Contents

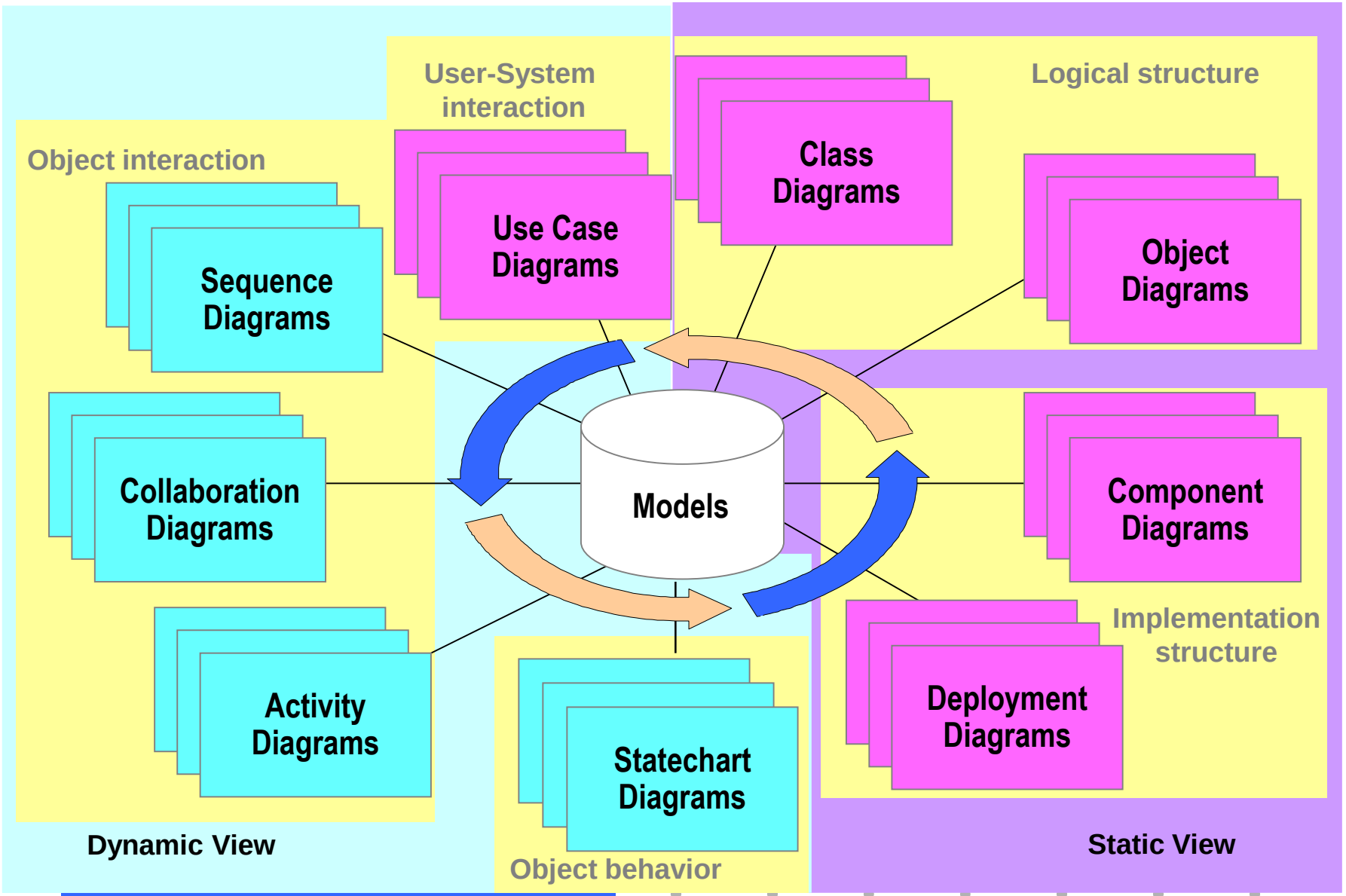
- Class and Object modeling
- Use-Case modeling

UML

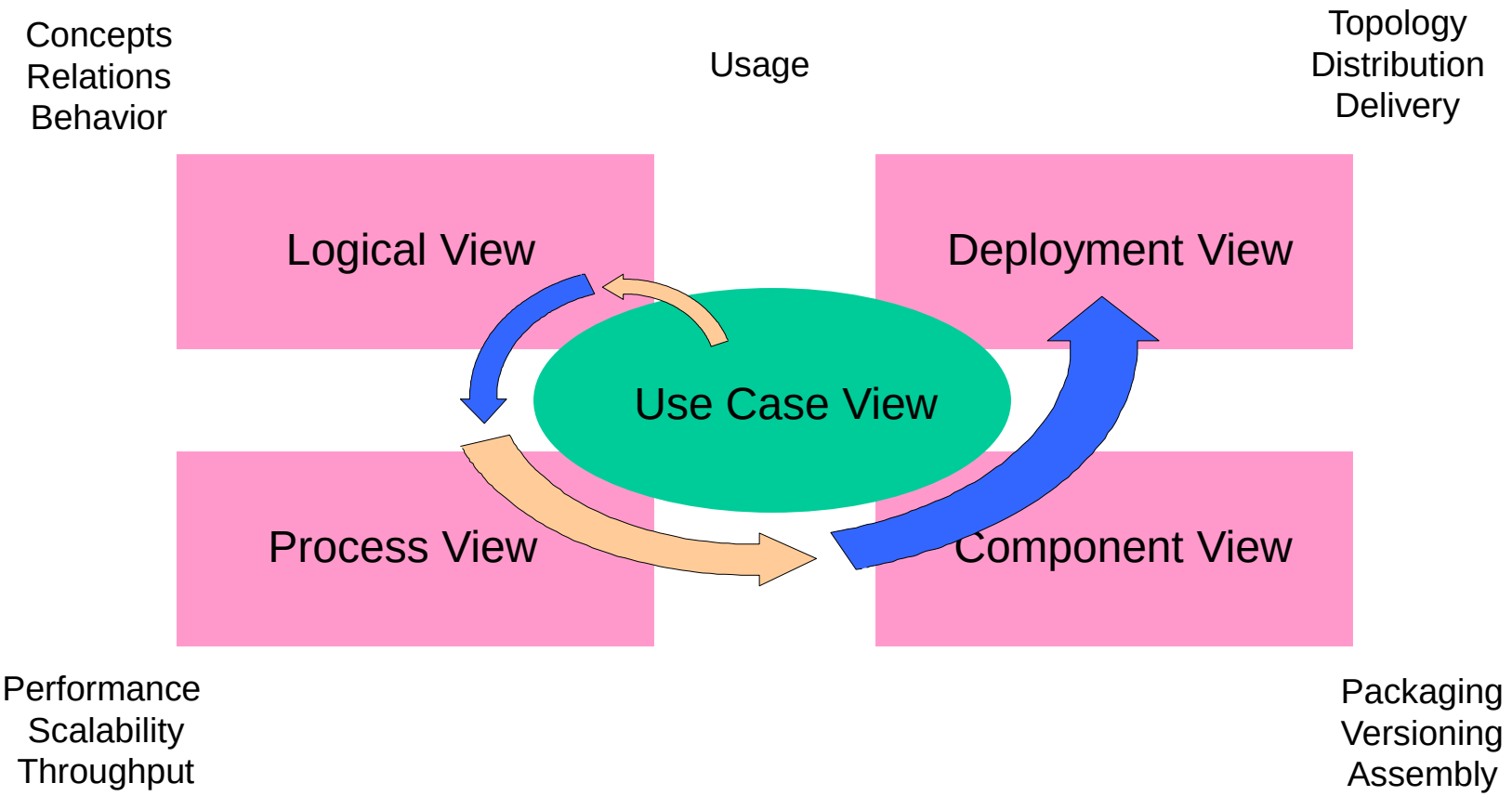
This is how the UML developers see their brain child:

"The **Unified Modeling Language** (UML) provides system architects working on object analysis and design with one consistent language for **specifying**, **visualizing**, **constructing**, and **documenting** the artifacts of **software systems**, as well as for business modeling."

UML Diagrams



4+1 Views


Concepts
Relations
Behavior
Usage
Topology
Distribution
Delivery
Logical View
Deployment View
Use Case View
Process View
Component View
Performance
Scalability
Throughput
Packaging
Versioning
Assembly

Purpose of Use-Case modeling

- Through Use-Case modeling basic needs of an actor are identified and it is described how these needs are satisfied by the system.
  - Who are the users of the system: Actors
  - Why do they use the system
  - How is each actor interacting with the system: the Use Cases
- The Use Cases describe the functions to be performed by the system from the perspective of certain actors
- The Use-Cases capture the requirements and help to trace the realization of the requirements during development

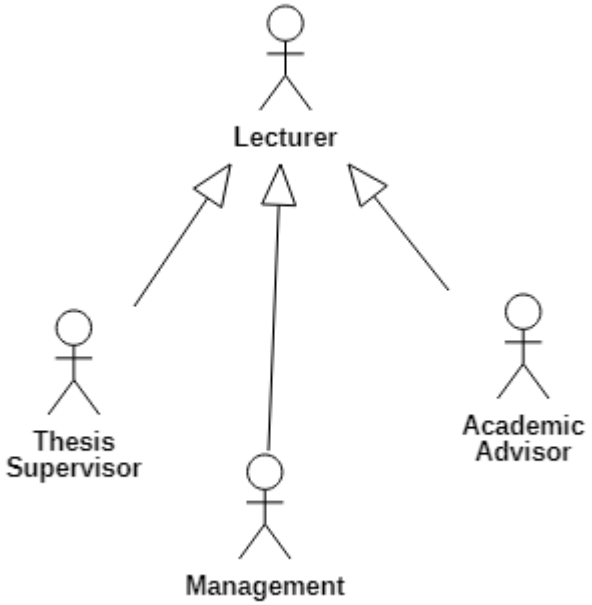
Actor

- An actor is a role played by an object outside of a system that interacts directly with the system
- A single object may play several roles and therefore be represented by several actor symbols in the model
- Actors either actively or passively participate in one or more Use-Cases

Staff Visitor

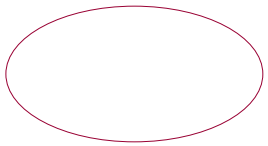
Actor

- An actor is a role played by an object outside of a system that interacts directly with the system
- A single object may play several roles and therefore be represented by several actor symbols in the model
- Actors either actively or passively participate in one or more Use-Cases



Use Case

- A Use-Case is the description of the interaction between an actor and a system that helps to fulfil an actors needs
- A Use-case is a step-by-step description of the interaction between an actor and a system to yield one specific result (out of many),
- The description can be formal or informal
- The Use Case descriptions are used to communicate with the Stakeholders of the system. They serve as a contract between the Stakeholders.



ChangeFloor

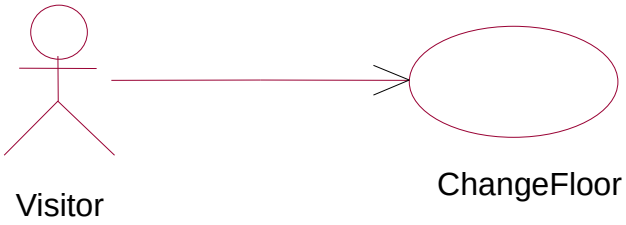
Activity: Verb or Gerund

A verbal noun

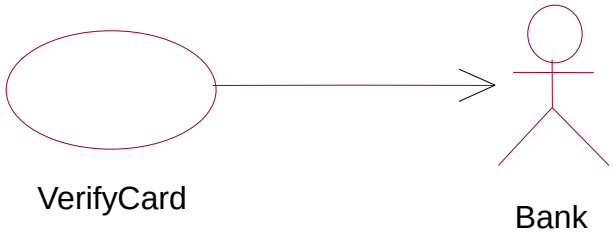
Camel Case

Relationships

- Actors are classes. They have the same relationships as classes
  - Specialization/Generalization
  - Associations
- Actors are involved in the execution of Use Cases
  - This is indicated by associations
  - The actor initiates the use case

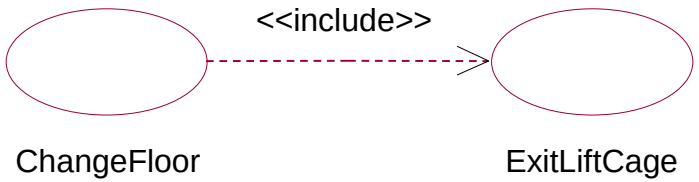


  - The use case triggers the actor

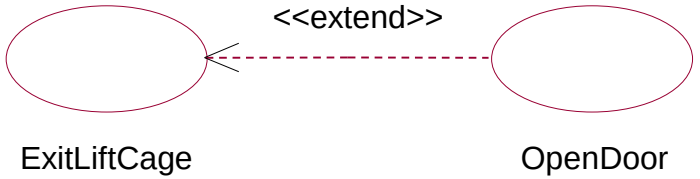


Use-Case Relationships

- Use Case are related through dependencies. There are two standard dependency-stereotypes:
  - An **include** dependency from Use Case A to Use Case B indicates that the Use Case A refers to and includes the behaviour as specified by B
    - This is used to separate often occurring sub-Use-Cases

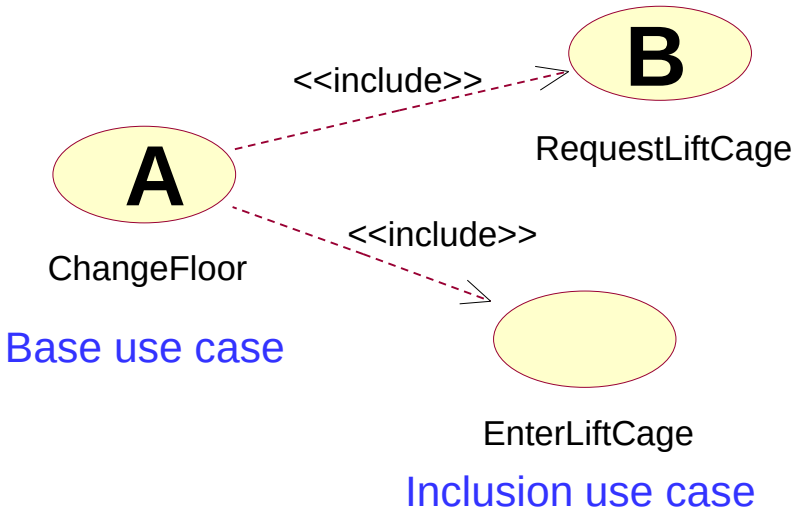


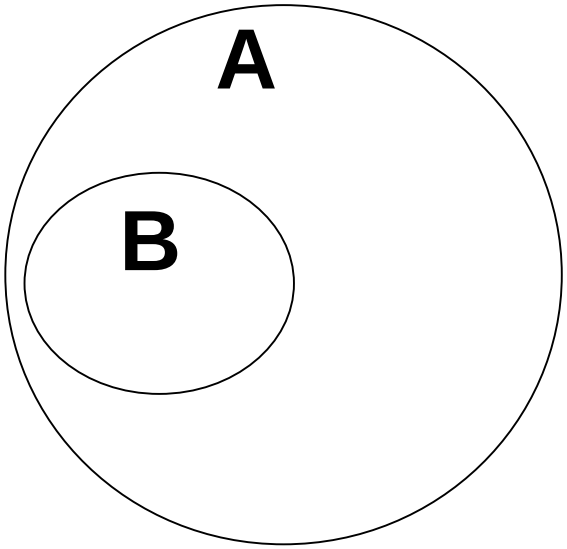
  - An **extend** relationship from Use Case A to Use Case B indicates that the Use Case B may optionally be extended with the behaviour specified by A. The description of B does not refer to A, A refers to B and indicates at what extension point it is to be inserted in A
    - This is used to separate important optional parts of a Use-Case that are best described separately



Include

- An include relationship between two Use Cases means that the base Use Case explicitly incorporates the behavior of another Use-case at a location specified in the base Use Case
  - Used to avoid describing the same flow of events several times
  - Used when similar behavior recurs across more than one use case



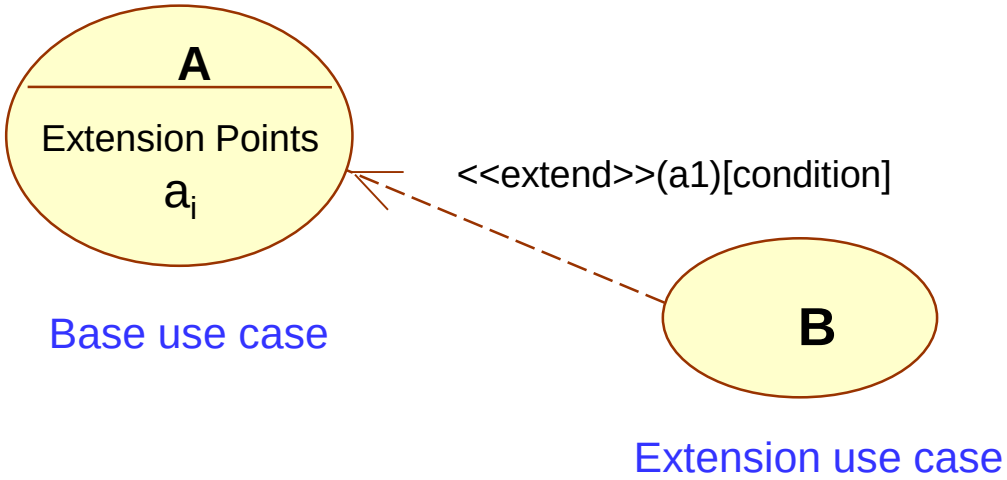


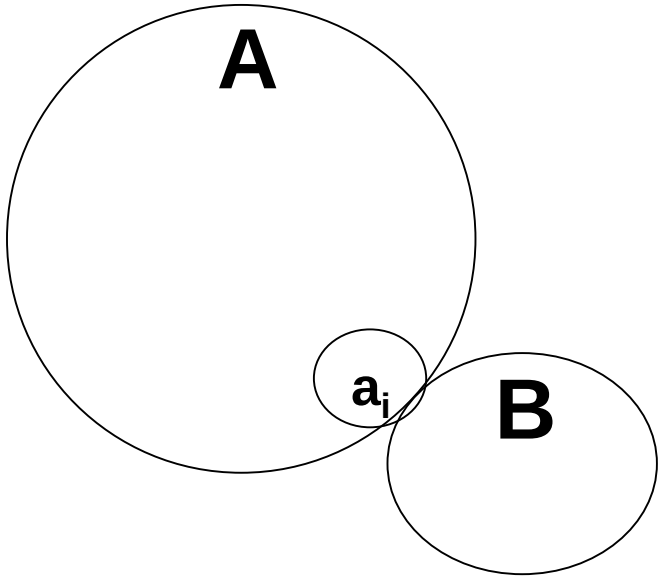
B part of A

B one of steps in A

Extend

- An “extend” relation between Use Cases means that the base Use Case may incorporate the behavior of the extension Use Case at the extension point(s) defined in the base Use Case but only if the possible condition is fulfilled
  - An extension Use-case is not necessarily a separately executable Use Case.

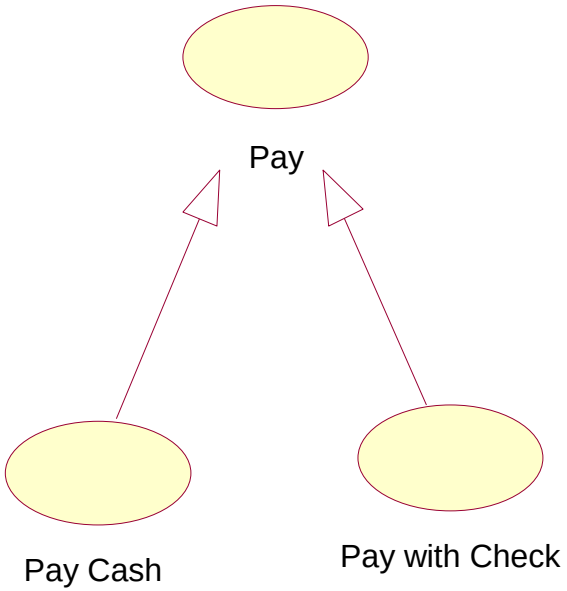




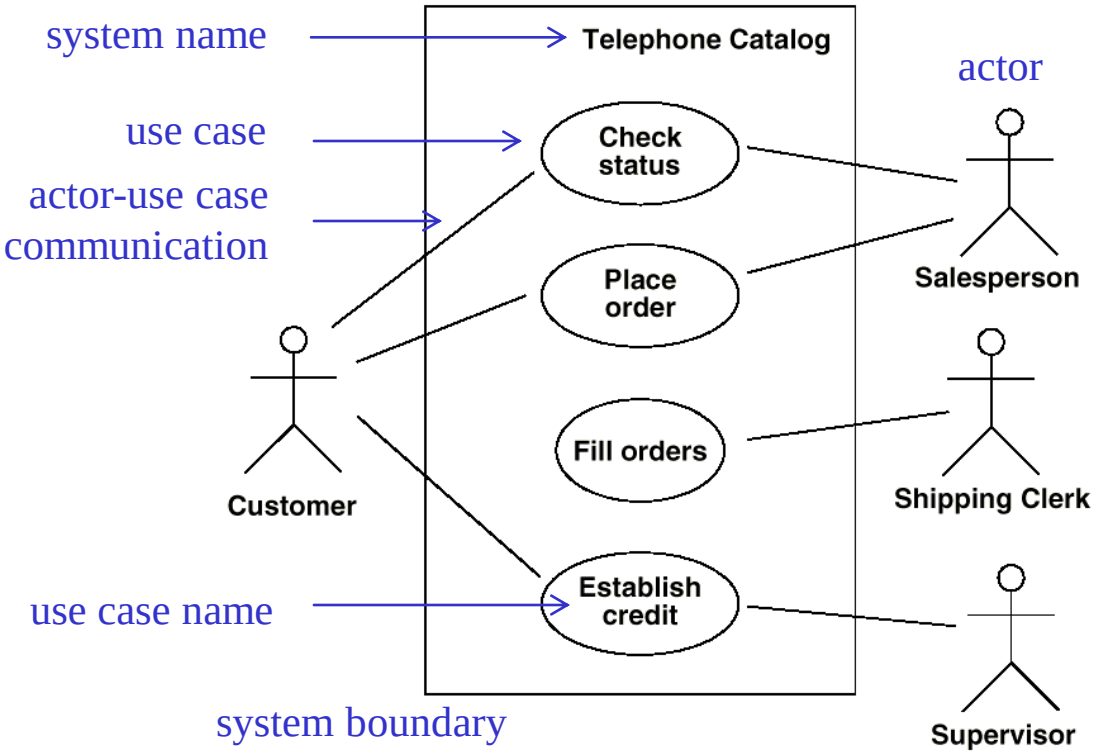
When A occurs, at the point of a_i, B may or may not occur

Use-Case Relationships

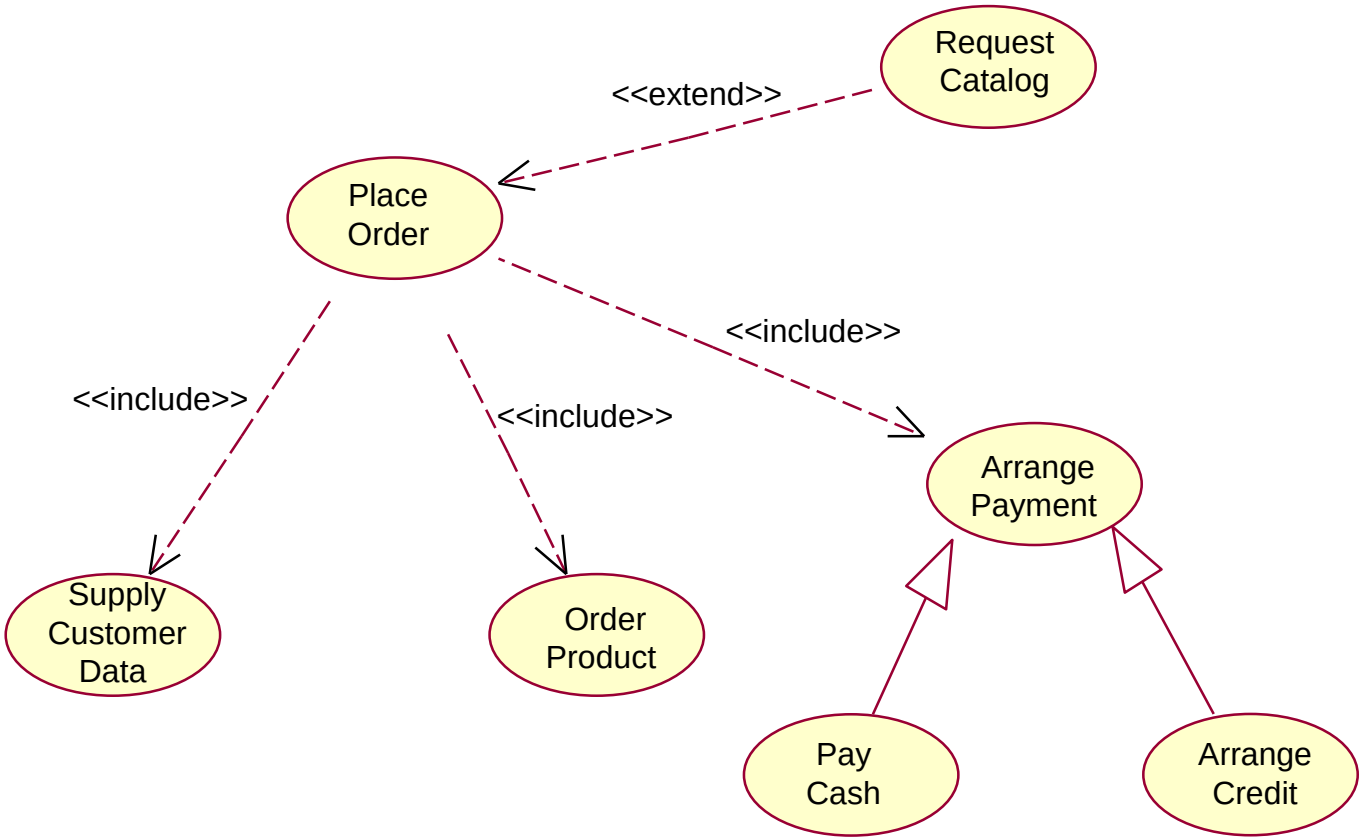
- Use Cases have also generalization and specialization relationships
  - In a more specialized Use Case more specific choices are made for some actions



Example UCD


system name
Telephone Catalog
actor
use case
Check status
actor-use case communication
Place order
Salesperson
Customer
Fill orders
Shipping Clerk
use case name
Establish credit
Supervisor
system boundary

Part of a Model



Use-Case Modeling

- Steps:
  - Find the end-users.
  - Find the actors (end-users roles)
  - Define the main Use Cases
  - Look for sub-Use-Cases
  - Find the Use-Case relationships
  - Describe each Use Case in a Use-Case Specification document

Find the End-Users

- Which people or systems are interacting with the system in one way or another
  - Which user employ the system's most obvious main functions
  - Which users employ the secondary functions, such as system maintenance and administration?

- The Vision document identifies stakeholders and end-users.
- End-users are stakeholders, but not all stakeholders are end-users

- Example: name some end-users for the hospital-lift case

End-Users in the Lift-Case

- End-Users
  - Visitor
  - Patient
  - Staff
    - Doctor
    - Nurse
    - Warden
    - Lift mechanic

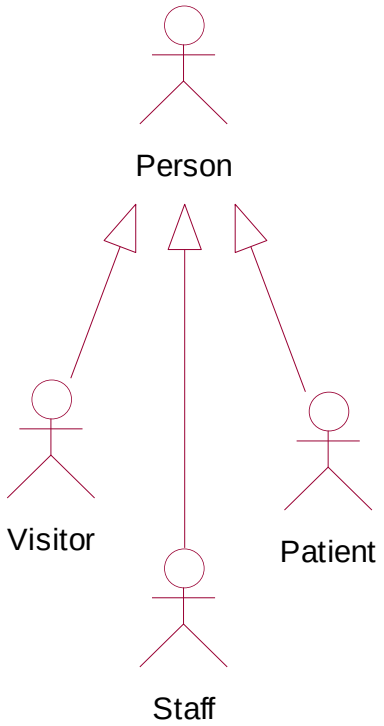
Find the Actors

- Determine the roles of each end-users with respect to the system.
- An end-user may have many roles, and a role may be played by many end-users. The end-user may only play one role at a time.

- Classify the roles and determine the actors.

Actors in the Lift-Case

- End-Users
  - Visitor
  - Patient
  - Staff
    - Doctor
    - Nurse
    - Warden
    - Lift mechanic



Actors and the Vision Document

- End-Users of the hospital lift system

Name	Description	Stakeholder
Warden	Care taker of the building	Hospital principal
Patient	Person treated in hospital	Hospital principal
Visitor	Acquaintance of the patient	Hospital principal
Nurse	A patients care taker	Hospital principal
Doctor	Diagnoses and treats the patients	Hospital principal
Maintainer	Diagnoses and cares for the lift system	Maintenance department

- Example user's profile

Patient

Representative	Hospital principal
Description	A person being treated in hospital
Type	CASUAL USER without any technical background.
Responsibilities	As a real patient his wishes with respect to floor movements are entered via a nurse
Success Criteria	Painless transportation
Involvement	Not involved
Deliverables	None
Comments / Issues	No problems expected

Define Use Cases

- For each Actor that has been identified ask the following questions:
  - What are the primary tasks that the Actor wants the system to perform?
  - Will the Actor create, store change, remove or read data in the system?
  - Will the Actor need to inform the system about sudden, external changes?
  - Does the Actor need to be informed about certain occurrences in the system?
- Name each Use Case in the list and define its scope

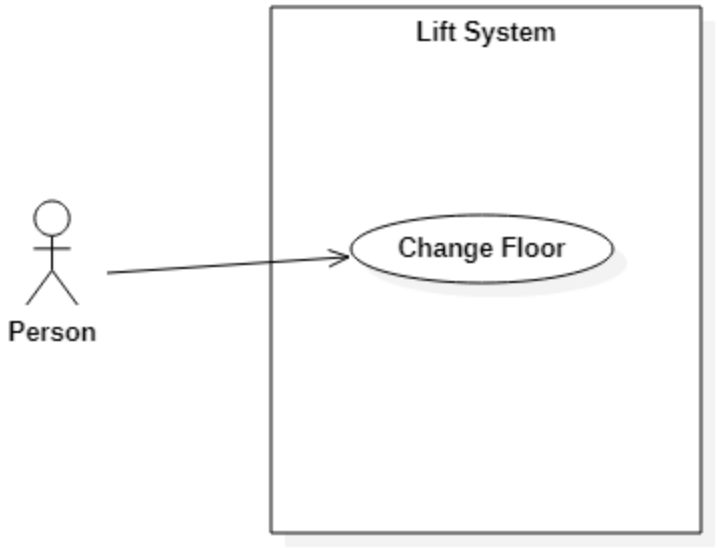
- Exercise: name some Use Cases for the hospital-lift case for each identified actor

Make Use-Case Model

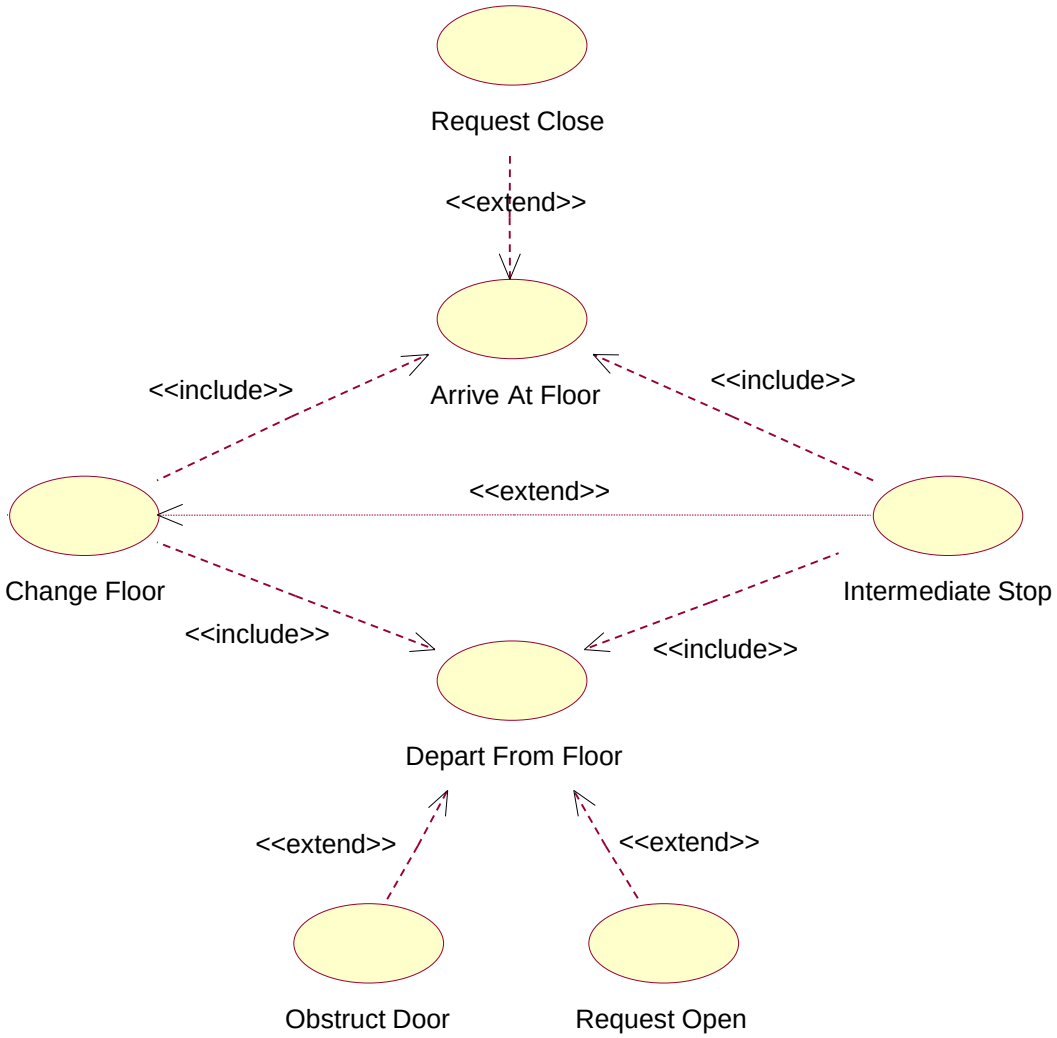
- Create a design project.
- Create a context diagram (Main Use Case Diagram).
- Draw actors and main Use Cases
  - Show the direction of the associations between them
- For each main use case, define sub-Use-Cases with sufficient granularity to see how actors are involved and what are the major forms of interaction between the actors and the system
- Draw each detail of main use case in a separate use case diagram.
- Indicate the relationships
- Check that the model is in accordance with the Use-Case descriptions

- Example: Make a Use-Case model for the lift system

Context Diagram: Lift System



Use Case Diagram: Change Floor



Describe the Use Case

- Define normal flow of events
  - How and when the use case begins and ends (including which actor initiates it)
  - When the use case will interact with the actors and what is exchanged with them
  - How and when the use case will need data stored in the system, or will store data in the system
- Exceptional events
  - Are there any exceptional events that may start an alternative flow of events
- Pre-conditions
  - Describes the conditions that must be met before the use case can start
- Post-conditions
  - Defines the main conditions that are fulfilled at the end of the flow of events defined in this use case

Use-Case Description (RUP Template)

Document Sections

1. Use Case Name
    - 1.1 Brief Description
2. Flow of Events
    - 2.1 Basic Flow
    - 2.2 Alternative Flows
    - 2.3 Exceptional Flows
3. Special Requirements [Non-functional requirements]
    - 3.1 < First special requirement >
4. Pre-Conditions
    - 4.1 < Pre-condition One >
5. Post-Conditions
    - 5.1 < Post-condition One >
6. Extension Points
    - 6.1 <name of extension point>

Description of Cockburn's Template

Use Case

Name:	<the name should be the goal as a short active verb phrase>
Context of use:	<a longer statement of the goal, if needed, its normal occurrence conditions>
Scope:	<design scope, what system is being considered black-box under design>
Level:	<one of: Summary, User-goal, Sub-function>
Primary Actor:	<a role name for the primary actor, or description>
Stakeholders & Interests:	<list of stakeholders and key interests in the use case>
Precondition:	<what we expect is already the state of the world>
Minimal Guarantees:	<how the interests are protected under all exits>
Success Guarantees:	<the state of the world if goal succeeds>
Trigger:	<what starts the use case, may be time event>

Main Success Scenario:

Put here the steps of the scenario from trigger to goal delivery, and any cleanup afterwards.

<step #> <action description>

Extensions

Put here there extensions, one at a time, each referring to the step of the main scenario.

<step altered> <condition>: <action or sub-use case>
<step altered> <condition>: <action or sub-use case>

Technology and Data Variations List

Put here the variations that will cause eventual bifurcation in the scenario.

<step or variation # > <list of variations>
<step or variation # > <list of variations>

Related Information

<whatever your project needs for additional information>

Description of Cockburn's Template

Use Case

Name:	<the name should be the goal as a short active verb phrase>
Context of use:	<a longer statement of the goal, if needed, its normal occurrence conditions>
Scope:	<design scope, what system is being considered black-box under design>
Level:	<one of: Summary, User-goal, Sub-function>
Primary Actor:	<a role name for the primary actor, or description>
Stakeholders & Interests:	<list of stakeholders and key interests in the use case>
Precondition:	<what we expect is already the state of the world>
Minimal Guarantees:	<how the interests are protected under all exits>
Success Guarantees:	<the state of the world if goal succeeds>
Trigger:	<what starts the use case, may be time event>

Main Success Scenario:

Put here the steps of the scenario from trigger to goal delivery, and any cleanup afterwards.

<step #> <action description>

Extensions

Put here there extensions, one at a time, each referring to the step of the main scenario.

<step altered> <condition>: <action or sub-use case>
<step altered> <condition>: <action or sub-use case>

Technology and Data Variations List

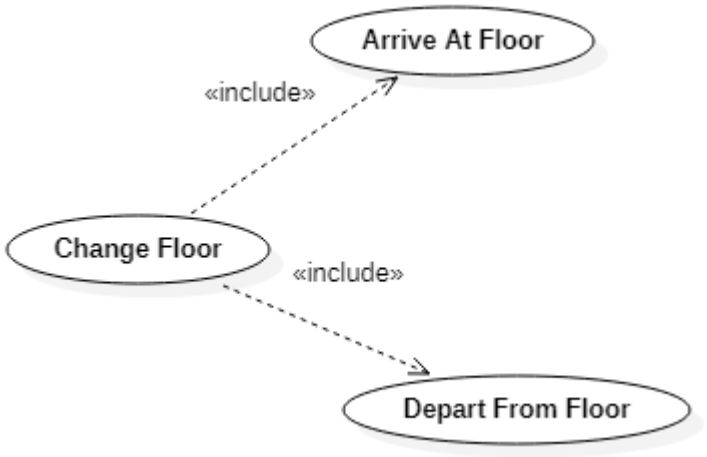
Put here the variations that will cause eventual bifurcation in the scenario.

<step or variation # > <list of variations>
<step or variation # > <list of variations>

Related Information

<whatever your project needs for additional information>

Use Case Description: Normal Flow



Use-Case Description: Normal Flow

Main success scenario of "Change Floor"

This Use Case describes the situation in which any person using the hospital lift intends to change from floor k ($0 \leq k < N$) to floor m ($0 \leq m < N$, $m \neq k$).

1. The person pushes a lift-request button: *the person is at floor k and requests a lift for going up or down depending on m>k or m<k.*
2. A lift cage **Arrives At Floor** k: *after a while a lift cage with the appropriate direction arrives*
   - 2.1 The lift decelerates and stops
   - 2.2 The chime sounds and the button light extinguishes.
   - 2.3 The door opens.
3. The person enters the lift cage possibly together with other persons
4. The person pushes a floor-request button for floor m.
5. Lift cage **Departs From Floor** k
   - 5.1 The door closes: *after a certain amount of time passes, the door closes automatically.*
   - 5.2 The lift accelerates until it reaches its normal speed
6. Lift cage **Arrives At Floor** m
7. User exists the lift cage.

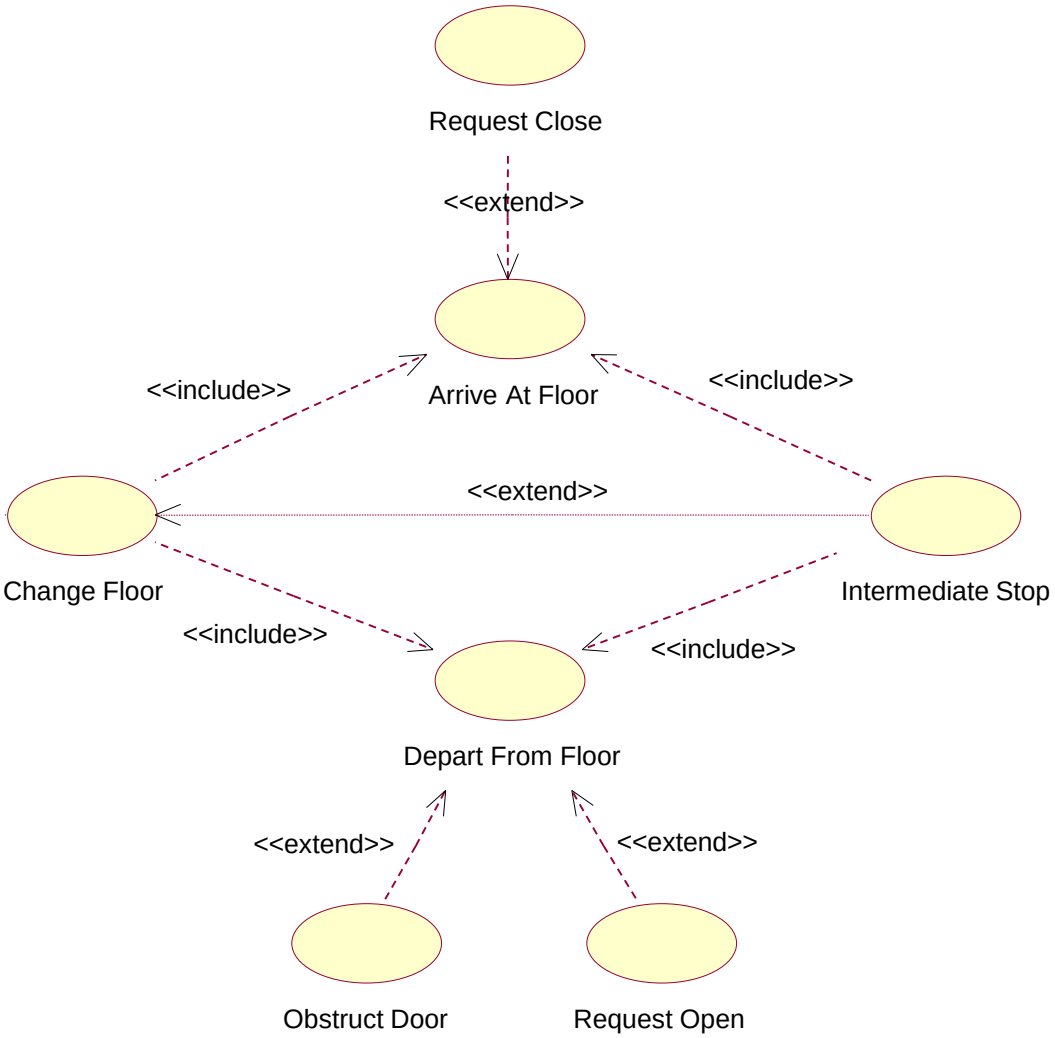
Often, steps 1 and 4 can be omitted because the buttons where already pressed by other actors. Also a lift may already have arrived when the person arrives. Step 4 can be done any time between steps 3 and 6.

Details of Use Case

- A Use-Case description focuses on the view of an actor and not on the ideas and needs of the (software) engineer.

- A Use Case should be as detailed as needed to understand how the system is used, i.e. a Use Case does not have to contain all details
  - Use Case ChangeFloor does not have to specify how long it will take before the doors will close again or how fast they close; yet, this needs to be specified somewhere
- A Use Case only models how the user perceives the system under study, it should not contain any details of the 'implementation'
  - Use-case model does not specify how are the doors operated, if lights are visible, how the system knows that it has reached the required floor, etc; but again, also this needs to be specified somewhere
  - The terminology of the user should be sufficient to describe the Use Cases

Use Case Description: Extends



Use-Case Description: Extends

Extensions

On sub-Use-Case **Arrive At Floor**:

2.3a As long as the door is open the person can **Request** the door to **Close**.

2.3a.1 Person pressing the close button

2.3a.2 The door starts closing immediately

On sub-Use-Case **Depart From Floor**:

5.1a As long as the door is closing something may **Obstruct** the **Door**.

5.1a.1 The obstruction interacts with the door

5.1a.2 The door opens completely immediately.

5.1b As long as the door is closing the person can **Request** the door to **Open**. Pressing the button, the door starts opening immediately.

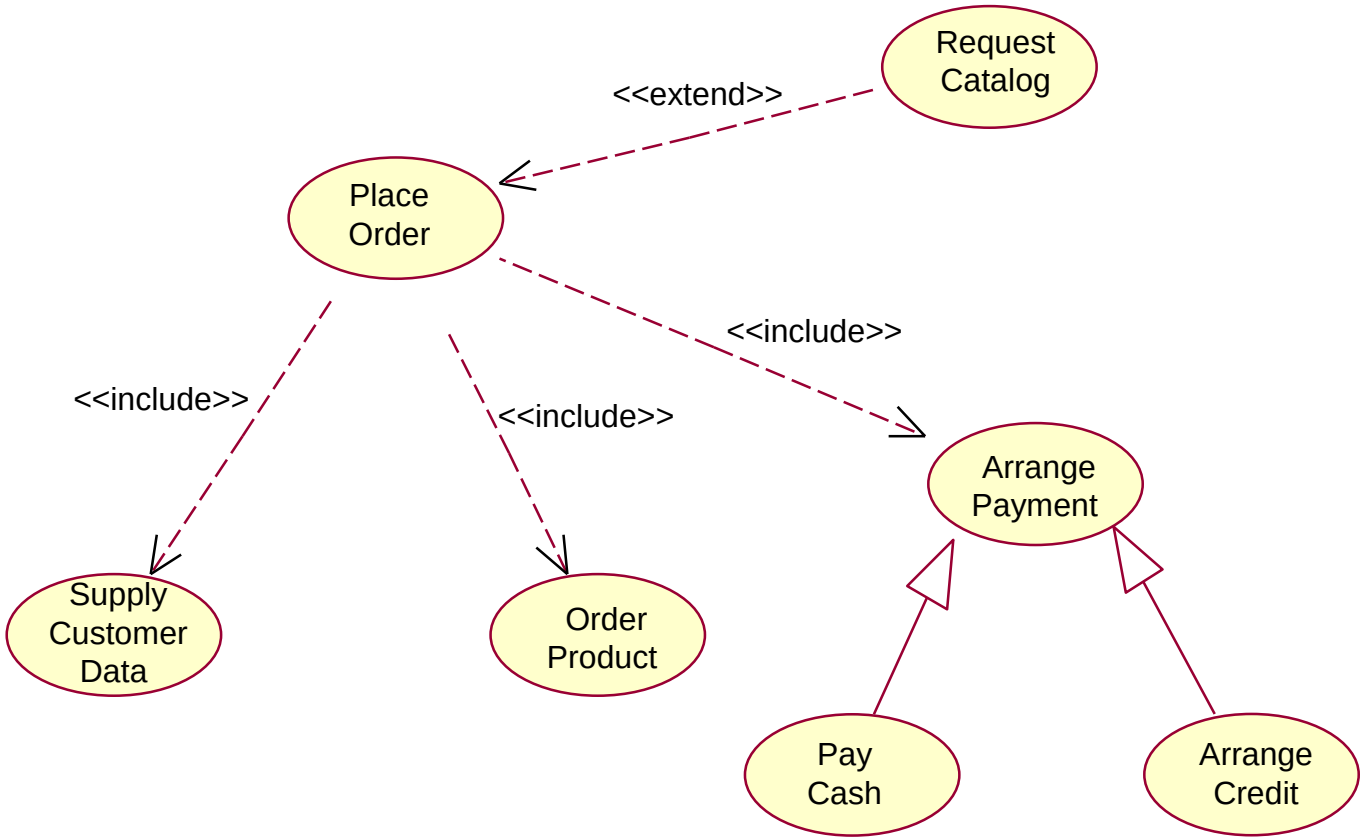
5a The lift can make several **Intermediate Stops** after departure

5a.1 The lift cage **Arrives At Floor** intermediate between departure and destination floor.

5a.2 Other persons enter the lift cage

5a.3 The lift cage **Departs From** the intermediate **Floor**

Use Case Description: Generalization



Use-Case Description: Generalization

Main success scenario of “Place Order”

This Use Case describes the situation in which any customer want to buy an product.

1 The customer **Order Product** from the viewed product list.

1.1 …

1.2 …

2 The customer **Supply Customer Data** on the available form.

2.1 …

2.2 …

3 The customer **Arrange Payment** of the product he has ordered..

Alternative scenario of “Arrange Payment”

On Use-Case **Place Order**

3a The customer **Pay Cash** of the product he ordered

1.1 …

1.2 …

3b The customer **Arrange Credit** for the payment of the product he ordered.

2.1 …

2.2 …

Review the Use-Case model

- Review the Use-case model against other existing system models:
  - Analysis Object model
- Review the Use-case model against the Stakeholder Requests and the Vision

Prioritizing the Use-cases

- Not all Use Cases should be implemented in the elaboration phase
  - To identify the architecture it is important to not drown in too many details
- The architecture can be based on a simpler version of the system

- Setting priorities may help select the Use Cases that should be addressed in each iteration
  - The degree of support of the architecture for extensibility can be “tested” by adding Use Cases in later iterations

Exercise

- Make an Use Case diagram for the Solitaire game
- Make a Use Case description for the Solitaire game