SWINBURNE UNIVERSITY OF TECHNOLOGY

SWE30003
Software Architectures and Design

Lecture 10
Service Oriented Architectures (SOA) and Web Services

1

Logistical matters

- Weekly submissions – A & Q
  - Week 2: 214 and 202 out of 270;
  - Week 3: 222 and 215 out of 270;
  - Week 4: 202 and 199 out of 266;
  - Week 5: 217 and 210 out of 265;
  - Week 6: 213 and 211 out of 265;
  - Week 7: 194 and 186 out of 265;
  - Week 8: 196 and 190 out of 263;
  - Week 9: 192 and 191 out of 263;
  - Week 10: ..... ;
  - **No late submission, hurdle requirement**
- Assignment 3: released … Questions?

2

2

Question to Answer – Week 9

Discuss the differences, advantages, and disadvantages of data-centred and data-flow architectures. Give examples where either of these two styles would be applicable.

3

3

Required Readings - Week 10

Service Oriented Architectures (SOAP, REST, and Micro-Services)

- Michael N. Huhns, Munindar P. Singh: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Comput.9(1): 75-81 (2005) (Link to an external site)
- Stefan Tilkov: A Brief Introduction to REST. InfoQ (2007) (Link to an external site)
- Chris Richardson: Microservices: Decomposing Applications for Deployability and Scalability. InfoQ (2014) (Link to an external site)
- P Jamshidi, C Pahl, NC Mendonça, J Lewis, S Tilkov: Microservices: The journey so far and challenges ahead. IEEE Software (2018) (Link to an external site)

4

4

Additional Resources/Readings

- Follow the links provided …
- Michael Papazoglou: *Web Services: Principles and Technology*, Pearson/Prentice Hall, 2008. Chapters 1 & 2 (and more)
- Leonard Richardson and Sam Ruby: *RESTful Web Services,* O'Reilly, 2007. Chapter 1 (and more)
- Webber, Parastatidis, Robinson: *REST in Practice*, O'Reilly, 2010 (Free download http://it-ebooks.info/book/393/)
- InfoQ (http://www.infoq.com): SOA, Web services, REST, Microservices, Enterprise Architecture, …

5

5

Content

- **Software services & web services**
- Some concepts you need to understand
- Service oriented architecture (SOA)
- Web application architectural styles and middleware
- WS* (SOAP) approach
- REST approach
- Microservices

SWINBURNE UNIVERSITY OF TECHNOLOGY

6

6

The Web

- Initially, **documents** – for sharing between humans: documents on the web
  - Web Server: hosting documents
  - Web Client (Brower): reading/using documents
- Distributed
- Searchable/addressable
- Extensible/unbounded
- …

SWINBURNE UNIVERSITY OF TECHNOLOGY

7

7

Enterprise (Software) Systems

- Distributed
- Integration (vs greenfield decomposition)
- Business process
- Cross-organisation, cross-geography
- Large-scale
- Addressable
- Then, **software** – for sharing between machines (& between humans): software on the web
  - How? … make software a "**service**" deployable on the web

SWINBURNE UNIVERSITY OF TECHNOLOGY

8

8

Service / Software Service / Web Service

A (software) service – beyond component:

- Independently deployable,
- Self-contained,
- Self-describing,
- Discoverable (open),
- Composable,
- Software system (component).

Not necessarily on the Web …

web service = service + web: a service deployed on the web using web technology.

9

9

Web Services in the Enterprise

Web Services: W3C Definition

- A Web service is a software system *identified* by a URL, whose public interfaces and bindings are *defined* and *described* ~~using XML.~~
- Its definition can be *discovered* by other software systems ?? Maybe if WADL is used
- These systems may then interact with the Web service using ~~XML based messages~~ conveyed by Internet protocols (~~not necessarily Web protocols!~~)

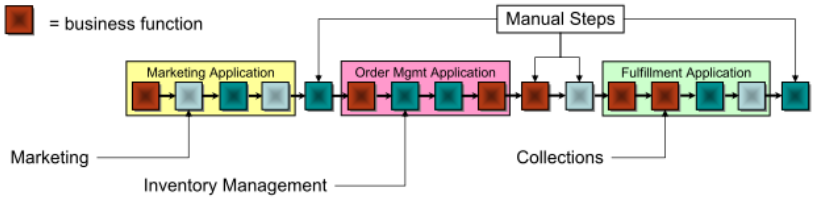
W3C definition does not apply to RESTful 'Web services'!

10

10

Web Services in the Enterprise – Example (business/sales process management)

Need to make an existing business process more flexible



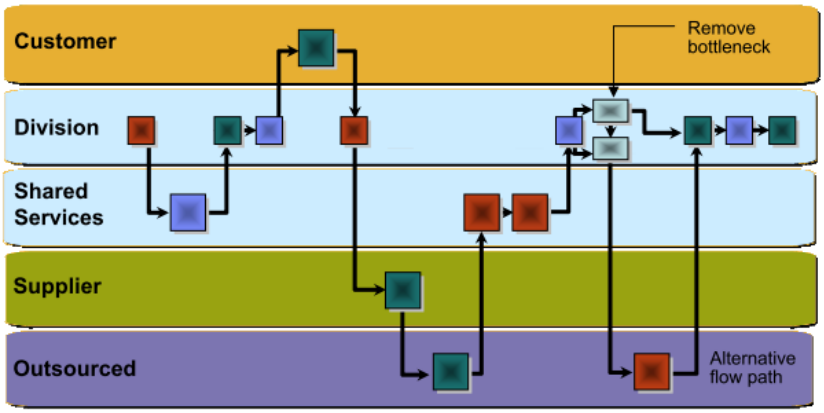
- Business process is embedded in three or four separate applications
- Business functions are tightly coupled within applications
- Business functions have unique and proprietary interfaces, restricting re-use
- Manual steps introduce functional gaps in the process
- Process cannot be easily measured or managed
- Changes to the process are difficult to implement

Example from IBM Rational

11

11

On demand flexibility: Improve the process



- Identify and remove bottlenecks in the process
- Customize the business rules and policies to better serve customers
- A more efficient business process costs less

12

12

Web Services in the Enterprise

Advantages of Web Services for Business

- Within an enterprise (Enterprise Application Integration)
  - Accelerate and reduce the cost of integration
  - Save on infrastructure deployment & management costs
  - Reduce skill requirements
  - Improve reuse
  - Improve flexibility leads quicker response to market opportunities
- Between enterprises (e-Business integration)
  - Providing service to a company's customers
  - e.g., an Insurance company wishes to link its systems to the systems of a new institutional customer (IAG for RACV, NRMA etc)
  - Accessing services from a company's partners and suppliers
  - e.g., dynamically link to new partners and suppliers to offer their services to complement the value the company provides (Master Card from Coles)
- *Standards and common infrastructure reduce the barriers*
- *Simplicity and reuse accelerates deployment*
- *Dynamics opens new business opportunities*

SWINBURNE UNIVERSITY OF TECHNOLOGY

13

13

Content

- Software services / web services
- **Some concepts you need to understand**
- Service oriented architecture (SOA)
- Web application architectural styles and middleware
- WS* (SOAP) approach
- REST approach
- Microservices

SWINBURNE UNIVERSITY OF TECHNOLOGY

14

14

Some underlying concepts …

- Functional and non-functional properties
- Stateless and stateful services
- Granularity
- Synchronicity
- Separation of interfaces and implementation
- Loose coupling

SWINBURNE UNIVERSITY OF TECHNOLOGY

15

15

Some concepts …

Service Properties & State

- Functional & non-functional properties:
  - The *functional* service description details the operational characteristics that define the overall behavior of the service,
  - The *non-functional* description targets service quality attributes, e.g., service metering and cost, performance metrics (response time or accuracy), security, authorization, authentication, scalability, & availability, etc.
- Stateless or stateful services:
  - Services that can be invoked repeatedly without having to maintain context or state are called *stateless*.
    - Simple informational services are stateless.
  - Services that require their context to be preserved from one invocation to the next are called *stateful*.
    - Complex services (business processes) typically involve stateful interactions.

SWINBURNE UNIVERSITY OF TECHNOLOGY

16

16

Some concepts …

Granularity

- Service granularity:
  - Simple services are discrete in nature, exhibit normally a request/reply mode of operation & are of fine granularity, i.e., they are atomic in nature.
  - Complex services are coarse-grained, e.g., a PurchaseOrder. These involve interactions with other services and possibly end-users in a single or multiple sessions.
  - Coarse-grained communication implies larger and richer data structures, (viz. those supported by XML).

SWINBURNE UNIVERSITY OF TECHNOLOGY

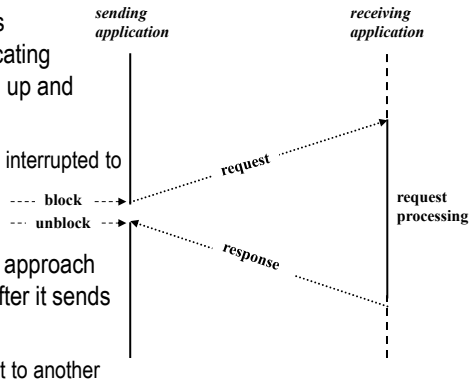
17

17

Some concepts …

Synchronous & asynchronous communication

- Synchronous communication which is synchronized between two communicating application systems, which must both up and running.
  - Execution flow at the client's side is interrupted to execute the call.
- Asynchronous communication where the caller employs a send and forget approach that allows it to continue to execute after it sends the message.
  - Here an application sends a request to another while it continues its own processing activities.



SWINBURNE UNIVERSITY OF TECHNOLOGY

18

18

Some concepts …

Service Interface (API) & Implementation

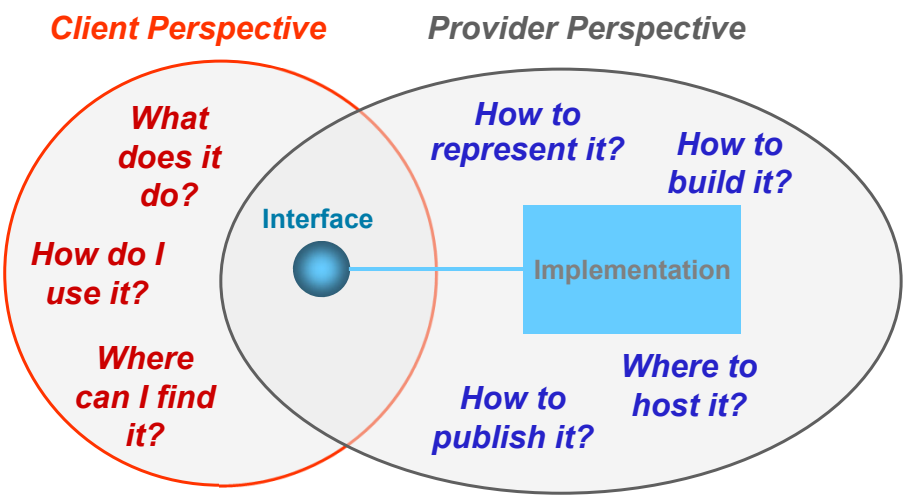
- The *service interface* defines service functionality visible to the external world and provides the means to access this functionality.
  - The service describes its own interface characteristics, i.e., the operations available, the parameters, data-typing and the access protocols, in a way that other software modules can determine what it does, how to invoke its functionality, & what result to expect in return.
- The *service implementation* realizes a specific service interface whose implementation details are hidden from its users.
  - Different service providers using any programming language of their choice may implement the same interface.

SWINBURNE UNIVERSITY OF TECHNOLOGY

19

19

Perspectives on Web Services



SWINBURNE UNIVERSITY OF TECHNOLOGY

20

20

Some concepts …

Loose Coupling

Coupling indicates the degree of dependency any two systems have on each other. Some types relevant to enterprise computing include:

Type of Coupling	Problem Description	Reduced by
Functional	One module relies on the internal workings of others	Well defined interface/API describing inputs and outputs
Interface	Interfaces reflect the implementation details	Avoid detailed mandatory parameters; Use more general methods and message passing (do not get too fine-grained)
Data Structure	Data passed between modules reflects internal representations, or is over-constrained	Explicitly define formats, e.g XML in interface; Allow multiple representations e.g.mime-types; 'Vertical' standards that define semantics of messages
Temporal	Client blocked while waiting response Process over-constrained	Asynchronous messages; Explicitly define behaviour in interface e.g. abstract BPEL; Interaction via events / event-driven processes
Address/URI	Addresses change Providers change URI patterns change	Do not hard code references; Virtualise services; Use middleware directory services

21

Content

- Software services / web services
- Some concepts you need to understand
- **Service oriented architecture (SOA)**
- Web Application architectural styles and middleware
- WS* (SOAP) approach
- REST approach

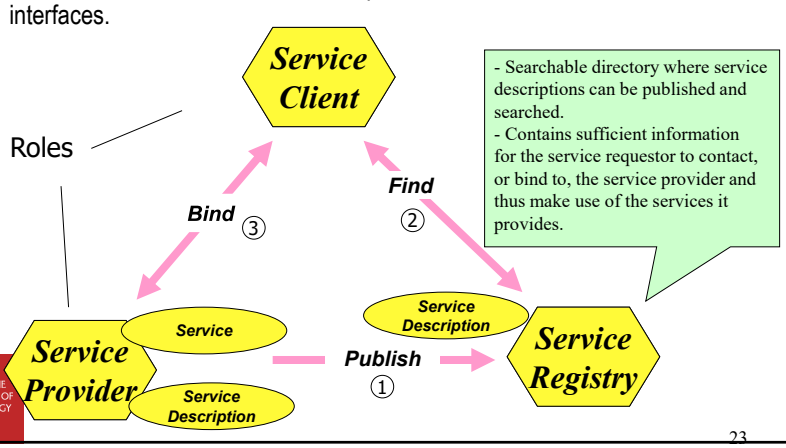
22

22

Web Architectural Styles – SOA

Service Oriented Architecture

- SOA is fundamentally of client-server style, and based on
  - the interaction between decoupled services and/or end-user applications
  - associated with messages & governed by policies.
  - services distributed in a network, via published and discoverable interfaces.



SWINBURNE UNIVERSITY OF TECHNOLOGY

23

23

Web Architectural Styles

Web Architectural Styles

Fundamentally, client-server…. Sub-styles:

- Method-oriented RPC (Remote Procedure Call)
  - Interface: Method, parameters
  - Interaction: synchronous request-response, request-acknowledge
- Message-oriented (aka Document style)
  - Interface: Domain specific operation, Message type
  - Interaction: Various synch/asynch, request-response, one way, solicit-notify
- Resource-oriented
  - Interface: generic CRUD, Resource id, Media type
  - Interaction: synchronous request-response, request-acknowledge
- Event-oriented
  - Interface: event type
  - Interaction: Subscribe-notify, publish-subscribe-notify

SWINBURNE UNIVERSITY OF TECHNOLOGY

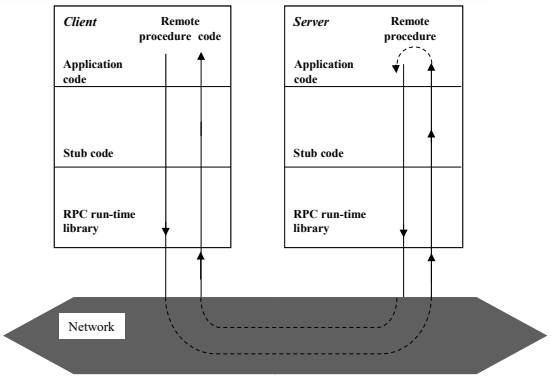
24

24

Web Architectural Styles - RPC

Remote Procedure Calls - an oo/component approach

- Object/component-oriented approach to distributed computing e.g. CORBA
- Application elements use a request/wait-for-reply (**synchronous**) model of communication.
- RPC-style programming leads to *tight coupling* of interfaces and applications.



- In an RPC environment each application needs to know the intimate details of the *interface* of every other application – the number of methods it exposes & the details of each method signature it exposes.

SWINBURNE UNIVERSITY OF TECHNOLOGY

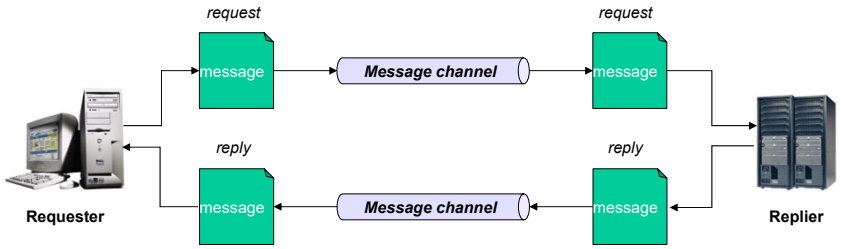
25

25

Web Architectural Styles - Messaging

Asynchronous request/reply messaging

- Most asynchronous messaging mechanisms follow the "fire-and-forget" messaging principle where the sending application can conduct its work as usual once a message was asynchronously sent.
  - The sending application assumes that the message will arrive safely at its destination at some point in time.
  - This mode messaging does not preclude the necessity to perform request/reply operations.



SWINBURNE UNIVERSITY OF TECHNOLOGY

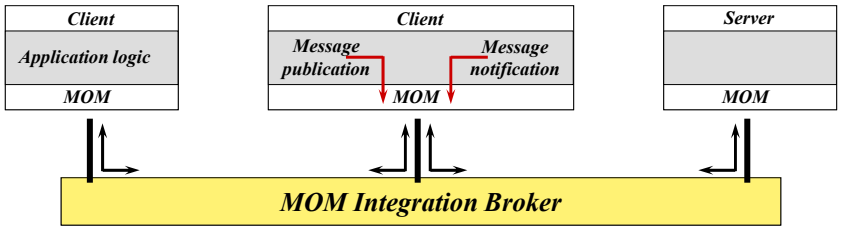
26

26

Web Architectural Styles – Message-oriented

Message-oriented Middleware

- MOM is an infrastructure that involves the passing of data between applications using a common communication channel that carries self-contained messages.
- Messages are sent and received asynchronously.
- The messaging system (integration broker) is responsible for managing the connection points between clients & for managing multiple channels of communication between the connection points.



SWINBURNE UNIVERSITY OF TECHNOLOGY

27

27

Web Architectural Styles – Resource-oriented

Resource-oriented Architecture

- Resources exposed on a network
- Resources uniquely identified
- Application is a series of *linked* resource updates between clients and servers
- Resource examples
  - something returned by an SQL query
  - text or media file
  - a transaction
  - a web site
  - a downloadable program
  - ...

CRUD: Create, Read, Update, Delete

SWINBURNE UNIVERSITY OF TECHNOLOGY

28

28

Web Architectural Styles – Event-oriented

EDA – Event Driven Architectures

- Event emitters and consumers
- Promotes further decoupling of business processes
- Same event can be consumed by a number of other (possibly anonymous) processes.
- Event emitter does not care who consumes the event
- Fits well with declarative rules based approach to defining business processes (as distinct from an imperative activity based approach as in BPEL (ie, the WS composition language))

http://www.infoq.com/presentations/SOA-Business-Autonomous-Components

SWINBURNE UNIVERSITY OF TECHNOLOGY

29

29

How does SOA help achieve loose coupling?

Type of Coupling	Problem Description	Reduced by
Functional	One module relies on the internal workings others	Well defined interface/API describing inputs and outputs
Interface	Interfaces reflect the implementation details	Avoid detailed mandatory parameters; Use more general methods and message passing (do not get too fine-grained)
Data Structure	Data passed between modules reflects internal representations, or is over-constrained	Explicitly define formats, e.g XML in interface; Allow multiple representations e.g.mime-types; 'Vertical' standards that define semantics of messages
Temporal	Client blocked while waiting response Process over-constrained	Asynchronous messages; Explicitly define behaviour in interface e.g. abstract BPEL; Interaction via events / event-driven processes
Address/URI	Addresses change Providers change URI patterns change	Do not hard code references; Virtualise services; Use middleware directory services

SWINBURNE UNIVERSITY OF TECHNOLOGY

30

30

Content

- Software services / web services
- Some concepts you need to understand
- Service oriented architecture (SOA)
- **Web application architectural styles and middleware**
- WS* (SOAP) approach
- REST approach
- Microservices

SWINBURNE UNIVERSITY OF TECHNOLOGY

31

31

Web Application (Standard) Approaches

- Web Service Standards – "WS*"
  - SOAP message and WSDL interface descriptions
  - Stack of other WS* Standards (UDDI, WS-Transaction, WS-Security, …)
  - Service composition and orchestration
    - e.g. BPEL – Business Process Execution Language
  - Middleware based - many frameworks and technologies being developed
    - e.g. Web Sphere, JBoss, Apache Axis, WSO2, …
- REST– based (**Re**presentational **S**tate **T**ransfer)
  - Light-weight approach, inter-operability based on HTTP
  - View machine clients like automatic 'browsers'
  - Resource/data centric
  - http://www.infoq.com/presentations/The-Counterintuitive-Web
- Microservices: functions (data) as service …

SWINBURNE UNIVERSITY OF TECHNOLOGY

32

32

Technologies for Web Architectures

Different Web 'protocol patterns' suit various architectural styles

Architectural Style

- RPC
- Message-oriented
- Resource-oriented
- Event-oriented

Standard Protocol patterns

- **WS* (SOAP, WDSL, …)**
- **RESTful / Microservices**

Note: There are many Web applications that do not follow these standard approaches

33

33

????

Which architectural style is the most "decoupled"?

Which architectural style is the hardest to change / evolve? a.k.a "Fragile"

Method-oriented

Message-oriented

Resource-oriented

Event-oriented

Coupling of reference address, interface, implementation, behaviour, …

Fragility - What breaks when something is changed?

34

34

Content

- Software services / web services
- Some concepts you need to understand
- Service oriented architecture (SOA)
- Web application architectural styles and middleware
- **WS* (SOAP) approach**
- REST approach
- Microservices

SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

35

35

WS* (SOAP) Architectures

- Evolution of the enterprise computing middleware approach
  - CORBA, MoM, ESB
- * = Standards needed, so enterprise system can interoperate
  - Description, Discovery, QoS, Security, Transactions, etc.
- "Big" Web Services – many big vendors developing middleware products to support standards based approach
  - IBM, BEA, Microsoft, JBoss, WSO2, ...
- An application protocol (**SOAP**), views/uses **HTTP** as a ***transport protocol*** to carry SOAP messages
  - Although almost always HTTP, such SOAP services can use other transports i.e. they are not necessarily "Web" services.
- **Calls/Messages** defined and exchanged by machine using **XML Schema**

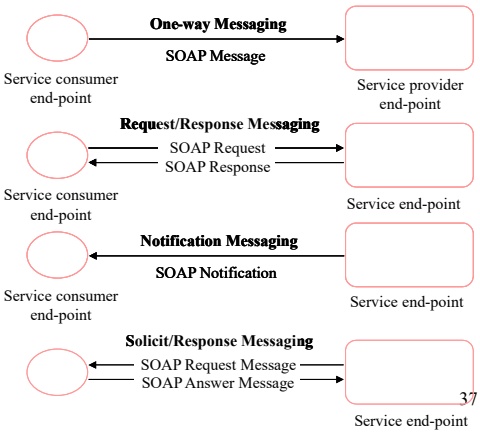
SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

36

36

Message Exchange Patterns

- Four common types of operations that represent possible combinations of input and output messages
- Support for both ***push*** and ***pull*** interaction models at the interface level.
- Not just ***client-server request - response***



37

37

Web Services and Messaging

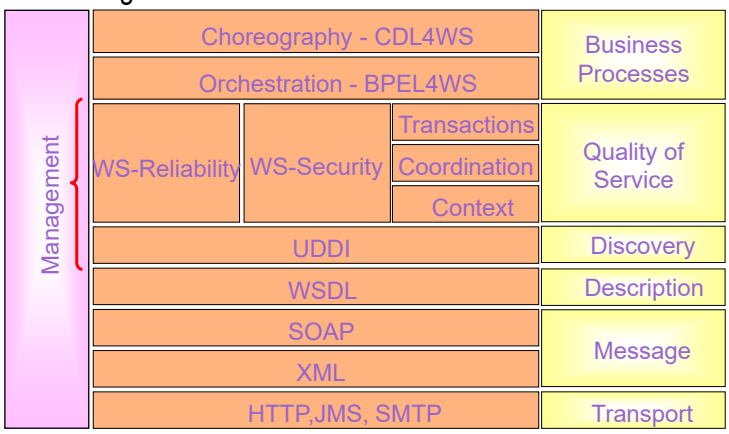
- WS* approach defines a standard messaging format called **SOAP** (originally defined as Simple Object Access Protocol)
- Can use multiple application protocols for transport:
  - HTTP, HTTPS, SMTP, ...
- **WSDL** (service interface description) is designed to work with SOAP

38

38

WS Technology Stack implementation of SOA

- Web services are implemented by a collection of several related technologies & standards.



WS* Standards are defined in XML Schema

39

39

WSDL – Web Service (Interface) Description

- Interface descriptions
- Structure of WSDL documents
- Abstract part
- Concrete part

40

40

An example: in programming language style (illustration only)

```
Service StokeQuote {
  //types
  struct TradePriceRequestType {              //RPC
        string tickerSymbol;
  };
  struct TradePriceType{
        float price;
  };

  Interface StokeQuotePortType {
        void GetLastTradePrice
              (in TradePriceRequestType   GetLastTradePriceInput,
               out TradePriceType   GetLastTradePriceOutput);
        … …
  };
};
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

41

41

Web services in context

- For "use of service" to happen, the consumer needs to understand and the provider needs to tell:
  - What the service can do – functionality: (by WSDL …)
  - How good the service can do it – qualities: (partly by other WS standards …)
  - How to use the service – usage: (partly by WSDL …)
- How can we achieve this "understanding":
  - Self-describing Web service interface (in WSDL, and …)
  - Platform independent (in XML)
  - Provider / Registry tells …
  - Consumer understands …

SWINBURNE UNIVERSITY OF TECHNOLOGY

42

42

Web Services Description Language

- The Web Services Description Language (WSDL) is the XML-based service representation language used to describe the details of the complete interfaces exposed by Web services and thus is the means to accessing a Web service.
  - For instance, neither the service requester nor the provider should be aware of each other's technical infrastructure, programming language or distributed object framework (if any).
- WSDL is a platform-independent description meant to be created and read by machines (not hand-written)
  - Generated from code objects *or* ide design tools

SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

43

43

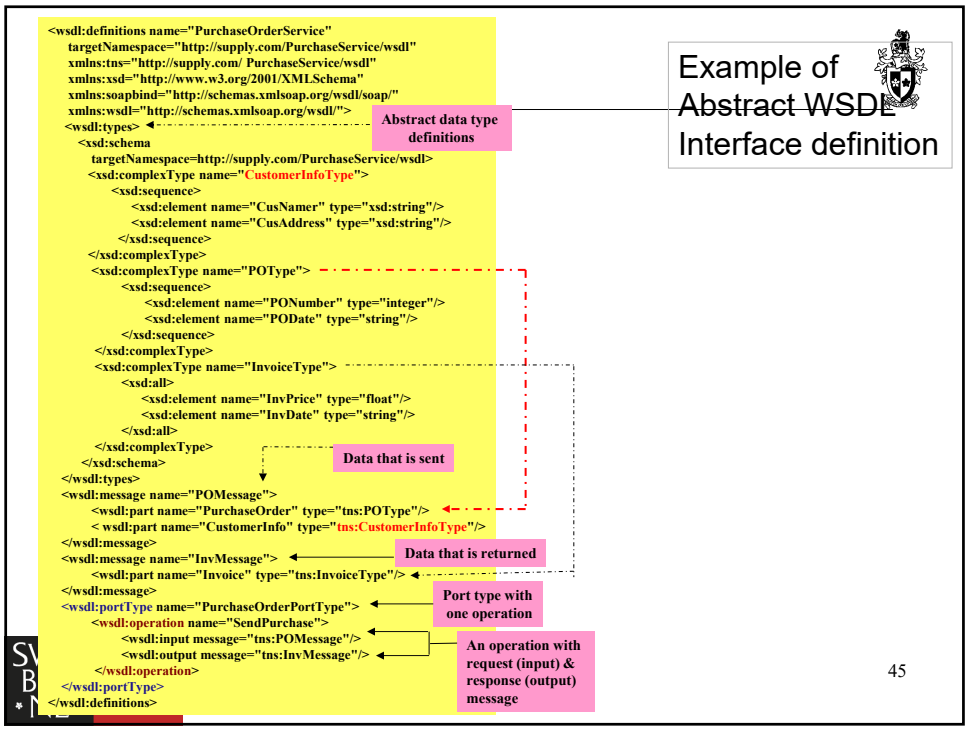
Structure of WSDL documents

- WSDL documents can be separated into distinct sections:
  - The ***service interface definition*** describes the general web service interface structure. This contains all the operations supported by the service, the operation parameters and abstract data types.
  - The ***service implementation part*** binds the abstract interface to a concrete network address, to a specific protocol and to concrete data structures.
- This enables each part to be defined separately and independently, and **reused** by other parts
- The combination of these two parts contains **sufficient information** to describe to the service requester how to invoke and interact with the web service at a provider's site.

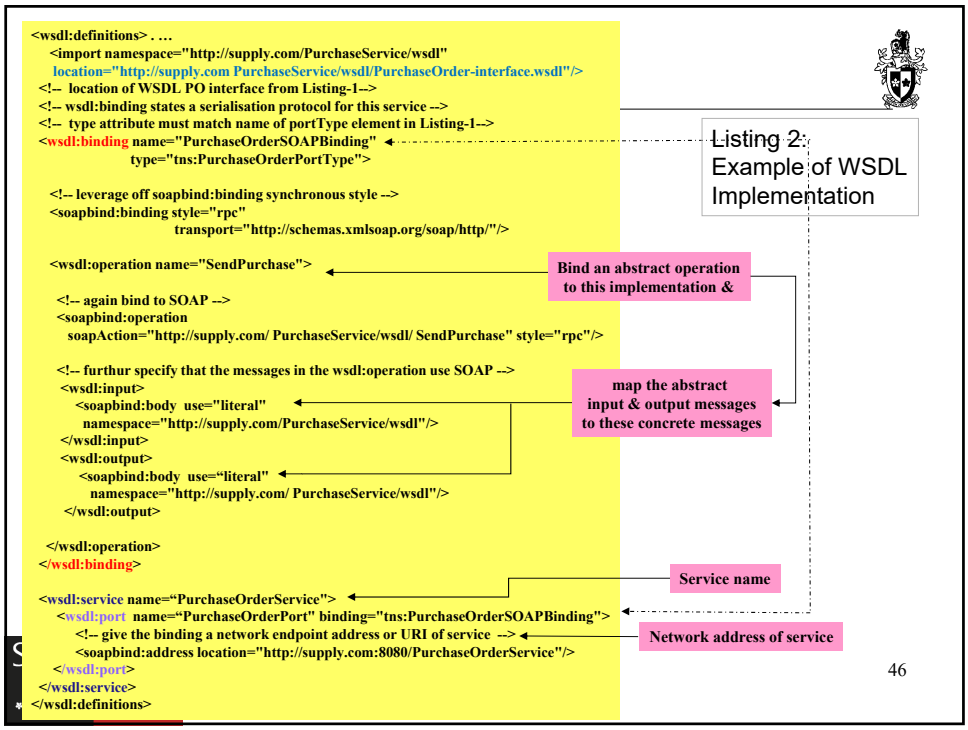
SWIN BUR NE SWINBURNE UNIVERSITY OF TECHNOLOGY

44

44


Example of Abstract WSDL Interface definition
Abstract data type definitions
Data that is sent
Data that is returned
Port type with one operation
An operation with request (input) & response (output) message
45

45


Listing 2: Example of WSDL Implementation
Bind an abstract operation to this implementation &
map the abstract input & output messages to these concrete messages
Service name
Network address of service
46

46

SOAP (Simple Object Access Protocol)

- an XML-based communication protocol for exchanging messages between computers, regardless of their operating systems or programming environments
- Overcomes the problems of conventional distributed object models
  - ☐ Does not require uniform object model across applications
  - ☐ Loose-coupling/strong encapsulation for applications (no internal object ref)
  - ☐ Designed to work with WSDL http://schemas.xmlsoap.org/wsdl/
  - ☐ Use HTTP - work over firewalls or proxy servers, e.g, most firewalls are configured to allow Hypertext Transfer Protocol (HTTP) to pass across

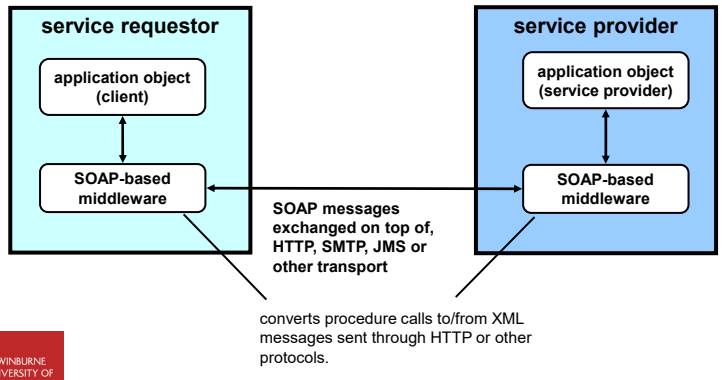
SWINBURNE UNIVERSITY OF TECHNOLOGY

47

47

What is SOAP?

- The standard messaging protocol used by Web services
- Primarily for inter-application communication ('server' to 'server')
- Use an XML-based scheme for encoding message (request and response) data
- Typically uses HTTP as a means for transport



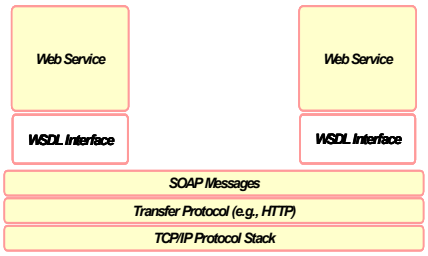
SWINBURNE UNIVERSITY OF TECHNOLOGY

48

48

SOAP as a lightweight protocol

- SOAP is a lightweight protocol that allows applications to pass messages and data back and forth between disparate systems in a distributed environment enabling remote method invocation.



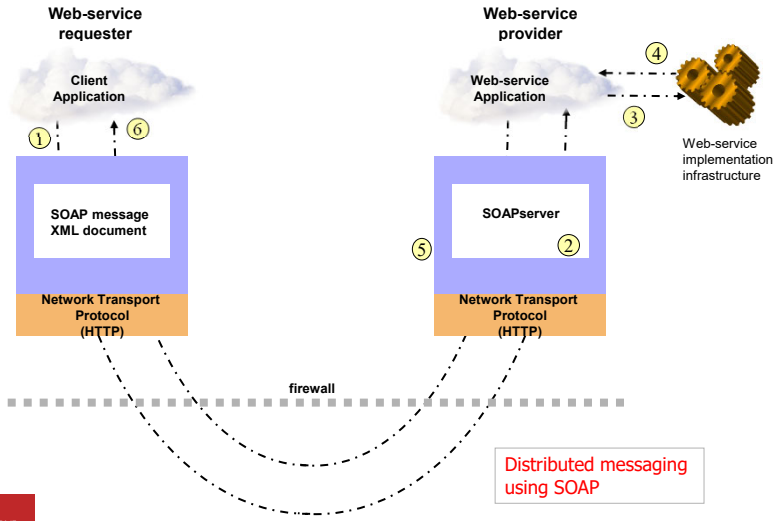
- Lightweight: SOAP protocol possesses only two fundamental properties. It can :
  - send and receive HTTP (or other) transport protocol packets, and
  - process XML messages.
- Supports both RPC and message (document) style interaction
- This can be contrasted with the heavyweight protocols such as ORPC protocols. Designed to work with these.

Guiding principle:
"First invent no new technology"
http://msdn.microsoft.com/en-us/magazine/bb985060.aspx

49

49

Distributed messaging with SOAP

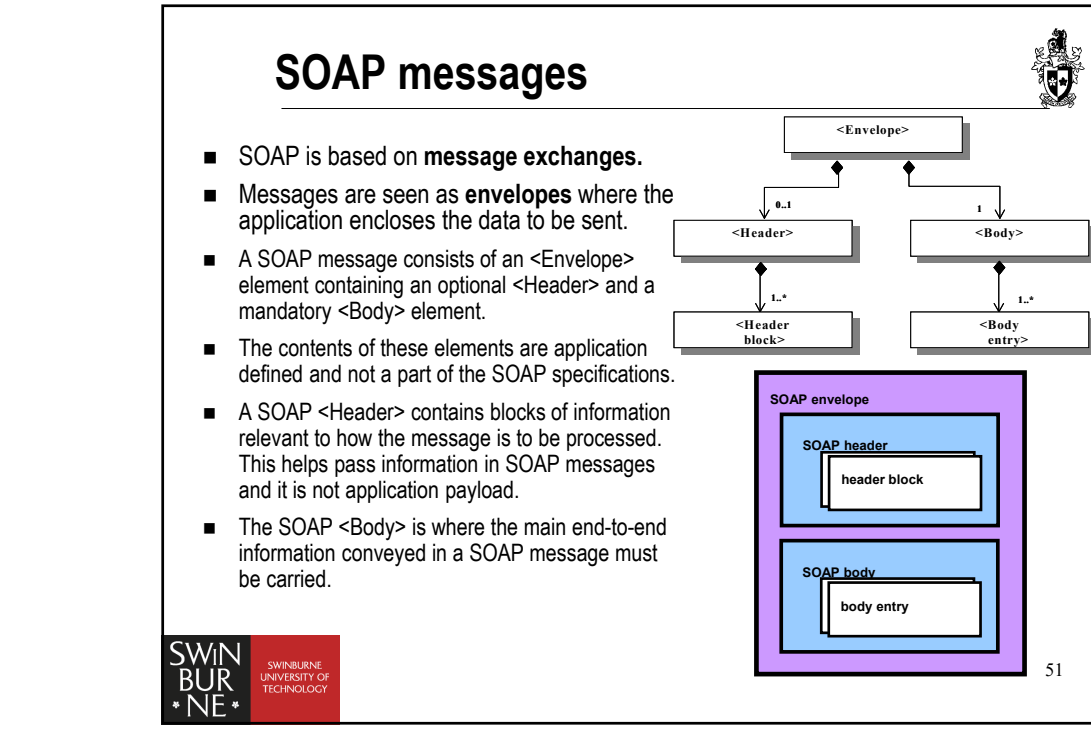


Distributed messaging using SOAP

50

50

SOAP messages



- SOAP is based on **message exchanges.**
- Messages are seen as **envelopes** where the application encloses the data to be sent.
- A SOAP message consists of an <Envelope> element containing an optional <Header> and a mandatory <Body> element.
- The contents of these elements are application defined and not a part of the SOAP specifications.
- A SOAP <Header> contains blocks of information relevant to how the message is to be processed. This helps pass information in SOAP messages and it is not application payload.
- The SOAP <Body> is where the main end-to-end information conveyed in a SOAP message must be carried.

SWINBURNE UNIVERSITY OF TECHNOLOGY

51

51

Example of RPC-style SOAP body

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
        <tx:Transaction-id
            xmlns:t="http://www.transaction.com/transactions"
             env:mustUnderstand='1'>
                512
        </tx:Transaction-id>
    </env:Header>
    <env:Body>
        <m:GetProductPrice>
           <m:product-id>  450R6OP  </m:product-id >
        </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

Corresponding method invocation in programming language style:

GetProductPrice("450R6OP");

SWINBURNE UNIVERSITY OF TECHNOLOGY

52

52

Example of RPC-style SOAP response message

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
      <--! - Optional context information -->
    </env:Header>
    <env:Body>
      <m:GetProductPriceResponse>
         <m:product-price>   134.32  </m:product-price>
      </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```

Corresponding method invocation in programming language style:

134.32

53

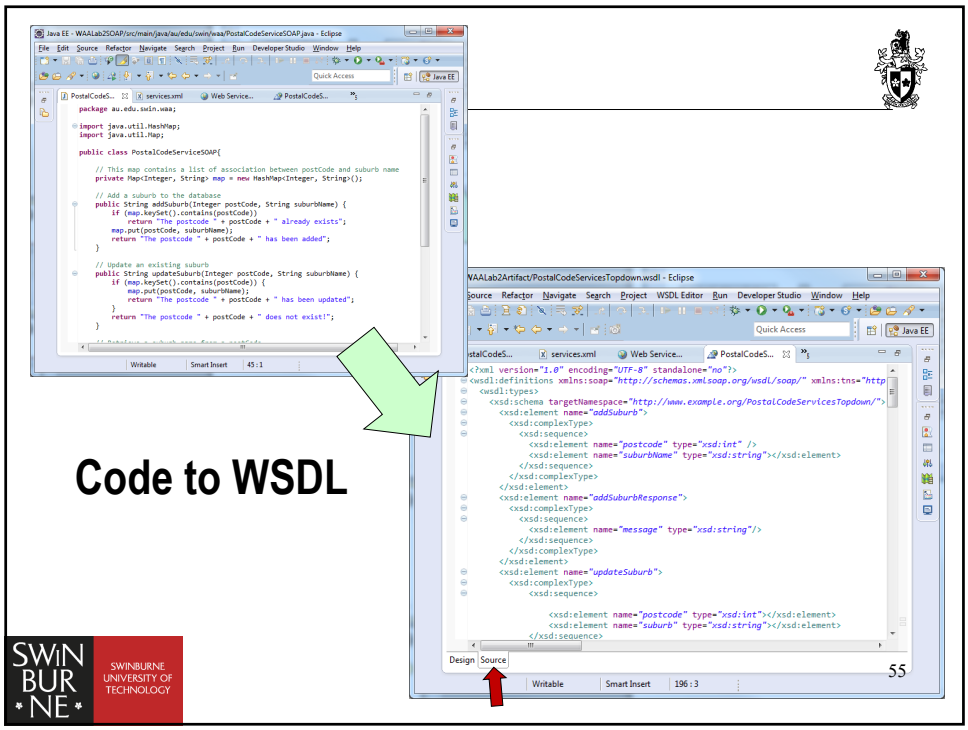
53

Implementing WS* Web Services

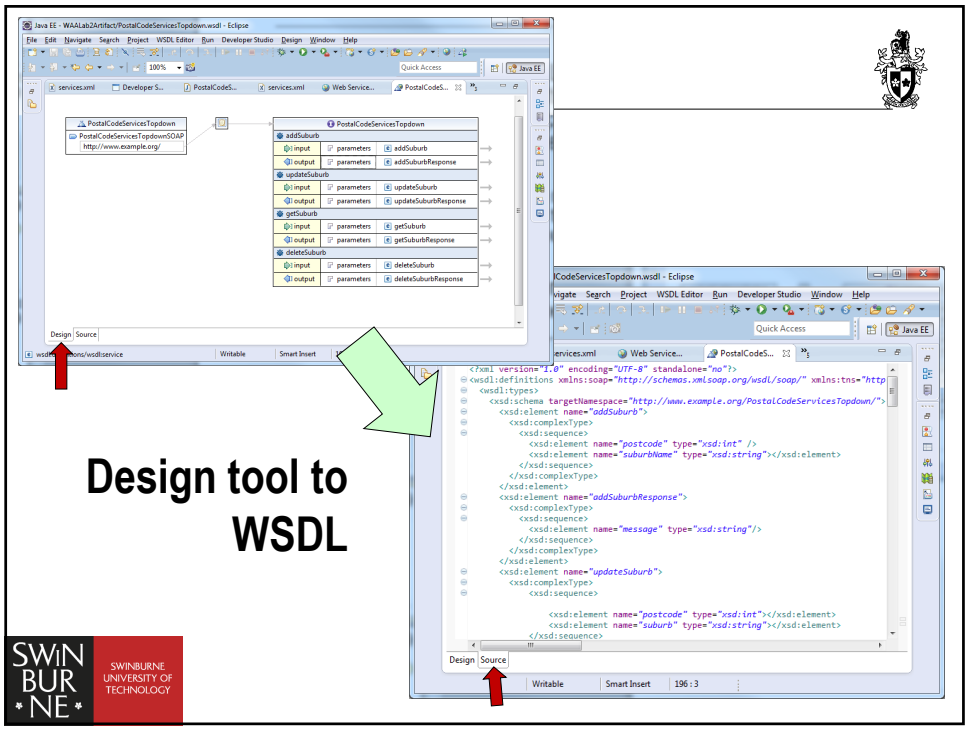
- Design and implement an OO software system
- Generate WSDL interface
- Deploy service
- Publish WSDL interface (API)
- Other services/systems use

54

54


Code to WSDL
SWINBURNE UNIVERSITY OF TECHNOLOGY
55

55


Design tool to WSDL
SWINBURNE UNIVERSITY OF TECHNOLOGY

56

Content

- Software services / web services
- Some concepts you need to understand
- Service oriented architecture (SOA)
- Web application architectural styles and middleware
- WS* (SOAP) approach
- **REST approach**
- Microservices

SWINBURNE UNIVERSITY OF TECHNOLOGY

57

57

REST based architectures

- Purist and Hybrid approaches
- RESTful Web Services (purists)
  - HTTP as an *application protocol*
  - 'Resource-oriented' architecture
    - Everything referenced as a URI
  - HTTP 'verbs' have specific meaning → uniform CRUD interface
    - **GET** – retrieve information on a resource
    - **POST** – Create a new resource
    - **PUT** – Modify an existing resource
    - **DELETE** – Delete an existing resource
    - Also use **HEAD** (get meta-data), **OPTIONS** (check which verbs supported)
  - E.g. Most Yahoo's web services, Amazon's S3 (Simple Storage Service)

SWINBURNE UNIVERSITY OF TECHNOLOGY

58

58

REST – a resource-oriented architecture

- **Re**presentational **S**tate **T**ransfer
- Expose resources to clients on a networked system
  - Everything referenced as a URI
- Client-server architecture
- Application state is driven by the client's updating resources across the resource network *using links* provided by the server(s)
  - *"Hypermedia is the engine of state"*
- What is a resource? Not just relational XML infosets
  - something returned by an SQL query, text file, media file, a transaction, a queue, a web site, a downloadable program, ...
- REST is a style or "design guideline" – *not* a standard

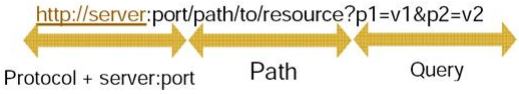
SWINBURNE UNIVERSITY OF TECHNOLOGY

59

59

REST – Resource identification

- All resources (services) referenced by a unique URI
  - e.g. GET



SWINBURNE UNIVERSITY OF TECHNOLOGY

60

60

REST Methods – strict HTTP

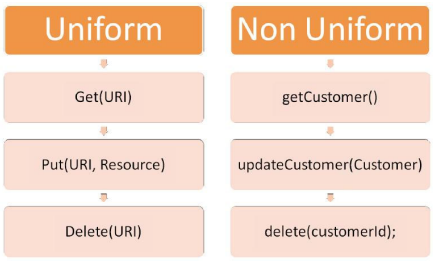
- RESTful Web Services (purist)
  - HTTP as an *application protocol*
  - HTTP 'verbs' have specific meaning → uniform CRUD interface for keeping resources updated:
    - **GET** – retrieve information on a resource
    - **POST** – Create a new resource
    - **PUT** – Modify an existing resource
    - **DELETE** – Delete an existing resource
    - Also use **HEAD** (get meta-data), **OPTIONS** (check which verbs supported)
  - E.g. Most Yahoo's web services, Amazon's S3 (Simple Storage Service)

SWINBURNE UNIVERSITY OF TECHNOLOGY

61

61

REST presents a uniform interface



SWINBURNE UNIVERSITY OF TECHNOLOGY

Graphic courtesy of CSE UNSW

62

62

REST Example

- Step 1 : Client asks for a list of parts
  - Getting the list of parts

SWINBURNE UNIVERSITY OF TECHNOLOGY

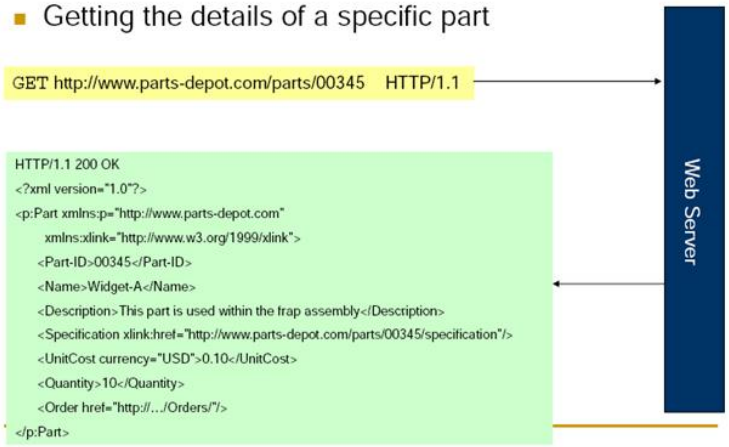
Graphic courtesy of CSE UNSW

63

63

REST Example

- Step 2. Given the part number ….
  - Getting the details of a specific part



SWINBURNE UNIVERSITY OF TECHNOLOGY

Graphic courtesy of CSE UNSW

64

64

REST example

- Placing an order (i.e. creating a new order resource)
  - Placing an order

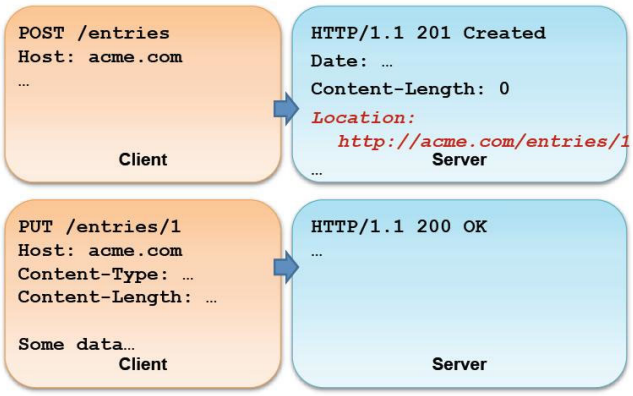
Graphic courtesy of CSE UNSW

65

65

REST example

- Updating a resource



Graphic courtesy of CSE UNSW

66

66

Implementing REST Web Services

- Design and implement an OO software system (main class with CRUD)
- Deploy service (with API)
- Publish API interface
- Other services/systems use

SWINBURNE UNIVERSITY OF TECHNOLOGY

67

67

WS* vs RESTful?

	WS*	RESTful
Easy to understand approach	Complex	Simpler
Easy to implement	Depends...	Depends...
Interface definition	Specific	Generic
Interface Contracts	Yes - WSDL	Not yet (WADL?)
Specific to the Web	No	Yes
Implementation of SOA	Yes	Not really
Scalable	Hmmm....	Yes
Addresses enterprise concerns (e.g. transactions, end-end security)	Yes	Not directly
Orientation	Method	Resource

SWINBURNE UNIVERSITY OF TECHNOLOGY

68

68

Content

- Software services / web services
- Some concepts you need to understand
- Service oriented architecture (SOA)
- Web application architectural styles and middleware
- WS* (SOAP) approach
- REST approach
- **Microservices**

SWINBURNE UNIVERSITY OF TECHNOLOGY

69

69

Microservices

An architectural approach - Some perspectives

- A type of SOA
- Functional perspective
  - Business functionality/capability oriented partition
- Modular
- Small (capability) & light-weight (technology: REST)
- Self-contained & deployable
- Oriented towards development, delivery & deployment
- Scalable
- Granularity vs Complexity …

SWINBURNE UNIVERSITY OF TECHNOLOGY

A compromise: (system) performance v (design) modularity 70

70

Question to Answer – Week 10

The spec of the "Question to Answer" is under the corresponding assignment setup, which will be released after this lecture.

SWINBURNE UNIVERSITY OF TECHNOLOGY

71

71

Required Reading Week 11

- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice* (**4th Edition**), Addison-Wesley, 2021, Chapter 22 (Documenting an Architecture).
  OR,
- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice* (**3rd Edition**), Addison-Wesley, 2013, Chapter 18.

SWINBURNE UNIVERSITY OF TECHNOLOGY

72

72