FIT2001- System development- Note

Systems Development (Monash University)

Seminar 1: Introduction to System Development

Information System:

Definitions: An integrated set of components of collecting, storing and processing data and for delivering information. 一套完整能收集，存储和处理信息的系统。
Including Components: People, Procedures, hardware/ software, databases, data warehouses, telecommunications.

System Development Life Cycle:

These processes are organised in **5 Phases:**

- **Initiation: Feasibility**
  - Affordable Time & Budget?
  - Expertise Availability?
  - System Requirements:
    - Mandatory V.S. Optional Features
    - System Compromises?
  - System Scope?
    - Scope Creep: A syndrome occurs during anything after the project begins, where there are continuous/uncontrolled growth & changes in a project/system's scope/functionalities.
    - Mitigations:
      - Reject addition of unnecessary features(JUST SAY NO).
    - Scope creep affects:
      - Budget
      - Schedule
      - Resources
- **Systems Analysis: What do clients want?**
  - Analysis of→In-Depth Analysis of system requirements
    - Task 1: Find stakeholders (internal/external)
    - Task 2: Identify the tasks (REQUIREMENTS) they want to perform
    - Task 3: Document those tasks and describe them
    - Task 4: Priority of those tasks (mandatory, optional)
  - Ensuring clients get what they want, but not necessarily the actions they assume it would be.
  - Create system models to confirm requirements and for design
  - Build V.S. Buy (More in Week 11).
- **Systems Design: How are you going to do it**
  - Modelling to show the integration of various components (Technical Architecture).
  - Task 1: Decide on a hardware and software platform for the new system.
  - Task 2: Design the software system, database, and user interfaces.
  - Task 3: Security considerations
- **Implement：Build/Develop - Construct & Test**

    - Build, Test, Validate
    - Conduct Integration, System and Acceptance Testing
    - Create User Documents, Training users
    - Install, Deploy new system
- **Support: Maintain it, Extend it**
    - Conduct post-implementation system review
    - Identify errors and enhancements
    - Monitor system performance

Key Roles and Skills for developers

- Key roles:
    - Managerial: Project Manager, Team Leader
    - Functional: System Analyst, Business analyst, Tester, Documenter, User Experience Designer
    - Technical: System Designer, Database Administrator, Solutions Architect, Developer/ Programmer, Tester, User Interface Developer, Security
- Other Roles: Quality Assurance, Documentation, Training and deployment

Critical Skills:

- **Understanding Business**: Awareness and sensitivity to the business processes and needs that require technology in the first place.了解业务的流程和需求
- **Broad and up-to-date understanding of technology:** Can be invaluable in creating the 'best' solutions for the organisation. 从多个方向上来决定和使用最佳的解决方案
- **Multiple perspectives:** The ability to understand that there are multiple perspectives to solving problems is required to find the best solution. 从多个视角解决问题。
- **People/ Soft skills:** The ability to interact with other people and to be a part of a team.和团队成员互动
- **Continuous Learning:** essential in a high-change industry.快速变化行业，保持学习而不被落后

Seminar 2: System Development Approaches

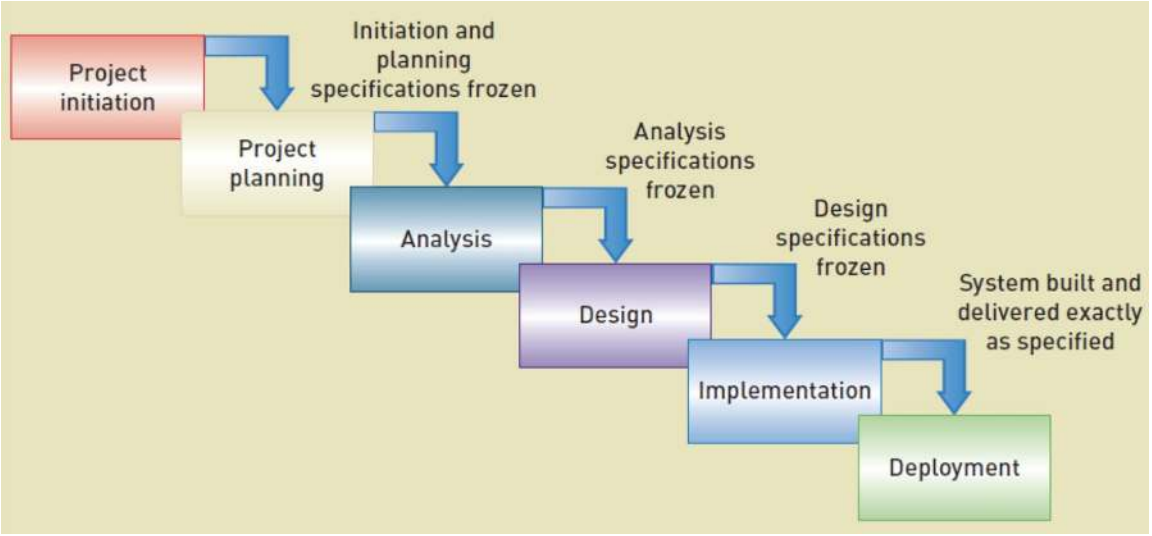
System Development Approaches

- Predictive:
    - Requirements well understood and well defined.
    - Low technical risk.
- Adaptive:
    - Requirements and needs are uncertain.
    - Iterative development
    - High technical risk.

Model & Approach of SDLC

- Waterfall Model (Predictive/Preventive)→Classic SDLC Approach
    - Sequential stages – no overlap or iteration

- Strong emphasis on planning and specifications development
- Works well for clearly defined projects



- Adaptive Approach ：
  - Iterative development：An approach to system development in which the system is “grown” piece by piece through multiple iterations
  - Each iteration is a miniature project with a well defined scope
  - At the end of each sprint, a potentially shippable product increment is delivered.
  - Every iteration sees new features added to the product, which results in the gradual project growth.
  - With the features being validated early and regularly, the chances of not delivering what the clients want reduces significantly. Let’s summarize the main Agile aspects:

Metric	Waterfall	Agile
Planning scale	Long-term	Short-trem
Distance between client and developer	Long	Short
Time from specification to implementation	Long	Short
Time to discover issues	Long	Short
Ability to meet deadlines	Poor	Good
Ability to respond quickly to change	Low	High

Frameworks / Methodology, Models, Techniques & Tools

Frameworks:

- Provide structure and direction on a preferred way to do something – guidance while being flexible

Methodologies

- A set of principles, tools and practices – conventions that an organisation / team agree to follow to achieve a particular goal.
    - **Models**: Is a representation (3D, 2D, mathematical) of an important aspect of the real world
    - **Techniques**: Represents a collection of guidelines that help an analyst complete an activity or task
    - **Tools:** Software applications that assist developers in creating models using a technique

Agile Development

FOUR (4) Agile Values

- Value 1: Responding to change following a plan
    - Quick changes are to be expected
    - Flexibility than rigidity in ideas, requirements over following a plan
- Value 2: Value individuals and interactions over processes and tools
    - Frequent communications among stakeholders are important
    - Do not over-emphasis on the use of tools like Lucid Charts, Visio etc
- Value 3: Value working software over comprehensive documentation
    - Less emphasis on proper documentation using models
    - Emphasises on the right amount of details
- Value 4: Value customer collaboration over contract negotiation
    - More and frequent inputs from users (rather than on fixed requirements)

12 Principles

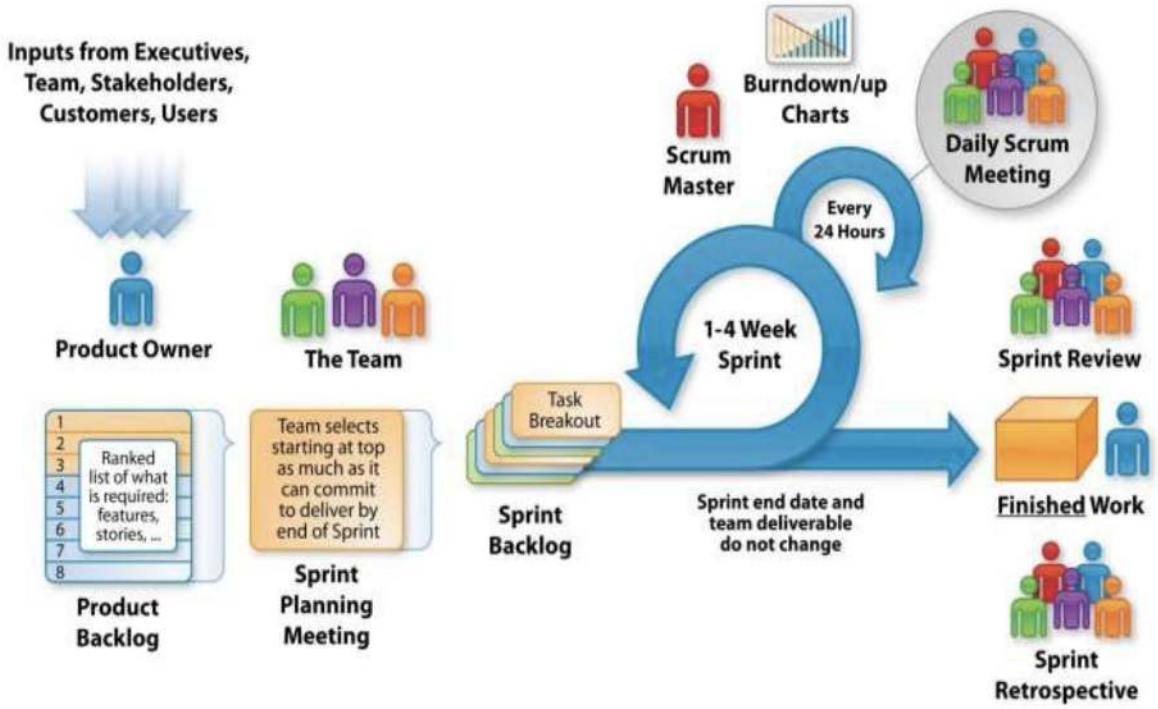
1. Software is the primary goal
2. Next effort is the secondary goal
3. Minimize modelling
4. Embrace change, change incrementally
5. Model with a purpose
6. Build multiple models
7. High-quality models, get feedback
8. Focus on content, not a representation
9. Communicate and learn from each other
10. Know models and how to use it
11. Adapt to specific project needs
12. Maximize stakeholder ROI

Agile frameworks--SCRUM

- A framework based on agile principles
- continuous improvement in process
- Delivers software (value) frequently
- A series of iterations called Sprints – typically 2-4 weeks long, based on an inspect and adapt cycle

- Produces outputs iteratively and incrementally, thus reducing risk and enhancing visibility

SCRUM Framework



SCRUM Roles

- **Product owner:** Client's representative, defines and prioritises product features, accept or reject work items
- **Scrum Master:** Coach for scrum team, applying agile principles, ensures team's productivity, builds a successful team
- **Product owner:** Client's representative, defines and prioritises product features, accept or reject work items
- **Scrum Master:** Coach for scrum team, applying agile principles, ensures team's productivity, builds a successful team
- **Development Team:** 5-9 members in a self-organizing, high performance, cross-functional team team (Developer, Tester, BA)

SCRUM Artifacts

- **Product Backlog:**
  - The single source of requirements
  - Cumulative list of desired deliverable for the project – every feature, enhancement, bug fix, documentation requirement, every bit of work required by the team
  - Prioritised to maximise value
- **Sprint Backlog**

    - A list of tasks the team must complete to deliver an increment of functional software at the end of each Sprint.
    - Once decided Team owns the Sprint Backlog – only they can decide on scope change
- **Sprint BurnDown Charts**
    - Shows the total estimated work remaining for the entire forecasted sprint backlog against time
- **Task Board (Kanban)**
    - Allows visibility, transparency across the project
    - Displays the live status of team work and focus
    - Backlog, To-do, In Progress (Doing) and Done status

Product Increment:

- A Product Increment is the end product for each sprint. It must:
    - Be of high enough quality to be given to users
    - Meet the Scrum team's current definition of DONE
    - Be acceptable to the product owner.

SCRUM Activities – A sprint cycle:

The Sprint is a timebox of 2-4 weeks during which the team produces a potentially shippable Product Increment.

- Start of the Sprint - Sprint Planning
    - Determine which items from the product backlog they will work on
    - Sprint Backlog – defines the scope of the sprint
        - Discussion with product owner – WHAT will we do
        - Team does the detailed plan – HOW will we do it
- During the Sprint - Daily Stand Up
    - Short (around 15 minutes) , discuss team coordinates & activities
        - 1. What I did since last daily scrum meeting
        - 2. What I am planning to work on today
        - 3. Impediments (Issues/blockers) if any?
- At the end of the Sprint
    - Sprint Review
        - A review to get ‘Product Increment’ feedback from the Stakeholders
        - Feedback goes into the ‘Product Backlog’ for future consideration.
        - Not intended to provide a status report
    - Sprint Retrospective
        - The team (including product owner) reflect upon how things went during the previous sprint
            - What went well
            - What could be Improved
        - Identify adjustments can make moving forward

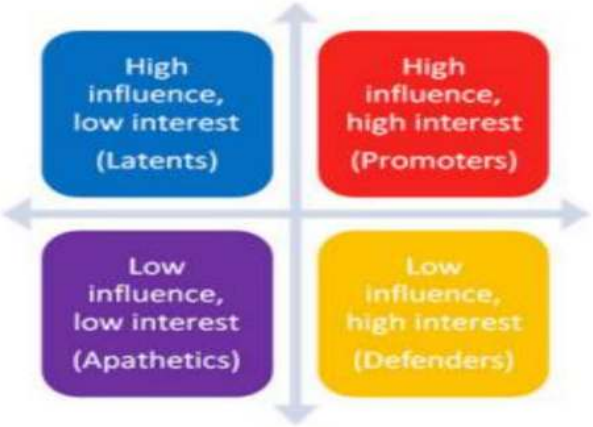
Stakeholders:
Persons who have an interest (stake) in the successful implementation of the system

FOUR Types:

- Type 1: Internal Stakeholders
  - persons within the organisation
- Type 2: External stakeholders
  - persons outside the organization, but have an interest
  - E.g. education department at the use of Moodle at Monash
- Type 3: Operational stakeholders
  - persons who regularly interact with the system
- Type 4: Executive stakeholders
  - persons who don't directly interact, but use the information or have a financial interest

Prioritise and understand your stakeholders:

- A stakeholder's position on the grid shows the actions a business analyst needs to take with him/her
- You need to understand how they feel about the project
- Determines how you engage/communicate with them



Seminar 3: Investigating System Requirements

Requirements
Represent the tasks a software system is required to perform within its constraints

- Two types of Requirements
  - Functional requirements
  - Non-functional requirements

Functional Requirements

- Represent the activities that a software system must perform
- Business functions that the end-users carry out (e.g. calculate a grade, calculate tax)
- Expressed in terms of models (e.g. user story, use case, activity diagram)

Non-Functional Requirements

- Represent other characteristics of software systems
- Like constraints (on what hardware and operating systems the system will operate)
- performance goals (e.g. speed)
- Various types:
    - Usability requirements
        - Represent operational characteristics
        - Like user interface, menu structure, colour choice, online help,
    - Reliability requirements
        - Shows dependability
        - E.g. services not available
    - Performance requirements
        - Response time (e.g. 1 second for logon)
        - Throughput (e.g. can support 100 students' usage of Moodle)
    - Security requirements
        - Password protection
        - Encryption
    - Design constraints
        - – Specific restrictions for hardware
        - – e.g. consume no more than 10MB memory, use CPU with over 1GHz speed and software
    - Implementation requirements
        - – Specific languages (e.g. C++ only), tools, protocols, etc.
    - Interface requirements
        - Interface links to other systems – e.g. in XML format only
    - Physical requirements
        - Physical facilities (e.g. size of computer is constrained to 15 cm)
        - equipment constraints (e.g. battery must be charged upto 24hours)
    - Supportability requirements
        - Automatic updates and enhancement methods
        - e.g. game installation on a home PC needs online help

Basic differences between Functional and Nonfunctional Requirements:

Functional Requirements	Non Functional Requirements
• Product features	• Product property
• Describe the actions with which the user work is concerned	• Describe the experience of the user while doing the work
• A functions that can be captured in use cases	• Non-functional requirements are global constraints on a software system that results in development costs, operational costs
• A behaviors that can be analyzed by drawing sequence diagrams, state charts, etc	• Often known as software qualities
• Can be traced to individual set of a program	• Usually cannot be implemented in a single module of a program

What is requirements gathering?

- Investigating requirements using a range of techniques
- Developing a deep understanding of the business domain
- Defining what the solution needs to be to meet the requirements
- Requirements must be verified by the client

Information gathering techniques

- Technique 1: Interviewing users and other stakeholders
- Technique 2: Distributing and collecting questionnaires
- Technique 3: Reviewing inputs, outputs, and documentation
- Technique 4: Observing and documenting business procedures
- Technique 5: Researching vendor solutions
- Technique 6: Prototyping
- Technique 7: Story-writing workshops

<u>Interviewing users and other stakeholders</u>

- An effective way but time-consuming
- How to prepare:
  - Set clear objectives & what information is needed
  - Select appropriate stakeholders to interview
  - Determine one-on-one or group interview
  - Review related documents and develop an agenda

- Documents objectives, nothing forgotten, logical progression
- Avoid long interviews & Choose several shorter interviews
- Obtain and discuss answers to the questions
- Record and document the answers
  - Ethics approval for recording
- Follow up as needed
  - in future meetings or interviews

Distributing and collecting questionnaires

On-line (e.g. survey monkey), Paper-based or E-mail

- Advantages
  - Cover a wide spectrum of people
  - Faster responses
  - Low costs of distribution
  - Good when the people are widely dispersed
  - Can give a preliminary insight into business
- Disadvantages
  - Low response rate, not many people are willing to respond
  - Not well suited for gathering detailed information

Questions must be carefully prepared to avoid multiple interpretations & must be written effectively

Reviewing inputs, outputs, and documentation

- Existing **internal business documents** and procedure descriptions
  - identify business rules, discrepancies, redundancies
  - be cautious of **outdated material**
  - obtain a **preliminary understanding of processes**
  - use as guidelines / visual cues to guide interviews
- External industry reports
  - Trade publications
  - Best practices

Observing and documenting business procedures

Be aware of the Hawthorne effect:

- Also referred to as the observer effect refers to a phenomenon, whereby workers:
  - improve or modify an aspect of their behaviour/activities,
  - or STOP WORKING in response to the fact that they are being observed

Research vendor solutions

- Many business problems have already been solved by other companies
- Positive contributions of vendor solutions – Frequently provide new ideas
  - May be state of the art
  - Cheaper and less risky

    - Get a trial version, technical specifications, on-site visits
- Risks
    - May not address all business requirements
    - Vendor support may not be there in future
    - Upgrade issues

Collecting active user comments, and suggestions through workshops:

- Get all stakeholders in a room for a couple of days and facilitate discussion.
    - Build models with everyone there.
    - People bounce ideas between each other.
- Requires a conference room, whiteboard, etc., facilitator and a scribe.
- Problems
    - Can be very difficult to organise, particularly when senior people are involved.
    - Requires taking people off work

Validating the requirements:

- Meet with users regularly to get feedback on your understanding of the system
- You must confirm that your understanding of the requirements is correct
- You are aiming for requirements that are:
    - Complete – all functions identified
    - Unambiguous – nothing vague or fuzzy
    - Sufficient – level of detail okay
    - Testable – can check if working as intended
    - Consistent – no conflicts among requirements

Seminar 4: Investigating and documenting system requirements User Stories, Activity Diagrams

Models - Why do we use them?

- The model serves as an abstraction - an approximate representation of the real item that is being built → A simplified picture of complex reality
- Reasons/advantage for modeling in systems analysis
    - Reducing complexity of systems to be built by abstraction
    - Communication with other development team members
    - Communication with stakeholders/users
    - Learning from the modelling process
    - Documenting all the details of requirements for future maintenance/enhancement - represents some key aspects of the system being built

User Stories: (Go into the Product Backlog)

- A user story represents:
  - a one-sentence description of a task
  - to be done by a user (ROLE)
  - to achieve a goal or result
  - to describe functional requirements
- Template:
  - As WHO I want WHAT so that WHY.
- Acceptance Criteria: Conditions of Satisfying the User Stories.
- EPIC - a LARGE story that is split into smaller user stories that share a strategic objective
- User Stories Advantages:
  - Easier to Communicate
  - Focus on end user value
  - Users do not need to be trained to understand User stories
  - Encourages collaboration and feedback
  - Avoids locking in design detail too early
  - Eliminates weighty documentation
- User Stories issues:
  - No design related issues
  - Leave the technical functions to the architect, developers, testers, etc.
  - Simplify user stories by splitting it if necessary

Story Mapping

- Arrange user stories into a useful model for understanding the functionality
- Identifying holes and omissions in the backlog, and effectively plan holistic releases that deliver value to the business
- Two dimensions
  - Sequence
  - Alternatives
- Process:
  - Start with the sequence of activities
  - Then look at possible alternatives
  - Prioritize the prioritize

Release planning

- A Release is a set of functionality that makes sense to deliver to users at the same time – can include stories from different Epics
- A Release can include multiple sprints

Activity Diagram：

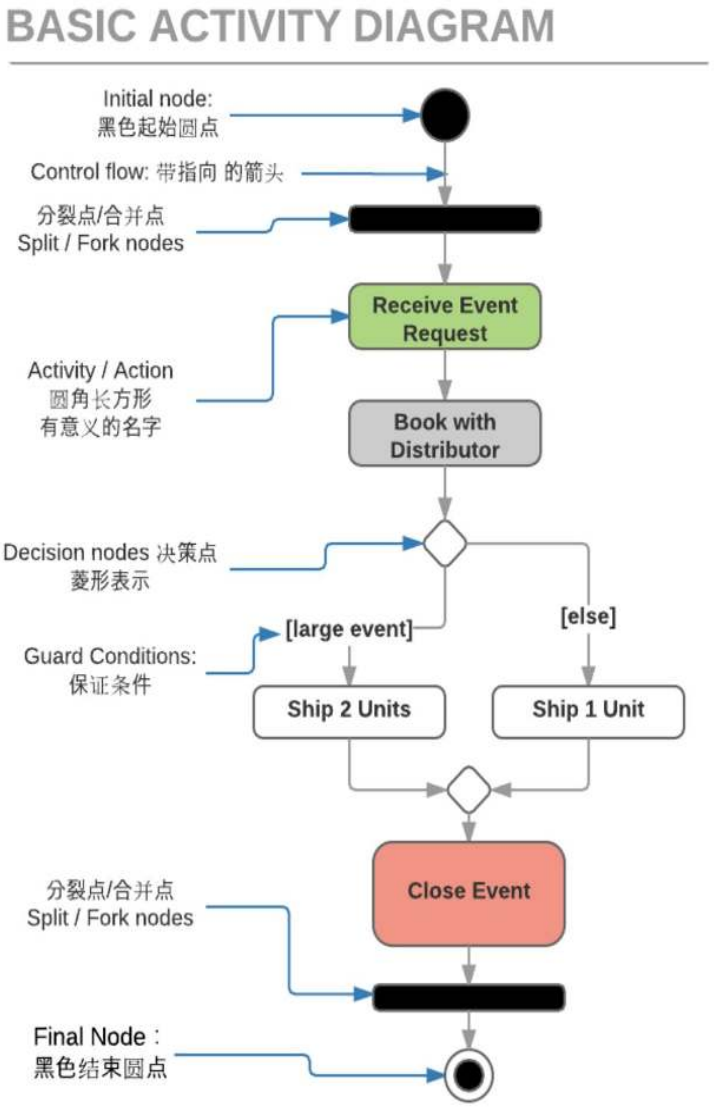
Technique to describe procedural logic/business processes/workflow, describes:

- user (or system) activities

- the person who does each activity
- the sequence of these activities

Symbols:

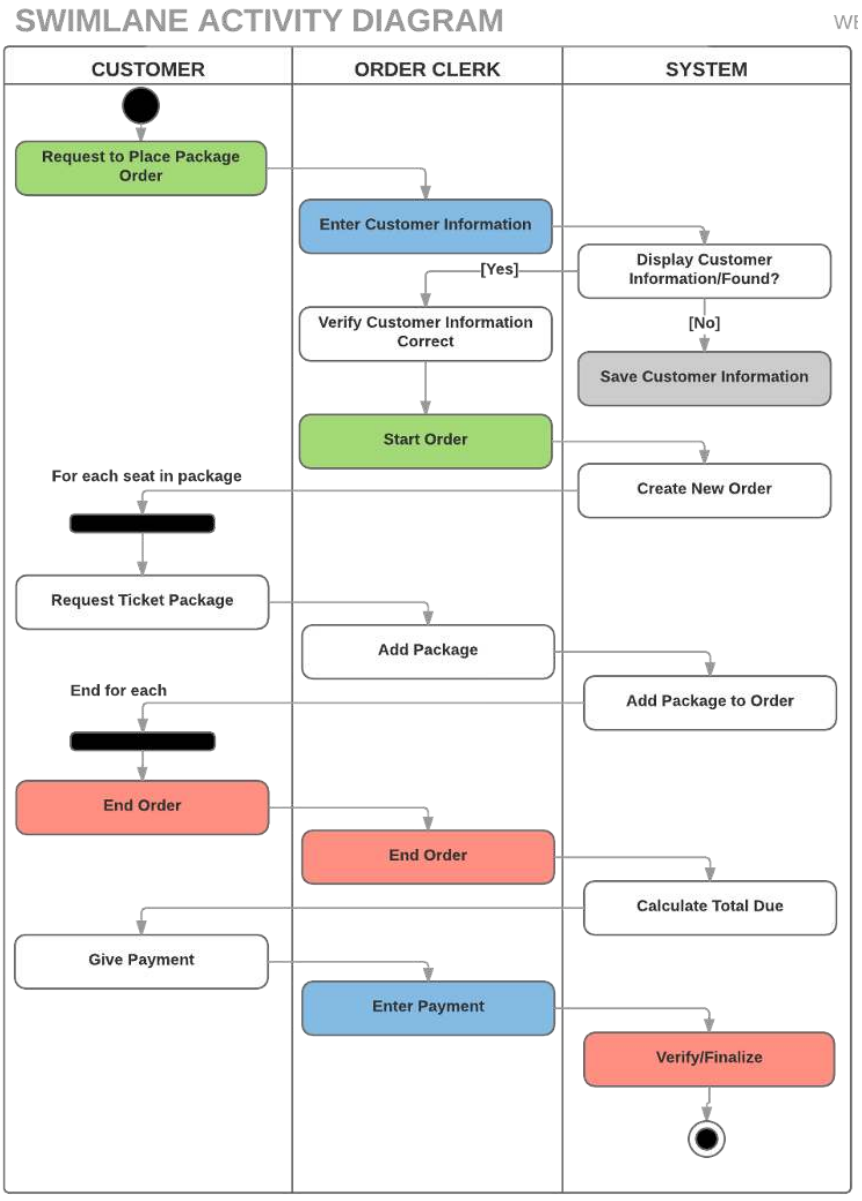
- Activity/Action: 圆角长方形+有意 义的名字
- Control flows: 有指向的箭头
- Notes:
- Initial node 初始点: 黑色圆圈
- Decision nodes/ Merge nodes 合并点和决定点:菱形
- Guard conditions 保证条件: 中 括号
- Final node 最终节点: 黑色圆， 外面有白色圈住
- Split/Fork nodes/Joint node 分裂 或者合并点:如图
- Actions 行动:并行的圆角长方 形
- Object flows:open arrow
- Object:长方形，选择性展示 object 的状态，用中括号
- Activity Partitions(Swim lanes)活动分区 一定要有 Swim lanes 并且命名:垂直的数列，标记人物，组织，部门或者负责活动的系统



How to draw a Activity Diagrams:

1. Identify actions
2. Organise the actions in order with flows
3. Identify any alternative flows and the conditions on them
4. Identify any actions that are carried out in parallel
5. Add fork and join nodes and flows to the diagram
6. Identify any processes that are repeated
7. Add decision and merge nodes, flows and guard conditions to the diagram
8. Add swimlanes to show the responsibilities, if not all the activities carried out by the same person, organization or department.
9. Name the swim lanes
10. Show activity in the appropriate swim lane

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



Email: admin@foruedu. com Mobile: 041 6626 066
Add: Lv2/33 Waverley Road, Malvern East, 3145

Seminar 5: Investigating and documenting system requirements Use Case Diagrams & Use Case Descriptions

Use Cases:

Use cases

- define functional requirements
- an activity that a system performs, usually in RESPONSE TO A REQUEST by a user
- involves a list of steps between a role “ACTOR" and a system, to achieve a goal
- describes interactions between an Actor and a system

User Story VS Use Case

- Similarity:
  - They both focus on the goals of users
  - They both describe functions that users want
- Difference:
  - The degree of details differ
  - Unlike user story, a use case can be depicted in three formats
    - – Tabular format without much details(verb-Noun)
    - – Tabular format with details (Narrative format)
    - – Graphical format

User Stories	Use Cases
Focussed on the result and benefit of system functions	More granualar, describes how the system will act
Minimal documentation	Detailed documentation
Small increments for getting feedback	Most done up front

Identifying Use Cases:

- 1.User Goals
  - Identify & Classify Users
  - Further classify potential users
  - Interview each type of user
  - Resolve inconsistencies

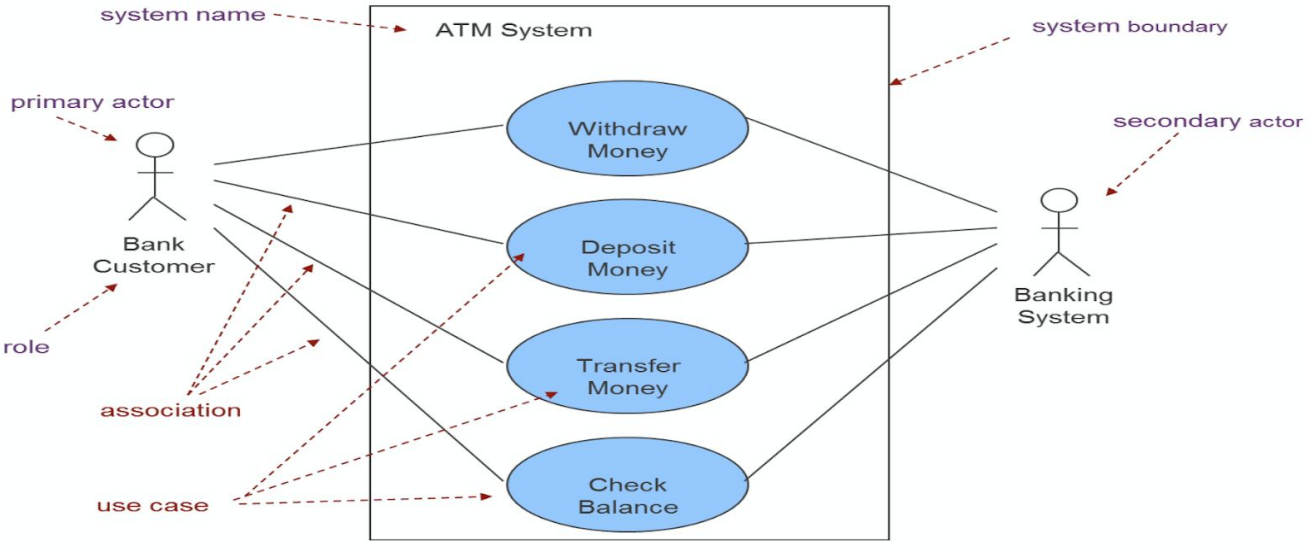
- Identify the same use cases for different users.
- Review & Complete

- 2. Event Decomposition → Identifying events that occur to which the system must respond. of Events:
  - b. Types
    - i. External→Occurs outside the system, usually initiated by an external object.
    - ii. Temporal→Occurs when reaching a certain point in time.
    - iii. State→Occurs upon being triggered by another event.

Use Case Diagram:

Elements:

- i. Actor→An person/entity that initiates the functionality provided by a Use Case (Role).
- ii. Primary V.S. Secondary
  - 1. Primary Actor→Using the system to achieve a goal.
  - 2. Secondary Actor→The system needs assistance to achieve the primary actor's goal(s).
- iii. Use Case→Interaction between the system and user/another system (Verb + Noun).
- iv. Association (Line)→Indicates that an actor utilising the functionality provided by a Use Case.
- v. System Boundary (Border)→Scope of the system.
- vi. Relationship→Dependency between the 2 Use Cases:
  - 1. Include→Commonality among Use Cases by Managing Redundancy (Triggering the parent ALWAYS triggers the child).
  - 2. Extend: Adds to existing functionality & characteristics of the parent (Triggering the parent OPTIONALLY triggers the child).



Email: admin@foruedu.com Mobile: 041 6626 066
Add: Lv2/33 Waverley Road, Malvern East, 3145

Seminar 6: Domain and Class Modelling

Class:

Class represent:

- THINGS that are of interest to stakeholders
- THINGS about which information needs to be collected & stored
- THINGS that people (stakeholders) deal with when they do their work (in business process)

Classes can be:

- Tangible (e.g. people, products, items)
- Intangible (e.g. event like sale, meetings, bookings, conversations)

Domain class model:

- A Domain class model is a graphical representation of relationships between "THINGS"
- Governed by business policies and practices
- A live collaborative artefact
- Refined & Updated throughout the systems development
- Domain models/UML class diagrams are static in nature (Time invariant)

Two techniques are used to identify class :

- Technique 1: Brainstorming
    - 1. Identify a user and a set of use cases (business functions)
    - 2. Brainstorm with the user to identify THINGS involved when carrying out a use case – THINGS about which information should be captured
    - 3. Ask questions to identify things:
    - – Do the THINGS refer to any locations? (e.g. Warehouse)
    - – Do the THINGS refer to any roles involved? (e.g. Customer)
    - – Do the THINGS involve any events? (e.g. Shipment)
- Technique 2: List of nouns from a business document
    - Identify all NOUNS involved in the narrative requirements about the system
    - Role, Items, Events, Reports

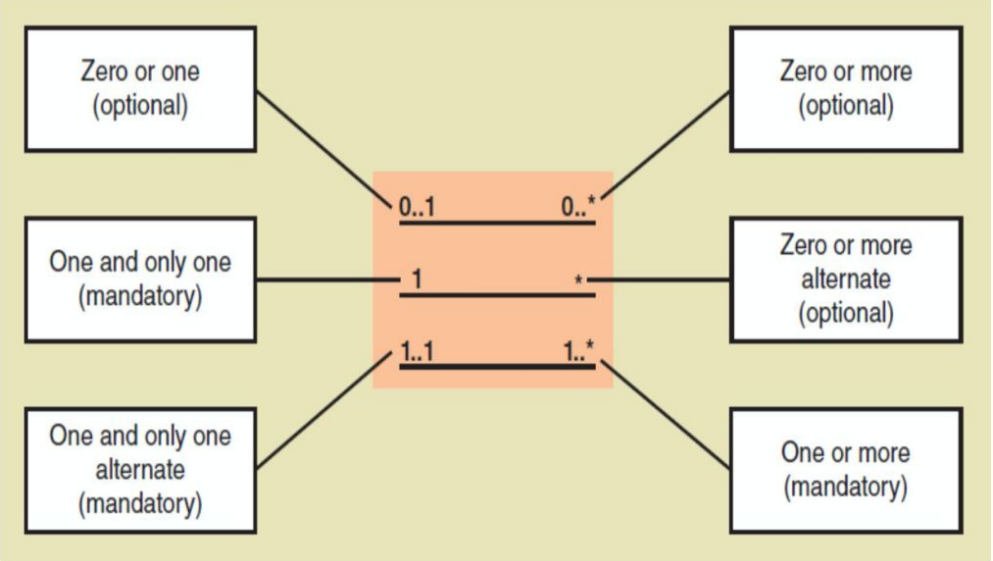
Association:

Association refers to "RELATIONSHIPS" that occur between classes (Things of interest) in a business context guided by business policies

- Key Characteristics
    - Characteristic 1: Classes are considered at the same level
    - Characteristic 2: One class having no more importance than the other

Multiplicity:

- Number of Association between classes established for each side of an association.
- These symbols indicate the number of instances of one class linked to one instance of the other class.



Characteristics (also called attributes) of class:

- Attribute→Represents a specific piece information about an object:
- Identifier→An attribute that uniquely identify an object
- Compound attribute→Contains a set of related attributes
- UML class diagram→Representing things as a class of objects
- Domain class has a name and attributes (no methods)
- Class name is always Capitalised
- Attribute use camelBack notation (words run together and second word is capitalised)

Types of Class:

- Abstract class
  - represents a generic class that allows subclasses to inherit characteristics (attributes) but has no instances of its own
- Concrete class
  - A class that has instances
  - It can be a subordinate (subclass) to an Abstract class
  - Inheritance: the concept that subclasses classes inherit characteristics of the generic superclass
- Association class
  - Used to capture certain characteristics of a relationship between 2 classes
  - But these characteristics belong to the relationship and not the classes.

Types of association (relationship):

- Hierarchical
  - Exists between subclasses classes (also called sub-ordinate) and Superclass
  - Often called an Inheritance Hierarchy
  - ‘is a’ relationship where subclasses inherit characteristics of the more general superclass
- Aggregation
  - one class is part of or a component portion of another class.
  - It represents a ‘has-a’ relationship
  - The component part can exist separately and can be removed and replaced
  - A unfilled diamond is used
  - Key Characteristics:
    - Aggregation relationship does not force ownership
    - Aggregation does not link the “whole” and “part” classes in such a way that if the whole is destroyed then parts are also destroyed.
- Composition
  - strong ownership, where there is a strong dependency between instances of the container class and instances of the contained class(es)
  - When the container is destroyed, normally each instance that it contains is destroyed as well.
  - A filled diamond is used