# Requirements Elicitation and Use Cases

CSCE 740 - Lecture 6 - 09/06/2016

# Virtual Elicitation Session

- Will open on Moodle (”Dropbox”) message board after class.
- I am a pretend customer - you ask questions to elicit requirements for the project.
- All elicitation questions **must** be posted there, so that answers are available for everybody.
- You can continue asking questions for the duration of the project (but don’t wait).

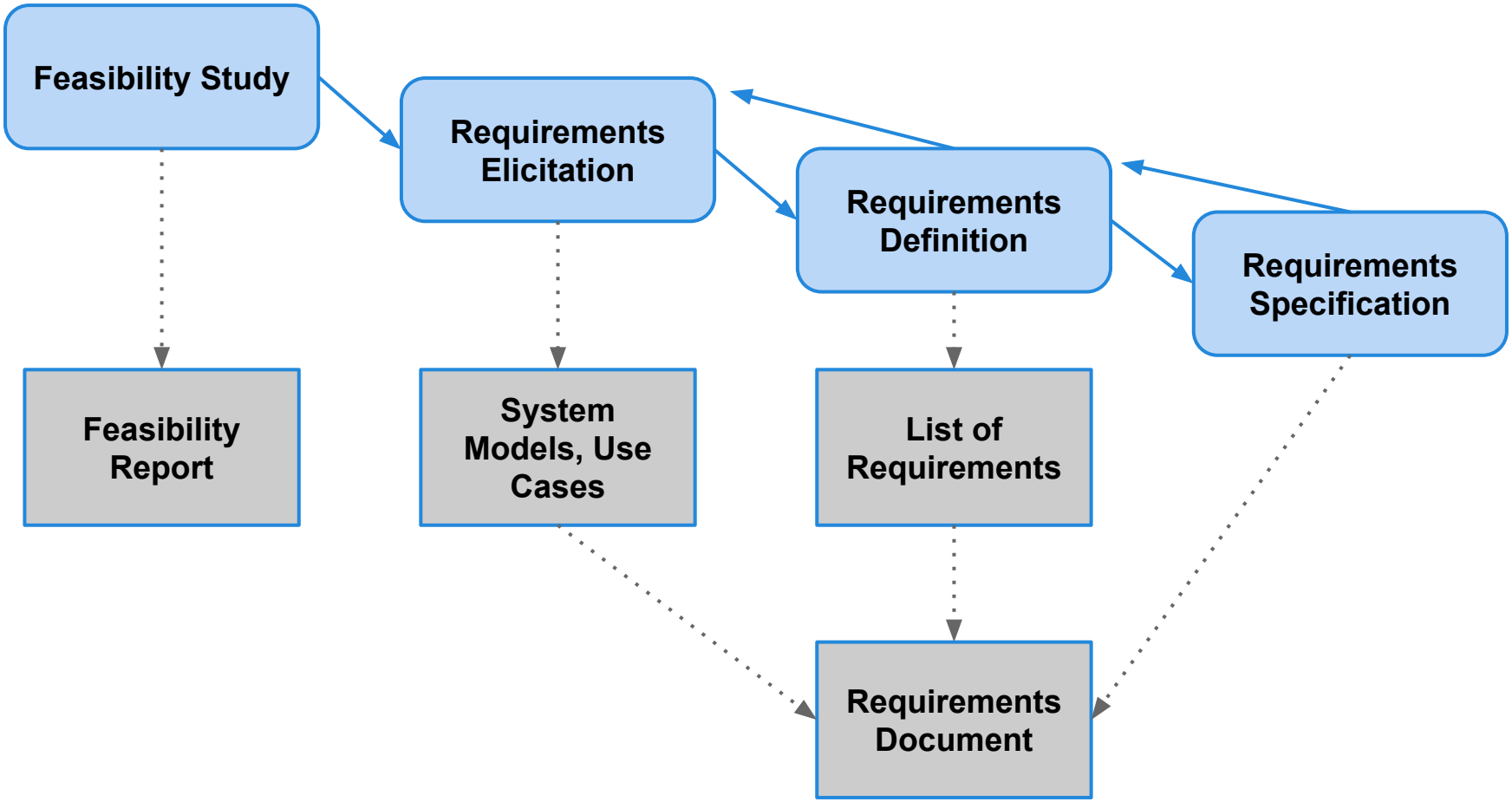
# Today's Goals

- Understand the concept of stakeholders.
- Discuss techniques for getting the information needed to develop a system.
- Discuss use cases and their role in brainstorming and explaining requirements.

# Requirements Elicitation

- The process of working with customers to learn about the application domain, the services that the system should provide, and the system's operational constraints.
- Involves all stakeholders:
  - end-users, managers, maintenance team, domain experts, trade unions, lawyers, etc…

# The Requirements Engineering Process

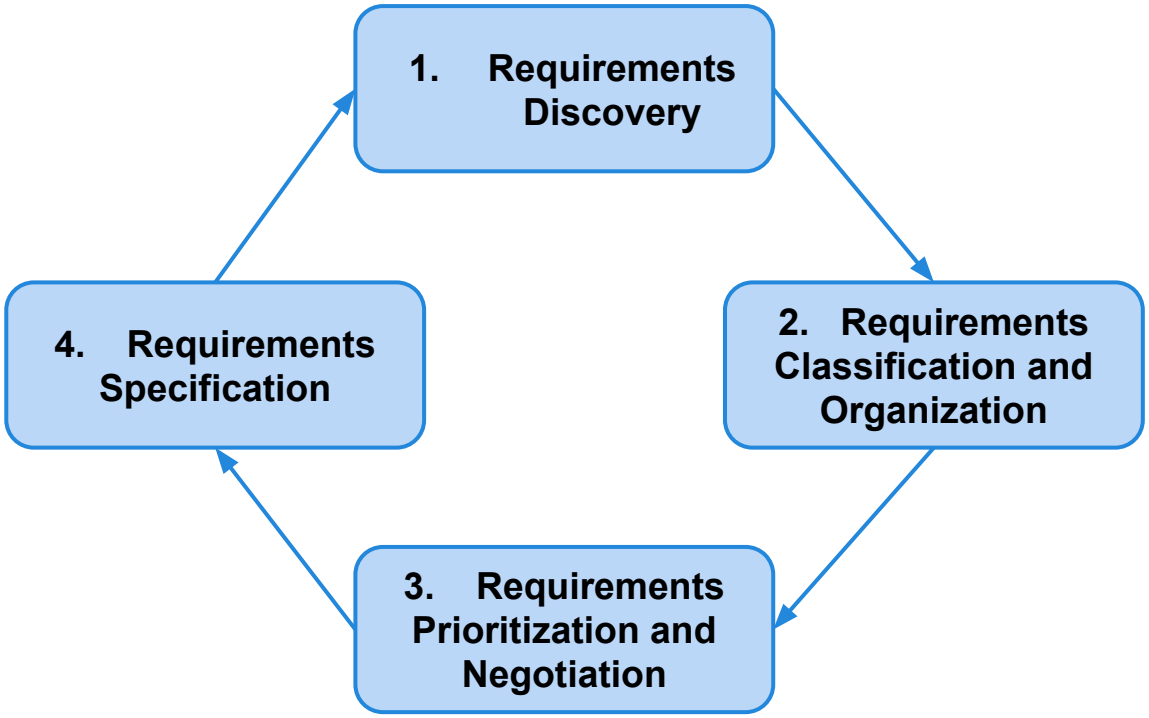


# Requirements Elicitation

# Why is Elicitation so $$%$% hard?

- Stakeholders don't know what they want from a computer system except in the most general terms.
- Stakeholders express requirements with implicit knowledge of their domain.
- Different stakeholders have different requirements.
- Economic and business environment is dynamic during elicitation.

# Requirements Elicitation Process



# Stakeholders Must Work Too

# Interview the Stakeholder

- Make sure you have the right answer.
- Make sure this is the official answer.
  - If an answer is unclear, keep asking questions.
  - If somebody else tells you a different answer, politely ask for clarification.
- Find out what they are willing to pay for each function (helps in prioritization).
  - If unwilling/unable to place a $ amount, ask how important it is for their daily work.

**These two often conflict!**

# Interview the Stakeholder

- Try not to alienate the stakeholder:
  - Avoid “We thought you knew that.” and “We always do it that way.”
- Hundreds of techniques
  - Most important: do your homework - research a problem before the interview.
  - Be polite, but firm - keep asking until you feel you could deliver a working function.
  - Follow up after writing requirements down.

# Viewpoint-Oriented Analysis

- Stakeholders represent different ways of looking at a problem (different viewpoints).
- Looking at problems from multiple viewpoints tends to lead to solved problems.
- There is no single correct way to analyze system requirements, collect the different viewpoints and work out the system that best matches all of them.

# Types of Viewpoint

- Receivers of services
  - People or systems that receive services from your system.
- Data sources or sinks
  - What kind of data is produced or consumed by the stakeholders and system?
- Experts in the domain
  - Tend to notice details that novices will miss.

# Banking Viewpoints

**What are some of the stakeholders to consider for a bank account management system?**

- Bank Teller
- Account Holder
- Merchant

**What are some of the things these stakeholders would want to accomplish?**

# What is a Use Case?

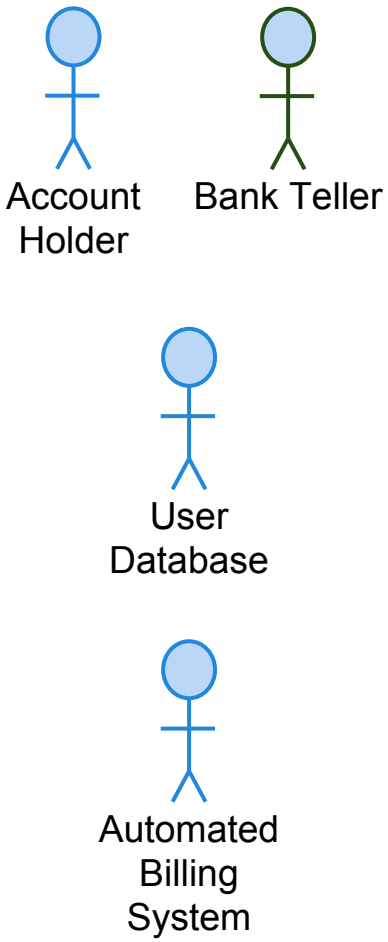
A **use case** captures some visible function of the system.

- A use case is a **goal** that an actor can accomplish using a system, through a series of **user interactions.**
  - Transfer funds.
  - Query balance.
- This may be a large or small function.
  - Depends on the chosen level of detail.
  - Withdraw Funds vs Validate PIN

# Use Cases

- Accompanied by a **use case description** detailing the user interactions required to accomplish the use case.
- Acquired through interviews with stakeholders and viewpoint analysis.
- Useful for eliciting and refining requirements.

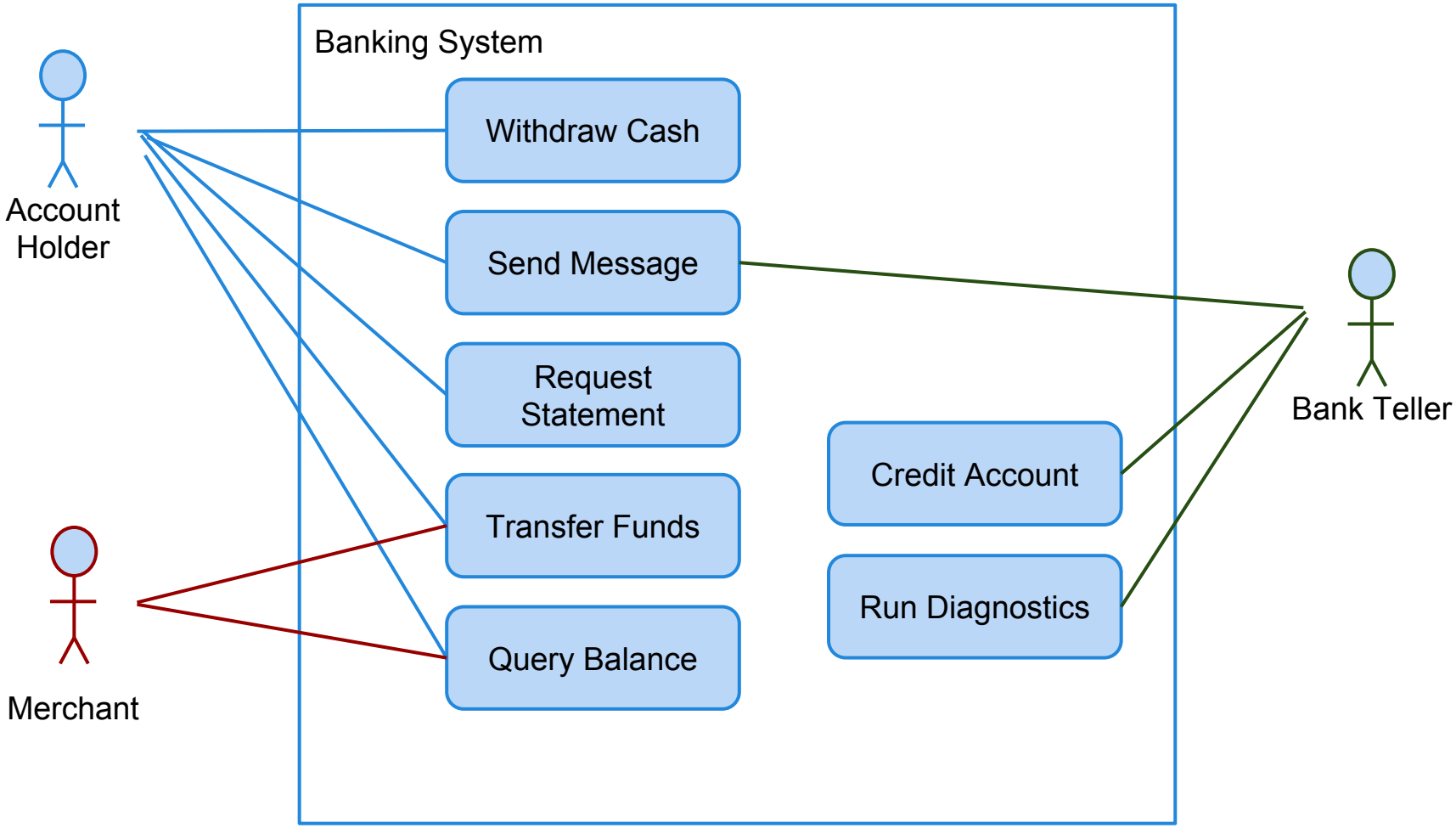
# What is an Actor?




An **actor** is a role a user plays with respect to the system.

- Actors carry out use cases. An actor can perform many use cases. A use case can involve multiple actors.
- A single user can be multiple actors, depending on how they use a system.
- Actors do not need to be human - can be an external system (hardware or software) that interacts with the system being built.

# Online Banking Use Case Diagram



# Use Cases vs User Interactions

- When formatting a document:
  - Bold a line of text.
  - Change bold text to italic.
  - Copy text from one document to the next.
- versus
  - Format a document.
  - Ensure consistent formatting in similar documents.
- The latter are goals (use cases - what they want to achieve), the former are interactions (what they do to achieve the goal).

# User Goals vs User Interactions

- To understand what the system should do, capture the user goals (the use cases).
- To understand how the user will achieve the goals, capture the sequences of user interactions.
- Start with the use case, and refine them into a series of user interactions (**the description**).

# Use Case Descriptions

- Sometimes, a goal can be achieved through multiple interaction sequences.
  - A scenario is **one possible** sequence of user interactions to get to a goal..
  - If a user can fail to achieve a goal, that is another scenario - called an **exception path**.
- A use case description should include all scenarios that can occur when attempting to achieve that use case.

# Use Case Descriptions

- Commonly, a use-case description has a common all-goes-well case and many alternative paths that encompass failure scenarios.
- Coming up with multiple scenarios is useful in brainstorming requirement and getting feedback from stakeholders.

# Use Case: Withdraw Cash

**Actor: Customer**

1. The customer inserts the card into the ATM.
2. The ATM accepts the card and asks the user for the PIN.
3. If the PIN is correct, the ATM asks the user for an account choice.
4. The user enters the account.
5. The ATM asks the user for a cash amount.
6. The user enters the amount.
7. The ATM asks the user to verify the account.
8. The user verifies the account.
9. If there are sufficient funds in the account, the money is dispensed and the amount is withdrawn from the account.

# Use Case Templates

- **Identifier:** Unique ID number
- **Iteration:** Version number
- **Summary:** User goal being fulfilled
- **Actors:** What users/databases/external systems are involved?
- **Basic Course of Events:** Sequence of user interactions.
- **Alternative Paths:** Alternate sequences of events that stem off from certain points.
- **Exception Paths:** Error sequences that stem off from certain points.
- **Extension Points:** Use-cases that resume from the end of this use-case.
- **Trigger:** Rationale, what causes this interaction sequence to begin.
- **Assumptions:** Constraints assumed on this use-case.
- **Precondition:** Conditions that must hold for this use-case to take place, may list other use-cases that need to be completed first.
- **Postcondition:** Side-effects of this use-case.
- **Author:** Who wrote this use-case.
- **Date:** When was this last modified?

# Withdraw Cash (In Sample Template)

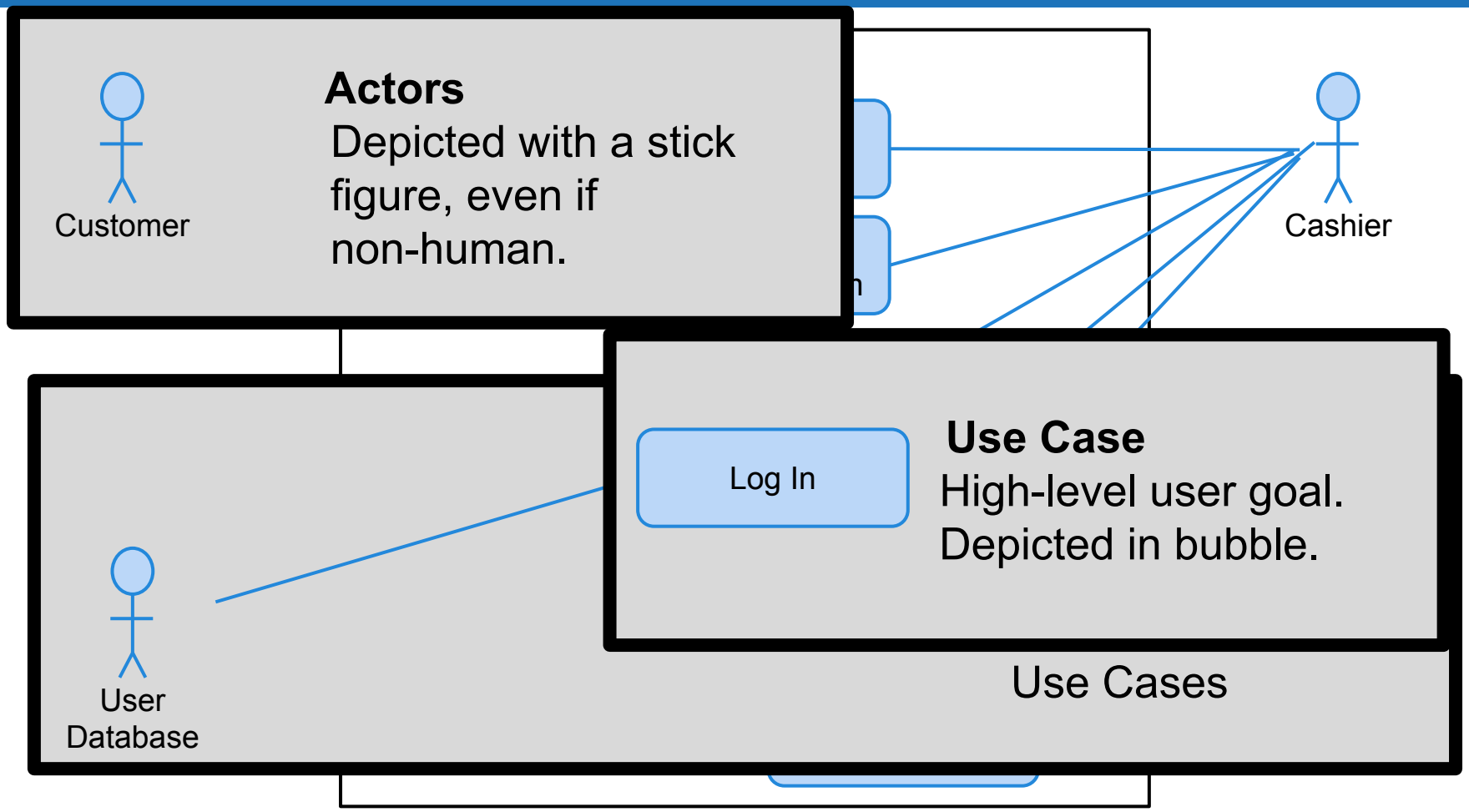
- **Summary:** The customer requests cash and the ATM dispenses the cash.
- **Basic Course of Events:**
  - 1. Completion of use case *Validate PIN*.
  - 2. The customer selects the withdrawal menu option.
  - 3. The ATM asks the customer for the account from which to withdraw the cash.
  - …
  - 9. If there are sufficient funds, the cash is dispensed and the amount is withdrawn from the account.
  - 10. Complete use case *Complete Transaction*.
- **Alternative Paths:** In steps 4, 6, and 8, the customer can cancel the transaction and go directly to step 10. If the customer does not confirm the account in step 8, proceed directly to step 10.
- **Exception Paths:** In step 9, if there are not sufficient funds, then an error message is displayed and execution proceeds to step 10.
- **Precondition:** The Validate PIN use case completed successfully.
- **Postcondition:** The cash is dispatched and the amount has been withdrawn from the selected account.

# Grocery Store System

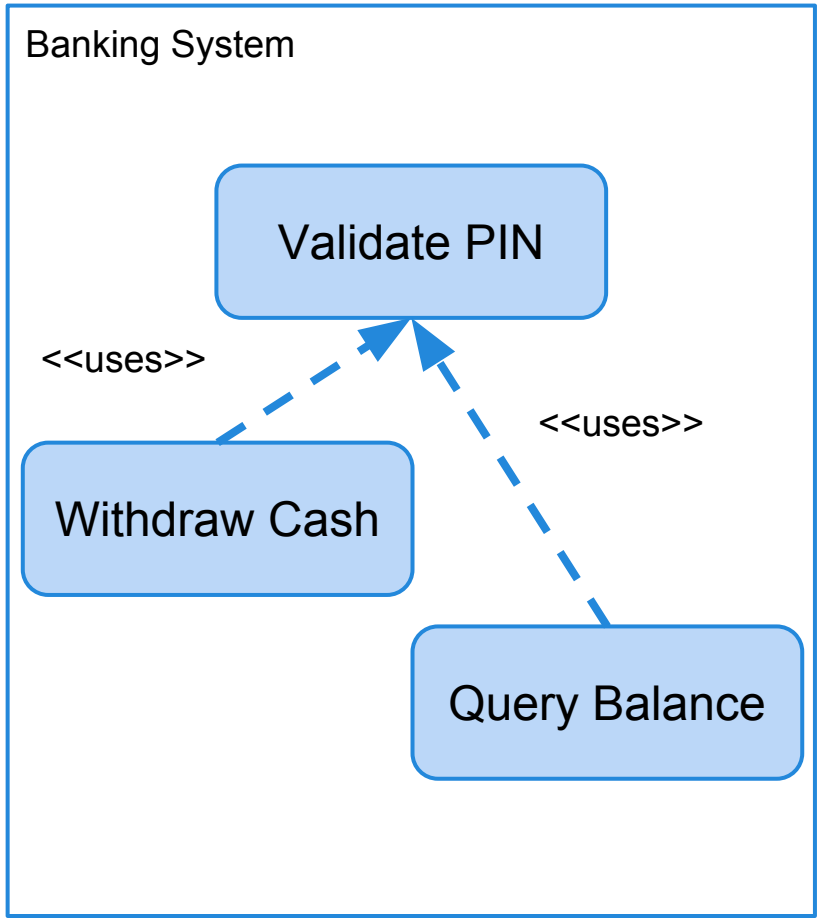
You are building a store management system. Customers enter the store and select vegetables. When they are finished, the bring their items to a cashier in order to purchase them. Cashiers, while logged in to their terminal, can also refund purchases and update the store inventory. A manager monitors the inventory and orders additional stock if needed.

**What are the actors and use cases of this system?**

# Grocery Store System Diagram



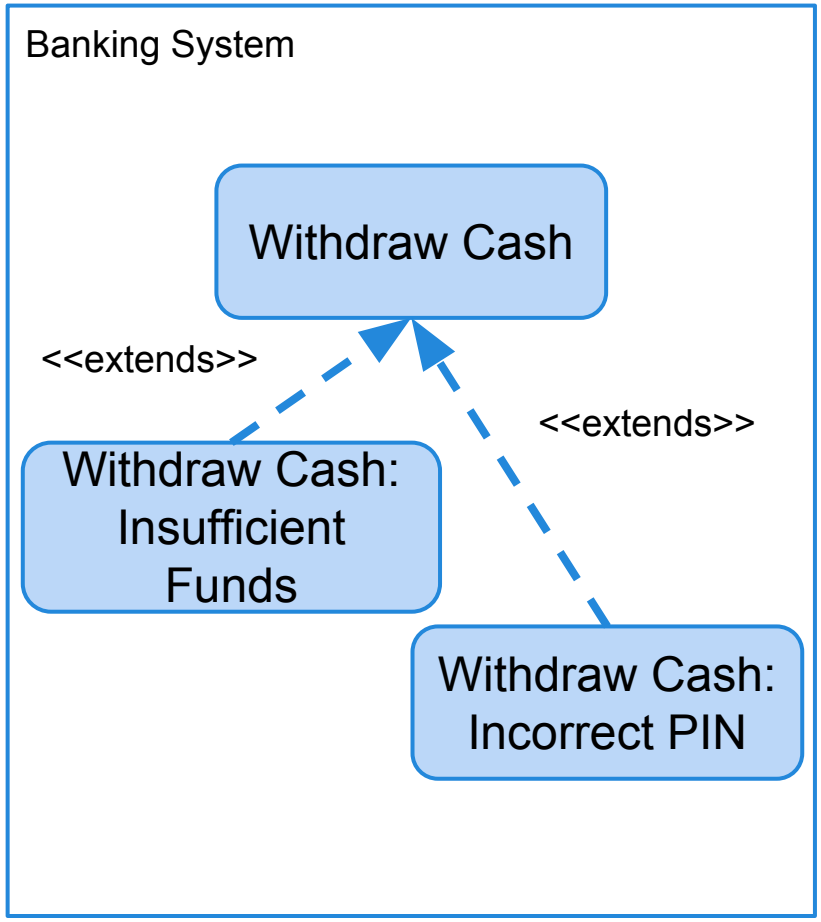
# Use Case Relationships: Uses




## Uses

- You have a piece of behavior that is similar across many use-cases.
- Break this out as a separate use-case and let the others “use” it.
- Avoids repetition in written use-cases.
  - Step 1: Complete “Validate PIN” use-case.
  - Step 2: Select account.
  - ...

# Use Case Relationships: Extends

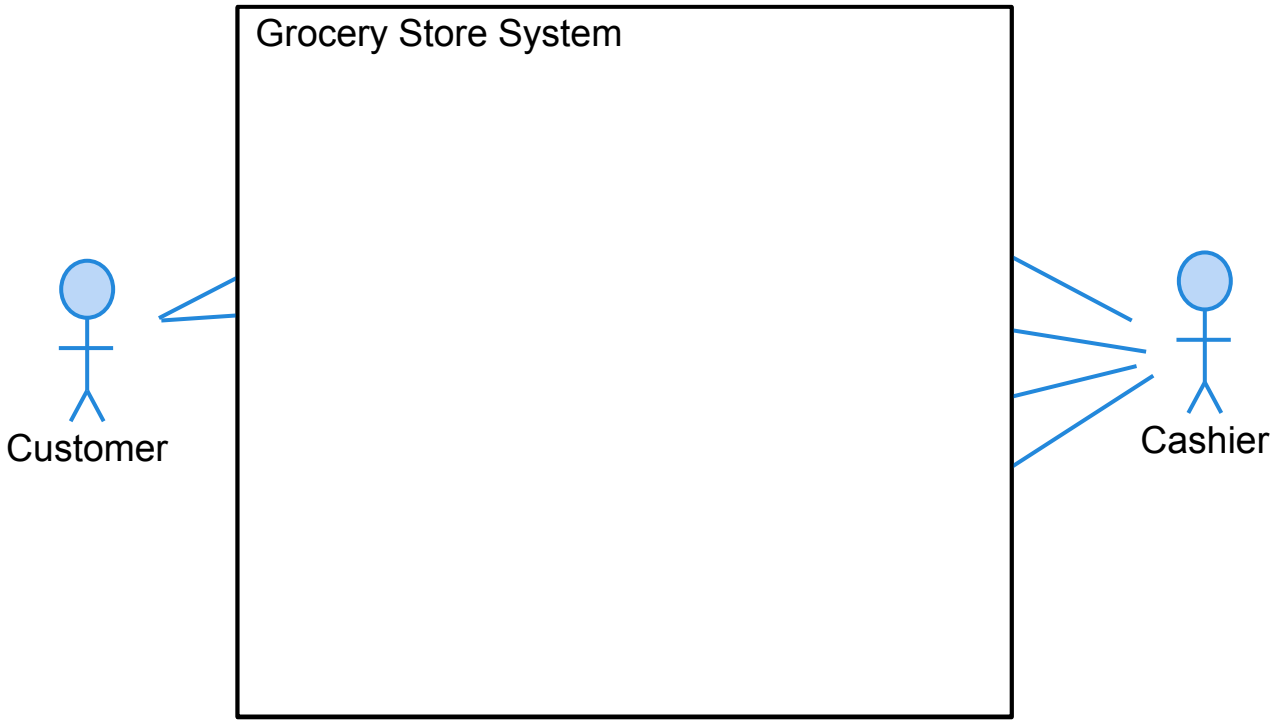



## Extends

- A use-case is similar to another one, but does a little bit more or takes an alternate path.
- Put the normal behavior in one use-case and the exceptional behavior somewhere else:
  - Capture the normal behavior.
  - Try to figure out what went wrong in each step.
  - Capture the exceptional cases in separate use-cases
- Allows for easier-to-understand written use-cases.

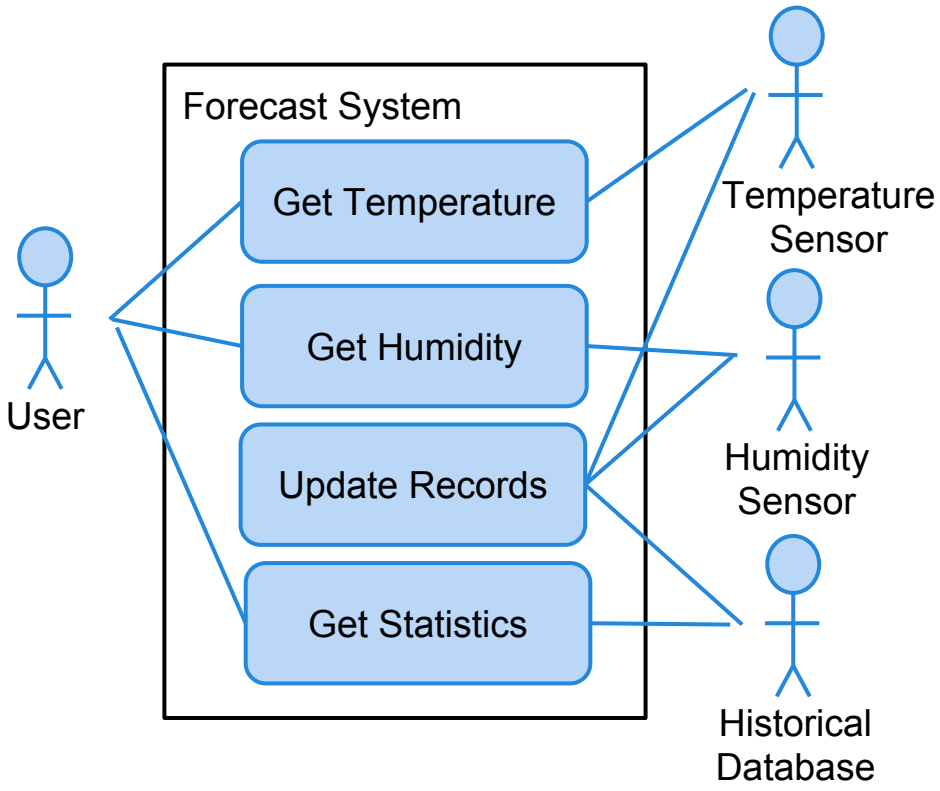
# Setting the System Boundary

The system boundary will affect your actors and use-cases.



# System Boundary - Weather Forecast

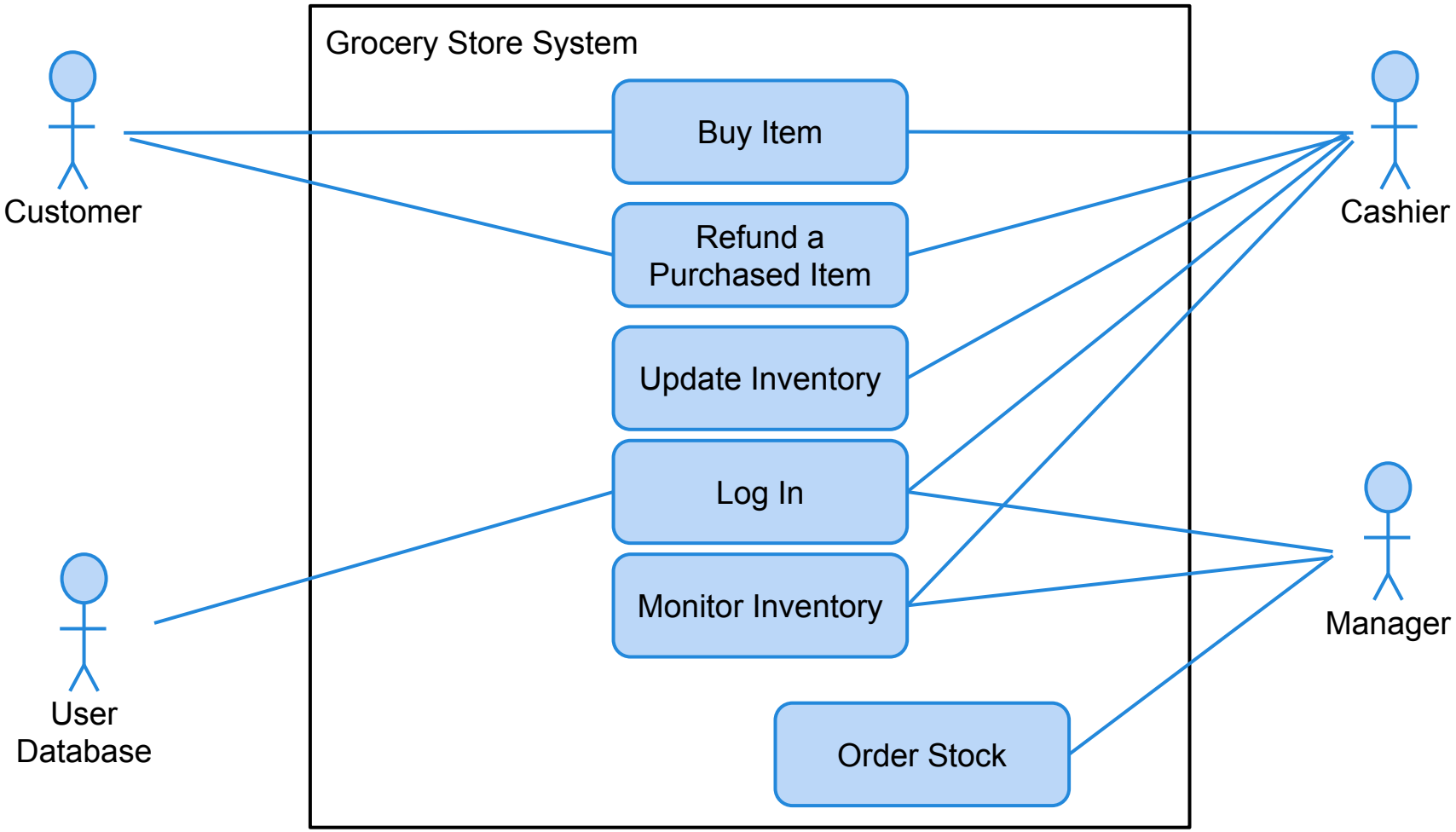
The system boundary will affect your actors and use-cases.




**Option 3: Computer+Sensors Boundary**

- System is everything you get with purchase. Sensors are internal now.
- Eliminates the Update Records use-case.

# Grocery Store System Diagram



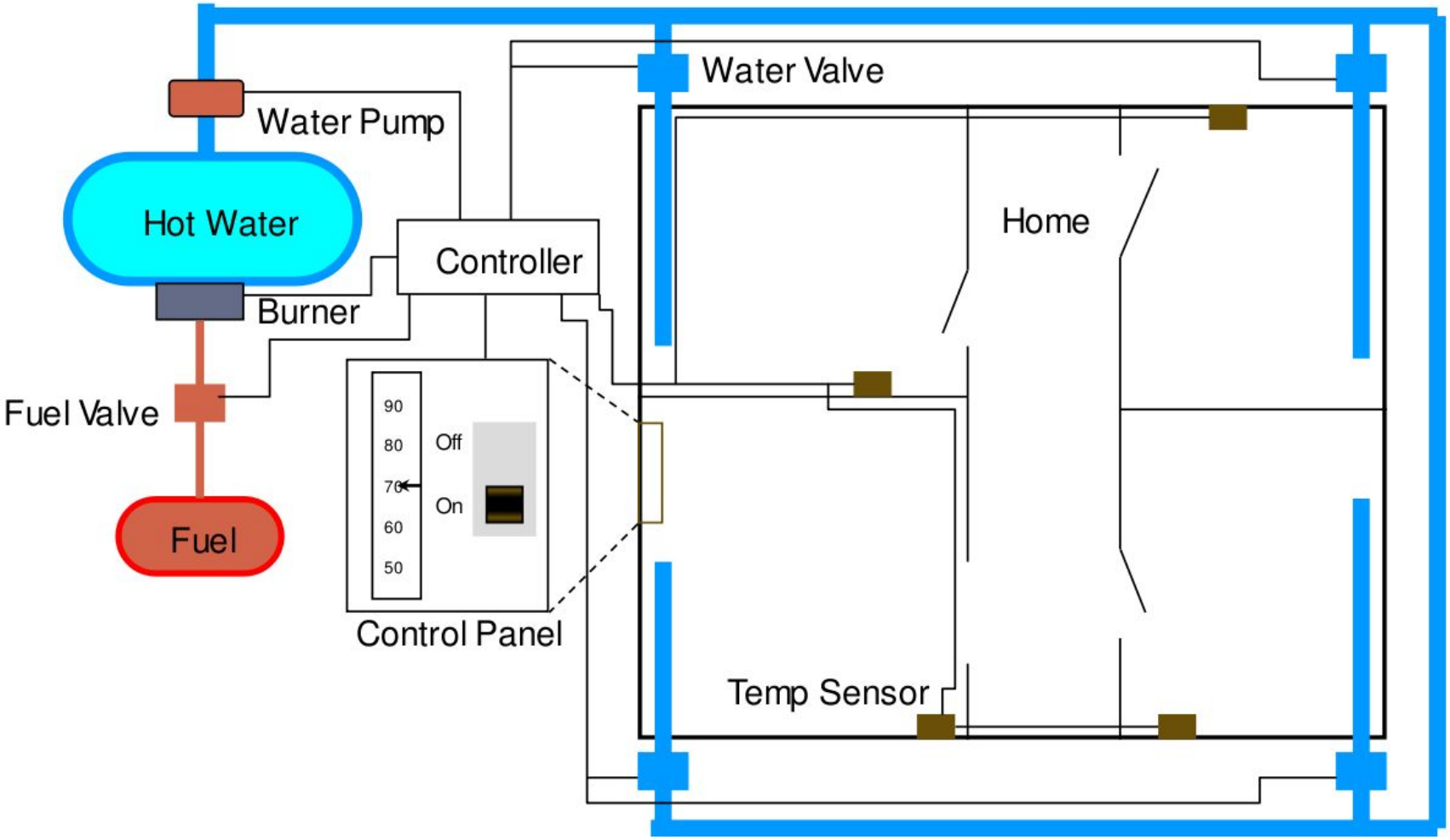
# Grocery Store System Scenario

- **Scenario: Buy Item**
- **Actors:**
  - Customer (initiator), Cashier
- **Description:**
  - The Customer arrives at the checkout with items to purchase.
  - For each item, the Cashier records the item and the software updates the payment total.
  - The Cashier accepts payment in either cash or credit card form and records payment information in the software.
  - If payment is successful, the software will print a receipt and the Customer collects the items and leaves the store.

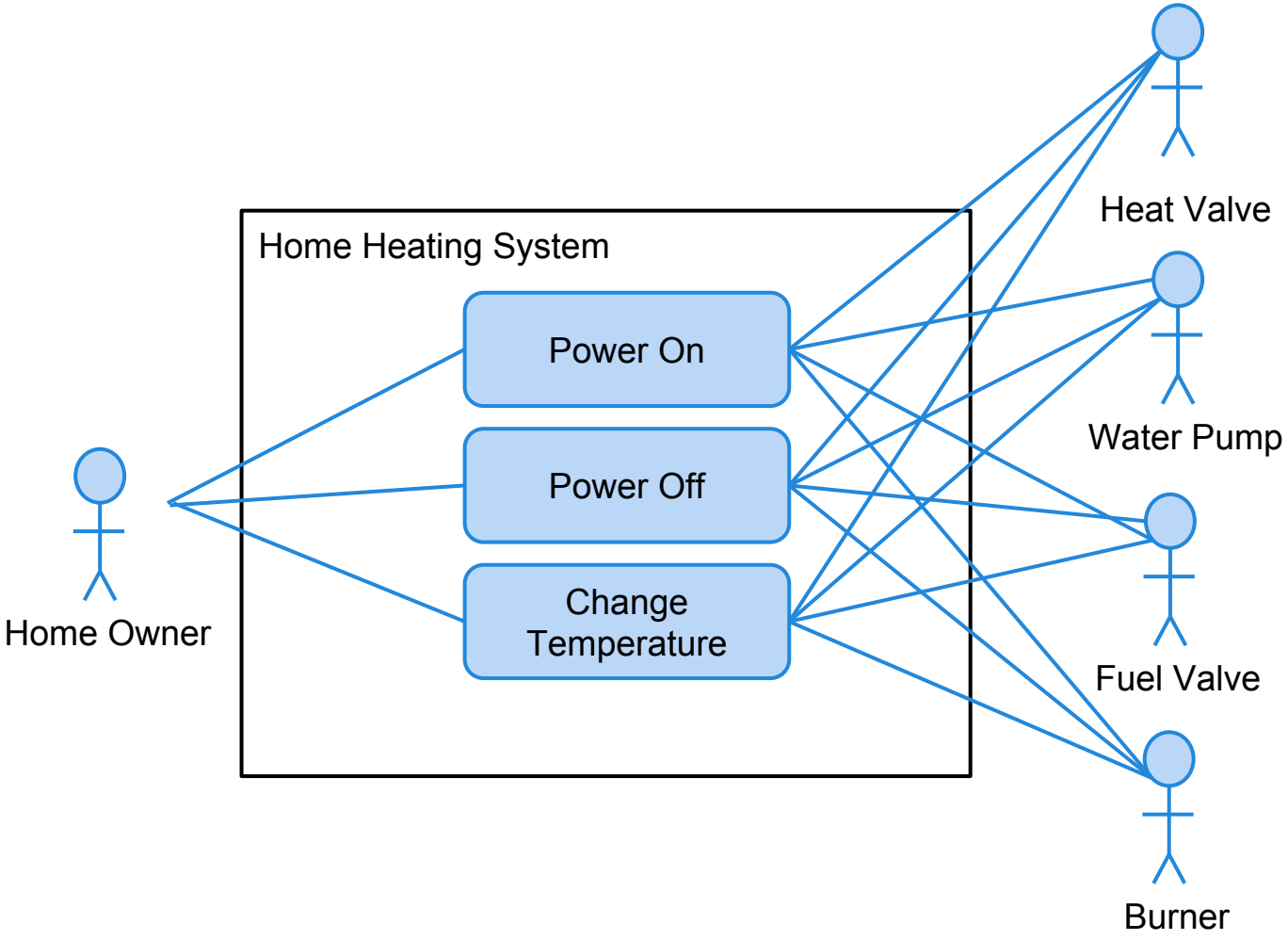
# Grocery Store System Use Case

- **Use-Case: Buy Item**
- **Actors:** Customer (initiator), Cashier
- **Description:**
  - The Customer arrives at the checkout with items to purchase.
  - For each item:
    - the Cashier records the item,
    - completes use-case “Update Inventory”,
    - and the software updates the payment total.
  - The Cashier accepts payment in either cash or credit card form and records payment information in the software.
  - If payment is successful, the software will print a receipt and the Customer collects the items and leaves the store.
- **Exception Paths:** If credit card payment is denied, then an error message will be displayed and the customer will not be allowed to leave with the items.
- **Preconditions:** Cashier must have completed use-case “Log In”

# The Home Heating System



# Home Heating System Diagram



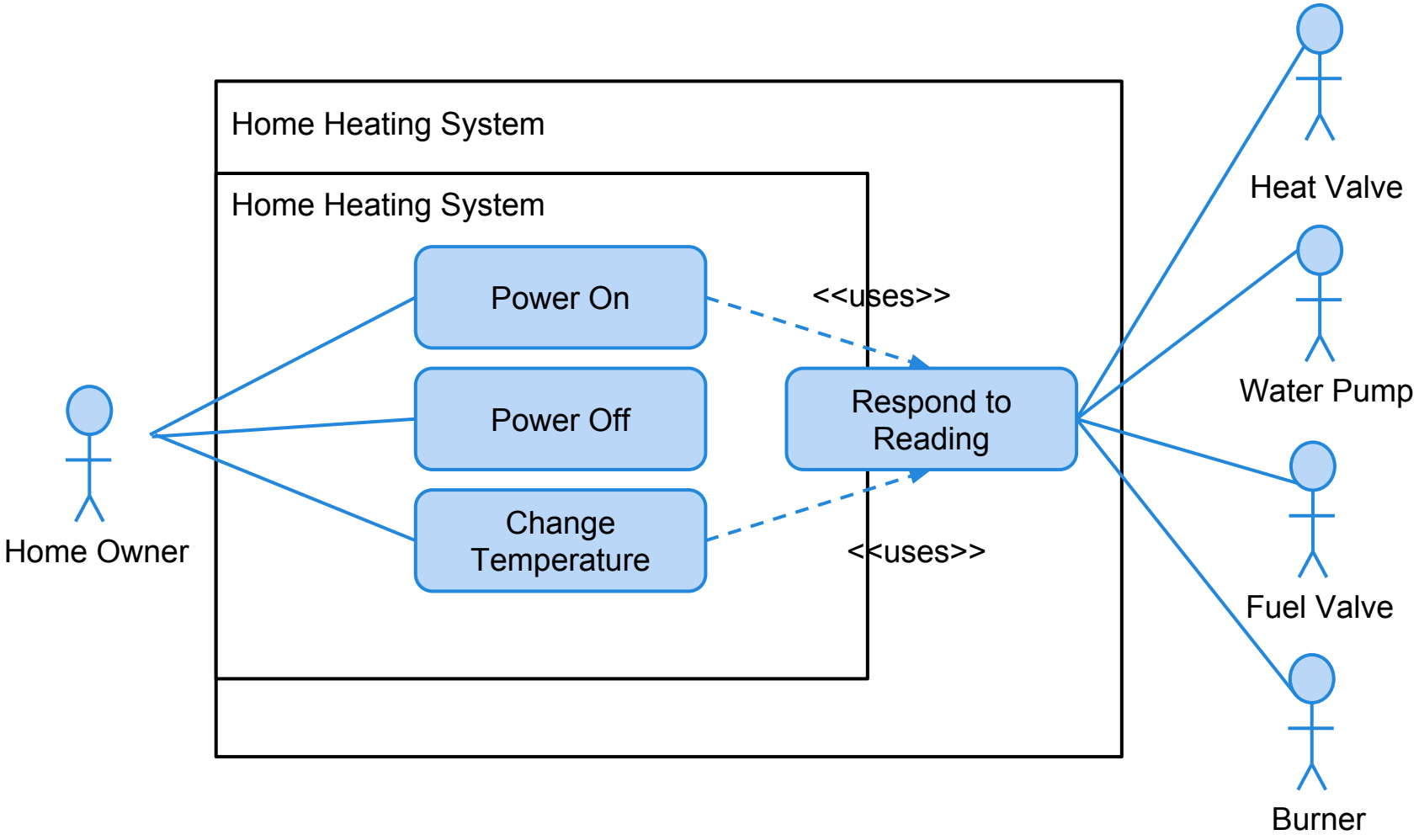
# Home Heating Use Case: Power On

- **Use-Case: Power On**
- **Actors:**
  - Home Owner (initiator)
- **Description:**
  - The Home Owner turns the power on.
  - Each room is temperature checked.
  - If a room is below the desired temperature:
    - the valve for the room is opened
    - the water pump is started
    - the fuel valve opened
    - and the burner ignited.
- **Alternate Paths:** If the temperature in all rooms is above the desired temperature, no actions are taken.

# Home Heating Use Case: Change Temperature

- **Use Case: Change Temperature**
- **Actors:**
  - Home Owner (initiator)
- **Description:**
  - The Home Owner adjusts the temperature using up and down buttons.
  - If no button (up or down) has been pressed for five seconds, then the current setting is taken as the desired temperature.
  - Each room is temperature checked.
  - If a room is below the desired temperature:
    - the valve for the room is opened
    - the water pump started
    - the fuel valve opened
    - and the burner ignited.
- **Alternate Paths:** If the temperature in all rooms is above the desired temperature, no actions are taken.

# Home Heating System Diagram

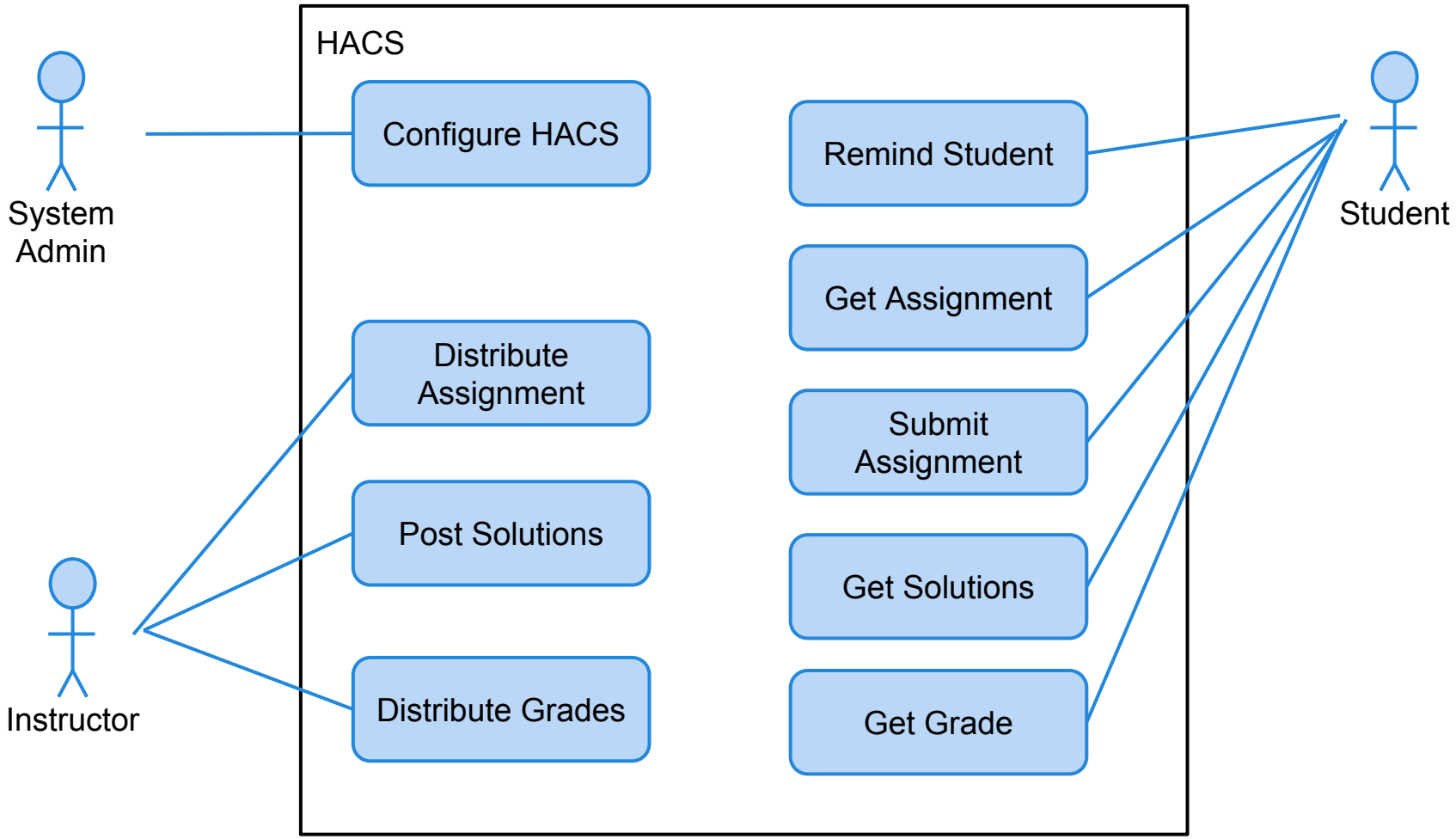


# Activity: HACS

Homework assignment and collection are an integral part of any educational system. Today, this is performed manually. We want to automate this with the Homework Assignment and Collection System (HACS).

HACS will be used by the instructor to distribute the homework assignments, review the students' solutions, distribute suggested solutions, and distribute student grades on each assignment. HACS shall also help the students by automatically distributing the assignments to them, providing a facility where the students can submit their solutions, reminding the students when an assignment is almost due, and reminding the students when an assignment is overdue.

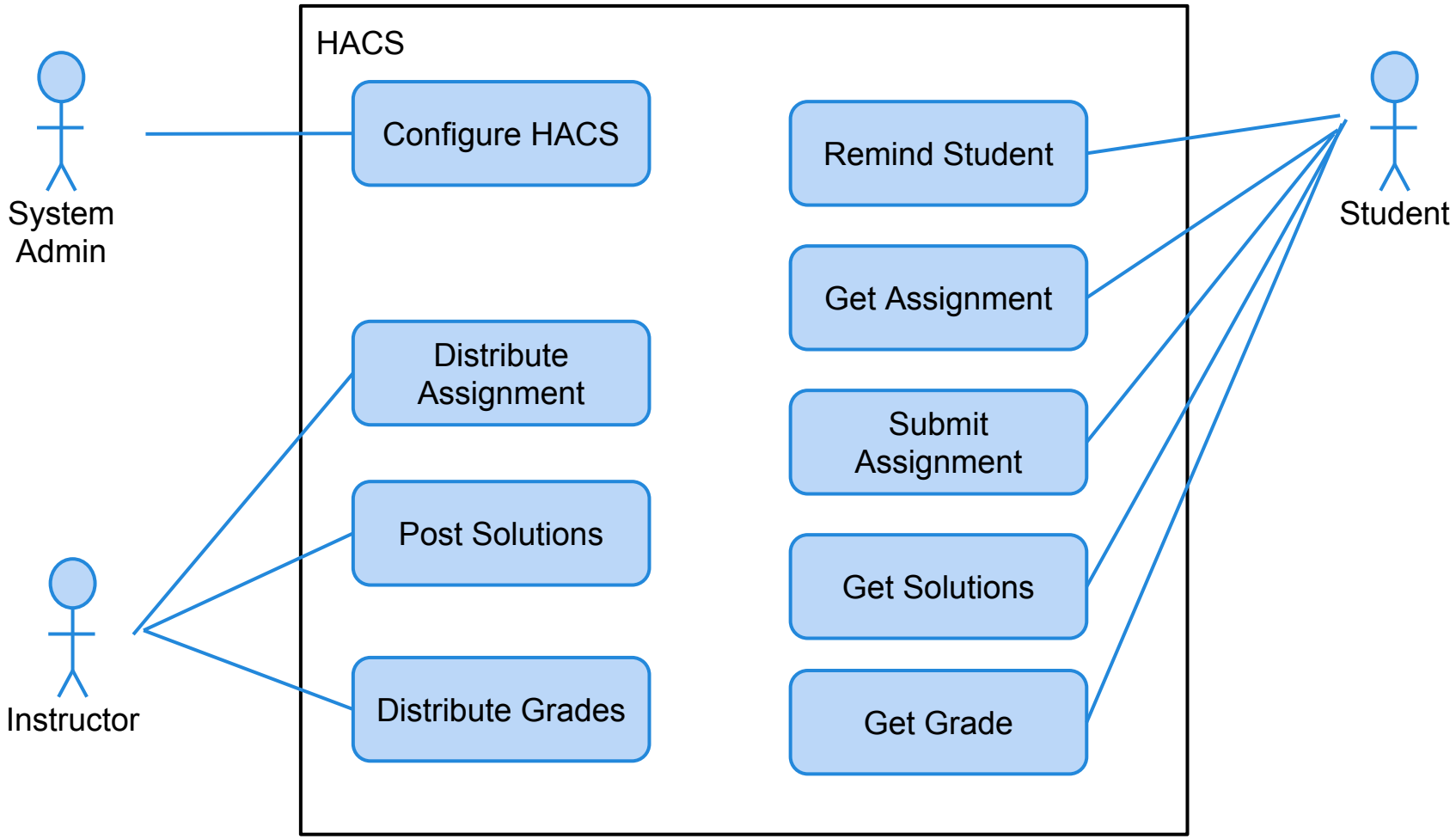
# HACS Use Case Diagram



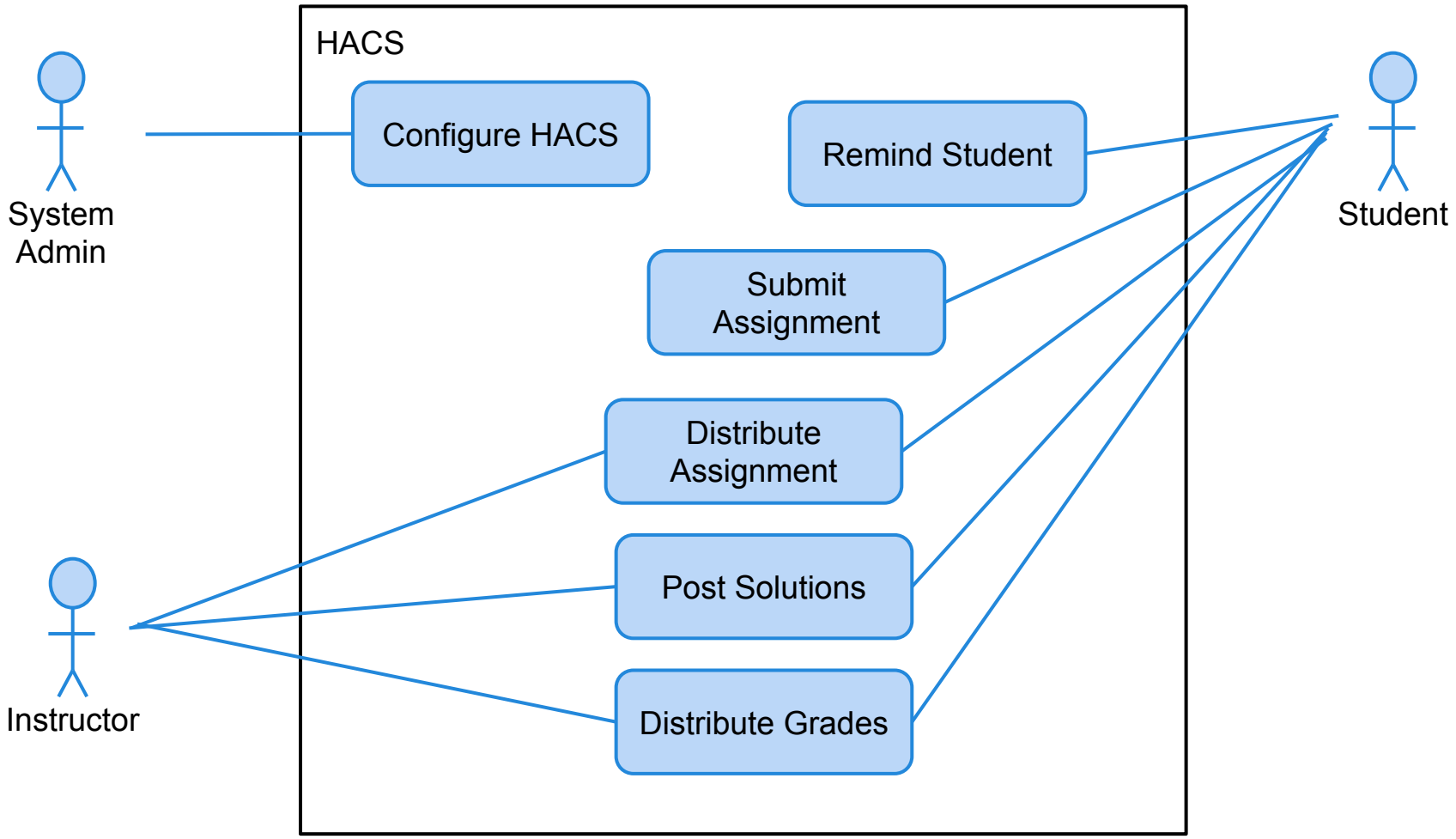
# HACS Use Case: Distribute Assignment

- **Use Case: Distribute Assignment**
- **Actors:** Instructor (initiator)
- **Description:**
  - The Instructor uploads an assignment to the system.
  - If the upload completes successfully, the Instructor will be asked to evaluate a preview of the file.
  - If the Instructor approves the file preview, HACS will ask for a due date.
  - Once the due date is submitted, the assignment will be added to the system and made readable for students, and the Instructor will be returned to the main menu.
- **Exception Paths:** If the file upload fails, an error message will be displayed, and the Instructor returned to the main menu.
- **Alternate Paths:** At any time, the Instructor may click the cancel button to return to the main menu.
- **Preconditions:** Use Case “Configure HACS” must be performed before assignments can be distributed.

# HACS Use Case Diagram



# HACS Use Case Diagram (Version 2)



# HACS Use Case: Distribute Assignment (Version 2)

- **Actors:** Instructor (initiator), Student
- **Description:**
  - The Instructor uploads an assignment to the system.
  - If the upload completes successfully, the Instructor will be asked to evaluate a preview of the file.
  - If the Instructor approves the file preview, HACS will ask for a due date.
  - Once the due date is submitted, the assignment will be added to the system and the Instructor will be returned to the main menu.
  - HACS will then make the assignment readable for students and e-mail each student a link to the file, along with a due date notice.
- **Exception Paths:** If the file upload fails, an error message will be displayed, and the Instructor returned to the main menu.
- **Alternate Paths:** At any time, the Instructor may click the cancel button to return to the main menu.
- **Preconditions:** Use Case “Configure HACS” must be performed before assignments can be distributed.

# When should we use Use Cases?

- In short… Always!
- Requirements specification is the hardest part of software development. Use cases are a powerful tool to understand:
  - Who your users are (including non-human systems).
  - What functions your system should provide.
  - How these functions work (at a high level).

# Things to Keep in Mind

- Remember:
  - Each use case will likely correspond to many requirements. Use cases are high level goals, requirements are low level statements of how to make that goal achievable.
  - Use cases represent an external view of the system. They do not tell you what your system objects are, and should not feature internal objects as actors.
  - No “rule of thumb” for how many use cases you should have:
    - Ask yourself: does this capture all of the goals a user might have when using my system?

# We Have Learned

- Develop use cases to identify requirements.
  - Ask the customer, “what do you want to accomplish?”
- Consider all stakeholders - different stakeholders have different viewpoints.
- Always have heavy customer involvement.
- Use scenarios and templates to avoid forgetting things, refine your use cases.
  - Better use cases lead to better requirements.

# Next Time

- Requirement refinement and testability.
- Reading:
  - Sommerville, chapter 8
    - Introduction, section 8.3.1, 8.3.2
- Virtual requirements elicitation
- Topics to think about:
  - Who are the stakeholders and actors?
  - What functionality does MEAT need to offer?
  - Ask many “what if?” questions!