Requirement Modelling Approaches

Requirements modelling can be done with two approaches structured analysis, object-oriented analysis.
Structured analysis considers data and the processes that transform the data as separate entities. Data objects are modelled in a way that defines their attributes and relationships. Processes that manipulate data objects are modelled in a manner that shows how they transform data as data objects flow through the system.
A second approach to analysis modelled, called object-oriented analysis, focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements.
Although the analysis model that we discuss combines features of both approaches. The question is not which is best, but rather, what combination of representations will provide stakeholders with the best model of software requirements and the most effective bridge to software design.
Each element of the requirements model in the following figure presents the problem from a different point of view.

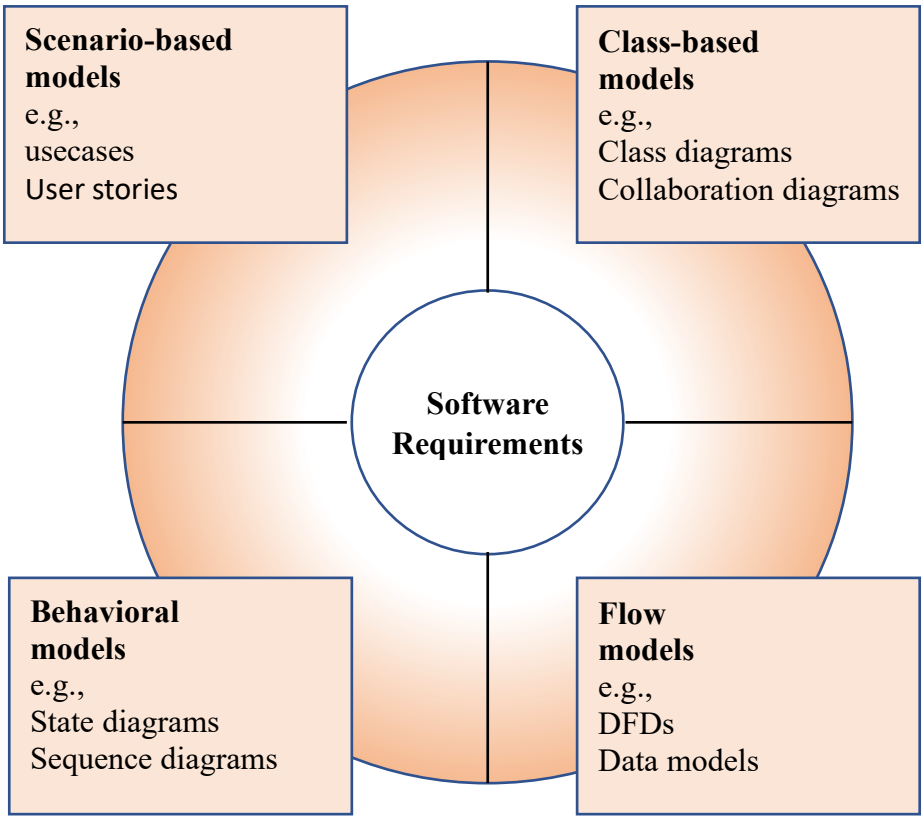

Figure: Elements of Requirement Model

Analysis modelling leads to the derivation of each of these modelling elements. software team must work on those modelling elements that add value to the model.

Scenario-Based Modelling

Requirements modelling with UML (Unified Modelling Language) begins with the creation of scenarios in the form of use cases, activity diagrams.

Creating a Preliminary Use Case
To begin developing a set of use cases, list the functions or activities performed by a specific actor. You can obtain these from a list of required system functions, through conversations with stakeholders, or by an evaluation of activity diagrams.

The ATM system, Customer actor performs the following actions

- Login into the system.
- Withdraw money.
- Deposit money.
- Balance enquiry.
- Take mini transaction statement.

The requirements gathering team develops use cases for each of the functions noted. use cases are written first in an informal narrative fashion. If more formality is required, the same use case is rewritten using a structured format.

Consider Use case: Withdraw amount. customer narrates this usecase as follows
Actor: Customer

Customer inserts their bank card into the card reader on the ATM. The system reads the bank card information from the card then system authenticates customer by validating the four-digit pin number entered. The system displays service options available on the machine, then customer selects withdraw cash option. The system prompts for the amount to be withdrawn by displaying the list of standard withdrawal amounts. The Customer selects an amount to be withdrawn. The system dispenses the requested amount of cash to the Customer. The system records a transaction log entry for the withdrawal. Customer access funds on hand. The system ejects the Customer's bank card. The system records a transaction log entry for the withdrawal.

Structured format results in sequence of action performed by customer

Use case: Withdraw cash.
Actor: homeowner

1. The customer inserts ATM card.
2. The customer selects the language.
3. The customer enters 4-digit ATM pin.
4. The customer selects the type of transaction.
5. The customer selects the type of account.
6. The customer enters the withdrawal amount.
7. The customer collects the cash.
8. The customer takes the printout if needed.
9. The customer goes for another transaction.

Refining a Preliminary Use Case
each step in the primary scenario is evaluated, and look for secondary scenario.
For example, consider steps 3 and 6 in the primary scenario presented:

3.The customer enters 4-digit ATM pin.
6.The customer enters the withdrawal amount.

Can the actor take some other action at this point? The answer is "yes."
Is it possible that the actor will encounter some error condition at 3rd step? "Customer could nor enter into the system". This error condition becomes a secondary scenario.
Is it possible that the actor will encounter some other behaviour at this point? If yes, the customer again inserts the card and then enter valid four-digit pin number.
Each of the situations described above is characterized as a use-case exception. An exception should be noted within the use case or In some cases, an exception will precipitate the development of another use case.

Writing a Formal Use Case
Usecase are written formally, by specifying its goal, prerequisites, trigger condition, scenarios and exceptions.

In many cases, there is no need to create a graphical representation of a usage scenario. However, diagrammatic representation can facilitate understanding, particularly when the scenario is complex.

The following figure depicts a preliminary use-case diagram for the ATM System.

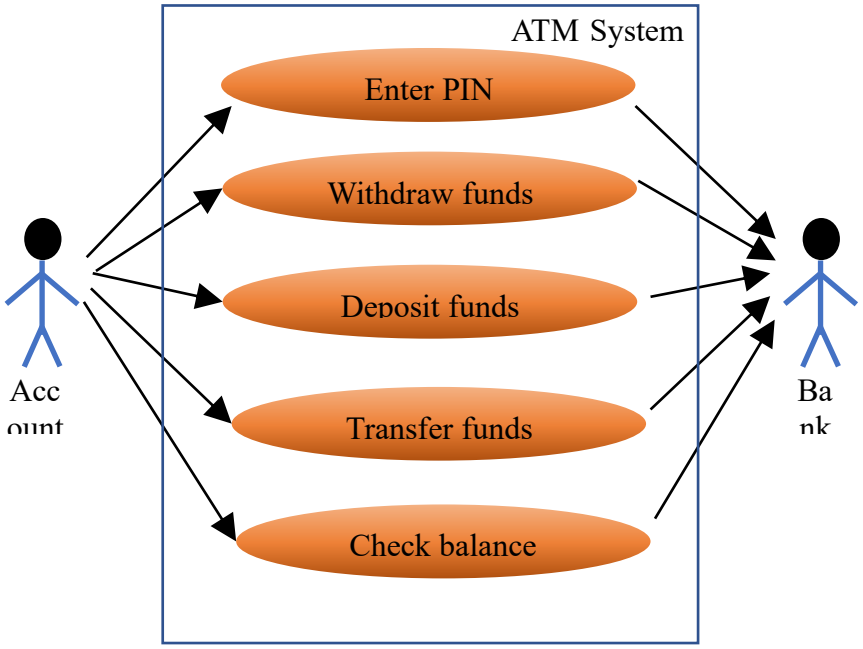

Figure: Preliminary usecase diagram for ATM system

Class-Based Modelling
The elements of a class-based model include classes and objects, attributes, operations, class responsibility-collaborator (CRC) models, collaboration diagrams, and packages.

class diagrams
- Identify the classes by examining use cases developed for the system to be built.
- Classes are identified by looking at the nouns such as external entities, things, occurrence of events, roles, organizational units, places, and structures.
- Develop meaningful set of attributes of the identified classes.

- Define operations that manipulate data, enquiry about state of the object, that perform computations, operations that monitor an object.
- To illustrate, we consider the Customer class defined for ATM system.

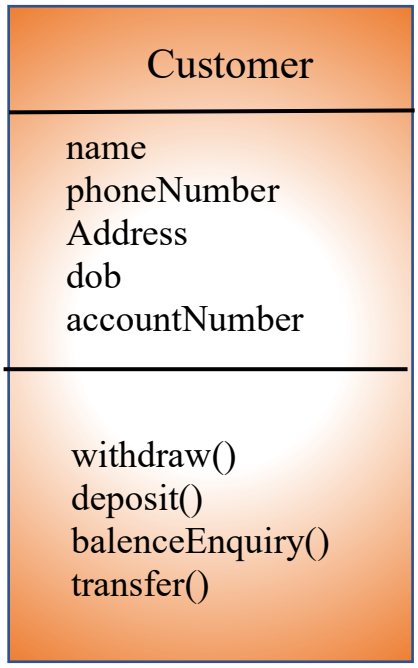

Figure: Class diagram for customer class in ATM System

Class Responsibility Collaborator Model

- CRC models helps us to identify and organize classes of that are relevant to product requirements.
- This model makes use of actual or virtual index cards, that are used to develop organised representation of classes.
- Index cards consist of responsibilities and collaborators.
- Responsibilities are anything that class knows or does something.
- Collaborators are those classes that provide information to fulfil a responsibility.

Class: Withdrawal Transaction	
Description	
Responsibilities	Collaborators
Get specifics from customer	ATM, Session
Send to bank	BANK
Dispense cash, issue receipt, notify bank	ATM

Figure: CRC Model virtual index card

Collaboration Diagram

- It is also called as communication diagram.
- Illustrate relationships and interactions among various classes.
- These are used to model dynamic behaviour of usecases.

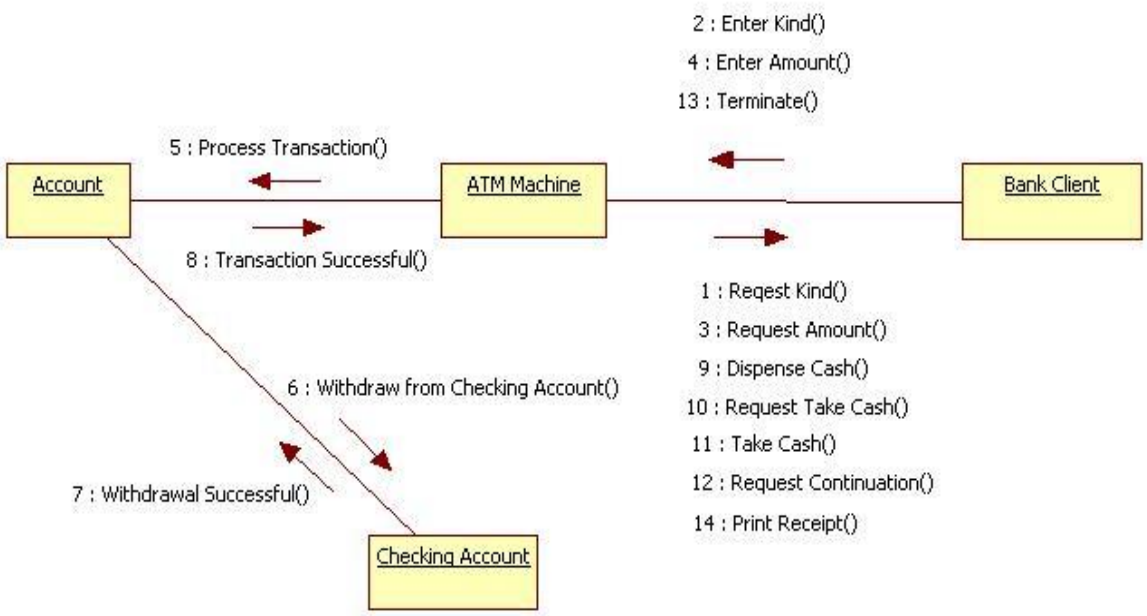

Figure: Collaboration Diagram for ATM System

https://images.app.goo.gl/nYZa8qwgeocXtLn

Text Books

1. Roger Pressman S., “Software Engineering: A Practitioner's Approach”, 7th Edition, McGraw Hill, 2010.
2. Sommerville, “Software Engineering”, Eighth Edition, Pearson Education, 2007.

Web Links

1. https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-355j-software-engineering-concepts-fall-2005/lecture-notes/cnotes2.pdf
2. https://drive.google.com/file/d/1noLGVIm2QpD_vmxMDziGyFVXGdI4BBUu/view
3. https://drive.google.com/file/d/1K327JTpX4P1DPjlNurjKznzZKoBk2Qot/view
4. http://www.cs.fsu.edu/~baker/swe1/restricted/templates/rr631gv1_stuwrk_withdraw_cash_use-case_spec.pdf
5. https://cdn.shopify.com/s/files/1/0457/4009/7694/files/software_engineering_pdf_pressman_7th_edition.pdf