

Use Cases

- One of the primary challenges is the ability to elicit the correct and necessary system requirements from the stakeholders and specify them in a manner understandable to them so those requirements can be verified and validated.

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

Fred Brooks

Use Case Diagram

– Guidelines & Caution

1. Use cases should ideally begin with a verb – i.e. generate report. Use cases should NOT be open ended – i.e. Register (instead should be named as Register New User)
2. Avoid showing communication between actors.
3. Actors should be named as singular. i.e. student and NOT students. NO names should be used – i.e. John, Sam, etc.
4. Do NOT show behavior in a use case diagram; instead only depict only system functionality.
5. Use case diagram does not show sequence – unlike DFDs.

Identifying of Actor and Usecase

- Customer can select items only after browsing the catalog.
- Customer check out. Checkout may not be successful
- Customer fills in shipping information.
- Customer fills in credit card information.

System Concepts for Use-Case Modeling

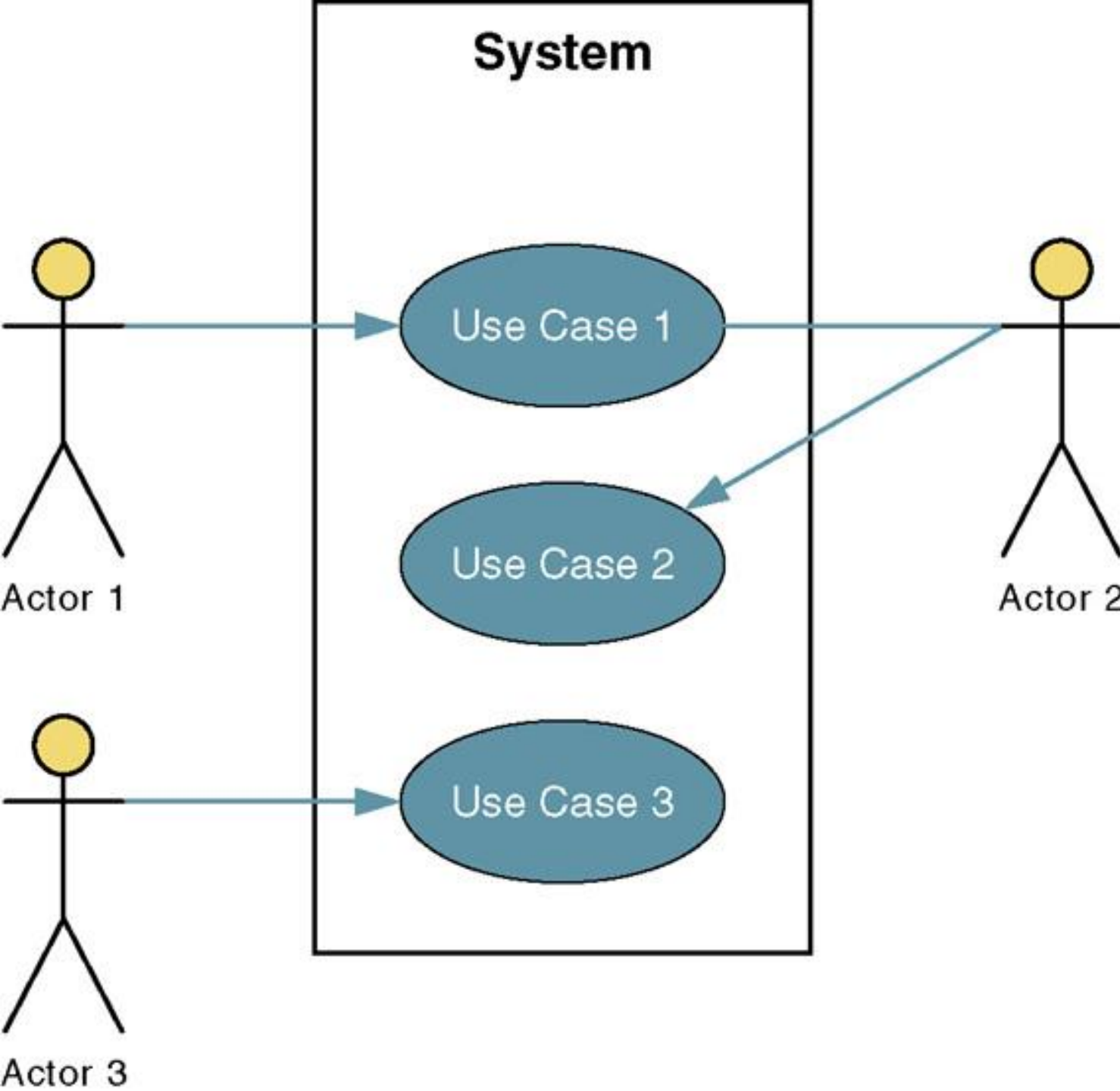
Use case – a behaviorally related sequence of steps (scenario), both automated and manual, for the purpose of completing a single task.

- Description of system functions from the perspective of external users in terminology they understand.

Use-case diagram – a diagram that depicts the interactions between the system and external systems and users.

- graphically describes who will use the system and in what ways the user expects to interact with the system.

Use-case narrative – a textual description of the event and how the user will interact with the system to accomplish the task.



Sample
Use-
Case
Model
Diagram

Basic Use-Case Symbols

Use case – subset of the overall system functionality

- Represented by a horizontal ellipse with name of use case above, below, or inside the ellipse.



Use Case
Symbol

Actor – anyone or anything that needs to interact with the system to exchange information.

- human, organization, another information system, external device, even time.



Actor Symbol

Formats of Use case

- Brief: one paragraph summary, usually the main success scenario.

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Formats of Use case

- Casual: informal paragraph format. Multiple paragraph that cover various scenario.

Handle Returns

Main Success Scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item ...

Alternate Scenarios:

If the credit authorization is reject, inform the customer and ask for an alternate payment method.

If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).

If the system detects failure to communicate with the external tax calculator system, ...

Formats of Use case

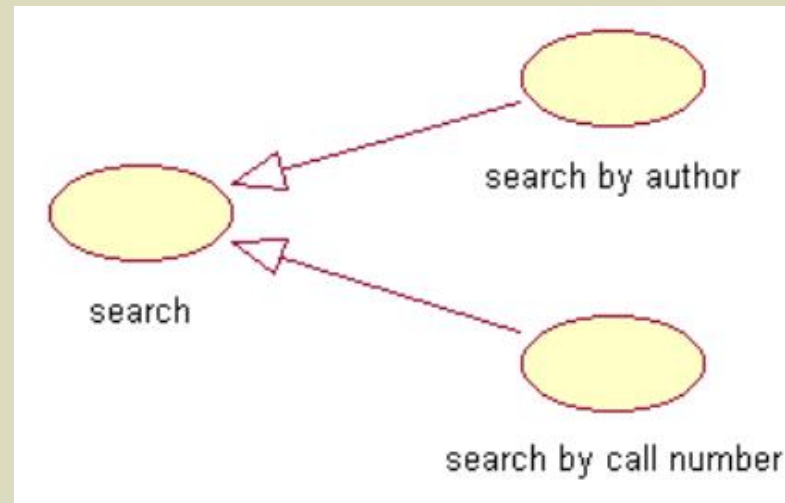
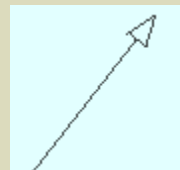
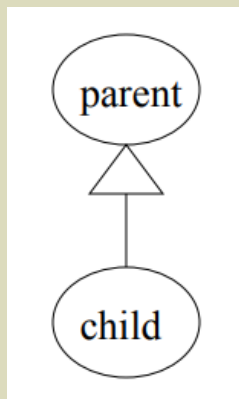
- Fully Dressed: the most elaborate. All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.

Use Case - Relationships and its Types

- Association relationship: Represent communication between actor and use case
- Often referred to as a communicate association
- use just a line to represent

Use Case - Relationships and its Types

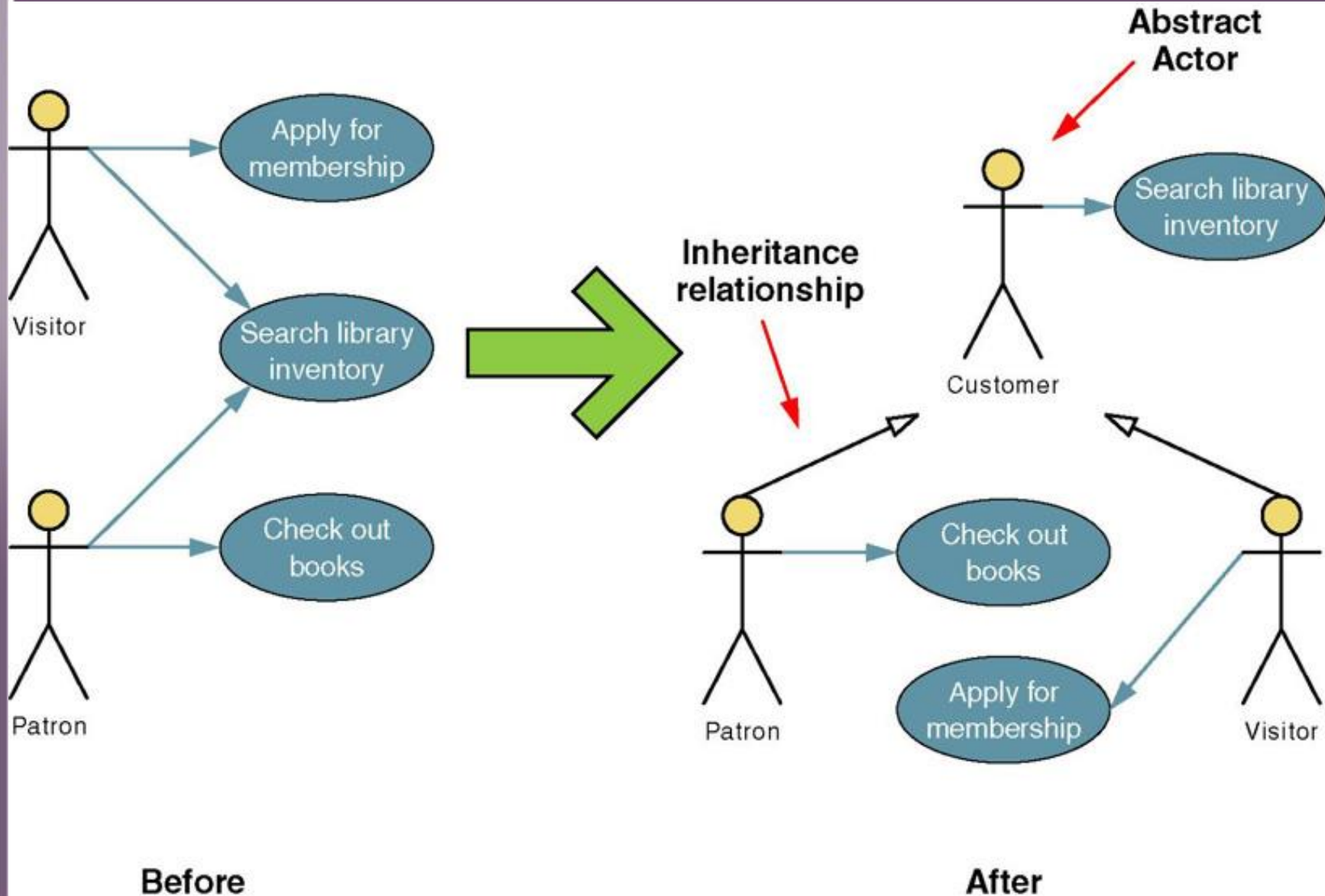
- Generalization:
- The child use case inherits the behaviour and meaning of the parent use case.
- The child may add to or override the behaviour of its parent.
- Notation:



Use Case - Relationships and its Types

- Generalization relationship between actors
 - actor generalization refers to the relationship which can exist between two actors
- Generalization relationship between use cases
 - use case generalization refers to the relationship which can exist between two use cases

Use Case Inheritance Relationship

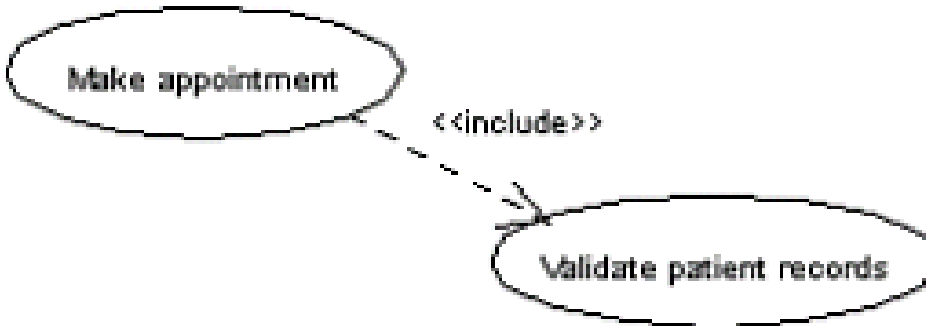


Use Case - Relationships and its Types

- **Include**

- Specifies that the source use case explicitly incorporates the behavior of another use case at a location specified by the source
- The include relationship adds additional functionality not specified in the base use case.
- `<<include>>` is used to include common behavior from an included use case into a base use case in order to support re-use of common behavior.
- Notation `..<<include>>..>`

<<include>>



In Figure, the functionality defined by the "Validate patient records" use case is contained within the "Make appointment" use case.


Hence, whenever the "Make appointment" use case executes, the steps defined in the "Validate patient records" use case are also executed.

Include Relationship

- A standard case linked to a mandatory use case.
- Example:
 - to *Authorize Car Loan* (standard use case), a clerk must run *Check Client's Credit History* (include use case).
- Standard use case can NOT execute without the include case (**tight coupling**) .

Use Case - Relationships and its Types

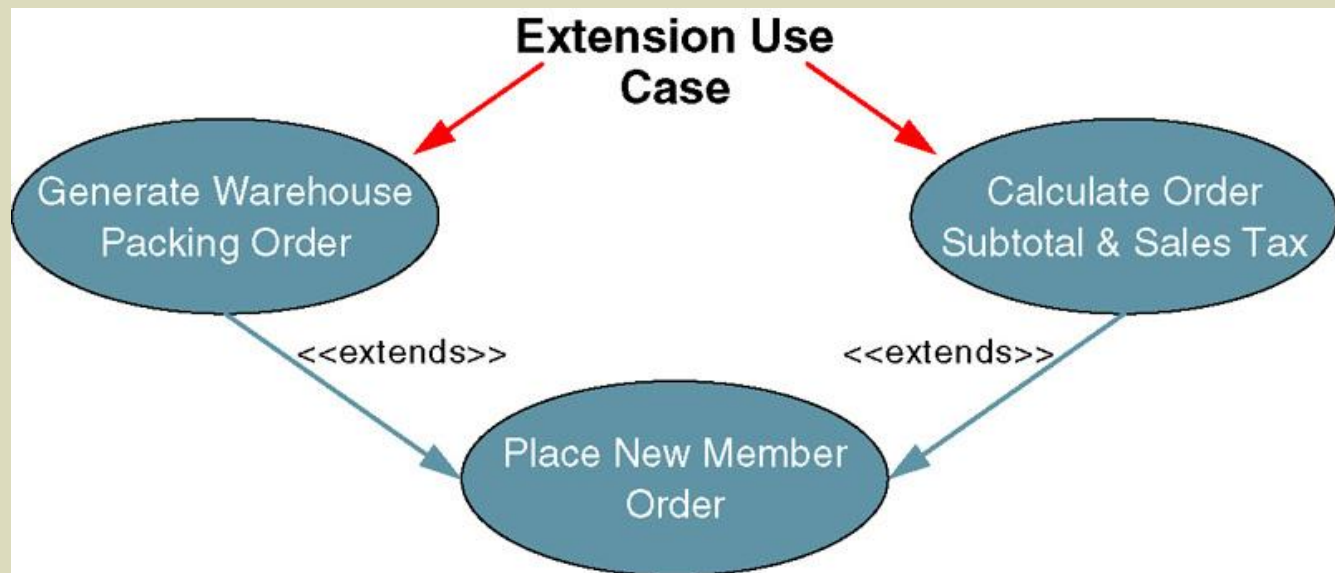
- **Extend**

- Specifies that the target use case extends the behavior of the source.
- The extend relationships shows optional functionality or system behavior.
- <<extend>> is used to include optional behavior from an extending use case in an extended use case.
- Notation The notation consists of a dashed line with an open arrowhead pointing towards the target use case, and the stereotype <<extend>> written above the line.

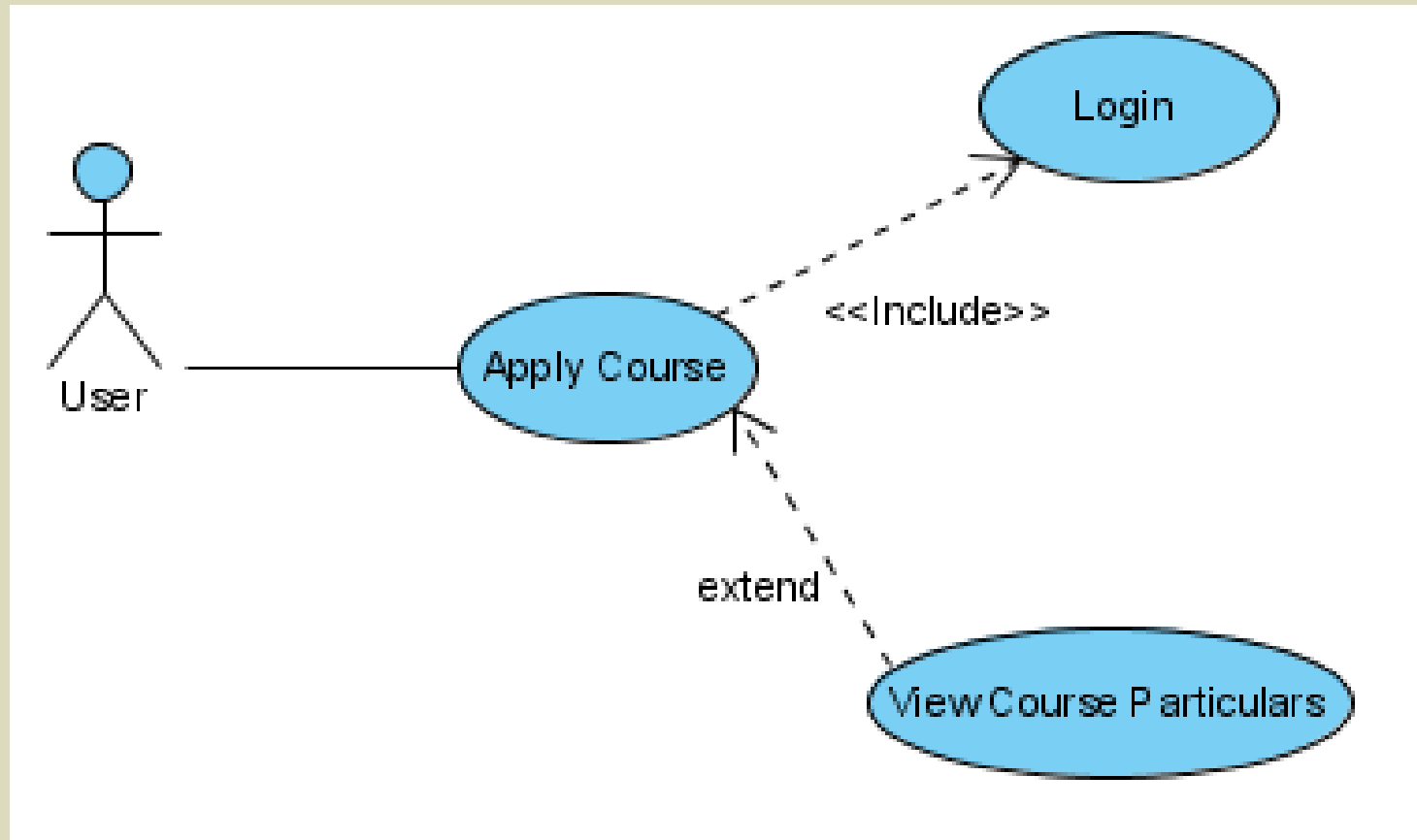
- Linking an optional use case to a standard use case.
- Example:

Register Course (standard use case)
may have *Register for Special Class*
(extend use case).
- The optional UC extends the standard UC
- Standard use case can execute without the extend case (**loose coupling**).

<<extend>>



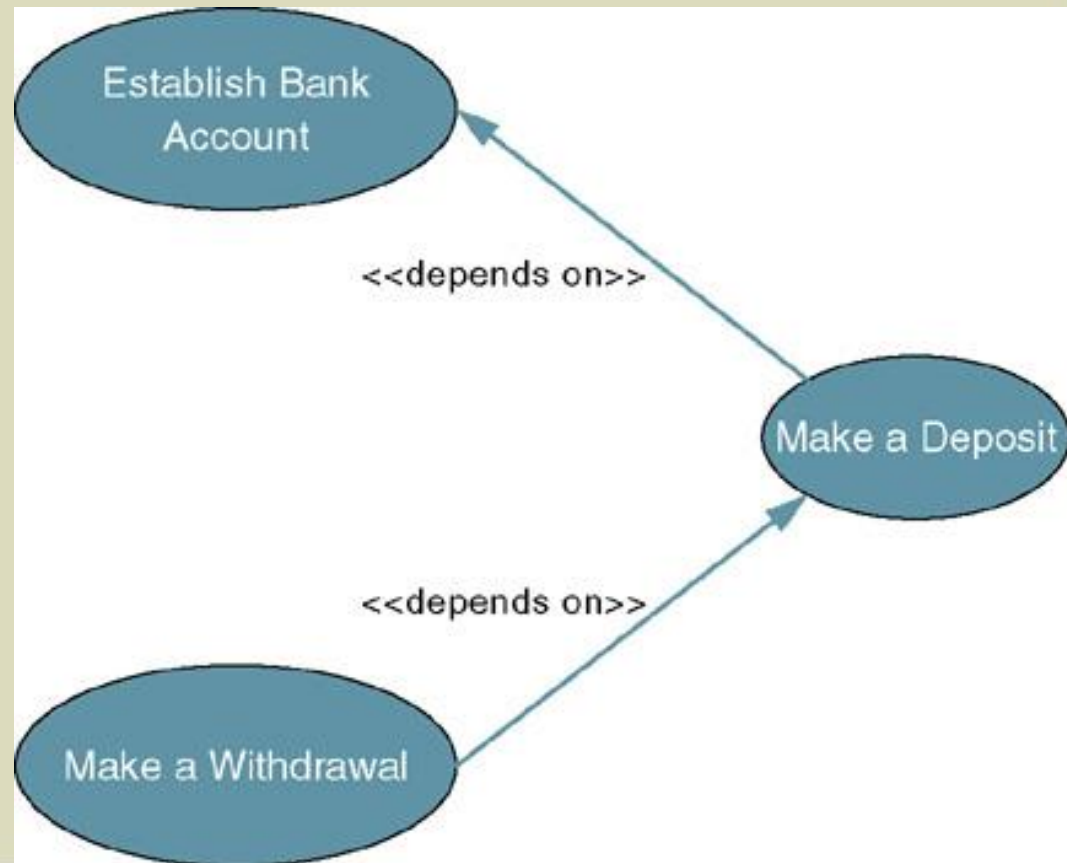
Example – Include and Extend



Use Case Depends On Relationship

Depends On – use case relationship that specifies which other use cases must be performed before the current use case.

- Can help determine sequence in which use cases need to be developed.
- Depicted as arrow beginning at one use case and pointing to use case it depends on.
- Labeled <<depends on>>.



Use Case - Boundary

- Boundary
 - A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.
 - A system boundary of a use case diagram defines the limits of the system.

Questions for Identifying People Actors

- Who is interested in the scenario/system?
- Where in the organization is the scenario/system be used?
- Who will benefit from the use of the scenario/system?
- Who will supply the scenario/system with this information, use this information, and remove this information?
- Does one person play several different roles?
- Do several people play the same role?

Questions for Identifying Other Actors

- What other entity is interested in the scenario/system?
- What other entity will supply the scenario/system with this information, use this information, and remove this information?
- Does the system use an external resource?
- Does the system interact with a legacy system?

Recipe for Creating Use Case

The short recipe for creating use cases is as follows:

1. Identify the actors
2. List their goals
3. Add brief descriptions to the goals
4. Create an initial use case for each goal
5. Describe the main success scenario for each use case
6. Identify the exceptions to the main success scenarios and work them out as extensions
7. Validate the use cases
8. Optimize the use cases

Courseware Management System

- Courses and Topics that make up a course.
- Manages Tutor who teach courses
- Course administrators who manage the assignment of the courses to tutors and as well as view courses.
- Course administrator and tutor manage the tutor information.
- Course administrator can view the tutor as well.
- Calendar or Course Schedule is generated as a result of the Students who refer to the Course schedule or Calendar to decide which courses for which they wish to take.
- Tutors can also view the course calendar.

Altered State University (ASU) Registration System

1. Professors indicate which courses they will teach on-line.
2. A course catalog can be printed
3. Allow students to select on-line four courses for upcoming semester.
4. No course may have more than 10 students or less than 3 students.
5. When the registration is completed, the system sends information to the billing system.
6. Professors can obtain course rosters on-line.
7. Students can add or drop classes on-line.

Grading System

- In abacus school, a teacher can record grades. Whenever teaches records the grades, they are also saved to disk. A teacher can update grades. Whenever grades are updated, the existing grade is loaded. Then the updated grade is saved to disk. A teacher, a registrar, and/or a student can view grades. Whenever any of these people view grades, they must always log onto the system. If their log on fails, they must re-authenticate their user name and password. In this system we can find a part-time student who is a kind of student. A registrar can generate report cards; the teacher can distribute report cards.

Registration of Course

- Registering a course at the Fast university for the undergraduate level, involves both the Academic manager and the student. The student submits to the academic department, the student Id and a desired course. The academic manager updates the final course selected by the student.

Fast Food management System

- Customers visit a Kolachi Restaurant that has a system that allows them (Customers) to make food orders by the help of service persons who take the orders. The system allows applicants as well to apply for jobs and the manager interviews and hire them. Any time a manager reorders supplies, he contacts the supplier after tracking the sales and Inventory data, to monitor deficiencies (shortage) in supplies before ordering. Whenever the manager produces management reports through the system, he does this after tracking the sales and inventory data to make sure that the data included in the report is up to date.