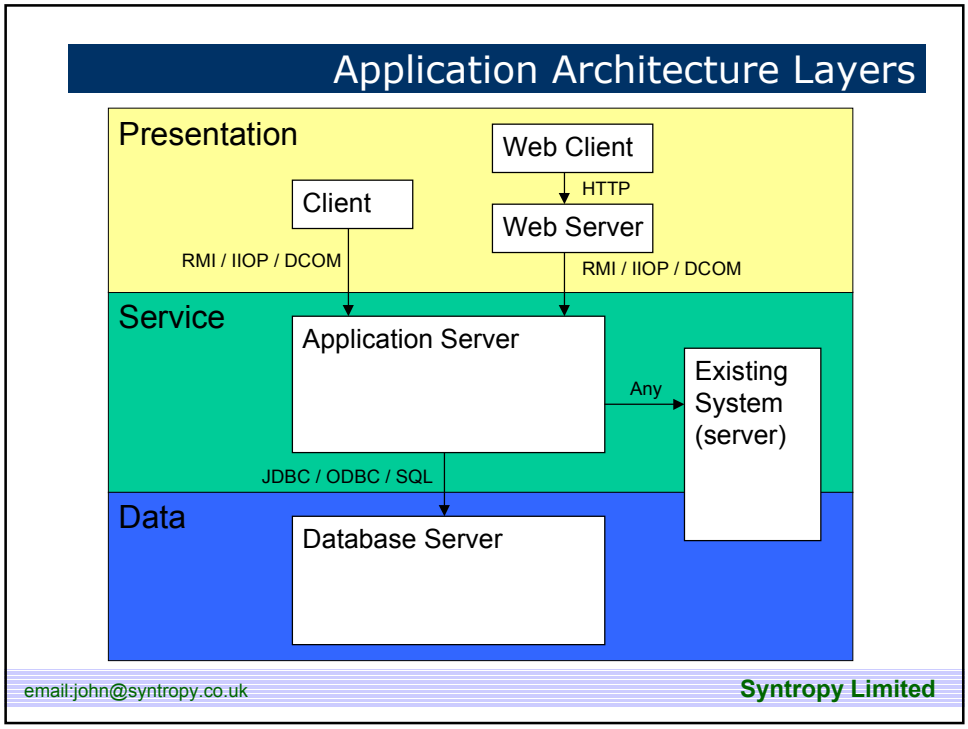
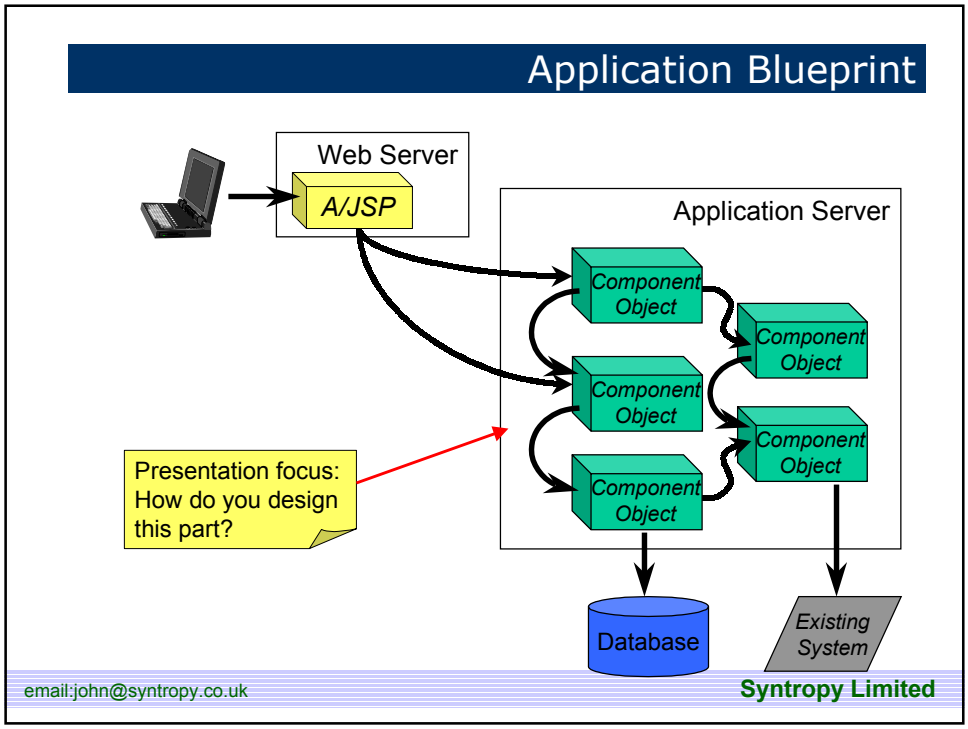
From Requirements to Components

John Daniels
Syntropy Limited
john@syntropy.co.uk

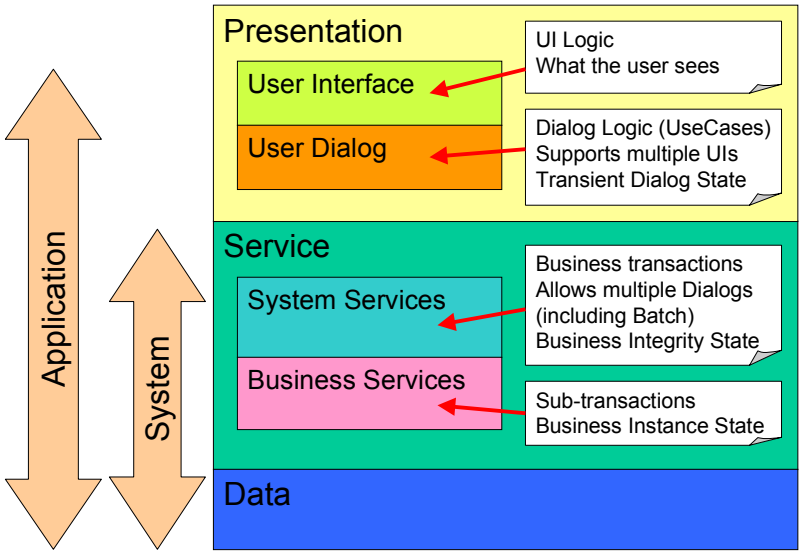
Agenda

- System architecture with components
- The Requirements workflow
- The Specification workflow
- Modelling components with UML





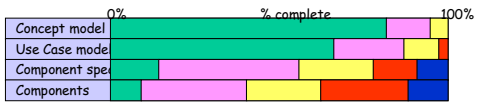
Finer-Grain Application Layers



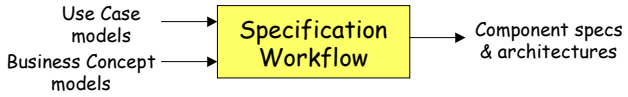
email:john@syntropy.co.uk

Management and Development Processes

- Management Processes
  - Schedule work and plan deliveries
  - Allocate resources
  - Monitor progress
  - Control risk

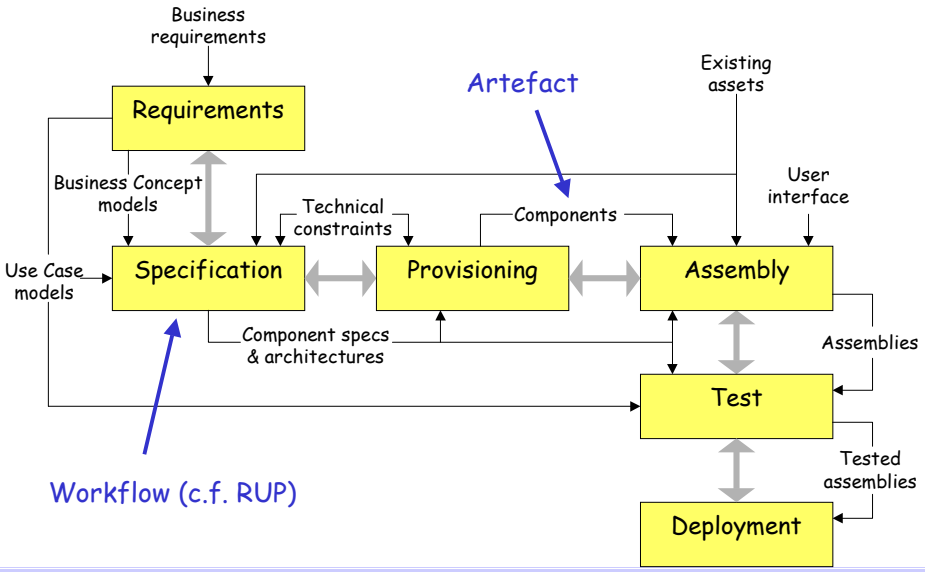


- Development Processes
  - Create working software from requirements
  - Focus on software development artifacts
  - Described independently of the management process
  - Defines ordering constraints and dependencies
  - Organized into Workflows

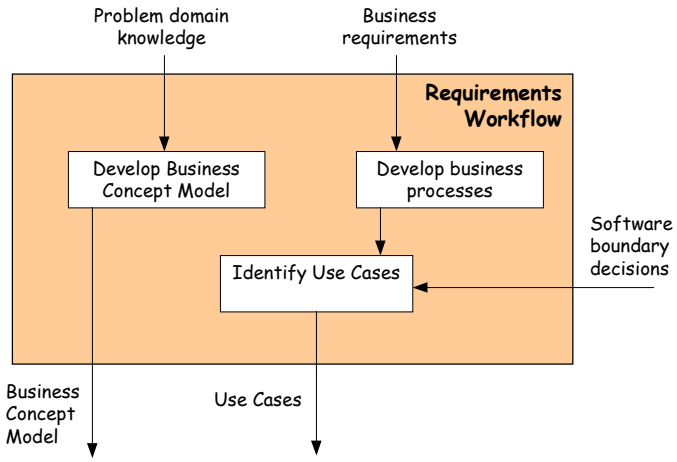


email:john@syntropy.co.uk

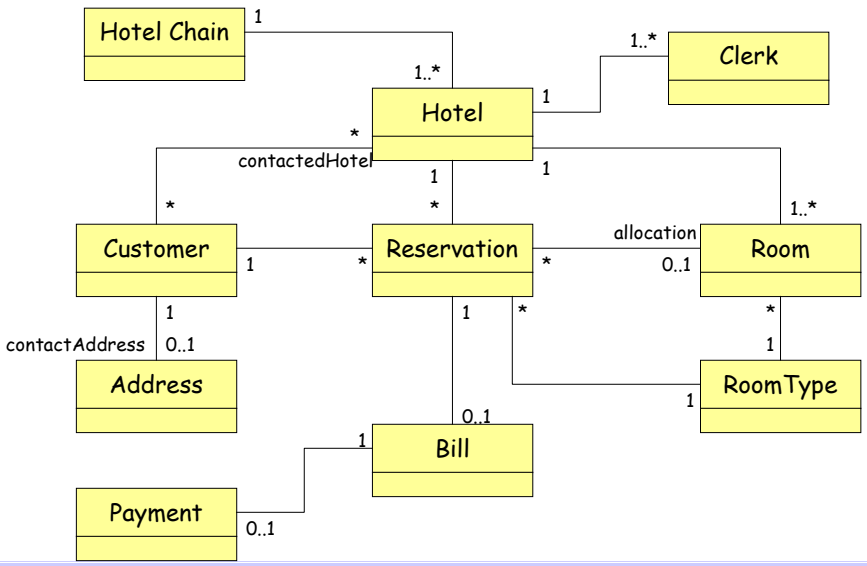
Workflows in the development process



The Requirements Workflow

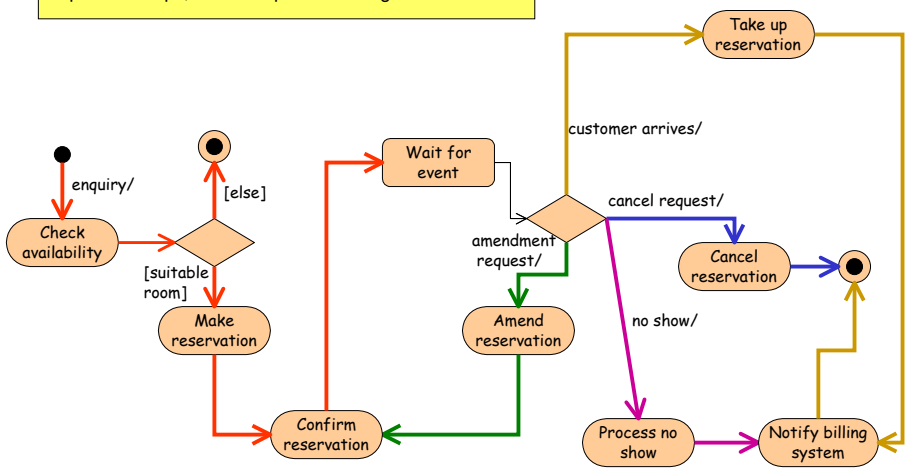


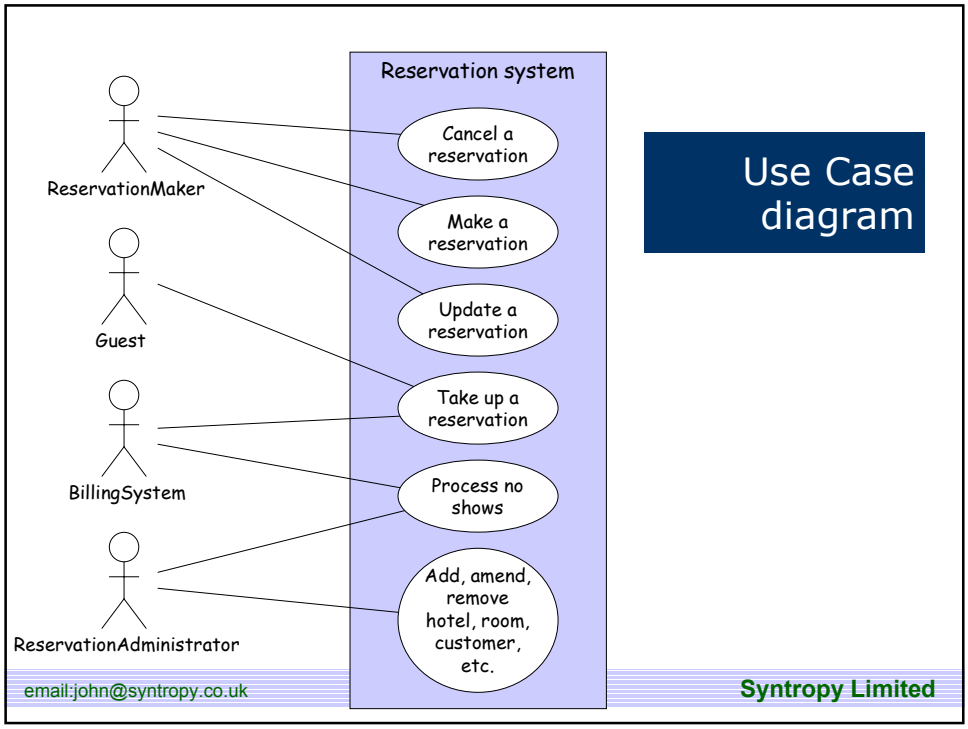
Business Concept Model



Identify Use Cases

A use case describes the interaction that follows from a single business event. Where an event triggers a number of process steps, all the steps form a single use case.





Name	Make a Reservation
Initiator	Reservation Maker
Goal	Reserve a room at a hotel

Steps or Extension Points

Main success scenario

1. Reservation Maker asks to make a reservation
2. Reservation Maker selects hotel, dates and room type
3. System provides availability and price
4. Reservation Maker agrees to proceed
5. Reservation Maker provides name and postcode
6. Reservation Maker provides contact email address
7. System makes reservation and gives it a tag
8. System reveals tag to Reservation Maker
9. System creates and sends confirmation by email

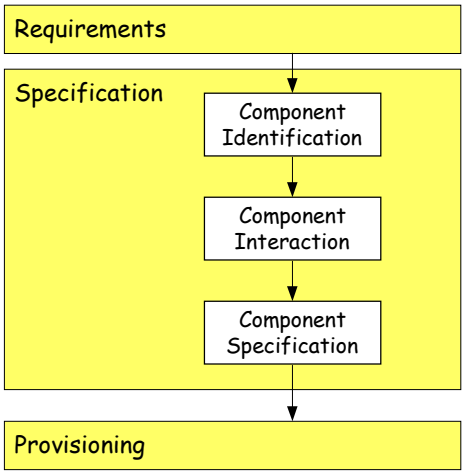
Extensions

3. Room Not Available
   a) System offers alternative dates and room types
   b) Reservation Maker selects from alternatives

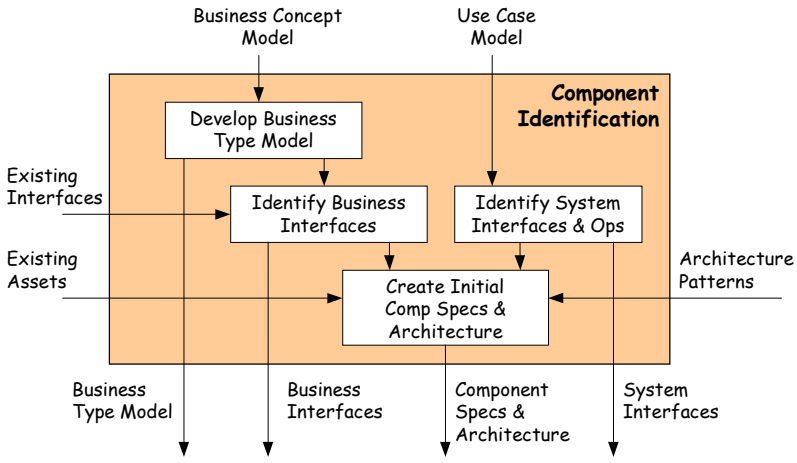
6. Customer already on file
   a) Resume 7

email:john@syntropy.co.uk Syntropy Limited

The Specification Workflow

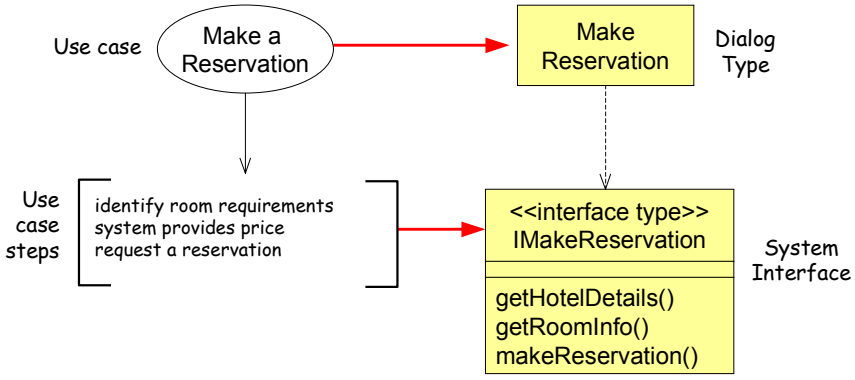


Component Identification

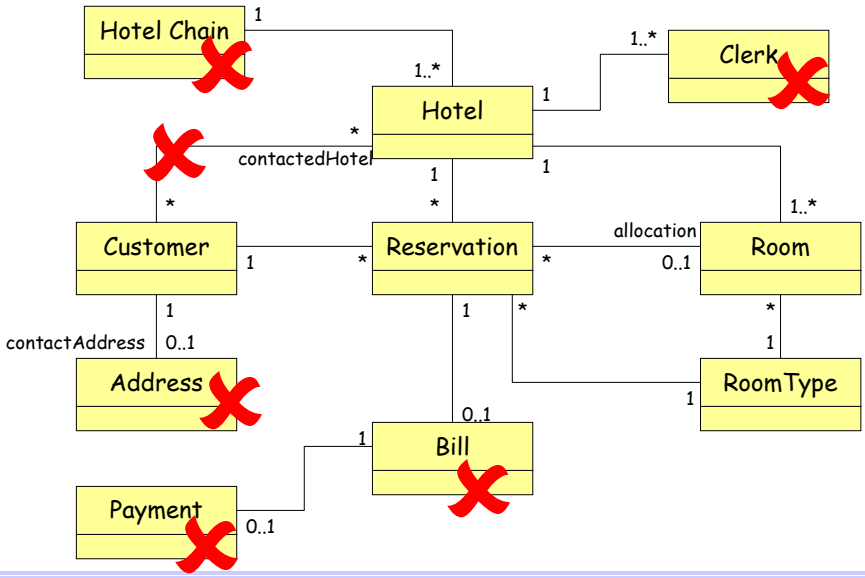


Identify System Interfaces and Operations

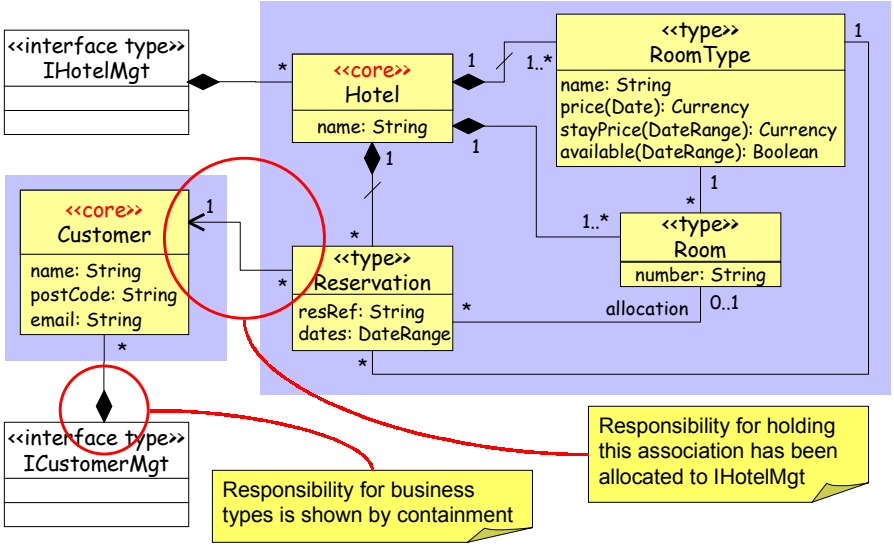
System interfaces act as facades - they are the point of contact for the UI and other external agents. They are supported by components in the system services layer.



Develop the Business Type Model

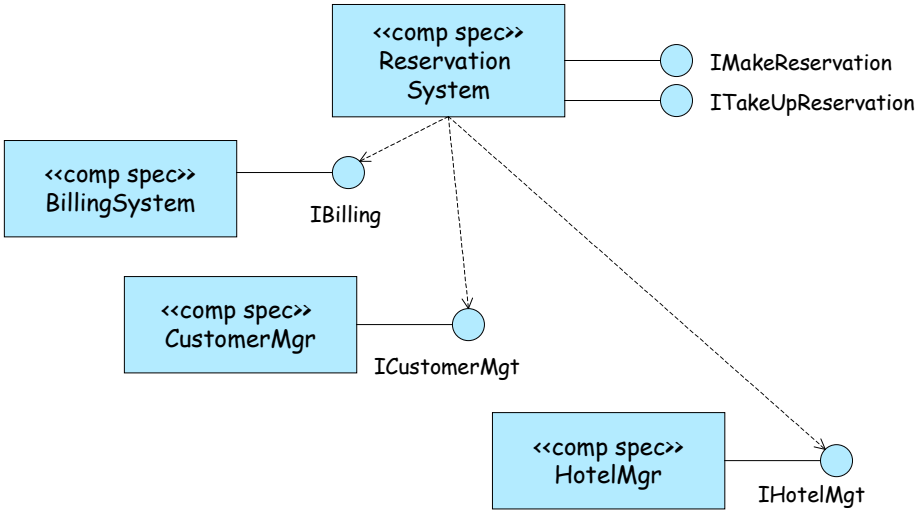


Identify business interfaces



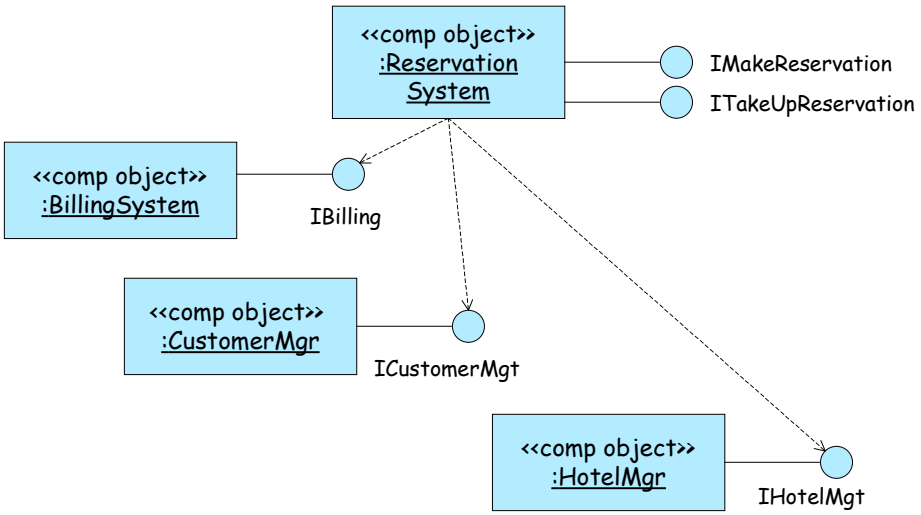
email:john@syntropy.co.uk
Syntropy Limited

Component architecture



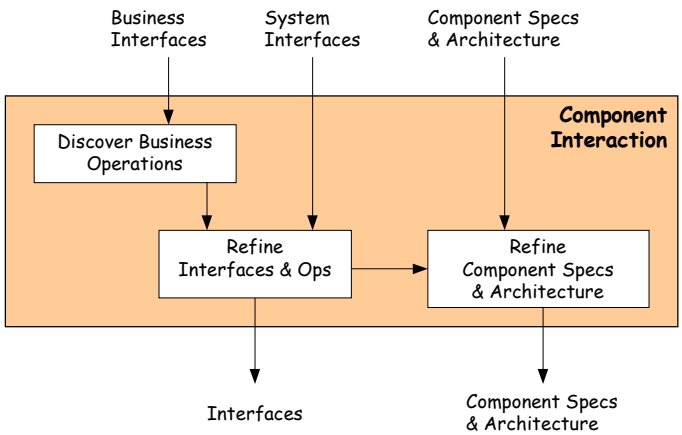
email:john@syntropy.co.uk
Syntropy Limited

Minimal component object architecture

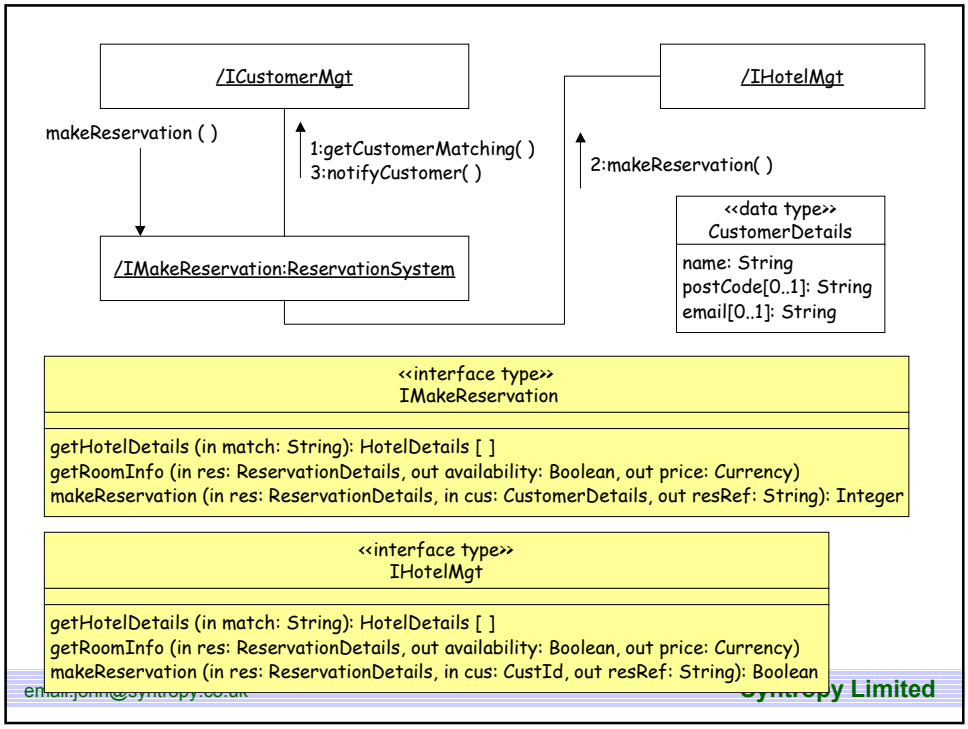


email:john@syntropy.co.uk

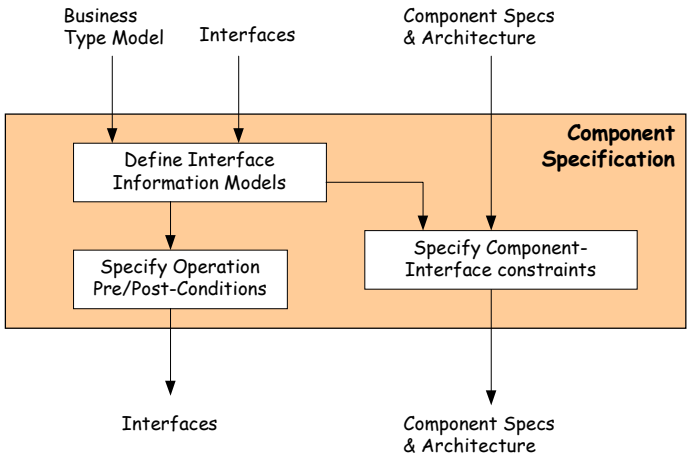
Component Interaction



email:john@syntropy.co.uk

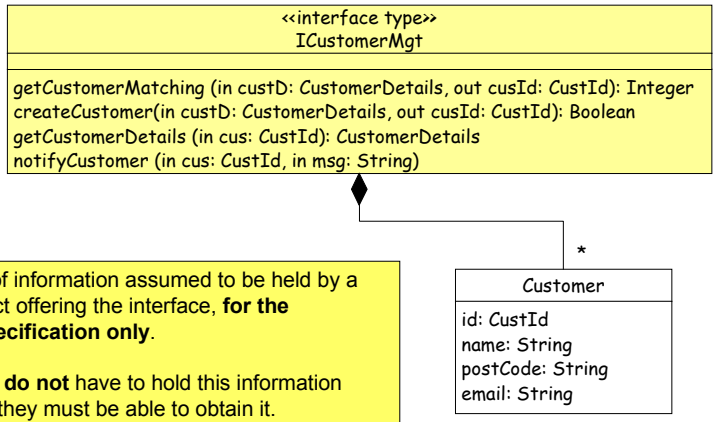


Component Specification



email:john@syntropy.co.uk

Interface information model



Defines the set of information assumed to be held by a component object offering the interface, **for the purposes of specification only**.

Implementations **do not** have to hold this information themselves, but they must be able to obtain it.

The model need only be sufficient to explain the effects of the operations.

The model can be derived from the Business Type Model.

Pre- and post-conditions

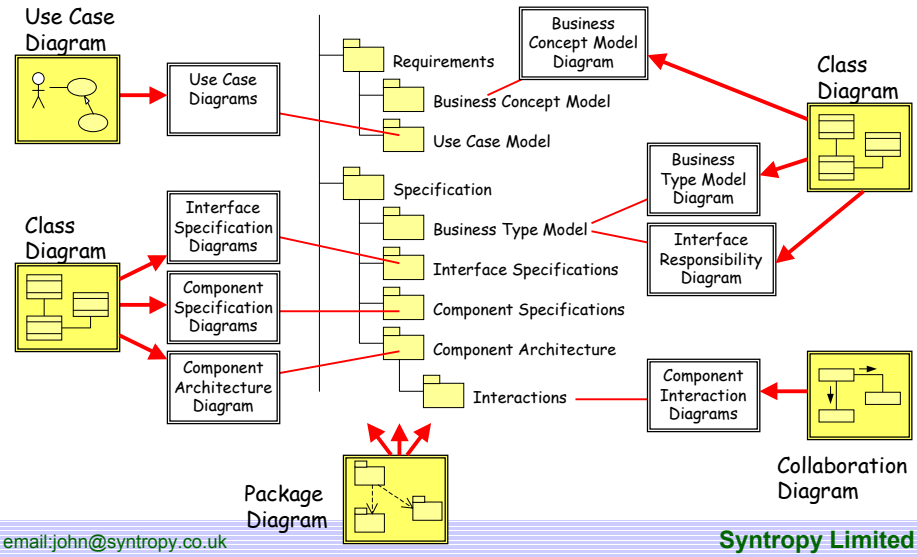
- If the pre-condition is true, the post-condition must be true
- If the pre-condition is false, the post-condition doesn't apply
- A missing pre-condition is assumed 'true'
- Pre- and post-conditions can be written in natural language or in a formal language such as OCL

```
context ICustomerMgt::getCustomerDetails (in cus: CustId): CustomerDetails

pre:
      -- cus is valid
      customer->exists(c | c.id = cus)

post:
      -- the details returned match those held for customer cus
      Let theCust = customer->select(c | c.id = cus) in
      result.name = theCust.name
      result.postCode = theCust.postCode
      result.email = theCust.email
```

UML diagrams



Want to know more?

- UML Components by John Cheesman and John Daniels, Addison-Wesley
- http://www.umlcomponents.com