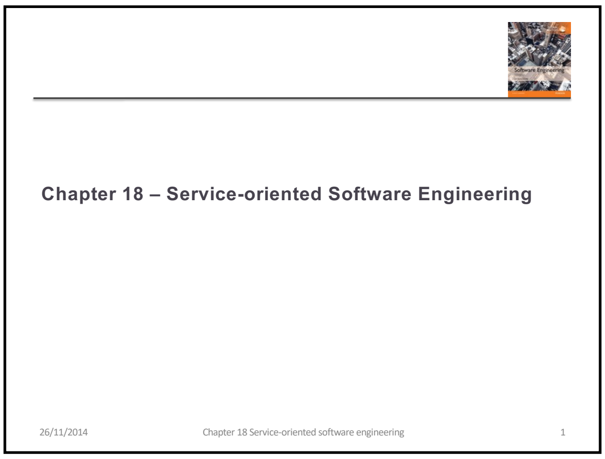
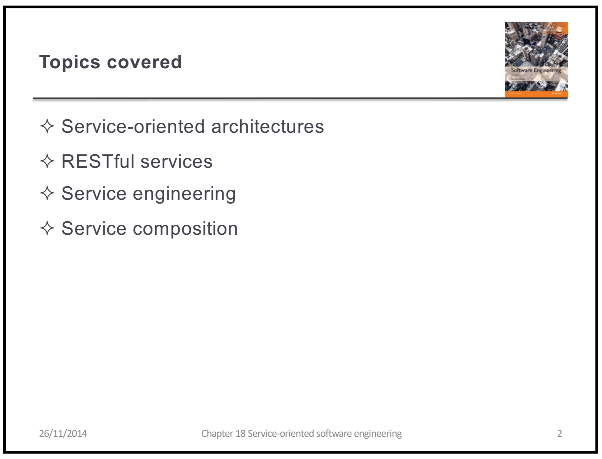


Chapter 18 – Service-oriented Software Engineering

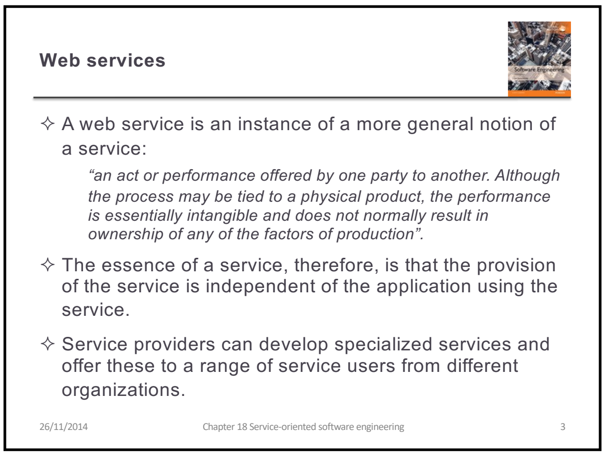
1



Topics covered

✧ Service-oriented architectures
✧ RESTful services
✧ Service engineering
✧ Service composition

2



Web services

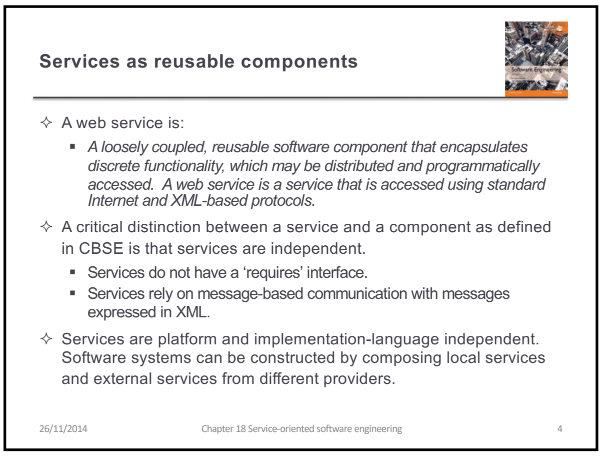
✧ A web service is an instance of a more general notion of a service:

"an act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production".

✧ The essence of a service, therefore, is that the provision of the service is independent of the application using the service.

✧ Service providers can develop specialized services and offer these to a range of service users from different organizations.

3



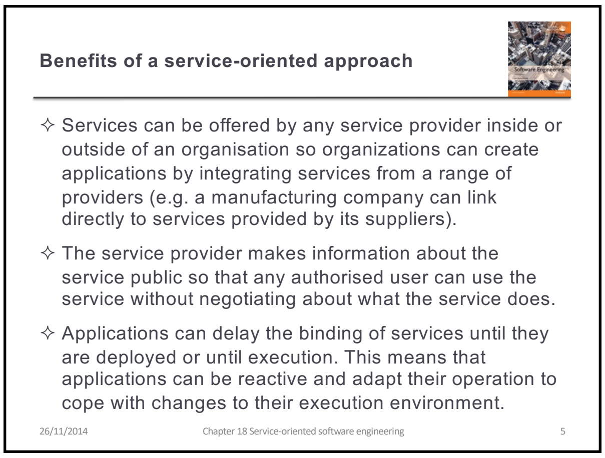
Services as reusable components

✧ A web service is:
- *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.

✧ Services are platform and implementation-language independent. Software systems can be constructed by composing local services and external services from different providers.

4



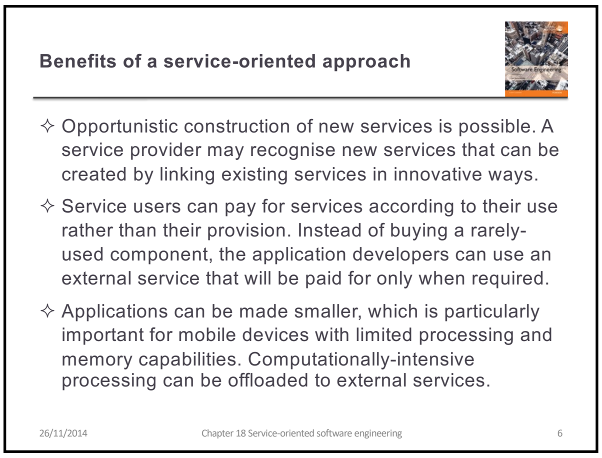
Benefits of a service-oriented approach

✧ Services can be offered by any service provider inside or outside of an organisation so organizations can create applications by integrating services from a range of providers (e.g. a manufacturing company can link directly to services provided by its suppliers).

✧ The service provider makes information about the service public so that any authorised user can use the service without negotiating about what the service does.

✧ Applications can delay the binding of services until they are deployed or until execution. This means that applications can be reactive and adapt their operation to cope with changes to their execution environment.

5



Benefits of a service-oriented approach

✧ Opportunistic construction of new services is possible. A service provider may recognise new services that can be created by linking existing services in innovative ways.

✧ Service users can pay for services according to their use rather than their provision. Instead of buying a rarely-used component, the application developers can use an external service that will be paid for only when required.

✧ Applications can be made smaller, which is particularly important for mobile devices with limited processing and memory capabilities. Computationally-intensive processing can be offloaded to external services.

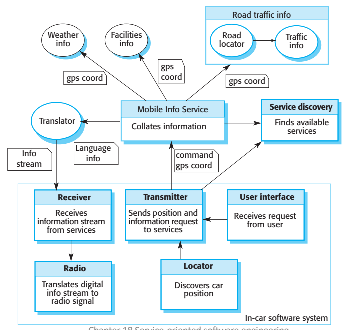
6

Services scenario

- An in-car information system provides drivers with information on weather, road traffic conditions, local information etc. This is linked to car audio system so that information is delivered as a signal on a specific channel.
- The car is equipped with GPS receiver to discover its position and, based on that position, the system accesses a range of information services. Information may be delivered in the driver's specified language.
- The in-car software includes five modules that handle communications involving the driver, GPS receiver and car radio. The car communicates with external mobile information services for a variety of data.

7

A service-based, in-car information system



8

Advantage of SOA for this application

- It is not necessary to decide when the system is programmed or deployed what service provider should be used or what specific services should be accessed.
  - As the car moves around, the in-car software uses the service discovery service to find the most appropriate information service and binds to that.
  - Because of the use of a translation service, it can move across borders and therefore make local information available to people who don't speak the local language.

9

Service-oriented software engineering

- As significant a development as object-oriented development.
- Service-oriented systems are essential to the cloud and mobile systems.
- Building applications based on services allows companies and other organizations to cooperate and make use of each other's business functions.
- Service-based applications may be constructed by linking services from various providers using either a standard programming language or a specialized workflow language.

10

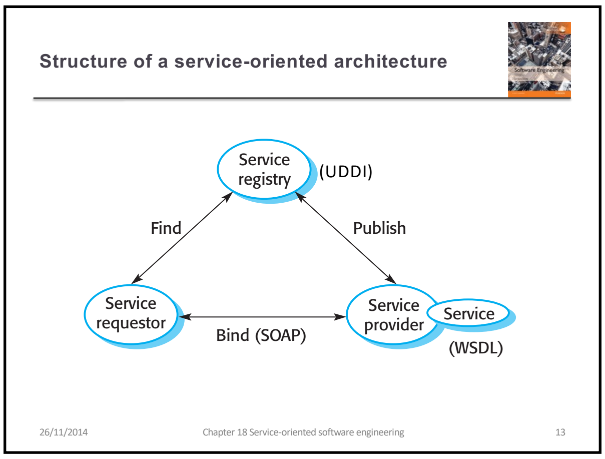
Service-oriented architectures

11

Service-oriented architectures

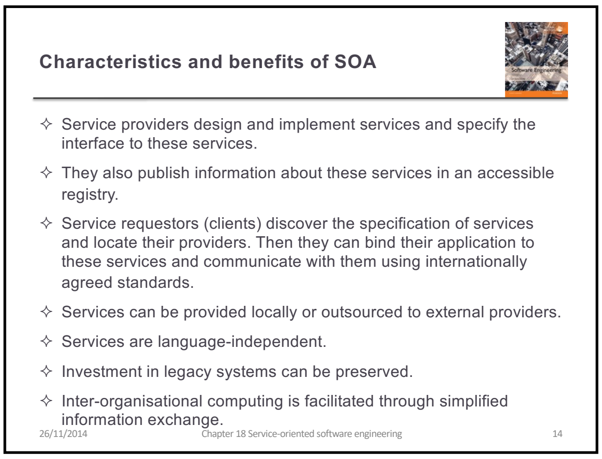
- Service-oriented architectures (SOA) is an architectural style based on the idea that executable services can be included in applications.
- A means of developing distributed systems where the components are stand-alone services.
- Services have well-defined, published interfaces, on which the appropriateness of the service is decided.
- Services may execute on different computers from different service providers.
- Standard protocols have been developed to support service communication and information exchange.

12



Structure of a service-oriented architecture

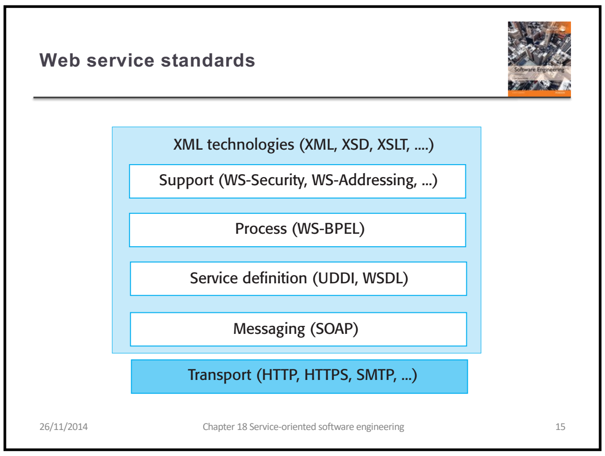
13



Characteristics and benefits of SOA

- ✧ Service providers design and implement services and specify the interface to these services.
- ✧ They also publish information about these services in an accessible registry.
- ✧ Service requestors (clients) discover the specification of services and locate their providers. Then they can bind their application to these services and communicate with them using internationally agreed standards.
- ✧ 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.

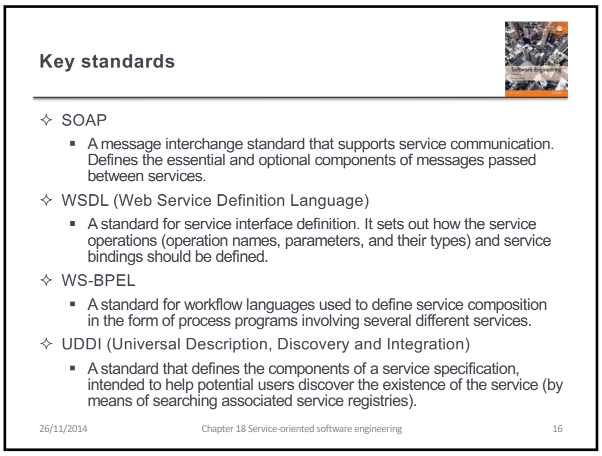
14



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, ...)

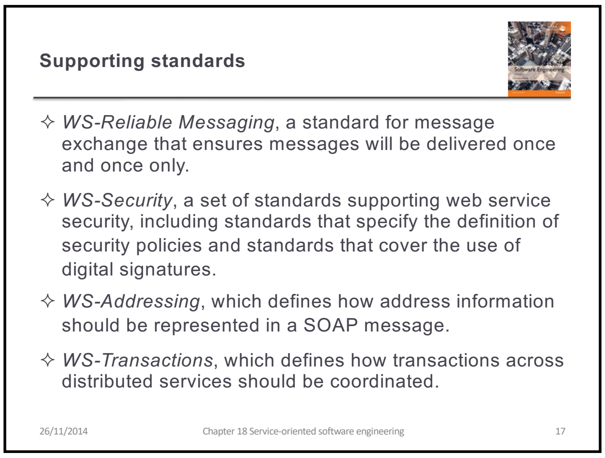
15



Key standards

- ✧ SOAP
  - A message interchange standard that supports service communication. Defines the essential and optional components of messages passed between services.
- ✧ WSDL (Web Service Definition Language)
  - A standard for service interface definition. It sets out how the service operations (operation names, parameters, and their types) and service bindings should be defined.
- ✧ WS-BPEL
  - A standard for workflow languages used to define service composition in the form of process programs involving several different services.
- ✧ UDDI (Universal Description, Discovery and Integration)
  - A standard that defines the components of a service specification, intended to help potential users discover the existence of the service (by means of searching associated service registries).

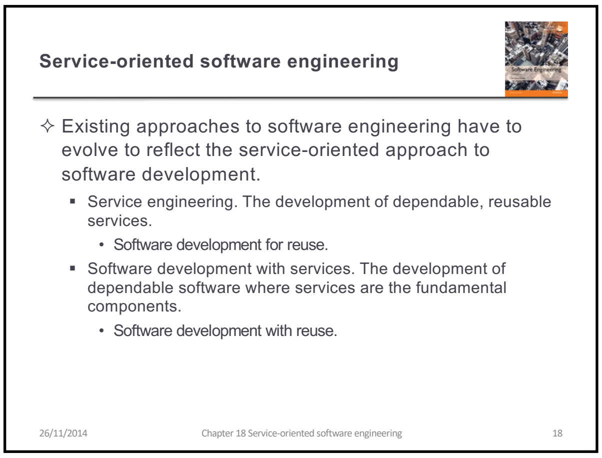
16



Supporting standards

- ✧ *WS-Reliable Messaging*, a standard for message exchange that ensures messages will be delivered once and once only.
- ✧ *WS-Security*, a set of standards supporting web service security, including standards that specify the definition of security policies and standards that cover the use of digital signatures.
- ✧ *WS-Addressing*, which defines how address information should be represented in a SOAP message.
- ✧ *WS-Transactions*, which defines how transactions across distributed services should be coordinated.

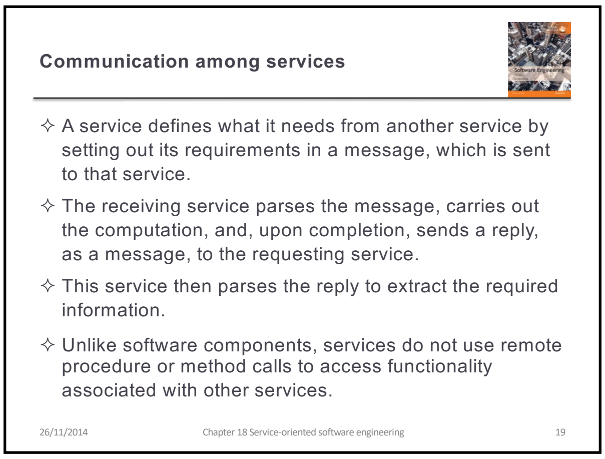
17



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.

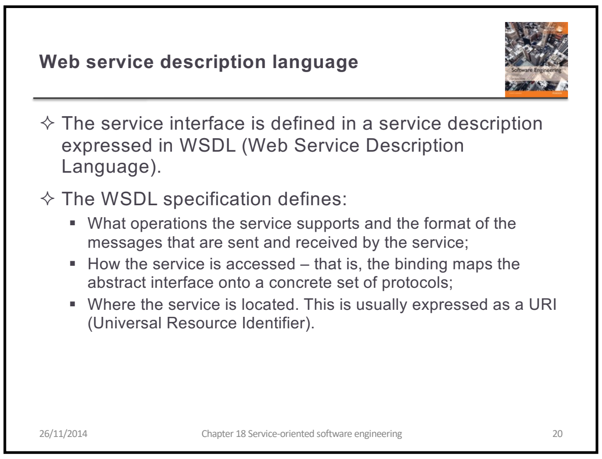
18



Communication among services

- ✧ A service defines what it needs from another service by setting out its requirements in a message, which is sent to that service.
- ✧ The receiving service parses the message, carries out the computation, and, upon completion, sends a reply, as a message, to the requesting service.
- ✧ This service then parses the reply to extract the required information.
- ✧ Unlike software components, services do not use remote procedure or method calls to access functionality associated with other services.

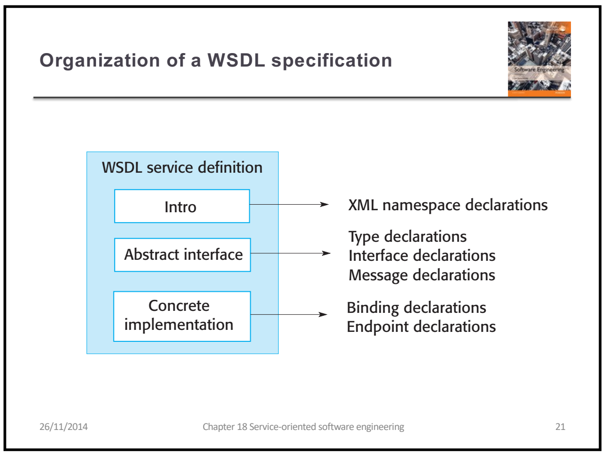
19



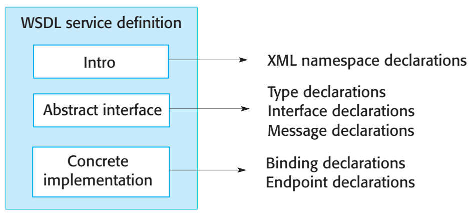
Web service description language

- ✧ The service interface is defined in a service description expressed in WSDL (Web Service Description Language).
- ✧ 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 onto a concrete set of protocols;
  - Where the service is located. This is usually expressed as a URI (Universal Resource Identifier).

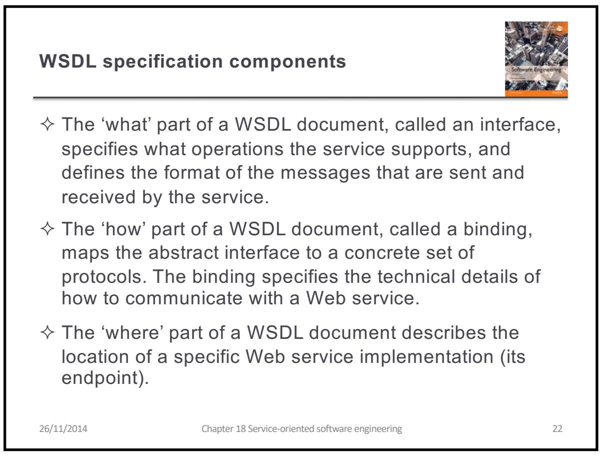
20



Organization of a WSDL specification



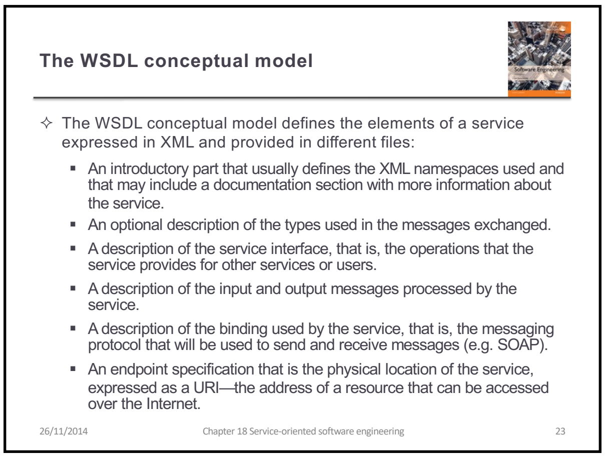
21



WSDL specification components

- ✧ The 'what' part of a WSDL document, called an interface, specifies what operations the service supports, and defines the format of the messages that are sent and received by the service.
- ✧ The 'how' part of a WSDL document, called a binding, maps the abstract interface to a concrete set of protocols. The binding specifies the technical details of how to communicate with a Web service.
- ✧ The 'where' part of a WSDL document describes the location of a specific Web service implementation (its endpoint).

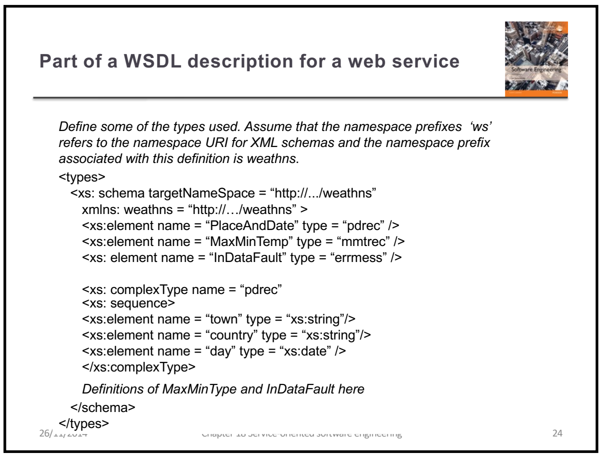
22



The WSDL conceptual model

- ✧ The WSDL conceptual model defines the elements of a service expressed in XML and provided in different files:
  - An introductory part that usually defines the XML namespaces used and that may include a documentation section with more information about the service.
  - An optional description of the types used in the messages exchanged.
  - A description of the service interface, that is, the operations that the service provides for other services or users.
  - A description of the input and output messages processed by the service.
  - A description of the binding used by the service, that is, the messaging protocol that will be used to send and receive messages (e.g. SOAP).
  - An endpoint specification that is the physical location of the service, expressed as a URI—the address of a resource that can be accessed over the Internet.

23



Part of a WSDL description for a web service

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>
```

24

Part of a WSDL description for a web service

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>
```

25

Part of a WSDL description for a web service

✧ The first part shows part of the element and type definition that is used in the service specification. This defines the elements `PlaceAndDate`, `MaxMinTemp`, and `InDataFault`.

✧ The second part of the description shows how the service interface is defined. The service `weatherInfo` has a single operation, although there are no restrictions on the number of operations that may be defined. The `weatherInfo` operation has an associated `in-out` pattern meaning that it takes one input message and generates one output message (more patterns are available such as `in-only`, `out-in`, `out-only`, etc.).

26

RESTful services

27

RESTful web services

✧ Current web services standards have been criticized as 'heavyweight' standards that are over-general and inefficient, as they try to cope with the development of complex services, involving dynamic binding, control for quality and dependability, etc.

✧ REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client.

✧ This style underlies the web as a whole and is simpler than SOAP/WSDL for implementing web services.

✧ RESTful services involve a lower overhead than so-called 'big web services' and are used by many organizations implementing service-based systems.

28

Resources

✧ The fundamental element in a RESTful architecture is a resource.

✧ Essentially, a resource is simply a data element such as a catalog, a medical record or a document such as a book chapter.

✧ In general, resources may have multiple representations i.e. they can exist in different formats.

- MS WORD (for editing)
- PDF (for presentation)
- Quark Xpress (for publishing)

29

Resource operations

✧ Create – bring the resource into existence

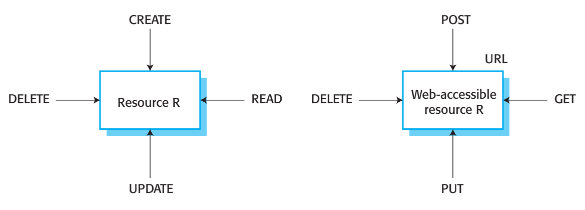
✧ Read – return a representation of the resource

✧ Update – change the value of the resource

✧ Delete – make the resource inaccessible

30

Resources and actions



(a) General resource actions

(b) Web resources

31

Operation functionality

- ✧ POST is used to create a resource. It has associated data that defines the resource.
- ✧ GET is used to read the value of a resource and return that to the requestor in the specified representation, such as XHTML, that can be rendered in a web browser.
- ✧ PUT is used to update the value of a resource.
- ✧ DELETE is used to delete the resource.

32

Resource access

- ✧ When a RESTful approach is used, the data is exposed and is accessed using its URL.
- ✧ Therefore, the weather data for each place in the database, might be accessed using URLs such as:
  - http://weather-info-example.net/temperatures/boston
    http://weather-info-example.net/temperatures/edinburgh
- ✧ Invokes the GET operation and returns a list of maximum and minimum temperatures.
- ✧ To request the temperatures for a specific date, a URL query is used:
  - http://weather-info-example.net/temperatures/edinburgh?date=20140226

33

Query results

- ✧ The response to a GET request in a RESTful service may include URLs.
- ✧ If the response to a request is a set of resources, then the URL of each of these may be included.
  - http://weather-info-example.net/temperatures/edinburgh-scotland
    http://weather-info-example.net/temperatures/edinburgh-australia
    http://weather-info-example.net/temperatures/edinburgh-maryland

34

Difference between RESTful and SOAP-based services

- ✧ RESTful services are not exclusively XML-based. When a resource is requested, created, or changed, the representation may be specified.
- ✧ This is important for RESTful services because representations such as JSON (Javascript Object Notation), as well as XML, may be used.
- ✧ These can be processed more efficiently than XML-based notations.
- ✧ RESTful services have become more widely used over the past few years because of the widespread use of mobile devices that have limited processing capabilities.

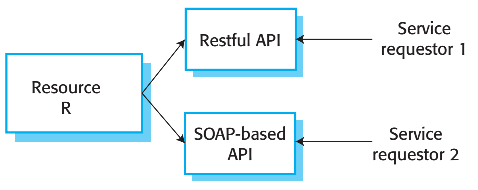
35

Disadvantages of RESTful approach

- ✧ When a service has a complex interface and is not a simple resource, it can be difficult to design a set of RESTful services to represent this.
- ✧ There are no standards for RESTful interface description so service users must rely on informal documentation to understand the interface.
- ✧ When you use RESTful services, you have to implement your own infrastructure for monitoring and managing the quality of service and the service reliability.
- ✧ However, it is often possible to provide both SOAP-based and RESTful interfaces to the same service or resource.

36

RESTful and SOAP-based APIs



37

Service engineering

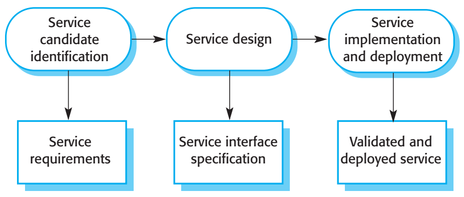
38

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.
- ✧ Generally useful functionality associated with that abstraction must be designed and the service must be robust and reliable.
- ✧ The service must be documented so that it can be discovered and understood by potential users.

39

The service engineering process


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

40

Stages of service engineering

- ✧ *Service candidate identification*, where you identify possible services that might be implemented and define the service requirements.
- ✧ *Service design*, where you design the logical service interface and its implementation interfaces (SOAP and/or RESTful).
- ✧ *Service implementation and deployment*, where you implement and test the service and make it available for use.
- ✧ The starting point for this process will often be an existing service or a component that is to be converted to a service.

41

Service candidate identification

- ✧ Services should support business processes.
- ✧ Service candidate identification involves understanding an organization's business processes to decide which reusable services could support these processes.
- ✧ Three fundamental types of service:
  - *Utility services* that implement general functionality used by different business processes (e.g. currency converter).
  - *Business services* that are associated with a specific business function (e.g. student registration in a university).
  - *Coordination services* that support composite processes (e.g. an ordering service in a company that allows orders to be placed with suppliers, goods accepted, and payments made).

42

Task and entity-oriented services

- *Task-oriented services* are those associated with some activity.
- *Entity-oriented services* are like objects. They are associated with system resource, which is a business entity such as a job application form.
- Utility or business services may be entity- or task-oriented, coordination services are always task-oriented.

43

Service classification

	Utility	Business	Coordination
Task	Currency converter 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	

44

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? Can this fit with a RESTful model?
- Is the service independent? That is, to what extent does it rely on the availability of other services?
- Does the service have to maintain state? Is a database required? Stateless services are easier to reuse.
- Could the service be used by clients outside the organisation?
- Are different users of the service likely to have different non-functional requirements?

45

Service identification example

- A large company, which sells computer equipment, has arranged special prices for approved configurations for some customers.
- To facilitate automated ordering, the company wishes to produce a catalog service that will allow customers to select the equipment that they need.
- Unlike a consumer catalog, orders are not placed directly through a catalog interface. Instead, goods are ordered through the web-based procurement system of each company that accesses the catalog as a web service.
- Most companies have their own budgeting and approval procedures for orders and their own ordering process must be followed when an order is placed.

46

Catalog services

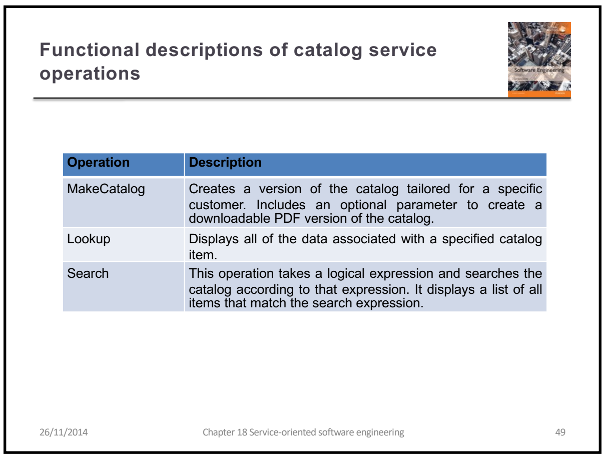
- Created by a supplier to show which good can be ordered from them by other companies.
- Service requirements:
  - Specific version of catalogue should be created for each client.
  - Catalogue shall be downloadable.
  - The specification and prices of up to 6 items may be compared.
  - Browsing and 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.

47

Catalogue: Non-functional requirements

- Access to the catalog service shall be restricted to employees of accredited organizations.
- The prices and configurations offered to each customer shall be confidential, and access to these shall only be provided to employees of that customer.
- The catalog shall be available without disruption of service from 0700 GMT to 1100 GMT.
- The catalog service shall be able to process up to 100 requests per second peak load.

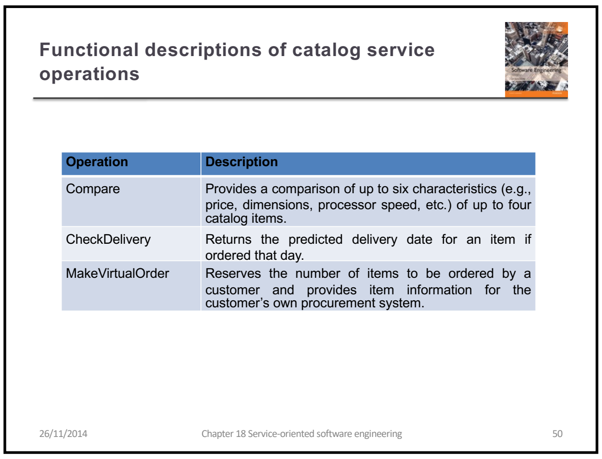
48



Functional descriptions of catalog service operations

Operation	Description
MakeCatalog	Creates a version of the catalog tailored for a specific customer. Includes an optional parameter to create a downloadable PDF version of the catalog.
Lookup	Displays all of the data associated with a specified catalog item.
Search	This operation takes a logical expression and searches the catalog according to that expression. It displays a list of all items that match the search expression.

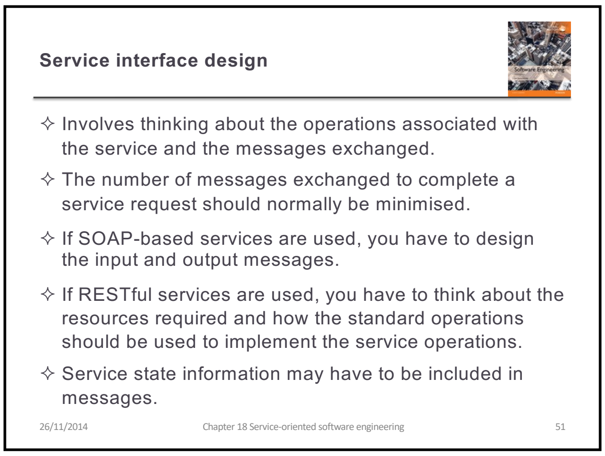
49



Functional descriptions of catalog service operations

Operation	Description
Compare	Provides a comparison of up to six characteristics (e.g., price, dimensions, processor speed, etc.) of up to four catalog items.
CheckDelivery	Returns the predicted delivery date for an item if ordered that day.
MakeVirtualOrder	Reserves the number of items to be ordered by a customer and provides item information for the customer's own procurement system.

50



Service interface design

✧ Involves thinking about the operations associated with the service and the messages exchanged.

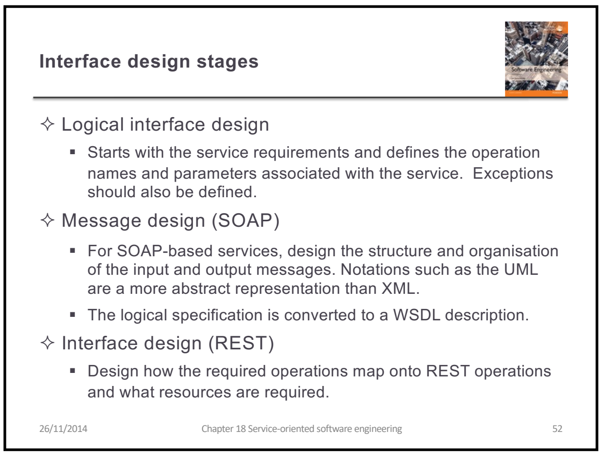
✧ The number of messages exchanged to complete a service request should normally be minimised.

✧ If SOAP-based services are used, you have to design the input and output messages.

✧ If RESTful services are used, you have to think about the resources required and how the standard operations should be used to implement the service operations.

✧ Service state information may have to be included in messages.

51



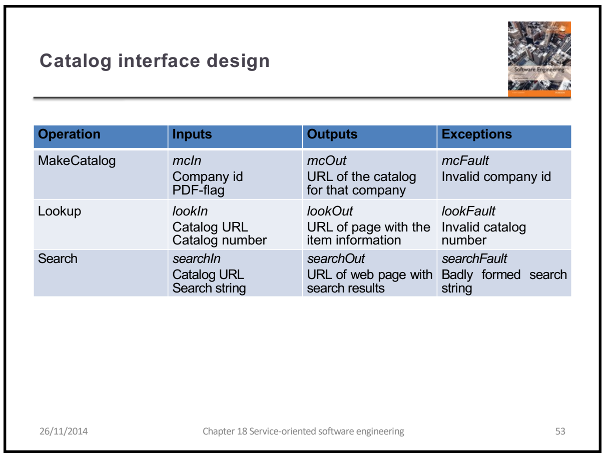
Interface design stages

✧ 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 (SOAP)
- For SOAP-based services, design the structure and organisation of the input and output messages. Notations such as the UML are a more abstract representation than XML.
- The logical specification is converted to a WSDL description.

✧ Interface design (REST)
- Design how the required operations map onto REST operations and what resources are required.

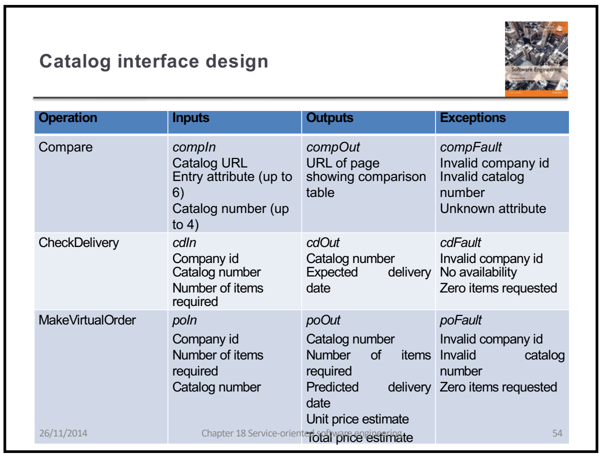
52



Catalog interface design

Operation	Inputs	Outputs	Exceptions
MakeCatalog	*mcIn* Company id PDF-flag	*mcOut* URL of the catalog for that company	*mcFault* Invalid company id
Lookup	*lookIn* Catalog URL Catalog number	*lookOut* URL of page with the item information	*lookFault* Invalid catalog number
Search	*searchIn* Catalog URL Search string	*searchOut* URL of web page with search results	*searchFault* Badly formed search string

53



Catalog interface design

Operation	Inputs	Outputs	Exceptions
Compare	*compIn* Catalog URL Entry attribute (up to 6) Catalog number (up to 4)	*compOut* URL of page showing comparison table	*compFault* Invalid company id Invalid catalog number Unknown attribute
CheckDelivery	*cdIn* Company id Catalog number Number of items required	*cdOut* Catalog number Expected delivery date	*cdFault* Invalid company id No availability Zero items requested
MakeVirtualOrder	*poIn* Company id Number of items required Catalog number	*poOut* Catalog number Number of items required Predicted delivery date Unit price estimate Total price estimate	*poFault* Invalid company id Invalid catalog number Zero items requested

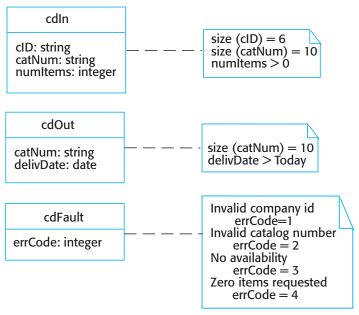
54

Definition of input and output messages

- ✧ In some cases, a textual description of the operations and their inputs and outputs is all that is required. The detailed realization of the service is left as an implementation decision.
- ✧ Sometimes, however, you need to have a more detailed design, and a detailed interface description can be specified in a graphical notation such as the UML or in a readable description format such as JSON.
- ✧ The following figure describes the inputs and outputs for the `checkDelivery` operation and shows how you can use the UML to describe the interface in detail.

55

UML definition of input and output messages



56

RESTful interface

- ✧ There should be a resource representing a company-specific catalog. This should have a URL of the form <base catalog>/<company name> and should be created using a POST operation.
- ✧ Each catalog item should have its own URL of the form:
  - <base catalog>/<company name>/<item identifier>.
- ✧ The GET operation is used to retrieve items.
  - **Lookup** is implemented by using the URL of an item in a catalog as the GET parameter.
  - **Search** is implemented by using GET with the company catalog as the URL and the search string as a query parameter. This GET operation returns a list of URLs of the items matching the search.

57

RESTful interface

- ✧ The **Compare** operation can be implemented as a sequence of GET operations, to retrieve the individual items, followed by a POST operation to create the comparison table and a final GET operation to return this to the user.
- ✧ The **CheckDelivery** and **MakeVirtualOrder** operations require an additional resource, representing a virtual order.
  - A POST operation is used to create this resource with the number of items required. The company id is used to automatically fill in the order form and the delivery date is calculated. This can then be retrieved using a GET operation.

58

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 and installing it on a web server. Current servers provide support for service installation.

59

Legacy system services

- ✧ Services can be implemented by implementing a service interface to existing 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.

60

Service descriptions

✧ Information about your business, contact details, etc. This is important for trust reasons. Users of a service have to be confident that it will not behave maliciously.

✧ An informal description of the functionality provided by the service. This helps potential users to decide if the service is what they want.

✧ A description of how to use the service (informal textual description of the input and output parameters or for more complex SOAP-based services, the WSDL description).

✧ Subscription information that allows users to register for information about updates to the service.

61

Service composition

62

Software development with services

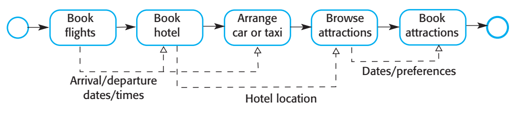
✧ Existing services are composed and configured to create new composite services and applications.

✧ These may be integrated with a user interface implemented in a browser to create a web application, or they may be used as components in some other service composition.

✧ 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 service which allows flights, car hire and hotel bookings to be coordinated.

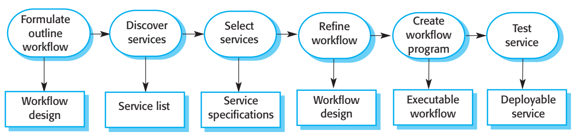
63

Vacation package workflow



64

Service construction by composition



65

Construction by composition

✧ *Formulate outline workflow*
- In this initial stage of service design, you use the requirements for the composite service as a basis for creating an 'ideal' service design.

✧ *Discover services*
- During this stage of the process, you search service registries or catalogs to discover what services exist, who provides these services and the details of the service provision.

✧ *Select possible services*
- Your selection criteria will obviously include the functionality of the services offered. They may also include the cost of the services and the quality of service (responsiveness, availability, etc.) offered.

66

Construction by composition

✧ *Refine workflow*
- This involves adding detail to the abstract description and perhaps adding or removing workflow activities.

✧ *Create workflow program*
- During this stage, the abstract workflow design is transformed to an executable program and the service interface is defined. You can use a conventional programming language, such as Java or a workflow language, such as WS-BPEL.

✧ *Test completed service or application*
- The process of testing the completed, composite service is more complex than component testing in situations where external services are used.

67

Workflow design and implementation

✧ WS-BPEL is an XML-standard for workflow specification. However, WS-BPEL descriptions are long and unreadable.

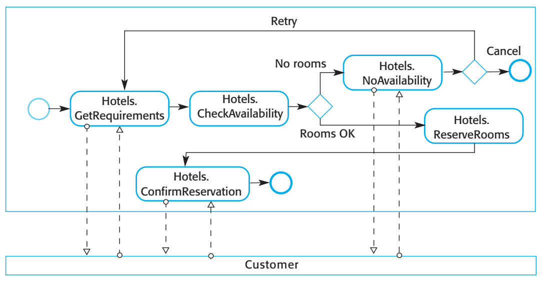
✧ Graphical workflow notations, such as BPMN, are more readable and WS-BPEL can be generated from them.

✧ In inter-organisational systems, separate workflows are created for each organisation and linked through message exchange.

✧ Workflows can be used with both SOAP-based and RESTful services.

68

A fragment of a hotel booking workflow



69

A simplified BPMN workflow model for the hotel booking scenario

✧ The model assumes the existence of a `Hotels` service with associated operations called `GetRequirements`, `CheckAvailability`, `ReserveRooms`, `NoAvailability`, `ConfirmReservation`, and `CancelReservation`.

✧ The process involves getting requirements from the customer, checking room availability, and then, if rooms are available, making a booking for the required dates.

✧ This model introduces some of the core concepts of BPMN that are used to create workflow models.

70

A simplified BPMN workflow model for the hotel booking scenario

✧ Rectangles with rounded corners represent activities. An activity can be executed by a human or by an automated service.

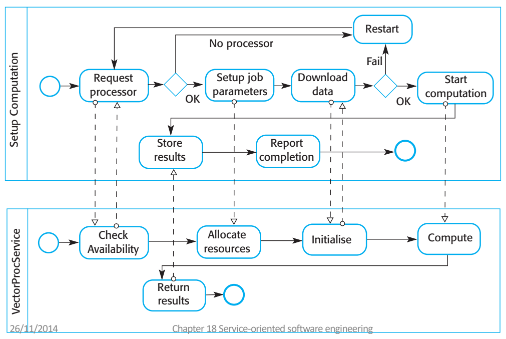
✧ Circles represent discrete events. An event is something that happens during a business process. A simple circle is used to represent a starting event and a darker circle to represent an end event.

✧ A diamond is used to represent a gateway. A gateway is a stage in the process where some choice is made.

✧ A solid arrow shows the sequence of activities; a dashed arrow represents message flow between activities.

71

Interacting or multiple workflows (involving more than one organization)



72

Interacting or multiple workflows (involving more than one organization)

- ✧ This example is drawn from high-performance computing, where hardware is offered as a service.
- ✧ Services are created to provide access to high-performance computers to a geographically distributed user community.
  - ▪ In this example, a vector-processing computer (a machine that can carry out parallel computations on arrays of values) is offered as a service (`VectorProcService`) by a research laboratory. This is accessed through another service called `SetupComputation`.
- ✧ This process is represented in BPMN by developing separate workflows for each of the organizations involved with interactions between them.

73

Testing service compositions

- ✧ 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.

74

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.

75

Key points

- ✧ Service-oriented architecture is an approach to software engineering where reusable, standardized services are the basic building blocks for application systems.
- ✧ Services may be implemented within a service-oriented architecture using a set of XML-based web service standards. These include standards for service communication, interface definition and service enactment in workflows.
- ✧ Alternatively, a RESTful architecture may be used which is based on resources and standard operations on these resources.
- ✧ A RESTful approach uses the http and https protocols for service communication and maps operations on the standard http verbs POST, GET, PUT and DELETE.

76

Key points

- ✧ Utility services provide general-purpose functionality; business services implement part of a business process; coordination services coordinate service execution.
- ✧ Service engineering involves identifying candidate services for implementation, defining service interfaces and implementing, testing and deploying services.
- ✧ The development of software using services involves composing and configuring services to create new composite services and systems.
- ✧ Graphical workflow languages, such as BPMN, may be used to describe a business process and the services used in that process.

77