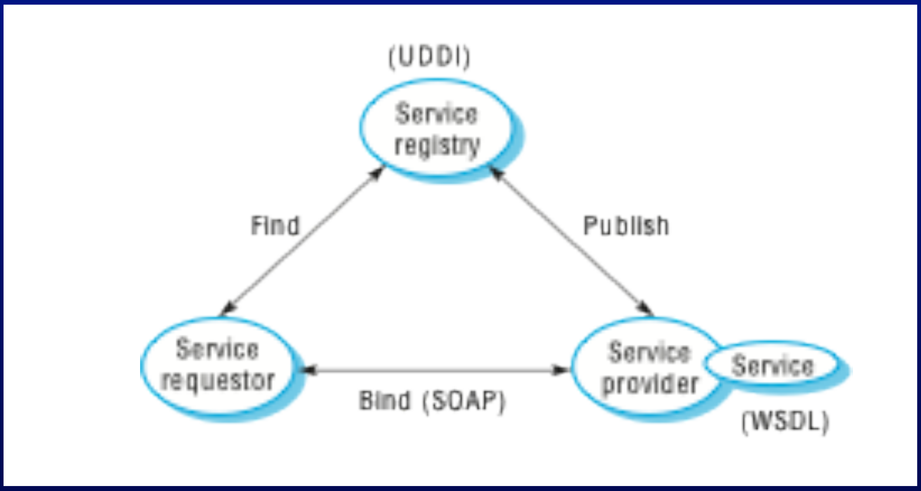
Topics covered

- Services as reusable components
- Service engineering
- Software development with services

Service-oriented architectures

- A means of developing distributed systems where the components are stand-alone services
- Services may execute on different computers from different service providers
- Standard protocols have been developed to support service communication and information exchange

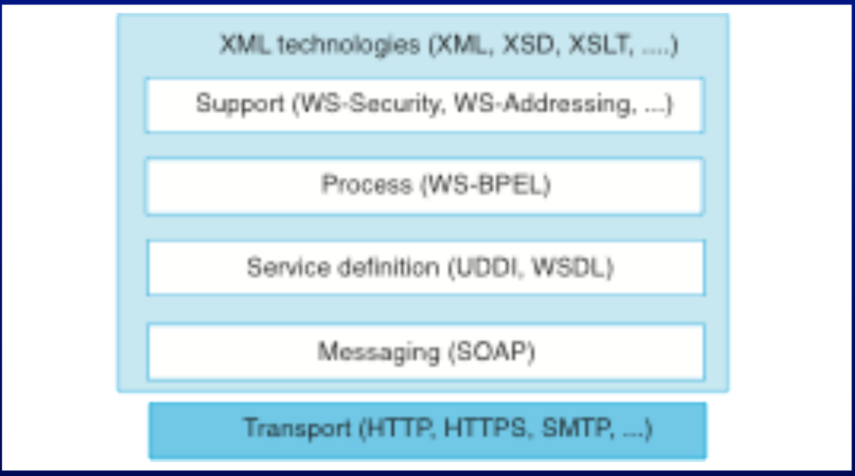
Service-oriented architectures



Benefits of SOA

- Services can be provided locally or outsourced to external providers
- Services are language-independent
- Investment in legacy systems can be preserved
- Inter-organisational computing is facilitated through simplified information exchange

Web service standards


XML technologies (XML, XSD, XSLT, ...)
Support (WS-Security, WS-Addressing, ...)
Process (WS-BPEL)
Service definition (UDDI, WSDL)
Messaging (SOAP)
Transport (HTTP, HTTPS, SMTP, ...)

Key standards

- SOAP
  - A message exchange standard that supports service communication
- WSDL (Web Service Definition Language)
  - This standard allows a service interface and its bindings to be defined
- UDDI
  - Defines the components of a service specification that may be used to discover the existence of a service
- WS-BPEL
  - A standard for workflow languages used to define service composition

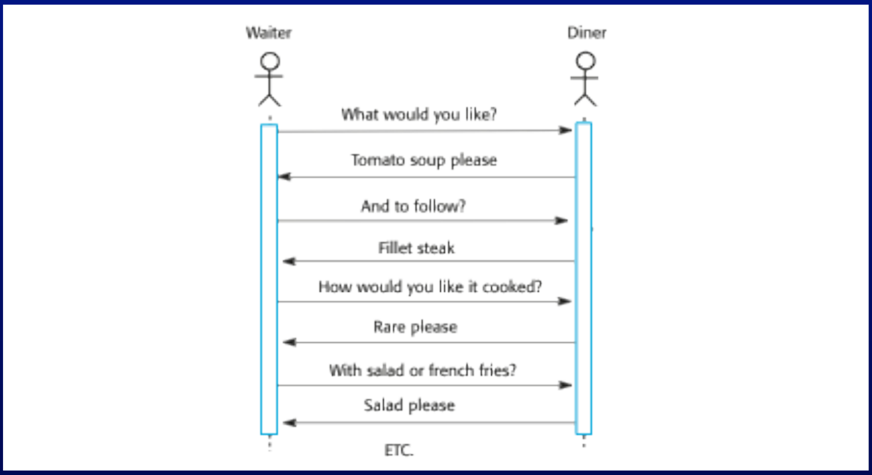
Service-oriented software engineering

- Existing approaches to software engineering have to evolve to reflect the service-oriented approach to software development
  - Service engineering. The development of dependable, reusable services
    - Software development for reuse
  - Software development with services. The development of dependable software where services are the fundamental components
    - Software development with reuse

Services as reusable components

- A service can be defined as:
  - *A loosely-coupled, reusable software component that encapsulates discrete functionality which may be distributed and programmatically accessed. A web service is a service that is accessed using standard Internet and XML-based protocols*
- A critical distinction between a service and a component as defined in CBSE is that services are independent
  - Services do not have a 'requires' interface
  - Services rely on message-based communication with messages expressed in XML

Synchronous interaction



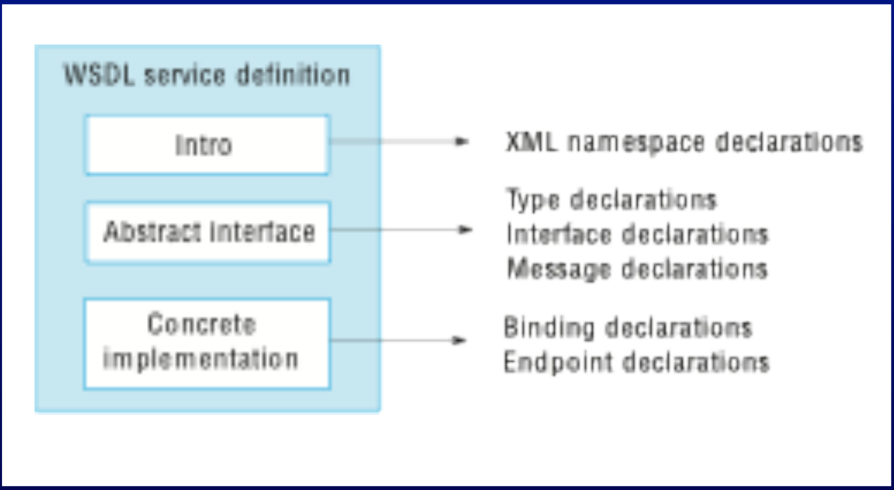
An order as an XML message

```
<starter>
 <dish name = “soup” type = “tomato” />
 <dish name = “soup” type = “fish” />
 <dish name = “pigeon salad” />
</starter>
<main course>
 <dish name = “steak” type = “sirloin”
cooking = “medium” />
 <dish name = “steak” type = “fillet”
cooking = “rare” />
 <dish name = “sea bass”>
</main>
<accompaniment>
 <dish name = “french fries” portions =
“2” />
 <dish name = “salad” portions = “1” />
</accompaniment>
```

Web service description language

- The service interface is defined in a service description expressed in WSDL. The WSDL specification defines
  - What operations the service supports and the format of the messages that are sent and received by the service
  - How the service is accessed - that is, the binding maps the abstract interface ontoa concrete set of protocols
  - Where the service is located. This is usually expressed as a URI (Universal Resource Identifier)

Structure of a WSDL specification



A WSDL description fragment

Define some of the types used. Assume that the namespace prefixes 'ws' refers to the namespace URI for XML schemas and the namespace prefix associated with this definition is weathns.

```
<types>
 <xs: schema targetNameSpace = “http://.../weathns”
   xmlns: weathns = “http://…/weathns” >
  <xs:element name = “PlaceAndDate” type = “pdrec” />
   <xs:element name = “MaxMinTemp” type = “mmtrec” />
   <xs: element name = “InDataFault” type = “errmess” />

   <xs: complexType name = “pdrec”
      <xs: sequence>
         <xs:element name = “town” type = “xs:string”/>
         <xs:element name = “country” type = “xs:string”/>
        <xs:element name = “day” type = “xs:date” />
     </xs:complexType>

  Definitions of MaxMinType and InDataFault here
  </schema>
</types>
```

A WSDL description fragment 2

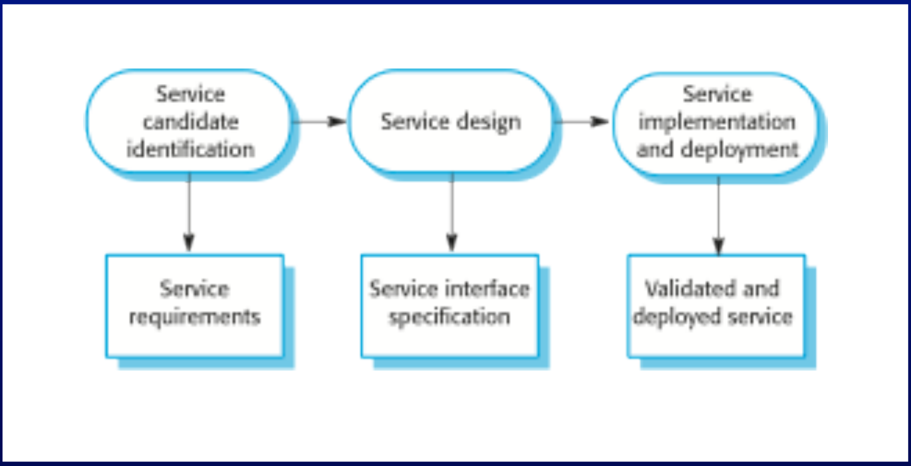
Now define the interface and its operations. In this case, there is only a single operation to return maximum and minimum temperatures

```
<interface name = “weatherInfo” >
 <operation name = “getMaxMinTemps” pattern = “wsdlns: in-out”>
 <input messageLabel = “In” element = “weathns: PlaceAndDate” />
  <output messageLabel = “Out” element = “weathns:MaxMinTemp” />
 <outfault messageLabel = “Out” element = “weathns:InDataFault” />
</operation>
</interface>
```

Service engineering

- The process of developing services for reuse in service-oriented applications
- The service has to be designed as a reusable abstraction that can be used in different systems
- Involves
  - Service candidate identification
  - Service design
  - Service implementation

The service engineering process


Service candidate identification
Service design
Service implementation and deployment
Service requirements
Service interface specification
Validated and deployed service

Service candidate identification

- Three fundamental types of service
  - Utility services that implement general functionality used by different business processes
  - Business services that are associated with a specific business function e.g., in a university, student registration
  - Coordination services that support composite processes such as ordering

Service classification

	Utility	**Business**	**Coordination**
Task	Currency convertor Employee locator	Validate claim form Check credit rating	Process expense claim Pay external supplier
Entity	Document style checker Web form to XML converter	Expenses form Student application form	

Service identification

- Is the service associated with a single logical entity used in different business processes?
- Is the task one that is carried out by different people in the organisation?
- Is the service independent?
- Does the service have to maintain state? Is a database required?
- Could the service be used by clients outside the organisation?
- Are different users of the service likely to have different non-functional requirements?

Catalogue services

- Created by a supplier to show which good can be ordered from them by other companies
- Service requirements
    - Specific version of catalogue should be created for each client
    - Catalogue shall be downloadable
    - The specification and prices of up to 6 items may be compared
    - Browsing and searching facilities shall be provided
    - A function shall be provided that allows the delivery date for ordered items to be predicted
    - Virtual orders shall be supported which reserve the goods for 48 hours to allow a company order to be placed

Catalogue - Non-functional requirements

- Access shall be restricted to employees of accredited organisations
- Prices and configurations offered to each organisation shall be confidential
- The catalogue shall be available from 0700 to 1100
- The catalogue shall be able to process up to 10 requests per second

Catalogue service operations

Operation	Description
MakeCatalogue	Creates a version of the catalogue tailored for a specific customer. Includes an o ptional parameter to create a downloadable PDF version of the catalogue.
Compare	Provides a comparison of up to 6 characteristics (e.g. price, dimensions, processor speed, etc.) of up to 4 catalogue items for comparison.
Lookup	Displays all of the data associated with a specified catalogue item.
Search	This operation takes a logical expression and searches the catalogue according to that expression. It displays a list of all items that match the search expression.
CheckDelivery	Returns the predicted delivery date for an item if it is ordered today.
MakeVirtualOrder	Reserves the number of items to be ordered by a customer and provides item information for the customer's own procurement system.

Service interface design

- Involves thinking about the operations associated with the service and the messages exchanged
- The number of messages exchanged to complete a service request should normally be minimised.
- Service state information may have to be included in messages

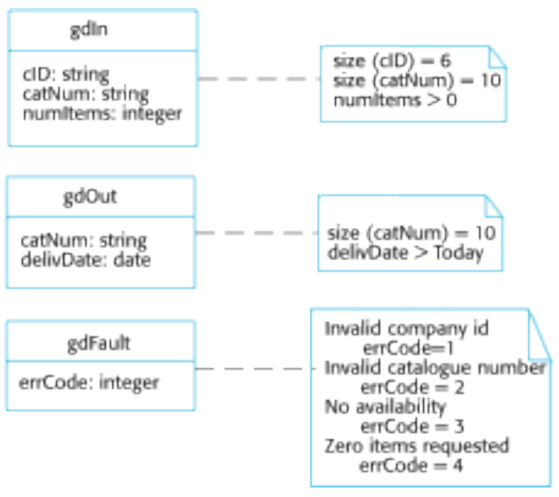
Interface design stages

- Logical interface design
  - Starts with the service requirements and defines the operation names and parameters associated with the service. Exceptions should also be defined
- Message design
  - Design the structure and organisation of the input and output messages. Notations such as the UML are a more abstract representation than XML
- WSDL description
  - The logical specification is converted to a WSDL description

Catalogue interface design

Operation	Inputs	Outputs	Exceptions
MakeCatalogue	*mcI n* Company id PDF-flag	*mcOut* URL of the catalogue for that company	*mcFault* Invalid company id
Compare	*compIn* Company id Entry attribute (up to 6) Catalogue number (up to 4)	*compOut* URL of page showing comparison table	*compFault* Invalid company id Invalid catalogue number Unknown attribute
Lookup	*lookIn* Company id Catalogue number	*lookOut* URL of page with the item information	*lookFault* Invalid company id Invalid catalogue number
Search	*searchIn* Company id Search string	*searchOut* URL of web page with search results	*searchFault* Invalid company id Badly-formed search string
CheckDelivery	*gdIn* Company id Catalogue number Number of items required	*gdOut* Catalogue number Expected delivery date	*gdFault* Invalid company id Invalid catalogue number No availability Zero items requested
PlaceOrder	*poIn* Company id Number of items required Catalogue number	*poOut* Catalogue number Number of items required Predicted delivery date Unit price estimate Total price estimate	*poFault* Invalid company id Invalid catalogue number Zero items requested

Input and output message structure



Service implementation and deployment

- Programming services using a standard programming language or a workflow language
- Services then have to be tested by creating input messages and checking that the output messages produced are as expected
- Deployment involves publicising the service using UDDI and installing it on a web server. Current servers provide support for service installation

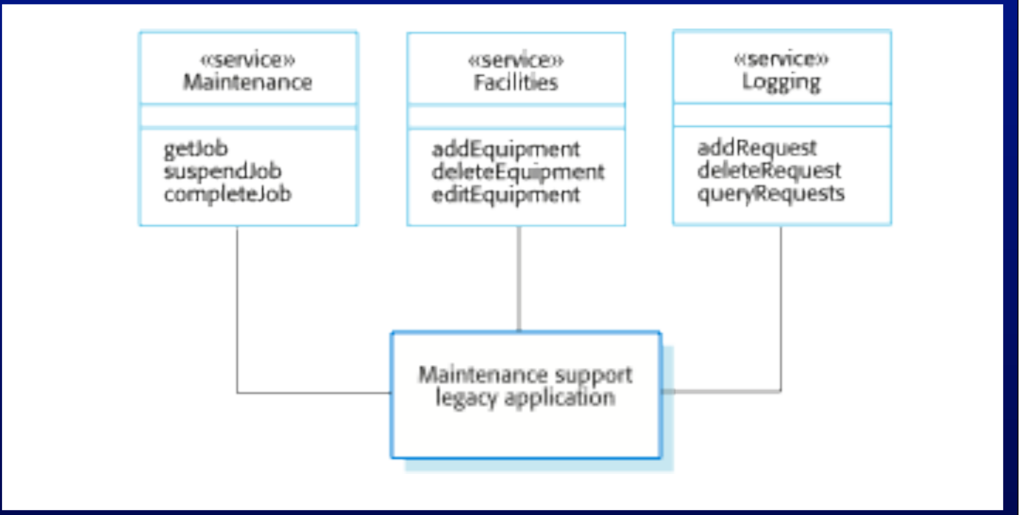
A UDDI description

- Details of the business providing the service
- An informal description of the functionality provided by the service
- Information where to find the service's WSDL specification
- Subscription information that allows users to register for service updates

Legacy system services

- An important application of services is to provide access to functionality embedded in legacy systems
- Legacy systems offer extensive functionality and this can reduce the cost of service implementation
- External applications can access this functionality through the service interfaces

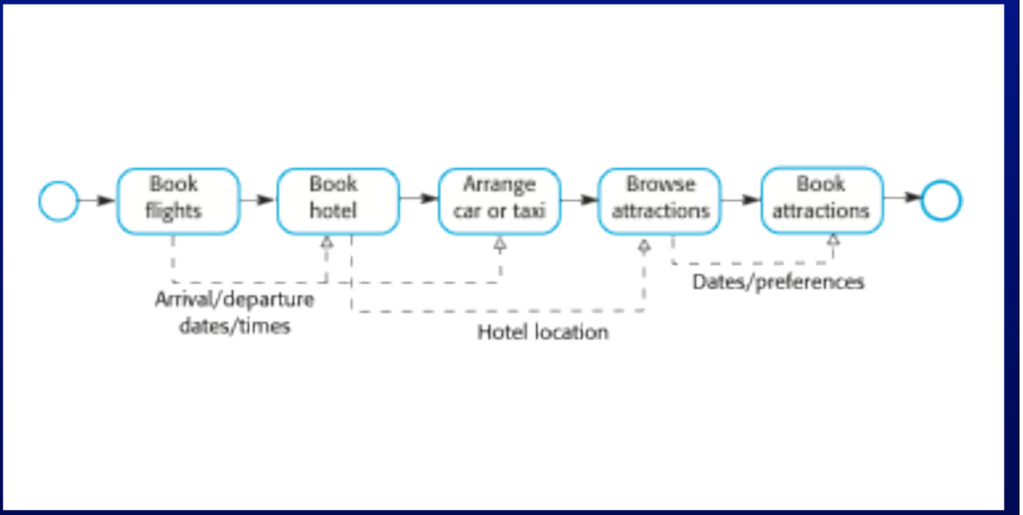
Legacy system access



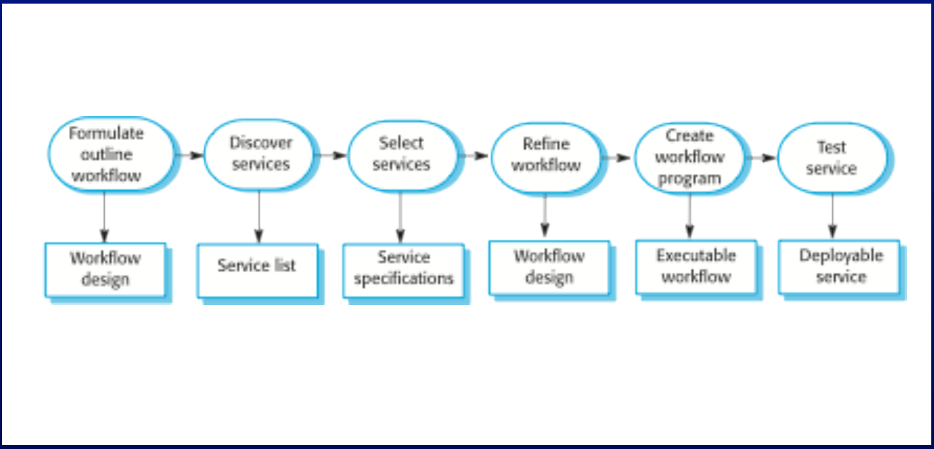
Software development with services

- Existing services are composed and configured to create new composite services and applications
- The basis for service composition is often a workflow
  - Workflows are logical sequences of activities that, together, model a coherent business process
  - For example, provide a travel reservation services which allows flights, car hire and hotel bookings to be coordinated

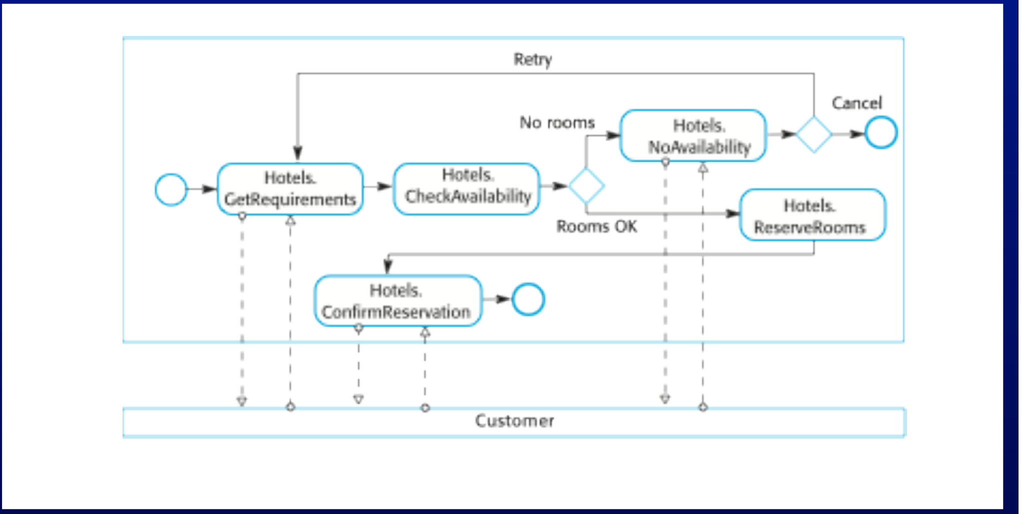
Vacation package workflow



Construction by composition



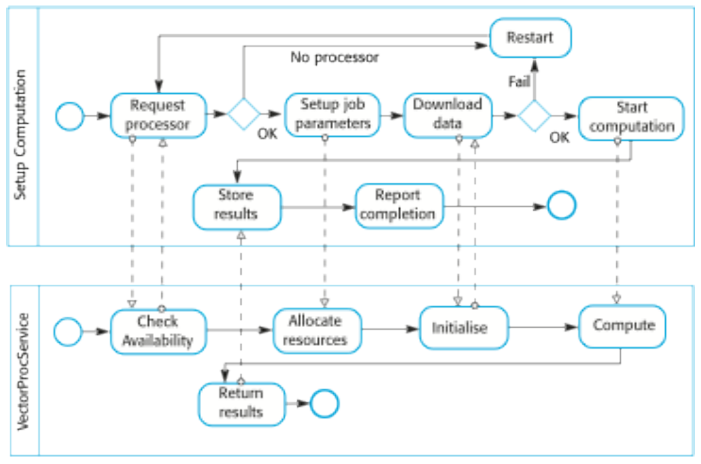
Hotel booking workflow



Workflow design and implementation

- WS-BPEL is an XML-standard for workflow specification. However, WS-BPEL descriptions are long and unreadable
- Graphical workflow notations, such as BPMN, are more readable and WS-BPEL can be generated from them
- In inter-organisational systems, separate workflows are created for each organisation and linked through message exchange

Interacting workflows



Service testing

- Testing is intended to find defects and demonstrate that a system meets its functional and non-functional requirements
- Service testing is difficult as (external) services are 'black-boxes'. Testing techniques that rely on the program source code cannot be used

Service testing problems

- External services may be modified by the service provider thus invalidating tests which have been completed
- Dynamic binding means that the service used in an application may vary - the application tests are not, therefore, reliable
- The non-functional behaviour of the service is unpredictable because it depends on load
- If services have to be paid for as used, testing a service may be expensive
- It may be difficult to invoke compensating actions in external services as these may rely on the failure of other services which cannot be simulated

Key points

- Service-oriented software engineering is based on the notion that programs can be constructed by composing independent services which encapsulate reusable functionality.
- Service interfaces are defined in WSDL. A WSDL specification includes a definition of the interface types and operations, the binding protocol used by the service and the service location.
- Services may be classified as utility services, business services or coordination services.
- The service engineering process involves identifying candidate services for implementation, defining the service interface and implementing, testing and deploying the service.

Key points

- Service interfaces may be defined for legacy software systems which may then be reused in other applications.
- Software development using services involves creating programs by composing and configuring services to create new composite services.
- Business process models define the activities and information exchange in business processes. Activities in the business process may be implemented by services so the business process model represents a service composition.
- Techniques of software testing based on source-code analysis cannot be used in service-oriented systems that rely on externally provided services.