SOFTWARE ENGINEERING

Chapter 5 – System Modelling

Topics covered

- Context models
- Interaction models
- Structural models
- Behavioral models
- Model-driven engineering

System modeling

- the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- helps the analyst to understand the functionality of the system and models are used to communicate with customers.

Existing and planned system models

- Models of the existing system
  - are used during requirements engineering
  - to help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses
  - then lead to requirements for the new system.
- Models of the new system
  - are used during requirements engineering
  - to help explain the proposed requirements to other system stakeholders
  - Engineers use these models to discuss design proposals and to document the system for implementation.

System perspectives

- An external perspective
  - model the context or environment of the system.
- An interaction perspective
  - model the interactions between a system and its environment, or between the components of a system.
- A structural perspective
  - model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective
  - model the dynamic behavior of the system and how it responds to events.

UML diagram types

- Activity diagrams
  - show the activities involved in a process or in data processing .
- Use case diagrams
  - show the interactions between a system and its environment.
- Sequence diagrams
  - show interactions between actors and the system and between system components.
- Class diagrams
  - show the object classes in the system and the associations between these classes.
- State diagrams
  - show how the system reacts to internal and external events.

Use of graphical models

- As a means of facilitating discussion about an existing or proposed system
  - Incomplete and incorrect models are OK as their role is to support discussion.
- As a way of documenting an existing system
  - Models should be an accurate representation of the system but need not be complete.
- As a detailed system description that can be used to generate a system implementation
  - Models have to be both correct and complete.

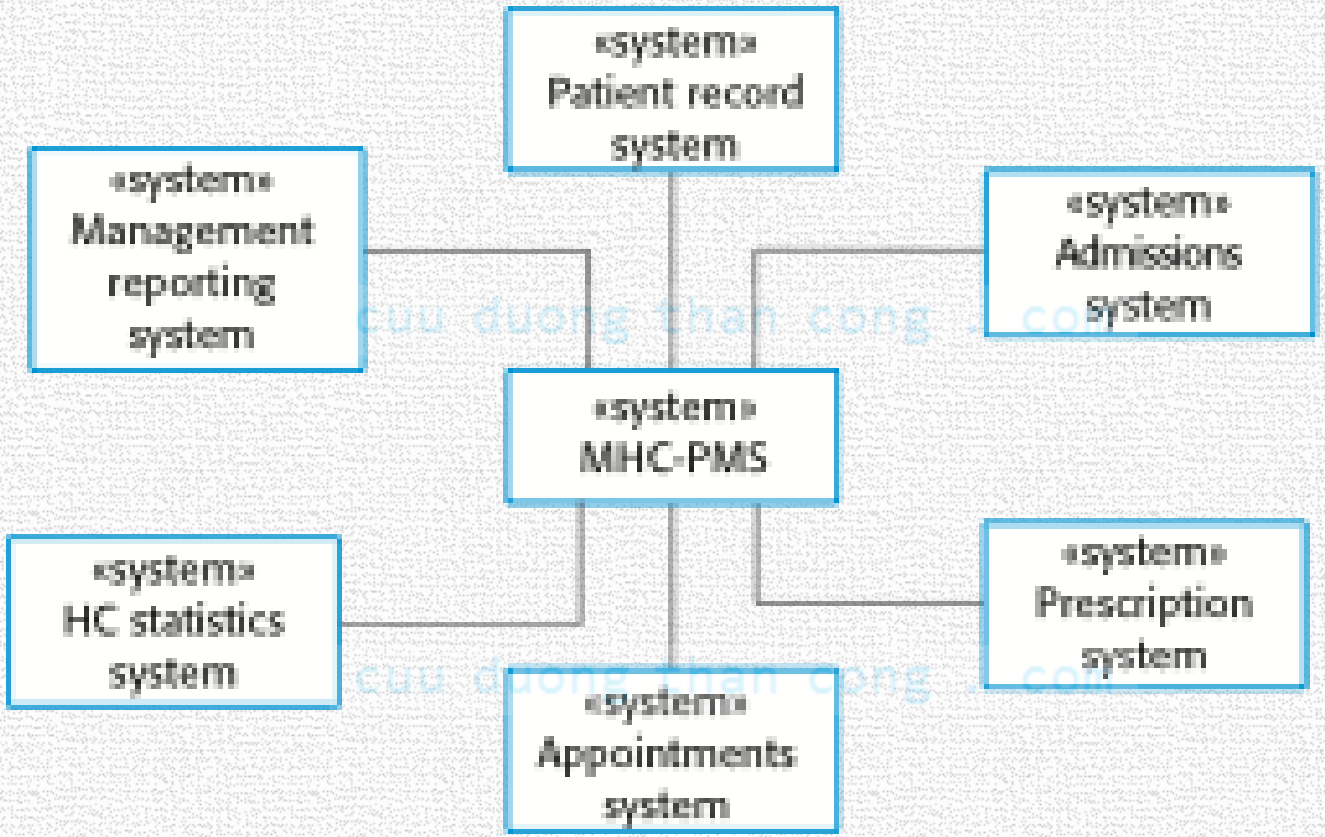
Context models

- Illustrate the operational context of a system
  - show what lies outside the system boundaries
- Social and organisational concerns may affect the decision on where to position system boundaries
- Architectural models show the system and its relationship with other systems.

System boundaries

- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

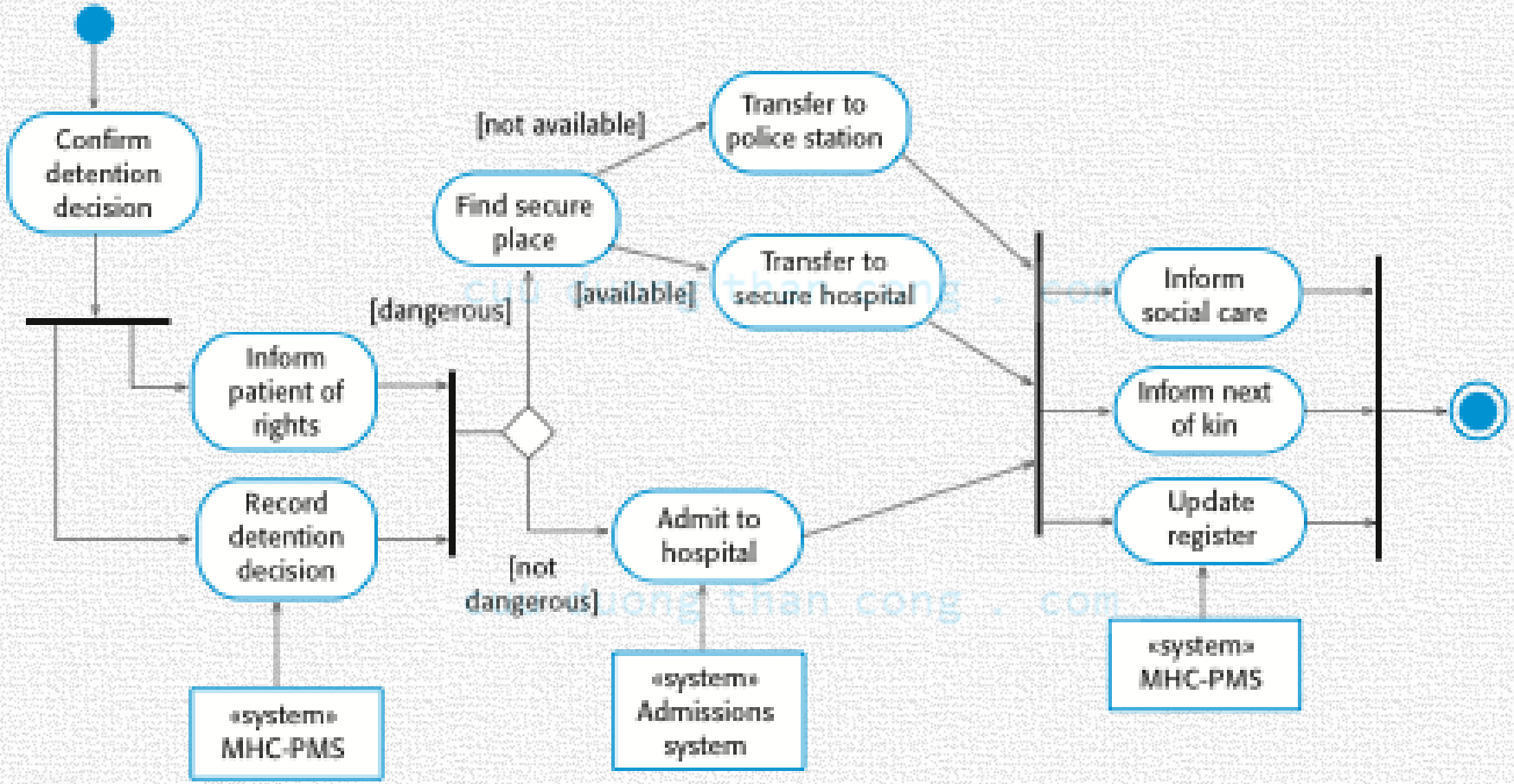
The context of the MHC-PMS



Process perspective

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.

Process model of involuntary detention



Interaction models

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

Use case modeling

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagramatically to provide an overview of the use case and in a more detailed textual form.

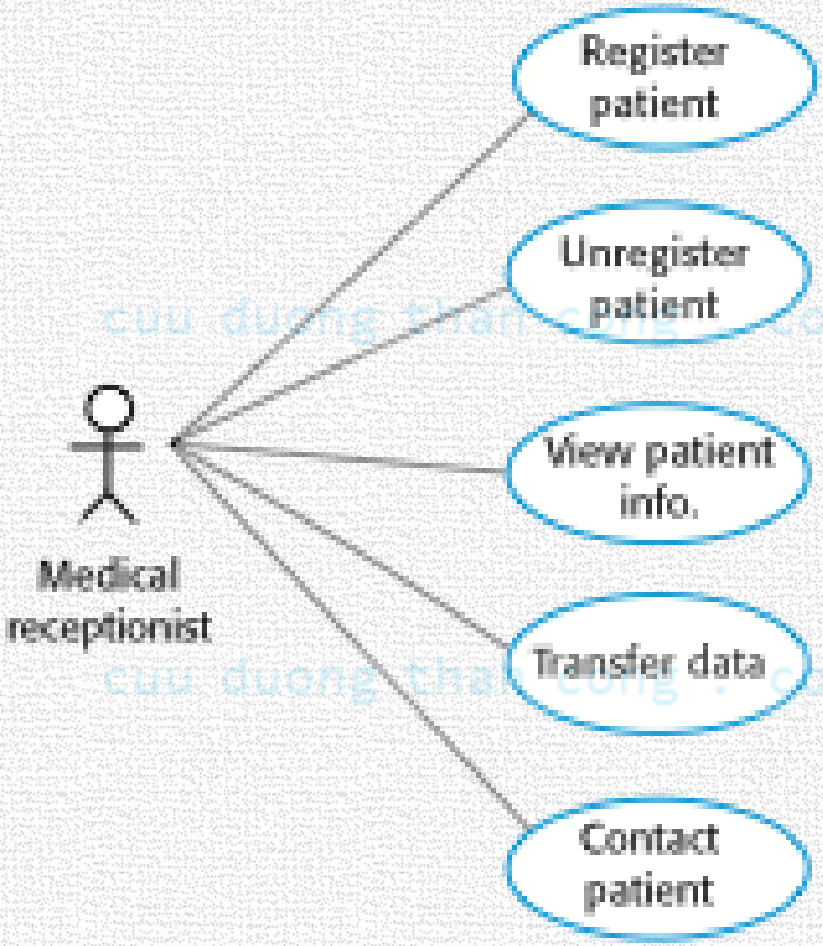
Transfer-data use case

- A use case in the MHC-PMS

Tabular description of the 'Transfer data' use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Use cases in the MHC-PMS involving the role 'Medical Receptionist'



Alternative Use-Case Scenarios

- Scenario
  - A single sequence of steps
  - No branching!!!
- Ex: Login use-case
  - Main scenario
    1. User enters username and password
    2. User click “Login” button
    3. System validates the username and password are correct
    4. System change status of the user to “logged in”
  - Alternative 1

    4A. [Validating is not passed] System display an error message
  - Alternative 2

    2A. User click “Cancel” button

cuu duong than cong . com

Alternative Use-Case Scenarios

Use Case ID:			
Use Case Name:			
Created By:		Last Updated By:	
Date Created:		Date Last Updated:	

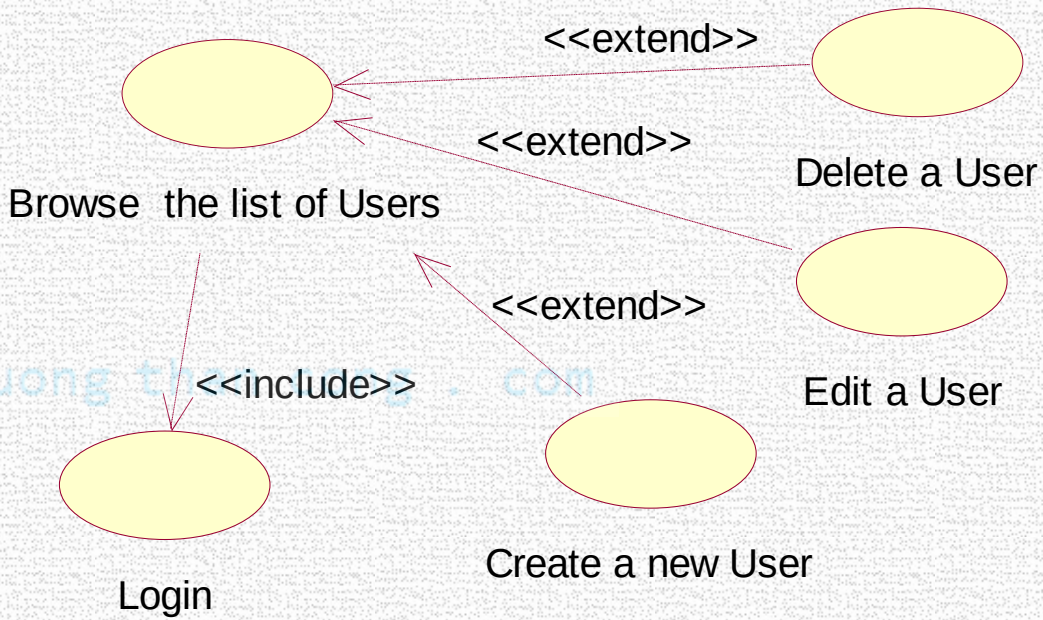
Actors:	
Description:	
Trigger:	
Preconditions:	1.
Postconditions:	1.
Normal Flow:	1.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Alternative Use-Case Scenarios

- Actor: A member of the public (MP)
- Use case: The MP is searching for club events on a particular date.
- Preconditions: The MP is at the CIS home page, but not logged in as a member.
- Scenario A:
  - 1. MP selects “Search Events” on MP home page
  - 2. System presents a page with choice of dates for the current month
  - 3. MP selects a date from among the choices
  - 4. System presents a page with events for that date, giving time and club name
  - 5. MP selects an event
  - 6. System presents a page with details of that event, including location, description and cost
- Exception:
  - 4. If there are no events for the selected date, System presents a page saying that there are no events for the selected date
- Alternative Scenario A1:
  - 3a. MP selects a different month
  - 3b. System presents a page with choice of dates for the current month

Advanced UML notations

- Extend and Include
  - Ex: Manage Users

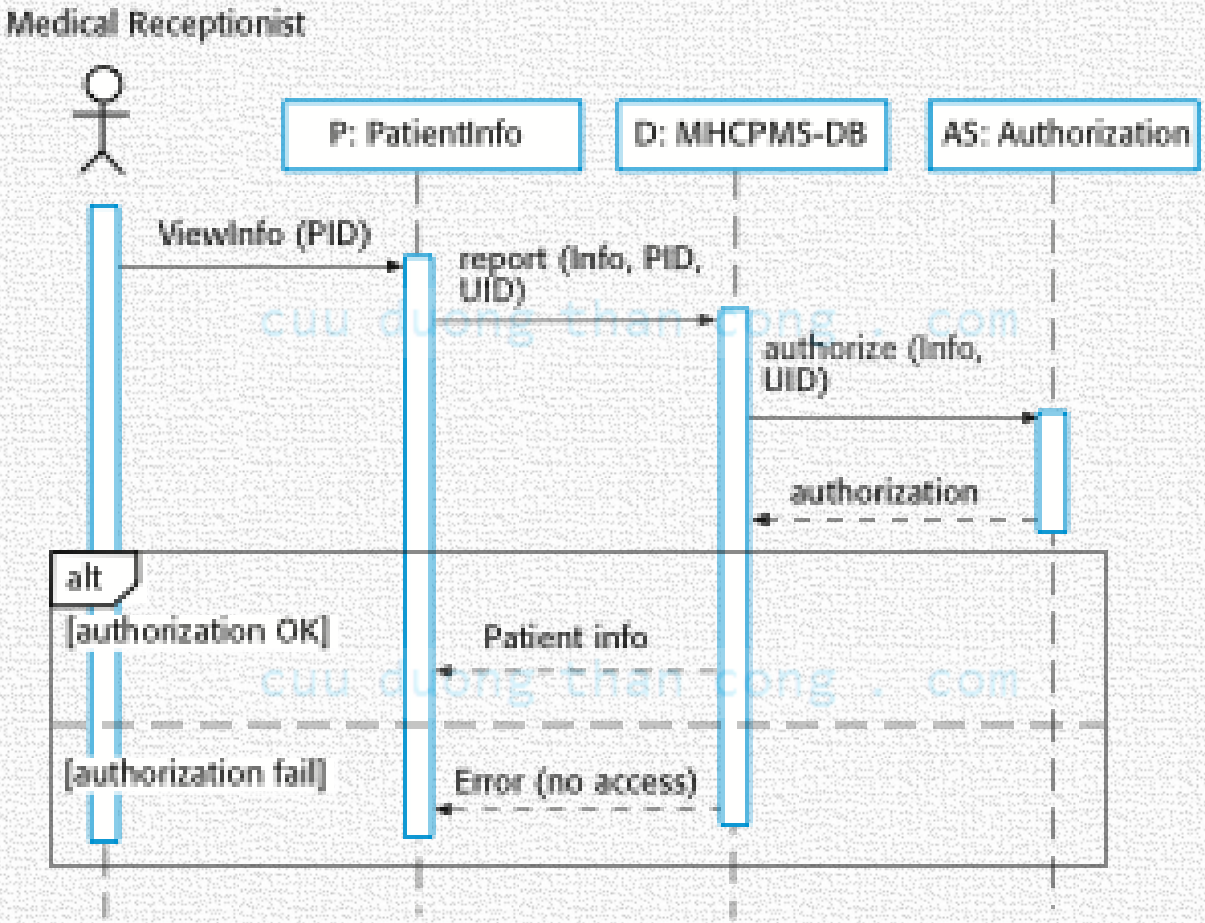


- Assumption
  - Ex: Change User Name user-case
    - Assumption: User is logged in
    1. User click “Change Name” menu item
    2. System display “Change Name” form
    3. User enters a new name
    4. User hits “Confirm” button
    5. System change the user name to the new name

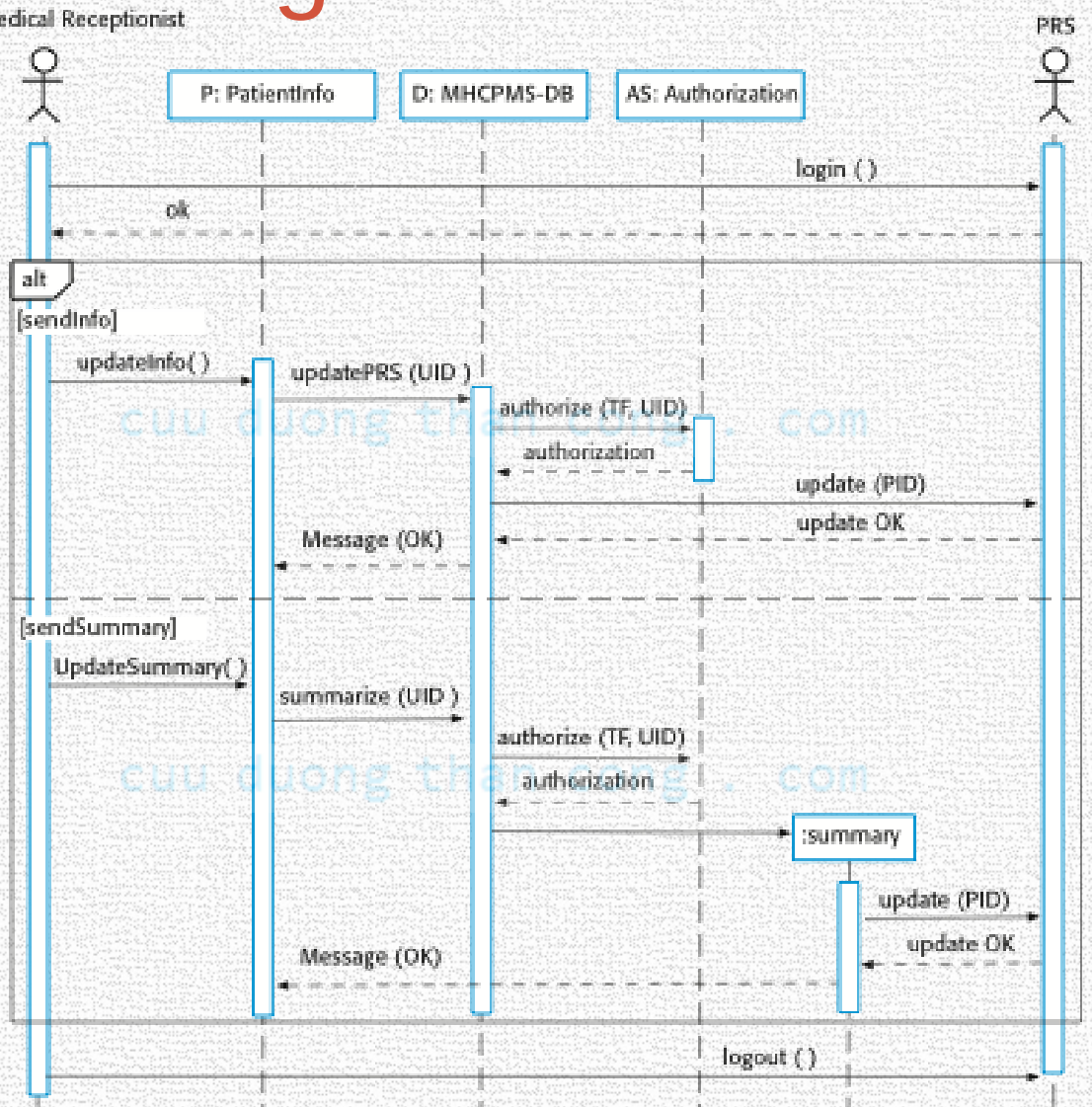
Sequence diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence diagram for View patient information

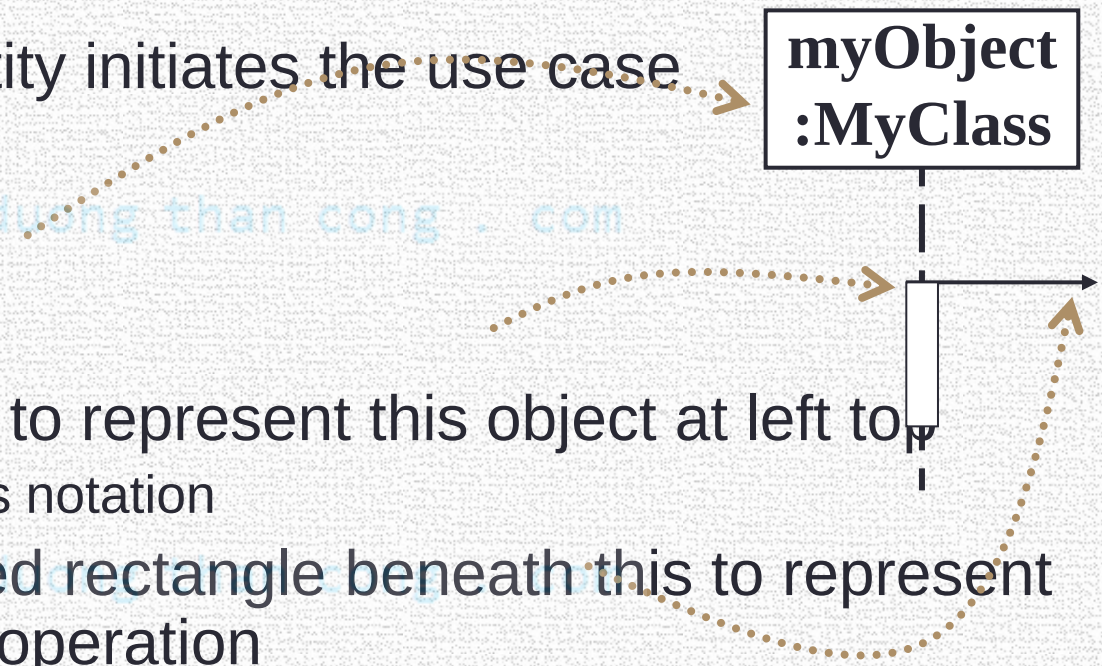


Sequence diagram for Transfer Data



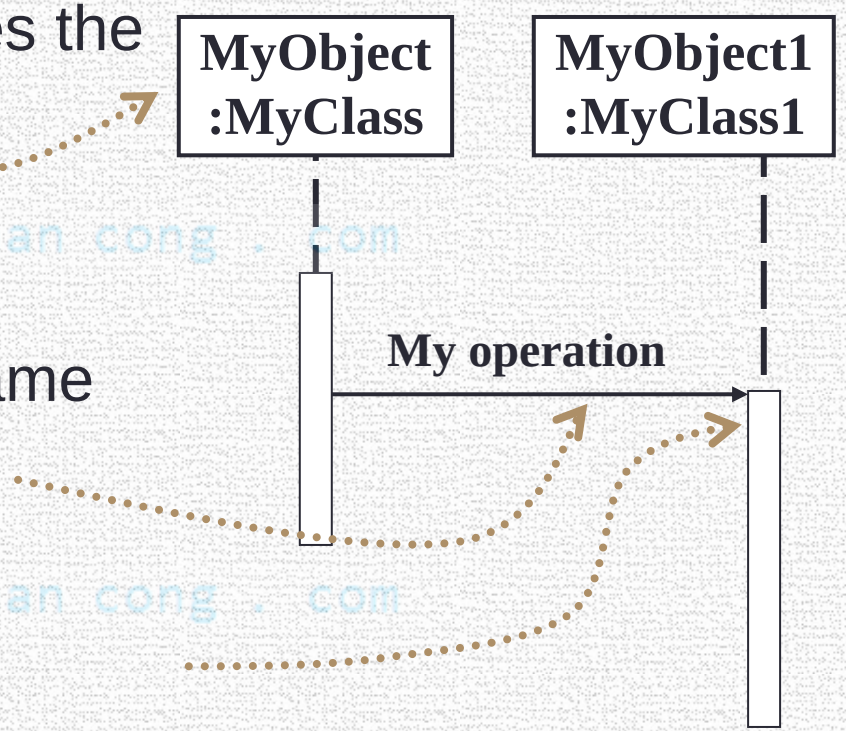
Build a Sequence Diagram

- 1. Identify the use case whose sequence diagram you will build
- 2. Identify which entity initiates the use case
  - the user, or
  - an object of a class
    - name the class
    - name the object
- 3. Draw a rectangle to represent this object at left to
  - use UML object:Class notation
- 4. Draw an elongated rectangle beneath this to represent the execution of an operation
- 5. Draw an arrow pointing right from it to indicate invoked functionality



Build a Sequence Diagram

- 6. Identify which entity handles the operation initiated
  - an object of a class
    - name the class
    - name the object
- 7. Label the arrow with the name of the operation
  - don't show return?
- 8. Show a process beginning, using an elongated rectangle
- 9…… Continue with each new statement of the use case.



Structural models

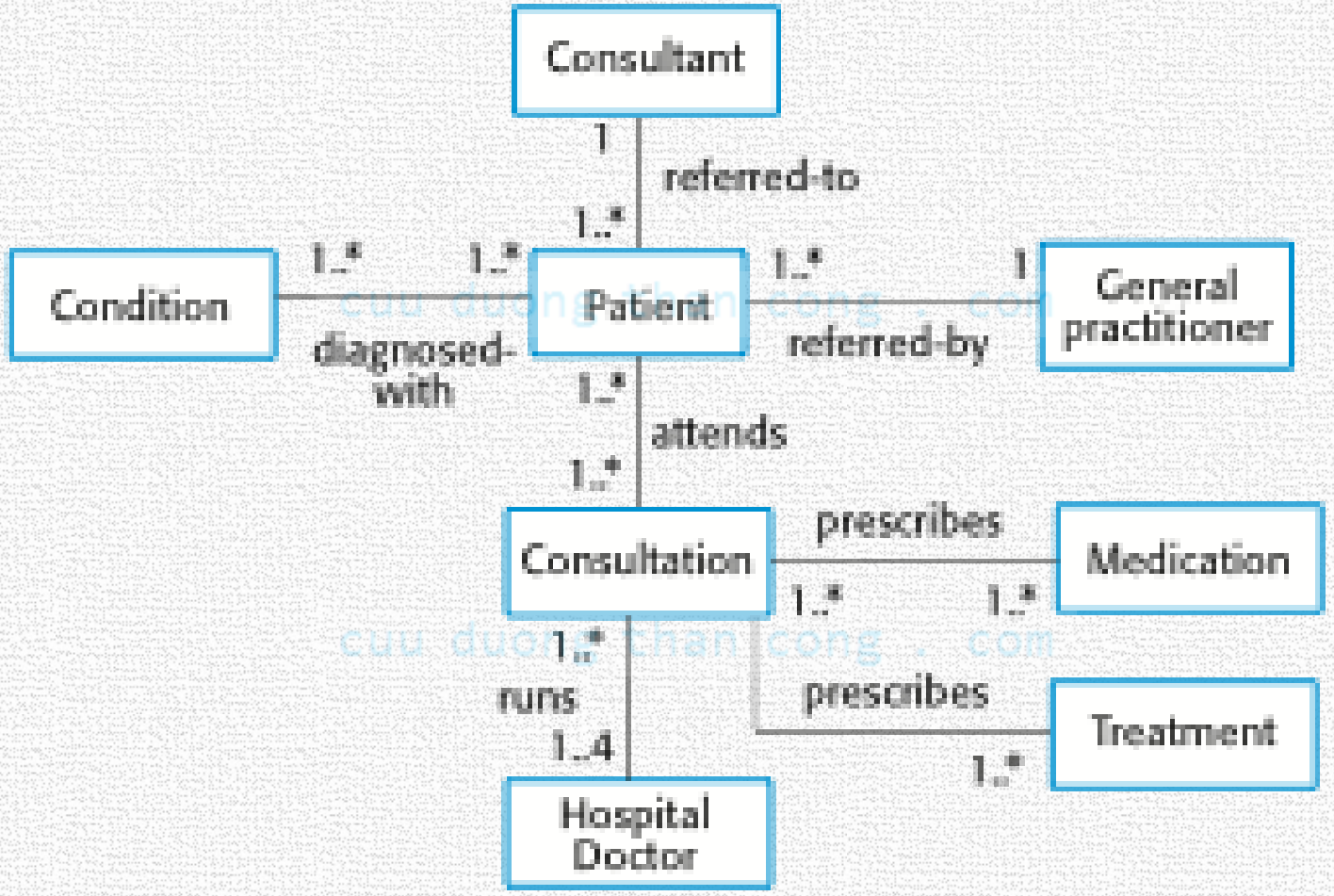
- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.

Class diagrams

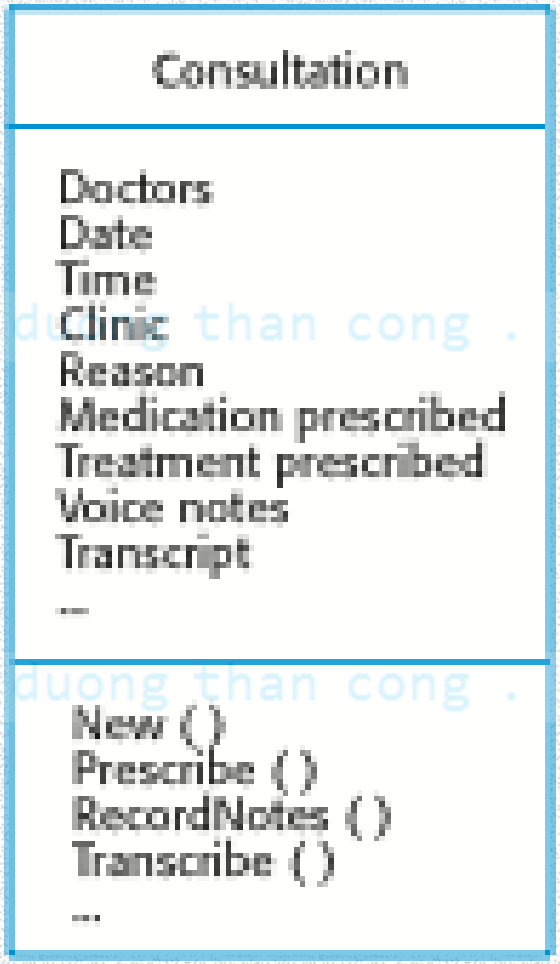
- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An object class can be thought of as a general definition of one kind of system object.
- An association is a link between classes that indicates that there is some relationship between these classes.
- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

UML classes and association

Classes and associations in the MHC-PMS



The Consultation class



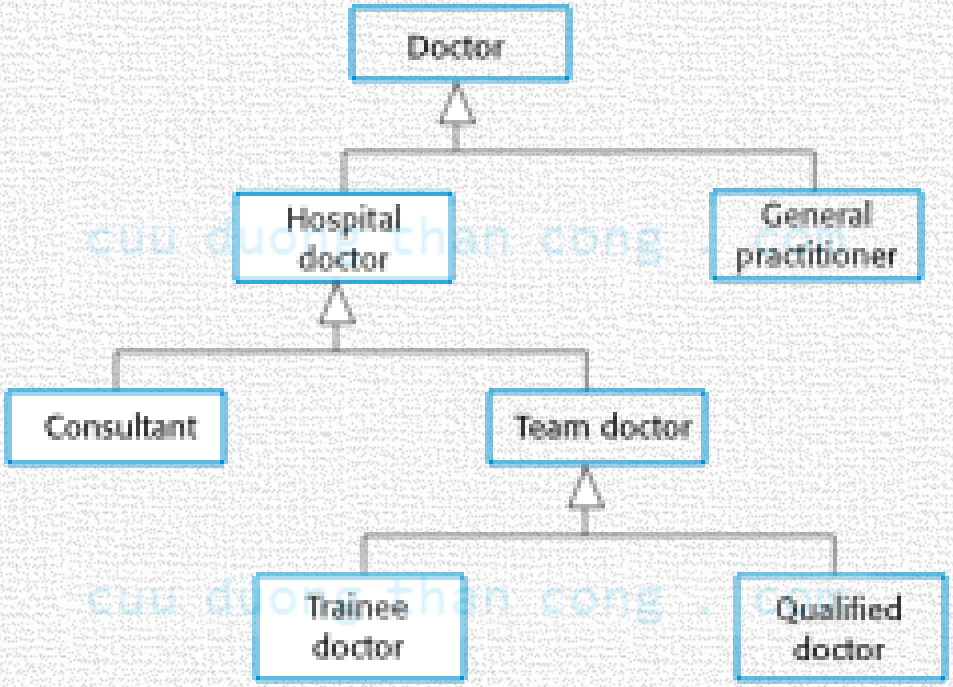
Generalization

- Generalization is an everyday technique that we use to manage complexity.
- Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, etc.) and learn the characteristics of these classes.
- This allows us to infer that different members of these classes have some common characteristics e.g. squirrels and rats are rodents.

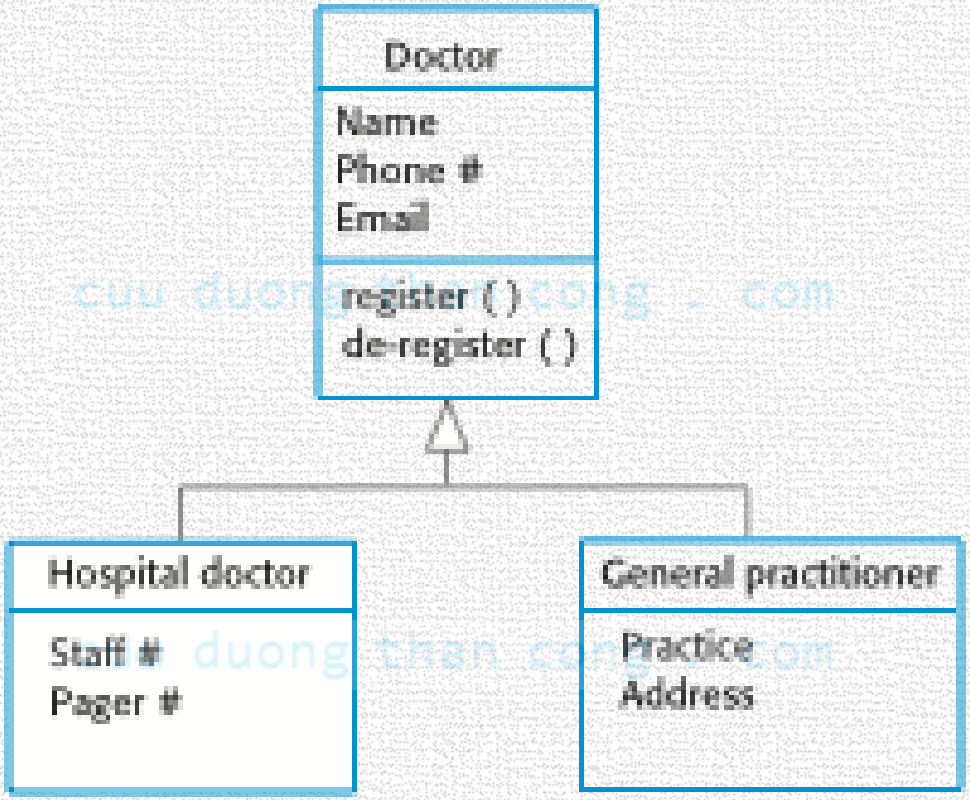
Generalization

- In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.
- In object-oriented languages, such as Java, generalization is implemented using the class inheritance mechanisms built into the language.
- In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level classes.
- The lower-level classes are subclasses inherit the attributes and operations from their superclasses. These lower-level classes then add more specific attributes and

A generalization hierarchy



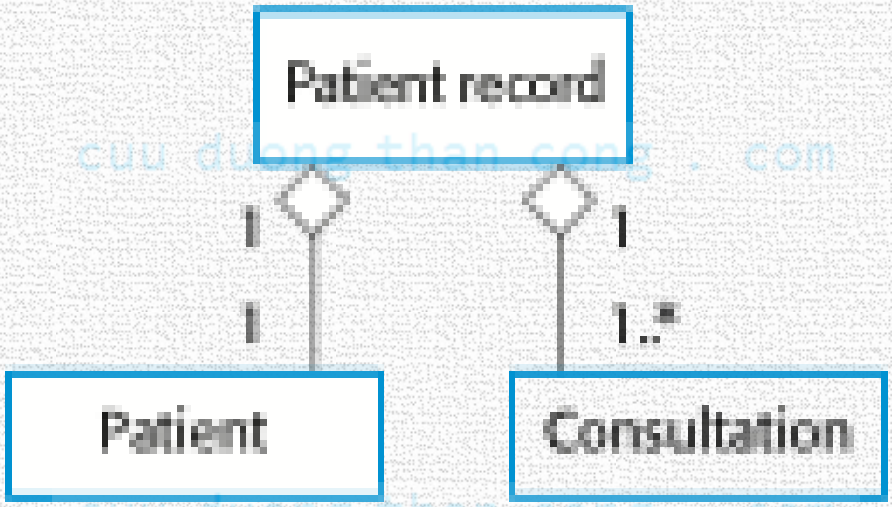
A generalization hierarchy with added detail



Object class aggregation models

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.

The aggregation association



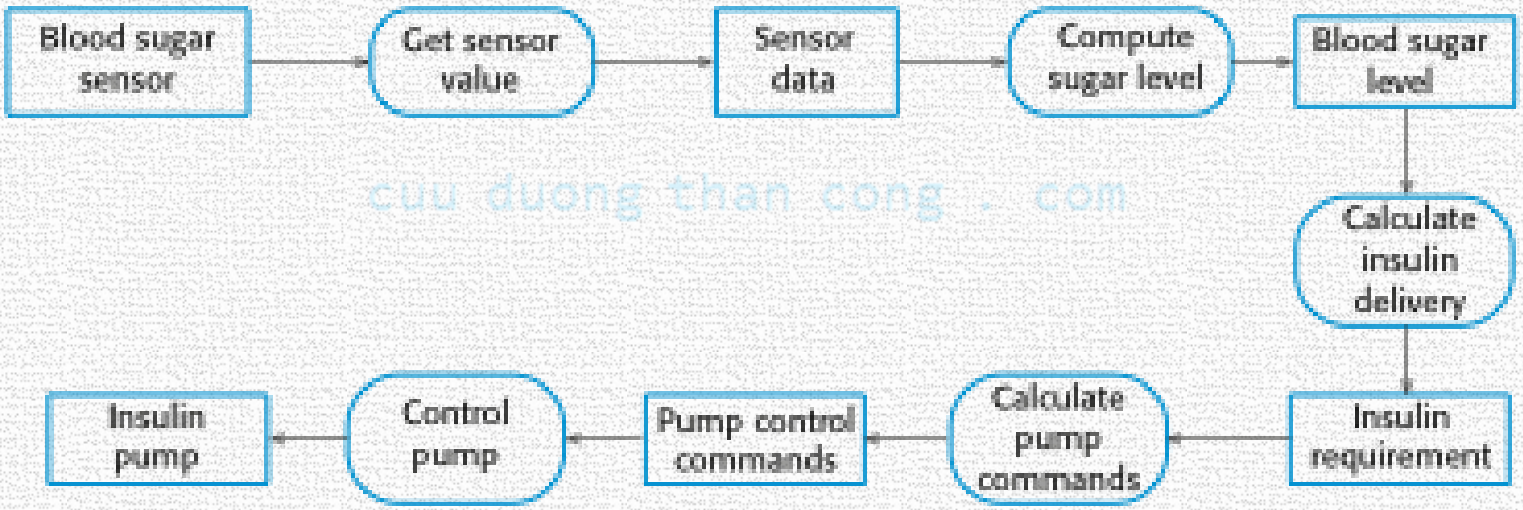
Behavioral models

- Behavioral models are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - **Data**: Some data arrives that has to be processed by the system.
  - **Events**: Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

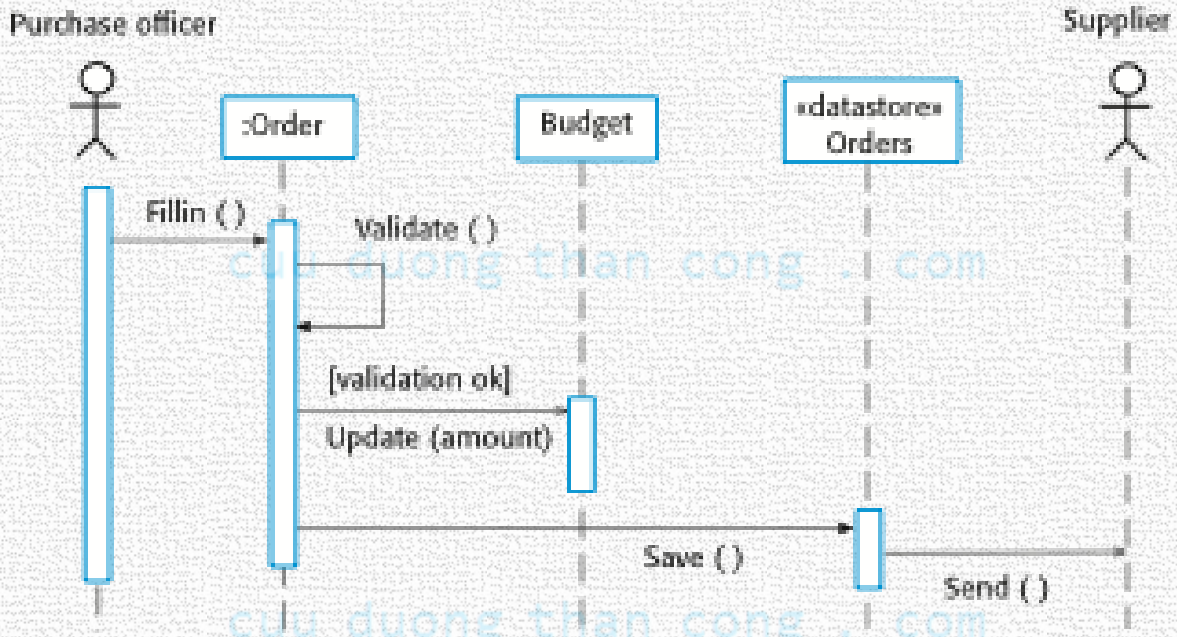
Data-driven modeling

- Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.
- Data-driven models show the sequence of actions involved in processing input data and generating an associated output.
- They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.

An activity model of the insulin pump's operation



Order processing



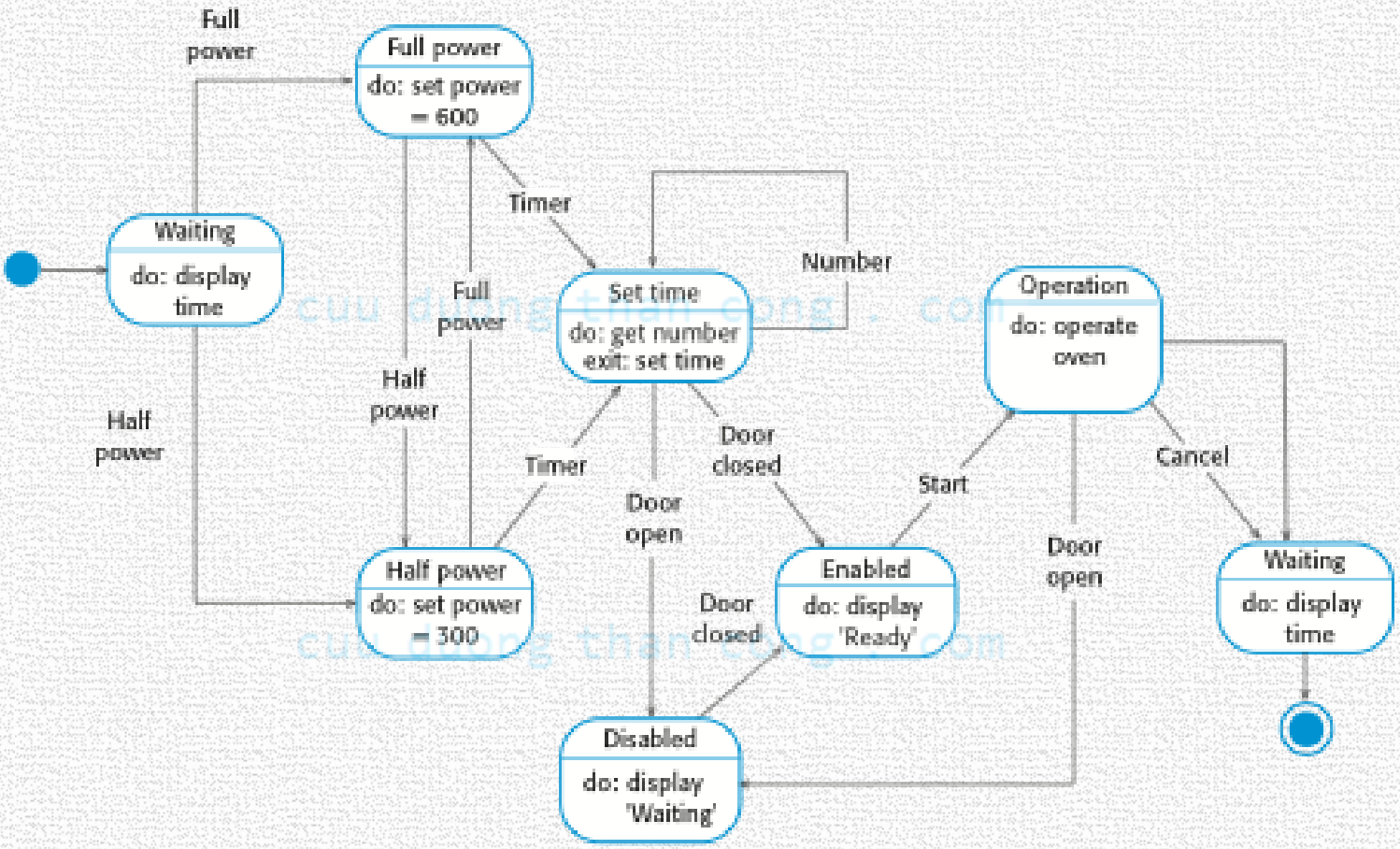
Event-driven modeling

- Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to events such as ‘receiver off hook’ by generating a dial tone.
- Event-driven modeling shows how a system responds to external and internal events.
- It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another.

State machine models

- These model the behaviour of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- Statecharts are an integral part of the UML and are used to represent state machine models.

State diagram of a microwave oven



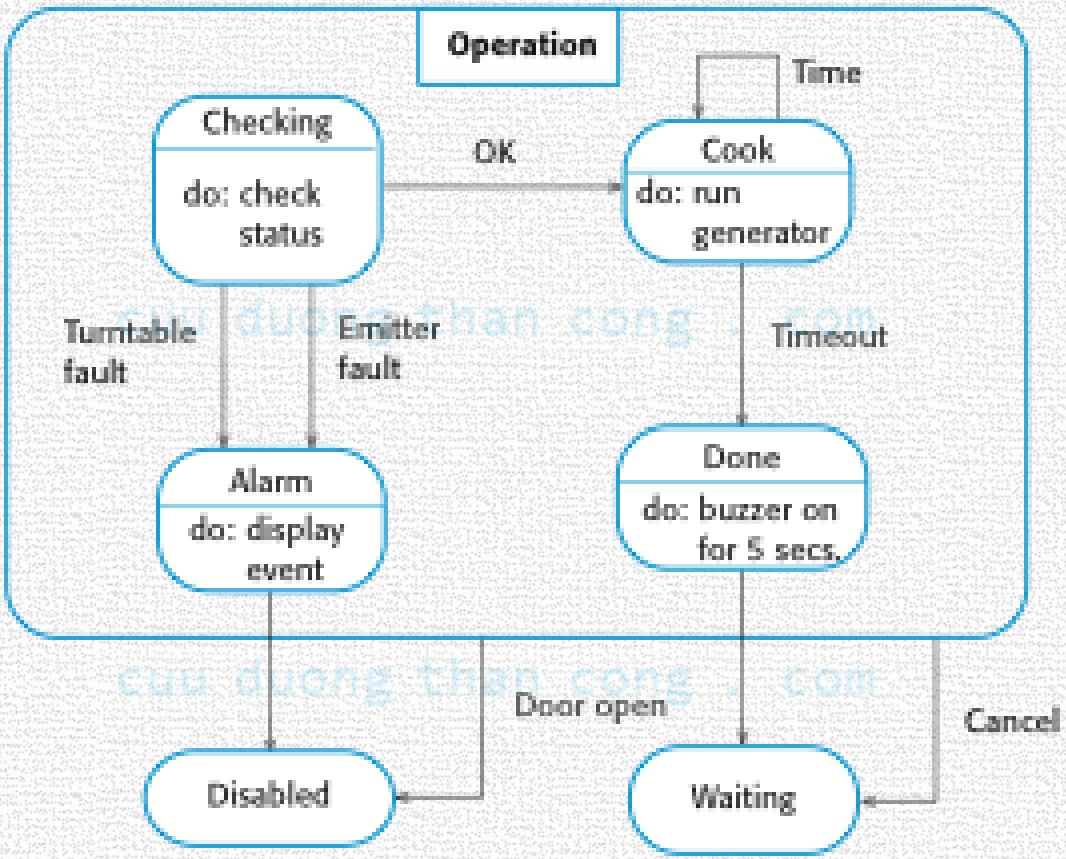
States and stimuli for the microwave oven (a)

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ‘Half power’.
Full power	The oven power is set to 600 watts. The display shows ‘Full power’.
Set time	The cooking time is set to the user’s input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ‘Not ready’.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ‘Ready to cook’.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows ‘Cooking complete’ while buzzer is sounding.

States and stimuli for the microwave oven (b)

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

Microwave oven operation



Model-driven engineering

- Model-driven engineering (MDE) is an approach to software development where models rather than programs are the principal outputs of the development process.
- The programs that execute on a hardware/software platform are then generated automatically from the models.
- Proponents of MDE argue that this raises the level of abstraction in software engineering so that engineers no longer have to be concerned with programming language details or the specifics of execution platforms.

Usage of model-driven engineering

- Model-driven engineering is still at an early stage of development, and it is unclear whether or not it will have a significant effect on software engineering practice.
- Pros
  - Allows systems to be considered at higher levels of abstraction
  - Generating code automatically means that it is cheaper to adapt systems to new platforms.
- Cons
  - Models for abstraction and not necessarily right for implementation.
  - Savings from generating code may be outweighed by the costs of developing translators for new platforms.

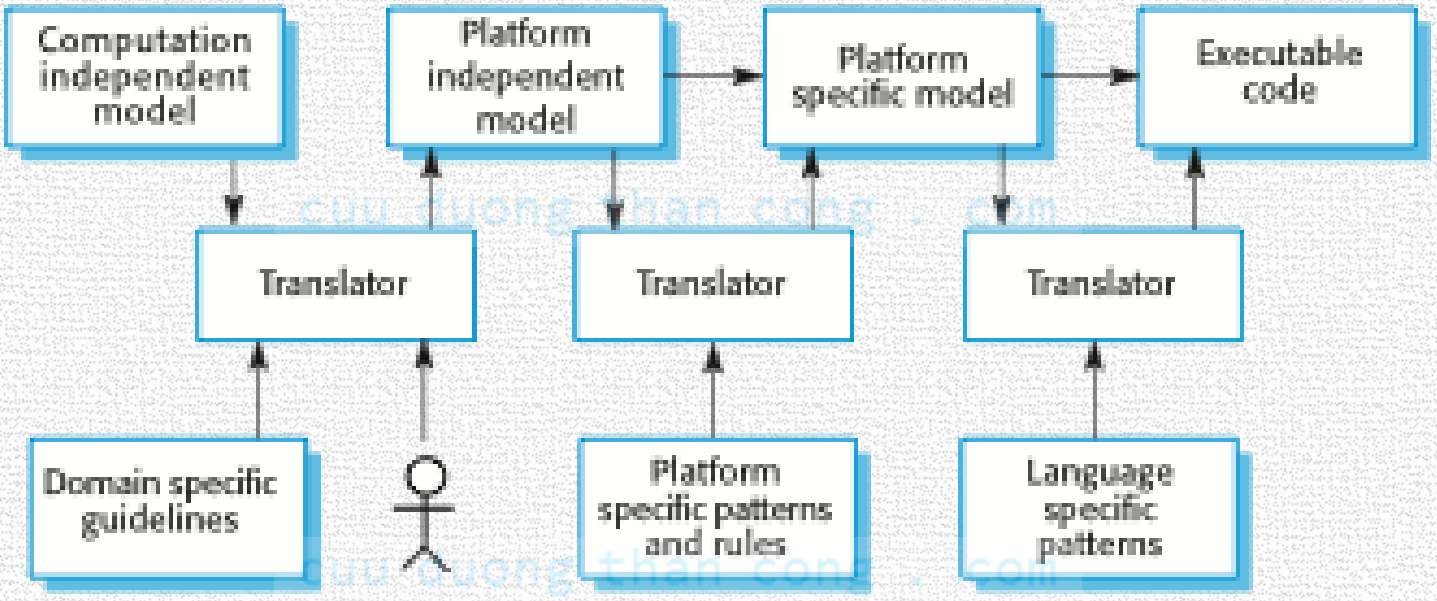
Model driven architecture

- Model-driven architecture (MDA) was the precursor of more general model-driven engineering
- MDA is a model-focused approach to software design and implementation that uses a subset of UML models to describe a system.
- Models at different levels of abstraction are created. From a high-level, platform independent model, it is possible, in principle, to generate a working program without manual intervention.

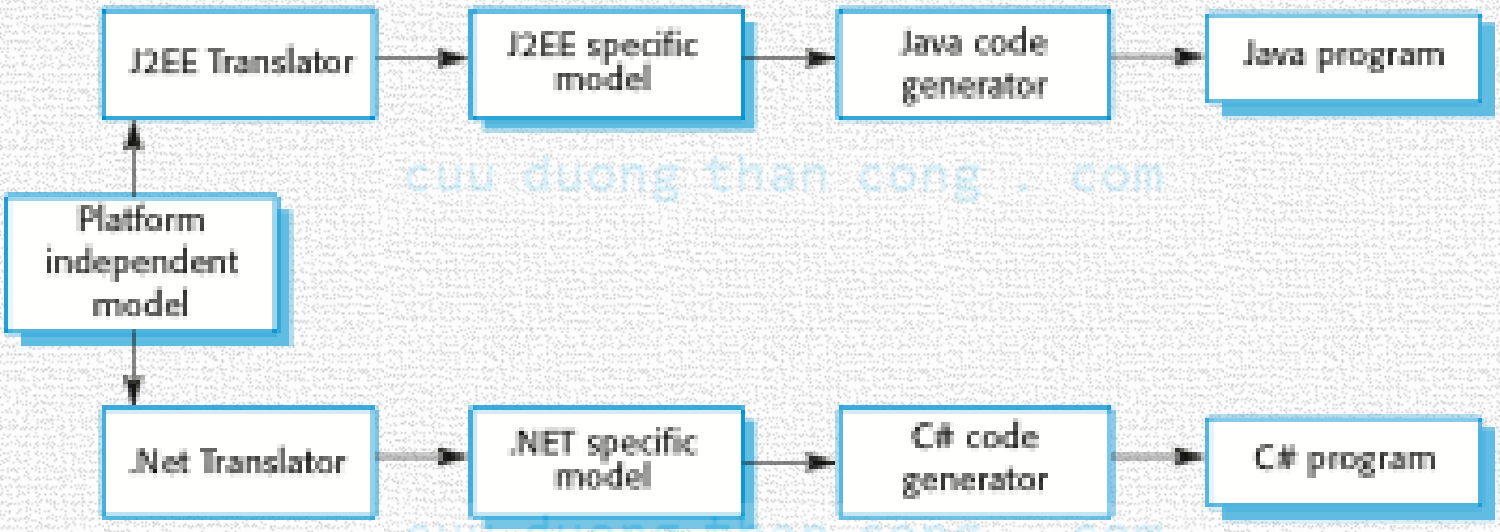
Types of model

- A computation independent model (CIM)
  - These model the important domain abstractions used in a system. CIMs are sometimes called domain models.
- A platform independent model (PIM)
  - These model the operation of the system without reference to its implementation. The PIM is usually described using UML models that show the static system structure and how it responds to external and internal events.
- Platform specific models (PSM)
  - These are transformations of the platform-independent model with a separate PSM for each application platform. In principle, there may be layers of PSM, with each layer adding some platform-specific detail.

MDA transformations



Multiple platform-specific models



Summary

- **A model is an abstract view of a system** that ignores system details. Complementary system models can be developed to show the system's context, interactions, structure and behavior.
- **Context models** show how a system that is being modeled is positioned in an environment with other systems and processes.
- **Use case diagrams** and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; **sequence diagrams** add more information to these by showing interactions between system objects.
- **Structural models** show the organization and architecture of a system. **Class diagrams** are used to define the static structure of classes in a system and their associations.

Summary (cont.)

- **Behavioral models** are used to describe the dynamic behavior of an executing system. This behavior can be modeled from the perspective of the data processed by the system, or by the events that stimulate responses from a system.
- **Activity diagrams** may be used to model the processing of data, where each activity represents one process step.
- **State diagrams** are used to model a system's behavior in response to internal or external events.
- Model-driven engineering is an approach to software development in which a system is represented as a set of models that can be automatically transformed to executable code.

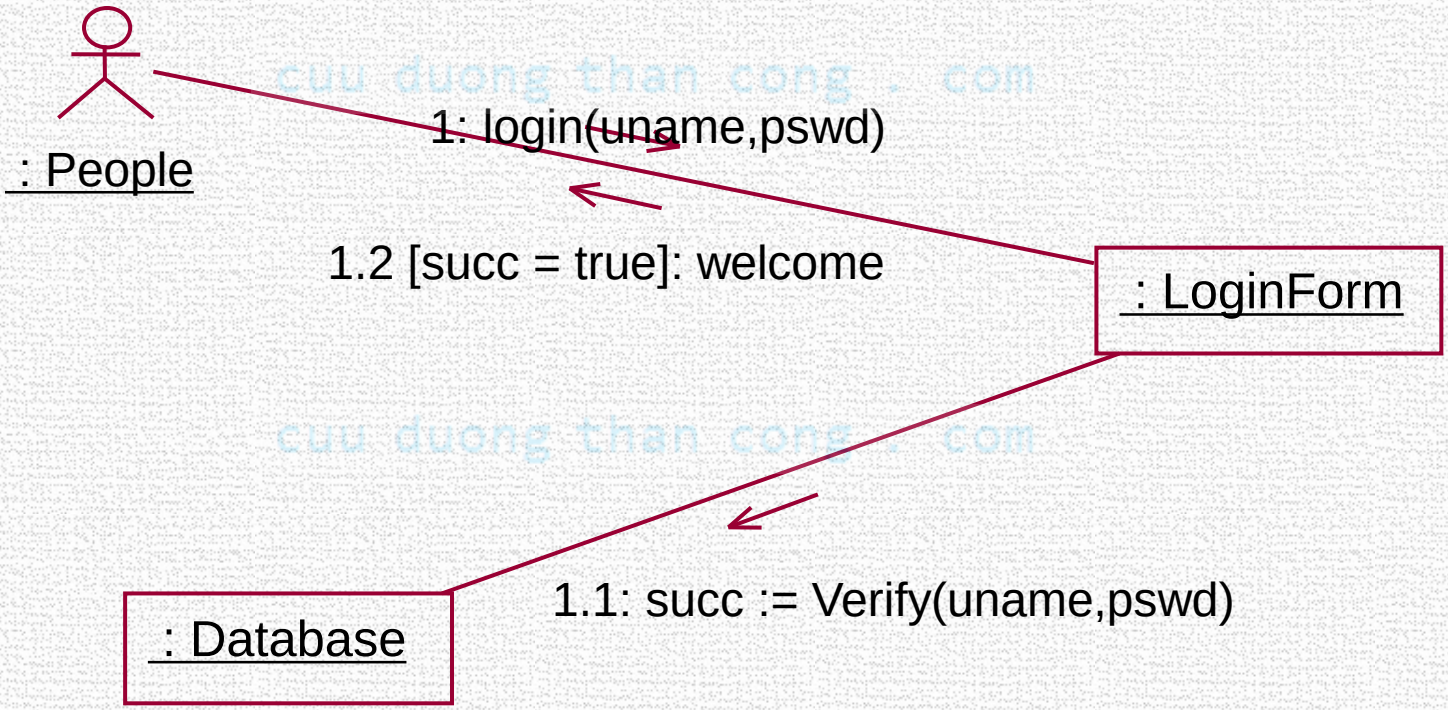
More

- Read [2] for all other diagrams in UML

cuu duong than cong . com

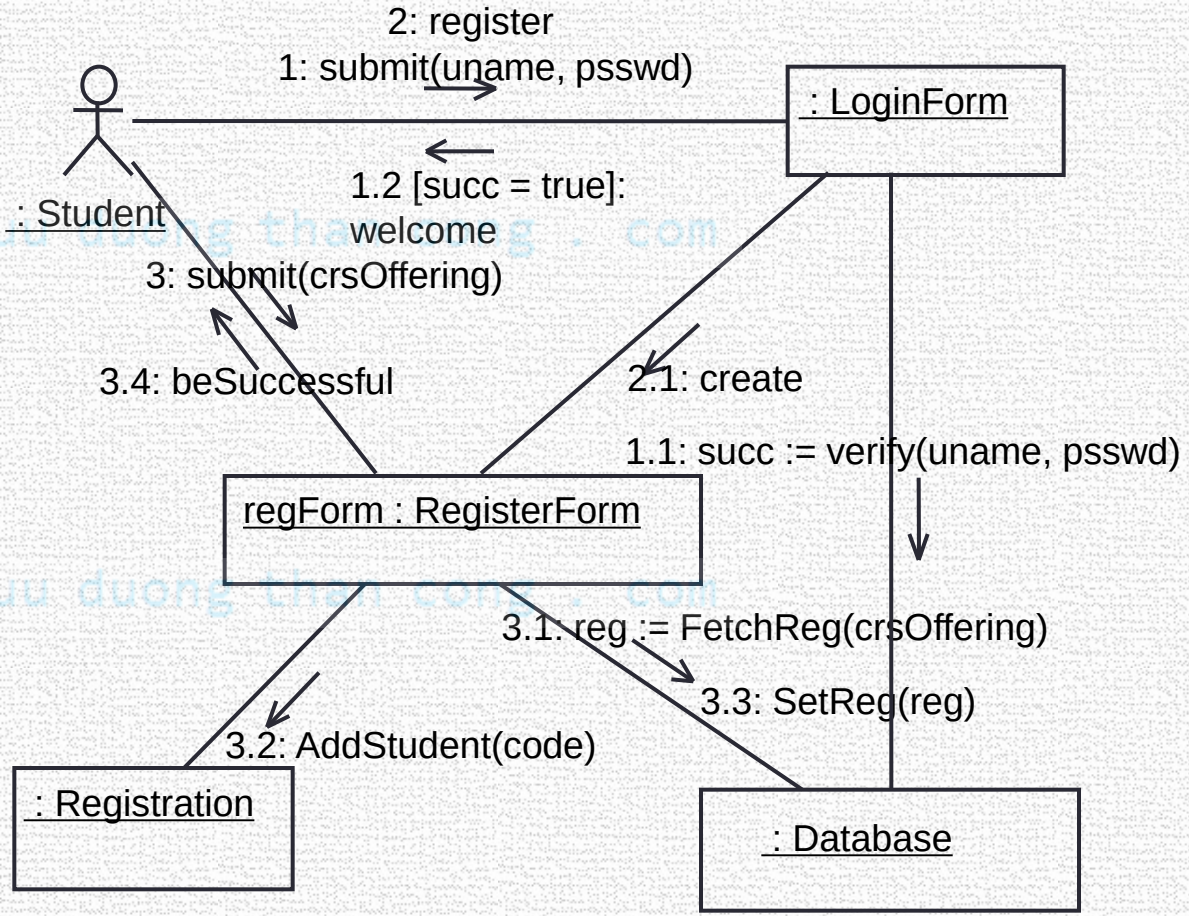
Collaboration diagrams – Example

- Collaboration diagram for use-case Login



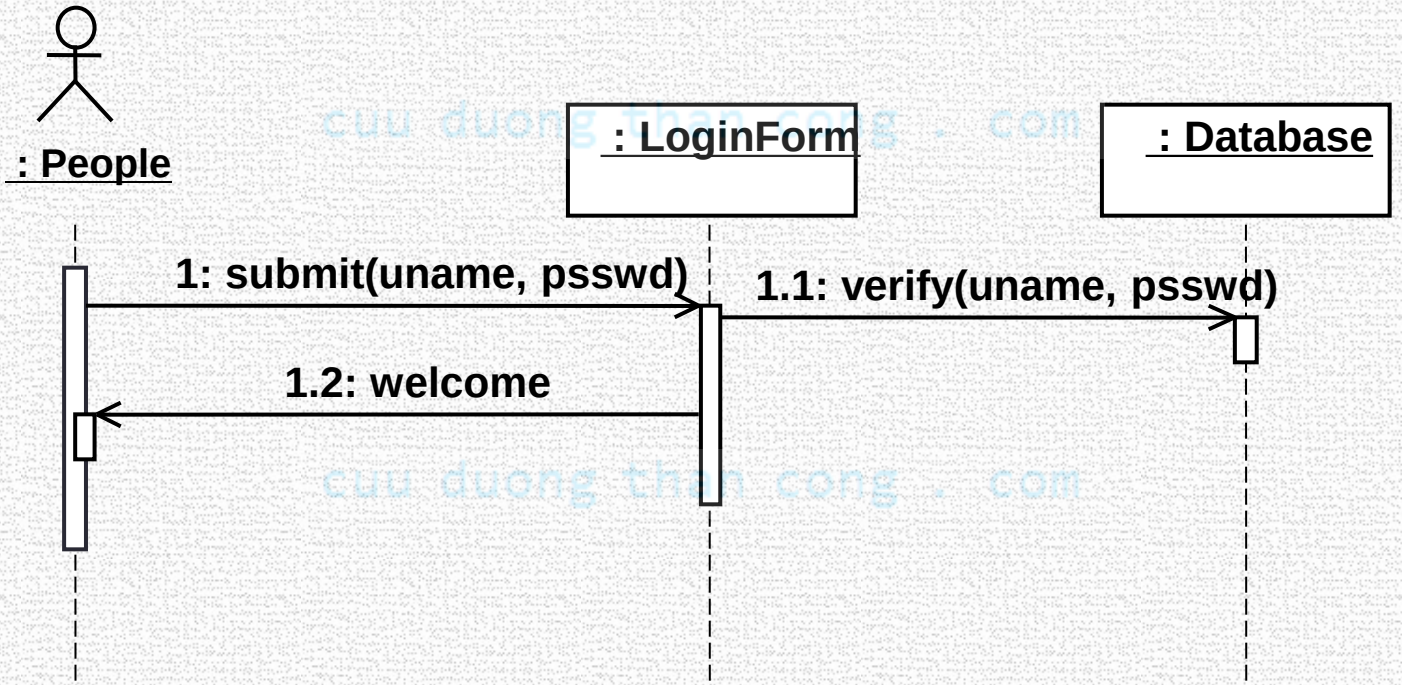
Collaboration diagrams – Example 2

- Collaboration diagram for use-case Registers course



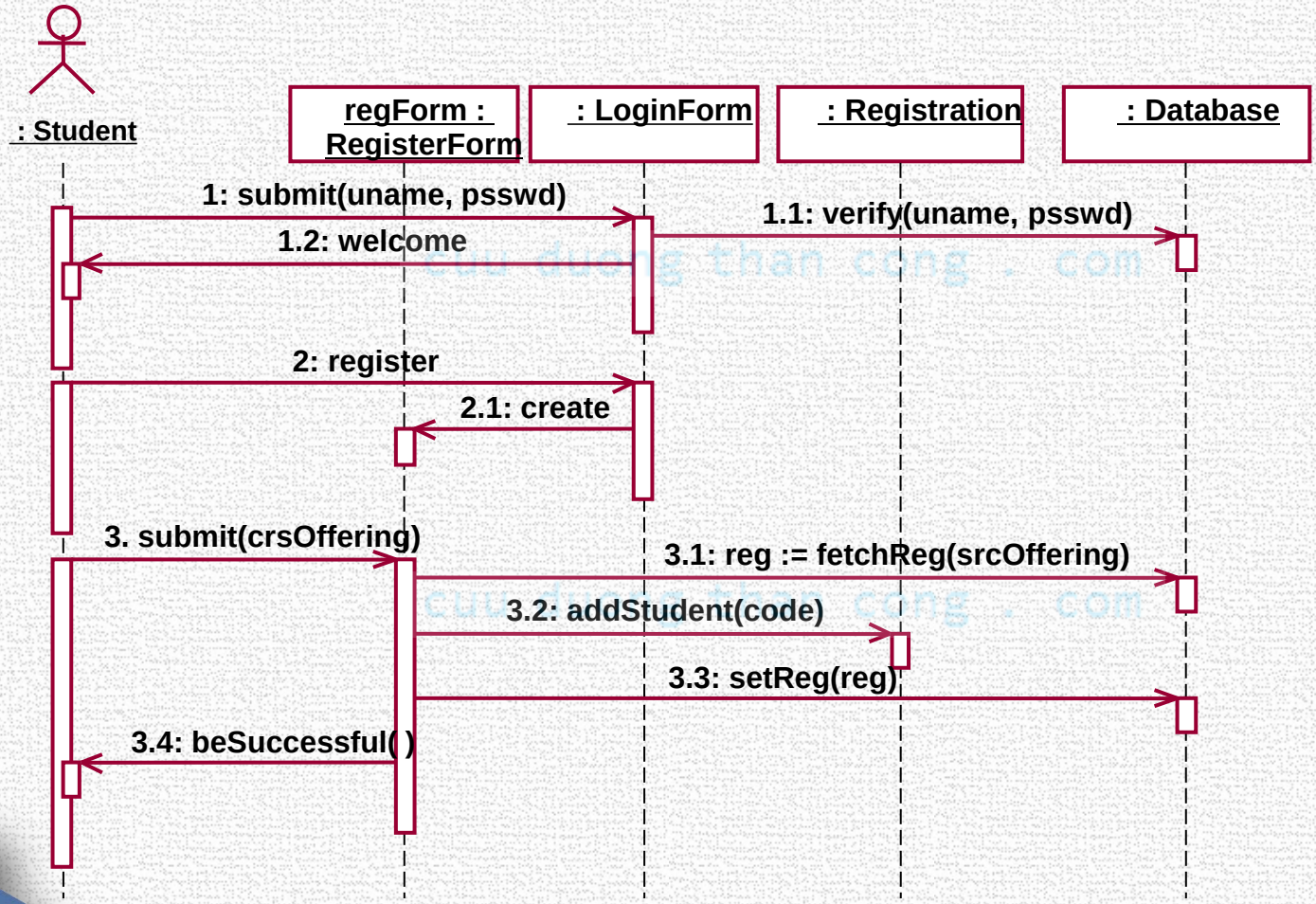
Collaboration diagrams vs. Sequence diagrams - Example 1

- Sequence diagram for use-case Login



Collaboration diagrams vs. Sequence diagrams - Example 1

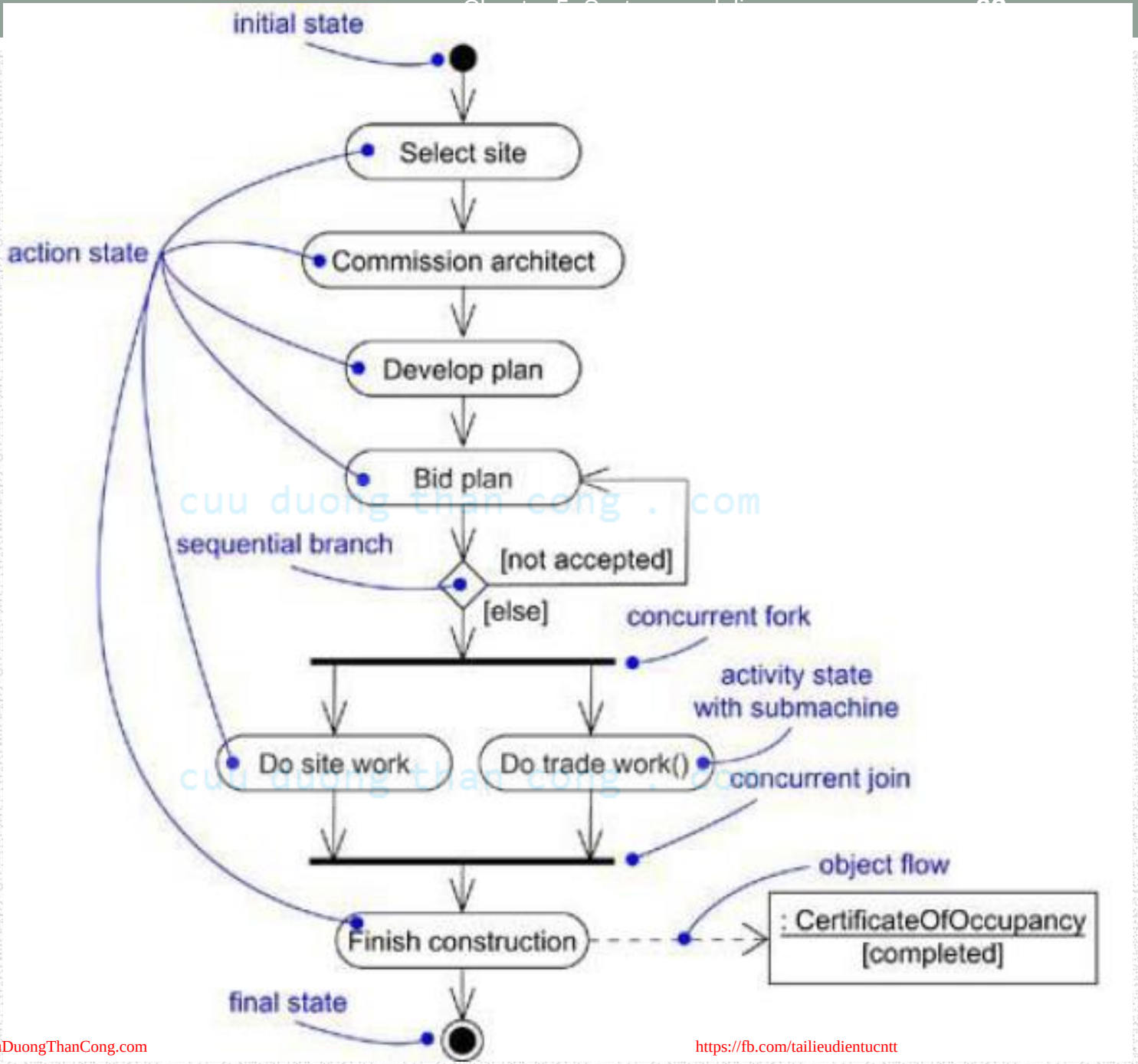
- Sequence diagram for use-case Register courses



Activity diagrams

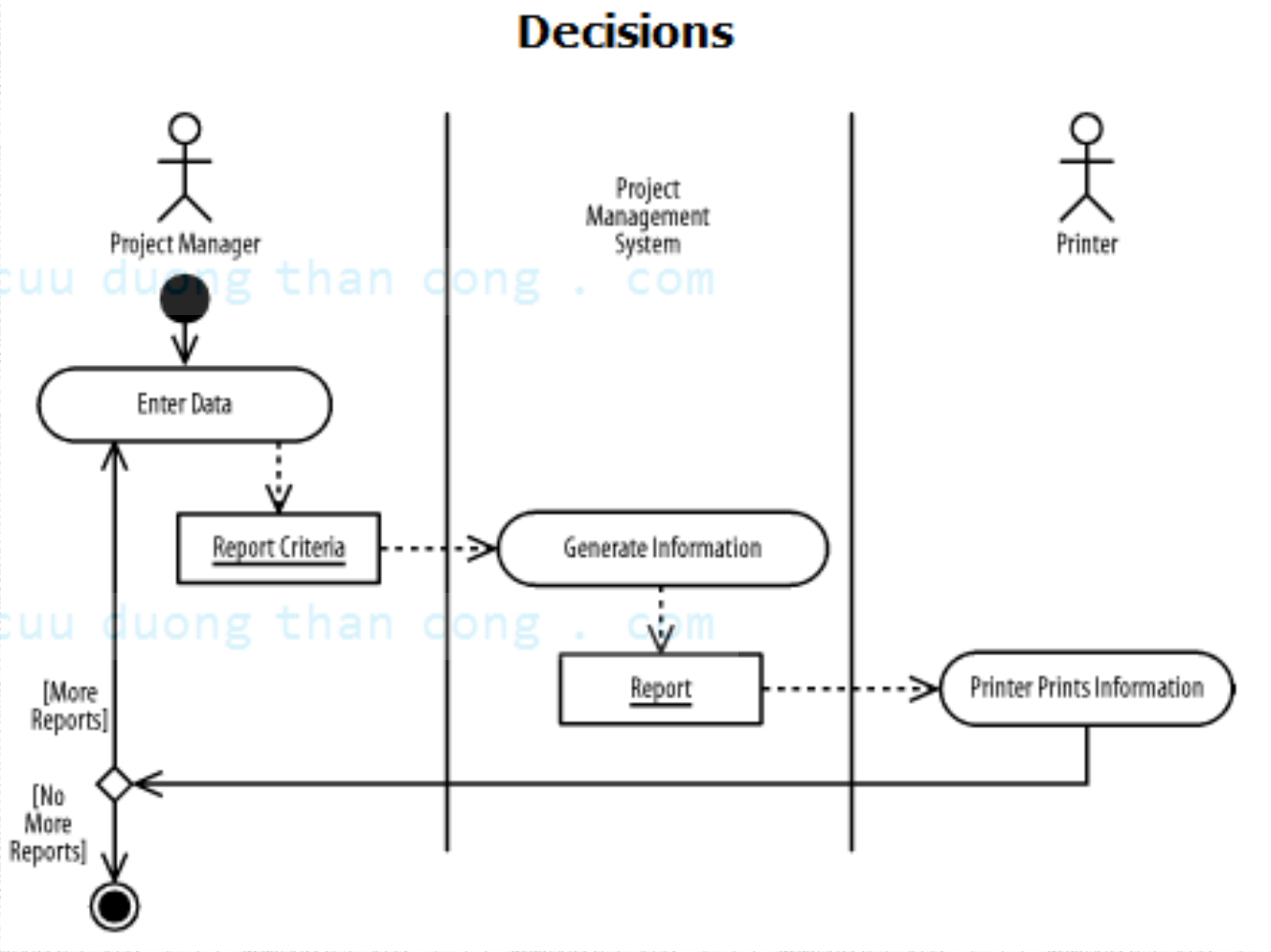
- An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within a system.
- Activity diagrams address the dynamic view of a system. They are especially important in modeling the function of a system and emphasize the flow of control among objects.
- Focuses on activities

Activity diagrams



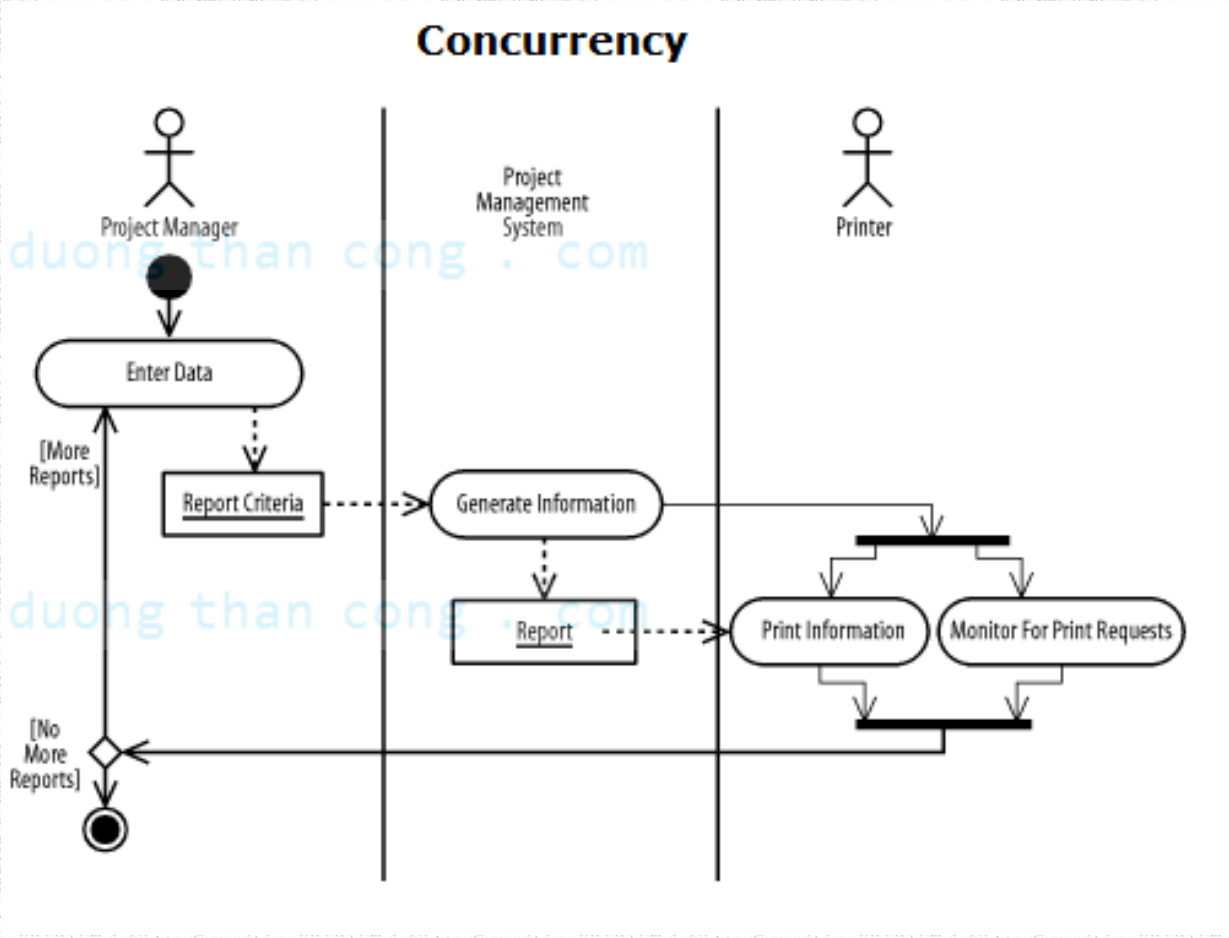
Activity diagrams

- Decisions: A decision involves selecting one control-flow transition out of many control-flow transitions based upon a condition



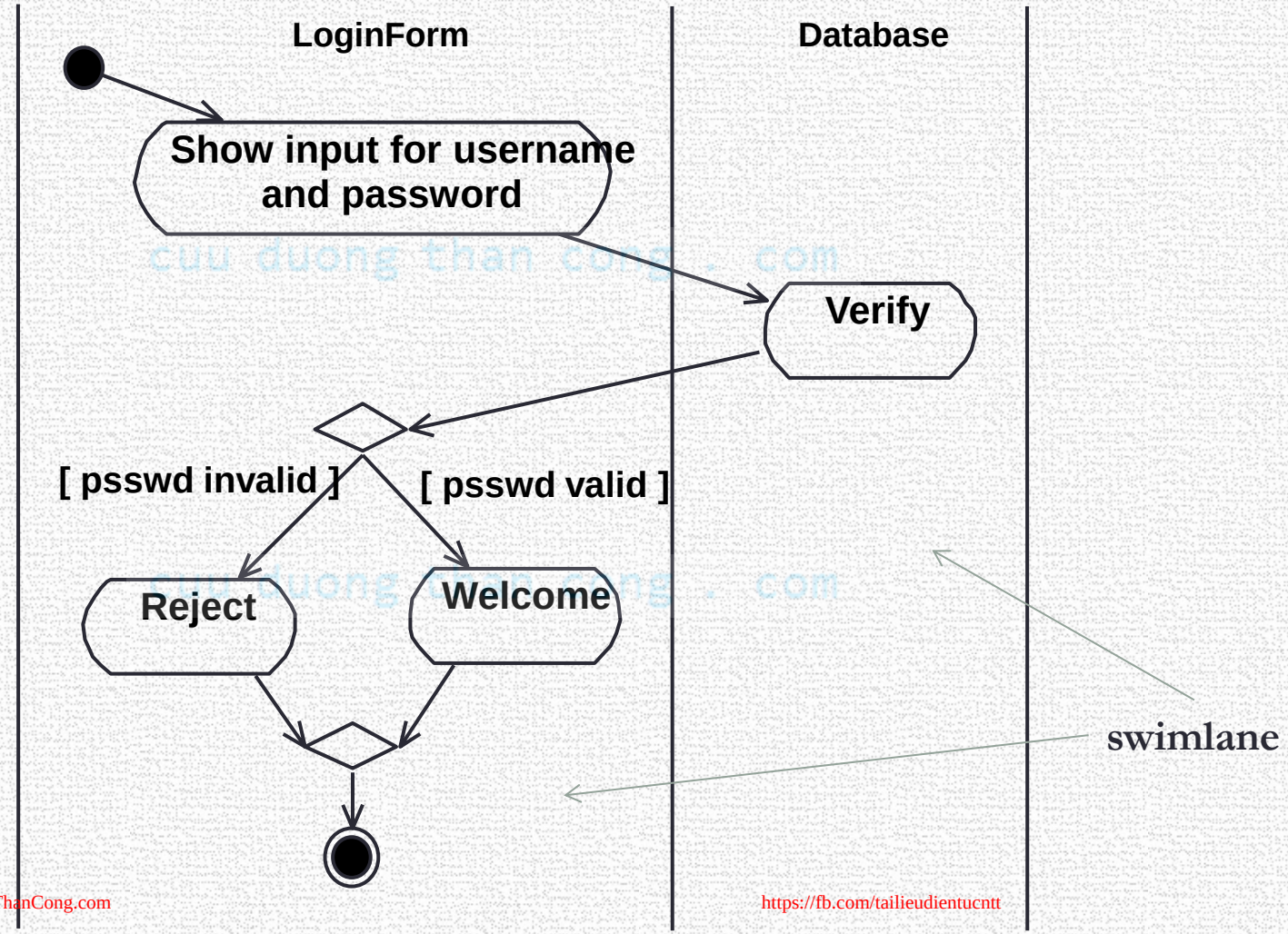
Activity diagrams

- Concurrency: Concurrency involves selecting multiple transitions simultaneously. For example, while the printer is printing a report, the printer must also monitor for other incoming print requests.



Activity diagrams – Example

- Activity diagram for submitting action in LoginForm



Activity diagrams – Example 2

- Activity diagram for submitting in RegisterForm