204362 – Object-Oriented Design

Modeling Concepts

Adapted for 204362
by Areerat Trongratsameethong

Simon Bennett, Steve McRobb, and Ray Farmer
Object-Oriented Systems Analysis and Design, 4th Edition, McGrawHill, 2010

Learning Objectives

Part I

- What is meant by a model
- The distinction between a model and a diagram
- The UML concept of a model
- How to draw activity diagrams to model process

Part II

- The purpose of activity diagrams
- The notation of activity diagrams
- How to draw activity diagrams

Part III

- About the Unified Software Development Process
- How phases relate to workflows in an iterative life cycle
- An approach to system development
- Major activities in the development process

Outline

- Introduction
- Models and Diagram
- Drawing Activity Diagram
- A Development Process
- Summary

Introduction

System Analysis and Design Process

ขบวนการในการพัฒนา Software มีขั้นตอนดังต่อไปนี้

1. เริ่มจากนักวิเคราะห์ธุรกิจ (BA) สร้างแบบจำลอง (Model) ของการทำงานในองค์กร
2. ต่อมานักวิเคราะห์ระบบ (SA) สร้างแบบจำลองพอสังเขป (Abstract Model) ของออบเจ็ค คือ สร้างออบเจ็คที่ยังไม่ต้องลงรายละเอียดมาก ในระบบ และแต่ละออบเจ็คปฏิสัมพันธ์กับออบเจ็คอื่นในระบบอย่างไร
3. หลังจากนั้นนักออกแบบ (Designer) จะสร้างแบบจำลองของระบบคอมพิวเตอร์ที่จะใช้ในการทำงานในองค์กร

Models and Diagram

- What is a model?
  - "A model is an abstract representation of something real or imaginary" e.g. map
  - Why use a model?
    - แบบจำลองสร้างได้เร็วกว่า และง่ายกว่า การสร้างของจริง
    - แบบจำลองสามารถใช้ในการ simulation ได้
    - แบบจำลองสามารถใช้ในการจำลอง สิ่งที่เราคิดค้น หรือสิ่งที่เราเรียนรู้ได้
    - เราสามารถเลือกรายละเอียดที่เราต้องการใส่ในแบบจำลองได้ ไม่จำเป็นต้องใส่ทั้งหมด
    - แบบจำลองสามารถนำเสนอความเป็นจริง หรือจินตนาการในเรื่องต่างๆ จากทุกๆเรื่องที่เกิดขึ้นได้
  - แบบจำลองทำให้เราสามารถจำลองสิ่งที่เป็นจริง หรือสิ่งที่ต้องการทำเพื่อให้เห็นภาพก่อนที่จะลงมือทำจริง

Models and Diagram

- Requirement Model
  - อธิบายว่า Software ควรจะทำอะไรบ้าง
  - นำเสนอสิ่งต่างๆในระบบ เช่น คน สิ่งของ แนวคิดที่สำคัญ เพื่อให้เข้าใจว่ามีอะไรเกิดขึ้นบ้างในระบบ
  - แสดงความเชื่อมโยง และการปฏิสัมพันธ์ระหว่างคน สิ่งของ และแนวคิด
  - แสดงรายละเอียดของสถานการณ์ในธุรกิจ ที่เพียงพอที่จะประเมินความเป็นไปได้ในการออกแบบระบบ
  - จะต้องทำให้เป็นระบบเพื่อให้เป็นประโยชน์ต่อการออกแบบ Software

Models and Diagram

- What is a diagram?
  - "A diagram is a visual or graphical representation of some part of a system"
  - Analyst และ Designer ใช้ Diagram ในการแสดงแบบจำลองของระบบ เหมือนกับที่สถาปนิกใช้ Drawing และ Diagram ในการนำเสนอแบบจำลองของตึก
  - Diagram ใช้สำหรับ
    - สื่อสารทางด้านความคิด
    - ทำให้เกิดแนวคิดใหม่ๆ และสร้างความเป็นไปได้
    - ทดสอบแนวคิด และใช้ในการคาดการณ์ (prediction)
    - ทำความเข้าใจโครงสร้าง และความสัมพันธ์

Models and Diagram

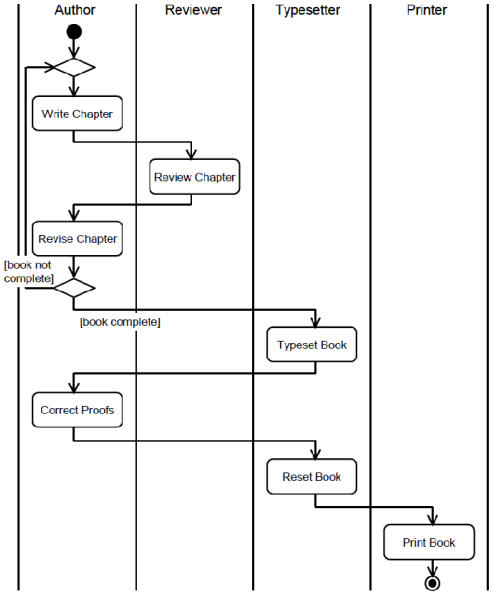
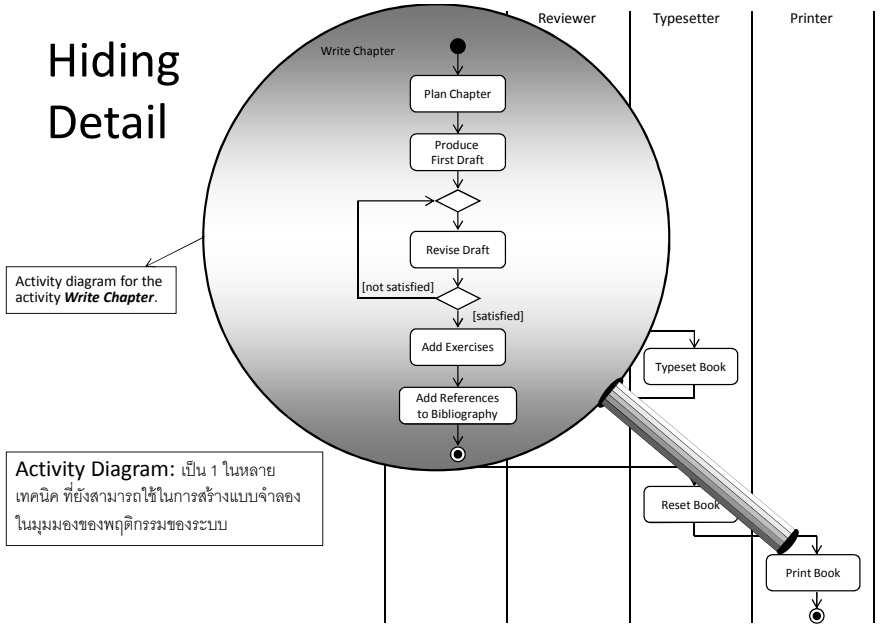


Figure 5.1 Activity diagram for producing a book.

Activity Diagram เหมาะสำหรับสร้างแบบจำลองลำดับการทำงานภายในองค์กรในภาพรวม อีกทั้งยังสามารถใช้ในการสร้างแบบจำลองที่ละเอียดขึ้นของการดำเนินงานต่างๆในองค์กร

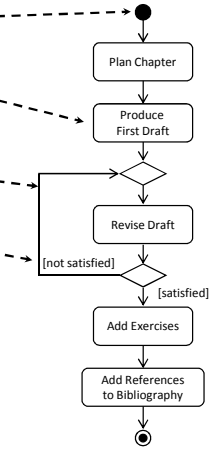
Hiding Detail



Activity Diagram: เป็น 1 ในหลายเทคนิค ที่ยังสามารถใช้ในการสร้างแบบจำลองในมุมมองของพฤติกรรมของระบบ

Diagrams in UML

- UML diagrams consist of:
  - icons
  - two-dimensional symbols
  - paths
  - Strings
- UML diagrams are defined in the UML specification.



Models and Diagram

- Models vs Diagrams?
  - Diagram ใช้แสดงบางแง่มุมของระบบ
  - Model แสดงมุมมองที่ครบถ้วนสมบูรณ์ของระบบ ณ เวลาใดเวลาหนึ่ง

Figure 5.3 Example from a diagram standard.

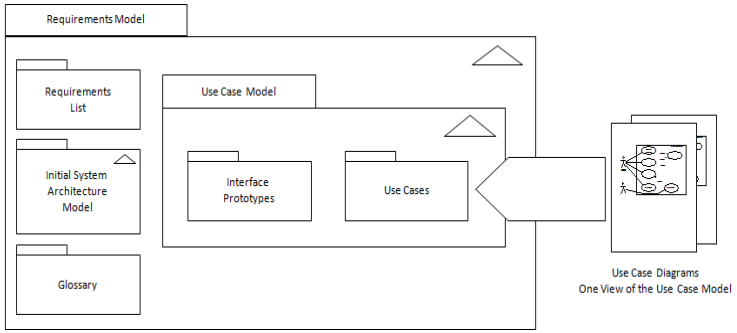


Figure 5.4 Illustration of a UML model and its relationship with one type of diagram

  - Requirement Model ในรูปที่ 5.4 ให้มุมมองที่ครบถ้วนสมบูรณ์ของความต้องการของระบบ ซึ่งอาจจะใช้ หนึ่ง หรือหลาย Diagram (ในที่นี้ใช้ หลาย Diagram) ในการครอบคลุมแง่มุมต่างๆ ของความต้องการของระบบ

Examples of Models

- Requirements Model
  - complete view of requirements
  - may include other models, such as a Use Case Model
  - includes textual description as well as sets of diagrams

Examples of Models

- Behavioural Model
  - shows how the system responds to events in the outside world and the passage of time
  - an initial model may just use Communication Diagrams
  - a later model will include Sequence Diagrams and State Machines

Models in UML

- A system is the overall thing that is being modelled
- A subsystem is a part of a system consisting of related elements
- A model is an abstraction of a system or subsystem from a particular perspective
- A model is complete and consistent at the chosen level of abstraction

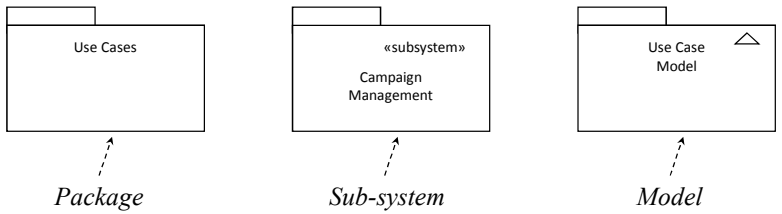
Models in UML

- Different models present different views of the system, for example:
  - use case view
  - design view
  - process view
  - implementation view
  - deployment view

(Booch et al., 1999)

Packages, Sub-systems and Models

- UML has notation for showing subsystems and models, and also for packages, which are a mechanism for organising models (e.g. in CASE tools)



Developing Models

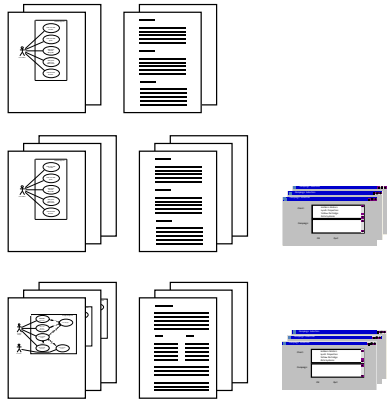
- During the life of a project using an iterative life cycle, models change along the dimensions of:
  - abstraction—they become more concrete
  - formality—they become more formally specified
  - level of detail—additional detail is added as understanding improves

Development of the Use Case Model

Iteration 1
Obvious use cases.
Simple use case descriptions.

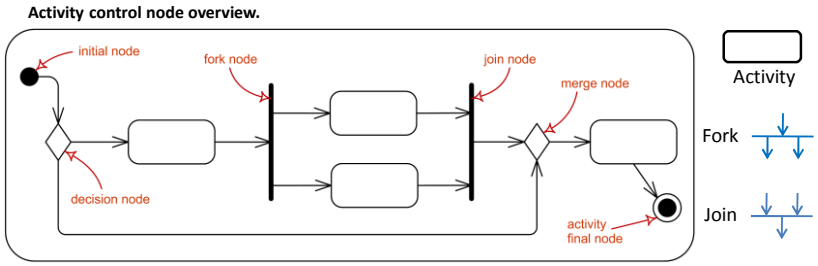
Iteration 2
Additional use cases.
Simple use case descriptions.
Prototypes.

Iteration 3
Structured use cases.
Structured use case descriptions.
Prototypes.



Drawing Activity Diagram

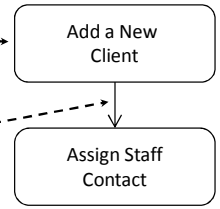
- Purpose of activity diagrams
  - ใช้ในการจำลองขบวนการ หรืองาน ในระบบงานธุรกิจ
  - ใช้อธิบายฟังก์ชันการทำงานของระบบ
  - ใช้อธิบายตรรกะของการดำเนินการต่างๆ
  - ในขบวนการพัฒนาซอฟต์แวร์โดยใช้ยูเอ็มแอล (Unified Software Development Process: USDP) ใช้ Activity Diagram ในการจำลองกิจกรรมต่างๆ



Reference: http://www.uml-diagrams.org/activity-diagrams-reference.html

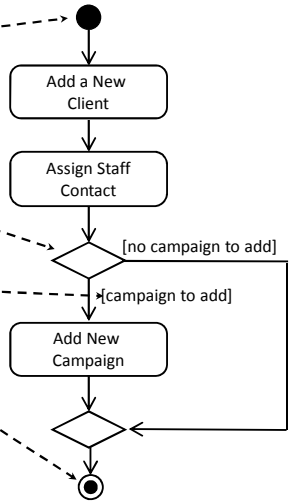
Notation of Activity Diagrams

- Actions
  - rectangle with rounded corners
  - meaningful name
- Control flows
  - arrows with open arrowheads



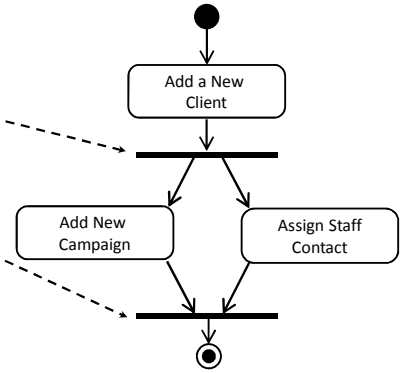
Notation of Activity Diagrams

- Initial node
  - black circle
- Decision nodes (and merge nodes)
  - diamond
- Guard conditions
  - in square brackets
- Final node
  - black circle in white circle



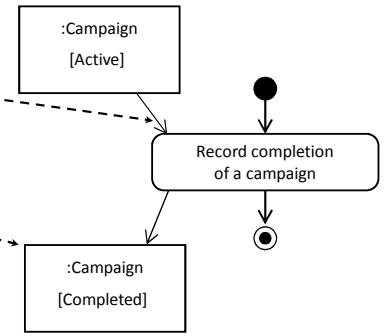
Notation of Activity Diagrams

- Fork nodes and join nodes
  - thick bar
- Actions carried out in parallel



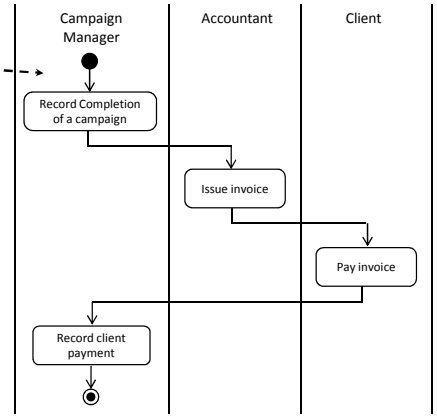
Notation of Activity Diagrams

- Object flows
  - open arrow
- Objects
  - rectangle
  - optionally shows the state of the object in square brackets



Notation of Activity Diagrams

- Activity Partitions (Swimlanes)
  - vertical columns
  - labelled with the person, organisation, department or system responsible for the activities in that column



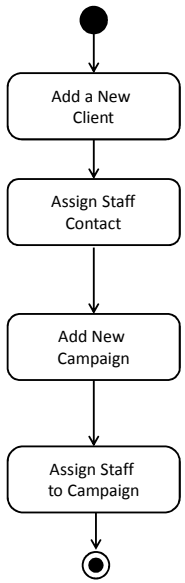
Drawing Activity Diagrams

- What is the purpose?
  - This will influence the kind of activities that are shown
- What is being shown in the diagram?
  - What is the name of the business process, use case or operation?
- What level of detail is required?
  - Is it high level or more detailed?

Drawing Activity Diagrams

- Identify actions
  - What happens when a new client is added in the Agate system?
    - Add a New Client
    - Assign Staff Contact
    - Add New Campaign
    - Assign Staff to Campaign
- Organise the actions in order with flows

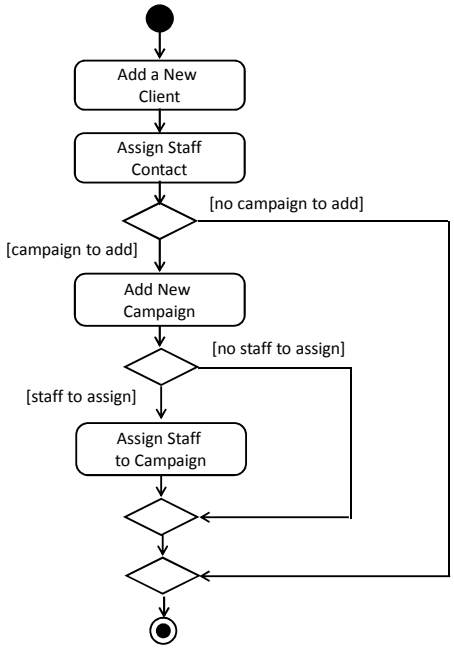
Drawing Activity Diagrams



Drawing Activity Diagrams

- Identify any alternative flows and the conditions on them
  - sometimes there is a new campaign to add for a new client, sometimes not
  - sometimes they will want to assign staff to the campaign, sometimes not
- Add decision and merge nodes, flows and guard conditions to the diagram

Drawing Activity Diagrams



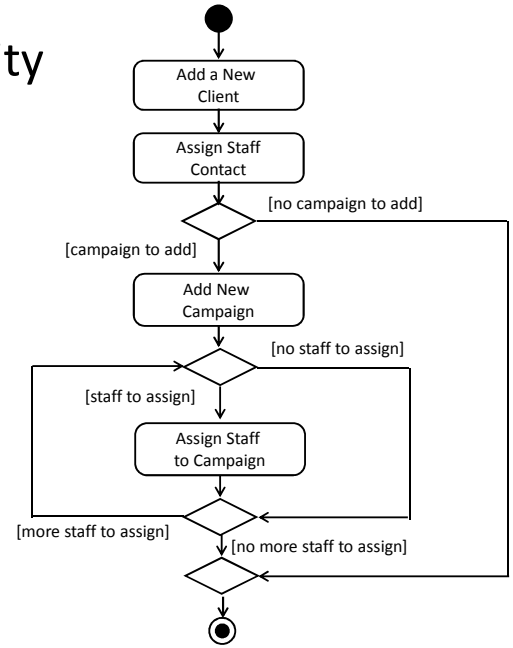
Drawing Activity Diagrams

- Identify any actions that are carried out in parallel
  - there are none in this example
- Add fork and join nodes and flows to the diagram

Drawing Activity Diagrams

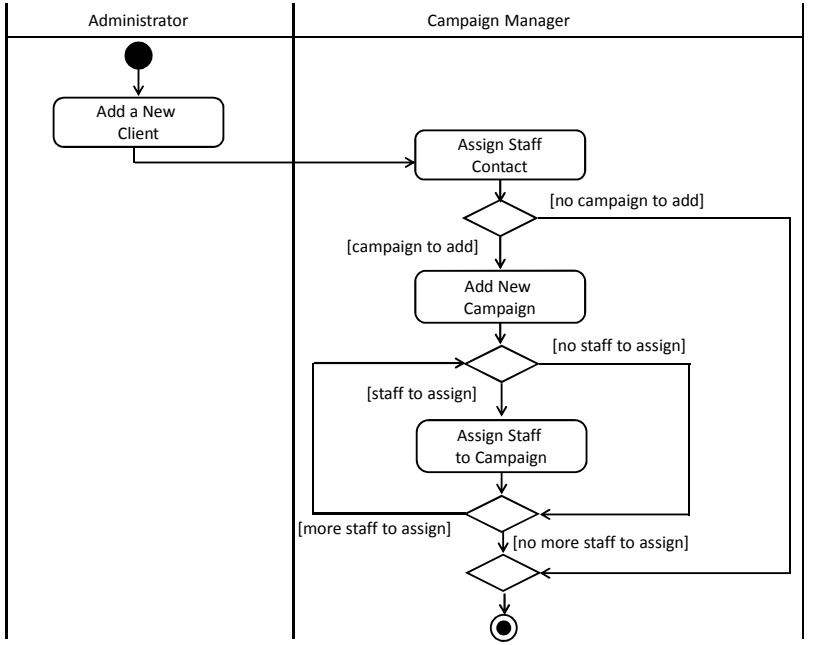
- Identify any processes that are repeated
  - they will want to assign staff to the campaign until there are no more staff to add
- Add decision and merge nodes, flows and guard conditions to the diagram

Drawing Activity Diagrams



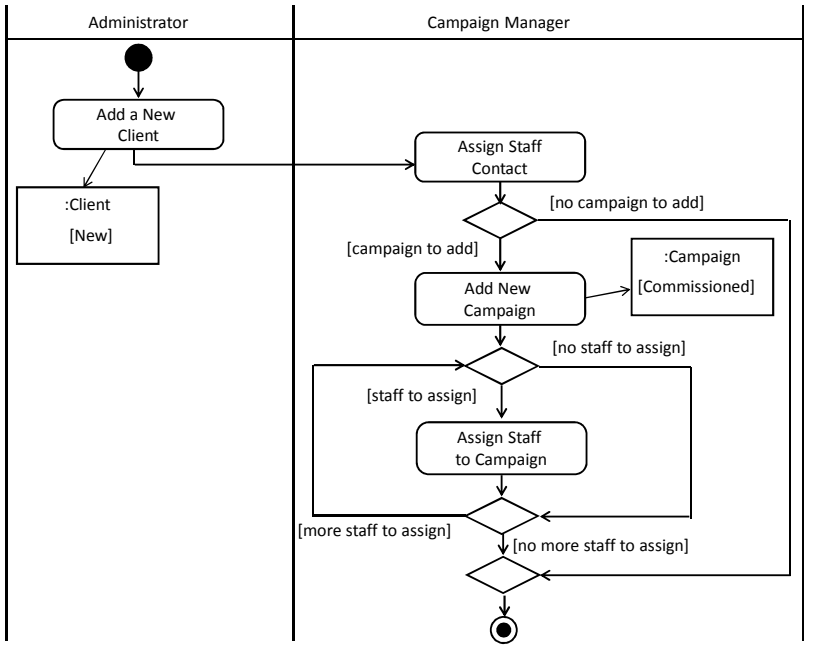
Drawing Activity Diagrams

- Are all the activities carried out by the same person, organisation or department?
- If not, then add swimlanes to show the responsibilities
- Name the swimlanes
- Show each activity in the appropriate swimlane

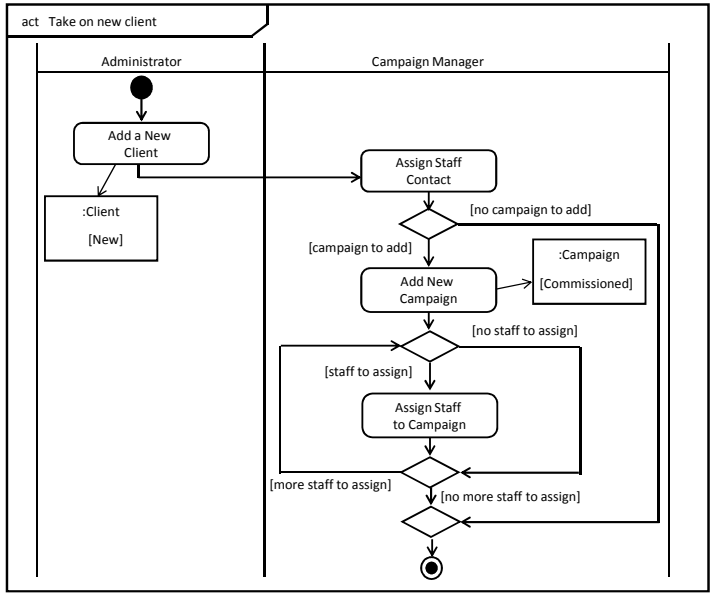


Drawing Activity Diagrams

- Are there any object flows and objects to show?
  - these can be documents that are created or updated in a business activity diagram
  - these can be object instances that change state in an operation or a use case
- Add the object flows and objects



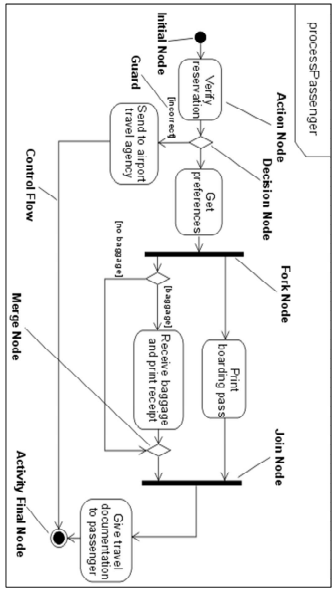
- When printed or copied a frame is used



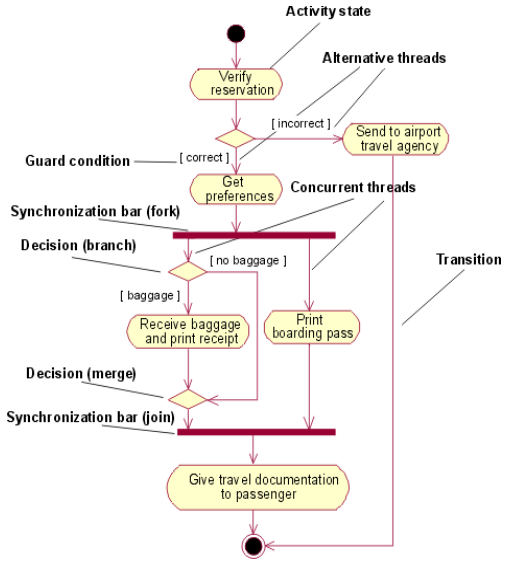
Notation of Activity Diagrams

- In UML 1.X multiple flows from an action were implicitly ORed
- In UML 2.0 they are implicitly ANDed
- Guard conditions do not have to be mutually exclusive, but it is advisable that they should be
- Decisions should be strictly nested, but…
- … a merge point can be combined with a following decision point

Activity Diagram – UML 2.0 vs UML 1.0



Example UML 2.0 activity diagram

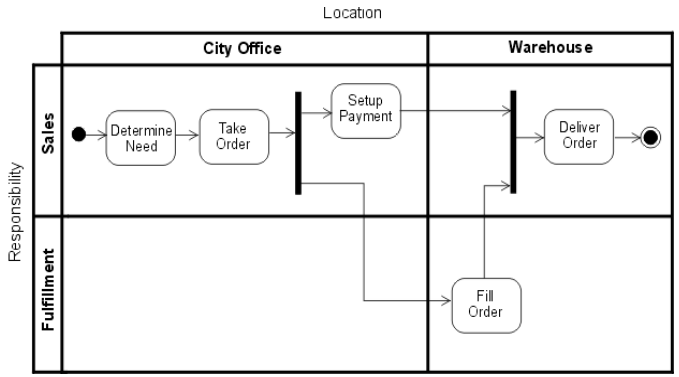


Example UML 1.x activity diagram

Reference: http://www.michael-richardson.com/processes/rup_for_sqa/core.base_rup/guidances/supportingmaterials/differences_between_uml_1_x_and_uml_2_0_CA70F2E6.html

UML 2.0 Activity Diagram - Activity Partitions

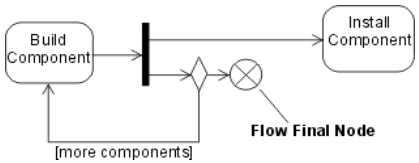
นอกจากจะแบ่ง Activity ในรูปแบบ Swimlane แล้ว ยังสามารถแบ่งกิจกรรมในมิติที่ 2 ได้ เช่น แบ่งตามแผนก



Activity partitions example using two-dimensional swimlane.

UML 2.0 Activity Diagram – Flow Final

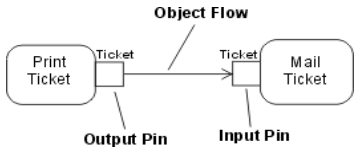
- Flow Final: An alternative to terminate a flow.
  - When control reaches any instance of Activity Final node, the entire activity (including all flows) is terminated.



Flow final control node.

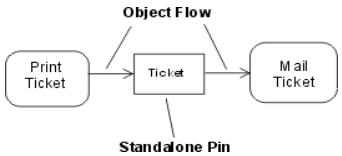
Activity Diagram – Object Nodes

- **Object Node:** An activity node that indicates that an instance of a particular classifier, possibly in a particular state, might be available at a particular point in the activity (for example, as output from, or input to an action).
  - Object nodes act as containers to and from which objects of a particular type (and possibly in a particular state) might flow.
  - New notation, called a ***pin***, has been introduced for object nodes in UML 2.0. Pins represent inputs to an action or outputs from an action and are drawn as small rectangles that are attached to the action rectangles.



UML 2.0 – Object Nodes and Pins.

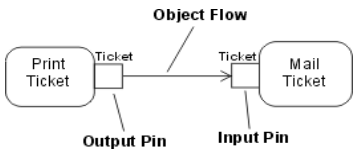
When the output pin on an action has the same name as the input pin on the connected action, the output and input pins may be merged to give a standalone pin.



UML 1.x – Object Nodes and Standard Pin.

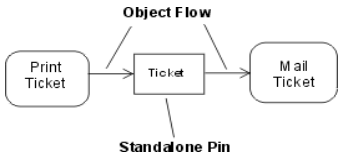
Activity Diagram – Object Nodes

- **Object Node:** An activity node that indicates that an instance of a particular classifier, possibly in a particular state, might be available at a particular point in the activity (for example, as output from, or input to an action).
  - Object nodes act as containers to and from which objects of a particular type (and possibly in a particular state) might flow.
  - New notation, called a ***pin***, has been introduced for object nodes in UML 2.0. Pins represent inputs to an action or outputs from an action and are drawn as small rectangles that are attached to the action rectangles.



UML 2.0 – Object Nodes and Pins.

When the output pin on an action has the same name as the input pin on the connected action, the output and input pins may be merged to give a standalone pin.

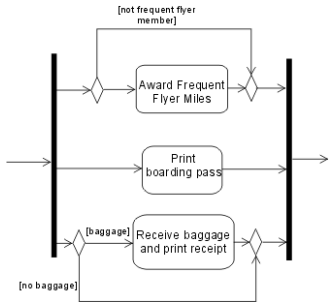


UML 1.x – Object Nodes and Standard Pin.

Activity Diagram – Semantic Differences

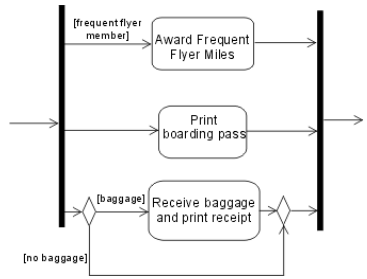
- **Passenger Check-in**: The agent needs to award the passenger frequent flyer miles.

if the passenger is not a frequent flyer, no token will ever reach the join along that flow and the model will stall because the join waits for tokens on all its flows before continuing. The model should be constructed as shown below.



UML 2.0 – Using decision and merge nodes instead of the guarded concurrent flow.

Placing the guard on the optional concurrent transition means that, the transition never starts, and the behavior is as if the transition were not shown in the model; accordingly, when the other two transitions complete, execution continues after the join.



UML 1.x – Using guarded concurrent transition.

Unified Software Development Process

- Developed by the team that created UML
- Embodies best practice in system development
- Adopts an iterative approach with four main phases
- Different tasks are captured in a series of workflows

Best Practice

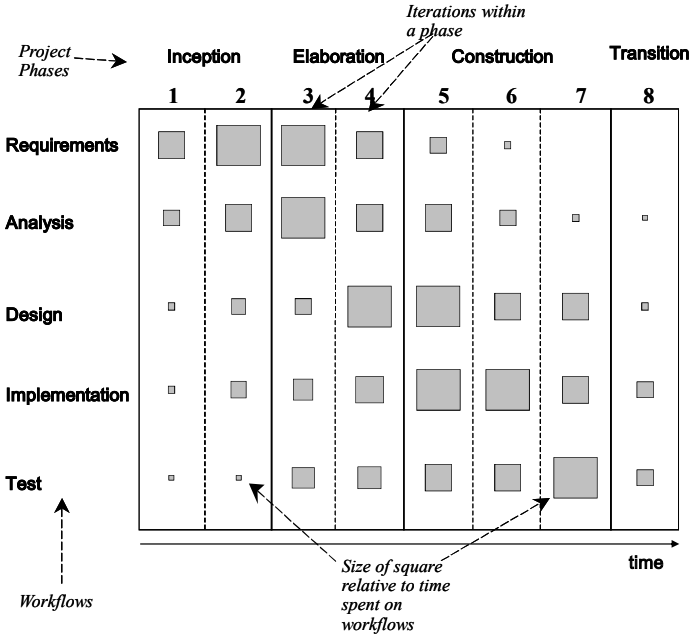
- Iterative and incremental development
- Component-based development
- Requirements-driven development
- Configurability
- Architecture-centrism
- Visual modelling techniques

Four Phases

- Inception
- Elaboration
- Construction
- Transition

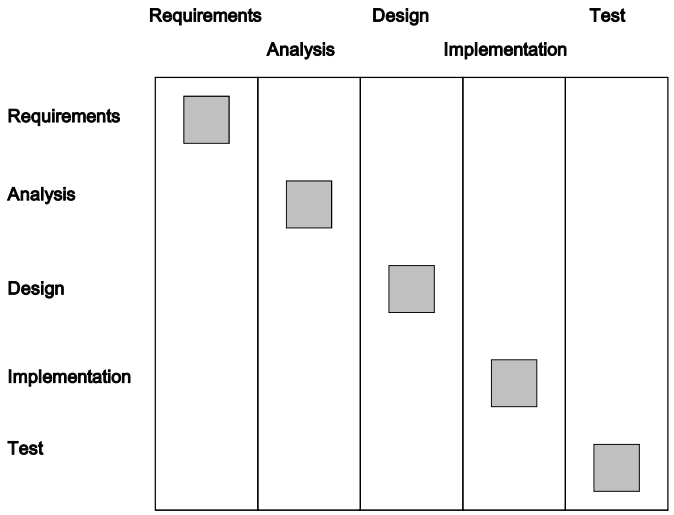
Phases, Workflows and Iterations

- Within each phase activities are grouped into workflows
- The balance of effort spent in each workflow varies from phase to phase
- Within phases there may be more than one iteration



Difference from Waterfall Life Cycle

- In a waterfall life cycle project the phases and the workflows are linked together
- In the Requirements phase, only Requirements workflow activities are carried out
- All Requirements activity should be completed before work starts on Analysis
- In an iterative life cycle project it is recognised that some Requirements work will be happening alongside Analysis work



Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Requirements Capture and Modelling	Requirements Elicitation Use Case Modelling Architectural Modelling Prototyping	Use Case Model Requirements List Initial Architecture Prototypes Glossary

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Requirements Analysis	Communication Diagrams Class and Object Modelling Analysis Modelling	Analysis Models

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
System Architecture and Design	Deployment Modelling Component Modelling Package Modelling Architectural Modelling Design Patterns	Overview Design and Implementation Architecture

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Class Design	Class and Object Modelling Interaction Modelling State Modelling Design Patterns	Design Models

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
User Interface Design	Class and Object Modelling Interaction Modelling State Modelling Package Modelling Prototyping Design Patterns	Design Models with Interface Specification

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Data Management Design	Class and Object Modelling Interaction Modelling State Modelling Package Modelling Design Patterns	Design Models with Database Specification

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Construction	Programming Component Re-use Database DDL Programming Idioms Manual Writing	Constructed System Documentation

Major Activities of the Development Process

Activity	Techniques	Key Deliverables
Testing	Programming Test Planning and Design Testing	Test Plans Test Cases Tested System
Implementation	Planning Training Data Conversion	Installed System

Summary

Part I:
- What is meant by a model
- The distinction between a model and a diagram
- The UML concept of a model

Part II:
- The purpose of activity diagrams
- The notation of activity diagrams
- How to draw activity diagrams

Part III:
- The Unified Software Development Process
- How phases relate to workflows in an iterative life cycle
- An approach to system development
- Major activities in the development process

References

- Booch, Rumbaugh and Jacobson (1999)
- Bennett, Skelton and Lunn (2005)
  (For full bibliographic details, see Bennett, McRobb and Farmer)
- The notation and semantics of activity diagrams have changed significantly since UML was first released. The original UML books by Rumbaugh, Booch and Jacobson are now out of date on the subject.
- Bennett, Skelton and Lunn (2005)
  (For full bibliographic details, see Bennett, McRobb and Farmer)
- Jacobson, Booch and Rumbaugh (1999)
- Kruchten (2004)
- Chapter 21 of Bennett, McRobb and Farmer includes more about the Unified Process as well as Agile alternatives
  (For full bibliographic details, see Bennett, McRobb and Farmer)