DES329 Final Summary

Systems Analysis and Design (SAD) – Onjira Sitthisak
SIIT DE-ASD Y3T2/2021 – By Paphana Yiwsiw (@waterthatfrozen)

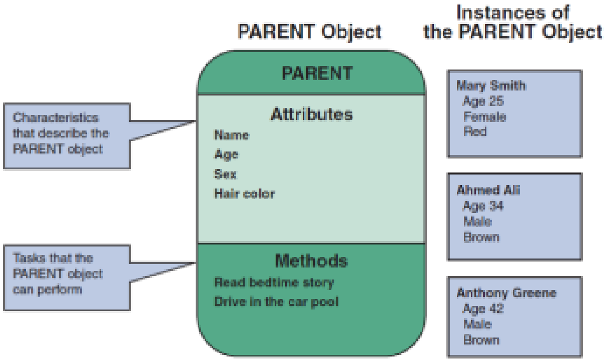
Lecture 6 - Object Modeling

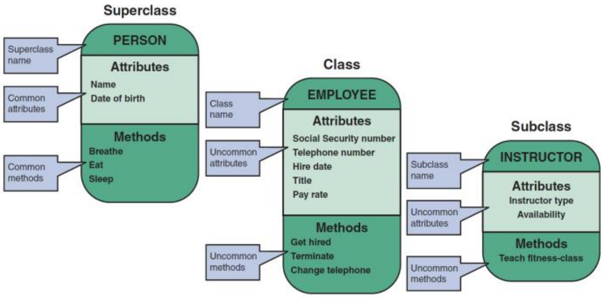
Object-Oriented Analysis

- Popular methodology because it can easily integrate with OOP such as C++, Java, Python.
- It is modular, reusable, easy to maintain.
- O-O analysis product is object model which represents IS in O-O objects and concepts.

Objects

- Person, place, event or transaction that is significant to the IS.



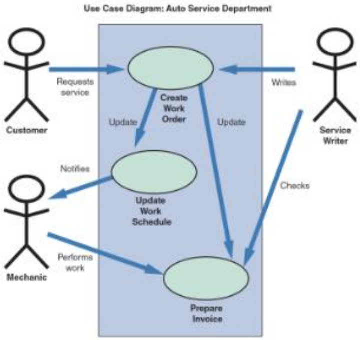
- **Attributes**
    - Describe the characteristics of an object, defined during the development process.
    - Object can have specific attribute called state that describes object's current status.
- **Methods**
    - Tasks or functions that the object performs when it receives messages.
    - Message is the command that tells an object to perform a certain method.
        - Polymorphism: message gives different meanings to different objects
        - Message to the object triggers changes within the object without specifying how the changes must be carried out.
            - Encapsulation is the idea that all data and methods are self-contained.
- **Classes**
    - An object belongs to a class
    - All objects within a class share common attributes and methods
        - Subclass: categories within a class
        - Superclass: class belonging to a general category.
        - 

Relationship Among Objects and Classes

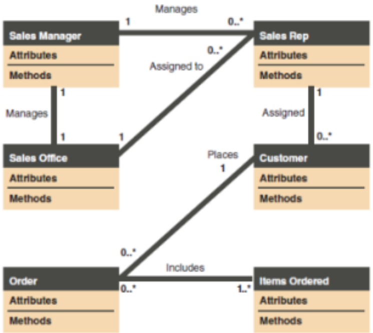
- Relationship
    - Describe what objects need to know about each other
    - Enable objects to communicate and interact as they function
- Inheritance
    - Strongest relationship
    - Enables an object to derive attributes from another object.

Unified Modeling Language

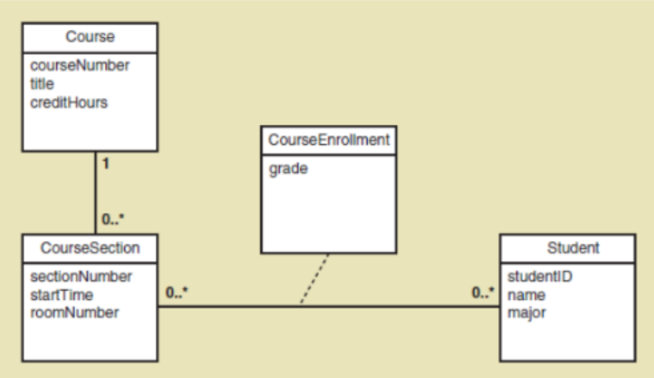
- Uses a set of symbols to represent components and relationship within the system
    - **Use case modelling**
        - Represent steps in business function or process
        - Actor is an external entity initiates a use case by request system to perform function or process
        - Use case description used to document the name, actor, and description of use case and provide step-by-step list of tasks and key descriptions and assumptions.
        - Use case diagrams is a visual summary of several related use cases within a system or subsystem.



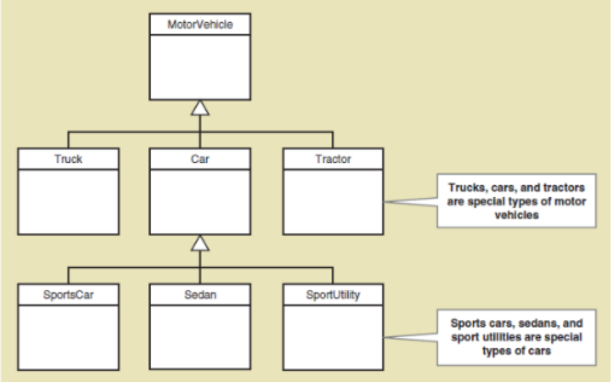
        - System boundary is represented by a rectangle which shows what is included in and not included in the system.
    - **Class diagram**
        - Show classes and relationships involved in a use case
        - Each class is represented by a rectangle: name, attributes, and methods
        - Lines between classes show relationships and labels identify related actions
        - Include concept of cardinality, describe how instances of class relate to another.
            - 0..* (zero or many)
            - 0..1 (zero or one)
            - 1 (one and only one)
            - 1..* (one or many)



- **UML domain class**
    - Class names and attributes use camelback notation
    - Words run together without space or underscore
    - Class name begins with uppercase letter; attribute begins with lowercase letter.
- **Association class**
    - Treated as a class in a many-to-many association because it has attributes that need to be remembered.



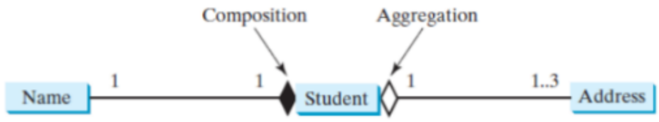
        - Reading associations from left to right: One course section has many course enrolments, each with own grade, applies to one specific students.
        - Reading associations from right to left: One student has many course enrolments, each with own grade, applies to one specific course section.
- **Generalization/Specialization Relationship**
    - Classify things in terms of similarities and differences
    - Generalization: group similar types of things
    - Specialization: group different types of things.



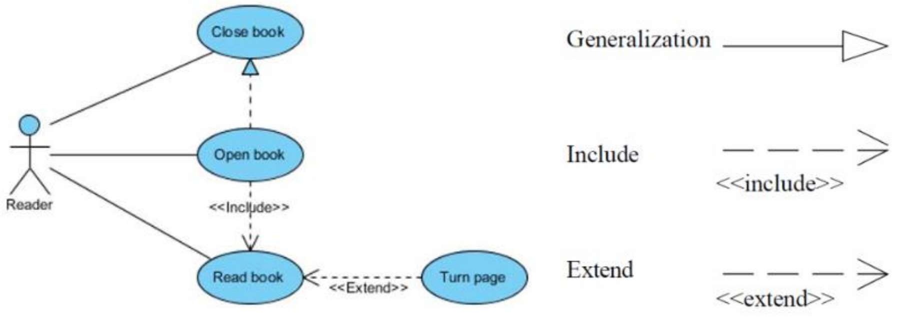
Trucks, cars, and tractors are special types of motor vehicles

Sports cars, sedans, and sport utilities are special types of cars

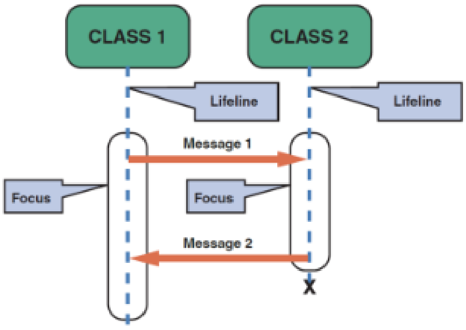
- **Whole-part relationship**
    - a relationship between classes that one is part of another class
    - Two types of whole-part relationship:
        - **Aggregation:** component part exists separately and can be removed and replaced. Ex. Car has wheels.
        - **Composition:** component part can no longer be removed. Ex. Hand has fingers.



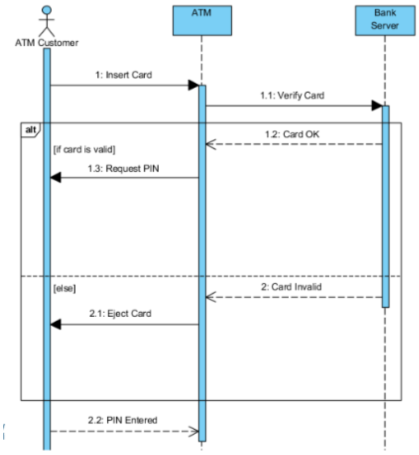
    - **Generalization vs Extend vs Include**
        - Generalization shows one is simply a special kind of another
        - Extend relationship indicates that one is another's variant, add actions. The use case is optional and comes after the base use case. Represented by dashed arrow in direction of the base use case with <<extend>>
        - Include relationship is used to extract use case fragments that are duplicated in multiple use cases, cannot stand alone, original use case is not completed without the included one. The use case is mandatory and part of base use case. Represented by dashed arrow with <<include>>



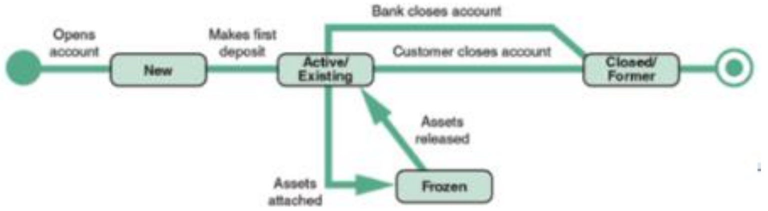
- **Sequence diagrams**
    - Dynamic model of a use case, showing the interaction among classes during a specfied time period.
    - Document the use case by showing classes, messages, and timing messages.
    - Include symbols that represent:
        - Classes: send/receive messages
        - Lifelines: time during which object aobe is able to interact with others
        - Messages: additional information about the contents
        - Focuses: object send or receive message.



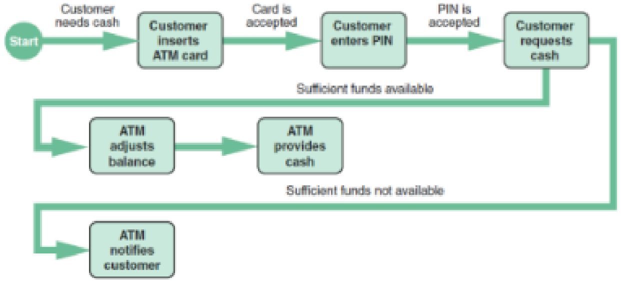
    - Example



- **State Transition Diagram**
    - Show how object changes from one state to another, depending on on events that afftect the object.
    - All possible states must be doucmented in the state digram.
    - State as rounded rectangle with state name inside.



- **Activity Diagram**
    - Show actions and events as they occur
    - Show the order in which the actions take place and identify the outcomes.



- **BPMN/Business Process Modeling Notation**
    - Represents the people, event, interactions in the system.
    - Can be used anytime in the development, compatible with object modelling.

Lecture 7 – User Interface Design

User Interfaces

- Describes how users interact with a computer system, features that affect two-way communications between user and computer.
- Central to usability: user satisfaction, support for business functions, and system effectiveness.
- In user-centered system, there is blurry line between input, output and interface because most of users work with many inputs, outputs, and data queries as they perform day jobs.

Human-Computer Interaction

- Relationship between the computer and user, includes all communications and instructions to enter input and obtain output in the form of screen displays or printed reports.

Seven Habits of Successful Interface Designers

- Understand the business: underlying business functions and how system support goals.
- Maximize graphical effectiveness, enables rapid learning if well-designed.
- Think like a user; must see from user perspective.
- Use models and prototypes; present initial designs as storyboard, test and provide feedback.
- Focus on usability; offer reasonable number of choices that can comprehend easily.
- Invite feedback; monitor usage, collect suggestions, observe features that used as intended.
- Document everything; document all designs for programmers to use later.

Guideline for User Interface Design

- Create an easy to learn and use interface
  - Focus on design objectives.
  - Create an easy to understand and remember designs.
  - Provide predictable and consistent commands, actions, response messages.
  - Allow users to easily correct their errors.
  - Label controls, buttons, icons clearly.
  - Select familiar images that user understands.
  - Provide clear concise, logical on-screen instructions.
  - Show all commands in a menu item lists, grey out unavailable one.
  - Make easy navigation and return to any level in menu structure.
- Enhance user productivity
  - Organize actual business operations' tasks, commands, groups functions.
  - Create alphabetical menu lists or place the selections used frequently at the top.
  - Provide shortcut for experienced users.
  - Use default values if majority of values in the field are the same.
  - Use duplicate value function but allow users to change it to their preferences.
  - Consider natural language feature that allows users to type commands or requests in normal text phrases.
  - Require user confirmation before data deletion and provide undo key.
  - Highlight the mistakes and errors in command from user input
  - Use links to assists users
  - Display message at logical places, display loading message when the processing too long

- Provide flexibility: offer several alternatives
- Provide users with help and feedback
  - Ensure that help is always available and provide with user-selected and context-sensitive help
  - Provide a direct route for users to back to the point where they left for help.
  - Include contact information
  - Allow message to remain on screen long enough so users can read
  - Let the user know the operational status of tasks such as successful or failed.
  - Use specific, understandable, professional messages.
- Create an attractive layout and design
  - Use appropriate colours to highlight screen in different areas. Some color that user used to are red = stop, yellow = caution, green = go.
  - Use fewer special effects and group objects and information that are related together.
  - Keep the screen uncluttered, display message and terms in consistent manner.
  - Ensure that commands and actions from different input will have same effects.
  - Require user to do the input confirmations
  - Provide keystroke alternative for each menu
  - Use familiar commands if possible such as copy for CTRL+C etc.
  - Avoid complex and technical terms.
- Enhance the interface
  - Opening screen is important
  - Use command button to initiate an action, add shortcuts.
  - Create customizable menu and tool bars.
  - If needed input data has many types, provide an explanation what is required.
  - Toggle button is easy to show the on/off status. Use dropdown box or list boxes to display the available choices.
  - When date input is required, use calendar.
- Focus on data entry screen
  - Use form filling when possible.
  - Restrict user access to screen locations where data is entered.
  - Provide leaving screen at any time without entering current input. (Go back)
  - Provide description of input fields and movement meaning among fields of the forms.
  - Allow users to CRUD records.
  - Design the form layout to match the source document
  - Display default value if any.
  - Display list of acceptable values and provide relevant error message if it is unaccepted.
  - Provide opportunities for user to confirm their input before submissions.
- Use validation rules
  - Sequence check: data must be in predefined sequence.
  - Existent check: data is required
  - Data type check: ensure that it is the required data type
  - Range check: used to verify data items fall within the acceptable range.
  - Reasonableness check: input data must be reasonable.
  - Validity check: data must have certain values.
  - Combination check: two or more fields are consistent with another when combined.
  - Batch control: totals used to verify the batch input.

- Manage data effectively
    - Data management impacts company efficiency, productivity, and security.
    - Enter and verify data as soon as possible
    - Each data item should have specific type
    - Collect input data as close to its source as possible.
    - Data is entered only once for more efficient design.
- Reduce input volume
    - Input necessary data only, no constant data input.
    - Do not input data that system can calculate or recall within the system.

Source Documents and Form Design

- GIGO/Garbage in, garbage out: Quality of the output depends on inputs.
- Source documents: collect input data, trigger action, provide a record of original transaction.
- Good form layout is easy to complete and provide enough space, order and placement is logical.

Printed Output

- Questions to ask before design printed output:
    - Why deliver as printing? / Who wants it? / What to be include? /
      Will it print on specific devices? / When and how often must be updated? /
      Do security or confidentiality issue exist?
- Report design
    - Organization wants to reduce flow of paper and printed reports
    - Use it as turnaround documents
    - Must be easy to read and well-organized.
    - Principles: Report and page header & footer / Repeating fields / Consistent design.
- Types of reports
    - Detailed: one or more lines of output for each processed record, lengthy
    - Exception: display only record with specific conditions.
    - Summary: reports that provide comprehensive data

Technology issues

- Output technology
    - Output can be delivered in many ways such as internet-based information delivery which allow users to download files to support their needs, pre-recorded video, email, blogs, instant message, wireless devices, audio, images, video, podcasts, etc.
- Input technology
    - Batch input: data entry performed at specified time
    - Online input: enable immediate validation and availability of data.
    - Decision depends on business requirements

Security and Control Issues

- Output security and control
    - Maintain output integrity and security of company, protect privacy rights
- Input security and control
    - Ensure that data is correct, complete, traceable, and secure.
    - Data security policies and procedures protect data from loss and/or damage.

Emerging Trends

- Modular design: module for small components, connected together.
- Responsive web design: GUI represented on different devices handled automatically.
- Prototyping: involves analysis, design, modelling, and testing
- System prototyping: full-featured, working model of IS
- Pros and Cons:
  - Pros: avoid misunderstanding / create accurate specification based on prototype / evaluate working model more effectively / help develop testing and training process / reduce risks that occur when a finished system fails to support business needs.
  - Cons: rapid pace can create quality problems / requirement cannot tested adequately using prototype / prototype can become difficult to manage in complex system.

Good Luck!