UNIVERSITY of GUELPH

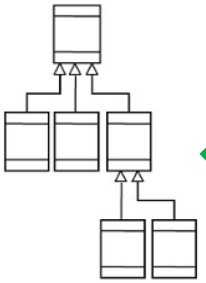
ENGINEERING

Use Cases

Fattane Zarrinkalam

Fowler: UML Distilled: A Brief Guide to the Standard Object Modeling Lang. 3rd Ed. (Chapter 9)

UML Diagrams

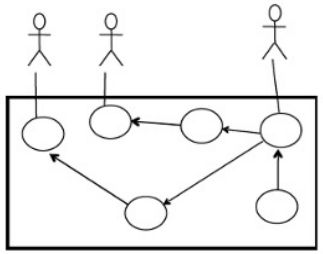


UML Class Diagrams

information structure

relationships between data items

modular structure for the system

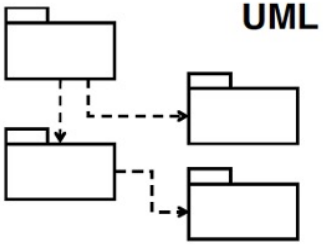


Use Cases

user's view

Lists functions

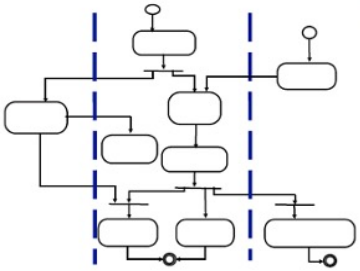
visual overview of the main requirements



UML Package Diagrams

Overall architecture

Dependencies between components

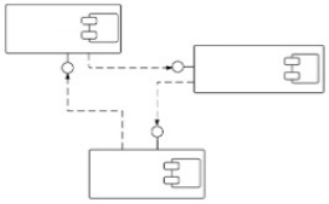


Activity diagrams

business processes;

concurrency and synchronization;

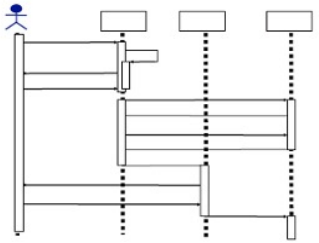
dependencies between tasks;



UML Component Diagrams

static implementation view of a system

Interfaces between components



UML Sequence Diagrams

individual scenario

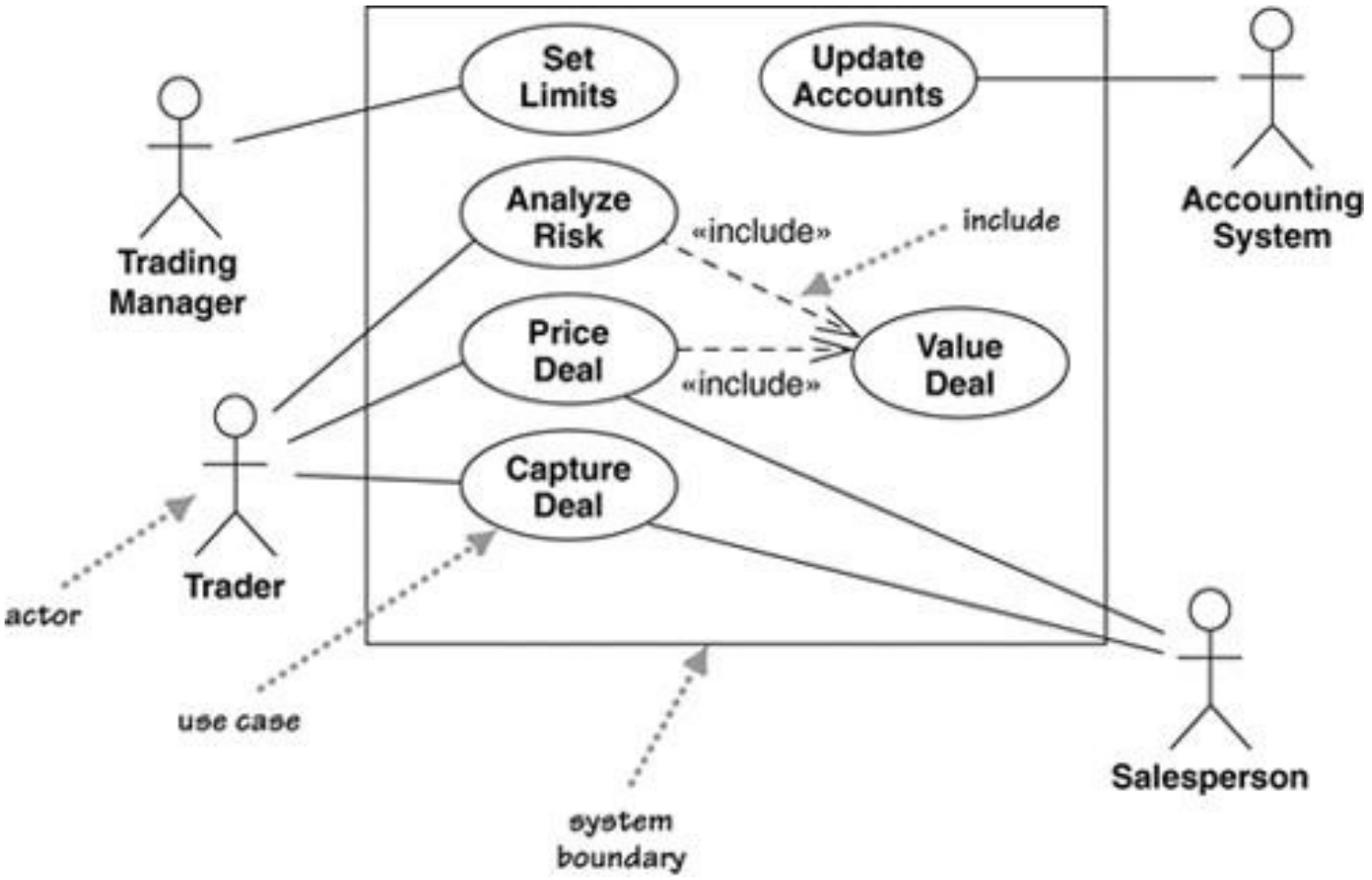
interactions between users and system

Sequence of messages

Use Case Modeling

- Typically starts early in a project and continues throughout a system development process.
- Is usually used during requirements activities to capture requirements that define what a system should do.
- It is usually done in a series of sessions between users and analysts of a system.
- A use-case diagram may be considered a "table of contents" for the functional requirements of a system.
  - Elements
    - Actors
    - Use Cases
    - Relationships

Use Case Diagram

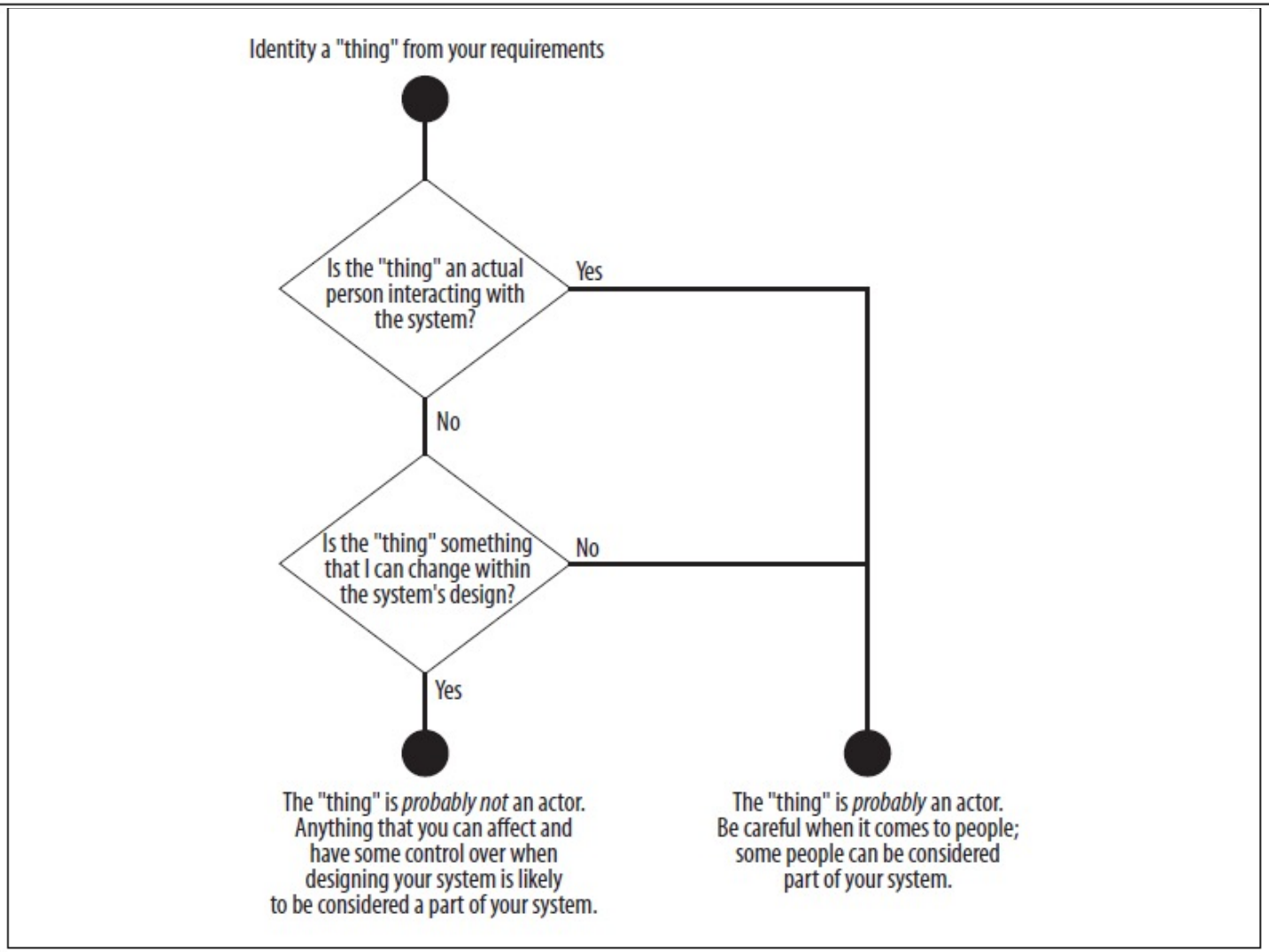


Actors

Actors

- An actor is a user or external system with which a system being modeled interacts.
- An actor is a role that a user plays with respect to the system.

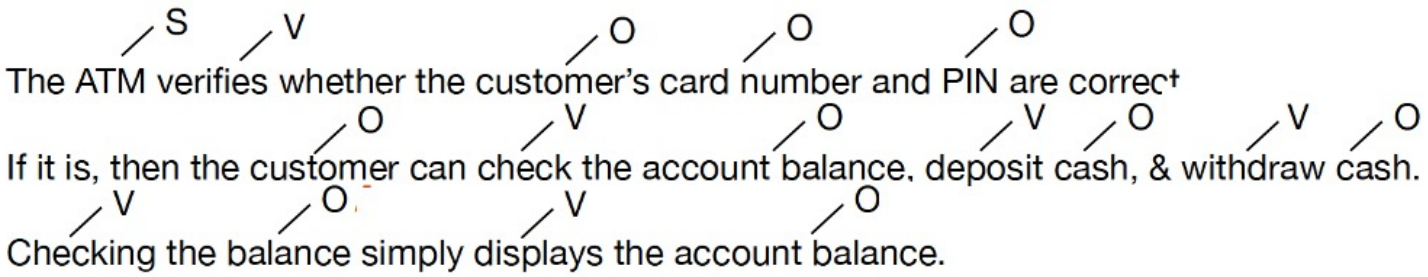
Finding Actors



K. Hamilton and R. Miles “Learning UML 2.0. O'Reilly”, ISBN-10: 0- 596-00982-8, 2006.

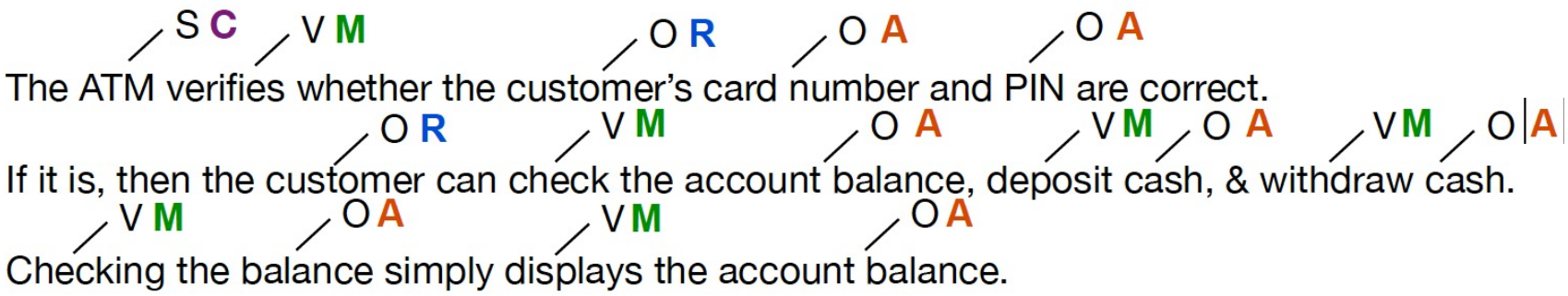
Finding Actors

- Browse through existing documents
  - noun phrases may be domain classes
  - verb phrases may be operations and associations
  - possessive phrases may indicate attributes



From Actors

- Analyze each subject and object as follows:
  - Does it represent a person performing an action? Then it is an actor, “R”
  - Is it also a verb (such as ‘deposit’)? Then it may be a method, “M”
  - Is it a simple value, such as “colour” (string) or “money” (number)?
    - Then it is probably an attribute, “A”
  - Which NPs are unmarked? Make them “C” for class

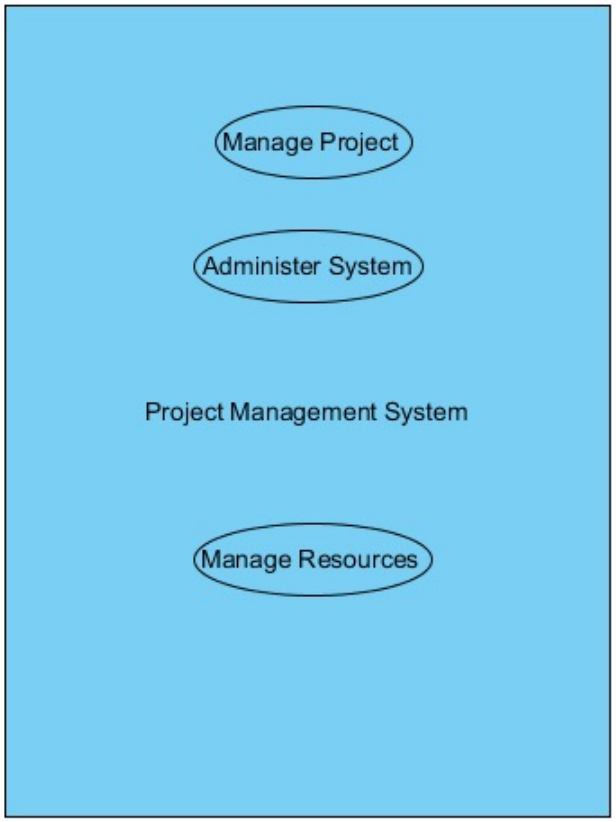


Use Cases

Use Case

- A use case is a functional requirement that is described from the perspective of the users of a system.

- Each use case is composed of one or more behavior sequences.
  - Action: processing performed by the system
  - Interaction: communication between the system and the actors



Describing Use Case

- For each use case:
  - a “flow of events” document, written from an actor’s point of view
  - describes what the system must provide to the actor when the use case is executed.
- Typical contents
  - How the use case starts and ends;
  - Normal flow of events;
  - Alternate flow of events;
  - Exceptional flow of events;
- Documentation style:
  - English language description
  - Sequence Diagrams - good for detailed design
  - Activity Diagrams - good for business process

Use Case Content

Buy a Product

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirmation email to customer

Extensions:
3a: Customer is Regular Customer
    1. System displays current shipping, pricing and billing information
    2. Customer may accept or override these defaults, returns to MSS at step 6
6a: System fails to authorize credit card
    1. Customer may re-enter credit card information or may cancel

Finding Use Cases

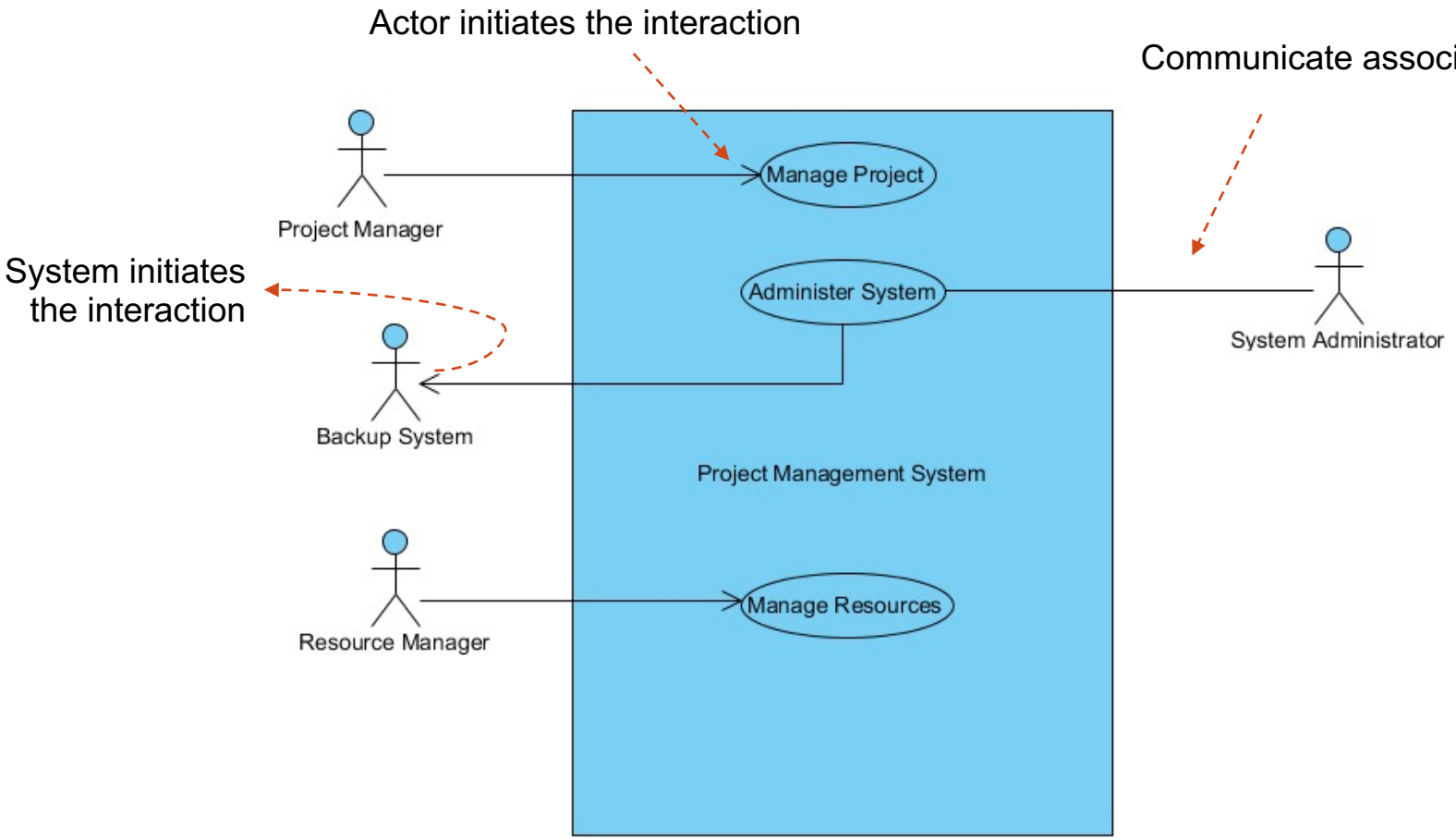
- For each actor, ask the following questions:
  - Which functions does the actor require from the system?
  - Does the actor need to read, create, destroy, modify, or store some kinds of information in the system ?
  - Does the actor have to be notified about events in the system?
  - Does the actor need to notify the system about something?
  - Could the actor's daily work be simplified or made more efficient through new functions provided by the system?

Relationships

Relationships

- Relationships in Use Case Diagram
  - Communicate Association
  - Dependency
    - Include
    - Extend
  - Generalization
    - Actor Generalization
    - Use Case Generalization

Communicate Association

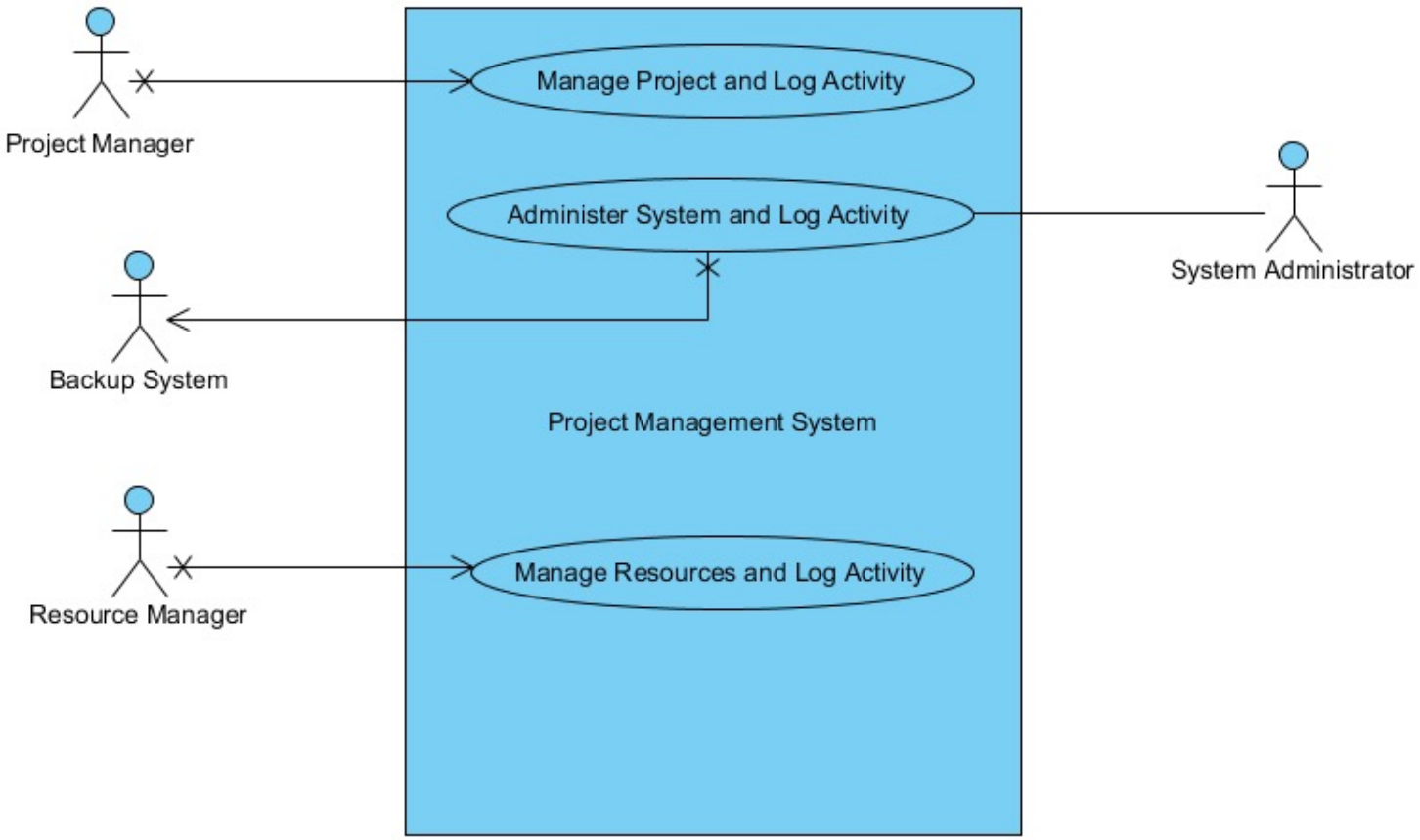


Dependency

- An include dependency from one use case (called the base use case) to another use case (called the inclusion use case) indicates that the base use case will include or call the inclusion use case.

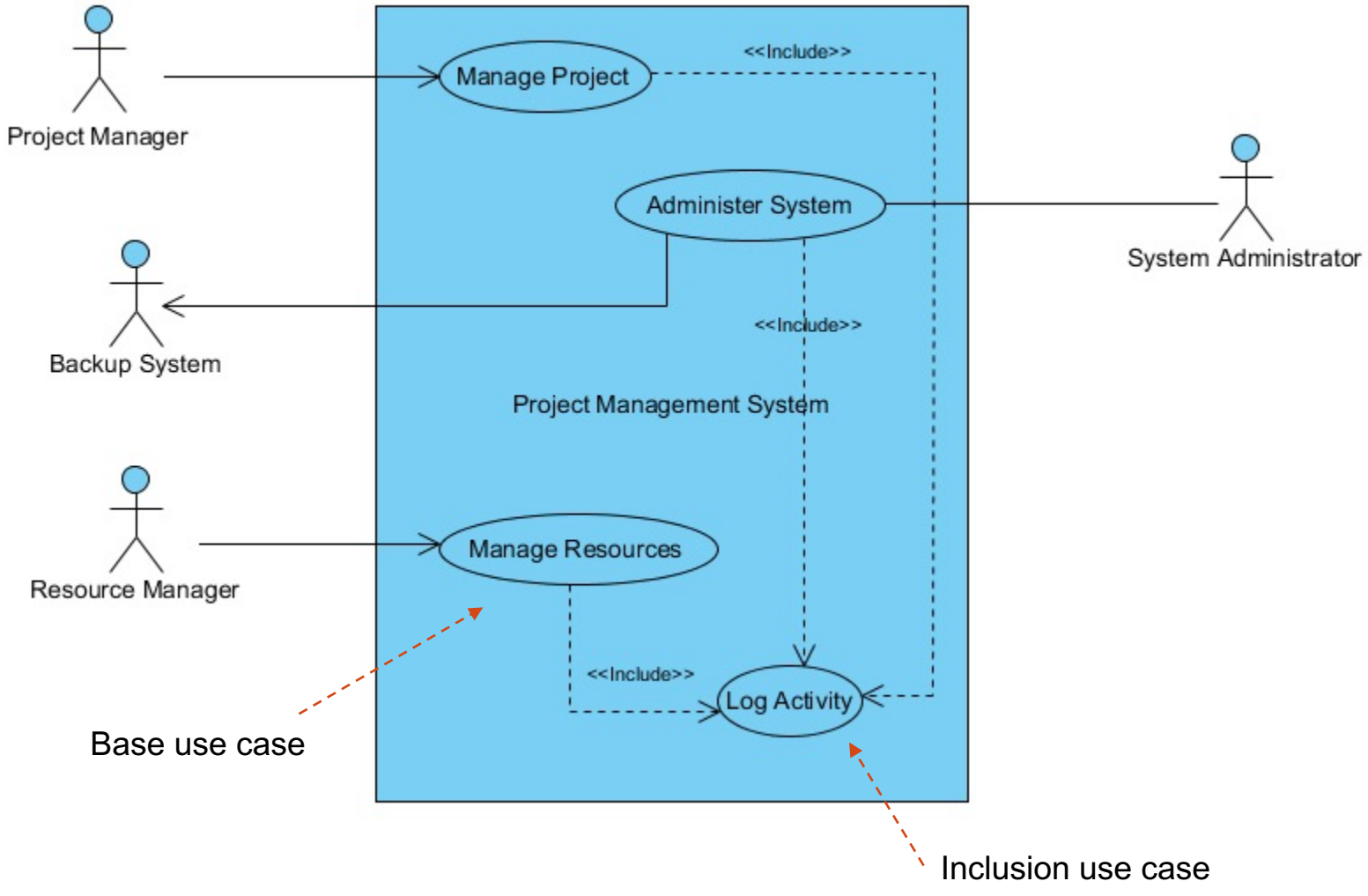
- An extend dependency from one use case (called the extension use case) to another use case (called the base use case) indicates that the extension use case will extend (or be inserted into) and augment the base use case.

Dependency :: Include



Log activity is common to these three use cases

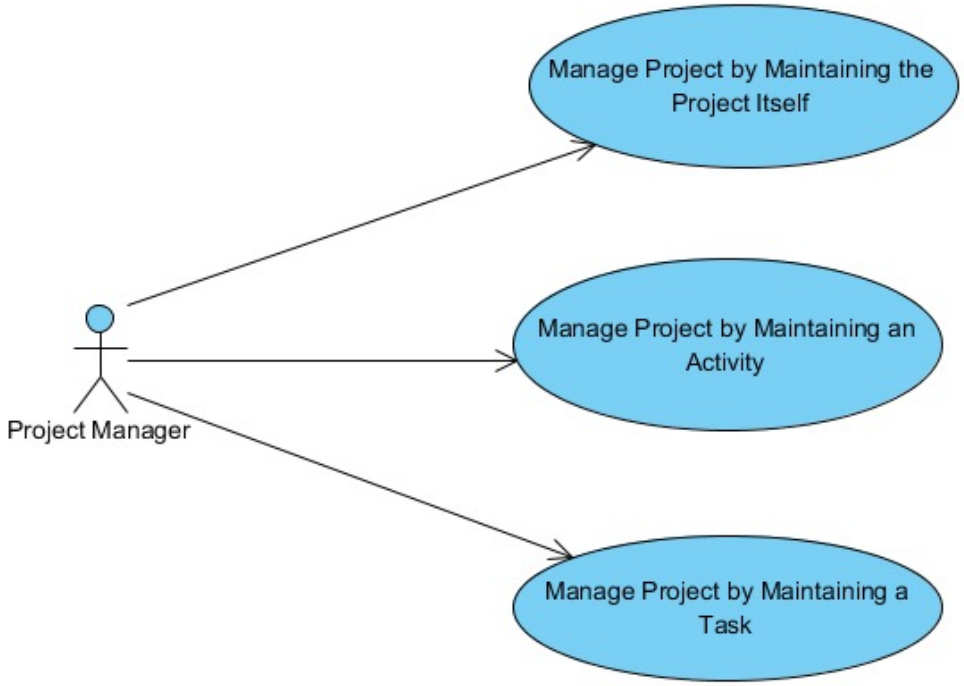
Dependency :: Include



Dependency :: Include

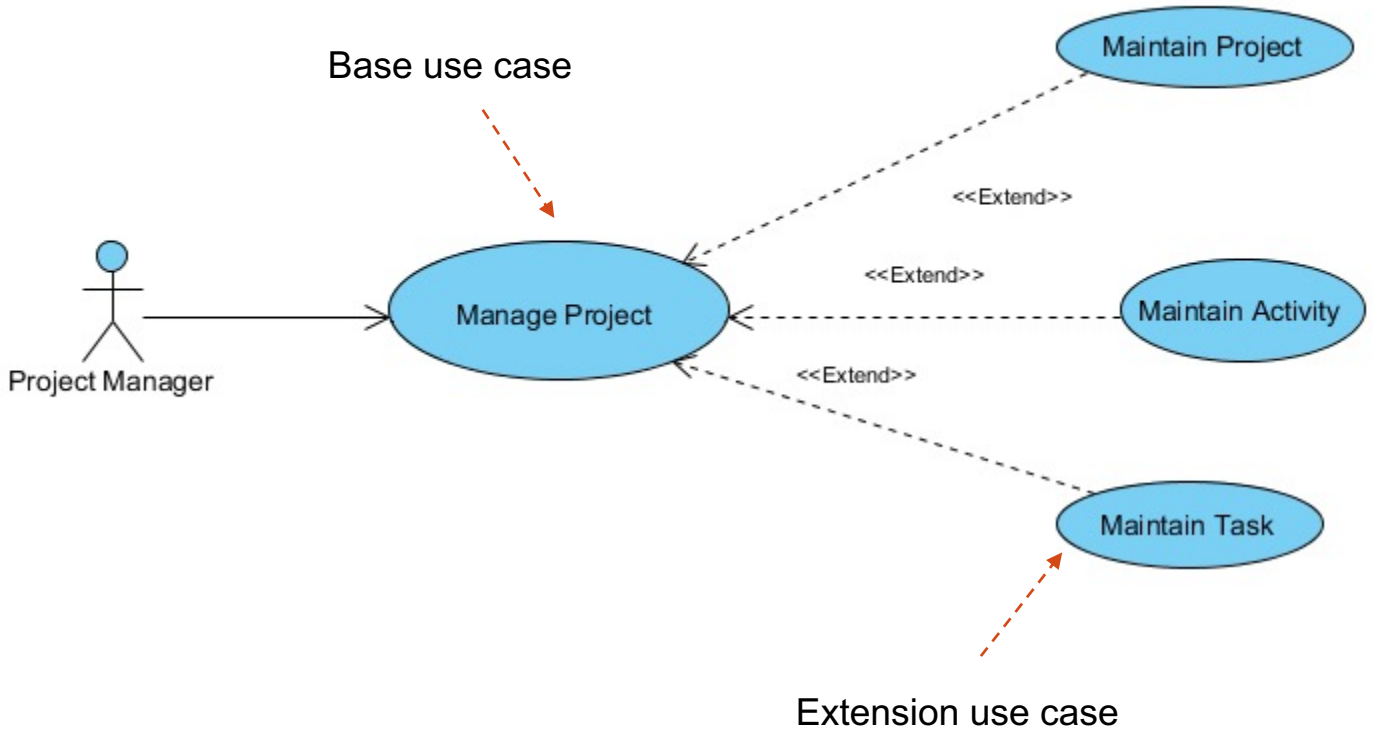
- The importance of reuse in use case diagram by using include dependency:
  - Reduces redundancy between use case descriptions
  - Removes the need for tedious copy-and-paste operations between use case descriptions, since updates are made in only one place instead of every use case.
  - Gives you a good indication at system design time that the implementation of an inclusion use case will need to be a reusable part of your system

Dependency :: Extend



Maintaining the project, its activities, and its tasks are options of managing a project.

Dependency :: Extend

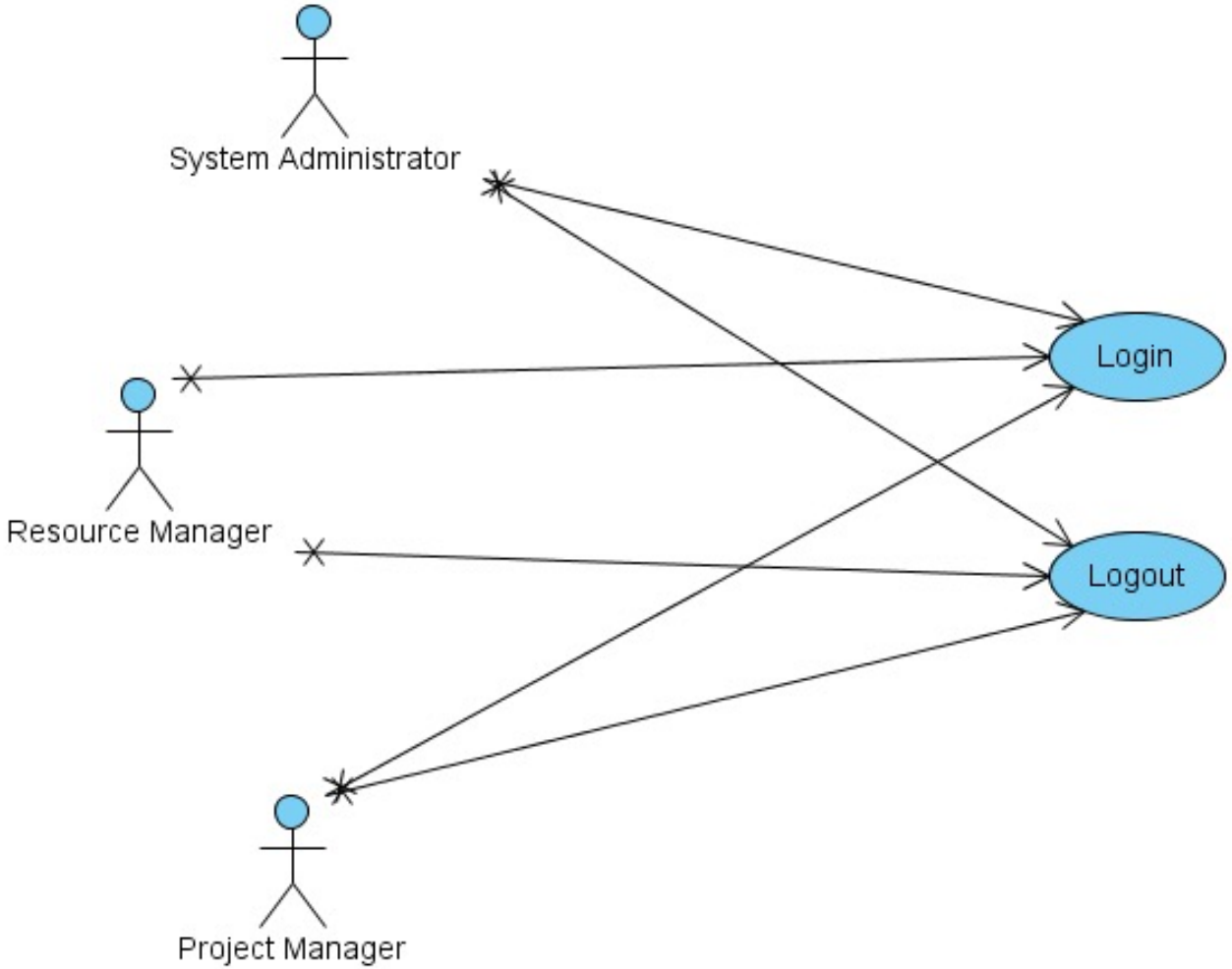


Generalization

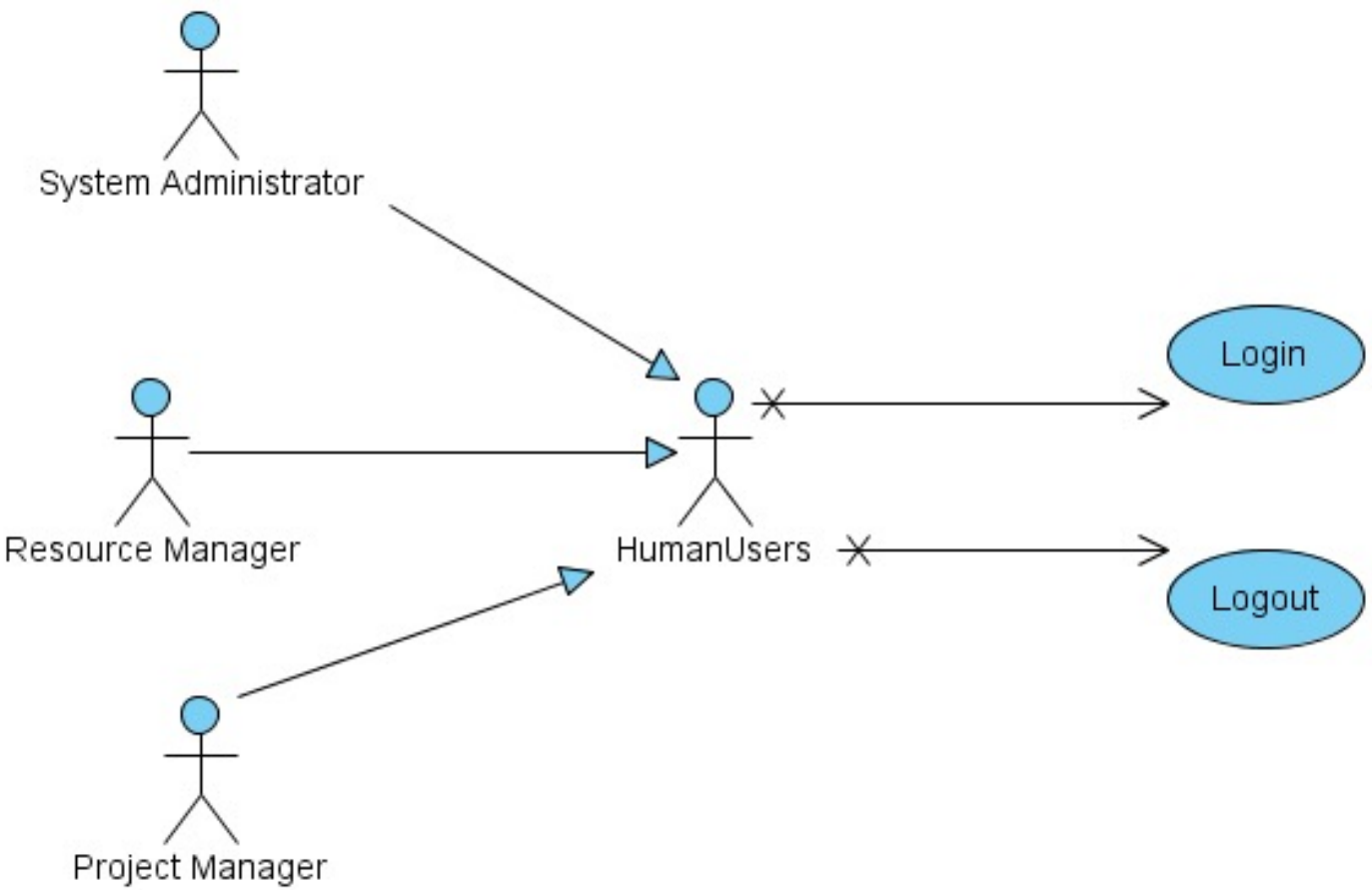
- Actor Generalization
  - Actors may be similar in how they use a system.

- Use case Generalization
  - Use cases may be similar in the functionality provided to users.

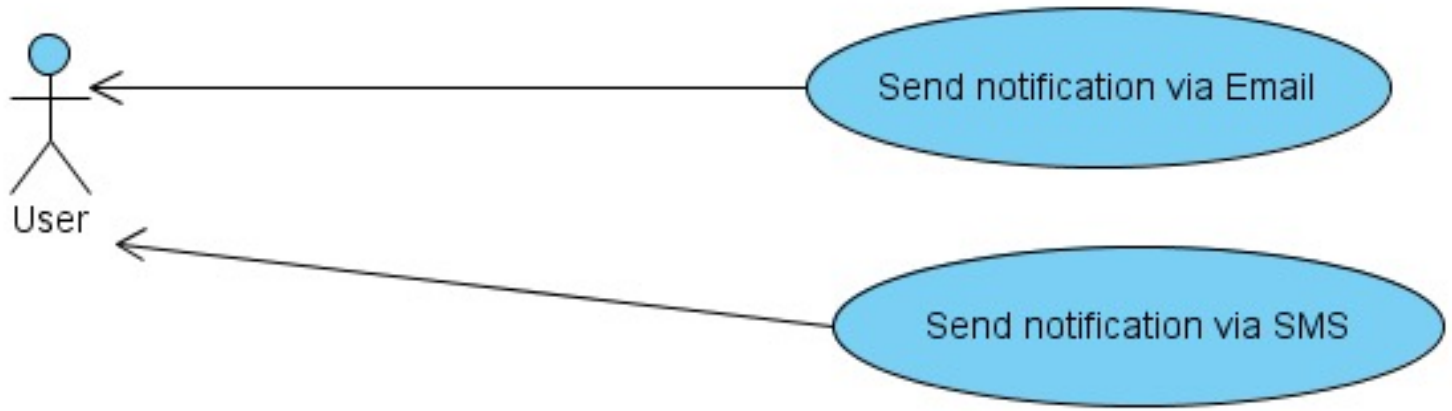
Actor Generalization



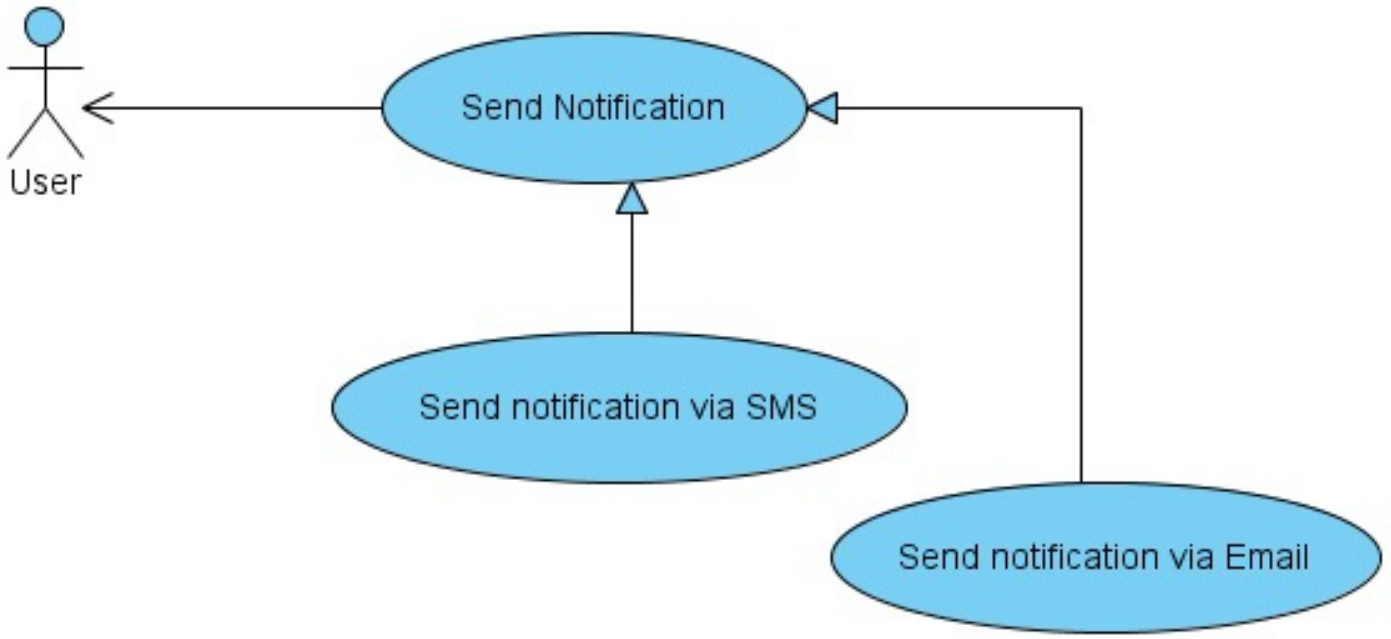
Actor Generalization



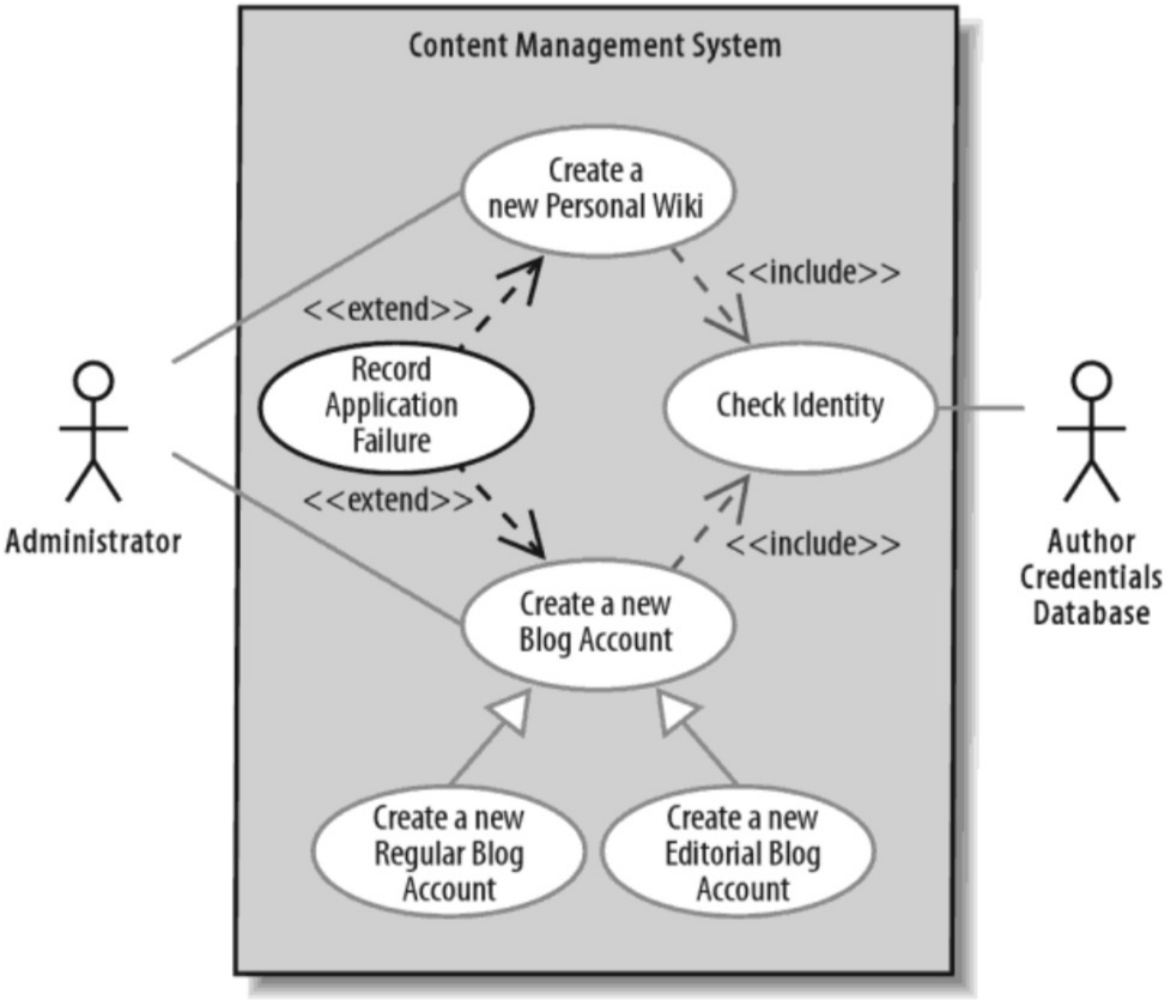
Use Case Generalization



Use Case Generalization



Dependency and Generalization



Summary

Summary

- Use cases are a valuable tool to help understand the functional requirements of a system.
- A first pass at use cases should be made early on. More detailed versions of use cases should be worked just prior to developing that use case.
- It is important to remember that use cases represent an external view of the system. As such, don’t expect any correlations between use cases and the classes inside the system.
- Elements
  - Actors
  - Use Cases
  - Relationships