

OBJECT-ORIENTED MODELLING (OOM) - 32536

MODULE 6 - 6A

**Use Case Models:
Actors, use Cases and Documentation**

**Use case diagrams
(POOA – Chapter 4 & 5)**

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

1

Who is an Actor?

- The *User* of the system is usually the Actor
- The Actor (and *not* the User) is shown sending and receiving messages to and from the System
 - Example: John the Branch Manager, John the Customer and John the Teller may be one and the same *person*
- External Devices may also be Actors
 - e.g. ATMs, Keypads, Printers
- External Systems may also be Actors
 - e.g. A Mainframe

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

4

Module Outline


- Introduction to Actors
 - Primarily Users of the System
 - Actors as Roles
 - Actors as Devices and External Systems
 - Actor Documentation and Hierarchies
- Introduction to Use Cases
- Use case Variations:
 - Essential versus Concrete Use cases
- Behavioural modelling with Use Case Diagrams
- Relationships in Use case diagrams

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

2

So, An Actor is..

- Any 'thing' that Triggers an Interaction (from Outside the System)
- Any 'thing' that Receives a Message (Outside the System)
- Any 'thing' that is Outside the System



This is the notation for an Actor

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

5

Sub-Module

Actors

Identifying and Documenting

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

3

Finding Actors

- Who provides/uses Information to/from the System?
- Who Supports this Functionality?
- Which other Systems will this System Interact with?
- The External Devices (Keypads, Printers etc.) that the system will Interact with

Process Comment: Produce first cut of the list of Actors, then Cull them after drawing Initial Use Case diagrams

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

6

Actor: Variations

• Primary versus Secondary Actors:

Primary: - The first or main Actor who uses the system (e.g. Teller)
 - The main Actor who benefits from the system (e.g. Customer)

Secondary: - The Actor who derives indirect benefits from or uses the system
 (e.g. Branch manager); Depends on Perspective

• Direct versus Indirect Actors:

Direct: - The ones who actually use the system benefits from the system (e.g. Teller)

Indirect: - The ones who benefits from the system but never get to use the system directly (e.g. Customer)

• Abstract versus Concrete Actors:

- Due to Generalization to reduce Complexity

"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

7

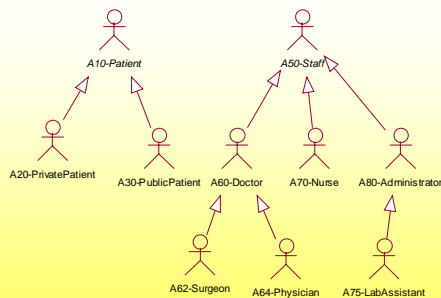
Documenting Actors (Only for Important / Complex Actors)

- Actor Thumbnail
 - <Name and Optionally Number for the Actor>
- Actor Type, Stereotype, Package
 - <Actor Type – Primary or Secondary; Human or Device etc; Stereotype and Package may optionally be added>
- Actor Description
 - <A line or two providing more details of the Actor>
- Actor Relationship
 - <With Use cases; Other Actors, esp. when Inheriting; >
- Interface Specifications
 - <Name of the Interface>
- Author & History
 - <Original Author and Modifiers of this Actor Documentation>
- Reference Material
 - <Relevant reference materials and sources for details>

"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

10

Abstract versus Concrete Actors and a corresponding Actor hierarchy in Hospital Management System



"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

8

Sub-Module

Use Cases



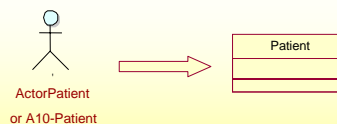
UC10

Identifying and Documenting

"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

11

Actor versus Class Confusion



"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

9

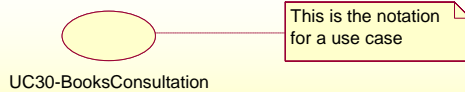
What is a Use Case?

- A series of interaction of an Actor with the System, which provides some concrete, measurable results to the User
- Documentation of that interaction as a series of steps - which may be atomic
- Pre- and Post-conditions, Context and other details related to the interaction between "Actor" and the system
- Has a strong Functional flavour and may be (has been) used for Non-OO, and even Non-IT purposes

"Copyright Practical OO Analysis,
 Thomson Publishing, 2005".

12

Notation for a Use case



UC30-BooksConsultation

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

13

Use Case Documentation

- The Flow within a Use case is the Behavioral aspect of the System
- This flow can be documented by either:
 - Organized Text (almost Mandatory) that describes the interactions of the Actor and the System
 - Activity diagrams that depict the flow of a use case
 - Sequence Diagrams (for selected Sequences - instances)
 - Pseudocode (if you want to use Object Constraint Language)
- A pre-determined Template (usually in Word) is a good starting point for documenting Organized Text of Use Cases

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

16

Sources of Use Cases

- User Workshops
 - Following the Joint Application Design style
 - Play acting various Scenarios with Users
 - Critical Requirements Analysis
- Formal and Informal Problem Statements
- Knowledge of Domain Experts
- Interviews and Discussions
- Existing Systems (esp. Legacy)
- Use Case Patterns/Published Literature/ URLs

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

14

Template: Documenting Use Cases

- Use Case Thumbnail
 - <Number and Name of the Use Case>
- Use Case Description (Goal)
 - <A short description of the use case; Goal or main purpose of the use case may also be written here>
- Stereotype and Package
 - <Description of the Stereotype and Package to which this use case belongs>
- Pre-Conditions
 - <Conditions that must be satisfied before this use case can begin - may include a list of other use cases>
- Post-Conditions
 - <Conditions that must be met at the end of this use case>
- Actors
 - <A list of the actors involved in this use case>
- Use Case Relationships
 - <Thumbnails of other Use Cases that are Included, Extended and Inherited>

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

17

Essential Use Case

User Action	System Response
Insert Card	Read stripe; Request PIN
Choose Withdraw	Ask Withdraw/Deposit/Enquiry
Take Cash	Dispense Cash

User Intention	System Responsibility
Identify Self	Verify Identity
Choose	Offer Choices
Transact	Provide for Choice

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

15

Template: Documenting Use Cases...

- Basic Course (Use Case Text)*
 - 1.0 <description of step>
 - 2.0 <description of step> (A1, E1, E2)
 - 3.0 <description of step> (A2, E3)
 - INCLUDE <Thumbnail of Use case/s Included>
 - EXTEND <Thumbnail of Use case/s Extended>
- Alternative Courses
 - <A1>
 - <A2>
- Exceptions
 - <E1>
 - <E2>
- Constraints
 - <special constraints/limitations in executing the use case>

*The use case text may also be written in an Action-Response or Intention-Responsibility format

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

18

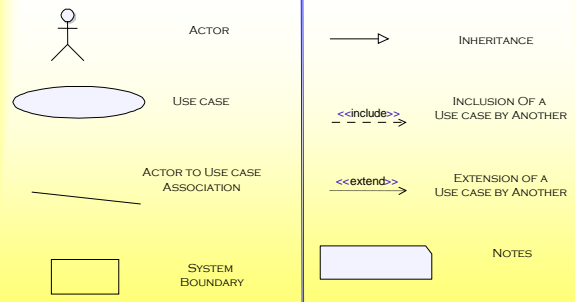
Template: Documenting Use Cases...

- User Interface Specifications
 - <Number and Name of UI specifications related with this use case, including web-screen specifications>
- Metrics
 - <Measurements related to this use case - if relevant to the programme; Also, perhaps, complexity factor>
- Priority
 - <The importance of the functionality described by this use case High/Medium/Low>
- Status
 - <The state of completeness of the documentation of this use case - Initial/Major/Final>
- Author & History
 - <Original Author and Modifiers of this Use Case>
- Reference Material
 - <Relevant reference materials and sources for details>

"Copyright Practical OO Analysis, Thomson Publishing, 2005".

19

Major Ingredients of a Use case Diagram



"Copyright Practical OO Analysis, Thomson Publishing, 2005".

22

SWOT of Use Case Diagrams

Strengths

1. Models the Important user as Actor
2. Organizes Requirements
3. Facilitates communication
4. Documents all Functionality
5. Reuses and extends requirements
4. Facilitates Traceability of Reqs.
5. Provides System Context/Boundary

Weaknesses

1. Not object-oriented
2. Have no documentation standards
3. Imprecise Relationships (arrowhead, stereotype)
4. Granularity is subjective
5. Has no Flow

Objectives

1. Visualize how system is used
2. Scope and Prioritize requirements
3. Facilitate project estimation
4. Facilitate contract management
5. Starting point for other diagrams

Traps

1. Can't express Non-functional req.
2. Ignoring internal use case doc.
3. Coding directly from use cases
4. Pedantic usage, in other modelling spaces
5. Use case driven project plans

"Copyright Practical OO Analysis, Thomson Publishing, 2005".

20

What is a Use Case Diagram?

- Primarily used to *visualize* the Use cases, corresponding Actors, and their interactions
- Notations: Actor, Use case, Relations
- Relations:
 - Relations in a Use case diagram are powerful mechanism to *organize* and *reuse* Requirements
 - Include; Extend; Generalize (Inherits)
- Context in a Use case diagram indicates the boundary of the system (between the Actor and the System)

"Copyright Practical OO Analysis, Thomson Publishing, 2005".

23

Sub-Module (6-A)

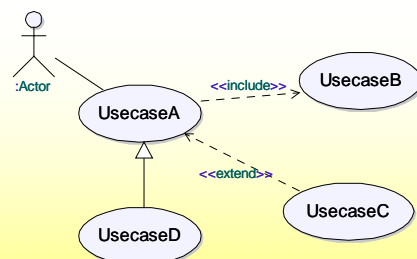
Use Case Diagrams

Behavioural modelling with Use Case Diagrams
Relationships in Use case diagrams

"Copyright Practical OO Analysis, Thomson Publishing, 2005".

21

UML Notation for Use Case Relationships



Use case A <<include>> Use case B
Use case C <<extend>> Use case A
Use case D <<inherit>> Use case A

Practical Warning:
Detailed discussion on Include versus Extends will waste time.

Practical Tip:
Use cases DON'T crash systems

"Copyright Practical OO Analysis, Thomson Publishing, 2005".

24

Include Relationship

- When Requirements tend to Exhibit Common behaviour, they should be Factored out
- These Common Behaviours form a Use case of their Own
- They are then “included” in the Original Use cases, and many more
- *Tip: Included use case is usually Mandatory; However, an If-Then-Else situation can't be easily Shown (add Notes)*

© Copyright Practical OO Analysis, Thomson Publishing, 2005.

25

Sub-Module

Putting Together a Use Case Diagram

Visual Representation of Requirements through Actors, Use cases and Relationships

© Copyright Practical OO Analysis, Thomson Publishing, 2005.

28

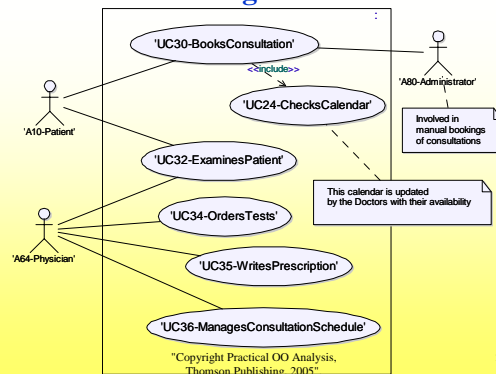
Extend Relationship

- Augments an existing Use case
- The Extended use case describes one set of interactions that augment the base use case
- Enables augmenting the Use case diagrams for new and unusual Requirements without disturbing Existing diagrams
- *Tip: Extends implies augmentation of a use case into one or more 'extended' use cases; Usually it's Optional execution of one of the Extended Use Case*

© Copyright Practical OO Analysis, Thomson Publishing, 2005.

26

“Consultation Details” Use Case Diagram



© Copyright Practical OO Analysis, Thomson Publishing, 2005.

29

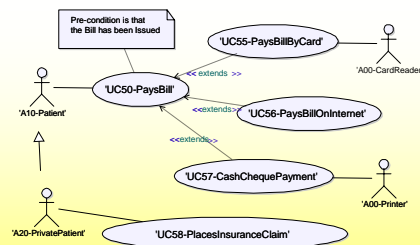
Inherits Relationship

- Will make sense when use cases are documented as 'essential' and 'concrete'
 - (Ref. Constantine and Lockwood)
- Essential use cases describe the 'intentional' behaviour – corresponding concrete behaviour can be inherited from it
- *Tip: This Inherits relationship has still not fully matured in UML, hence it is best used only when understood within the project*

© Copyright Practical OO Analysis, Thomson Publishing, 2005.

27

“Accounting” Use Case Diagram

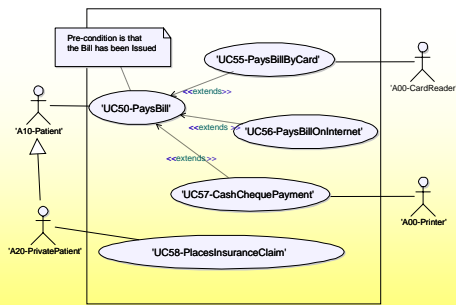


Note: In this use case diagram, the system boundary has not been drawn. However, observe how the use cases are all grouped together in the centre of the diagram and the actors are on the outside. This improves the aesthetics of the use case diagram.

© Copyright Practical OO Analysis, Thomson Publishing, 2005.

30

“Accounting” Use Case Diagram



"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

31

Conclusions(2)

- Use Case Diagrams are drawn during analysis to Visualize
 - ✓ Users and their Interaction with the system.
 - ✓ Relationship between various Use Cases are discussed (Include, the most relevant relationship; Others are Extend and Inherit)

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

34

Use Case Diagrams: Common Pitfalls!

- Treated as Data Flow Diagrams
 - Use Activity Diagrams to Show the Flow
- Looks like a Spider’s web
 - Abstract both Actors and Use cases and draw separate ‘abstract’ use case diagrams
- Too Coarse or Too Fine Granularity
 - Experience needed to Get it Right
 - Perhaps Use Case Diagram Patterns can Help?
- Use Case Diagrams treated as Deliverables
 - More than 50% work is Documenting Use Cases
- Force fitting Actors that Don’t Exist

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

32

WORKBOOK



EXERCISES

For The Class Room!!

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

35

Conclusions(1)

- Use Case modelling Includes:
 - ✓ Identification of Actors (primarily Users, but also other external interfaces)
 - ✓ Documentation of Actors
 - ✓ Creation of Actor Hierarchy for Simplification
 - ✓ Identification of Use Cases based on the Actors Identified
 - ✓ Documentation of the Use Cases in sufficient details (to be able to identify Classes)
- ✓ Pre-prepared Template for Actor and Use case documentation is helpful
- ✓ Use case Variations: Essential versus Concrete

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

33

Workbook Exercises(1)

1. Identify and List 5 additional Actors from the hospital domain. Ensure at least one Actor is a non-human or a device. Stereotype them appropriately.
2. Place the Actors you identified above in an Actor diagram with appropriate Actor hierarchy. Add notes for clarification.
3. Document any one of the actors you have identified above. Use the template provided in the appendix.
4. Identify and document TWO use cases: one in the shorter version and another in greater detail using the longer and more formal version of use case documentation. Use the template provided in the appendix.

"Copyright Practical OO Analysis,
Thomson Publishing, 2005".

36

Workbook Exercises(2)

1. Name and Draw two *additional* use case diagrams from the Hospital domain. Include at least two actors (from the previous module's Classroom exercises) and three use cases spread over the two diagrams.
2. In the first use case diagram, show the include relationship between two use cases clearly. In addition, show an actor-to-actor inheritance relationship. Stereotype the actors use cases where relevant.
3. Ensure you have at least one non-human actor in this first diagram.
4. Ensure you have shown the system boundary in this first diagram.
5. In the second use case diagram, show the extend relationship clearly. Again stereotype the actors and use cases.
6. Add notes on both diagrams to provide further explanations.



PROJECT WORK : (DURING TUTORIALS IN THE LAB);

Follow the Project Work Requirements given at the End of
the Chapter;
Discuss Project Work with Team Members and Tutor;
Carry Out the Project Work