UML簡介

資訊科技系
林偉川

Requirements engineering processes

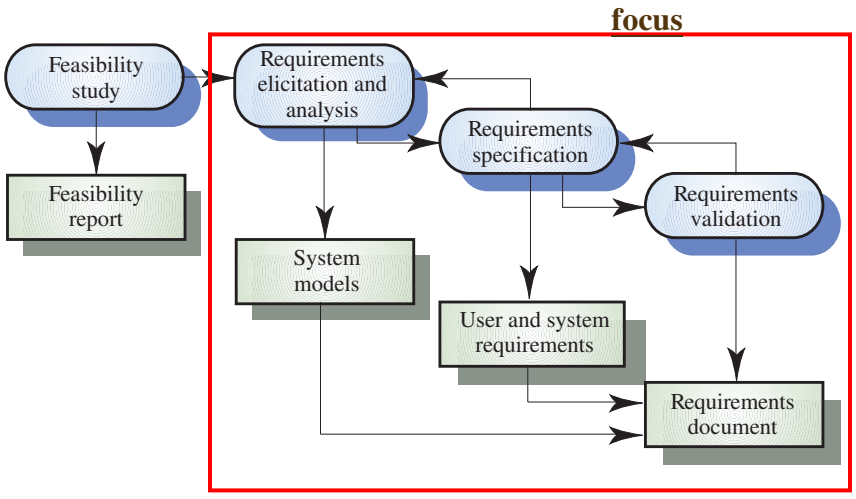
- Requirement engineering is a process that involves all of the activities required to create and maintain a system requirements document. There are 4 generic RE process activities: **Feasibility study, Requirement elicitation and analysis, Requirement specification, Requirement validation**

Requirements engineering processes

- The processes used for RE vary widely depending on the **application domain**, the **people involved** and the **organisation** developing the requirements
- However, there are 4 generic activities common to all processes
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management

3

The 4 requirements engineering process



4

Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile
- **Input** is an outline description of the system and how it will be used within an organization. **Output** is a report which recommands whether it is worth or not

5

Feasibility studies

- A short focused study that checks
  - If the system contributes to organisational objectives(make money?)
  - If the system can be engineered using current technology and within budget and schedule
  - If the system can be integrated with other systems that are used (legacy system)

6

Feasibility study implementation

- Feasibility study involves information assessment (answer to 3 above questions), information collection and report writing
- Questions for people in the organisation
  - What if the system wasn't implemented?
  - What are problems with current process?
  - What direct contribution will the system make to the business objectives?
  - Can information be transferred to and from other organizational systems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system and what is not?

7

Feasibility study implementation

- The report of proposing changes to the scope, budget and schedule of the system and further high-level requirements for the system

8

Requirement elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery
- Involves **technical staff** working with customers to find out about the **application domain**, the **services** that the system should provide and the system's operational **constraints**
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade union representatives, etc. These are called ***stakeholders*** (some direct or indirect influence on the system requirement)

9

Problems of requirements analysis

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own **terms**
- Different stakeholders may have conflicting or common requirements
- Organisational and political factors may influence the system requirements
- The economic and business environment during the analysis process is dynamic. New stakeholders may emerge and new requirement emerges

10

Process activities(iterative process)

Generic process activities of elicitation and analysis process :

- Domain understanding
- Requirements collection – interact with stakeholders

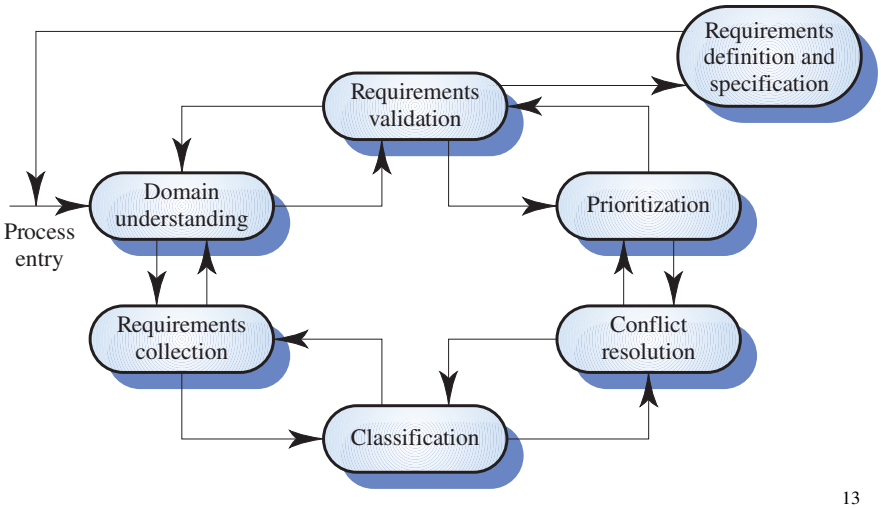
11

Process activities(iterative process)

- Classification – restructure the un-structure requirement
- Conflict resolution – because multiple stakeholders are involved
- Prioritisation – interact with stakeholders to decide
- Requirements checking – consistent and complete with stakeholders

12

The requirements analysis process



13

System models

- Different models may be produced during the requirements analysis activity
- Requirements analysis may involve three structuring activities which result in these different models
  - Partitioning. Identifies the structural (part-of) relationships between entities
  - Abstraction. Identifies generalities among entities
  - Projection. Identifies different ways of looking at a problem
- 3 techniques for requirement elicitation: Viewpoint-oriented elicitation (brain-storming perspective), scenarios (UML), and ethnography

14

System models

- Structured analysis methods
- Prototyping
- No perfect approaches to requirement analysis

15

Viewpoint-oriented elicitation

- Stakeholders represent different ways of looking at a problem or problem viewpoints
- This multi-perspective analysis is important as there is no single correct way to analyse system requirements

16

Viewpoint-oriented elicitation

- The stakeholder's perspectives are not completely independent and may usually overlap
- Key strength of VP-oriented analysis provides a framework for discovering conflicts in the requirements proposed by different stakeholders

17

Banking ATM system

- The example used here is an auto-teller system which provides some automated banking services
- I use a very simplified system which offers some services to customers of the bank who own the system and a narrower range of services to other customers
- Services include cash withdrawal, message passing (send a message to request a service), ordering a statement and transferring funds

18

Auto-teller viewpoints

Stakeholders for ATM include:

- Current Bank customers(receive services)
- Representatives of other banks(其他銀行行員)
- Managers and counter staff of bank branches
- Database administrators(integrate the customer's data)
- Bank security manager(ensure the system not pose a security hazard)
- Bank's marketing department(use DB data for marketing analysis)➔**business intelligent by data mining**
- Hardware and software maintenance engineers

19

Types of viewpoint

A VP can be considered as:

- Data sources or sinks
  - ◆Viewpoints are responsible for producing or consuming data. Analysis involves checking that data are produced and consumed and that assumptions about the **source and sink of data** are valid(**SADT or CORE**)

20

Types of viewpoint

- Representation frameworks
  - ◆Viewpoints represent particular types of system model. These may be compared to discover requirements that would be missed using a single representation. Particularly suitable for **real-time systems**(**ER or state-machine model**)
- Receivers of services
  - ◆Viewpoints are external to the system and receive services from it. It examines the services received by different VPs, collects and resolves the conflicts. Most suited to **interactive systems**

21

Advantage of Receivers of services

- Natural to think of end-users as receivers of system services
- Relative easy to decide if something is a valid viewpoint
- Viewpoints are a natural way to structure requirements elicitation
- It is relatively easy to decide if a viewpoint is valid
- Viewpoints and services may be suitable to structure non-functional requirements

22

Method-based analysis

- Widely used approach to requirements analysis. Depends on the application of a structured method to understand the system
- Methods have different **emphases**. Some are designed for **requirements elicitation**, others are close to design methods
- A Viewpoint-Oriented Requirement Definition(VORD) has been designed as a service-oriented framework for requirement elicitation and analysis. It also illustrates the use of viewpoints

23

VORD process model

- Viewpoint identification
  - Discover viewpoints which receive system services and identify the services provided to each viewpoint
- Viewpoint structuring (generalization v.s. specialization)
  - Group related viewpoints into a hierarchy. Common services are provided at higher-levels in the hierarchy and are inherited by low-levels VPs (OO)

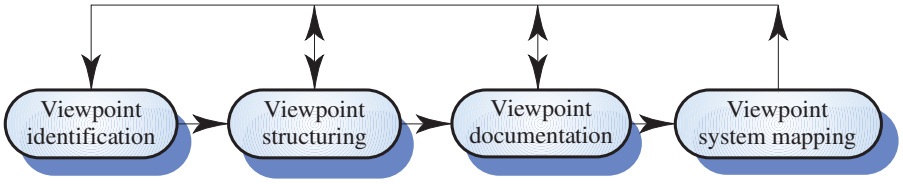
24

VORD process model

- Viewpoint documentation
  - Refine the description of the identified viewpoints and services
- Viewpoint-system mapping
  - Transform the analysis to an object-oriented design. Identify objects in an OOD way using service information provided by VPs (encapsulated in VP)

25

The VORD method



26

Object-Oriented approach

- Information hiding or Encapsulation (Data Abstraction)
  - Object, Attributes, Methods
- Inheritance (Single or Multiple)
  - Class hierarchy, reuse
- Polymorphism
  - Operator overloading, operator overwriting
- VORD uses diagrammatic notations including:
  - **Viewpoint hierarchy diagram**
  - **Event scenario**

27

VORD standard forms

Viewpoint template

Reference:	The viewpoint name.
Attributes:	Attributes providing viewpoint information.
Events:	A reference to a set of event scenarios describing how the system reacts to viewpoint events.
Services	A reference to a set of service descriptions.
Sub-VPs:	The names of sub-viewpoints.

Service template

Reference:	The service name.
Rationale:	Reason why the service is provided.
Specification:	Reference to a list of service specifications. These may be expressed in different notations.
Viewpoints:	List of viewpoint names receiving the service.
Non-functional requirements:	Reference to a set of non-functional requirements which constrain the service.
Provider:	Reference to a list of system objects which provide the service.

28

ATM examples

Functions

- Accept customer requests
  - Withdraw cash
  - Check their balance
  - Transfer funds from one account to another
  - Cheque book service
- Account information DB update

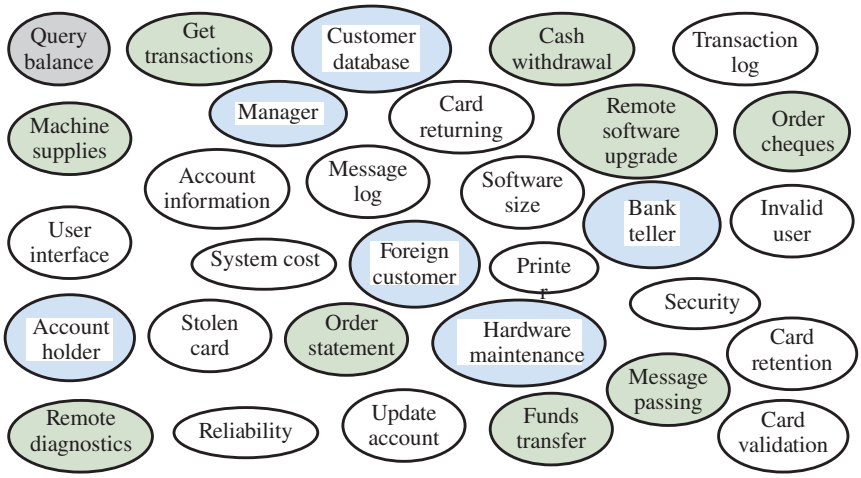
29

Viewpoint identification

- Identify all possible VPs is the most difficult stage
- Brainstorming approach to propose the potential services, data inputs, non-functional requirements, control events, exceptions, and VPs
- VPs are shown as light blue bubbles and services are shown as dark blue bubbles

30

Viewpoint identification



31

Viewpoint service information

A ACCOUNT HOLDER	B FOREIGN CUSTOMER	C BANK TELLER
Service list	Service list	Service list
Withdraw cash Query balance Order cheques Send message Transaction list Order statement Transfer funds	Withdraw cash Query balance	Run diagnostics Add cash Add paper Send message

32

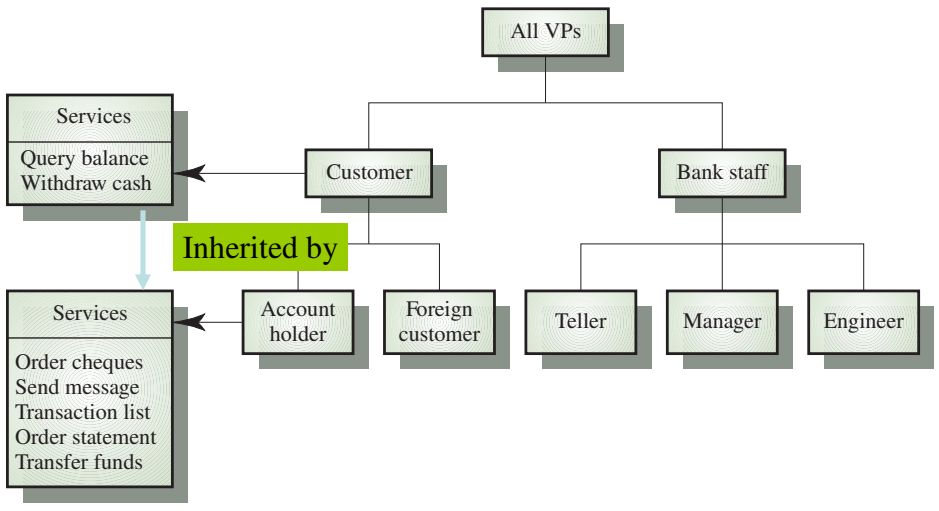
Viewpoint data/control

ACCOUNT HOLDER

Control input	Data input
Start transaction Cancel transaction End transaction Select service	Card details PIN Amount required Message

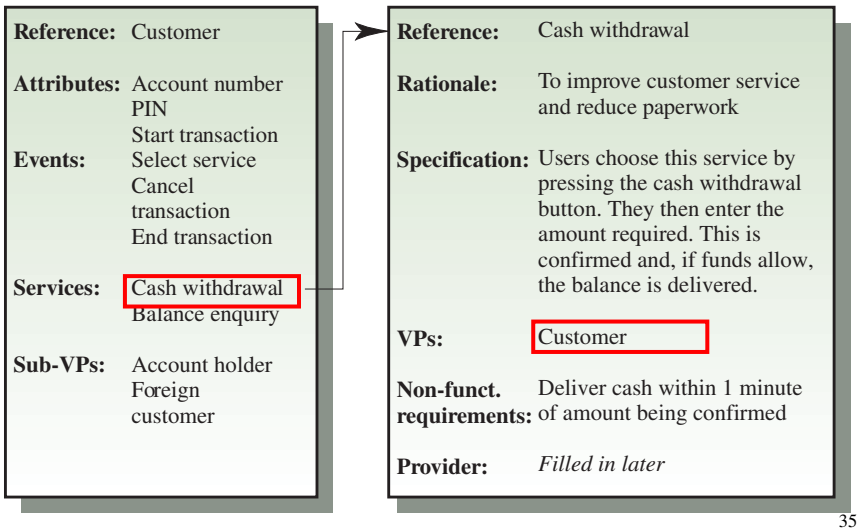
33

Viewpoint hierarchy



34

Customer/cash withdrawal templates



35

Scenarios

- Scenarios are descriptions of how a system is used in practice
- They are helpful in requirements elicitation as people can relate to these more readily than abstract statement of what they require from a system
- Scenarios are particularly useful for adding detail to an outline requirements description
- The scenarios starts with an outline of the interaction during elicitation, details are added to create a complete description of that interaction

36

Scenario descriptions

A scenario may include:

- A system state at the beginning of the scenario
- Normal flow of events in the scenario
- What can go wrong and how this is handled
- Other activities which might be going on at the same time
- System state after completion of the scenario

37

Event scenarios

- Event scenarios include a description of data flows and the actions of the system and document the exceptions which can arise

38

Event scenarios

- Event scenarios may be used to describe how a system responds to the occurrence of some particular event
- A diagrammatic conventions used in event scenarios:
  - Data provided from a VP and delivered to a VP
  - Control information
  - Exception processing
  - The next expected event

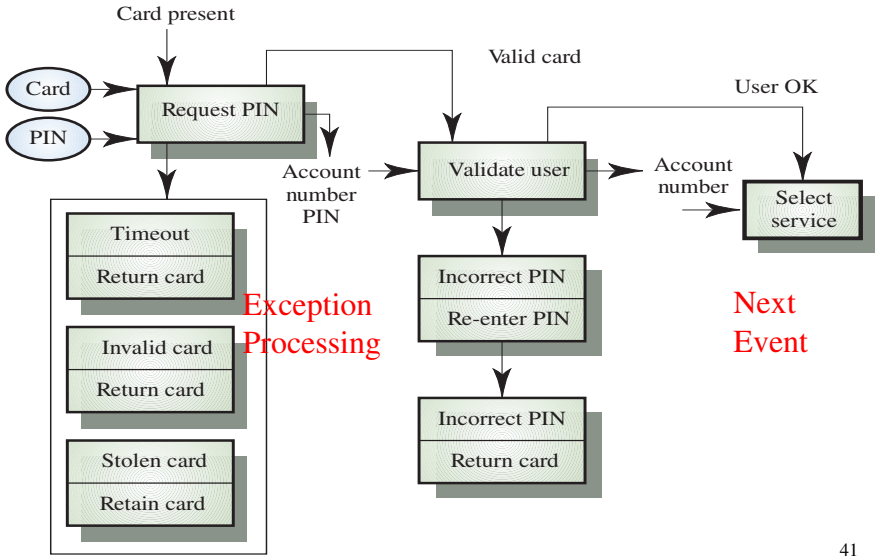
39

Notation for data and control analysis

- **Ellipses**. data provided from or delivered to a viewpoint(Card, PIN)
- **Control information** enters and leaves at the top of each box(Card present, Valid card)
- **Data** leaves from the right of each box(Account number PIN, Account number)
- **Exceptions** are shown at the bottom of each box
- **Name of next event** is in **a shaded box**

40

Event scenario - start transaction



41

Exception description

- Most methods do not include facilities for describing exceptions

42

Exception description

- In this example, at request PIN stages, exceptions are
  - Timeout. Customer fails to enter a PIN within the allowed time limit➔ The card is returned
  - Invalid card. The card is not recognised and is returned➔ The card is returned
  - Stolen card. The card has been registered as a stolen card➔ The card is retained by the machine
- At validate user stage, exceptions are Incorrect PIN checking(return card or re-enter PIN)

43

What is UML?

UML(Unified Modeling Language) is a well-defined and widely accepted language to build Object-Oriented and Component-based system developed by Rational Corp.

UML Combined OOSE/Booch,Jacobson and OMT/Rumbaugh is adopted by OMG in Nov/1997 and accepted by IBM,HP ...etc..

44

The characteristics of UML:

- **Use Case view**: find out the **internal** Use Case and external Actor
- **Logical view:describe the** internal static structure **and** dynamic behavior
- **Component view: describe the** module construction

45

The characteristics of UML

- **Concurrency view**:describe the procedure of modules
- **Deployment view: describe the** physical architecture **of system**

46

UML 的三個組件

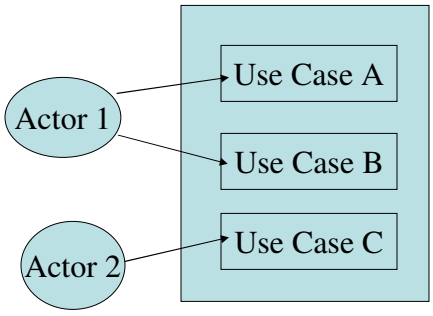
- 語言(language)
  - 讓人們能針對一項主題來進行溝通。在系統開發中，這包括需求與系統。
- 模型(model)
  - 是主題的表示方式。
- 統一(unified)
  - OMG 組織(Object Management Group，為確認工業標準化的組織)與 Rational Software Corporation 建立了 UML，並同時引進了資訊系統與技術工業的最佳工程實務。這些實務包含了應用技術，讓人們能更成功地開發系統。

47

UML使用的圖示工具:

1. Use case diagram

- 以使用者觀點找出系統運作之 Use Case及 external Actors



48

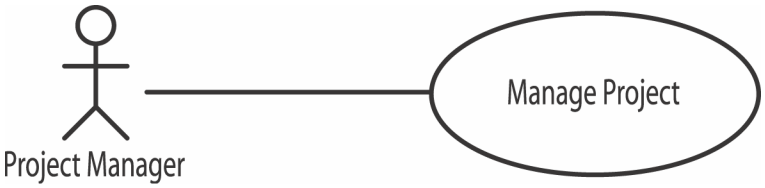
UML使用的圖示工具:

- **Use case diagram**
  - Use case 文字表示法

Use case name :
Actor's name :
Condition :
Function :
Use case description :

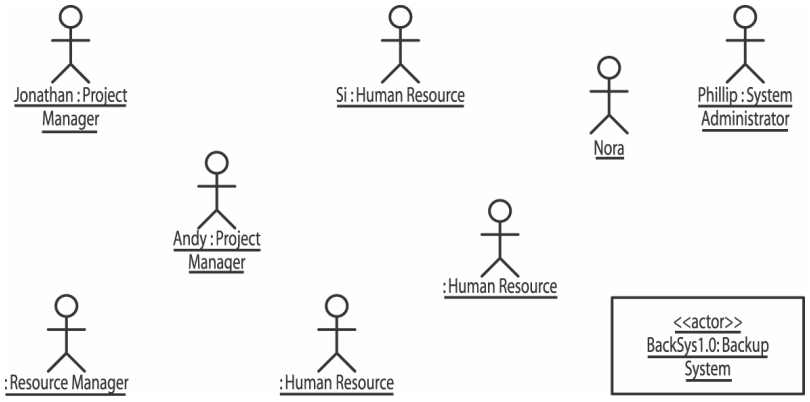
49

使用案例圖範例



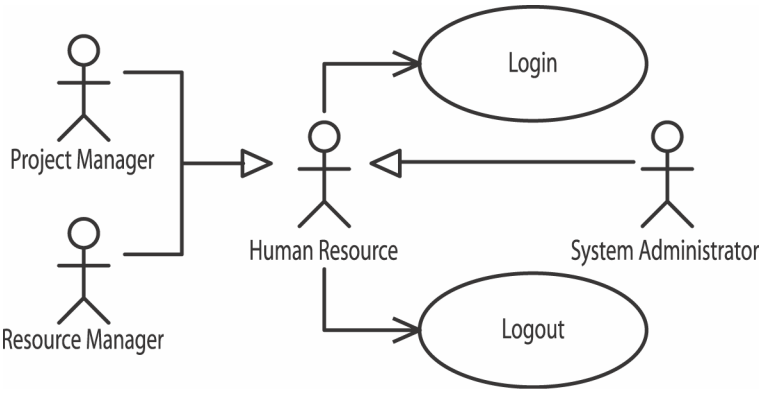
50

動作者



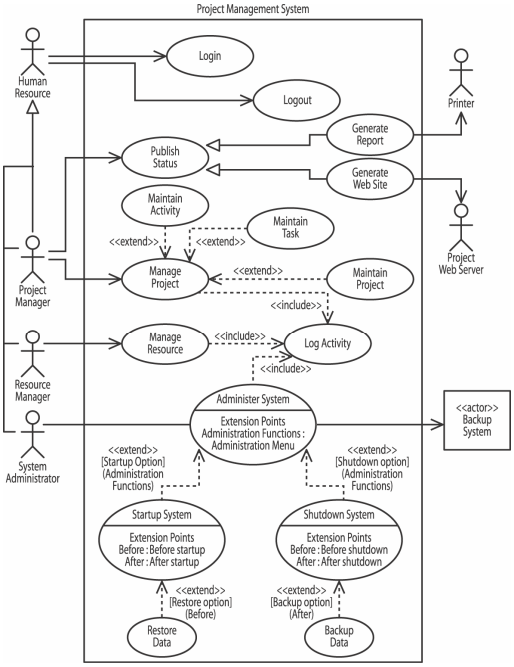
51

動作者泛化關係



52

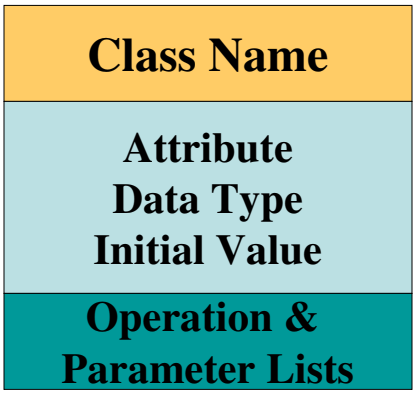
使用案例圖範例



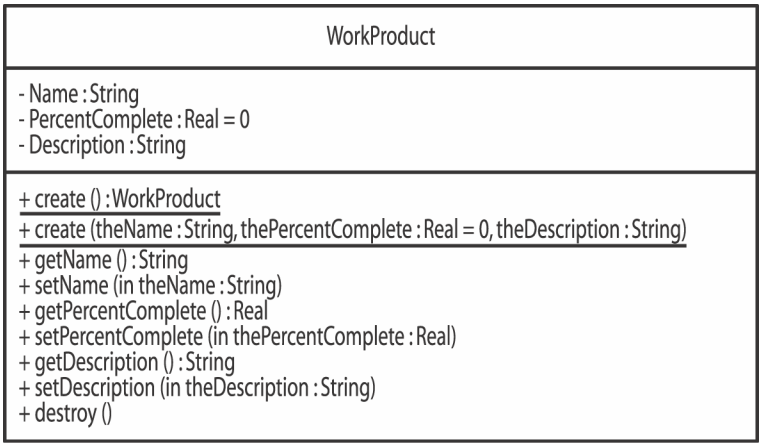
UML使用的圖示工具:

2. Class diagram

- 歸納Use Case diagram以整合說明同一類特性物件的靜態結構關係與內部關聯性



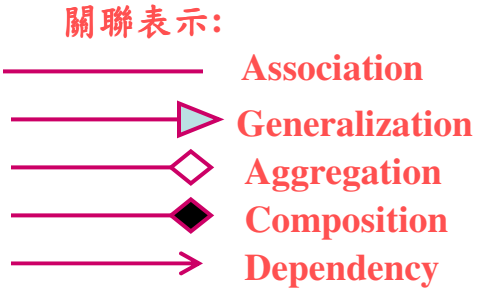
類別圖範例 II



55

UML使用的圖示工具:

- **Class diagram**
  - 類別間關聯表示圖



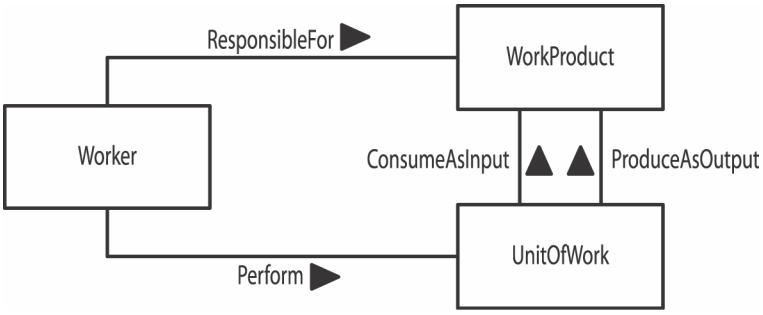
56

關聯

- 關聯定義了連結類型，且是類別間的通用關係。
  - 二元關聯 (binary association) 相關於兩個類別。
    - 在類別圖中，雙向關係的顯示方式是連接兩個相關類別間的實線路徑。
    - 二元關聯可能會標示一個名稱。
    - 此名稱的讀法一般是由左而右、由上往下；否則，在它旁邊會有一個小型的黑色實心三角形，此三角形的頂點則指出名稱的讀取方向，但僅供說明之用。
    - 二元關聯必須使用動詞來命名。

57

二元關聯



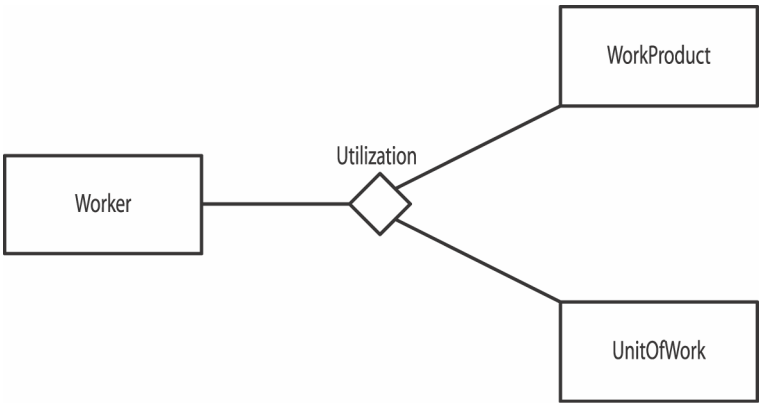
58

關聯

- 多元關聯 (n-ary association) 會相關於三個以上的類別。
  - 在類別圖中，多元關聯的顯示方式是一個大菱形，且會以實線，從此菱形連至每個類別。
  - 多元關聯可能會標示名稱。此名稱的讀法則與二元關聯的方式相同。
  - 多元關聯也常用動詞來命名，但仍有例外。

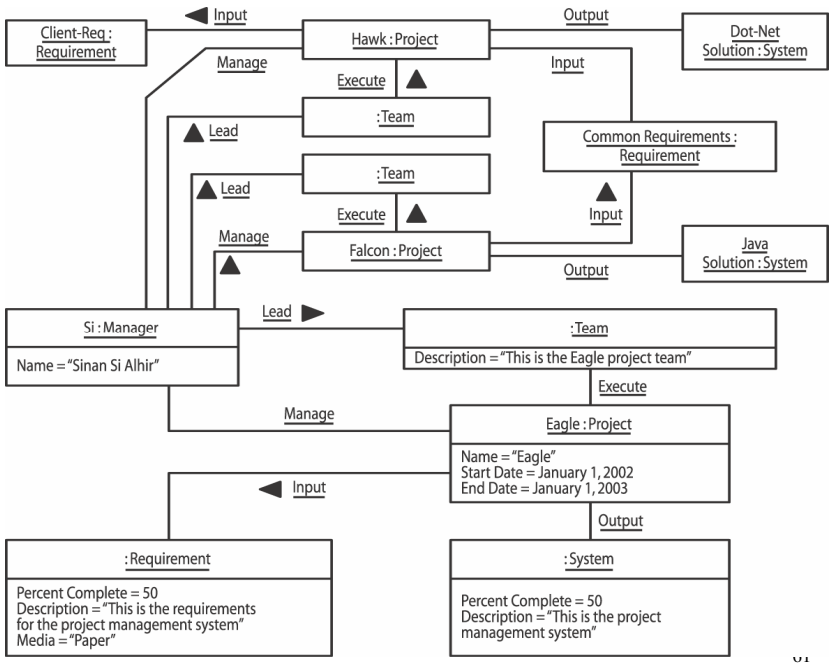
59

多元關聯



60

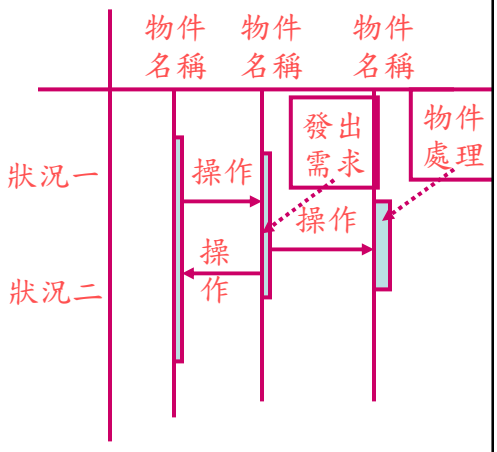
物件圖範例



UML使用的圖示工具:

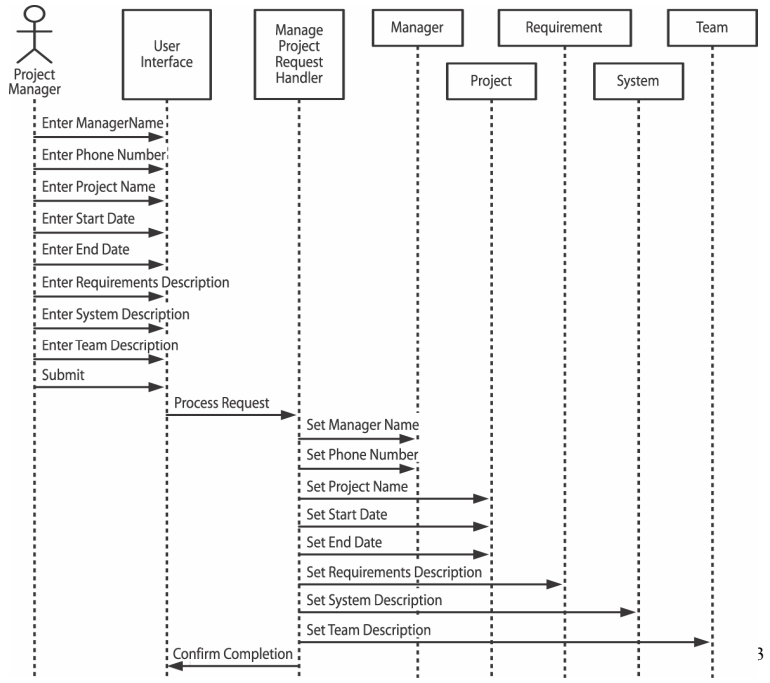
3. Sequence diagram

– 事件時間之發展,並表現物件活動之先後關係



62

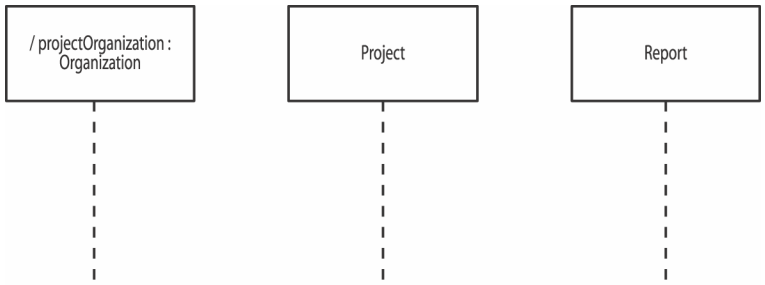
循序圖範例



3

生命線

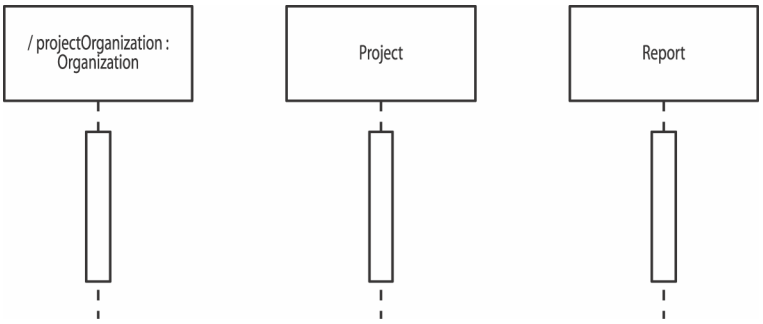
- 生命線 (lifeline) 的顯示方式是來自某元素的垂直虛線，這表示該元素的存在時間。



64

活化期間

- 選擇性的活化期間會在生命線中顯示一個瘦高的長方形，以表示在這段期間，元素會執行某項操作。長方形的頂端會與初始時間對齊，而其底端則會與完成時間對齊。

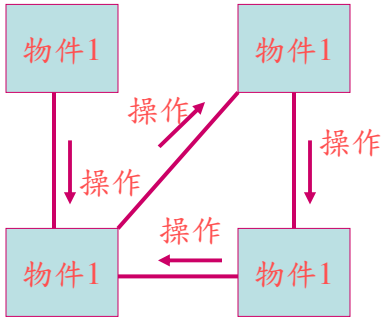


65

UML使用的圖示工具:

4. Collaboration diagram

– 以系統整體之宏觀表現，來說明各物件間的聯結與訊息交換的情形



66

狀態圖

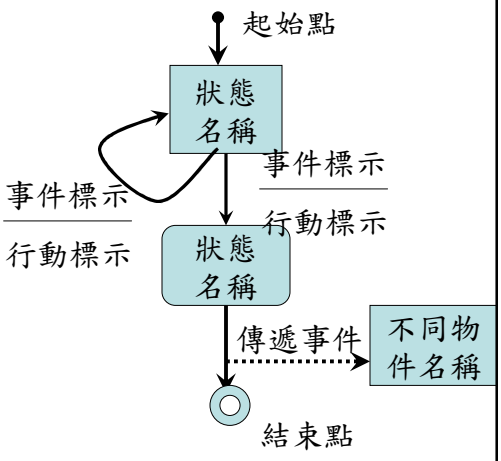
- 狀態模型是一種行為模型的特殊類型，會與元素的生命週期相關。通常會將狀態模型與互動和合作模型一起使用，以探究互動和合作元素的生命週期。
- 狀態圖(state diagram)，會詳述元素的生命週期，並有下列元素：
  - 狀態
    - 以圓角長方形來顯示，這表示元素的狀況或情況。
  - 事件
    - 此為接收訊息的發生事件。
  - 轉換
    - 顯示方式是從來源狀態至目標狀態的實線，並標上事件，這表示若元素在來源狀態中，且發生了此事件，則將輸入目標狀態。

67

UML使用的圖示工具:

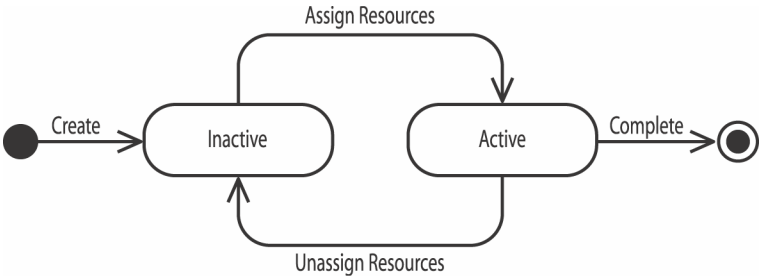
5. State diagram

- 描述類別內部因事件驅動所展現的行為,是系統微觀的狀態變化概念圖



68

狀態圖範例

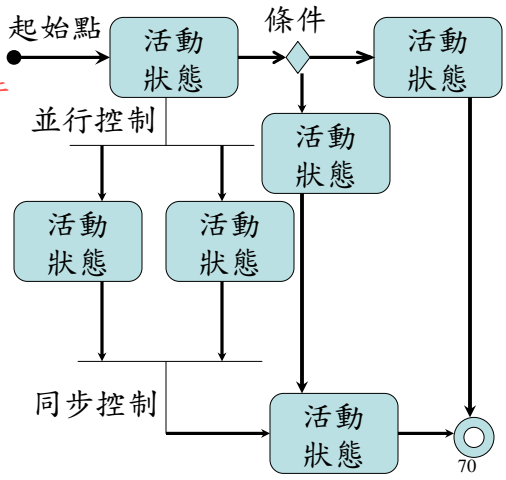


69

UML使用的圖示工具:

6. Activity diagram

- 描述事件發生時,物件的處理程序及執行某項操作的經歷過程



70

活動圖

- 活動模型化是行為模型的特殊化類型，並與模型化元素的活動與職責相關。通常會與循序模型及合作模型一同使用，以探索互動與合作元素的活動和職責。
- 活動圖 (activity diagram) 會詳述元素的活動與責任，並含有下列元素：
  - 動作狀態
    - 以上下是直線但兩側為弧形的形狀來顯示，這表示處理程序。
  - 控制流程轉換
    - 顯示方式為從來源動作狀態至目標動作狀態的實線，表示一旦此來源動作狀態完成處理程序之後，該目標動作狀態便會開始它的處理程序。
  - 初始動作狀態
    - 顯示方式為實心的小圓形，源自初始狀態的控制流程轉換會指定最初的動作狀態。

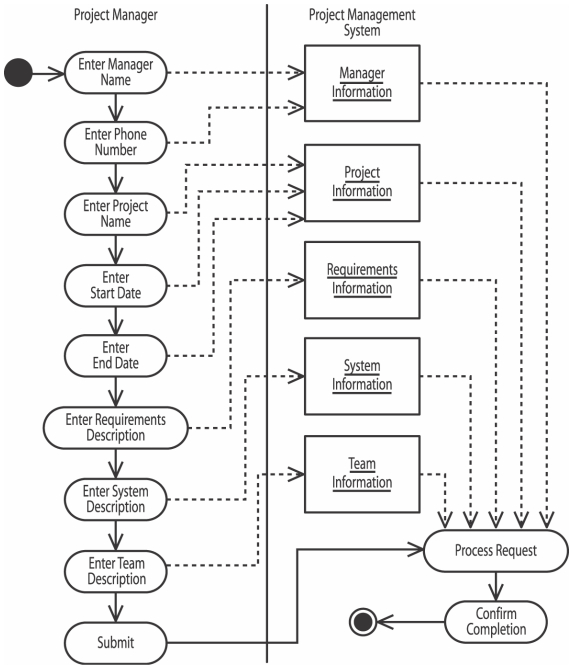
71

活動圖

  - 最終動作狀態
    - 顯示方式是在實心的小圓形外圍加上一個圓圈(如同牛的眼睛)，至最終狀態的控制流程轉換會指定最終動作狀態。
  - 物件流
    - 顯示方式為動作狀態與物件之間的虛線箭頭，這表示該動作狀態會輸入或輸出此物件。指向動作狀態的輸入物件流表示此動作狀態會輸入此物件。指向物件的輸出物件流則表示此動作狀態會輸出此物件。

72

活動圖範例

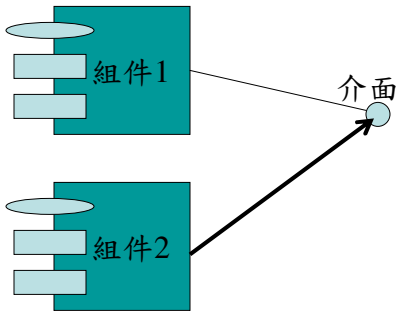


73

UML使用的圖示工具:

7. Component diagram

– 以實體觀點來描述類別與物件的放置情形,亦即表達出原始碼及軟體元件架構

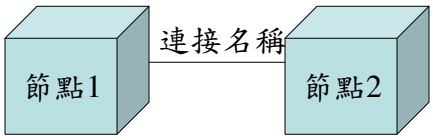


74

UML使用的圖示工具:

8. Deployment diagram

- 表現系統運作時之實體架構,如網路協定、主從架構及終端節點等



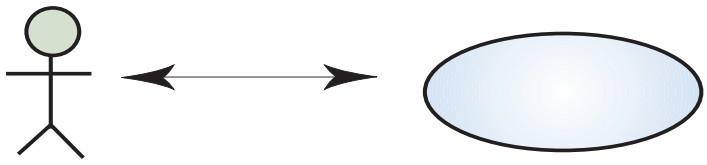
75

Use cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and names the type of interaction
- A set of use cases should describe all possible interactions with the system
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system

76

Lending use-case



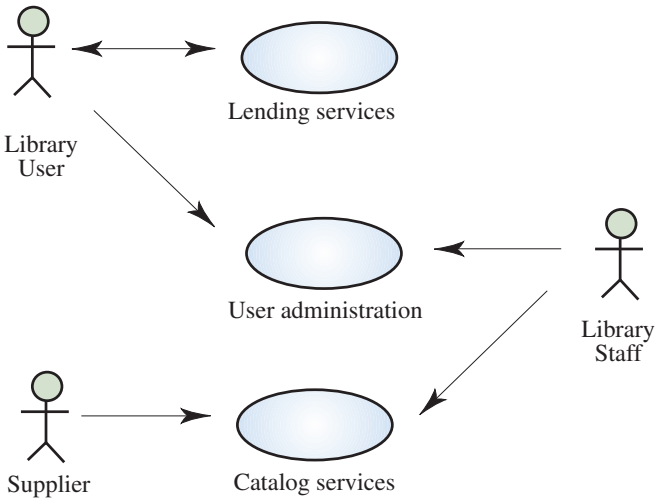
Actor

Lending services

Class of interaction

77

Library use-cases



78

Catalogue management(sequence diagram)