Usecases

LECTURE 5

Types of Languages

Domain specific Language (DSL)

- Language specially designed to perform a task in a certain domain e.g HTML, C++, SQL etc
- A formal process able language targeting at a specific viewpoint or aspect of a software system
- Its semantics flexibility and notation is designed in order to support working with that viewpoint as good as possible

General Purpose language (GPL)

- GPL is not domain specific e.g UML, Petrinets, Graphs etc
- A GPL provides notations that are used to describe a computation in a human-readable form that can be translated into a machine-readable representation
- A GPL is a formal notation that can be used to describe problem solutions in a precise manner.
- A GPL is a standardized communication technique for expressing instructions to a computer.

Anatomy of Language

Abstract Syntax

- Describes the structure of the language and the way the different primitives can be combined together, independently of any particular representation or encoding

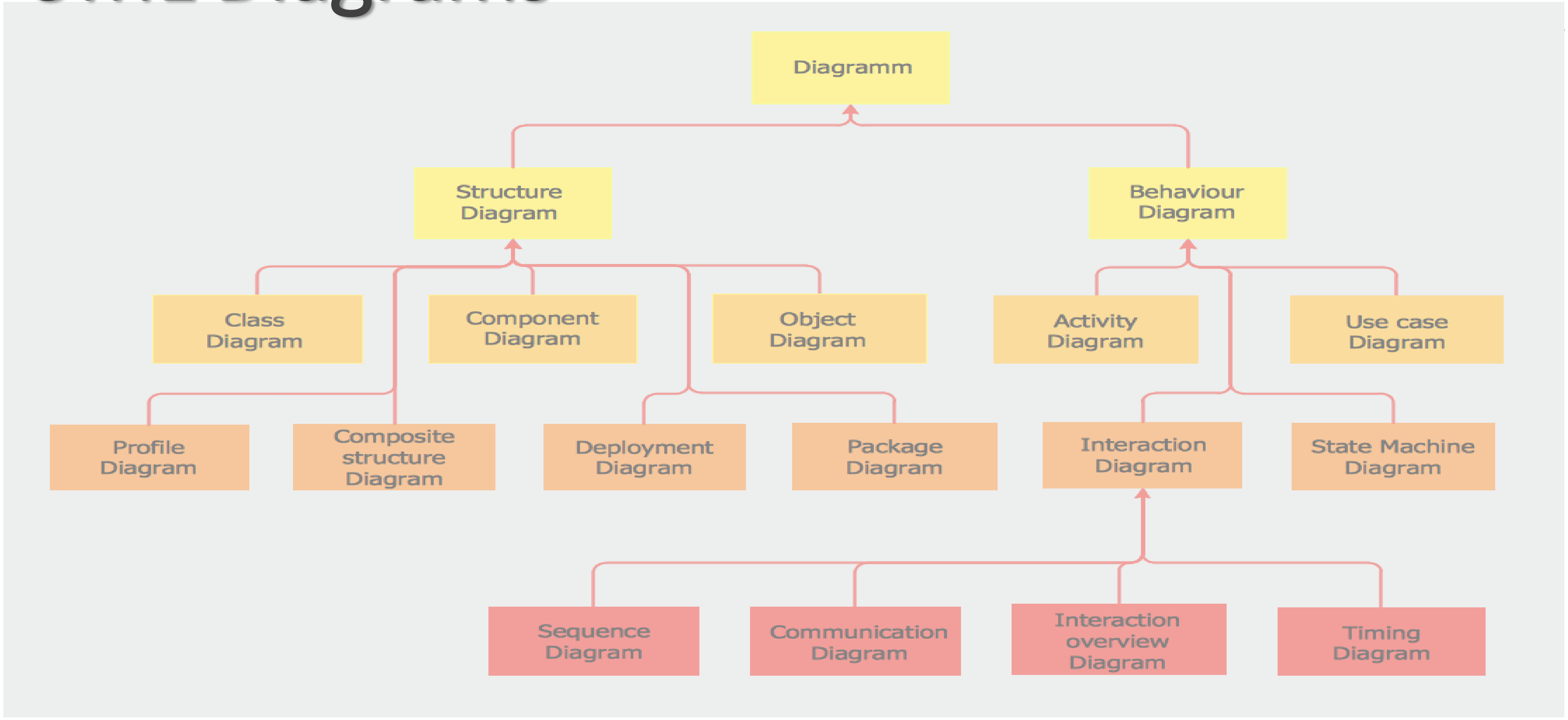
Concrete Syntax

- Describes specific representations of the modeling language, covering encoding and/or visual appearance

Semantics

- Describing the meaning of the elements defined in the language executed? and the meaning of the different ways of combining them.

UML Diagrams



What is a use case?

- A requirements analysis concept
- A case of a use of the system/product
- Describes the system's actions from a the point of view of a user
- Tells a story
- A sequence of events involving
- Interactions of a user with the system
- Specifies one aspect of the behavior of a system, without specifying the structure of the system
- Is oriented toward satisfying a user's goal

How do we describe use cases?

- Textual or tabular descriptions
- User stories
- Diagrams

Textual and Tabular Description

Use Case 1.1 *Withdraw money from an ATM*

Withdraw money from an ATM

Primary actors: Customer
Secondary actors: ATM Technician
Bank

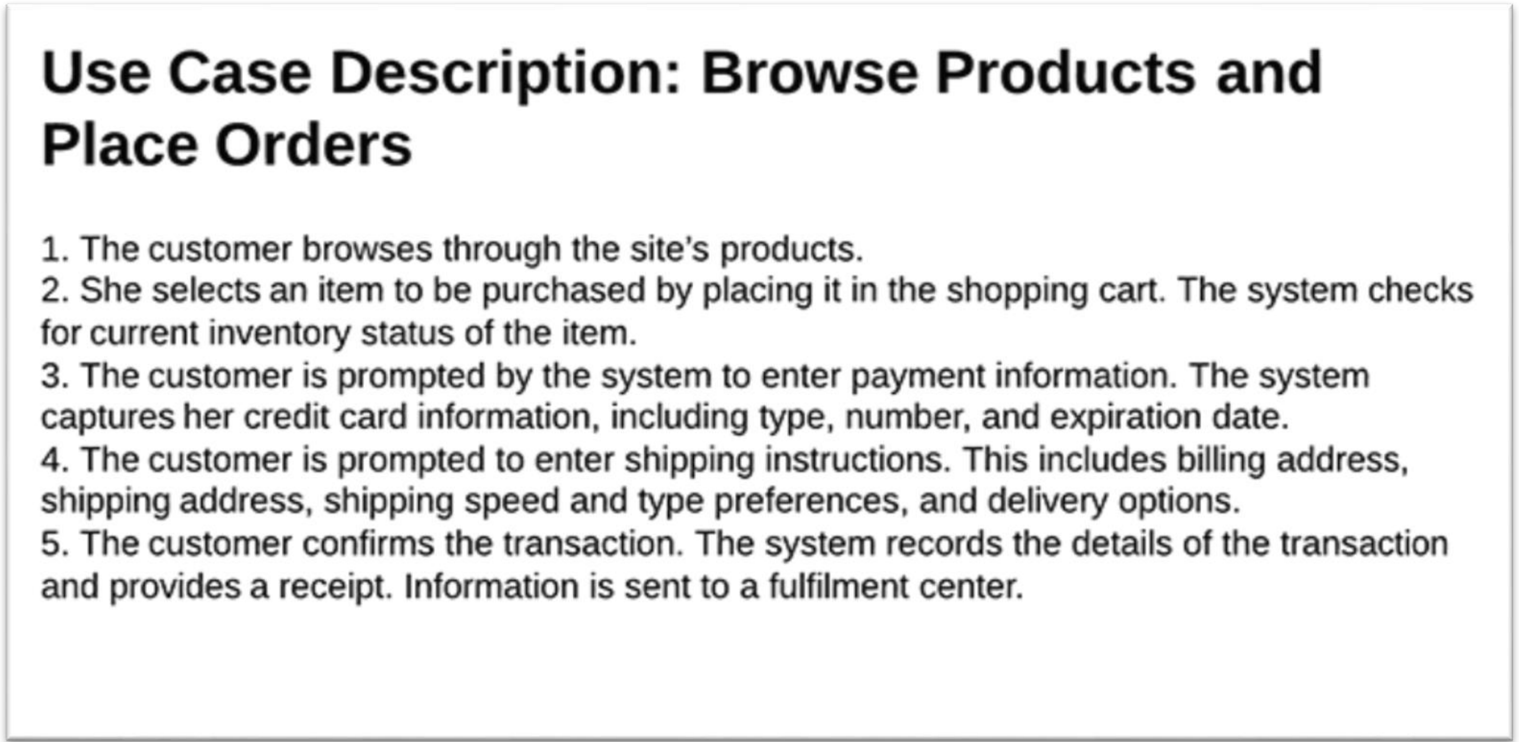
Preconditions: Network connection is active
ATM has available cash

Basic flow of events:
1. Bank customer inserts debit card and enters PIN.
2. Customer is validated.
3. ATM displays actions available on ATM unit. Customer selects Withdraw Cash.
4. ATM prompts account.
5. Customer selects account.
6. ATM prompts amount.
7. Customer enters desired amount.
8. Information sent to Bank, inquiring if sufficient funds/allowable withdrawal limit.
9. Money is dispensed and receipt prints.

Alternative flows
2a. Customer is not validated.
  2a1. ATM displays error message.
7a. Customer selects invalid amount.
  7a1. ATM prompts user to re-enter valid amount.
8a. Customer has insufficient funds.
  8a1. ATM displays error message.
  8a2. ATM shows available withdrawal limit, redirects to step 6.
9a. ATM has insufficient cash.
  9a1. ATM Technician is alerted.
  9a2. ATM displays error message and phone number to call.
9b. Cash gets stuck in dispensing.
  9b1. ATM displays error message.

What is a user story?

- An abbreviated description of a use case
- Description of scenario or events in the bulleted points
- Answers 3 questions:
  1) Who?
  2) Does what?
  3) And why?



Use Case Description: Browse Products and Place Orders

1. The customer browses through the site's products.
2. She selects an item to be purchased by placing it in the shopping cart. The system checks for current inventory status of the item.
3. The customer is prompted by the system to enter payment information. The system captures her credit card information, including type, number, and expiration date.
4. The customer is prompted to enter shipping instructions. This includes billing address, shipping address, shipping speed and type preferences, and delivery options.
5. The customer confirms the transaction. The system records the details of the transaction and provides a receipt. Information is sent to a fulfilment center.

Use Case Diagram

- A picture
  - describes how actors relate to use cases
  - and use cases relate to one another
- Diagrams are not essential
- They are helpful in giving an overview, but only secondary in importance to the textual description
- They do not capture the full information of the actual use cases
- In contrast, text is essential

Use Case Diagram – Objective

- Built in early stages of development
- Purpose
  - Specify the context of a system
  - Capture the requirements of a system
  - Validate a systems architecture
  - Drive implementation and generate test cases
  - Developed by Requirement analysts and domain experts

Use Case Diagram Elements

- **Actors**

- something with a behavior or role, e.g., a person, another system, organization.

- **Scenario**

- a specific sequence of actions and interactions between actors and the system, a.k.a. a use case instance

- **Use case**

-a collection of related success and failure scenarios, describing actors using the system to support a goal

- **Other Elements**

-Associations, include, exclude

What is an Actor?

- Include all user roles that interact with the system
- Include system components only if they responsible for initiating/triggering a use case.
- For example, a timer that triggers sending of an e-mail reminder
- Each Actor must be linked to a use case, while some use cases may not be linked to actors.
- Actor is someone interacting with use case (system function).
- Named by noun

Kind of Actors

- **Primary** - a user whose goals are fulfilled by the system

  importance: Define user goals

- **Supporting/Secondary** - provides a service (e.g., info) to the system

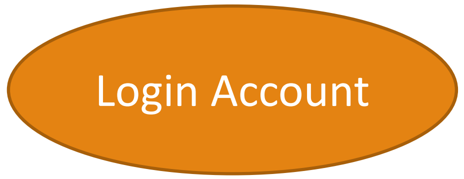
  importance: clarify external interfaces and protocols

- **Offstage** - has an interest in the behavior but is not primary or supporting, e.g., government

  importance: ensure all interests (even subtle) are identified and satisfied

UseCase

- System function (process – automated or manual).
- Named by verb
- Each Actor must be linked to a use case, while some use cases may not be linked to actors.
- Represented by oval in Diagrams



Other Elements

―――	Connection between Actor and Use Case
▭	Boundary of system
<<include>> →	Include relationship between Use Cases (one UC must call another; e.g., Login UC includes User Authentication UC)
<<extend>> →	Extend relationship between Use Cases (one UC calls Another under certain condition; think of if-then decision points)

Linking Usecases

Association relationships

Generalization relationships

One element (child) "is based on" another element (parent)

Include relationships

One use case (base) includes the functionality of another (inclusion case)

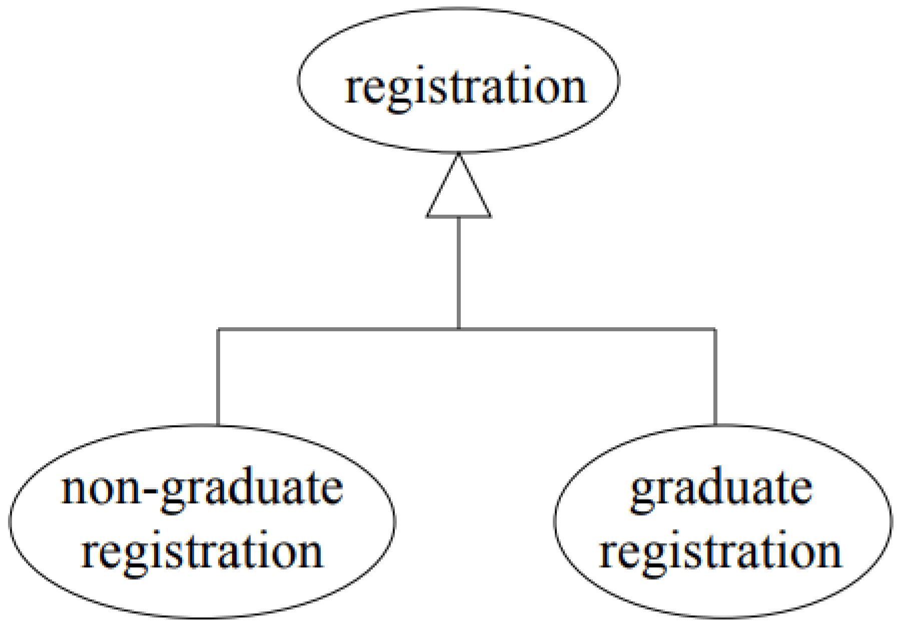
Supports re-use of functionality

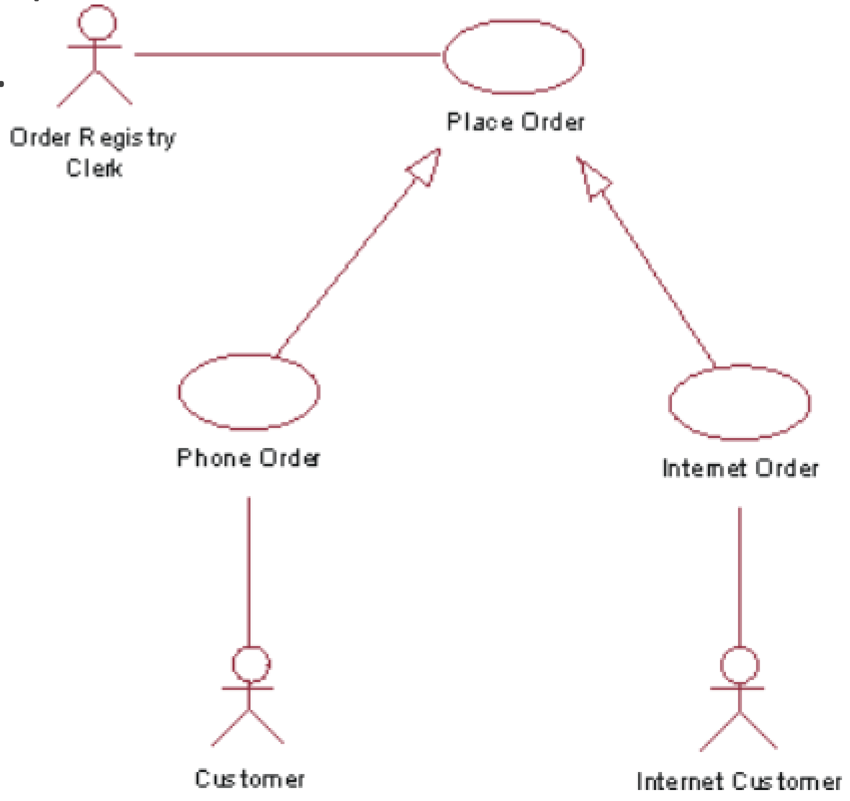
Extend relationships

One use case (extension) extends the behavior of another (base)

Generalization

- The child use case inherits the behavior and meaning of the parent use case.
- The child may add to or override the behavior of its parent.

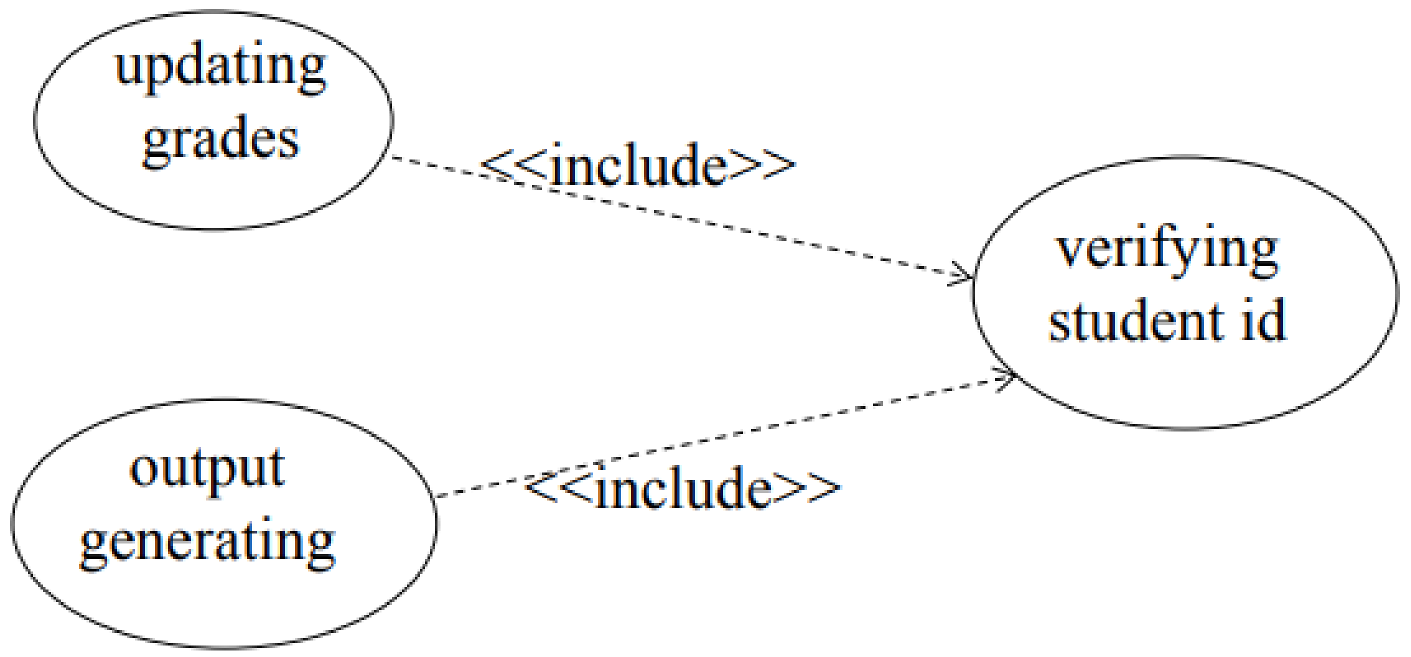




Include

- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.
- Include relationship – a standard case linked to a mandatory use case.
- Standard use case can NOT execute without the include case
- Example: to Authorize Car Loan (standard use case), a clerk must run Check Client’s Credit History (include use case)
- The standard UC includes the mandatory UC (use the verb to figure direction arrow)

Include



Extend

- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.
- Extend relationship – linking an optional use case to a standard use case
- Example: Register Course (standard use case) may have Register for Special Class (extend use case) – class for non-standard students, in unusual time, with special topics, requiring extra fees…)
- The optional UC extends the standard UC
- Standard use case can execute without the extend case

How to create use case diagrams

- List main system functions (use cases) in a column:
  - think of business events demanding system's response
  - users' goals/needs to be accomplished via the system
  - Create, Read, Update, Delete (CRUD) data tasks
  - Naming use cases
  - user's needs usually can be translated in data tasks
- Draw ovals around the function labels
- Draw system boundary
- Draw actors and connect them with use cases (if more intuitive, this can be done as step 2)
- Specify include and extend relationships between use cases (yes, at the end - not before, as this may pull you into process thinking, which does not apply in UC diagramming).