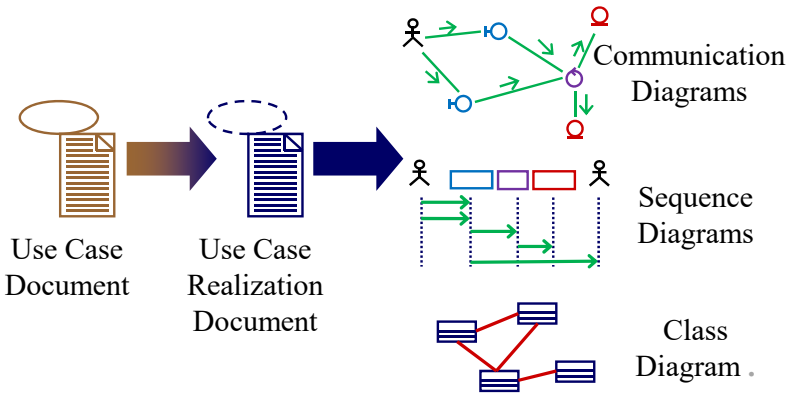
Object-Oriented Design

Prof. T.H. Tse
Department of Computer Science
Email: thtse@cs.hku.hk
URL: hku.hk/thtse.

Object-Oriented Design



Object-Oriented Design

- Categorize design classes into three types
- Describe the activities for object-oriented design
- Construct ***communication diagrams***, ***sequence diagrams***, ***state machines***, and ***class diagram***
- Identify ***activity diagrams*** and ***implementation diagrams*** .

Design Classes

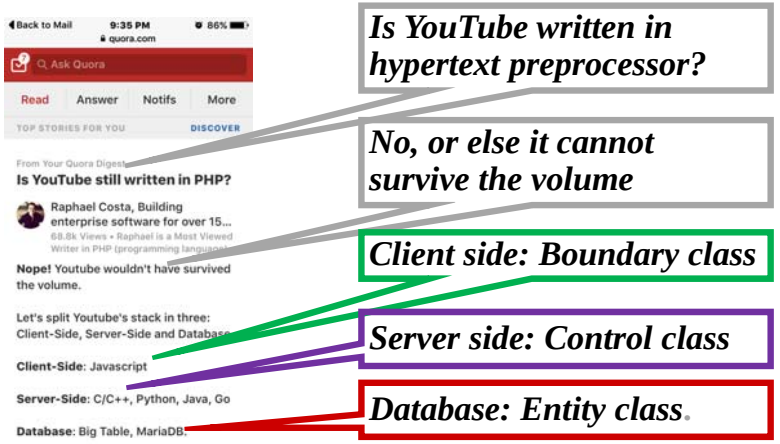
- Distribute system functionality across three types of classes
- Simplifies maintenance, revisions, and visualization of classes .

Design Classes

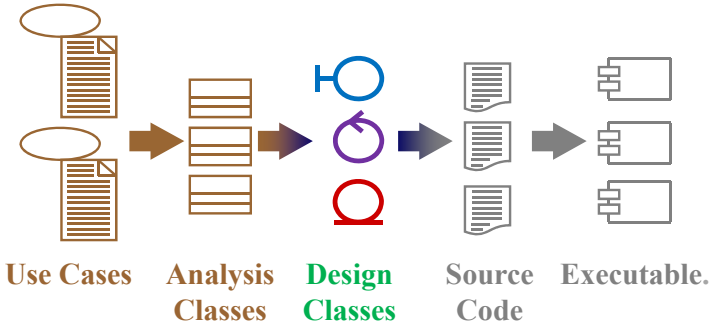
- ***Entity Classes***
  - Contain objects that ***store and manipulate*** actual data
- ***Boundary Classes***
  - Contain objects that represent ***interfaces*** with the actors
- ***Control Classes***
  - Contain objects that involve ***decisions in application processes*** .

Design Classes

Motivating Example

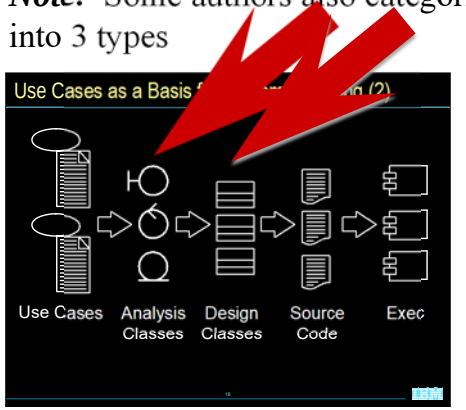


Design Classes



Design Classes

- ***Note:*** Some authors also categorize analysis classes into 3 types



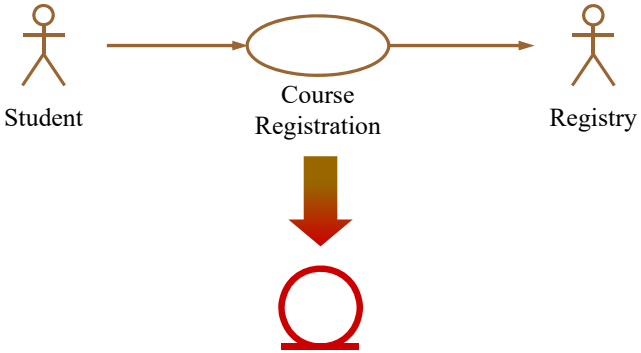
Design Classes

Entity Classes

- Usually contain objects that correspond to items in real life
- Encapsulate
  - attributes and
  - operations that update the attributes
- Entity objects are persistent
  - They continue to exist between use case executions because the attributes are typically stored in a database, allowing for manipulation and later retrieval .

Design Classes

Entity Classes



Design Classes

Boundary Classes

- Intermediate between the system and the external world
- May include user interface, system interface, and device interface
- Responsibilities include:
  - Translating user input into information that the system can understand and use in business events
  - Taking data pertaining to a business event and translating them for appropriate presentation to users
- At least one boundary class per actor / use case pair .

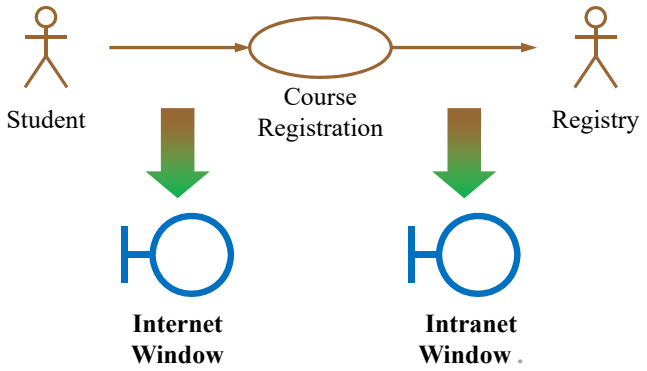
Boundary Class Example

The Future of Shopping

https://www.youtube.com/watch?v=CKnEtjpuEUg

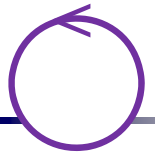
Design Classes

Boundary Classes



Design Classes

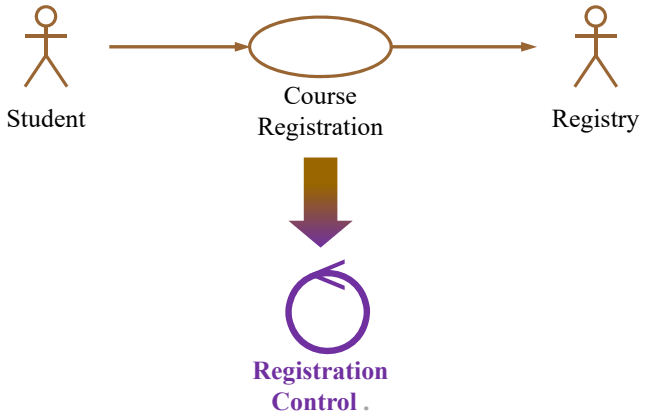
Control Classes



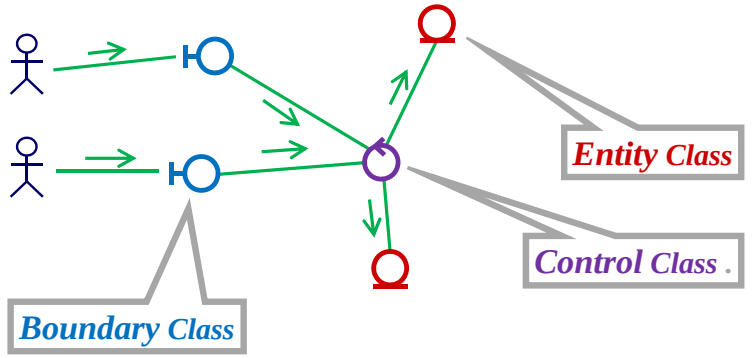
- Contain behaviour that does not naturally reside in entity or boundary classes
- Decouple ***boundary classes*** (front end) and ***entity classes*** (storage) from one another
- Contain the logic or rules of events for directing and managing the interactions among objects
- Allow the system to be more tolerant of changes and simplify maintenance
- Usually one control class per use case .

Design Classes

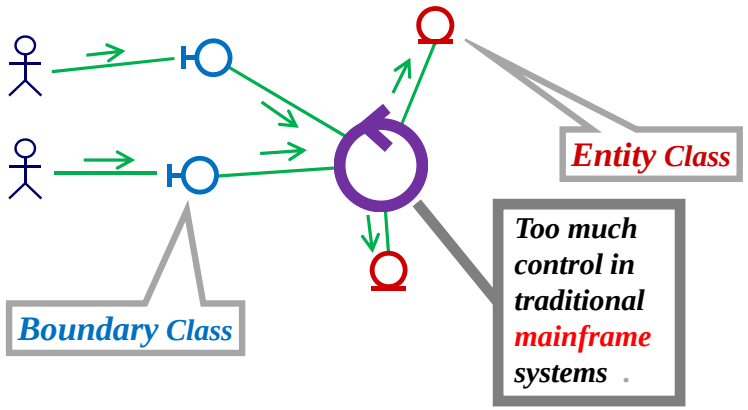
Control Classes



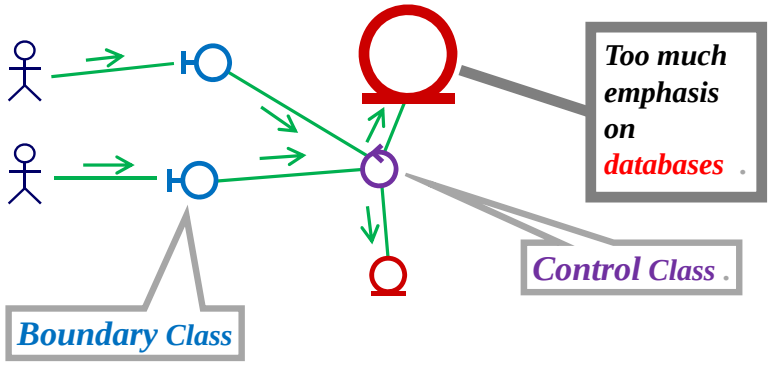
Communication Diagrams



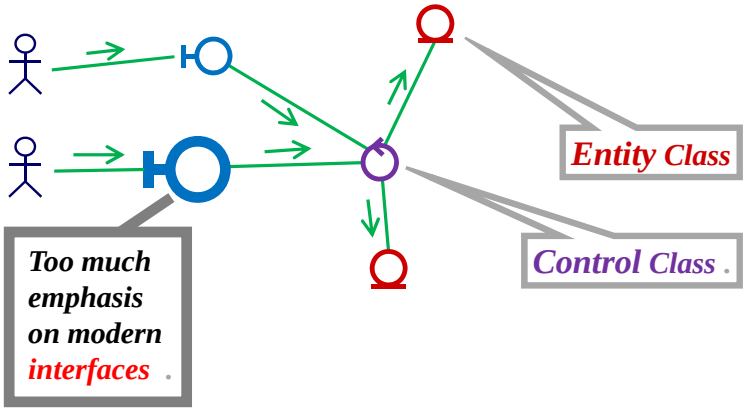
Division of Labour: Example 1



Division of Labour: Example 2



Division of Labour: Example 3



Object-Oriented Design Process

- Realize use case model to reflect implementation environment
- Model object interactions and behaviour that support use case scenarios
- Update class diagram to reflect implementation environment.

Object-Oriented Design Process

- ***Realize use case model to reflect implementation environment***
- Model object interactions and behavior that support use case scenarios
- Update class diagram to reflect implementation environment.

Realizing the Use Case Model
to Reflect the Implementation Environment

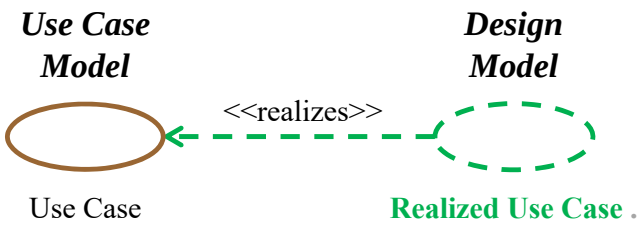
- Realize the use cases to include details of how the actor will actually interact with the system and how the system will respond
- This process is essential because
  - The details will be necessary for program implementation
  - The details will also be necessary for preparing user manuals and test scripts
  - They may help to discover new use cases .

Realizing the Use Case Model
to Reflect the Implementation Environment

Step 1. Transform "Analysis" Use Cases to "Design" Use Cases

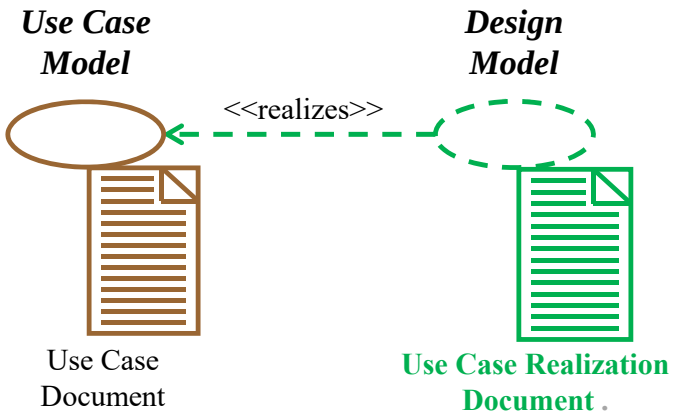
- Realize every analysis use case to reflect implementation environment
- Keep the ***analysis*** use cases separate from the ***design*** use cases, to allow reusing use cases for changes in implementation .

Realizing the Use Case Model
to Reflect the Implementation Environment



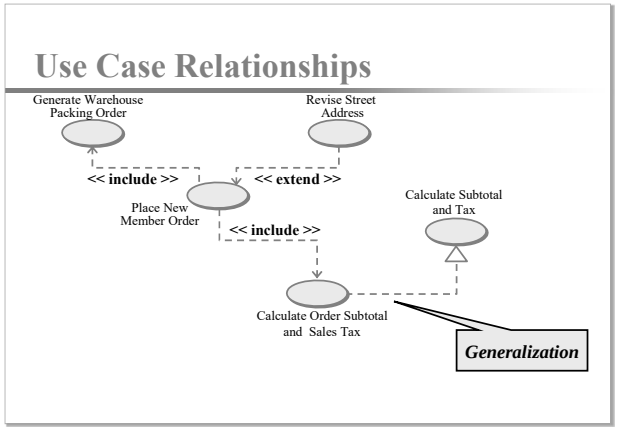
Realizing the Use Case Model

to Reflect the Implementation Environment



Realizing the Use Case Model

to Reflect the Implementation Environment



Analysis Use Case

Author: S. Shepherd Date: 01/07/2047

Use Case Name	Place Member Order	
Actor(s)	Member, warehouse	
Description	Describes the process of a member submitting an order for SoundStage products.	
Reference	MSS-1.0	
Normal Course of Events	**Actor Action** **Step 1**. Initiate this use case when a member submits an order. **Step 9**. Conclude this use case when the member receives the order confirmation notice.	**System Response** **Step 2**. Validate member's personal information such as address against what is currently on file. **Step 3**. Check member's credit status with accounts department system to ensure no outstanding payment. **Step 4**. For each product ordered, validate the product number, check availability in inventory, and record the ordered product information such as quantity ordered. **Step 5**. Calculate order subtotal and sales tax. **Step 6**. Verify member's credit card information based on the amount due. **Step 7**. Create packing slip for the member order containing all ordered products available and route it to warehouse. **Step 8**. Generate order confirmation notice indicating the status of the order and send it to the member.

Analysis Use Case

Alternative Courses	**Alt. Step 2A.** If member has indicated an address change in the order, update the member's record. **Alt. Step 3A.** If accounts department system returns a credit status that the member is in arrears, send order rejection notice to the member. **Alt. Step 4A.** If product number is invalid, send notification to member requesting them to resubmit product number. If the ordered product is not available, record the product information and mark as "back order". **Alt. Step 6.** If member's credit card information is invalid or if member is in arrears, send credit problem notice to member. Modify the order's status to "on hold pending payment".
Pre-Condition	Member has logged in.
Post-Condition	Member order has been recorded and packing slip has been routed to warehouse.
Assumption	None at this time.

Transform into Design Use Cases

Author: Pat Chan		**Date: 01/07/2047**
Use Case Name	Place Member Order	
Actor(s)	Member + Order Specialist	
Description	Describes the process of a member submitting an order for SoundStage products.	
Reference	MSS-1.1	
Typical Course of Events	**Actor Action**	**System Response**
	The main window is currently displayed on the screen waiting for the order specialist to select a menu option.	
	Step 1. Initiate this use case when an order is received from the member and the order specialist selects the option ⟨Place Member Order⟩.	**Step 2.** Display a dialogue window requesting Member No. be entered.
	Step 3. The order specialist enters Member No. and clicks ⟨OK⟩.	**Step 4.** Verify whether the Member No. is valid. If so, display a member dialogue window with the following information: Member Name, Member Status, Member Street Address, Member P.O. Box, Member City, Member State, Member Zip, Member Daytime Phone Number, and Member Email Address, along with a menu button labeled ⟨Update Member⟩. Also display two read-only windows containing order related fields (initially blank). The first window contains Order Number, Order Creation Date, Subtotal Cost, Sales Tax, and Order Status. The second window contains the individual ordered products. This is a multi-lined, scrollable window containing Product Number, Description, Quantity Available, Quantity Ordered, Quantity Backordered, Suggested Retail Price, Purchased Unit Price, Extended Cost, and Credits Earned.

Transform into Design Use Cases

Typical Course of Events	**Step 5.** The order specialist checks whether the member has made any address or phone number changes on the order. If changes have been made the order specialist clicks the ⟨Update Member⟩ menu button to invoke use case *Revise Street Address*. Otherwise, the order specialist clicks ⟨Enter Ordered Product⟩ menu button.	**Step 6.** Display a dialogue window prompting the user to enter the Product No.
	Step 7. The order specialist enters the Product No. of the item being ordered and then clicks ⟨Enter⟩.	**Step 8.** Validate Product No. and retrieve and display the Description, Credit Value, Quantity In Stock, and Suggested Retail Price. Then, advance the cursor to the Quantity Ordered field.
	Step 9. The order specialist enters the Quantity Ordered and a Purchase Unit Price if it differs from the Suggested Retail Price and clicks ⟨Enter⟩. If Purchase Unit Price is not entered it will default to the Suggested Retail Price.	**Step 10.** Validate the inputs and then calculate the new Extended Cost by multiplying the Purchase Unit Price by the Quantity Ordered. If the Quantity Ordered is greater than the Quantity in Stock, calculate Quantity Backordered by subtracting Quantity in Stock from Quantity Ordered. Display all fields in the ordered product read-only window.
		Step 11. Display a window with the message "Is the Ordered Product Information Correct?" If so, display another window with the message "Additional Products to be Entered?" Otherwise, go to Step 7.
	Step 12. If the order specialist clicks ⟨Yes⟩, go back to **Step 5**.	**Step 13.** Assign a unique Order No. and an Order Status of "O", and then calculate Order Subtotal and Order Sales Tax by invoking use case *Calculate Order Subtotal and Sales Tax*. Display all fields in the Order read-only window and then display a message "Commit Order?" with ⟨Yes⟩ and ⟨Cancel⟩ buttons.

Transform into Design Use Cases

Typical Course of Events	**Step 14.** The order specialist clicks ⟨Yes⟩.	**Step 15.** Reduce the Quantity in Stock by the Quantity Ordered for each product being ordered (replacing any negative result with zero) and then save all entries. Then, verify the member's status and creditworthiness by invoking use case *Verify Member Status*.
	Step 17. This use case concludes when the system displays a message "Order *N* has been processed successfully. Would you like to enter another one?" If the order specialist clicks ⟨Yes⟩ the use case is repeated. If the order specialist clicks ⟨No⟩, display the main system window.	**Step 16.** Generate a packing order by invoking use case *Generate Warehouse Packing Slip* and generate an order confirmation notice indicating the status of the order by invoking use case *Generate Order Confirmation Notice*. If there are invalid entries submitted by the member, invoke use case *Generate Order Error Report*.
Alternate Courses	**Alt. Step 4A.** If the Member No. is invalid, display a window with the error message "Member Number Not on File". The order specialist can then re-enter the number or cancel the transaction. **Alt. Step 8A.** If the Product No. is invalid, display a window with the error message "Product Number not Valid". The order specialist either corrects the entry or advances to the next entry. In a later step, generate an error report for any invalid entries. **Alt. Step 10A.** If the Quantity Ordered or Purchase Unit Price is invalid, highlight the field(s) in error and then display a window with the error message "Highlighted fields invalid". The order specialist either corrects the entry or advances to the next entry. In a later step, generate an error report for any invalid entries. **Alt. Step 13A.** If any of the products being ordered is not available, record the order status as "P" (for partially filled). **Alt. Step 16A.** If member's credit card information is invalid or if the member is found to be in arrears, send a credit problem notice to the member. Modify the order's status to "H" (for on-hold pending payment). Exit transaction.	
Precondition	Order specialist has logged on the system and has authorization for this transaction. The main window is currently displayed on the screen.	
Postcondition	Member order has been recorded, the Packing Order has been routed to the Warehouse, and a confirmation notice has been generated and sent to the member.	
Assumptions	None at this time.	

Example 1

Before Design

Step 2. Validate the member's personal information such as address against what is currently recorded on file .

Example 1

After Design

Step 2. Display a ***dialogue window*** requesting Member No. be entered

Step 3. The order specialist enters Member No. and ***clicks*** ⟨***OK***⟩

Example 1

After Design

Step 4. Verify whether Member No. is valid. If so, display a member dialogue window with the ***following information***: Member Name, Member Status, Member Street Address, Member PO Box, Member City, Member State, Member Zip, Member Daytime Phone Number, and Member Email Address, along with a ***menu button*** labeled ⟨Update Member⟩

Example 1

After Design

Step 4. ...

Also display two ***windows*** containing order-related fields (***initially blank***). The first window contains Order Number, Order Creation Date, Subtotal Cost, Sales Tax, and Order Status. The second window contains the individual ordered products. This is a ***multi-lined, scrollable window*** containing Product Number, Description, Quantity Available, Quantity Ordered, Quantity Backordered, Suggested Retail Price, Purchased Unit Price, Extended Cost, and Credits Earned

Example 1

After Design

Step 5. The order specialist checks to see if the member has made any address or phone number changes on the order. If changes have been made, the order specialist clicks the ⟨Update Member⟩ menu button to invoke use case *Revise Street Address*. Otherwise, the order specialist clicks ⟨Enter Ordered Product⟩ menu button.

Example 1

Use Precise Technical Terms

Traditional Mouse

- ***Click*** = press and release
- ***Double click*** = click and click
- ***Move*** = move pointer without pressing
- ***Drag*** = move pointer with pressing .

Example 1

Use Precise Technical Terms

Touchscreen

- ***Tap*** = touch and lift finger
- ***Double tap*** = tap and tap
- ***Pinch*** = touch the screen with 2 fingers and move them closer together
- ***Spread*** = touch the screen with 2 fingers and move them apart .

Example 2: Web Design

Before Design

Display the web page responsively

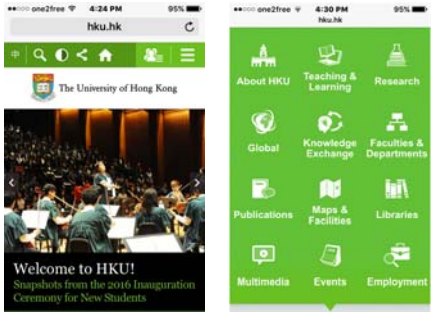
Fits desktop and mobile devices, all platforms, all screen sizes, and landscape and portrait modes .

After Design

Desktop Version

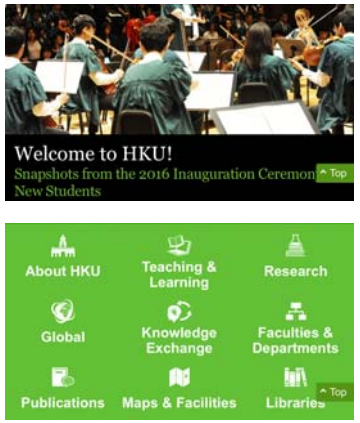
After Design

iPhone Portrait Version



After Design

iPhone Landscape Version



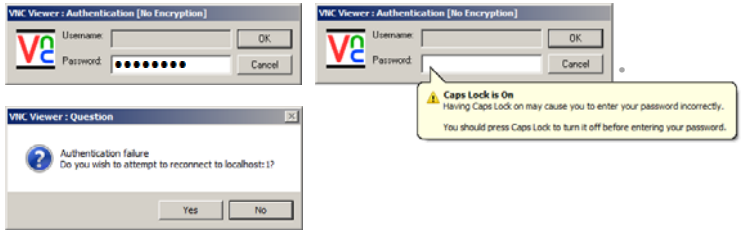
Example 3

Before Design

Enter user name and password .

Example 3

After Design



Example 4: Machine Setting Problems

After Design

Machine Setting Problem(s)

You cannot enter this application because of the following reason(s), please follow the instruction(s) to resolve the issue(s) and try again.

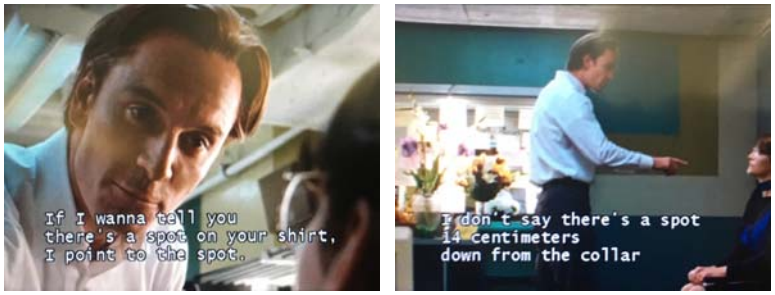
- A required version of Java Virtual Machine (JVM) is currently not enabled or supported by your browser. Please refer to http://www.gov.hk/en/about/helpdesk/softwarerequirement/onlineservice.htm for the required Java Runtime under different operating systems. You may also refer to FAQs about GovHK Online Services - Other Technical Questions and Trouble Shooting for further information.
  For assistance, please contact eTAX Help Desk.

The configuration of your computer and the configurations tested on eTAX are shown below:

	Current Configuration	Configurations tested on eTAX
OS Name	Windows	Windows, Linux, Mac
OS Version	Windows 7	Windows 8 & 7, Vista, XP, 2000, Mac OS X
Browser Name	Microsoft Internet Explorer	Microsoft Internet Explorer (Windows), Firefox, Safari (Mac), Google Chrome
Browser Version	10.0	Internet Explorer 6.0 - 10.0; Firefox 3.5 - 17.0; Safari 4 - 6; Google Chrome 13 - 23
Java Virtual Machine(JVM)	Unknown	Enabled
Java Runtime Environment(JRE) Vendor		Sun Microsystems / Oracle Corporation (Windows & Linux), Apple Computer Inc. / Apple Inc. (Mac)
Java Runtime Environment(JRE) Version		1.5, 1.6, 1.7
JavaScript	Enabled	Enabled
Cookies	Enabled	Enabled

Example 5: Graphical Interface

Design by Steve Jobs

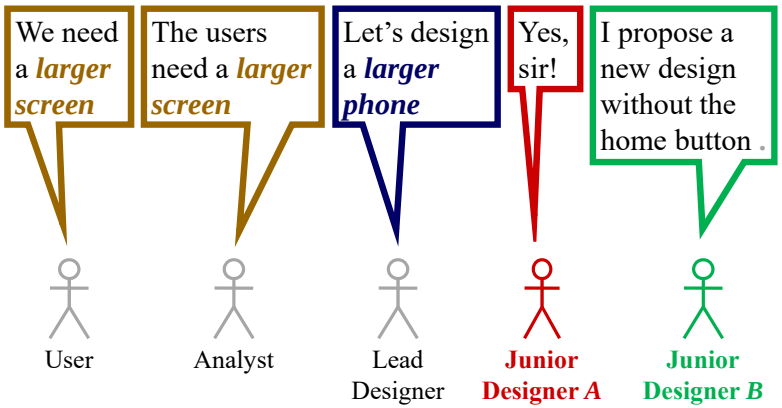


Example 6: The Future of Shopping

Design by Cisco

The Role of a Designer

iPhone Example



We Learn from Mistakes

Customer Profile

- ***User Requirement***
  - The system asks for a customer profile, including such entries as the date of birth, the annual household income, and the number of years of investment experience
- ***Analysis Use Case***

  Step 3. The system asks for a customer profile
- ***Design Use Case***

  Step 3. The system asks for a customer profile, which captures the Date of Birth, Annual Household Income, and No. of Years of Investment Experience .

Realizing the Use Case Model

to Reflect the Implementation Environment

Step 2. Update Use Case Diagram and other Documentation to Reflect any new Use Cases

- Update
  - the use case model diagram and
  - the actor and use case glossaries

  to reflect any new information introduced in step 1 .

Is the Design Final?

Example:

Rewrite the entire system?

How can object classes help? .

Object-Oriented Design Process

- Realize use case model to reflect implementation environment
- ***Model object interactions and behaviour that support use case scenarios***
- Update class diagram to reflect implementation environment .

Model Object Interactions and Behaviour
that Support Use Case Scenarios

- Identify and categorize the design classes required by the functionality specified in each use case, and identify object interactions, responsibilities, and behaviour .

Model Object Interactions and Behaviour
that Support Use Case Scenarios

Step 1. Identify and Categorize Design Classes

- Examine each design use case to identify and categorize the classes into 3 types (***boundary***, ***control***, and ***entity***)

Recall:

Boundary, Control, and Entity Classes

BOUNDARY CLASSES	CONTROL CLASSES	ENTITY CLASSES
Member Services Main Window	Order Processor	Member
Member Dialogue Window	Packing Slip Generator	Member Order
Order Processing Main Window		Product
Member Order Window		Product Ordered
Product Ordered Window		
Product Dialogue Window		
Message Window		
Warehouse Printer		

Model Object Interactions and Behaviour
that Support Use Case Scenarios

Step 2. Identify Attributes

- During analysis, some attributes may have been discovered
- Examine each use case for additional attributes that have not yet been identified, and update the class diagram to include such attributes.

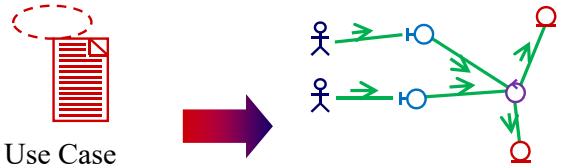
Model Object Interactions and Behaviour
that Support Use Case Scenarios

Step 3. Model High-Level Object Interactions

- For ***each use case***, create a ***communication diagram*** to model high-level interactions of design objects
- Include actors, boundary, control, and entity objects, as well as messages among objects .

Model Object Interactions and Behaviour
that Support Use Case Scenarios

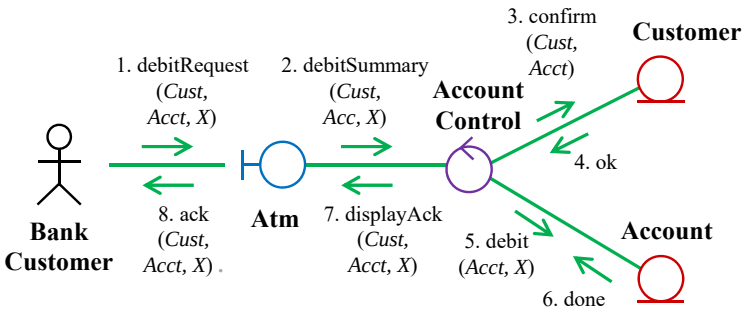
Step 3. Model High-Level Object Interactions



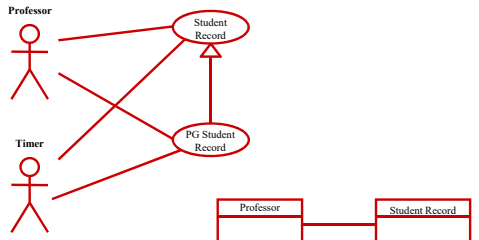
Use Case Realization Document

Communication Diagram .

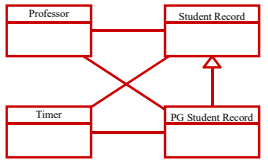
Communication Diagram
Example



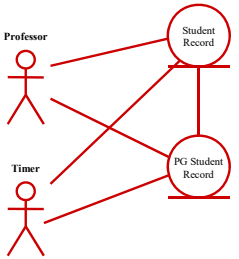
We Learn from Mistakes



Analysis Use Case Diagram



Class Diagram



Communication Diagram .

Model Object Interactions and Behaviour
that Support Use Case Scenarios

Step 4. Identify Behaviour

- Tasks include:
  - 4.1 Examine use case descriptions to identify behaviour
  - 4.2 Examine class diagram for additional behaviour
  - 4.3 Associate behaviour with class types
  - 4.4 Verify the results .

Identify Behaviour

Author: Pat Chan		**Date: 01/07/2047**
Use Case Name	Place Member Order	
Actor(s)	Member + Order Specialist	
Description	This use case describes the process of a member submitting an order for SoundStage products.	
Reference	MSS-1.1	
Typical Course of Events	**Actor Action**	**System Response**
	The main window is currently displayed on the screen waiting for the order specialist to select a menu option.	
	Step 1. Initiate this use case when an order is received from the member and the order specialist selects the option "Submit Member Order".	**Step 2.** Display a dialogue window requesting Member No. be entered.
	Step 3. The order specialist enters Member No. and clicks ⟨OK⟩.	**Step 4.** Verify whether the Member No. is valid. If so, display a member dialogue window with the following information: Member Name, Member Status, Member Street Address, Member P.O. Box, Member City, Member State, Member Zip, Member Daytime Phone Number, and Member Email Address, along with a menu button labeled ⟨Update Member⟩. Also display two read-only windows containing order related fields (initially blank). The first window contains Order Number, Order Creation Date, Subtotal Cost, Sales Tax, and Order Status. The second window contains the individual ordered products. This is a multi-lined, scrollable window containing Product Number, Description, Quantity Available, Quantity Ordered, Quantity Backordered, Suggested Retail Price, Purchased Unit Price, Extended Cost, and Credits Earned.

Model Object Interactions and Behaviour
that Support Use Case Scenarios

Step 4. Identify Behaviour

Task 4.1 Examine use case descriptions to identify behaviour

- ***For each use case description***, highlight all action verb phrases

They represent actor or system behaviour, which are potential methods .

Model Object Interactions and Behaviour
that Support Use Case Scenarios

Task 4.2 Examine class diagram for additional behaviour

- Use case descriptions may not reveal all behaviour
- ***Associate each class*** with **CRUD** behaviour:
  - **C**reate a new object
  - **R**ead attribute values of an object
  - **U**pdate attribute values of an object
  - **D**elete an object .

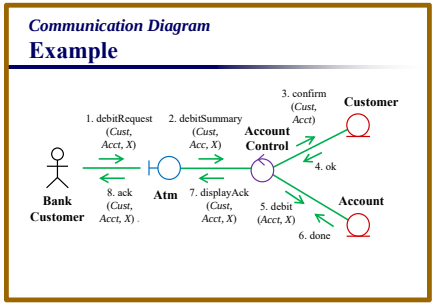
Model Object Interactions & Behaviour
that Support Use Case Scenarios

Task 4.2 (continued)

- Moreover, an object may not be able to do everything by itself
- Need collaboration by communicating with other objects
- Examine ***use case scenarios*** and ***communication diagrams*** to find interactions .

Model Object Interactions & Behaviour
that Support Use Case Scenarios

Task 4.2 (continued)



Model Object Interactions & Behaviour
that Support Use Case Scenarios

Task 4.2 (continued)

- Make full use of encapsulation
- *Example:* debit (amount)
  - Pass "`old balance`" and "`new balance`" between calling and called objects ***??***

    Implement the method in calling object ***??***
  - Pass message "`debit(amount)`" to called object

    Implement the method in called object .

Model Object Interactions & Behaviour
that Support Use Case Scenarios

Task 4.3 Associate behaviour with class types

- ***For each identified behaviour***, determine whether it is manual or automated
- If automated, associate it with appropriate class type .

Behaviour and Responsibilities

BEHAVIOUR	AUTOMATED/MANUAL	CLASS TYPE
Display dialogue window for Member No.	Automated	Boundary
Enter Member No.	Manual	
Click ⟨OK⟩	Manual	
Verify Member No.	Automated	Entity
Display club member dialogue window	Automated	Boundary
Display blank order window	Automated	Boundary
Display blank member ordered products window	Automated	Boundary
Check address and phone no. change	Manual	
Click ⟨Update Member⟩ menu button	Manual	
Click ⟨Enter Ordered Product⟩ menu button	Manual	
Display dialogue window for Product No.	Automated	Boundary
Enter Product No.	Manual	
Click ⟨Enter⟩	Manual	
Validate Product No.	Automated	Entity
Display product details	Automated	Boundary
Enter inputs	Manual	
Click ⟨Enter⟩	Manual	
Validate inputs	Automated	Entity
Calculate Extended Cost	Automated	Entity
Calculate Quantity Backordered	Automated	Entity
Compare Quantity Ordered with Quantity in Stock	Automated	Control
Calculate Quantity Backordered	Automated	Entity
Displays member ordered product window	Automated	Boundary
Display ordered product confirmation window	Automated	Boundary
Display dialogue window for additional products	Automated	Boundary
Assign unique Order No. and Order Status of "O"	Automated	Control
Display order window	Automated	Boundary

Condensed Behaviour List

BEHAVIOUR	CLASS TYPE
Verify Member No.	Entity
Validate Product No.	Entity
Validate inputs	Entity
Calculate Extended Cost	Entity
Calculate Quantity Backordered	Entity
Calculate Quantity Backordered	Entity
Display club member dialogue window	Boundary
Display blank order window	Boundary
Display blank member ordered products window	Boundary
Display dialogue window for Member No.	Boundary
Display dialogue window for Product No.	Boundary
Display product details	Boundary
Displays member ordered product window	Boundary
Display ordered product confirmation window	Boundary
Display dialogue window for additional products	Boundary
Display order window	Boundary
Compare Quantity Ordered with Quantity in Stock	Control
Assign unique Order No. and Order Status of "O"	Control

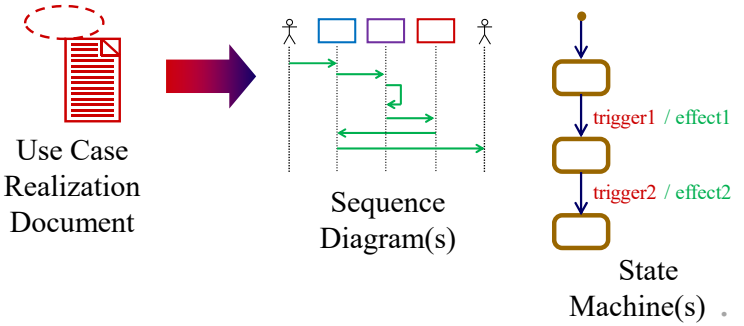
Model Object Interactions & Behaviour
that Support Use Case Scenarios

Task 4.4 Verify the results

- Walkthrough with relevant users
- Or, through role playing
  - Human participants play the roles of actors or objects
  - Simulate messages by passing items (such as a ball) among participants .

Model Object Interactions & Behaviour
that Support Use Case Scenarios

Step 5. Model Detailed Object Interactions

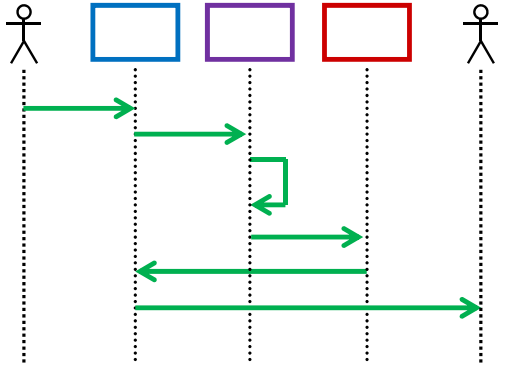


Model Object Interactions & Behaviour
that Support Use Case Scenarios

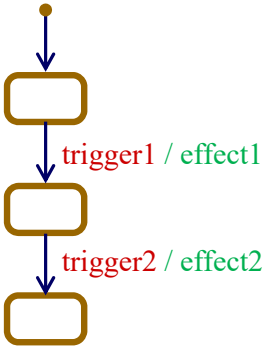
Step 5. Model Detailed Object Interactions

- For ***each use case***, construct ***sequence diagram(s)*** showing how the objects will interact to support the scenarios
- For ***each class***, construct a ***state machine*** .

Sequence Diagrams

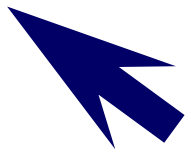


State Machines



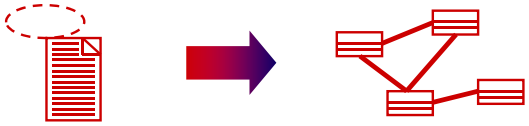
Object-Oriented Design Process

- Realize use case model to reflect implementation environment
- Model object interactions and behaviour that support use case scenarios
- ***Update class diagram to reflect implementation environment*** .



Update Class Diagram

to Reflect Implementation Environment



Use Case Realization Document

Class Diagram .

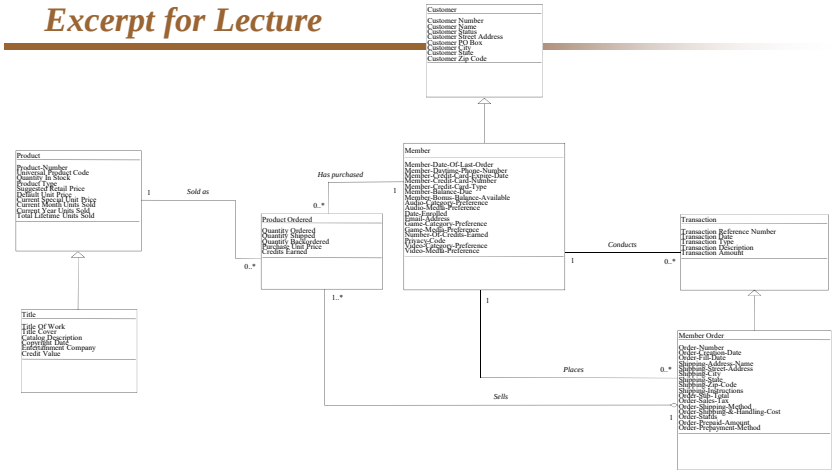
Update Class Diagram

to Reflect Implementation Environment

- Refine the class diagram to include behaviour or methods
  - Give a unique name to every behaviour or method
    - Normally the names reflect the naming convention in system implementation .

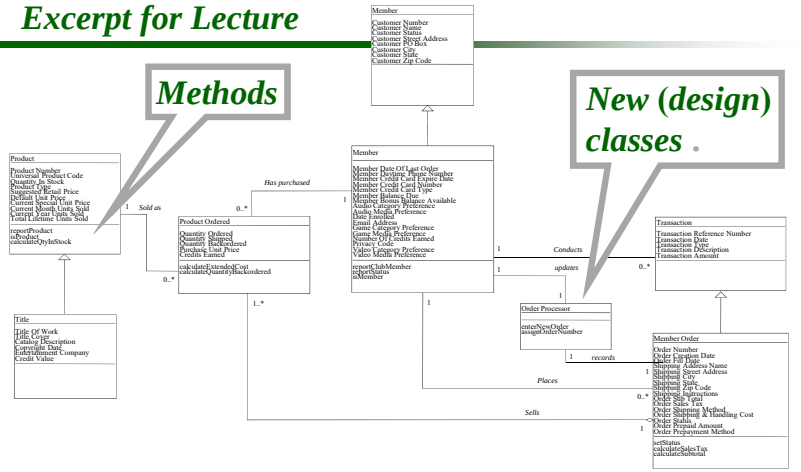
Analysis Class Diagram

Excerpt for Lecture

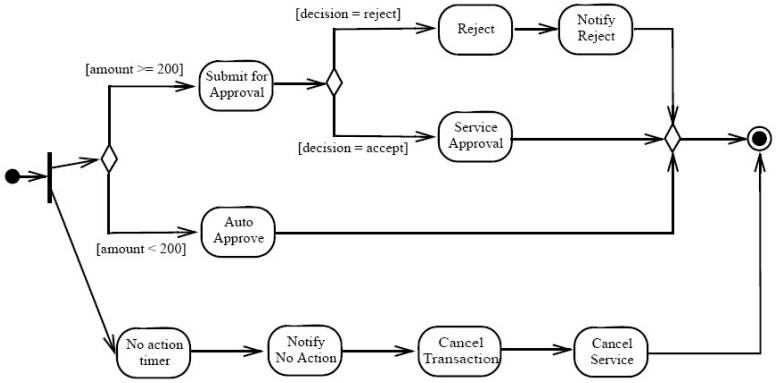


Design Class Diagram

Excerpt for Lecture



Activity Diagrams (Optional)



Component Diagrams