From Components to Web Services

Dr Balbir Barn
bbarn@acm.org
Thames Valley University

ICEIS 2004

Contents

Part I: Basics

- Tutorial Aims and Objectives
- The Web Services Vision
- Business Drivers (and Challenges)
- What are Web Services?
- How do Components fit in?
- Component Concepts
- Web Service Basic Technologies and Standards

Preliminaries

- Your background
- My background
- What you hope to get out of the tutorial

Aims and objectives

Aim:

- To provide a solid understanding of web service concepts and methods for designing web services

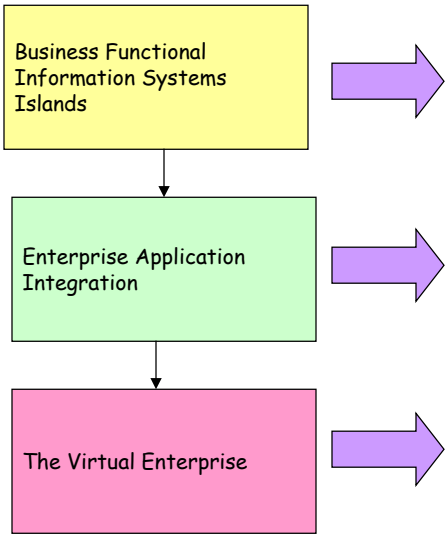
Objectives

- Understand component concepts and their relevance to web service design
- Demonstrate the application of CBD methods to web service design
- Early introduction of UML 2.0 concepts and their relevance to web service assembly

The web services vision

- The changing shape of the enterprise
  - From Stovepipe to Value Chains to Externalization
  - And now: The Virtual Networked Enterprise
- Service based enterprises
  - Business functions provide services to each other at different stages of a business process
  - A business process can viewed as set of services
  - How the service is provided is of no consequence to the user of the service
  - The outsourcing model
    - Execution separate from the use of the service
    - Logistics; call centres all can be outsourced….

The web services vision – enterprise application integration



Each business function has its own systems built with its preferred technologies; manages it own data. There is limited data integration between systems

A common solution with in the org is to use an EAI product to connect applications using messages and hub and spoke architectures.
Transform data from one system to another.

Integration of applications across firms where there severe challenges
Change definitions of enterprise architecture.
The Grid Market Place where EAI is a basic requirement
=> Web Service Technology

Business Drivers (and challenges)

- Connecting Computer Systems
- Two major themes
  - E-business
    - Exposing business processes over the Web
    - The drive towards portals
      - Development portals
      - Business service portals
  - Business process management
    - Systematic automation of every day business processes
    - Re-evaluation / positioning of monolothic applications as components and / or services
    - Focus on integration

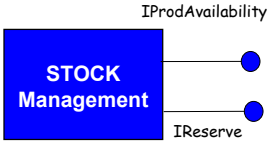
What are web services?

Web Service Definition
" A software application identified by a Unique Resource Identifier, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. A web service supports direct interaction with other software agents using XML-based messages exchanged via Internet based protocols"[1]

- Web Elements
  - Web-based protocols
  - Interoperability
  - XML-based
- Services aspects
  - Modular
  - Available
  - Described
  - Implementation-independent
  - Published

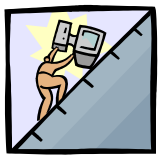
How do components fit in?

- Arisen from similar business drivers
- Characteristics that exist with Web services
  - Separation between implementation and specification
  - Focus on business objects
  - Transparency of location
- Business Object | Component | Service Views

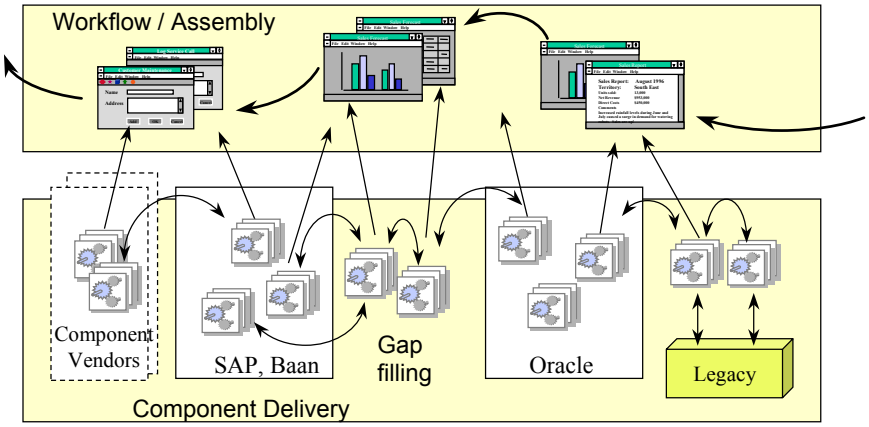


Component Based Development

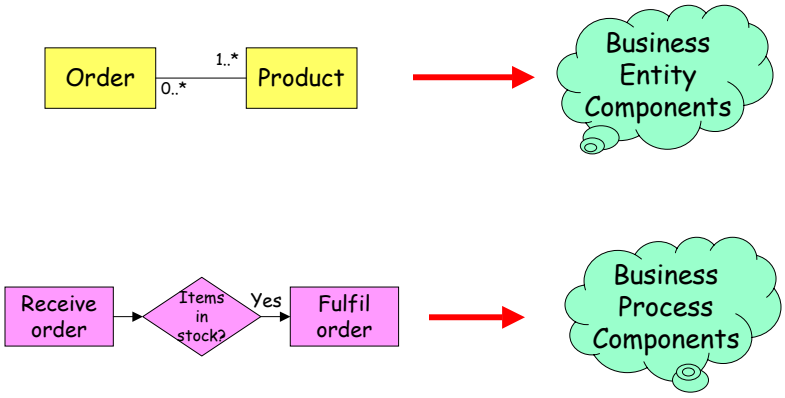
- Fashionable from 1996 to 2002 ☹
- CBD Drivers
  - ***Business Need***
    - Build for change
    - The Web and E-Commerce
    - Integrating Business
  - ***Middleware maturity*** (From CORBA/COM to EJB and DCOM)
  - ***Method Maturity*** (and unification)
  - Processes
- CBD Concepts
  - Specification, Interfaces, Encapsulation Boundaries; Identity….



Application integration with components



Components can model anything!

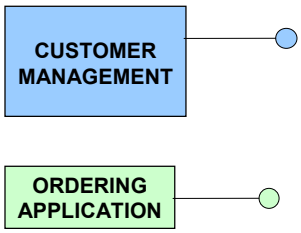


Different uses of “component”

- “Component” is not a well-defined term
- Many people say they are “doing components” but they are doing many different things
- Ranges from loose notion of “useful package” to fully standardised plug-in part
- We use “component” to mean something quite specific...

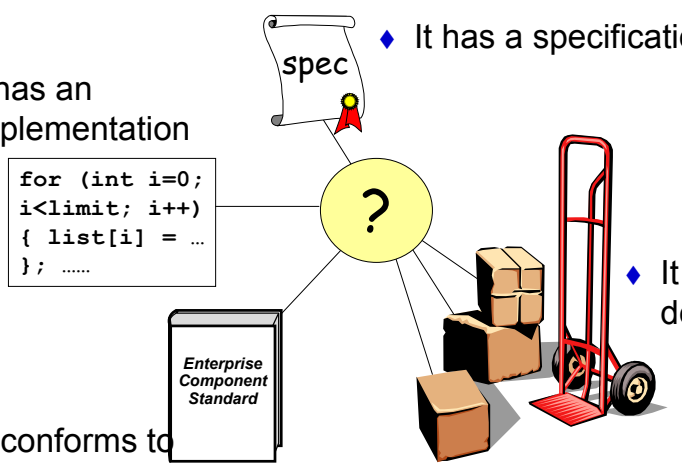
Component Types

- Business Component
  - Entity Component
  - Process Component
- Application Component
  - a special sort of Process Component
- Business Infrastructure Component
- Technical Infrastructure Component



What is a Component (Lite)?

- It has a specification
- It has an implementation
- It can be deployed
- It conforms to a standard
- It can be packaged up

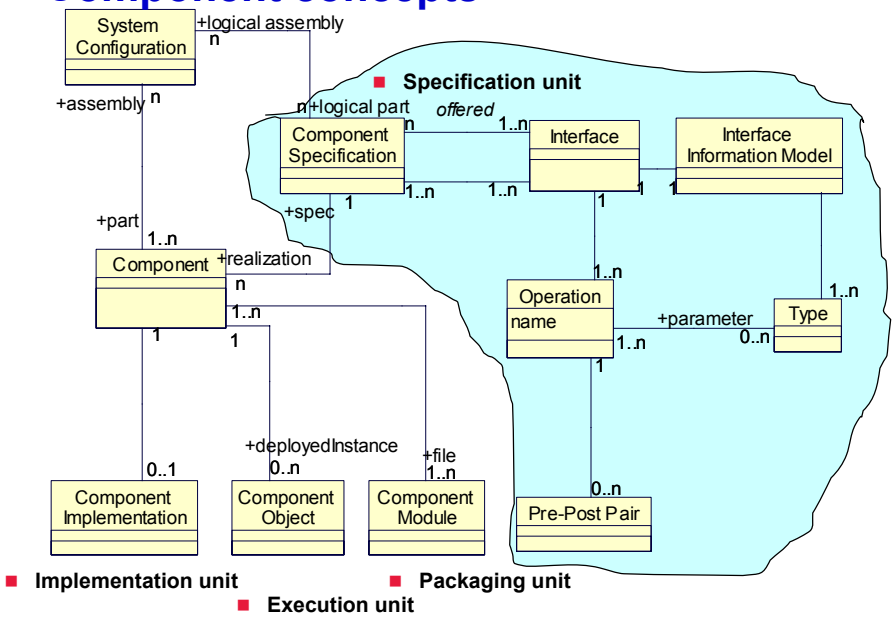


Drilling down…

Component Reference Model

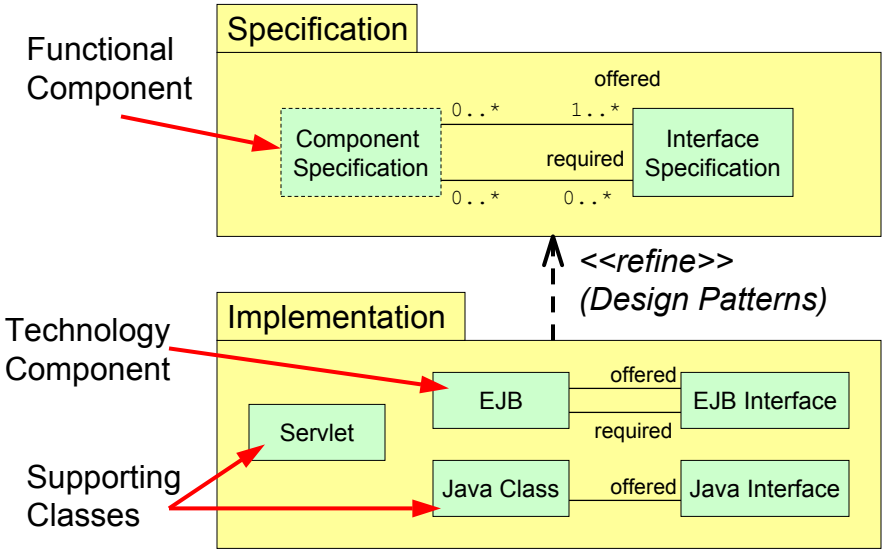
- The Conceptual Model needs to address the following key characteristics:
  - Encapsulation Boundary
  - Independent Deployability
  - Independent Replaceability
  - Precise specification of behaviour
  - Separation between specification and implementation

Component concepts



- Implementation unit
- Execution unit
- Packaging unit

J2EE Functional Component Structure



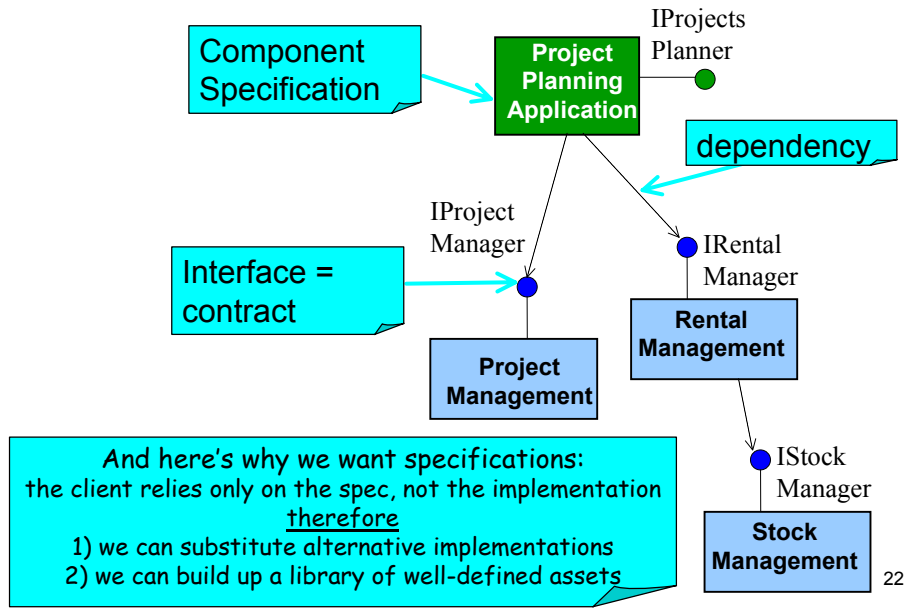
Acknowledgement to John Cheesman 2004

What is a Component Specification?

- A Component Specification
  - defines an implementation unit
  - defines a deployment unit
  - is a description of the guaranteed behavior of a component
  - is expressed in terms of interfaces and constraints

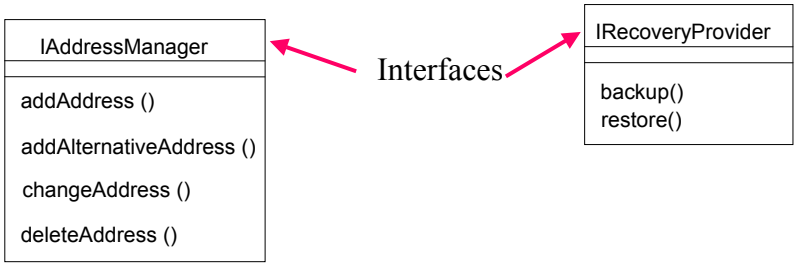
- The specification may include implementation constraints and execution constraints, which impose additional rules upon every implementation or executable.

Why have Component Specifications ?

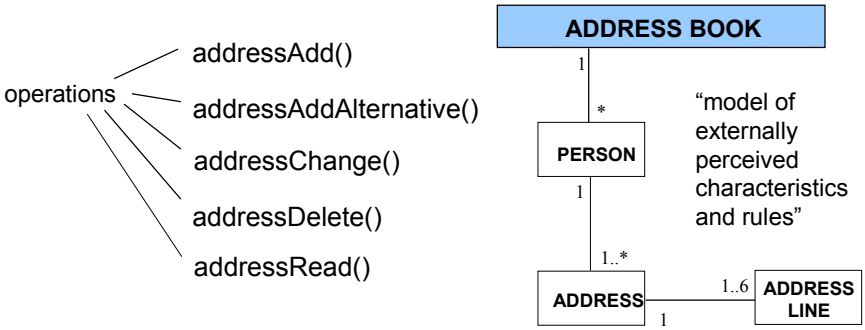


What is an Interface?

- An interface ...
  - is a collection of related operation specifications
    - must be implemented together
  - defines dependencies on other interfaces
    - e.g. IOrderManager must use ICustomerManager
  - is stable once published



Component Specification



- The total description of component behavior
- Any implementation and executable must conform to this specification

Why have a Component Reference Model?

- Multiple component technologies
  - Understand current and emerging technologies from a common reference point
  - Independence from technologies
- Application Development toolset and Methods
  - Support for development and consumption of components
  - Method standardization and development

Web Services Concepts in more detail

Web service basic technologies and standards

- Requirements for a Web Service Architecture

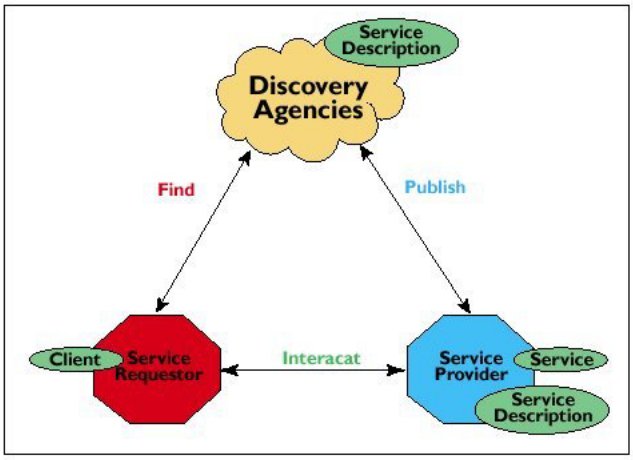
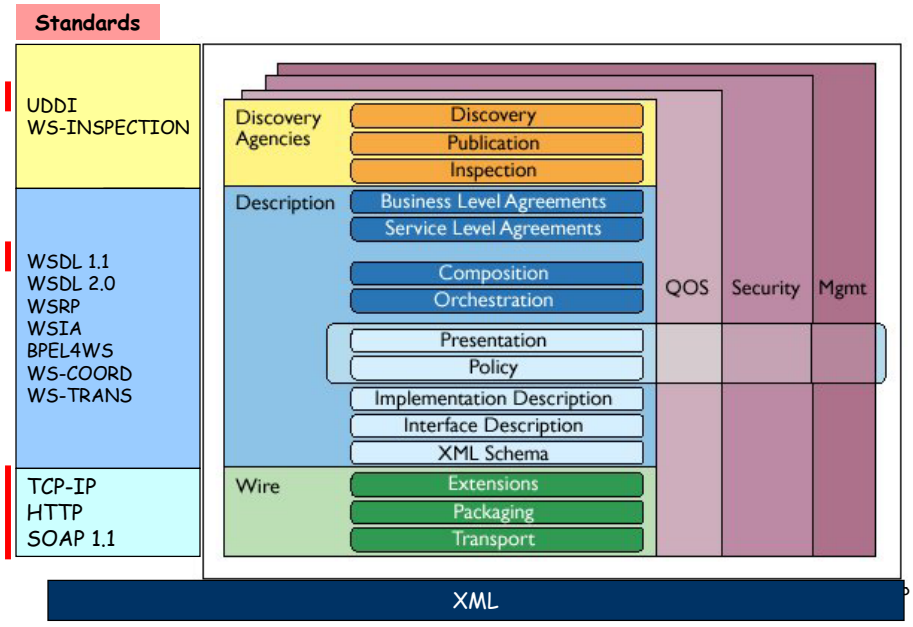


Diagram from Web Services Architecture Draft; www.w3.org/TR/2002/WD-wsa-xxx

Web Service Stack and its evolution



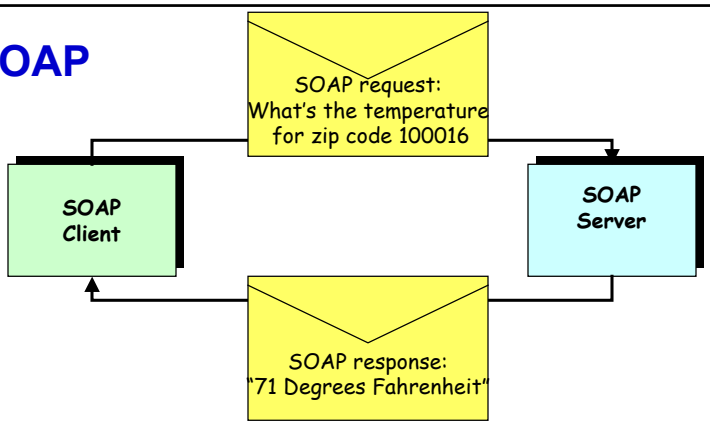
Using the basic services

- Existing services – WSDL; UDDI and HTTP / SOAP 1.1
- Still excellent application integration technology

Web Service Example

- What do we need to specify the supply and use of this web service?
  - What can I do?
  - Where am I located?
  - How can you access me?
  - How do you know that I am the right service for you?
  - How can I get this data to you and you will know what the form of the data is?

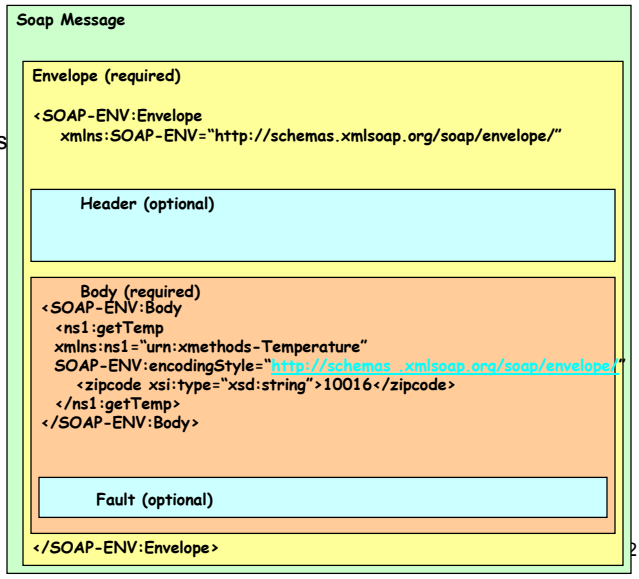
SOAP



- SOAP provides XML based protocols for exchange of data via HTTP
  - Other low-level protocol bases also possible (e.g. Message based)
- Like CORBA, RMI (Java)
  - Except XML based
- Many implementations available (not all compatible with each other)

Details of SOAP Messages

- SOAP Elements
- SOAP Envelope
  - Method names
  - Method parameters
  - Who should process the data
  - Failure/Error Encoding
- Data Encoding
  - Data typing rules based on W3 XML Schema
- RPC Conventions
  - One-way
  - Two-way
  - Method response



Details of a SOAP Request

```
POST /PubsWS/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://Semoris/XmlWebServices/GetAuthorName"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAuthorName xmlns="http://Semoris/XmlWebServices/">
      <s>string</s>
    </GetAuthorName>
  </soap:Body>
</soap:Envelope>
```


Details of a SOAP Response

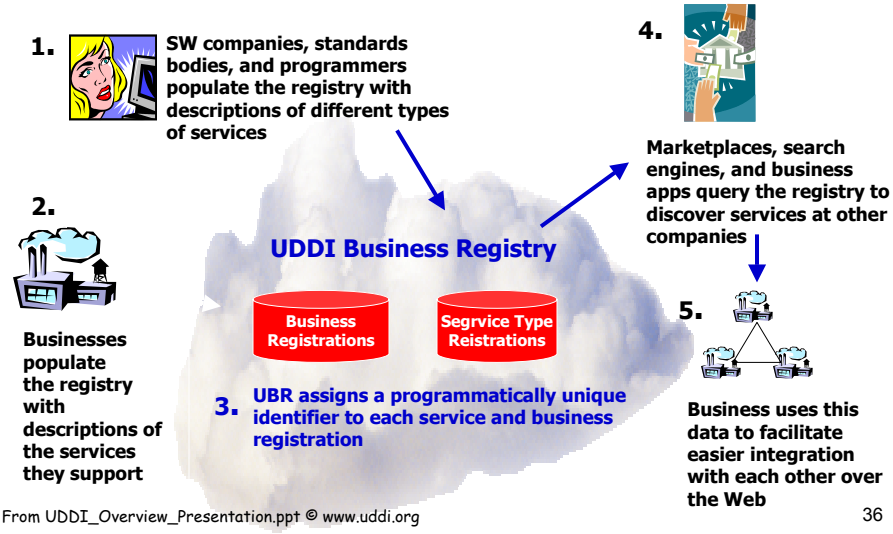
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetAuthorNameResponse
xmlns="http://Semoris/XmlWebServices/">
      <GetAuthorNameResult>
        <xsd:schema>schema</xsd:schema>xml</GetAuthorNameResult>
    </GetAuthorNameResponse>
  </soap:Body>
</soap:Envelope>
```

What is UDDI?

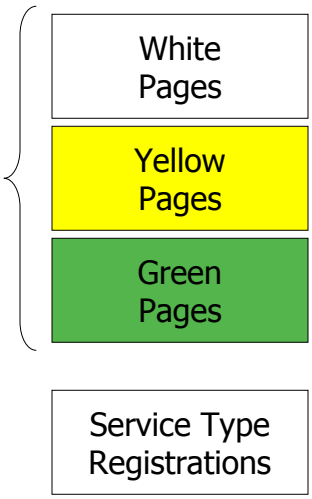
- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a business registry on the web
- Partnership among industry and business leaders
- *Universal Description*, *Discovery*, and *Integration*

How UDDI v1 Works



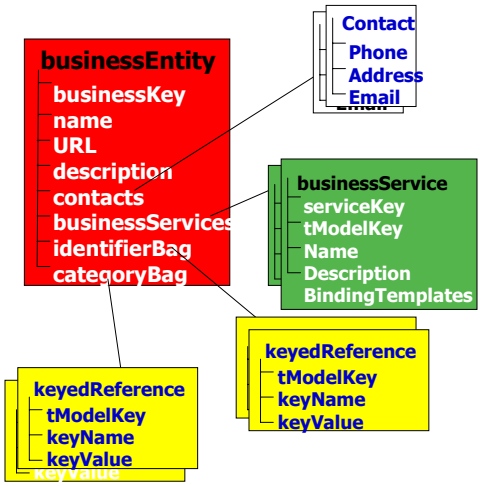
Registry Data

- Businesses register public information about themselves
- Standards bodies, Programmers, Businesses register information about their Service Types

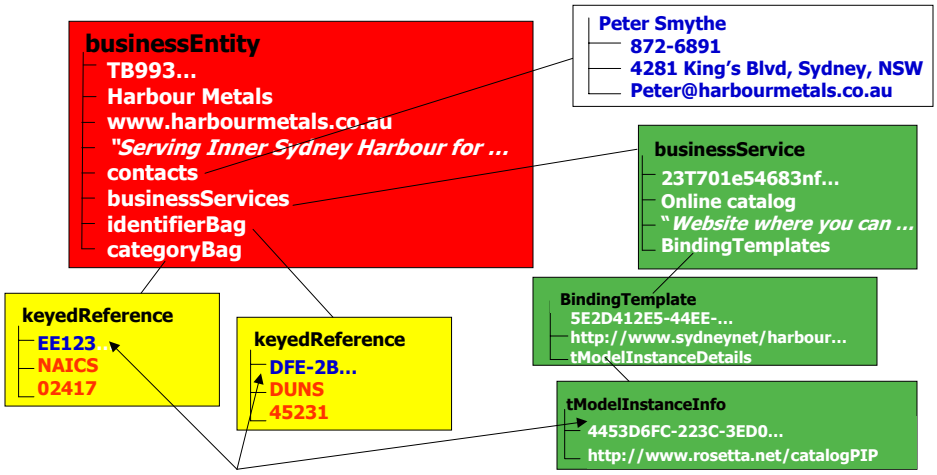


Business Registration

- XML document
- Created by end-user company (or on their behalf)
- Can have multiple service listings
- Can have multiple taxonomy listings

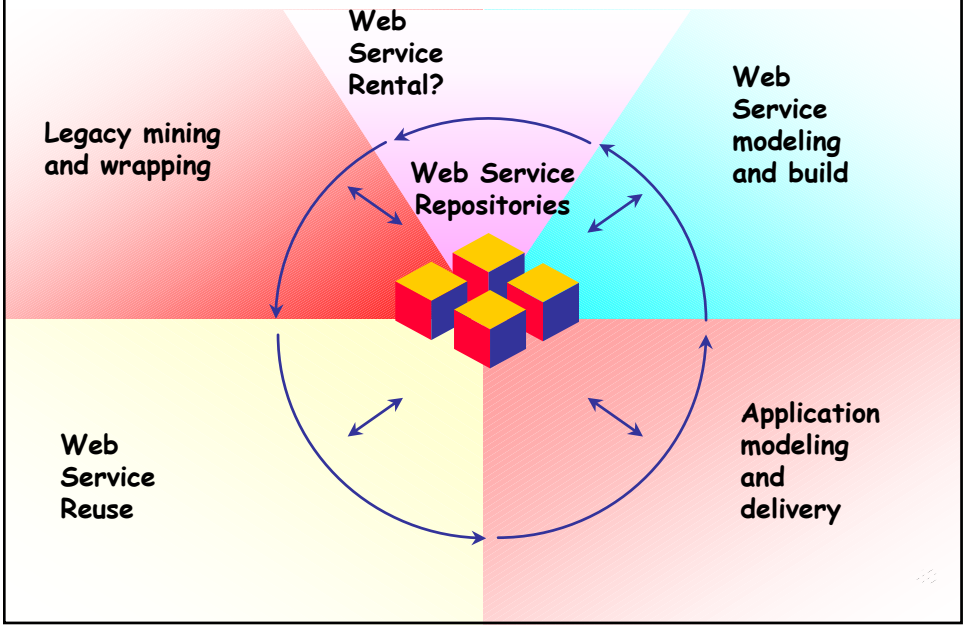


Example of a Registration



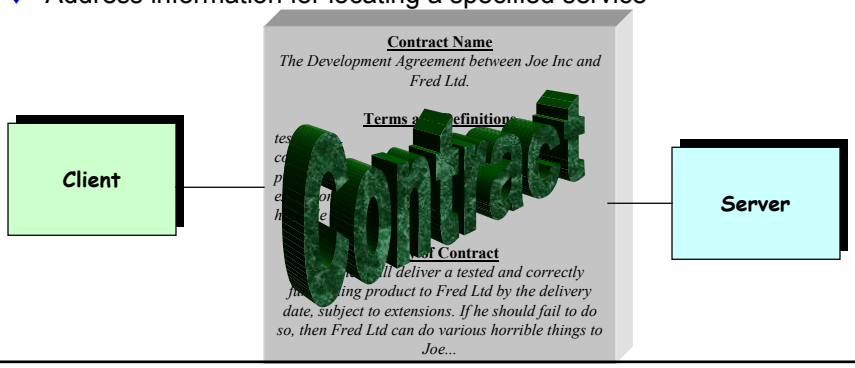
From UDDI_Overview_Presentation.ppt © www.uddi.org

Web Service Activities

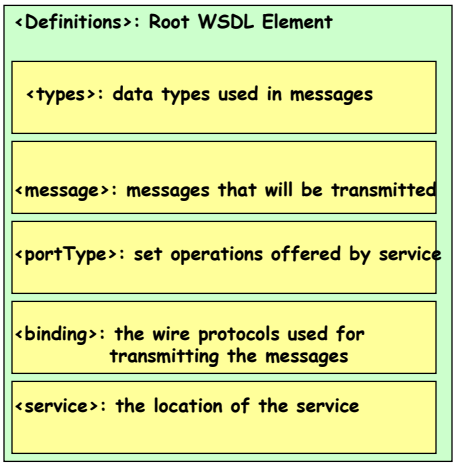


WSDL 1.1

- XML-Based protocol
- Enables a precise description of a web service
- Analogous to CORBA IDL (Interface description language) used for CORBA components
  - Interface information describing all publicly available functions
  - Data type information for all message requests and message responses
- Binding information about the transport protocol
- Address information for locating a specified service



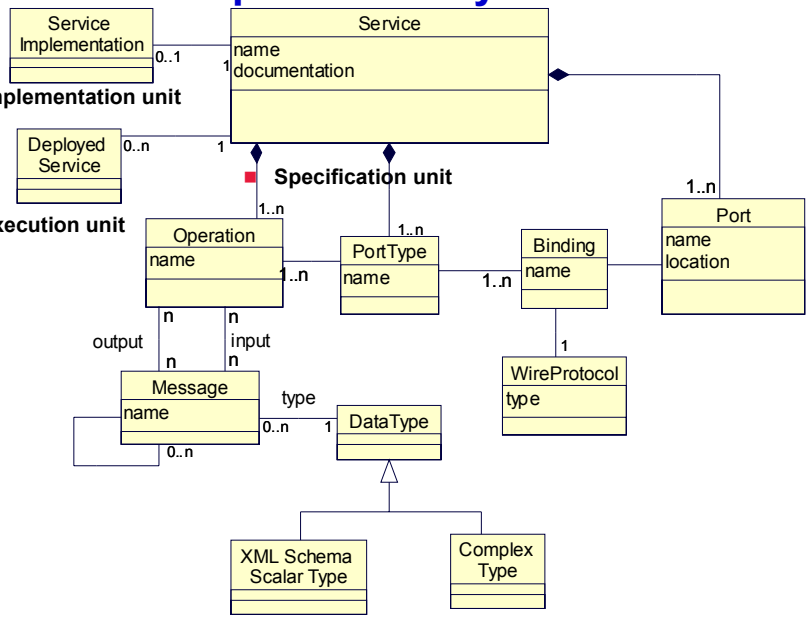
Anatomy of a WSDL specification



- definitions
  - Root element;
  - Name
  - Declaration of namespaces
- types
  - Describes all the data types used between the client and server
  - W3C XML Schema
- messages
  - Describes one-way message request or response
- portType
  - Defines sets of operations utilizing messages
- Binding
  - Specifics about the wire protocols
- service
  - Defines the address for invoking the service
  - (URL for the SOAP service)

WSDL concepts summary

- **Implementation unit**
- **Execution unit**



WSDL Characteristics

- “Componentized” for specification re-use
  - Data Type packages
  - Message packages
  - Fault Message packages
  - Import facility

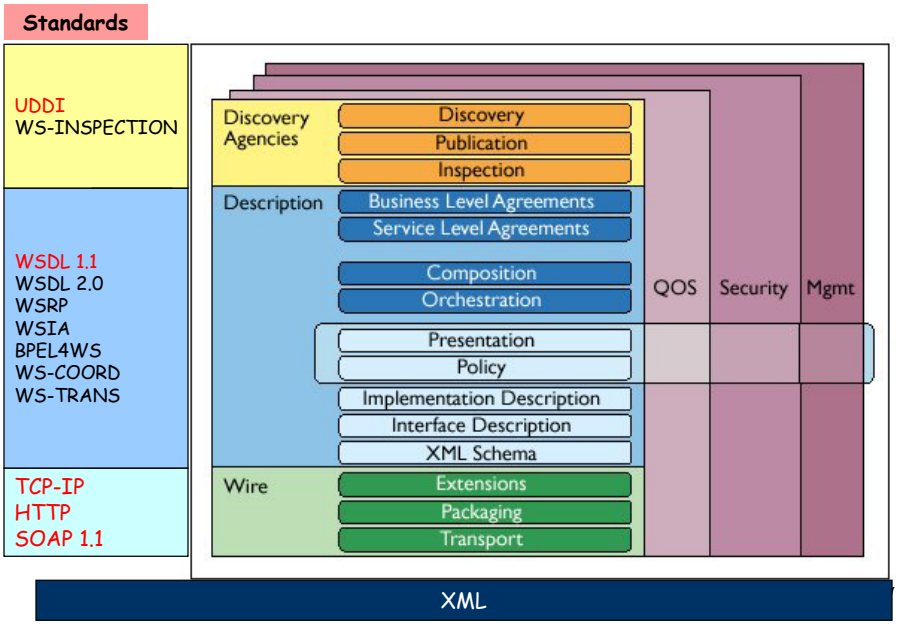
Mapping Component concepts to Web Service concepts – Exercise 1

Component Concept	Web Service Concept
System Configuration	
Component Specification	
Interface	
Interface Information Model	
Operation	
Pre-Post Specification	
Information Type	
Component	
Component Implementation	
Component Object	
Component Module	

What does this mean?

- **Mappings indicate that:**
- **We can leverage approaches and maturity of CBD practice to the design and implementation of web services**
- **We can identify additional requirements on standards**
- **We can tailor existing methods for software development**
- **We can review the applicability of UML 2.0 in the context of web services modelling.**
- **This tutorial will focus on methods for modelling web services**

Status of standards



Part II: Methods for specifying services and assemblies

Considerations and Inputs:

- Component Based Design Techniques
  - Interface specification
  - Interaction modelling
  - Assembly considerations
- UML 2.0 semantics and notation
- Web service specification standards
- Method areas addressed:
  - Business application assembly from web services
  - Business process assembly and automation

Hotel Case Study

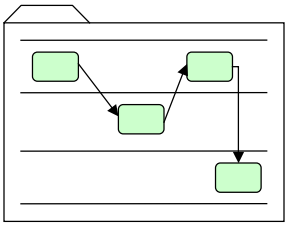
- Adapted from (Cheesman, Daniels 2001)
- There is a requirement to develop a new web based information system to support reservations and other related functions. Currently, the Hotel group has already purchased a Billing System with which the organization is relatively satisfied. The Group has also an existing contract with a on-line credit checking agency. There are basic functional requirements to handle the creation of new reservations, updates and canceling of reservations. Changes to reservations, modification of customer information and other mechanisms to contact customers for various notification purposes form some additional requirements.
- The Hotel Group offers a variety of hotels, room types and prices vary according to market demand.
- At the moment, the implementation strategy has not been determined but a J2EE architecture or a web services model is being considered as the Hotel Group want to recoup the investment in this system by selling services (by consumption) or components to smaller hotel management companies who have similar requirements.

Exercise:

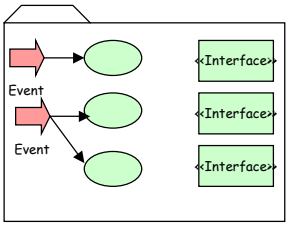
1. Identify Business (Web) Services or components that you think that this system will provide or utilize.
2. How would you group operations to web services?
3. Provide a short rationale for your choices/

- Timings:
- 10 minutes
- 5 minutes feedback

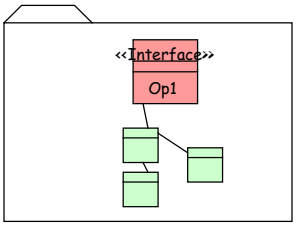
Business application assembly from web services



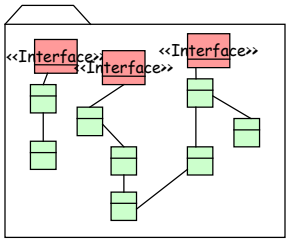
Business Process Modelling



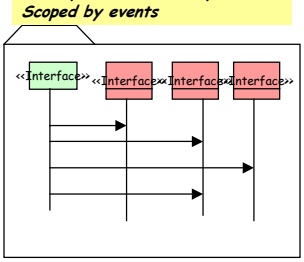
Decomposition into sub-processes
Scoped by events



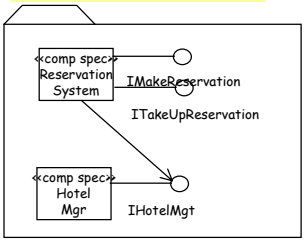
Business Service Specification



Domain Modelling
Identiifying Business Services

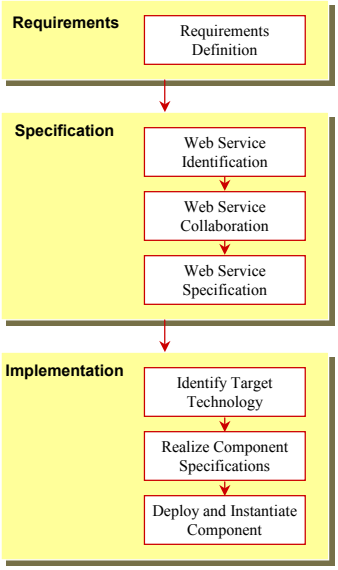


Interaction Modelling
Refining Business Services
And operations



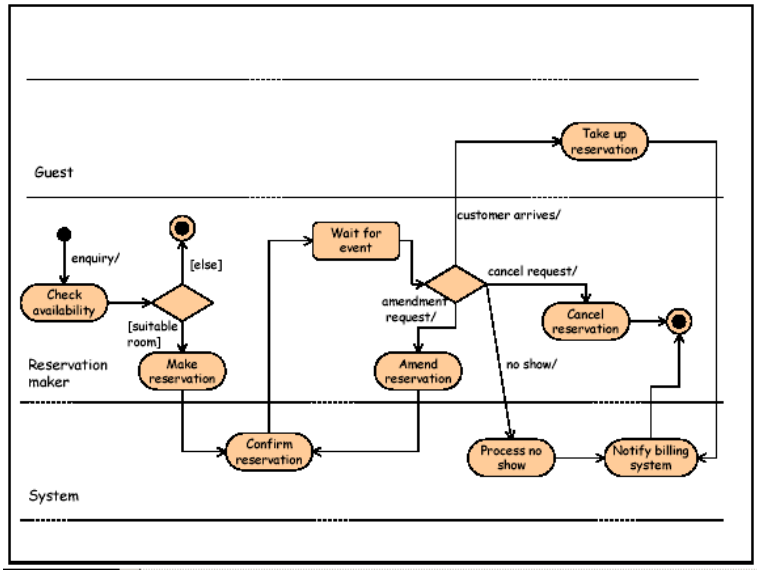
Application Assembly - Specification

Business Web Service Development Process

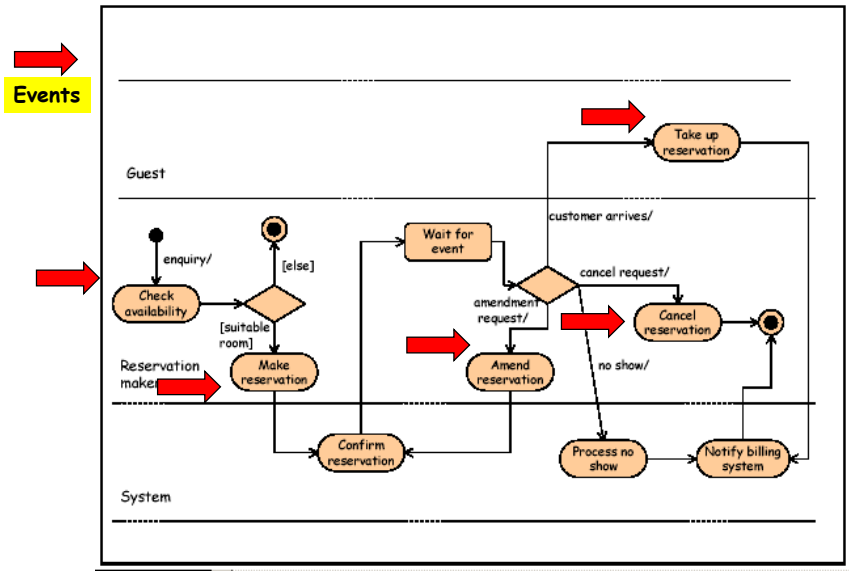


- Requirements definition
  - Develop the Business process model
  - Develop the Business context model
  - Produce Use case context model
  - Specify Use cases
- Web Specification
  - Specify Web Service Interfaces
  - Partition components
    - Business Services
    - Application Services
  - Specify Web service collaborations
- Implementation
  - Identify target technologies
  - Realize component specifications
  - Deploy and instantiate components

Business Processes



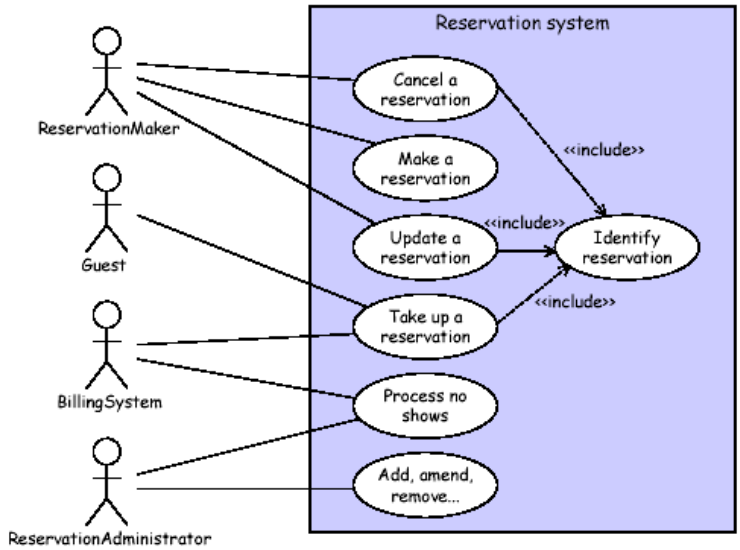
Business Processes



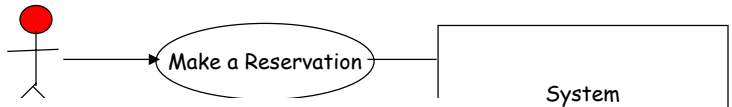
Identifying System Interfaces

- Systems require dialog between actors and the functionality of the system
- The Dialog is represented by the step logic in the Use Case Descriptions
- Each use case represents sets of dialogs with the system. These become system interfaces.

Use Case Diagram showing <<includes>>



Defining a System Interface



Name	Make a reservation
Initiator	Reservation Maker
Goal	Reserve room(s) at a hotel

Main success scenario

1. Reservation Maker asks to make a reservation
2. Reservation Maker selects in any order hotel, dates and room type
3. System provides price to Reservation Maker
4. Reservation Maker asks for reservation
5. Reservation Maker provides name and postcode (zip code)
6. Reservation Maker provides contact email address
7. System makes reservation and allocates tag to reservation
8. System reveals tag to Reservation Maker
9. System creates and sends confirmation by email

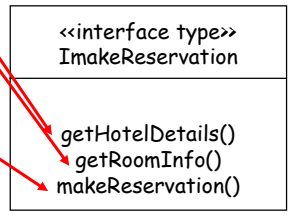
Extensions

3. Room not available
   a) System offers alternatives
   b) Reservation Maker selects from alternatives

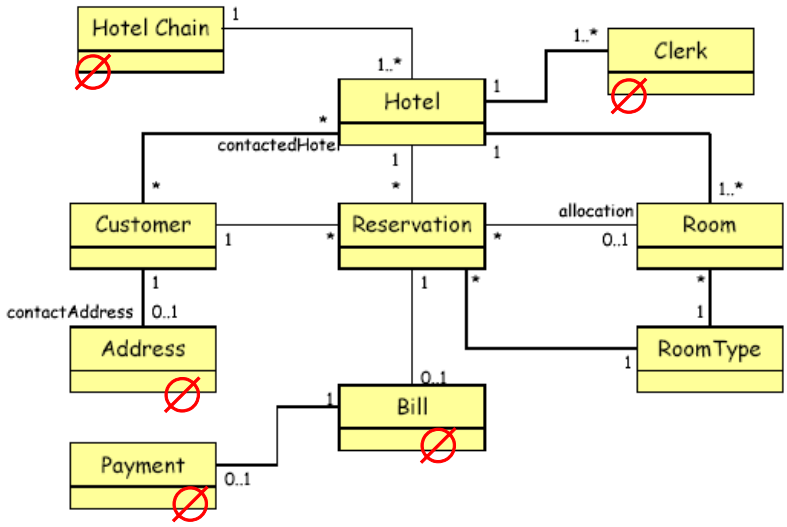
3b) Reservation Maker rejects alternatives
   a) Fail

4. Reservation Maker declines offer
   a) Fail

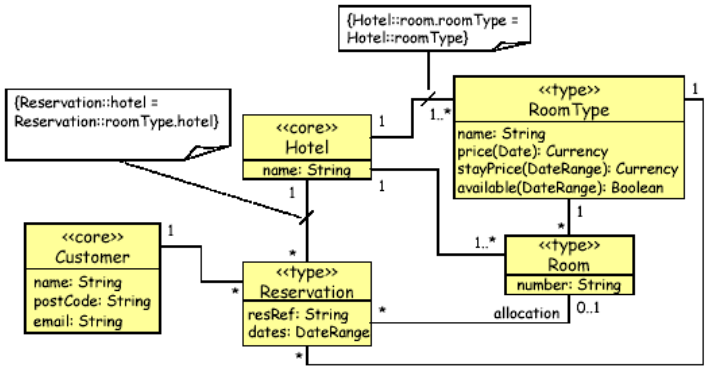
6. Customer already on file (based on name and postcode)
   a) Resume 7



Scoped Business Type Model

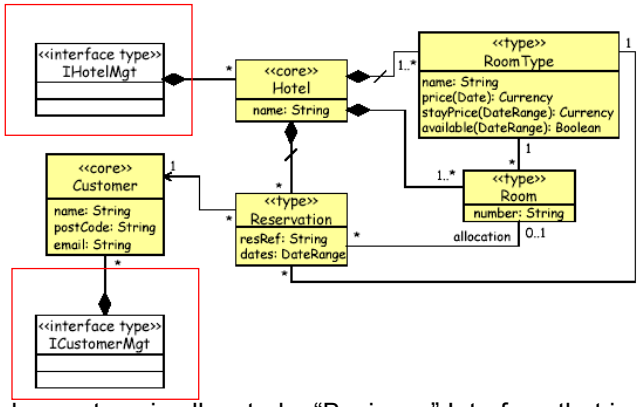


Identifying Core Business Types



- Identify core types by thinking about which information is dependent on other information and which information can stand alone
- Characterised by:
  - A unique business identifier (primary key)
  - Independent existence (no mandatory associations)

Define Business Interfaces

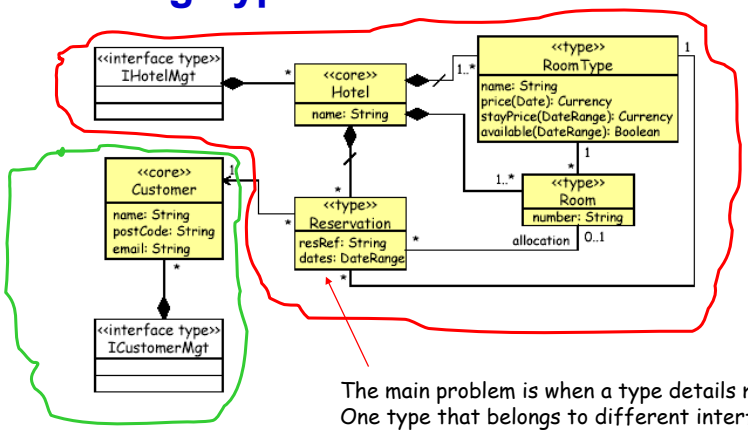


- Each core type is allocated a “Business” Interface that is used to manage instances of core types.
- This is more efficient than creating a “Icustomer” Interface. As we would then end up with many component objects rather than a single component object which simply manages a data structure representing a customer object.

Allocating Responsibilities

- Goals:
  - We want each type (core or otherwise) to be owned by just one Interface
- Core Types are allocated to the “managing” interface (indicated by a containment symbol)
- If a type provides details (is related to) to a core type then it is also allocated to the same interface.

Allocating Types to Interfaces

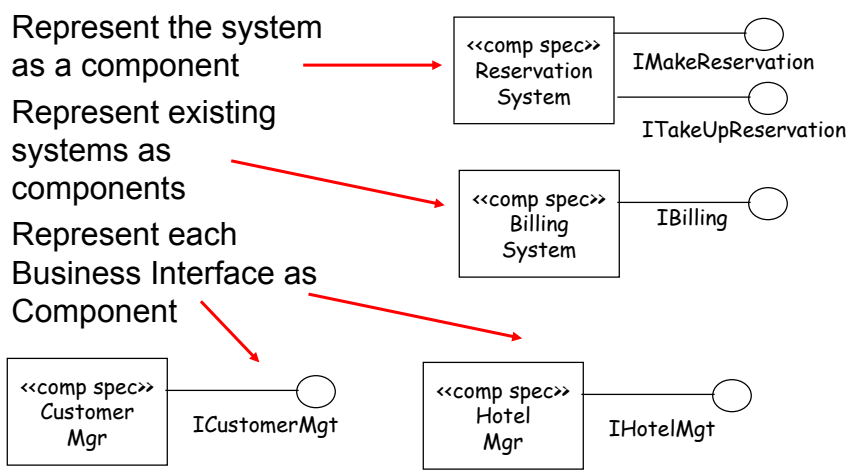


The main problem is when a type details more than One type that belongs to different interfaces.

Here: Reservation is a detail of Room and Hotel.
It could go to either.
We choose to allocate it to the IHotelMgt Interface Because the only information that Hotel needs to Know about Customer is the customer id.

System Component Specification and others

- Represent the system as a component
- Represent existing systems as components
- Represent each Business Interface as Component



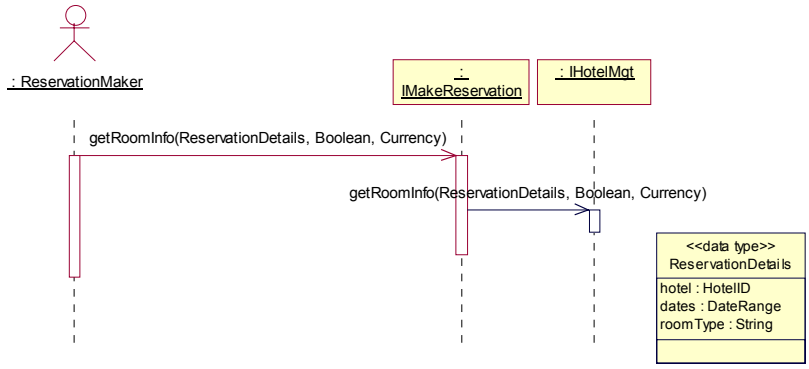
Discovering Business Operations

- A rigourous process that:
- Takes each operation on the system interfaces
- Draws a collaboration diagram that “traces” or executes the constraints or invocations of the system interface operation
  - Collaboration diagrams show the invocation of business operations between Business Interfaces
- Each requirement for an operation results in an operation being added to the Interface

Interaction Diagrams – Client Server Models

- Functionality of a system is “implemented” by one object requesting a service from another object
  - The client server model
- The ***Server*** may provide the service requested by requesting services of other objects and then responding back to the original ***client.***
- Diagrams representing the requests of services and their fulfillment are called Interaction Diagram
  - Sequence Diagrams
  - Collaboration Diagrams

Dealing with getRoomInfo



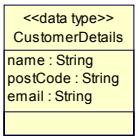
- We want to get details about a room so we supply info about the required reservation
- We will get back availability and the cost
- Again there is simple delegation to an operation on the Business Interface

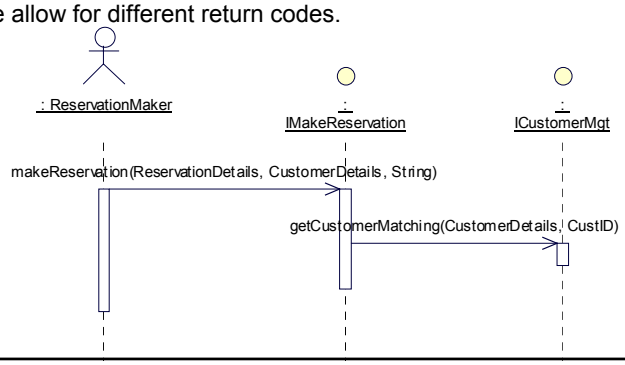
Dealing with makeReservation

- From the use case; the makeReservation operation must create a reservation and notify the customer via email.
- We first define the operation on the system interface.

makeReservation(res : ReservationDetails, cus : CustomerDetails, resRef : String) : Integer

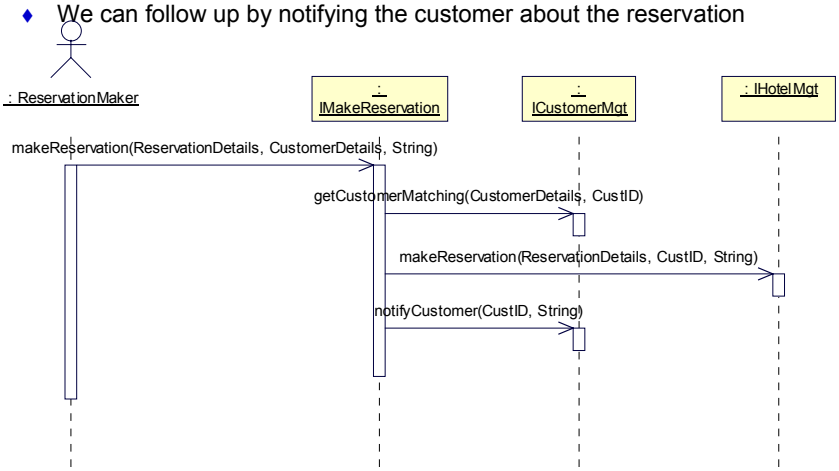
- A new structure – CustomerDetails is also defined
- The operation will return a reservation reference string but because there is no guarantee of finding an existing reference we allow for different return codes.



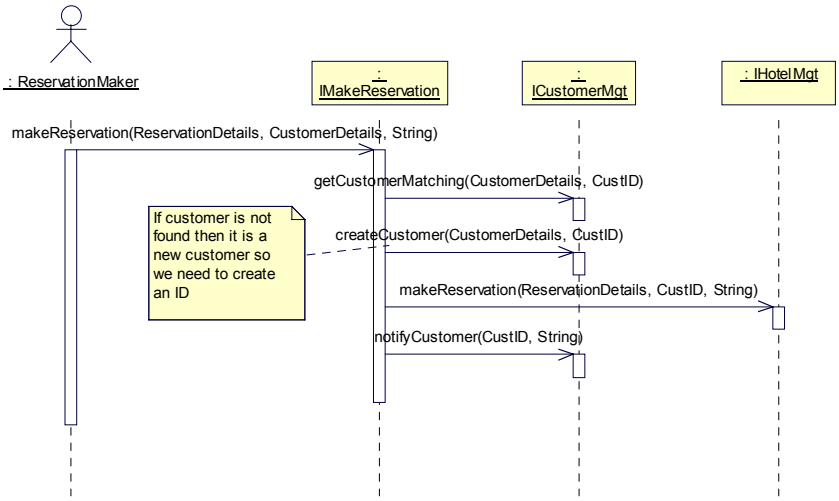


makeReservation….

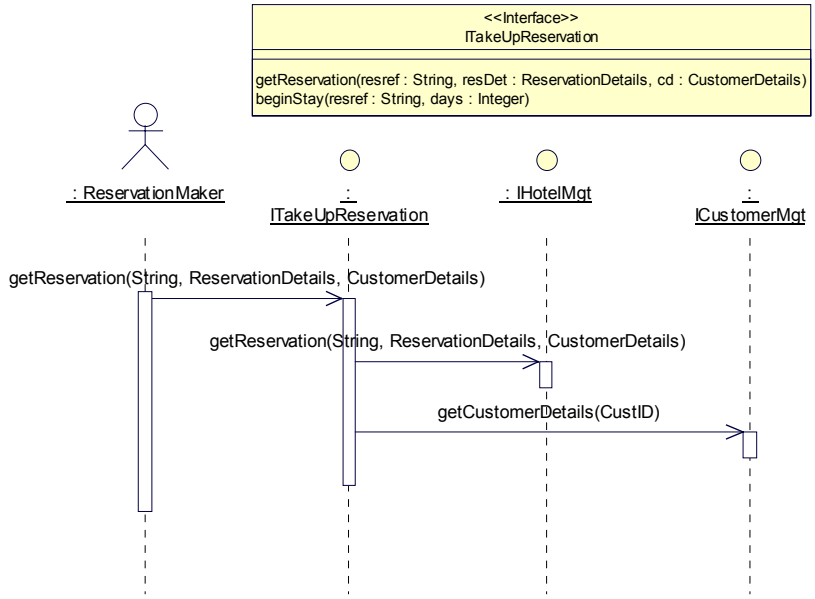
- Having got the customer ID of the matching information we supplied we can now get the Hotel Management component interface to make the reservation with the customer ID that we have located.
- We can follow up by notifying the customer about the reservation



When the customer does not exist…



Completing the Interactions



Contracts

- Usage Contracts
  - (aka Client Supplier Contract)
  - The contract between a component specification interface and a client that wishes to use a service of the interface
- Realization Contract
  - The contract between a Component Specification and its implementation

Usage Contracts

- Describe the relationship between a service interface and its clients

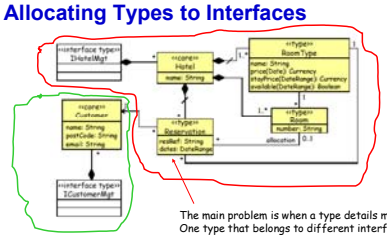
Using:

- An Interface concept – grouping a collection of related operations
- Each operation specified in terms of an
  - Operation signature
  - An Interface information model – things that need to remembered by an interface in order for an operation to fulfill its obligations
  - Preconditions
  - Postconditions
- Use Design by Contract Principles

Constructing an Interface Information Model

- Interface Information Model
  - A limited "view" of the underlying Business Type Model that is "Scoped" by an interface
- A first cut Interface Information Model is constructed by:
  - Following all the relationships from the Interface
  - Adding any additional data types that you constructed during Component Interaction Modelling
  - Add additional relationships to simplify operation specfication

Allocating Types to Interfaces

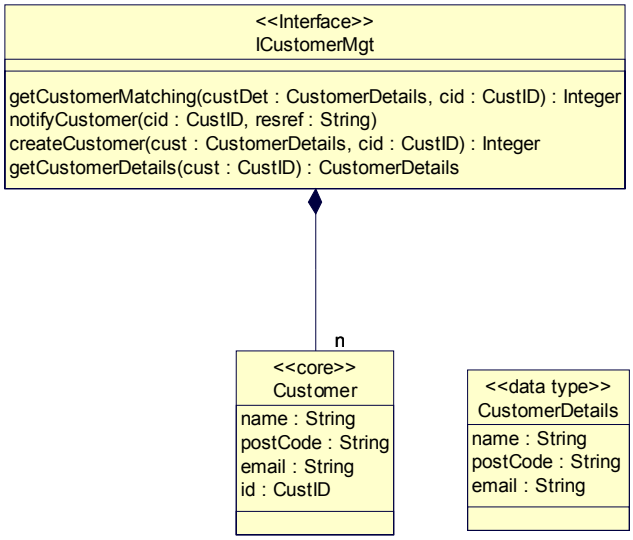


The main problem is when a type details more than One type that belongs to different interfaces.

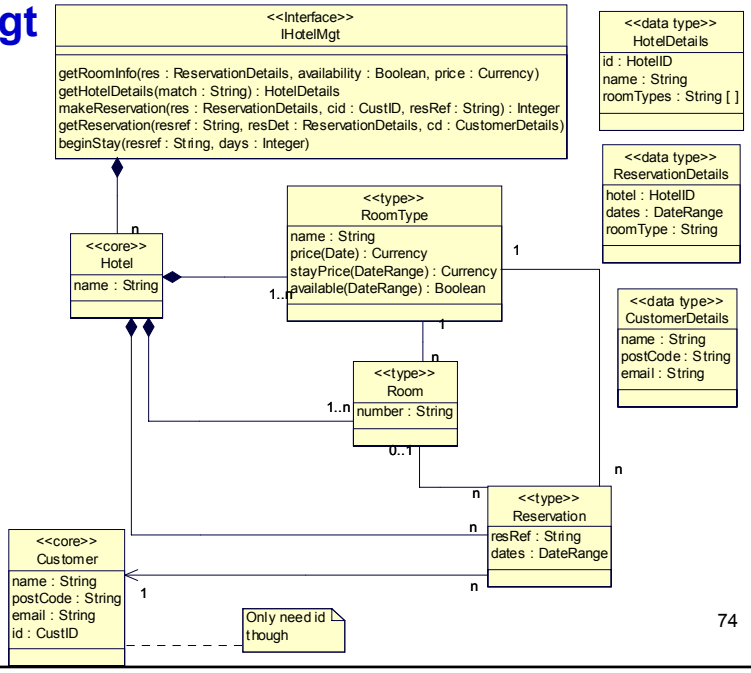
Here: Reservation is a detail of Room and Hotel.
It could go to either.
We choose to allocate it to the IHotelMgt Interface
Because the only information that Hotel needs to
Know about Customer is the customer id.

15

Interface Information Model for ICustomerMgt



Interface Information Model for IHotelMgt



UML Interface Definitions to WSDL specification

- We now have interface definition that:
  - Precisely describes operations and their parameters
  - A description of the information that the interface needs to remember (the types and their relationships)
- We need to produce the WSDL that will define the web service that implements the interface
  - Using a simple pattern
  - Using toolsets to help us do the work

Web Service Specification Pattern

- Problem
  - WSDL describes operations, parameters, their types and the wire protocol for transmission of data. Our interface design approach describes operations, parameters, their types and a more complete type model that represents all the information that the interface needs to remember. So when considering the design and implementation of the interface as a web service we need to consider how the interface information will be persisted.
- Solution
  - We will treat the Business web service as a Session. All the type model will be represented as a classes. Types specifically used to define complex data types will be implemented as Structs. The Business web service will manage instances of objects.
- Advantages and Disadvantages
  - The interface can be implemented exactly as modelled
  - Persistence of objects to a databases are the responsibility of the interface operations and/or helper classes
- Example Walkthrough

Using Dot NET and MS Visual Studio

- Visual Studio Project – Web Service
- Implementation of web service contains a mix of implementation code and specification code
- Web Service Specification
  - Service Directives
  - Method Directives
- Complex types used in parameters

Web Service Directives

```
namespace IHotelMgt
{
        /// <summary>
        [WebService(Namespace="http://walkthrough/HotelMgt/",
                Description="The Hotel Management service for reservations")]

        public class HotelMgtService : System.Web.Services.WebService
        {
}


[WebMethod(Description="This method returns the details of a room
availability for a reservation " +
                "when supplied with the hotel id")]
[SoapMethodAttribute()]

public void getRoomInfo(ReservationDetails res, int availability, float
currency)
{
…
}
```

Parameter Types and Interface Information Model

```
public struct ReservationDetails
{
        public int hotelID;
        public string startDate;
        public string endDate;
        public string roomType;
}
public struct CustomerDetails
{
        public string name;
        public string postCode;
        public string emailAddress;
}
```

- **Parameter Types**

- **Interface Information Model**

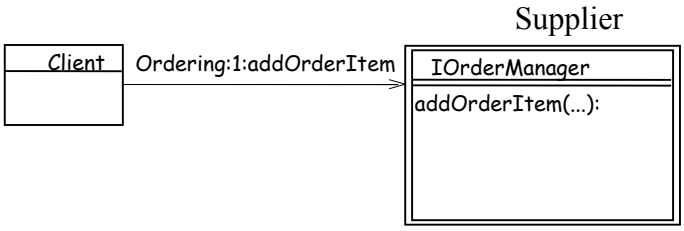
```
public class RoomType
{
        public string name;
        public int price;
}
public class Hotel
{
        public int hotelID;
        public string name;
        public Room[] roomset;
}
public class Room
{
        public string roomMum;
        public Hotel owningHotel;
        public RoomType rt;
}
```


The Resulting WSDL document

Design by Contract

- A contract between people is a written document that serves to clarify the terms of the relationship where both parties accept obligations and can found their rights.
- Contracts in software describe
  - the services (operations) that are provided by an interface (component)
  - the conditions under which the service will be provided
  - specification of the results of the service that is provided given that the conditions are fulfilled
- If either party fails to meet the conditions -
  - the contract is broken
  - it is clear who is responsible for breaking the contract

Contracts in Software



	Obligations	Benefits
Client	Client is obliged to call addOrderItem with an existing current order (Pre Condition)	Get current order updated with new order item
Supplier	Supplier is obliged to add new order item to the existing current order specified (Post Condition)	Simpler processing – thanks to the assumption that the order is current

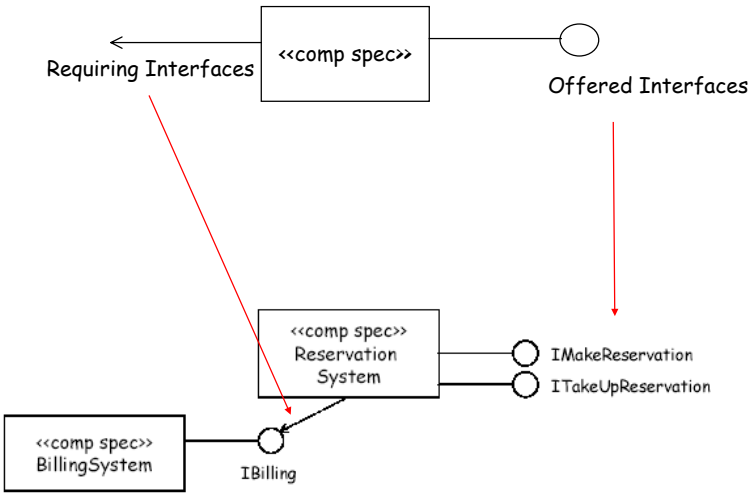
Part III: Advanced Concepts

- Types of component architecture
- Business Process Execution Language for Web Services
- UML 2.0

Component Specification Architecture

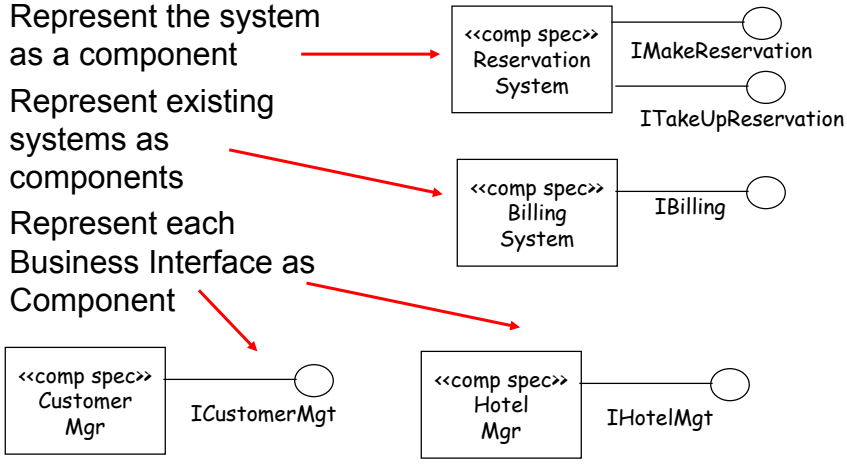
- In defining an initial component specification architecture we need to consider:
  - Interfaces identified as part of the component modelling process
  - Existing interfaces to systems
  - Architectural constraints

Component Spec Contracts

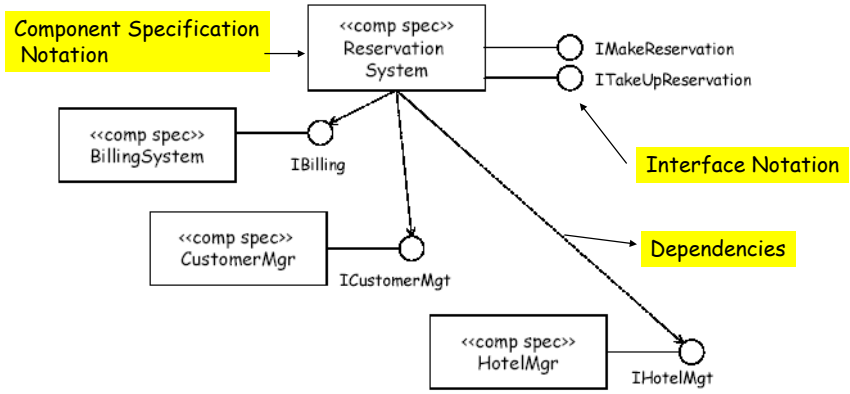


System Component Specification and others

- Represent the system as a component
- Represent existing systems as components
- Represent each Business Interface as Component



An Initial Architecture



Applications and Assemblies

- What is an Application?

  An application is the software that a user runs, that provides a set of useful, related services for that user or their organization and which the user perceives to be a single entity

- Examples?

Assembly

- What is an assembly?
- An assembly is the software that integrates (through calls) several pre-built components or software parts.
- The assembly may itself be a component if it offers programmable interfaces.
- The assembly may constitute an application if it is the software that a user runs and perceives to be a single entity…

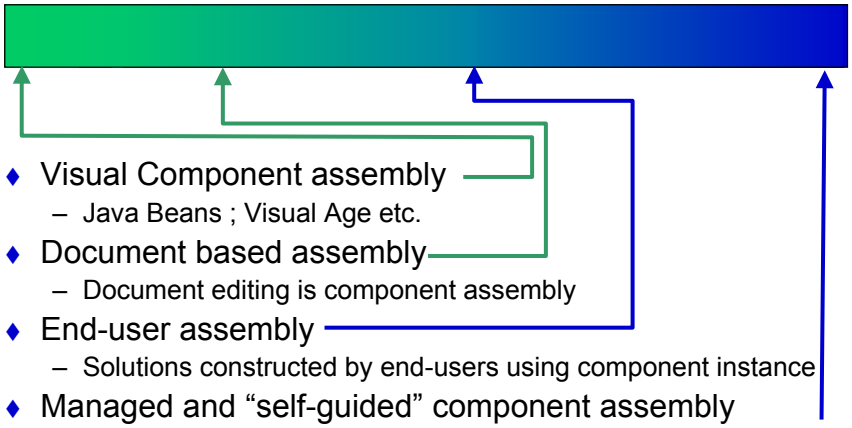
Assembly – at type or instance?

- Does application assembly happen at type level or instance level?
- Consider construction of GUIs using Visual Modelling tools:
  - Instance Level
  - Types are stored in a palette
  - Instances are dragged onto a development area
  - Connected using simple event driven scripting

Instance Level Assembly

- What are the implicit restrictions imposed by instance level assembly?
- Instances are predetermined at assembly time
- I.E. Major software benefit of being able to create new instances at will and any number is lost!
- If the same set of instances are used in many situations then we can construct template assemblies
- Check out www.groove.net

Assembly Styles



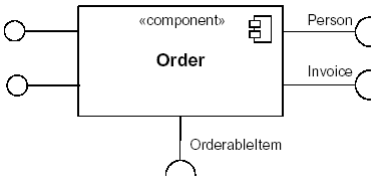
- Visual Component assembly
  - Java Beans ; Visual Age etc.
- Document based assembly
  - Document editing is component assembly
- End-user assembly
  - Solutions constructed by end-users using component instance
- Managed and “self-guided” component assembly
  - Intelligent agents using a rule base to interrogate servers to locate component instances to construct an assembly (Research area!)
  - Service based

Assembly Specification

- Architectural Description Languages
- Dependency wiring
- Port – Connector Modelling (Garlan and Shaw)
- BML
- Business Process Execution Language for Web Services
- **UML 2.0 Specification**

UML 2.0 Components - a single slide

- Components have "provided" and "required" interfaces
- Components have a realization concept
- Components can be assembled into larger grained software component assemblies or applications
  - Using Connectors (simple and advanced form)



UML 2.0 Specification – Component Spec Architecture

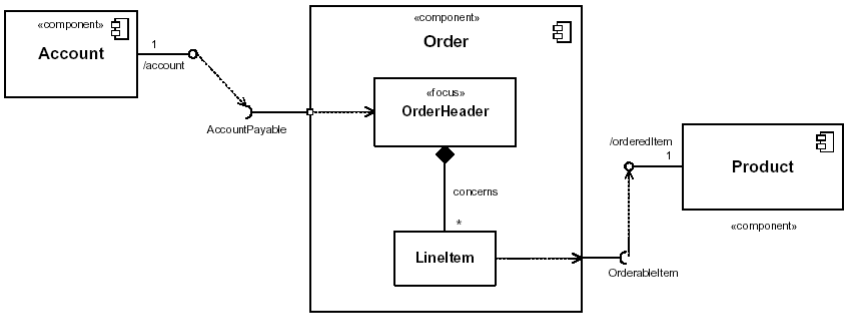


Figure 89 - Example of a platform independent model of a component, its provided and required interfaces, and wiring through dependencies on a structure diagram.

Component Assembly in UML 2.0

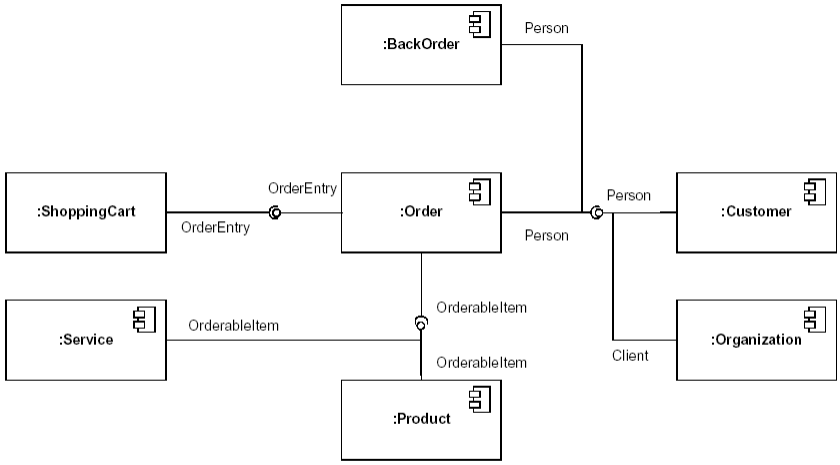


Figure 90 - Example of a composite structure of components, with connector wiring between provided and required interfaces of parts (Note: "Client" interface is a subtype of "Person").

Business Process Execution

- Web Services goal:
  - To achieve interoperability between applications using web services and a standard process integration model
- Complex systems integration
  - WSDL
    - a stateless model
    - Synchronous or uncorrelated asynchronous model
  - Business model interactions include:
    - Are Sequences of peer-peer communications
    - Synchronous or asynchronous
    - Within stateful, long-running interactions
  - So Platform independent Protocols are needed.

Business Process Execution Language for Web Services – BPEL4WS

- Draft Standard – part of w3.org
- Convergence of:
  - XLANG – Microsoft Biztalk Server
  - WSFL – Web Service Flow Language (IBM)
- Status:
  - 2nd Draft public release as BPEL4WS specification
  - Part of several Business Process Execution toolsets including
    - www.Collaxa.com

BPEL4WS in a nutshell

- Description of business protocols that is
  - Platform-independent
  - Captures all business behavioural aspects that have inter-enterprise business significance
  - That is executable
- Concepts
  - Support for data dependent behaviour
    - Use of conditional and time-out clauses
  - Specification of exceptional conditions, their effects and recovery sequences
  - Long running interactions (nested) and coordination of their units of work and at different levels of granularity
- XML based
- Layered on top of WSDL
- See refs at end.

Concluding Remarks

- The tutorial has formalised the relationship between component based development and web services.
- The mappings suggest that there is great opportunity to leverage CBD practice and apply them to web services.
- An example of such leverage has been demonstrated in the presentation of an adapted CBD method.

References

- Fremantle, P; Weerawarana, S; Khalaf, Rania. (2002). Enterprise Services. CACM Vol 45 (10).
- Kreger, H. (2003). Fulfilling the web services promise. CACM. Vol 46(10).
- Donato, C.,Durchlag, S and Hagel III, J. (2001). "Web services: enabling the collaborative enterprise". (http://e-serv.ebizq.net/wbs/donato_1a.html)
- McKeen, J.D and Smith H.A (2003). Making IT Happen. John Wiley and Sons, Chichester, UK.
- OMG (2003). UML 2.0 Specification.
- Cheesman, J., Daniels, J. (2001) *UML Components*; Addison-Wesley
- D'Souza, D., Cameron Wills, A. (1998) *Objects, Components and Frameworks with UML;* Addison-Wesley.
- Atkinson, C et al (2002). Component-based Product Line Engineering with UML; Addison Wesley.
- Szyperski, C. (1997) Component Software: Beyond Object-Oriented Programming, Addison Wesley
- Barn, B., Brown, A.W (1999). Enterprise-Scale CBD: Building Complex Computer Systems From Components; 9th International Conference on Software Technology and Engineering Practice (STEP'99), Pittsburgh, Pennsylvania, USA.
- Barn, B., Cheesman, J. and Brown, A.W (1998). Tools and Methods for Component Based Development. Proceedings of Tools-26, Santa Barbara, CA, USA, Aug 98.
- Yang, J. (2003) Web service componentization. CACM Vol. 46. No. 10.
- Meredith, L.J and Bjorg, S. (2003). Contracts and types. CACM Vol. 46. No. 10.
- Curbera, F, Khalaf, R., Mukhi, N., Tai, S and Weerawarana, S. (2003). The next step in web services. CACM Vol. 46. No. 10.
- W3C. (2001). Web Service Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315
- Business Process Execution Language for Web Services (2003). http://www-1-6.ibm.com/developerworks/webservices/library/ws-bpel
- Thai, T, Lam, Q. H. (2002). *.Net Framework Essentials*. O'Reilly, CA, USA.
- Cerami, E. (2002). *Web Services Essentials*. O'Reilly, CA, USA.
- Scribner, K, Stiver, M.C. (2002). *Applied Soap: Implementing .NET XML Web Services.* SAMS, USA.