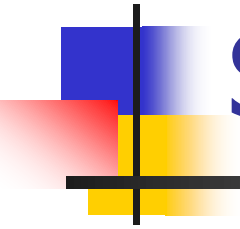
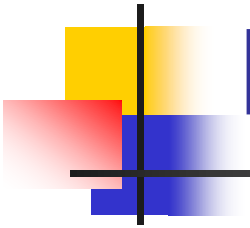


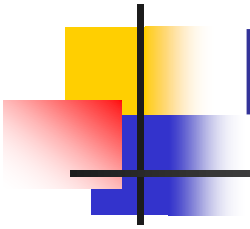
Software Engineering Unit-2

Part 1 Requirements Engineering



Requirements

- *A customer walks into your office, sits down, looks you straight in the eye, and says, "I know you think you understand what I said, but what you don't understand is what I said is not what I meant."*
- The broad spectrum of tasks and techniques that lead to an understanding of requirements is called *requirements engineering*.



Requirements engineering

- It encompasses seven distinct tasks:
  - Inception
  - Elicitation
  - Elaboration
  - Negotiation
  - Specification
  - Validation
  - Management

Requirements engineering

- **Inception.**
  - Stakeholders from the business community (e.g., business managers, marketing people, product managers) define a business case for the idea, try to identify the breadth and depth of the market, do a rough feasibility analysis, and identify a working description of the project's scope.
  - Establish a basic understanding of the problem.
  - Nature of the solution that is desired.
  - Collaboration between the other stakeholders and the software team.

Requirements engineering

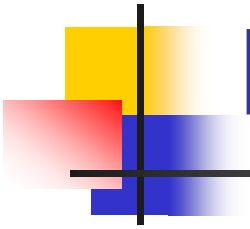
- **Elicitation.**
  - An important part of elicitation is to establish business goals.
  - Goals need to be prioritized.
  - Key points/Problems that need to be addressed.
    - *Problems of scope: system is ill-defined, unnecessary technical detail.*
    - *Problems of understanding: poor understanding of the capabilities and limitations of their computing environment.*
    - *Problems of volatility: occur when the requirements change over time.*

Requirements engineering

- **Elaboration.**
  - It focuses on developing a refined requirements model that identifies various aspects of software function, behavior, and information.
  - Creation and refinement of user scenarios.
    - Each user scenario is parsed to extract classes, attributes, services.
    - The relationships and collaboration between classes are identified.

Requirements engineering

- **Negotiation.**
  - Customers, users, and other stakeholders are asked to rank requirements and then discuss conflicts in priority.
  - Requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction.



Requirements engineering

- **Specification.**
  - A specification can be a written document, a set of graphical models, a formal mathematical model, a collection of usage scenarios, a prototype, or any combination of these.

Requirements engineering

- **Specification.**

Software Requirements Specification Template

A *software requirements specification* (SRS) s a work product that is created when a detailed de- cription of all aspects of the software to be built must e specified before the project is to commence. It is im- ortant to note that a formal SRS is not always written. In act, there are many instances in which effort expended n an SRS might be better spent in other software en- ineering activities. However, when software is to be leveloped by a third party, when a lack of specification ould create severe business issues, or when a system is xtremely complex or business critical, an SRS may be stified.

Karl Wiegers [Wie03] of Process Impact Inc. has leveloped a worthwhile template (available at **ww.processimpact.com/process_assets/ rs_template.doc**) that can serve as a guideline for ose who must create a complete SRS. A topic outline llows:

able of Contents

levision History

1. **Introduction**
   - 1.1 Purpose
   - 1.2 Document Conventions
   - 1.3 Intended Audience and Reading Suggestions
   - 1.4 Project Scope
   - 1.5 References
2. **Overall Description**
   - 2.1 Product Perspective
   - 2.2 Product Features
   - 2.3 User Classes and Characteristics
   - 2.4 Operating Environment
   - 2.5 Design and Implementation Constraints
   - 2.6 User Documentation
   - 2.7 Assumptions and Dependencies
3. **System Features**
   - 3.1 System Feature 1
   - 3.2 System Feature 2 (and so on)
4. **External Interface Requirements**
   - 4.1 User Interfaces
   - 4.2 Hardware Interfaces
   - 4.3 Software Interfaces
   - 4.4 Communications Interfaces
5. **Other Nonfunctional Requirements**
   - 5.1 Performance Requirements
   - 5.2 Safety Requirements
   - 5.3 Security Requirements
   - 5.4 Software Quality Attributes
6. **Other Requirements**

Appendix A: Glossary
Appendix B: Analysis Models
Appendix C: Issues List

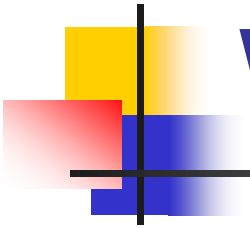
A detailed description of each SRS topic can be ob- tained by downloading the SRS template at the URL noted in this sidebar.

Requirements engineering

- **Validation.**
  - Requirements validation examines the specification to ensure that all software requirements have been stated unambiguously; that inconsistencies, omissions, and errors have been detected and corrected; and that the work products conform to the standards established for the process, the project, and the product.

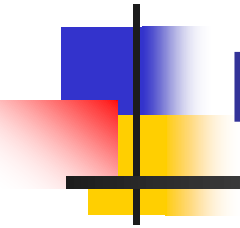
Requirements engineering

- **Requirements management.**
  - It is a set of activities that help the project team identify, control, and track requirements and changes to requirements at any time as the project proceeds.

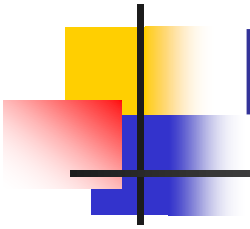


Wrong Requirements

- The software should be user friendly.
- The probability of a successful unauthorized database intrusion should be less than 0.0001.



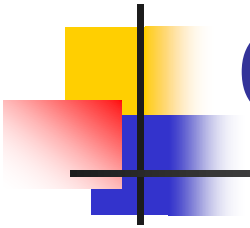
ELICITING REQUIREMENTS



Eliciting Requirements

- **Collaborative Requirements Gathering.**
- **Quality Function Deployment.**
- **Usage Scenarios.**
- **Elicitation Work Products.**
- **Agile Requirements Elicitation.**
- **Service-Oriented Methods.**

Collaborative Requirements Gathering



- Meetings (either real or virtual) are conducted and attended by both software engineers and other stakeholders.
- Rules for preparation and participation are established.
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
- A “facilitator” (can be a customer, a developer, or an outsider) controls the meeting.
- A “definition mechanism” (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room, or virtual forum) is used.

Collaborative Requirements Gathering-Safe Home Case study

Our research indicates that the market for home management systems is growing at a rate of 40 percent per year. The first *SafeHome* function we bring to market should be the home security function. Most people are familiar with "alarm systems" so this would be an easy sell.

The home security function would protect against and/or recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, carbon monoxide levels, and others. It'll use our wireless sensors to detect each situation, can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

Stakeholders
Business manager
Marketing Person

Objects
control panel,
smoke detectors,
window and door sensors,
motion detectors,
an alarm, an event (a sensor has been activated),
a display,
a PC,
telephone numbers,
a telephone call

Services
configuring the system,
setting the alarm,
monitoring the sensors,
dialing the phone,
programming the control panel, and
reading the display

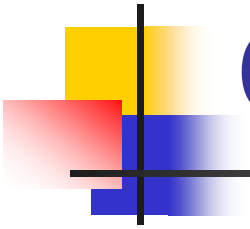
Collaborative Requirements Gathering-Safe Home Case study

mini-spec

The control panel is a wall-mounted unit that is approximately 230 x 130 mm in size. The control panel has wireless connectivity to sensors and a PC. User interaction occurs through a keypad containing 12 keys. A 75 x 75 mm OLED color display provides user feedback. Software provides interactive prompts, echo, and similar functions.

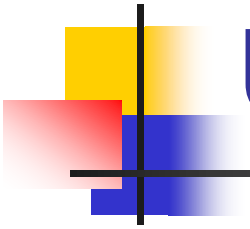
Can we build the system?

- Will this development process allow us to beat our competitors to market?
- Do adequate resources exist to build and maintain the proposed system?
- Will the system performance meet the needs of our customers?



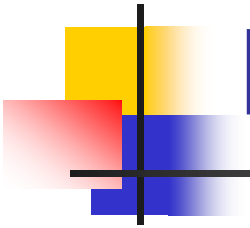
Quality Function Deployment

- It is a quality management technique that translates the needs of the customer into technical requirements for software.
- QFD uses customer interviews and observation, surveys, and examination of historical data (e.g., problem reports) as raw data for the requirements gathering activity.



Usage Scenarios

- The scenarios, often called *use cases provide a* description of how the system will be used.
- Use cases are discussed in greater detail.

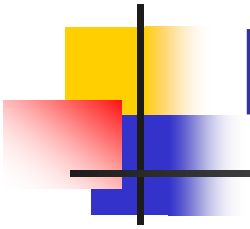


Elicitation Work Products

- Work products include:
- (1) a statement of need and feasibility,
- (2) a bounded statement of scope for the system or product,
- (3) a list of customers, users, and other stakeholders who participated in requirements elicitation,
- (4) a description of the system's technical environment,
- (5) a list of requirements (preferably organized by function) and the domain constraints that applies to each,
- (6) a set of usage scenarios that provide insight into the use of the system or product under different operating conditions, and
- (7) any prototypes developed to better define requirements

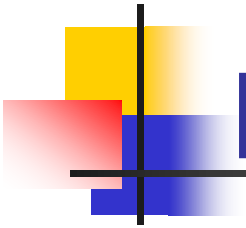
Agile Requirements Elicitation

- Agile process, requirements are elicited by asking all stakeholders to create *user stories*.
- **Service-Oriented Methods**
  - Requirements elicitation in service-oriented development focuses on the definition of services to be rendered by an application.
  - Include ethnographic studies, innovation workshops, and early low-fidelity prototypes.



Developing Use Cases

- A scenario that describes a "thread of usage" for a system
- *Actors* represent roles people or devices play as the system functions
- *Users* can play a number of different roles for a given scenario

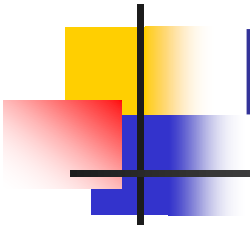


Developing a Use-Case

- What are the main tasks or functions that are performed by the actor?
- What system information will the actor acquire, produce or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

Template for a Use case

Use case:	
Primary actor:	
Goal in context:	
Preconditions:	
Trigger:	
Scenarios	
1……. 2…..	
Exceptions: 1….. 2……	

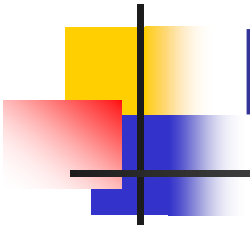


Example: *SafeHome*

- Four actors:
  - **Homeowner**
  - **Setup manager**
  - **Sensors**
  - **Monitoring and response subsystem**

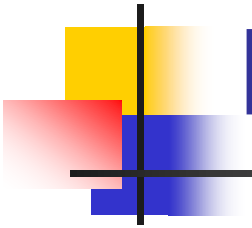
Example: *SafeHome*

Use case:	***Initiate Monitoring***
Primary actor:	**Homeowner**
Goal in context:	To set the system to monitor sensors when the homeowner leaves the house or remains inside.
Preconditions:	System has been programmed for a password and to recognize various sensors.
Trigger:	The homeowner decides to "set" the system, that is, to turn on the alarm functions.



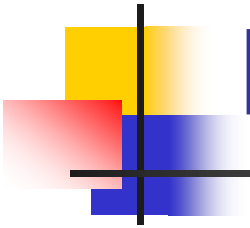
Example: *SafeHome*

Scenario
1. Homeowner: observes control panel 2. Homeowner: enters password 3. Homeowner: selects “stay” or “away” 4. Homeowner: observes read alarm light to indicate that *SafeHome has been armed*
Exceptions: 1. Control panel is *not ready: homeowner checks all sensors to determine which are* open; closes them. 2. Password is incorrect (control panel beeps once): homeowner re enters correct password.



Example: *SafeHome*

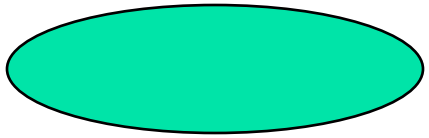
Use case:	**Access camera surveillance via the Internet—display camera views (ACS-DCV)**
Primary actor:	**Homeowner**
Goal in context:	To view output of camera placed throughout the house from any remote location via the Internet.
Preconditions:	System must be fully configured; appropriate user ID and passwords must be obtained.
Trigger:	The homeowner decides to take a look inside the house while away.



Example: *SafeHome*

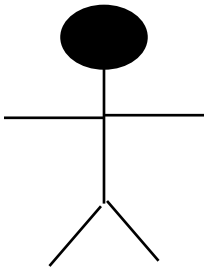
Scenario
1. The homeowner logs onto the *SafeHome Products* website. 2. The homeowner enters his or her user ID. 3. The homeowner enters two passwords (each at least eight characters in length). 4. The system displays all major function buttons. 5. The homeowner selects the "surveillance" from the major function buttons.
Exceptions: 1. ID or passwords are incorrect or not recognized— see use case **Validate ID and passwords.** 2. Surveillance function not confi gured for this system—system displays appropriate error message;

Use Case Symbols



- Use Case

- Actor

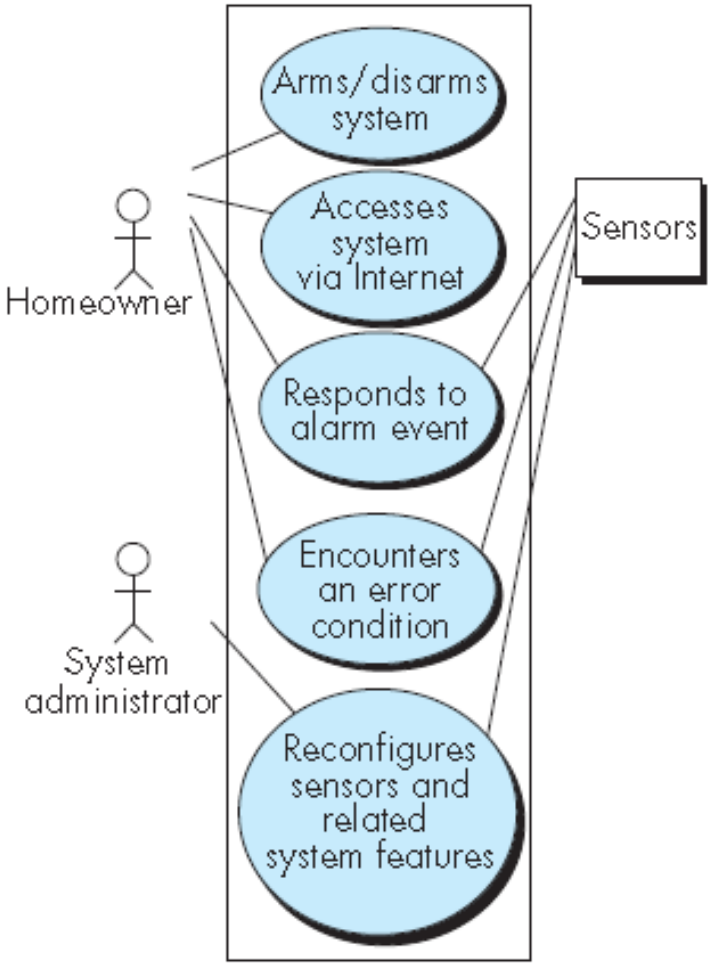


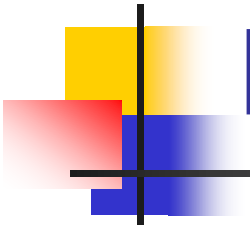
- Boundary

- Connection

Use case diagram

- **UML use case diagram for *SafeHome* home security function**

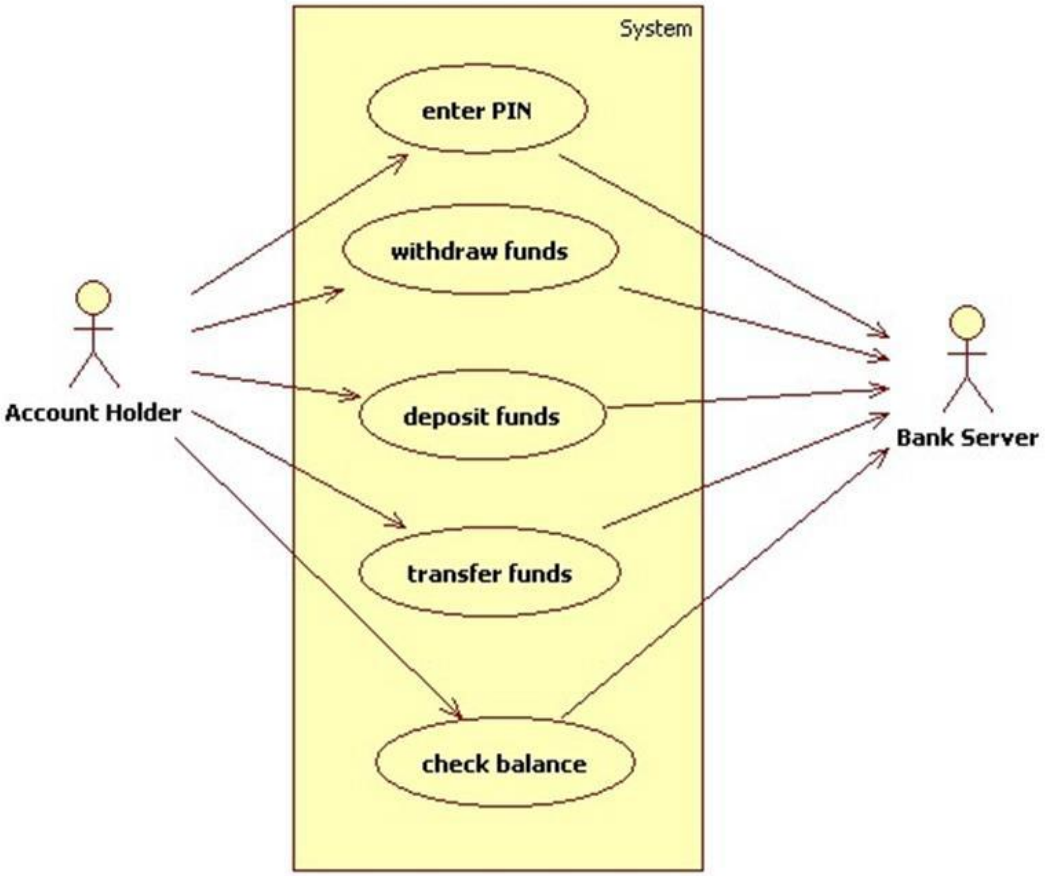




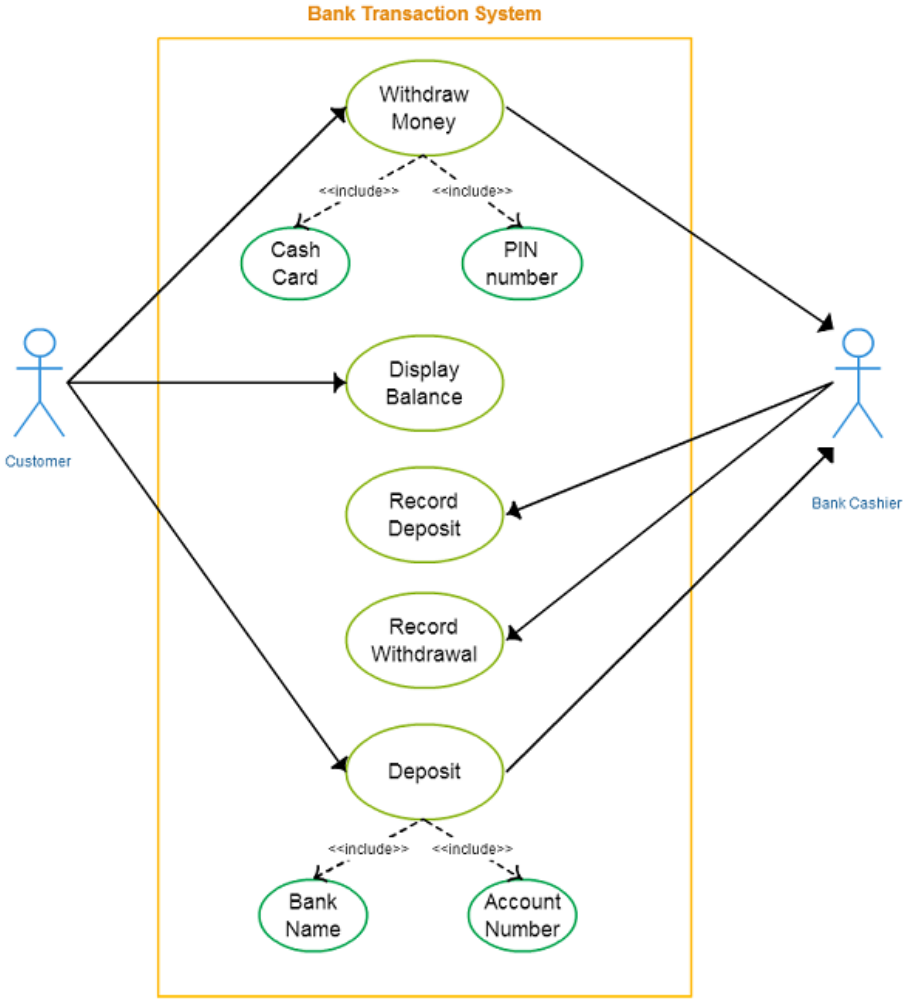
Exercise-Use case

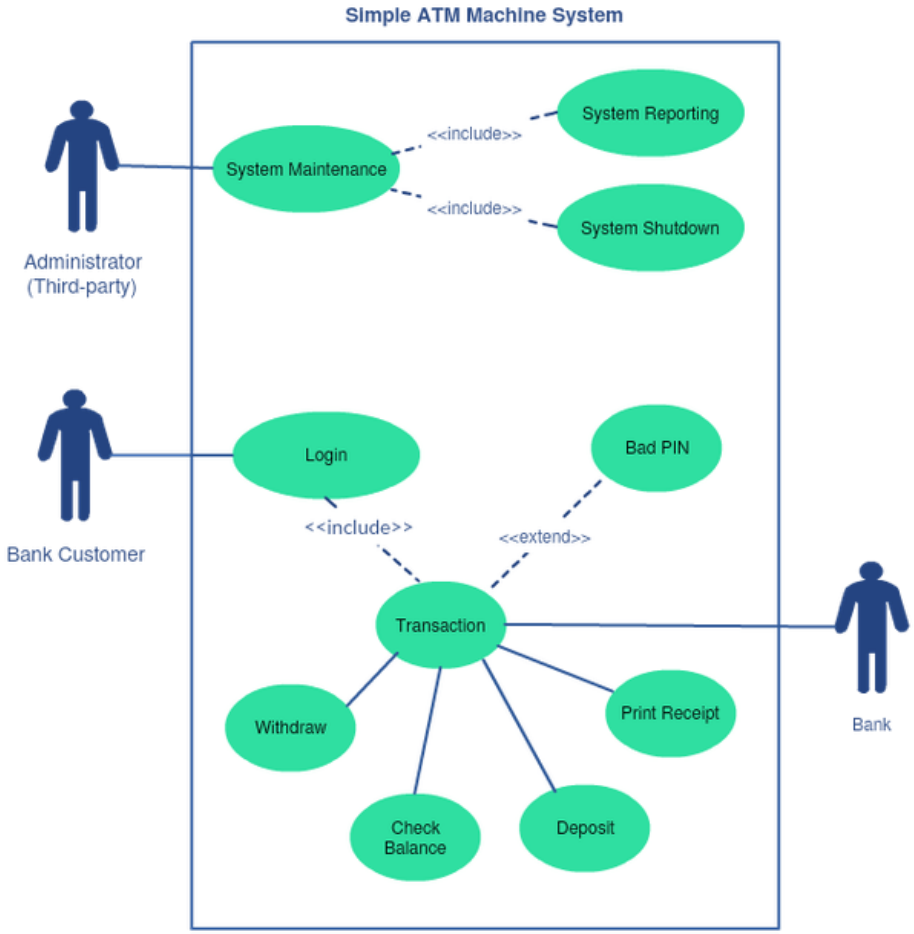
- Create a use case diagram for ATM.
  - Identify the actors.
  - Identify the scenarios.

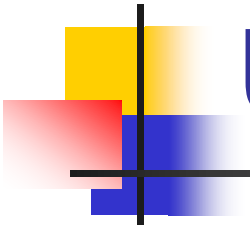
Exercise-Use case



Exercise-Use case

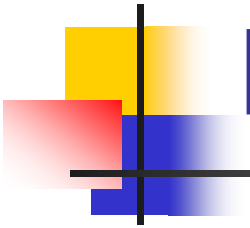






Use case Exercises

- **Write a Use Case for Cash Withdrawal in an ATM.**



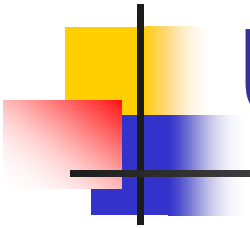
Example: **Cash Withdrawal**

Use case:	**Cash Withdrawal**
Primary actor:	**Customer**
Goal in context:	To withdraw the cash from the ATM using the provided ATM card at a specific location nearest to the customer.
Preconditions:	ATM card is configured and working.
Trigger:	The customer chooses the option "Cash Withdrawal" from the given services menu in ATM.

Example: **Cash Withdrawal**

Scenario
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. The ATM asks the user for a cash amount. The user enters the amount. 5. The ATM asks the user to verify the account. 6. The user verifies the account. 7. If there are sufficient funds in the account, the money is dispensed and the amount is withdrawn from the account.
Exceptions: 1. If the ATM does not have enough cash, an error should be prompted.

Use case Exercises



- **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?**