Software Requirements Specification

Instructor: Yongjie Zheng
February 5, 2020

CS 441: Software Engineering

Requirements Engineering

- Requirements Elicitation and Analysis
  - Identify stakeholders
  - Interview, observation, etc.
  - Requirements classification and prioritization.
- Requirements Specification
  - User requirements
    - Natural languages
  - System requirements
    - Use case model, formal specification, etc.
- Requirements Validation and Verification
  - Review, model checking, etc.

IEEE Std 830–1998: a template for organizing a requirements document

1. *Introduction*
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms and abbreviations
   - 1.4 References
   - 1.5 Overview
2. *Overall description*
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
   - 2.6 Requirements subsets
3. *Specific requirements*

3. *Specific requirements*
   - 3.1 External interface requirements
     - 3.1.1 User interfaces
     - 3.1.2 Hardware interfaces
     - 3.1.3 Software interfaces
     - 3.1.4 Communications interfaces
   - 3.2 Functional requirements
     - 3.2.1 User class 1
       - 3.2.1.1 Functional requirement 1.1
       - 3.2.1.2 Functional requirement 1.2
       - . . .
     - 3.2.2 User class 2
     - . . .
   - 3.3 Performance requirements
   - 3.4 Design constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements

Requirements Specification: Criteria

- **Correct**: The specification can be validated against other documents and by the end-user.
- **Unambiguous**: We must be able to uniquely interpret requirements.
- **Complete**: All significant matters relating to functionality, performance, constraints, and the like, should be documented.
- **Consistent**: Different parts of it should not be in conflict with each other.

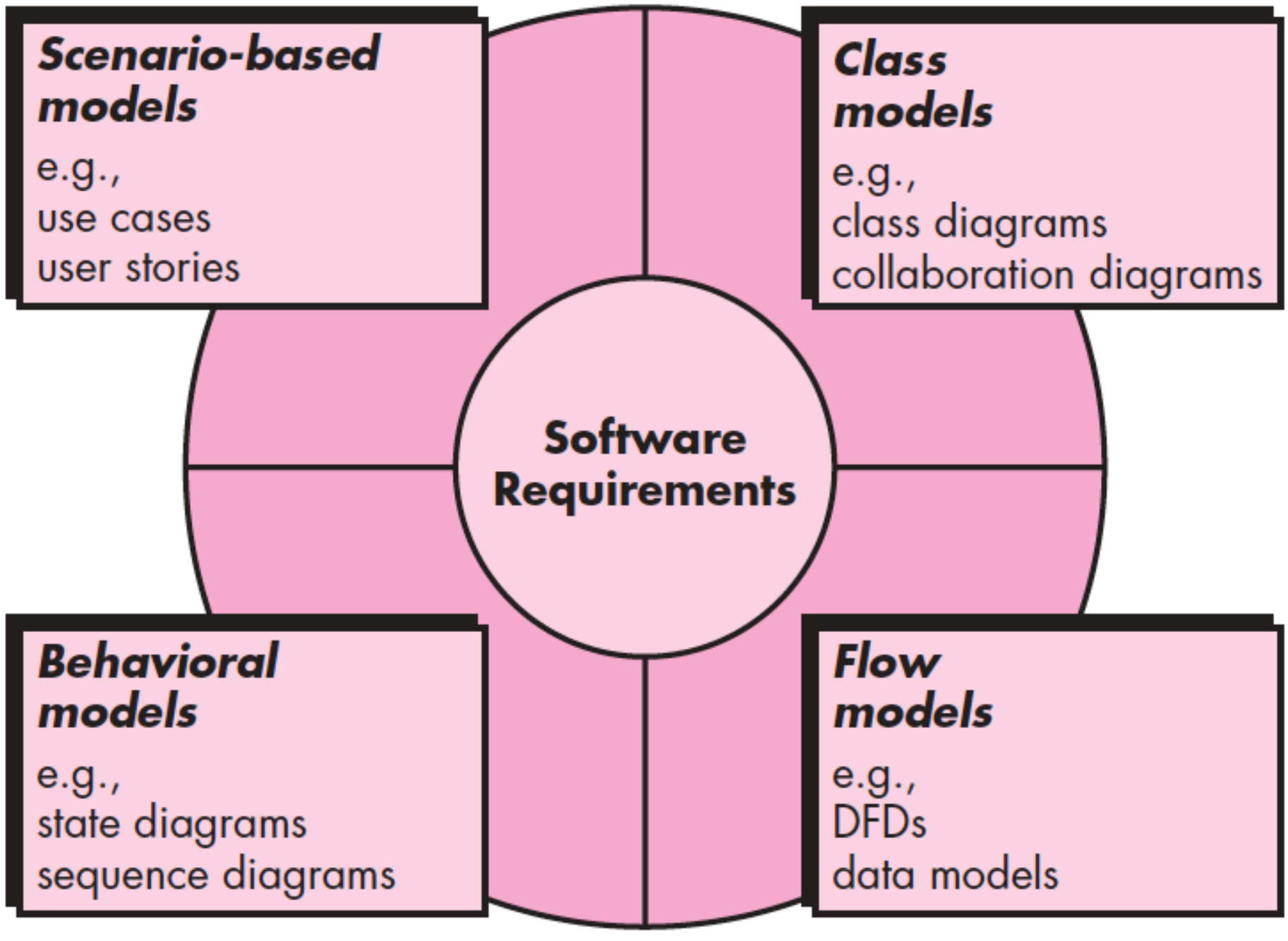
Requirements Specification: Criteria

- **Verifiable**: There must be a finite process to determine whether or not the requirements have been met.
- **Modifiable**: The specification must be easy to change.
- **Traceable**: The origin and rationale of each and every requirement must be traceable. A clear and consistent numbering scheme makes it possible that other documents can uniquely refer to parts of the requirements specification.

Verifiable

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Source Sommerville, 2004



Source: Roger Pressman. 2009. Software Engineering: A Practitioner's Approach (7 ed.). McGraw-Hill, Inc., New York, NY, USA.

UML Use Case Model

- A **use case model** consists of
  - Template-based description of all use cases
  - One or more use case diagrams
- A **use case** consists of
  - A number of positive and negative use case scenarios
- A **use case diagram** consists of
  - An overview of the use cases of a system
  - Relationship between actors and use cases as well as the interrelations among use cases

Use Case

- A use case is a description of system behavior in terms of scenarios illustrating different ways to succeed or fail in attaining one or more goals.
- Scenario, or use case scenario: a sequence of steps describing an interaction between a user and a system.
- Use case: a set of scenarios tied together by **a common user goal**.
  - Main Success Scenario (MSS): all-goes well
  - Extensions: variations on MSS

Use Case Template

- A use case template is often used to guide the documentation of use cases. This is a tabular structure that may consist of the fields of
  - Name
  - Goal
  - Primary actors
  - Scenarios
  - Pre- and post-conditions
- Use cases are text documents
  - The amount of detail depends on the significance of the use case

Use Case

- **Name**: use case has a name describing what is achieved by the interaction with the actor. The name can be a few words in length and it must be unique for every use case.
- **Goal**: the goal of the use case should be described in one or two sentences.
- **Actor(s)**: since a use case has no meaning outside the context of its use by an actor, each actor that participates in the use case must be listed with the use case.

Use Case

- **Scenarios**: the heart of the use case is the event flow, usually a textual description of the interactions between the actor and the system. The flow of events can consist of both MSS, which is the main path through the use case, and extensions, which are executed only under certain circumstances.

- **Pre-conditions**: pre-conditions are those conditions that must be present in order for a use case to start. They usually represent some system state that must be present before the use case can be used.

- **Post-conditions**: post-conditions describe the state of the system after a use case has run its course. They often represent persistent data that is saved by the system as a result of executing the use case.

An Example of Use Case

Buy a product

Main Success Scenario

1. Customer browses and selects items to buy
2. Customer goes to check out
3. Customer fill in shipping information
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 confirming email to customer

Extensions

3a. Customer is a regular customer

3a1. System displays shipping, pricing, and billing information

3a2. Customer may accept or override these defaults, goes to MSS at Step 6

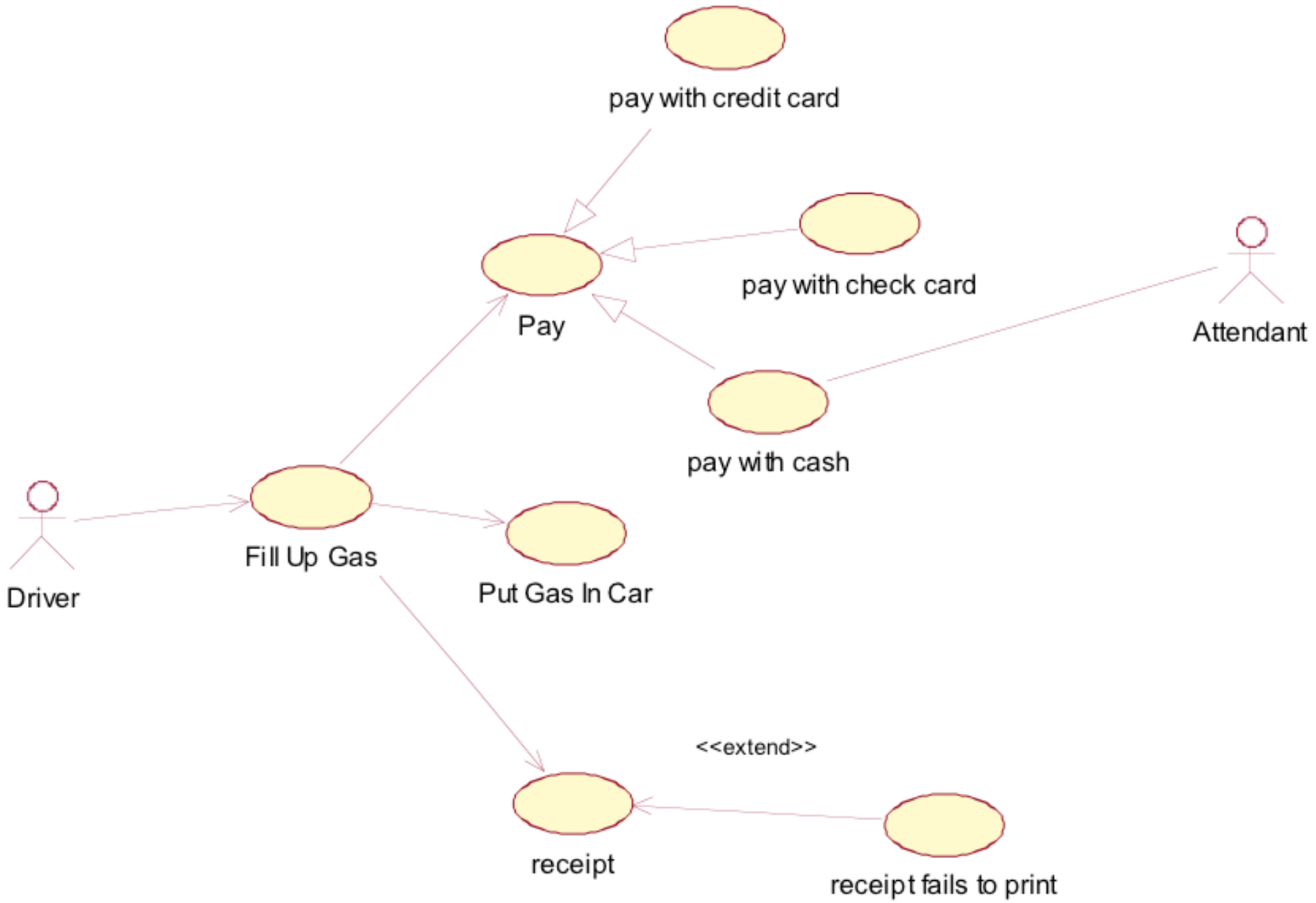
6a. System fails to authorize credit purchase

6a1. Customer may re-enter credit card information or may cancel

Use Case Diagram

- A use case diagram is a graphical notation that provides an overview of the use cases of a system.
- The main purpose of a use case diagram
  - Show all the names of the use cases, like a table of contents
  - Show relationships between actors and use cases
  - Show relationships between use cases

An Example of Use Case Diagram



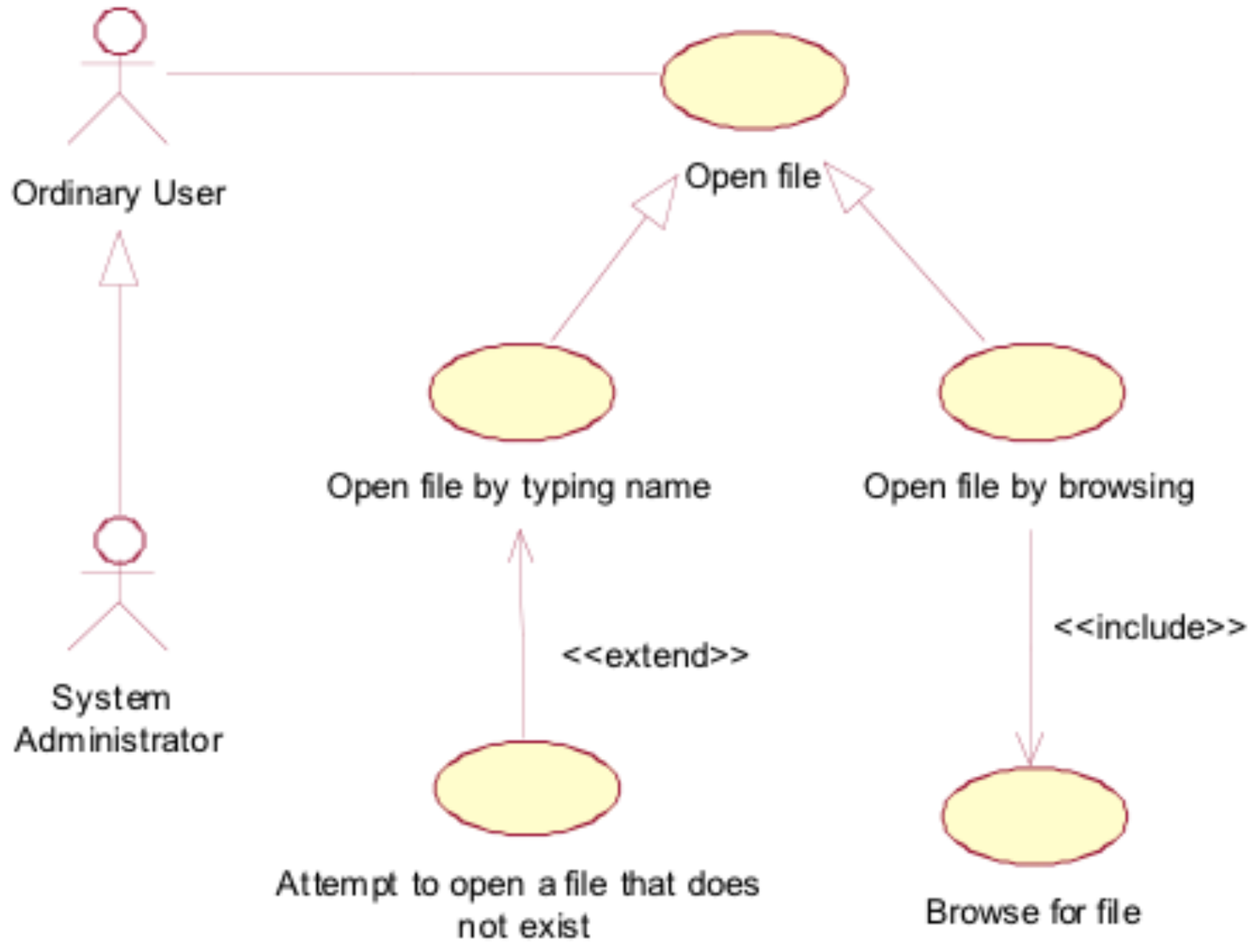
Elements of Use Case Diagrams

- Actor
  - A role that a user plays with respect to the system
  - Represented by a stick figure
- Use Case
  - Represented by an oval
- Relationship
  - Actor - Use Case
    - Actors "carry out" use cases
    - Represented by undirected lines
  - Use Case - Use Case
    - Represented by directed lines with the arrow pointing from subject to object: *Extend*, *Inherit*, *Include*.

Use Case Relationships

- Inclusion
  - One use case is a sub step of another use case, and can be reused by different use cases to avoid repetition.
  - <<include>>
- Generalization
  - One use case is similar to another use case but does a bit more, like inheritance relationship.
  - Solid line + white triangle
- Extension
  - One use case covers exceptions of another use case.
  - <<extend>>

An Example


Ordinary User
Open file
Open file by typing name
Open file by browsing
System Administrator
<<extend>>
<<include>>
Attempt to open a file that does not exist
Browse for file