



# Software Design & Architecture

## Use Cases & Human Values in SE

Pengyu Nie

Acknowledgements: slides adapted from previous versions by Mei Nagappan and Shane McIntosh, which are adapted from previous versions by Zhen Ming Jiang, Ahmed E. Hassan, Reid Holmes. Some slides adapted from William Y. Arms.

# Agenda

- User scenarios
- Use cases
  - use case diagram
  - actors
- Human values in Software Engineering

# User Scenario

- Scenario: a scene that illustrates some interaction with a proposed system
- User scenarios capture the system, as viewed from the outside, e.g., by a user, using specific examples

# Describing a User Scenario

- Typical (minimal) components:
  - a statement of the **purpose** of the scenario
  - the individual **user** or **transaction** that is being followed through the scenario
  - assumptions about **equipment** or software
  - the **steps** of the scenario

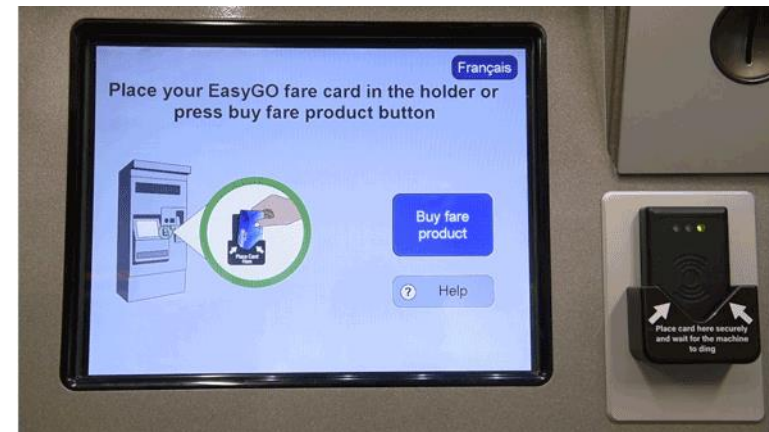
# Developing a Scenario

- Interview (or simulate a interview) with clients/customers:
  - what are the steps to use the system for a given purpose?
  - what questions can be asked?

# Developing a Scenario: Example

- **Purpose:** Scenario that describes the use of ION light rail ticket vending machine to buy a refillable card (EasyGo) by a typical rider
- **Individual:** *[Who is a typical rider?]* Customer A, *[Why getting a refillable card instead of purchasing one-time ticket?]* needs to commute via ION light rail and wants to get a refillable card
- **Equipment:** Ticket vending machine located at some ION light rail stations
- **Scenario:**
  1. Customer A selects “Buy fare product” on the welcome screen  
*[If the system is not at the welcome screen, how to get back to it?]*

*[questions to ask]*



# Developing a Scenario: Example (cont.)

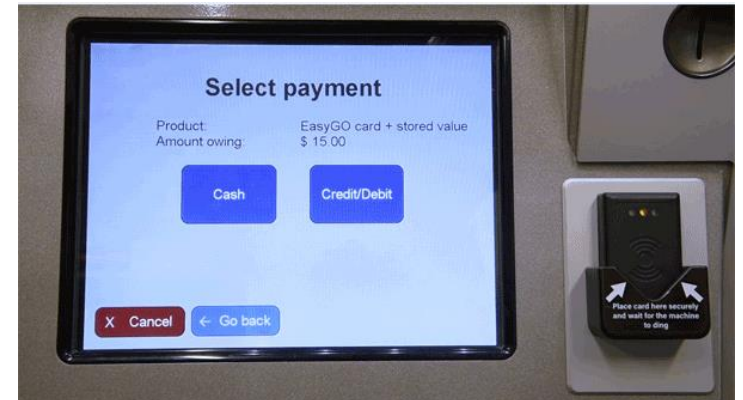
- **Scenario:**
  2. The system displays a list of options  
*[What options / product types should be included?]*
  3. Customer A selects “New EasyGo fare card”
  4. The system displays a list of options  
*[Can customers buy new card and load money in one transaction?]*
  5. Customer A selects “New card + stored value”  
*[If the customer picks the wrong option here (e.g., new card only), is it possible to go back to last step?]*



# Developing a Scenario: Example (cont.)

- **Scenario:**

6. A screen is displayed to allow adding stored values in the unit of \$5, \$10, \$20, or \$50  
*[What is the minimum/maximum load amount?]*
7. Customer A adds \$10 stored value, then selects “Pay”
8. The system displays two options: cash or credit/debit
9. Customer A selects “Credit/Debit”  
*[If the customer picks the wrong option here, is it possible to go back to last step?]*
10. Customer A scans credit card on the payment terminal  
*[Is there a timeout to wait for the payment terminal?]*
11. After waiting a few seconds, a screen displays transaction complete message *[What if transaction failed?]*
12. The EasyGo card and receipt is dispensed into the slot





# Developing a Scenario

- Interview (or simulate a interview) with clients/customers:
  - what are the steps to use the system for a given purpose?
  - what questions can be asked?
- Developing a scenario with clients clarifies many **functional requirements** that must be agreed before a system can be built, e.g., policies, procedures, etc.
- The scenario will often clarify the requirements for the **user interface**
  - but the design of the user interface should not be part of the scenario
- A complex system might need many scenarios

# Analyzing Special Requirements

- User scenarios are very useful for analyzing special requirements
  - **Reversals**: In a financial system, a transaction is credited to the wrong account. What sequence of steps are used to reverse the transaction?
  - **Errors**: A mail order company has several copies of its inventory database. What happens if they become inconsistent?
  - **Malfeasance**: In a voting system, some voters have houses in two cities. What happens if they attempt to vote in both of them?
- “If anything can go wrong, it will”

-- Murphy’s Law
- Create a scenario for everything that can go wrong and how the system is expected to handle it

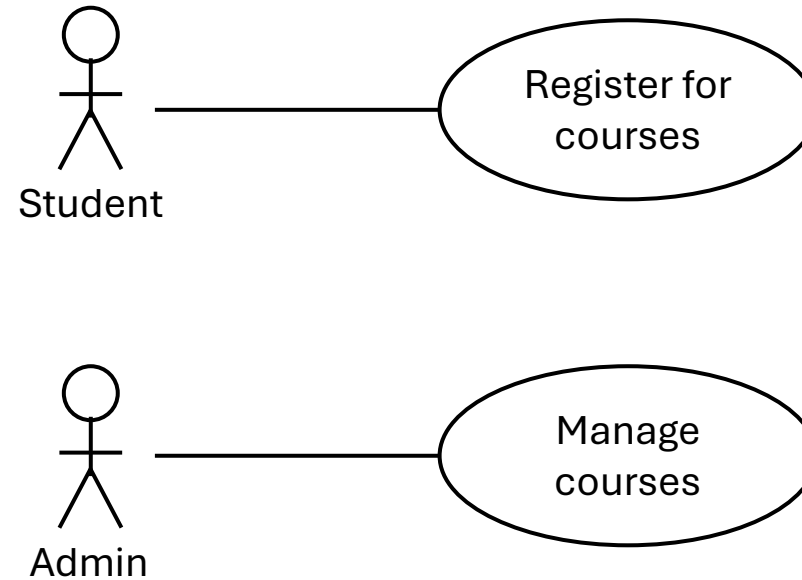
# Modeling Scenarios as Use Cases

- Scenarios are useful in discussing a system with a client, but requirements need to be made more precise before a system is fully understood
- **Use case** is a tool for modeling requirements

# Use Case Diagram

**Actors**

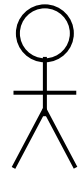
**Use Cases**



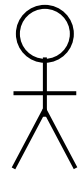
A **use case diagram** shows the relationships between actors and their interactions with a system

# Actors

- An **actor** is a user of a system in a particular **role**
  - One user may have multiple roles  
(e.g., you can be a student for course X and a TA for course Y)
- An actor can be human or an external system
- Avoid generic names (“user”, “client”)









Student




Admin

# Use Cases

- A **use case** is a task that an actor needs to perform with the help of the system
- Tips on what NOT to include as use cases
  - Do not include tasks that are required for technical reasons but have no business value, e.g.,  login,  verify data,  connect to database
  - Do not break down a task into individual steps with little business value if standalone, e.g.,  “manage users” instead of  {“add users”, “delete users”, “modify users”}

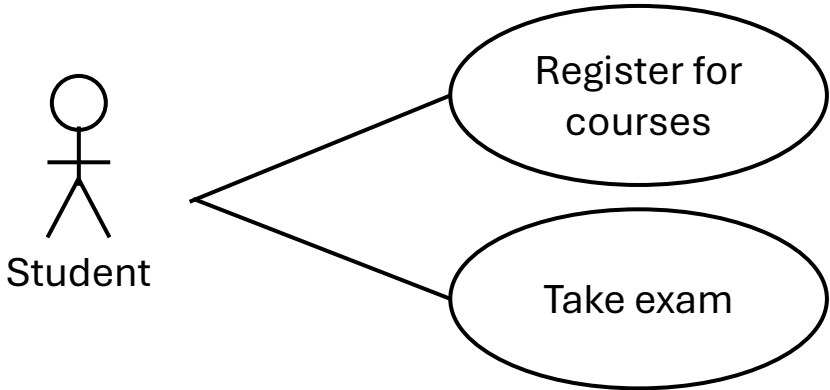


Register for courses

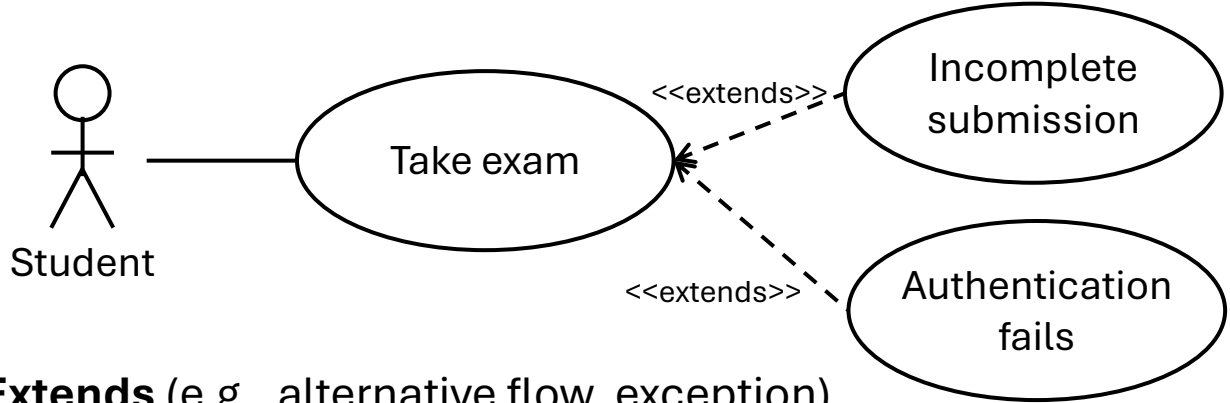


Manage courses

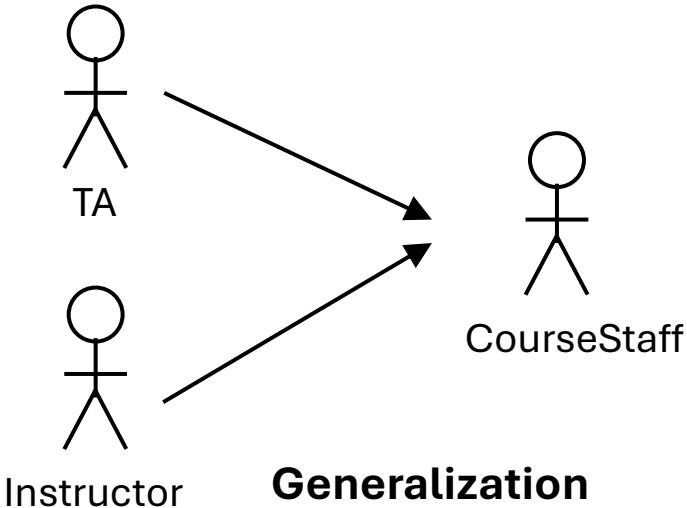
# Relationships



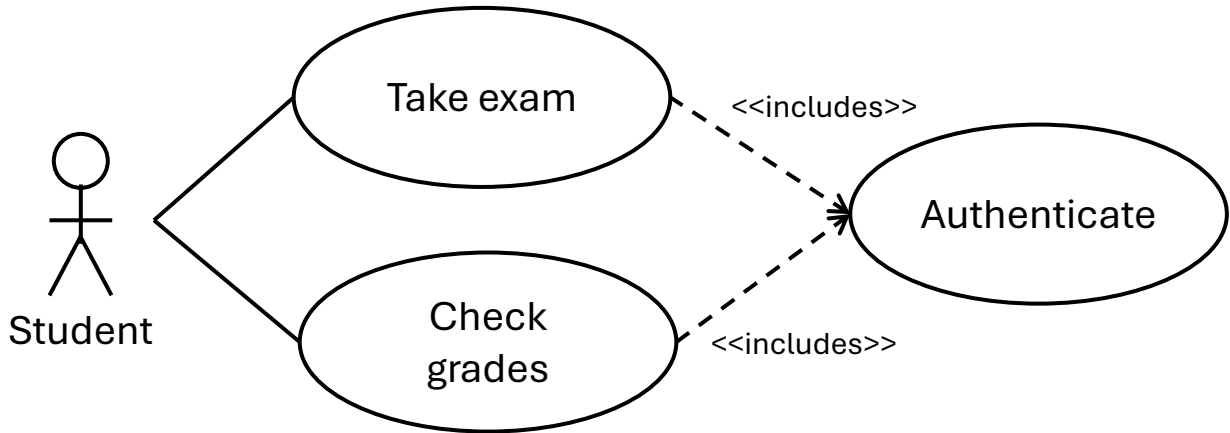
**Association** (bi-directional)



**Extends** (e.g., alternative flow, exception)



**Generalization**



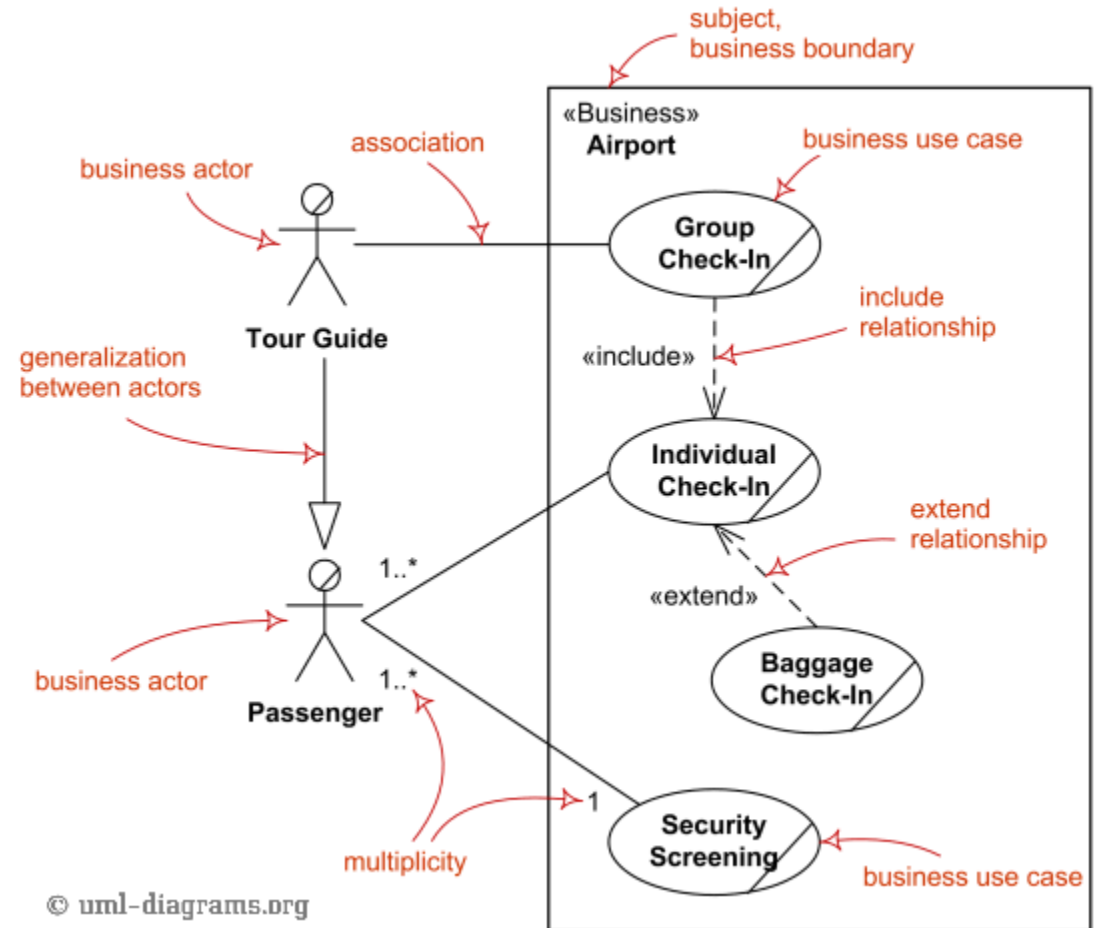
**Includes**

# Use Case Diagrams in UML

- “A use case specifies the behaviour of a system or a part of a system, and is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor.”

-- The UML User Guide [Booch, 1999]

- More about UML (Universal Modeling Language) next week!





# Human Values in Software Engineering

- -> Talk by Jon Whittle at SIGGRAPH Frontiers  
[https://www.youtube.com/watch?v=845fORAf0FQ&t=548s&ab\\_channel=ACMSIGGRAPH](https://www.youtube.com/watch?v=845fORAf0FQ&t=548s&ab_channel=ACMSIGGRAPH)
  - Introduction and motivation: [4:23](#)-14:56
  - What are human values: [18:39](#)-20:53

# Human Values (Summary)

- Most software is built for humans to use
- But software is often fraught with problems of a human nature
  - software may cause intentional or unintentional harm
  - software products are built with a purpose in mind, but that purpose may harm human users by not respecting their values
- Harm can be caused:
  - by a **feature** with (hopefully unintended) consequences
  - due to the **lack of a feature** that a subset of users would deem necessary

# Exercise: Features with Consequences

- What might be some (unintended) consequences of the following features? How they might be avoided?
  - SelfieDrone: A drone system that can follow a user, take amazing selfies, and can even remove photobombers from your IG-worthy pictures
  - Messaging: An instant messaging platform like Discord or Slack, which allows any user to message anyone else on a platform
  - ResumeFilter: A system that automatically screens resumes to select qualified interviewees
  - InfiniteScrolling: The “infinite scrolling” feature built into most social media and shopping apps

# Agenda (recap)

- User scenarios
- Use cases
  - use case diagram
  - actors
- Human values in Software Engineering

*These will also be a part of P2: Project Proposal*

**P0: Team Formation due this Friday!**