Understanding the customer's requirements for a software system

Requirements Analysis

Announcements

- Homework 1 – Correction in “Resume” button functionality. Download updated Homework 1 handout from web page.
- Papyrus Tool - Installation instructions on Eclipse 4.2 and step by step guide to start a new UML model are available on the webpage.

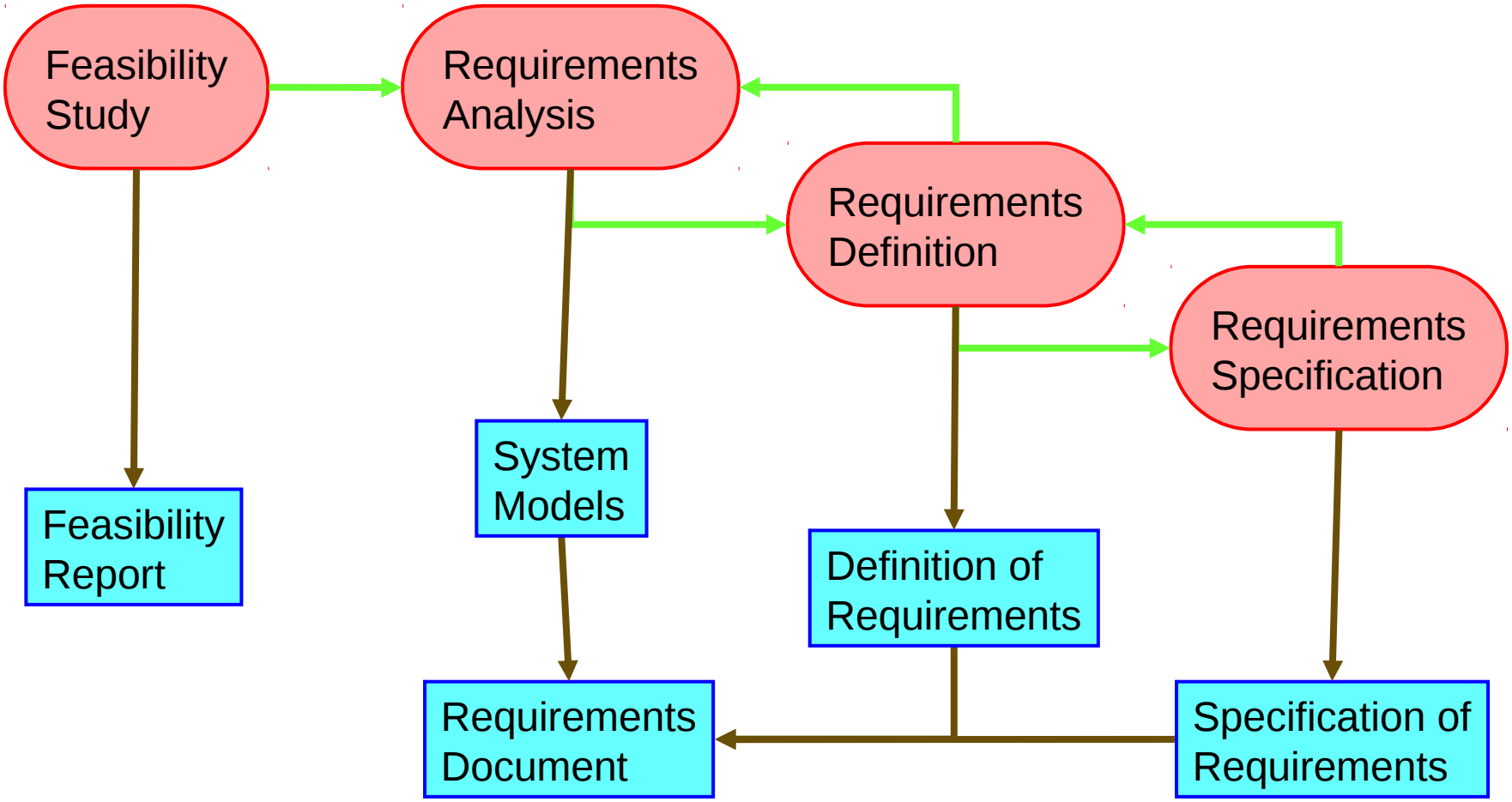
Goals For Today

- Understanding the concept of Stakeholder
- Discuss a few techniques for getting all the information we need to develop a system
  - Requirements Elicitation/Analysis

Requirements Analysis

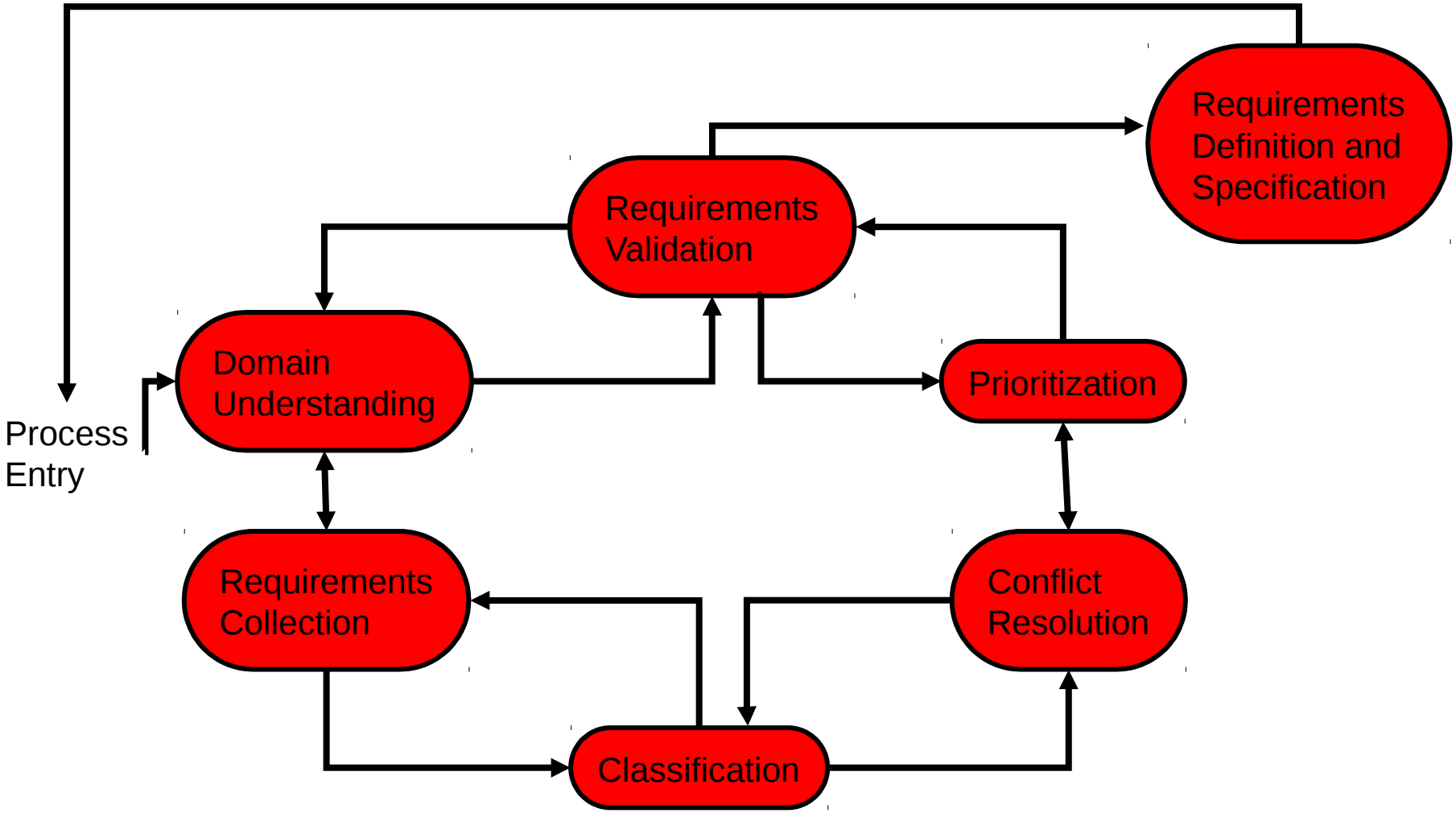
- Sometimes called requirements elicitation or requirements discovery
- Involves supplier (technical or non-technical) staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.
  - **These are called stakeholders**

The Requirements Engineering Process



Requirements Elicitation

The Requirements Analysis Process



Interview the User

- Make sure you have the right user
- Is this user's answers official?

- Find out what they are willing to pay for each function
  - Helps in prioritization
- Try to be context free
  - Avoid
  - We thought you knew that.
  - We always do it that way.
- Hundreds of techniques
  - More or less suitable for your organization and problem
  - Be creative and do some research before the process starts

Viewpoint-Oriented Analysis

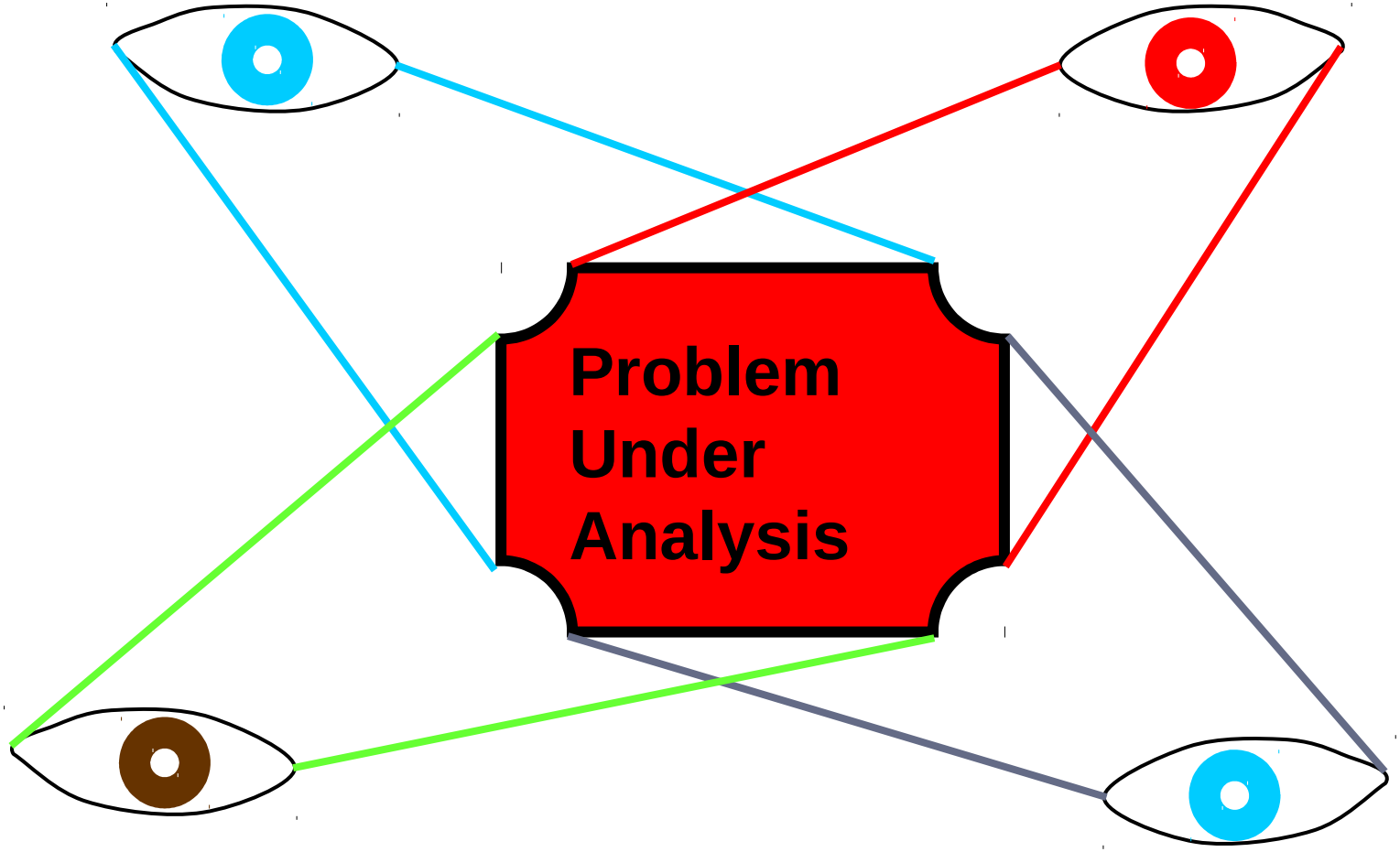
- Stakeholders represent different ways of looking at a problem or problem viewpoints
- This multi-perspective analysis is important as there is no single correct way to analyze system requirements

  - Ian Sommerville

Stakeholders Must Work

Stakeholders Must Work

Multiple Problem Viewpoints



Types of Viewpoint

- Data sources or sinks
  - Viewpoints are responsible for producing or consuming data
- Receivers of services
  - Viewpoints are external to the system and receive services from it
- Experts in the domain
- Representation frameworks
  - Viewpoints represent particular types of system models

Autoteller Viewpoints

Autoteller Viewpoints

Bank customers

Representatives of other banks

Hardware and software maintenance engineers

Marketing department

Bank managers and counter staff

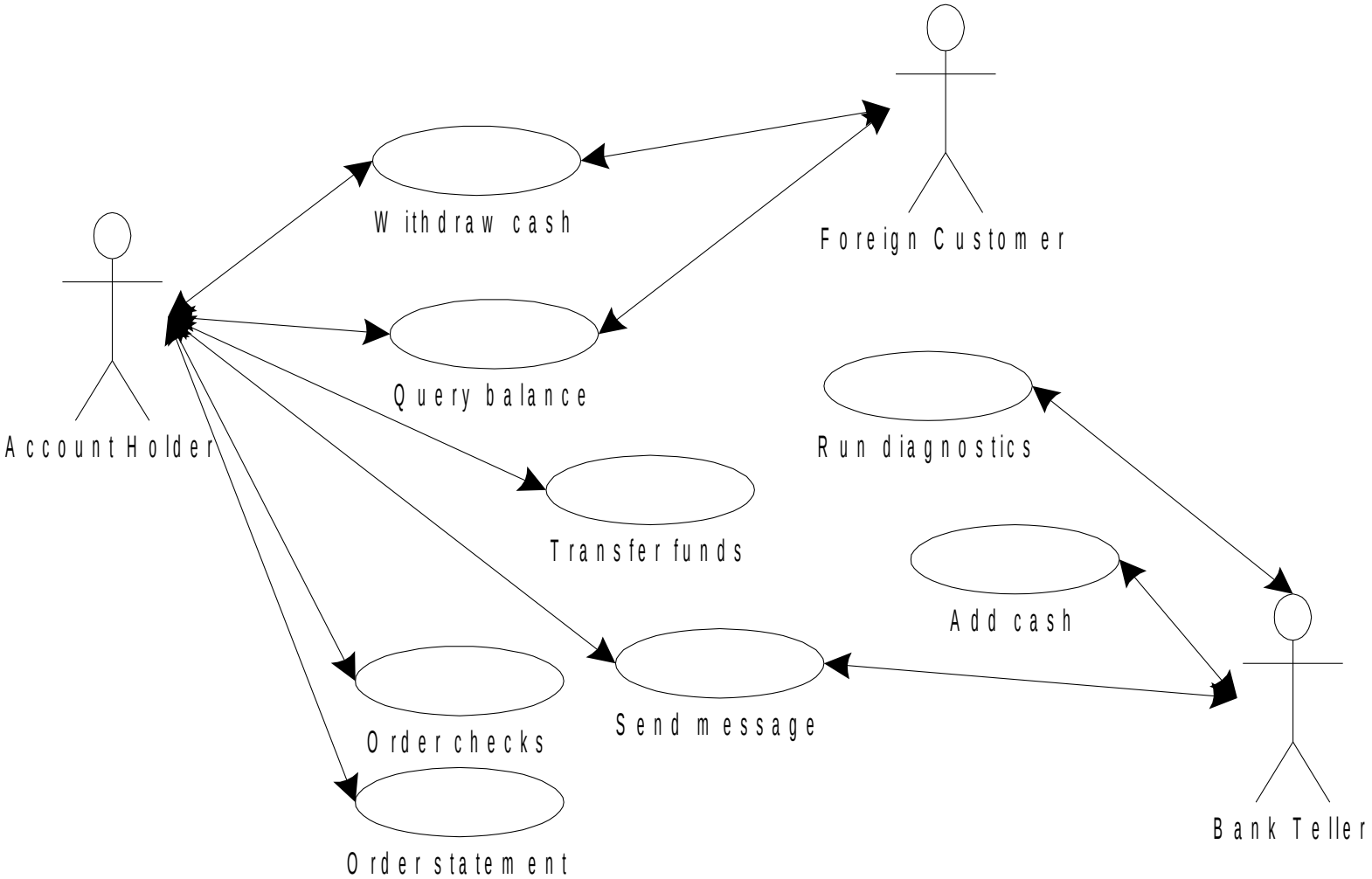
Database administrators

Security staff

Communications engineers

Personnel department

Use Cases in UML



What is a Use-Case

- A use-case captures some user visible function
- This may be a large or small function
  - Depends on the level of detail in your modeling effort
- A use-case achieves a discrete goal for the user
- Examples
  - Format a document
  - Request an elevator
- How are the use cases found (captured or elicited)?

User Goals versus User Interactions

- Consider the following when formatting a document
  - Define a style
  - Change a style
  - Copy a style from one document to the next
  - versus
  - Format a document
  - Ensure consistent formatting of two documents
- The latter is a user goal
  - Something the user wants to achieve
- The former are user interactions
  - Things the user does to the system to achieve the goal

Goals and Interactions

- There is a place for both goals and interactions
- Understand what the system shall do
  - Capture the user goals
- Understand how the user will achieve the goals
  - Capture user interactions
  - Sequences of user interactions
- Thus, start with the user goals and then refine the user goals into several (many) user interactions
- Users are referred to as **Actors** (does not have to be human)
  - Primary Actor- is the actor with the goal the use case is trying to satisfy
  - Secondary Actor – Actor with which the system communicates while carrying out the use case.

Use Case Goal – Buy a Product

The customer browses the catalogue in an on-line shop and adds the desired items to the basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit cards and confirms the sale both immediately and with a follow up email.

- Other scenarios – Credit card authorization might fail, you may have a regular customer for whom you don't need to capture shipping and credit card information.
- Scenarios are different, yet the goal of the user in all the scenarios is the same - “Buy a Product”.
- A **use case** is a set of scenarios tied together by a common user goal.

Buy a Product Use-Case

Buy a Product

Main Success Scenario

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills shipping info.
4. Systems presents full pricing information.
5. Customer fills in credit card info.
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 and billing information

.2:Customer may accept or override these defaults, returns to MSS at step 6

6a: System fails to authorize credit card purchase

.1: Customer may reenter credit card information or may cancel

Use-Cases for the Viewpoints

Cash withdrawal - correct PID.
The customer inserts the card in the ATM. The ATM accepts the card and asks the user for the PID. If the PID is correct, the ATM asks the user from which account the funds should be drawn. The user enters the account. The ATM asks the user for the amount. The user enters the amount. The ATM asks the user to verify the account. The user verifies the account. If there are sufficient funds in the account, the money is dispensed and the amount is withdrawn from the account.

Structure the Use-Cases

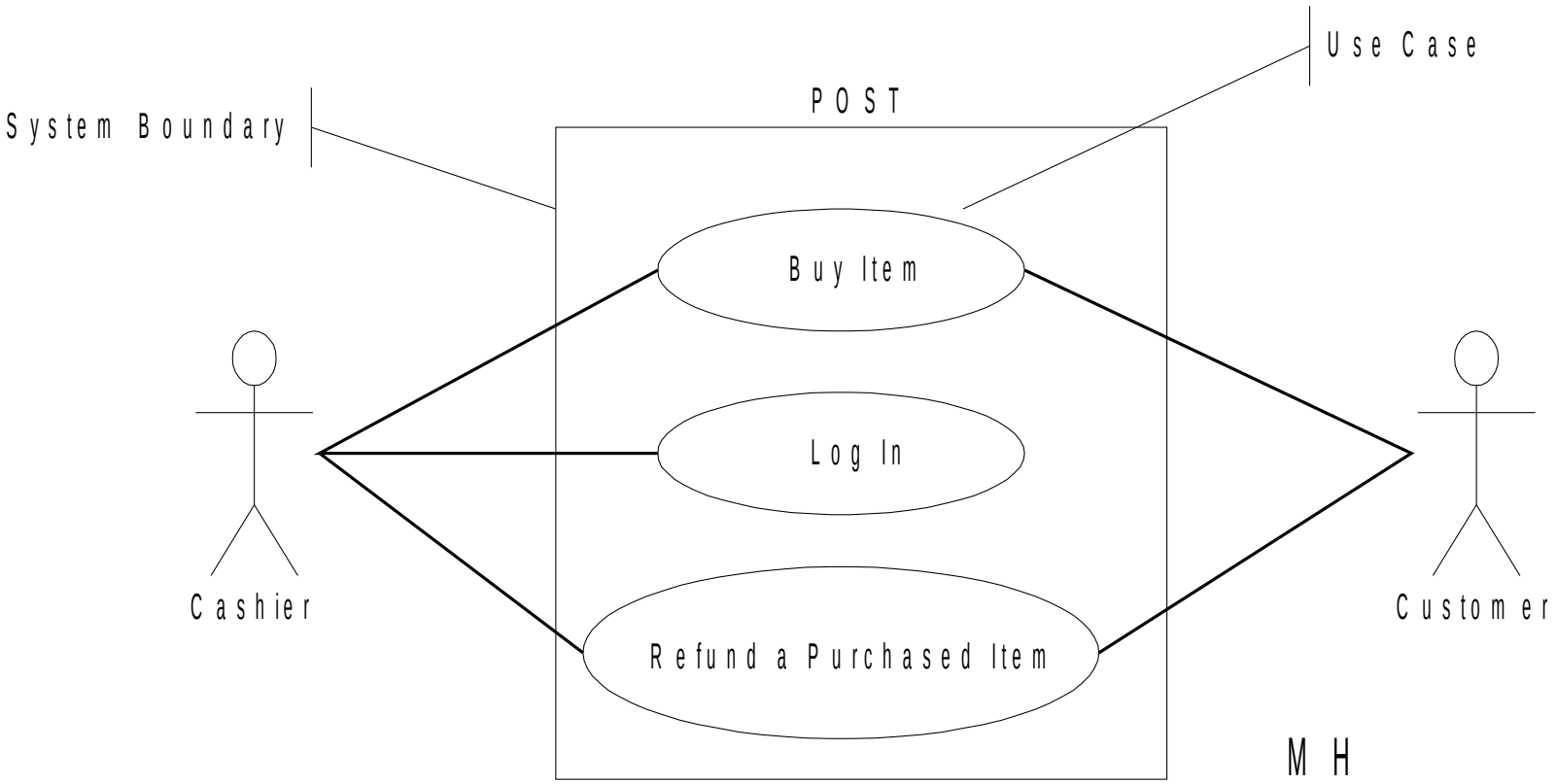
- **Cash withdrawal - correct PID**

1. The customer inserts the card in the ATM.
2. The ATM accepts the card and asks the user for the PID.
3. If the PID is correct, the ATM asks the user from which account the funds should be drawn.
4. The user enters the account.
5. The ATM asks the user for the 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.

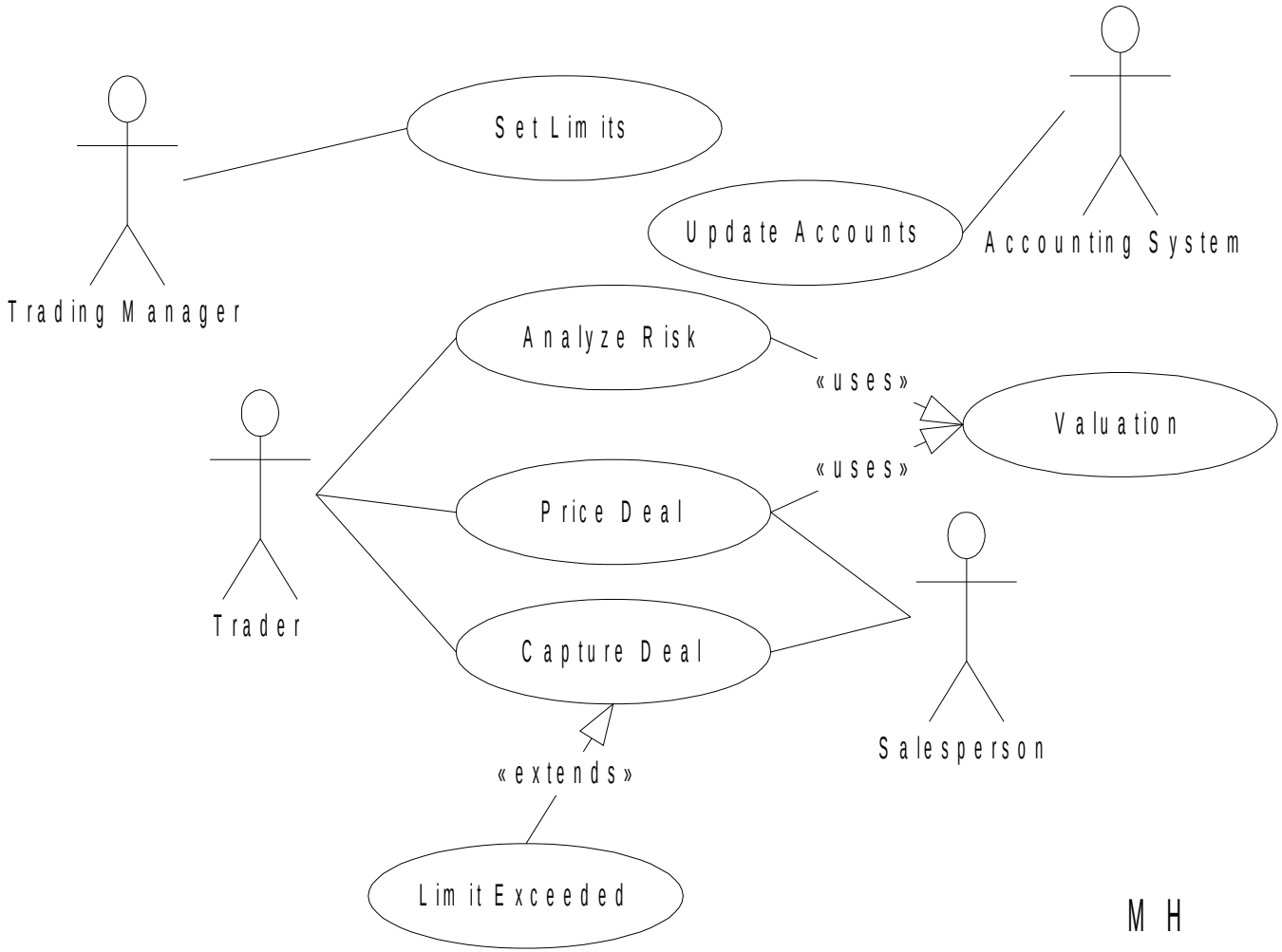
Use-Case Templates Help

- Example on web

Use-Case Diagrams



Example from Fowler

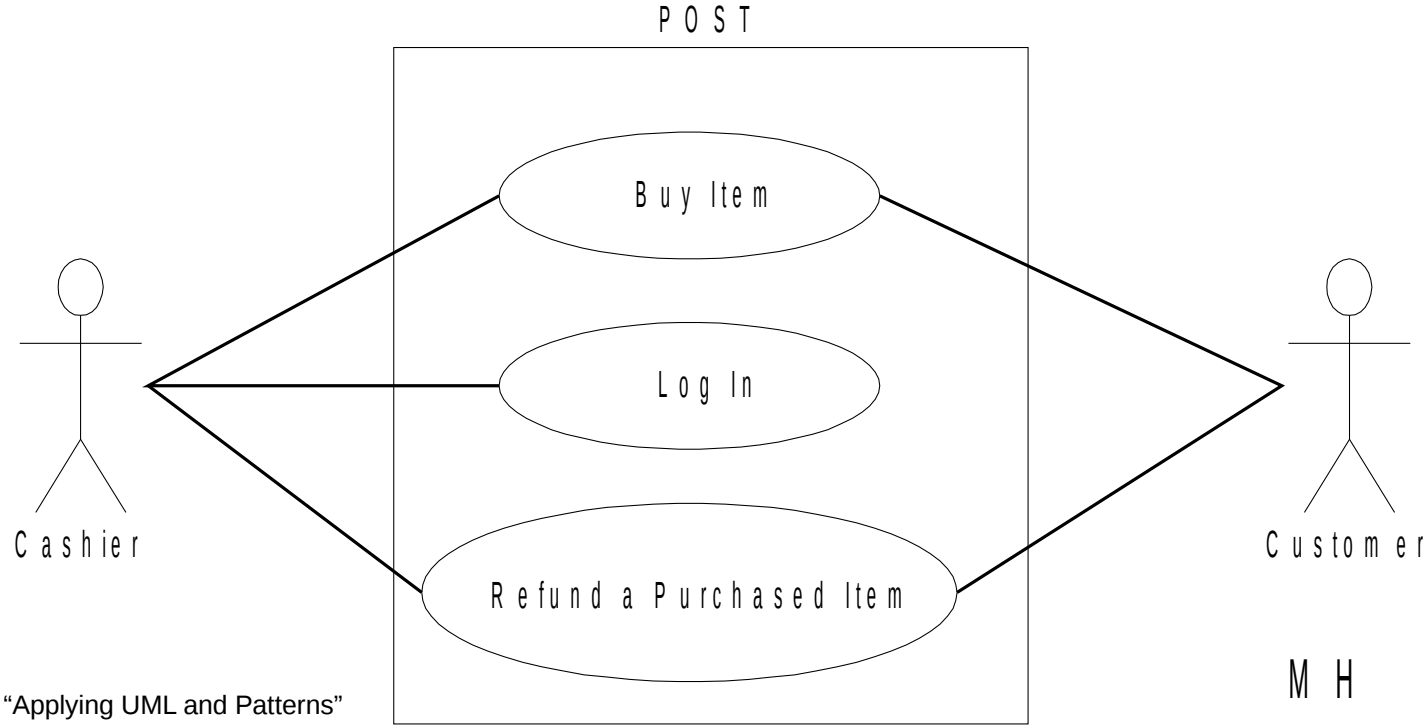


Uses and Extends

- 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 other ones “use” it
  - Examples include
    - Valuation
    - Validate user interaction
    - Sanity check on sensor inputs
    - Check for proper authorization
- Extends
  - A use-case is similar to another one but does a little bit more
  - Put the normal behavior in one use-case and the exceptional behavior somewhere else
  - Capture the normal behavior
  - Try to figure out what can go wrong in each step
  - Capture the exceptional cases in separate use-cases
  - Makes it a lot easier to understand

Setting the System Boundary

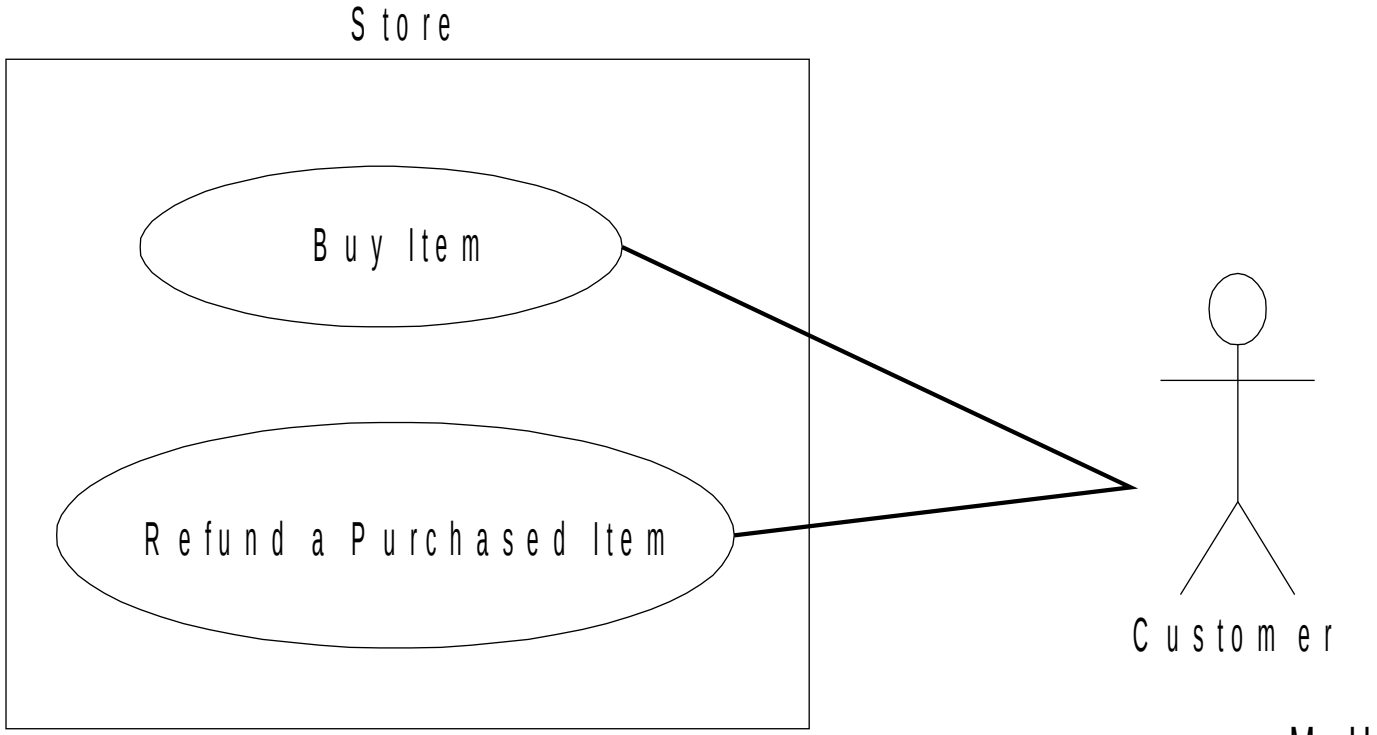
- The system boundary will affect your actors and use-cases



Adapted from Larman “Applying UML and Patterns”

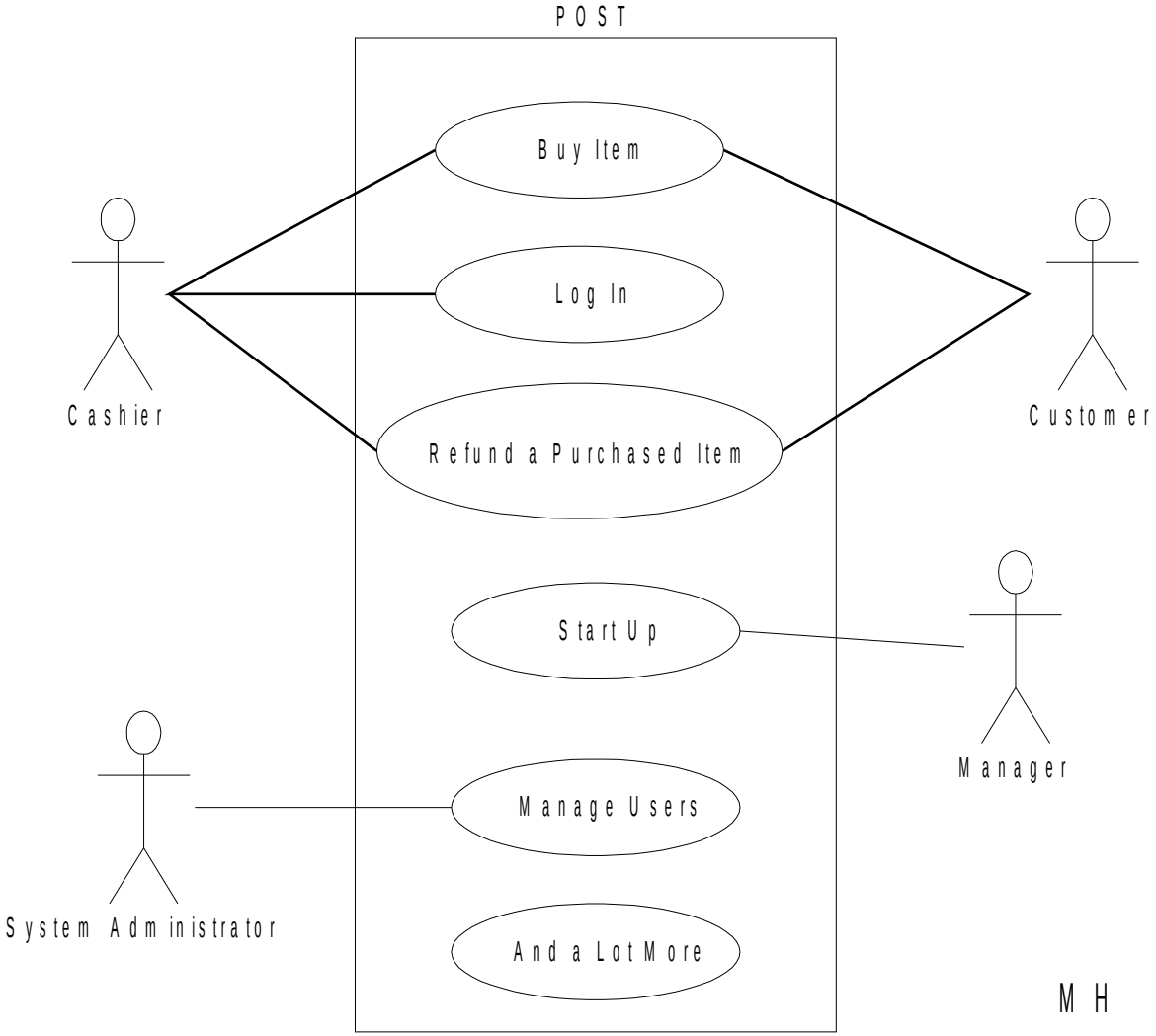
A Different Boundary

- Let us view the whole store as our system



Adapted from Larman “Applying UML and Patterns”

Partial POST



POST Scenario

Use case: **Buy Item**

Actors: Customer (initiator), Cashier

Type: Primary

Description: The Customer arrives at the checkout with items to purchase. The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items

POST Expanded Scenario

Use case: **Buy Item**

Actors: Customer (initiator), Cashier

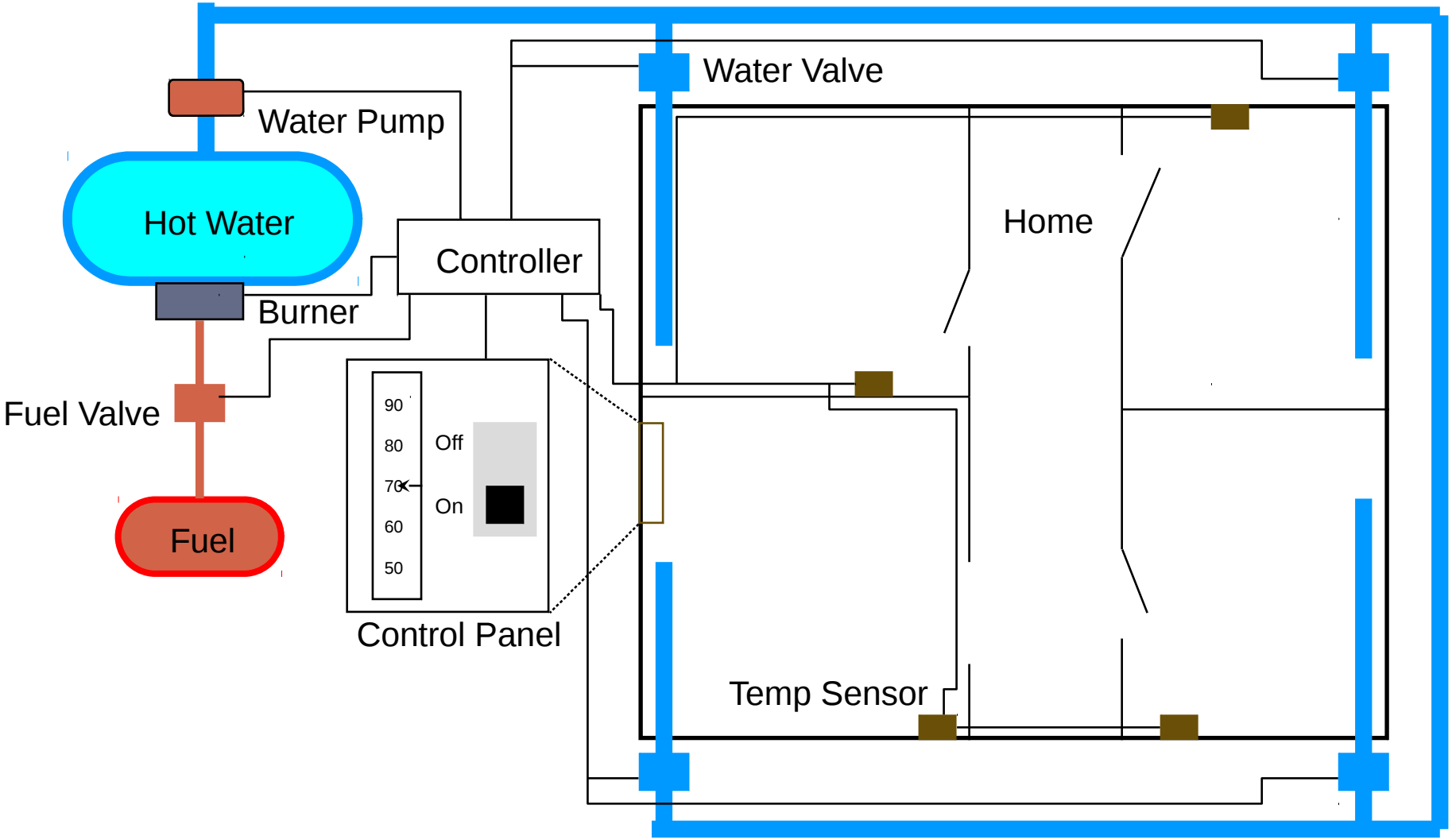
Type: Primary and essential

Description: The Customer arrives at the checkout with items to purchase. The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items.

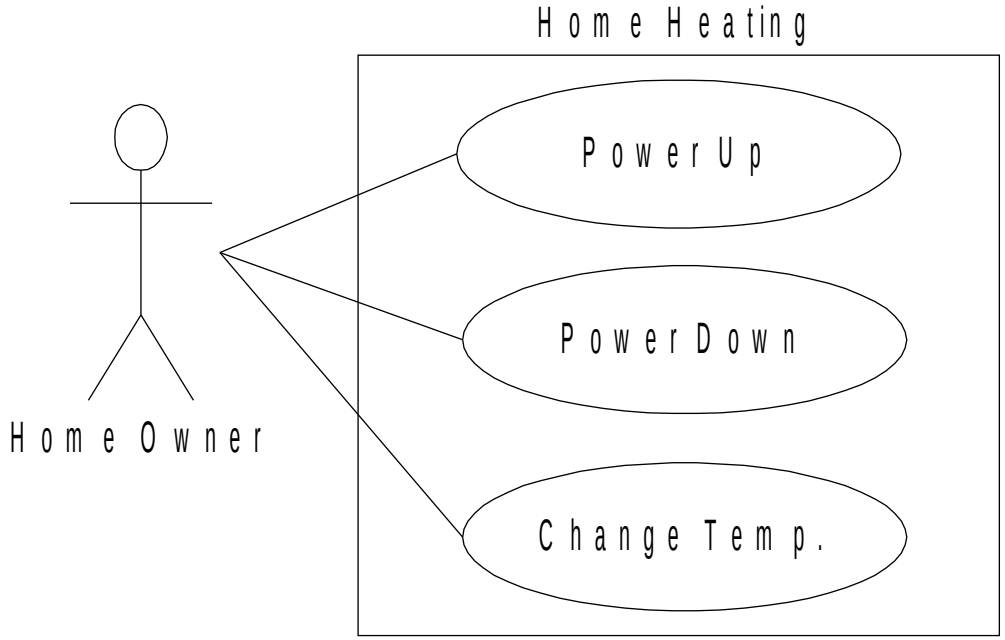
Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: Cashier must have completed the *Log In* use-case

The Home Heating System



Home Heating Use-Case Diagram

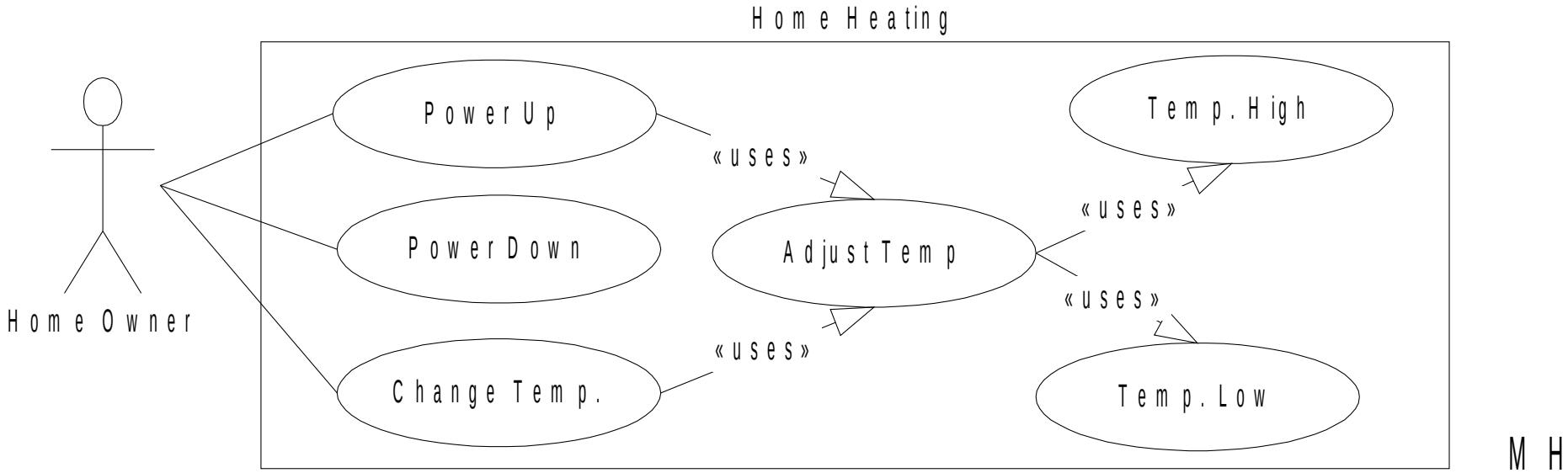


M H

Home Heating Scenario

Use case: **Power Up**
Actors: Home Owner (initiator)
Type: Primary and essential
Description: The Home Owner turns the power on. Each room is temperature checked. If a room is below the the desired temperature the valve for the room is opened, the water pump started, the fuel valve opened, and the burner ignited.
If the temperature in all rooms is above the desired temperature, no actions are taken.
Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: None

Modified Home Heating

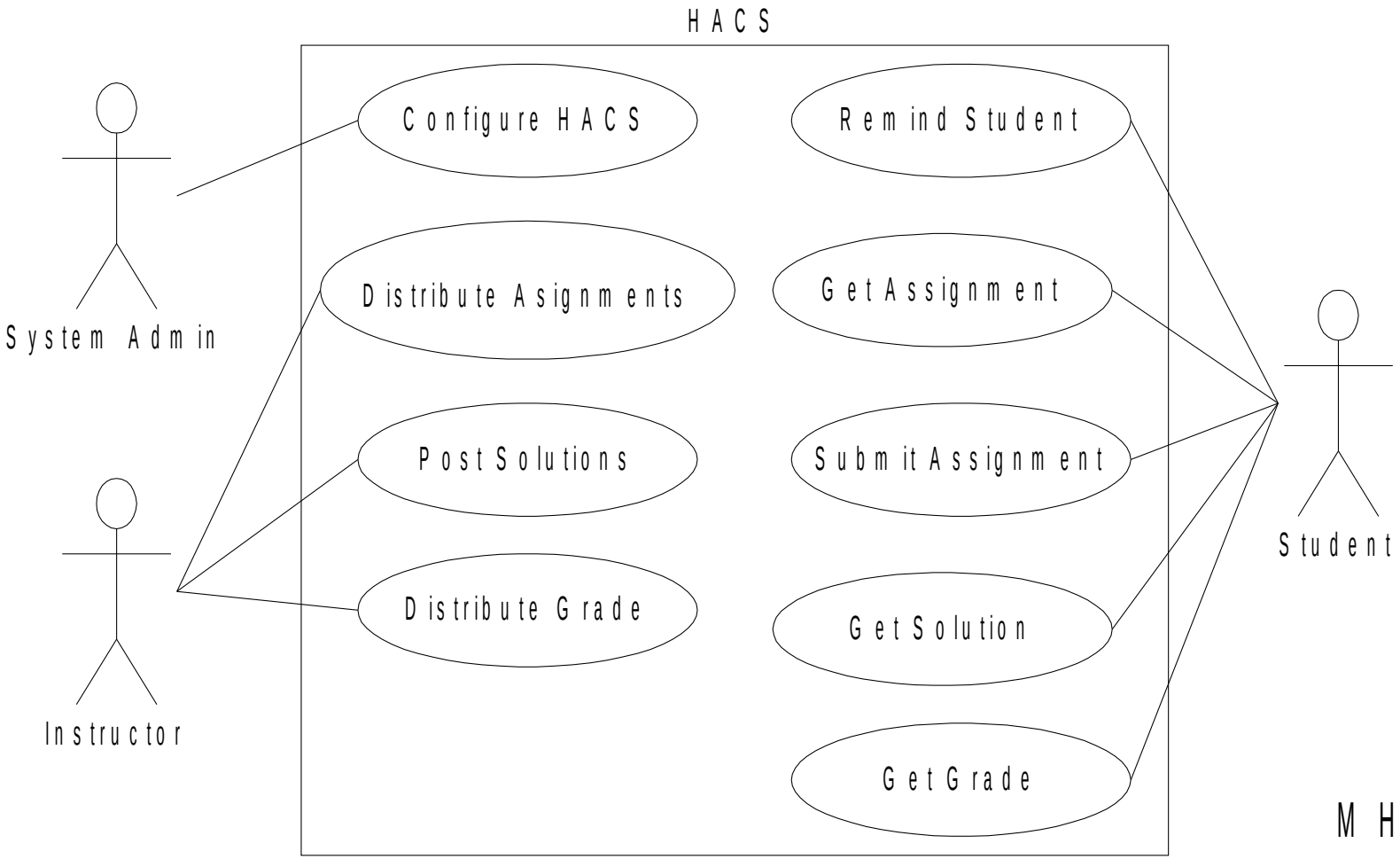


M H

HACS

- Homework assignment and collection are an integral part of any educational system. Today, this task is performed manually. What we want the homework assignment distribution and collection system (HACS for short) to do is to automate this process.
- HACS will be used by the instructor to distribute the homework assignments, review the students' solutions, distribute suggested solution, and distribute student grades on each assignment.
- HACS shall also help the students by automatically distribute the assignments to the students, provide a facility where the students can submit their solutions, remind the students when an assignment is almost due, remind the students when an assignment is overdue.

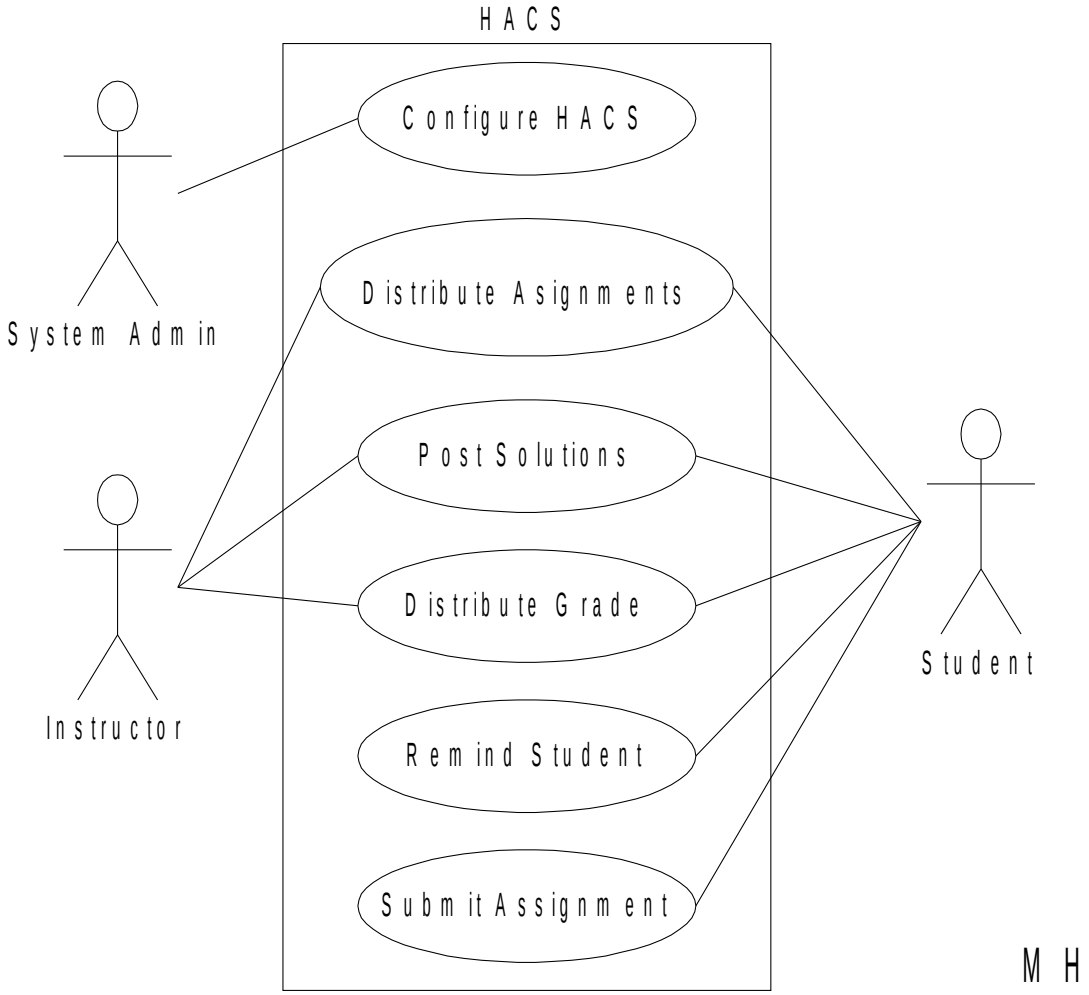
HACS Use-Case Diagram



HACS Scenario

Use case: Distribute Assignments
Actors: Instructor (initiator)
Type: Primary and essential
Description: The Instructor completes an assignment and submits it to the system. The instructor will also submit the due date and the class the assignment is assigned for.
Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: *Configure HACS* must be done before any user (Instructor or Student) can use HACS

Alternate HACS



Alternate HACS Scenario

Use case: Distribute Assignments
Actors: Instructor (initiator), Student
Type: Primary and essential
Description: The Instructor completes an assignment and submits it to the system. The instructor will also submit the delivery date, due date, and the class the assignment is assigned for. The system will at the due date mail the assignment to the student.
Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: *Configure HACS* must be done before any user (Instructor or Student) can use HACS

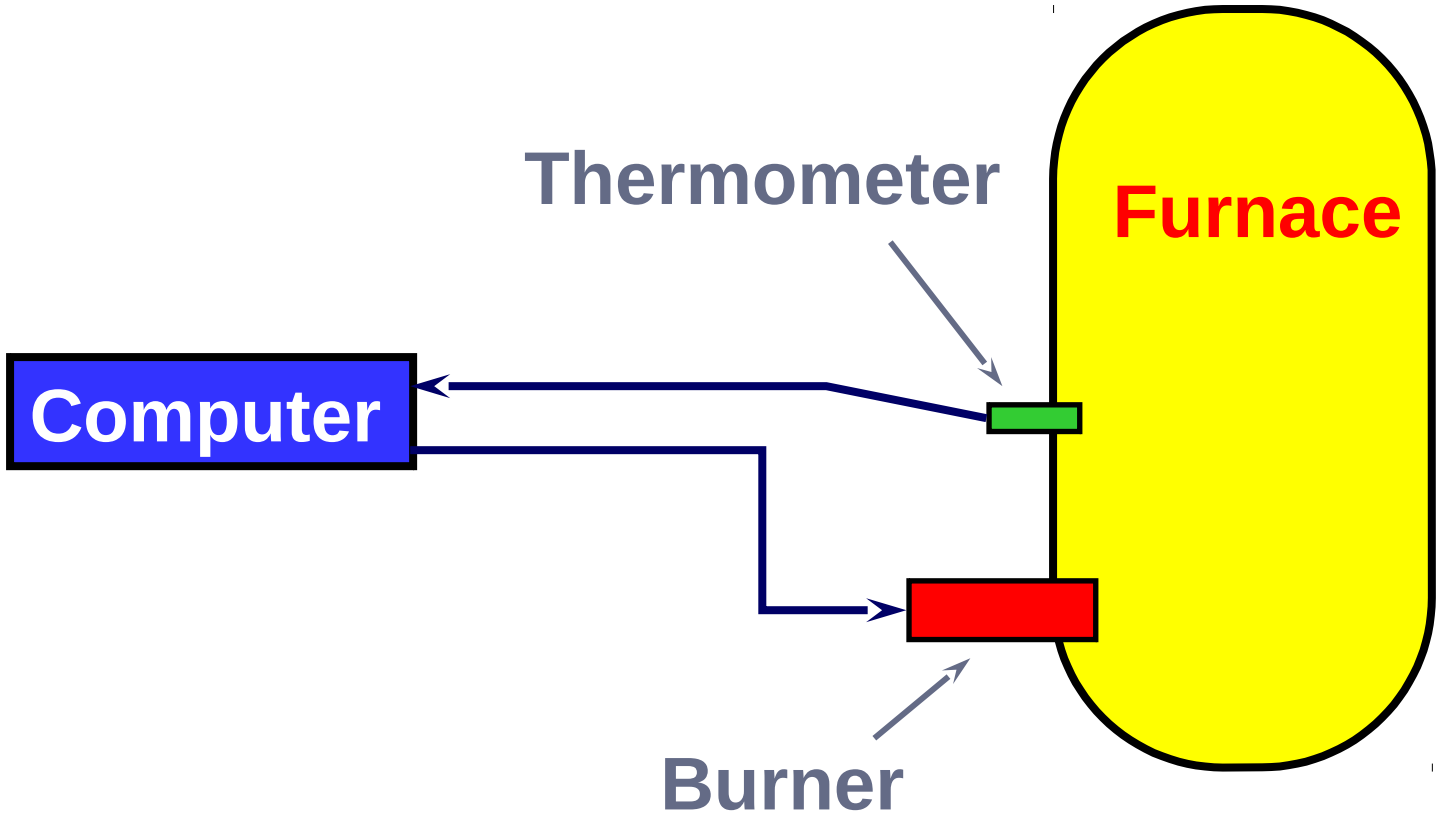
When to use Use-Cases

- In short, always!!!
- Requirements is the toughest part of software development
  - Use-Cases is a powerful tool to understand
  - Who your users are (including interacting systems)
  - What functions the system shall provide
  - How these functions work at a high level
- Spend adequate time on requirements and in the elaboration phase

Checklists and Testing

Use Checklists to Avoid Forgetting

Missing Requirements



How Do We Avoid Forgetting?

- Find skilled application specialists
  - “Domain Engineers”
  - Let them develop the requirements
- Consider as many viewpoints as you can
- Gather use-cases with the customers
- Use checklists and guidelines to point out common problems
  - Application independent guidelines and properties
  - System specific guidelines and properties

Properties of Embedded Systems

- Is the software's response to out-of-range values specified?
- Is the software's response to not receiving an expected input specified?
- If input arrives when it should not, is a response specified?
- Is startup behavior adequately specified?
- Are all possible scenarios covered?
  - Have we specified exceptional behavior?

Checklists are Effective

- On two NASA spacecraft projects 192 critical errors were found during integration and testing
- 142 of those errors were addressed by a simple safety-checklist
- Most problems with unexpected input
  - Unexpected value as well as unexpected timing

Key Points

- Consider all stakeholders to complete the functional and non-functional requirements
- Always have a heavy customer involvement
- Use checklists with known problem areas to avoid forgetting things
  - Learn from other’s (and your own) mistakes

If you can’t test it, it is not a requirement!

Derive Test Cases for the Requirements

Requirements Verifiability

- Consider the requirement:
  - The system should be easy to use by experienced engineers and should be organized in such a way that user errors are minimized.
- The problem with this requirement is its use of vague terms such as “errors shall be minimized”
- The error rate must be quantified
  - Experienced engineers shall be able to use all the system functions after a total of two hours training
  - After this training, the average number of errors made by experienced engineers shall not exceed two per day

Typical Requirements

- After a high temperature is detected, an alarm must be raised quickly
- Novice users should be able to learn the interface with little training
- **How in the world do you test requirements like this??**

Test the Requirement

Test Case 1

Input

Artificially raise the temperature above *threshold*

Test procedure

Measure the time it takes for the *alarm* to come on

Expected output

The *alarm* shall be on within 2 seconds

Test the Requirement

Test Case 2

Input

Identify 10 *new users* and put them through the training course (max 6 hours)

Test procedure

Monitor the work of the users for 10 days after the training course has been completed

Expected output

The average error rate over the 10 days shall be less than 3 *entry errors* per 8 hours of work

“Fixed” Requirements

- When the temperature rises over threshold, the alarm must be turned on within 2 seconds

- New users of the system shall be able to use the system after 6 hour of training
  - Using the system is defined as making less than 3 entry mistakes per 8 hours of operation

Test Cases for Various Levels

- Requirements Definition
  - 1. One person must be able to load the boat on the car rack
- Requirements Specification
  - 1.1 The boat must be lighter than 100 lb.
  - 1.2 The boat must have handles to help one person lift it
  - 1.3 The car rack must be padded so the boat can easily slide into the rack
  - 1.4 Etc.

Key Points

- Do yourself and the testing group a favor—**Develop Test Cases for Each Requirement**
- If the requirement cannot be tested, you most likely have a bad requirement
  - Rewrite so it is testable
  - Remove the requirement
  - Point out why this is an untestable requirement
- **Your requirements and testing effort will be greatly improved**