OOAD

Key concepts

Nguyễn Anh Hào
0913609730 – nahao@ptithcm.edu.vn

Content

1 The system ?
2 System thinking
3 Modeling, Structured Approach (SADT)
4 OO Approach & OOAD

1. What is a system ?

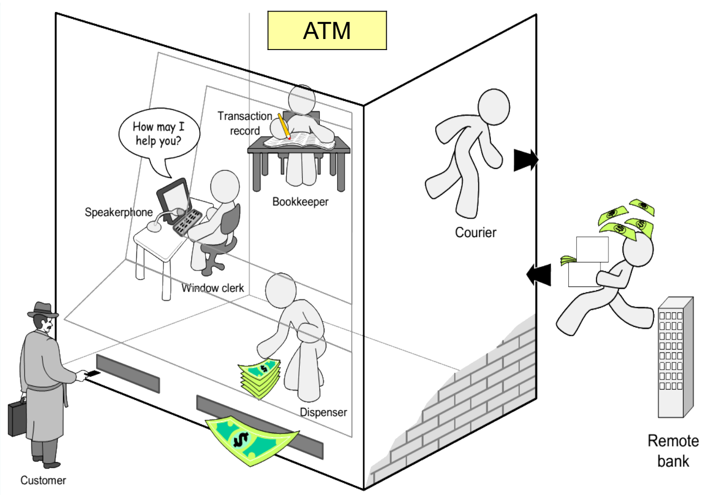
- There are many things called systems: electrical systems, transportation systems, education systems, etc. So what characteristics do these systems have in common?
  - All created by people intentionally, with a purpose.
  - There are many parts that compose in some way. Unlike a bag containing many items: If you arbitrarily remove or repair a part, the system will be damaged.
- Is refrigerator a system ?
  - a) No, if one simply uses its functions for some benefits → it is a tool.
  - b) Yes, if it is damaged, and should be repaired → we must find out which part is damaged in oder to fix it → it is considered as a system.

System definition

- A system is a collection of many components that work together to perform one (or several) system functions.
  - Each component of the system is not capable of performing all the system's functions on its own.
  - Collaboration (coordination) between components helps the system perform its functions.
  - It is an inevitable relationship between components, in which each component has its own capability, forming the concept of "collaboration" or "a system".

Example: ATM

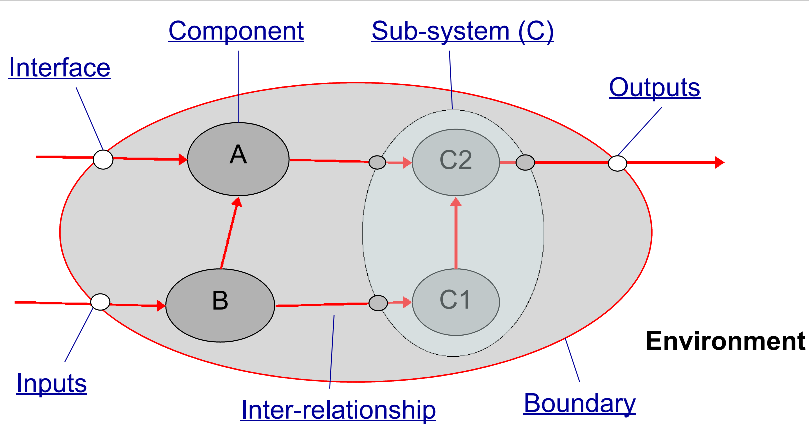
For the customer:
ATM is just a tool.



ATM components:

Window clerk | Book-keeper | Dispenser | Courier

The system attributes

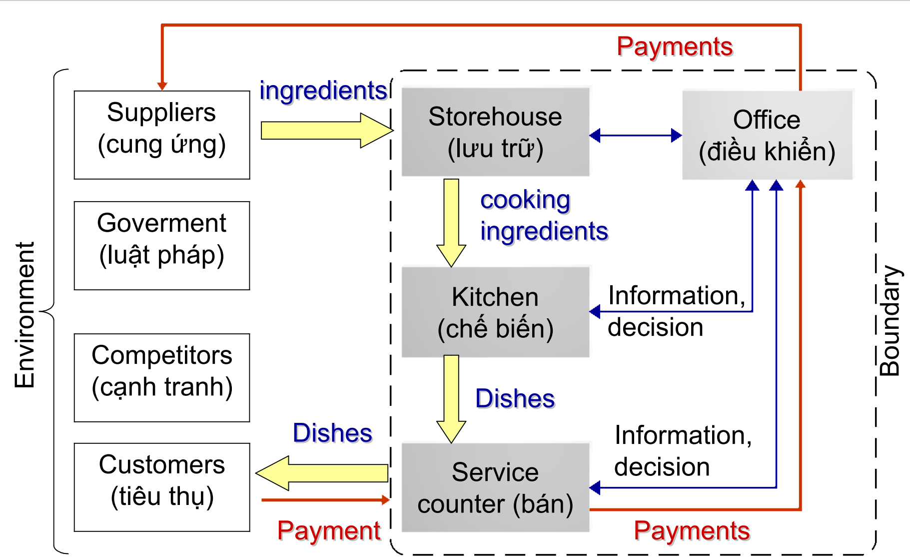


Each component is a "system tool" for the system (namely A,B,C,..). A system tool can be a subsystem (C includes C1,C2). The functions that C performs for the system will be performed by two C's components C1 and C2 working together.

The system attributes

- The environment is what lies outside the system's boundaries and interacts with the system (example, user). Any system created by humans must have benefits for use (through its functions); that is the system's responsibilities in its working environment.
- The system's functions transform inputs into expected outputs; they are performed by collaboration components within the system.
  - Inputs = things that the environment provides via system interface
  - outputs = what the environment wants to get via system interface
- Collaboration is an inter-relationship.

Example: Restaurant is a system



Example: Sub-systems

- If we consider the restaurant as a system, then the storehouse, kitchen, service counter, and office are all sub-systems of the restaurant.
- If the storehouse has a storehouse management software system for storehouse keeper, then the storehouse is a subsystem with 2 components: storehouse management system (automatic), and storehouse keeper (labor).
- Storehouse management system is also a sub-system including computers (hardware) and storehouse management software (programs, databases) to provide information to storehouse keeper.
- Storehouse management software (SW module) is also a sub-system.

Example: Food Ordering System

- **Types of IS**: TPS = record order & pay, MIS = statistics, DSS = Suggested food combo
- **Basic Components** : Hardware = server machine, Software = app, Data = order details, People =customers & staff, Process = order process → food process → food deliver.
- **Structured to OOP**: single function with data → Order class contains item info (attribute) + confirmation (method).
- **Encapsulation**: Customers do not know the kitchen details, just order and receive food.
- **Inheritance** : The Pizza class inherits from the Food class.
- **Relationships** : class Order “has-a” list of MenuItem, class Customer “places” Order...
- **Reuse** : Google Maps API integration,..

2. Principles of Software Development

1. Keep it Simple – avoid unnecessary complexity
2. Modularity – break down into manageable parts
3. DRY – Don't Repeat Yourself
4. Maintainability – easy to update and fix
5. Scalability – can handle growth

System thinking

View the system as a subsystem of a larger system to define its necessary functions for the larger system which is its <u>environment</u>.

1 What is the role of the system in the environment ?
   – Consider the system as a tool used in its environment

2 What does the system need to do for this role?
   – Define system's functions in its environment

3 How is each of these functions performed ?
   – Define components, roles and their collaboration

4 What does each component need to do for the system?
   – Define component's functions for the system

*** a component is also a subsystem (this is a recursive thinking process).*

System's environment

- A system is a subsystem of a larger system, called environment.
- Environment determines the existence of the system. For example: system = restaurant/company/hospital:
  - The system has an accepted responsibilities in its environment (providing food/goods/treating disease).
  - The system needs many things from its environment to survive (money, electricity, water, human resources,...)
  - The system needs to change according to the changing environment.

System creation & modification

- Reasons for creation and modification
  - People need new tool
  - People need to improve existing tool
- To create a good tool, we should:
  - consider the tool's functions if they satisfy demands of working environment.
  - consider the tool as a system (consider its internal structure), for fabrication or repair.

System analysis & design

15

- System analysis design is a series of processes of creating or modifying a system in a controlled manner

- System analysis: Is the process based on evidence (data obtained from reality) to accurately determine the requirements for the system
- System design: Is the process of determining the necessary coordinate components to solve its requirements.

- Meaning: the system can create a positive impact (benefit) to the current environment (make the environment better).

System understanding

16

- Understanding a system is a process of collecting information ("know") and systematizing the information in order to explain the necessary structure of the system ("understand").

- From there, we can determine exactly the system's problem: what needs to be fixed, added or removed. It is the most important and most difficult job, because we have to think about how the system is created.

- There are so many things must be known and understood about the system, for example: its functions, components, inter-relationships,... So, where should we start from, and how to do ?

System thinking in System AD

1 Consider the system as a tool, learn about real-life situations that need helps from this tool.
2 Systematize (link) everything known, summarize with models (e.g. draw modeling diagrams).
3 Analyze the models (diagrams) to determine the required internal structure of the system.
   a. What requirements do these situations place on the system (system's responsibilities in the environment)
   b. How the system interacts with the environment to fulfill its responsibilities.
   c. What are necessary components in the system.
   d. How to coordinate system components to perform system interactions.

3. Modeling & Approaches

- A model is a means to 'summarize' the important characteristics of a system (images, formulas, etc.)
  - For example: map, diagram, flow-chart...

- Models help us generalize everything to easily focus and understand.
- The model is based on grammar, semantics and context.
- Modeling is the creation of models for the real world.
  - For example: drawing flowchart
- An approach is a way to understand and model a system. Each approach has different modeling methods. Common approach for System AD:
  - Structured Analysis Design Technique
  - Object Oriented Analysis Design

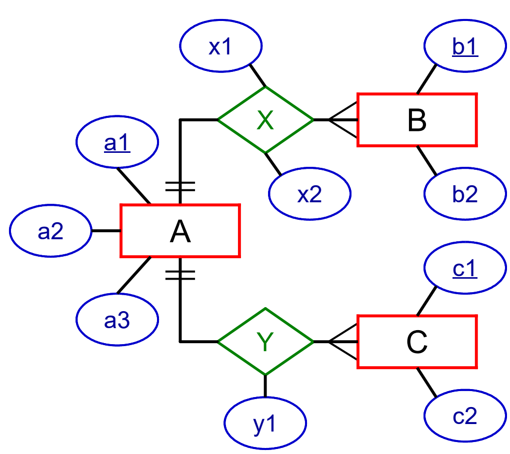
Structured Approach

- The structural approach is a way of analyzing and solving problems based on logical thinking.
  - Completely understand the problem (analysis) to find a solution (design) that satisfies all known requirements and constraints.
- View the system: data and process.
  - Data: Abstracts real-world objects into notions of entities and relationships (ERD schema) that have the necessary properties for the system's data collection, storage, and processing
  - Process: Defining and decomposing the system's process to a level that simple enough to properly understand (DFD diagram)

ERD: Entity Relationship Diagram

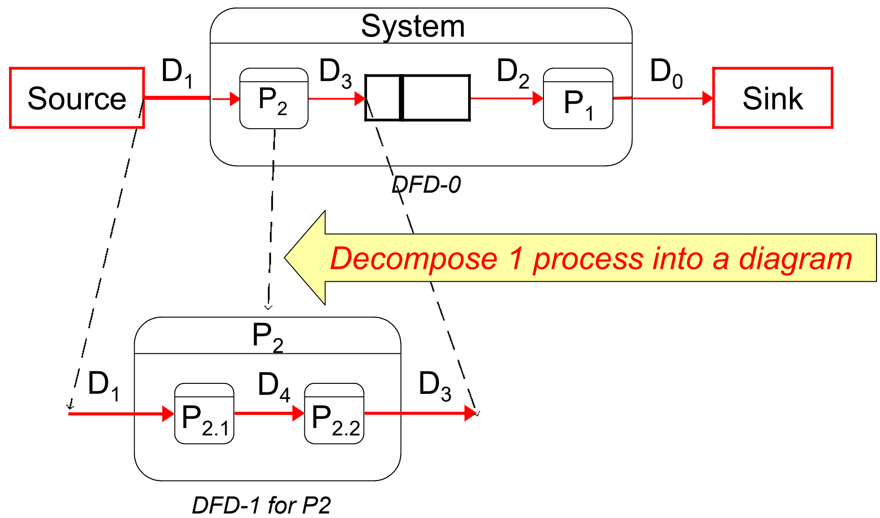
❖ ERD is used to define the concepts of system data

1. Entities
2. Relationships
3. Entity Attributes
4. Entity keys
5. Cardinality
6. Relation types



DFD: Data Flow Diagram

❖ DFD is used to define system processes.



4. Object Oriented Approach

- *Problems arise from the **real world**, and the real world always has (or has potential) solutions to the human problems.*
- The overall philosophy of OOAD is to define a software system as a collection of real-world objects of various types that interact with each other through well-defined interfaces
  - Find the objects from the 'real-world' that have capacity to work as necessary components of the system;
  - Assign responsibilities to each object component
  - Define how component objects interact/collaborate to perform system functions

Object-Oriented Analysis, Design and Implementation: Introduction, Page 3.

OO Approach vs Structured Apporach

- **Structural orientation** is based on decomposing the system/subsystem problem into small problems that must be solved (process), then finding solutions (designing processing functions) for the problem from theory.
- **Object orientation** solves system problems by searching objects from the real world (or imitating them) that can participate in the system, and coordinate them together to to solve the problems in usecases (collaboration).

Objects in the OO approach

- An object is something in reality that humans recognize: the name, properties, and behaviors.
  - Classification clearly highlights an object's ability to perform its role in the system; and inheritance creates flexibility in design.
- Describing (or modeling) objects is based on principles of object-oriented approach

OO: Principle 1

An object is described by its class

- Classification of objects is the simplest way to know about objects.
  - For example: Bob is a dog, so he can bark, because dogs can bark
- In classification, the properties and behaviors of objects belonging to the class all share the same properties and behaviors of the class.
- The object class is also classified into a more general class (superclass); This is a generalized concept.
  - For example: doctors are employees of the hospital, employees are citizens (so doctors are also citizens).

OO: Principle 2

Object has inheritance rights

- Every child class inherits everything from the parent class; including relationships of the parent class.
- A child class can inherit from many parent classes: that is multi-inheritance.
  - For example, a programmer class inherits from two classes: employee (name, age) and programming profession (can write code).
- Inherited behavior can be changed in the child class to make the behavior more sophisticated, which is polymorphism in inheritance.

OO: Principle 3

Objects have their own freedom to develop

- Encapsulation protects this right; It separates two views of the object:

1 Outside view: other objects only know the services & properties that an object provides (what it can do) and they cannot know how it does.

2 Inside view: the private properties and behavior are hidden; Objects can freely change their private behavior and properties without affecting other objects.

OO: Principle 4

The object decides ifself how to respond to the requests

Tom & Mary are eating in a restaurant. Tom needs to ask Mary to help him get the salt shaker next to her.

What will Tom say to Mary ?

- (1) "Give me the salt shaker"
- (2) "Take your hand off your glass and reach for the salt shaker, take it and hold it towards me until I can hold it"

(1) is a request (a message carrying a service request) used in object orientation. Everything that follows is decided by the recipient. Mary has the right to refuse, or ask someone else to do it for her (→ trust mechanism).

(2) is a detailed & precise request for the action that needs to be done (and cannot be done otherwise), used in the structure-oriented approach (→ command mechanism).

OO: Principle 5

An object's behavior depends on its state

- The specific value of the attribute determines the state of the object. An object's state is a set of its attribute values, for example: an object has 2 attributes A and B, a and b are 2 data values of A and B then 1 state of this object is (a,b). Objects have many different states. For example: a traffic control light pole has 3 states: Green, Yellow, Red.
- The change in the object's state is due to the object itself reacting to trigger events (in the light pole is the timeout signal of each color). The transition to a new state (S_2) is determined by two factors: the current state (S_1), and the trigger event e: $S_2 = \delta (S_1,e)$, δ is called a state transition function (hàm chuyển trạng thái)

OOAD Benefits

1. Objects often reflect real entities in application systems. This makes it easier for a designer to come up with classes in the design (encapsulation), easier to understand.
2. It is possible to isolate the varying parts of a system into classes, and consider changes in these classes only (small scope of change). It reduces the risks involved in system development.
3. Helps increase productivity through reuse of existing software components (packages): Inheritance makes it relatively easy to extend and modify functionality provided by a class (modularity

OOAD Processes

1 OO Analysis process

- Consider the system as a tool, identify situations in the environment where the system is used (usecases) by several other objects (collaborate with it) to solve that situation.
- Describe the system's support in each usecase using UML diagrams/schemas (usecases' scenario).
- Find the necessary component objects for the system to participate in processing usecases, and describe them into class and state diagrams.
- Based on the drawn schemas for usecase, objects' methods and properties are defined (class detail)

2 OO Design process

- Concretize the conceptual objects for the system into design objects (stereotype: interface, process, entity)
- Defines interactions between design objects for collaboration and minimizing dependencies, based on principles (e.g. SOLID, reuse, standards,...)

OOAD UML diagram supports